Лекция 10

Рекуррентные нейронные сети и внимание

Машинное обучение

Андрей Фильченков / Сергей Муравьев

06.11.2020

План лекции

- Последовательности и временные ряды
- Рекуррентные нейронные сети
- Модуль памяти
- Больше связей
- Механизм внимания
- Трансформеры
- Векторные представления слов

- В презентации используются материалы курсов:
  - Д. Полыковского и др. «Нейронные сети в машинном обучении»
  - А. Алексеева «Введение в обработку естественного языка»
  - A. Ng “Recurrent neural networks”
- Слайды доступны: **shorturl.at/ltVZ3**
- Видео доступны: **shorturl.at/hjyAX**

План лекции

- Последовательности и временные ряды
- Рекуррентные нейронные сети
- Модуль памяти
- Больше связей
- Механизм внимания
- Трансформеры
- Векторные представления слов

Области применения

- Временные ряды
- Естественные языки
- Речь
- Динамические системы
- Изображения и видео

- В целом, произвольные последовательности

Методы обработки последовательностей

- Спектральные
- Временные
- Частотно-временные

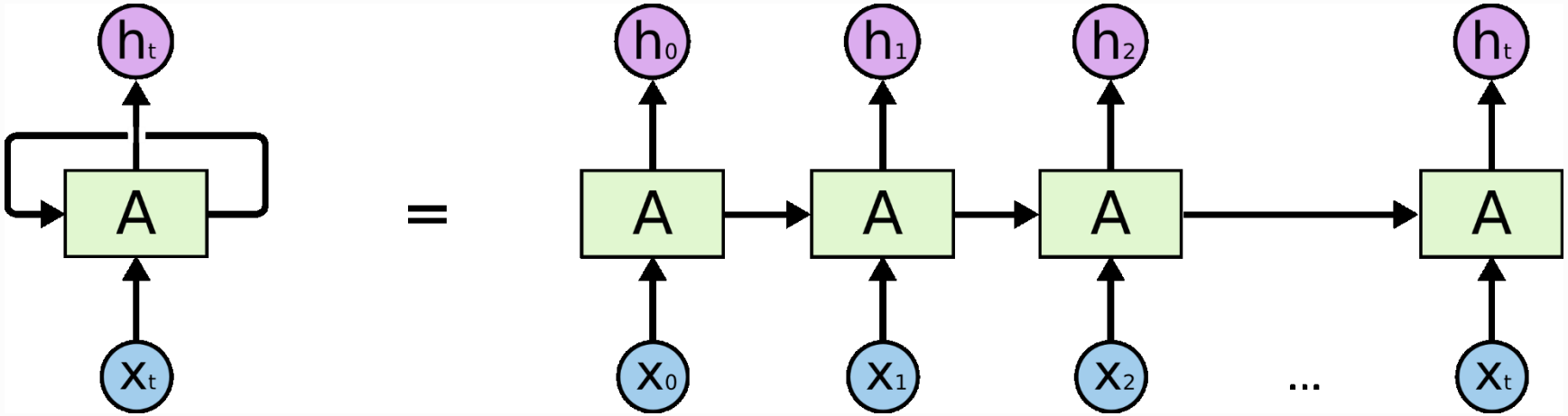
Вероятностные графические модели

Временной ряд можно рассматривать как результат стохастического процесса в предположении о некоторых условных независимостях

- Скрытые марковские модели
- Динамические байесовские сети

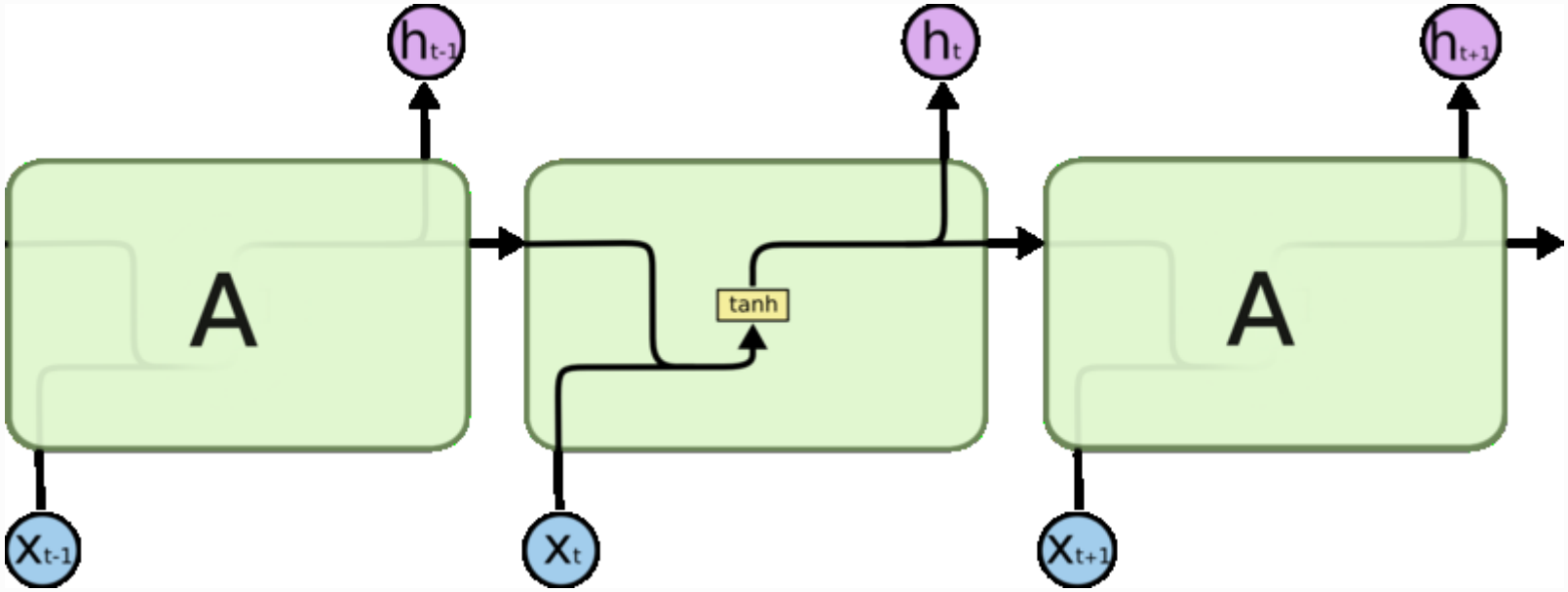
Рекуррентная нейронная сеть

Сеть с петлями или развернутая сеть без петель



Развернутая RNN

В развернутом виде сигнал проходит по идентичным ячейкам



План лекции

- Последовательности и временные ряды
- Рекуррентные нейронные сети
- Модуль памяти
- Больше связей
- Механизм внимания
- Трансформеры
- Векторные представления слов

Сети прямого распространения

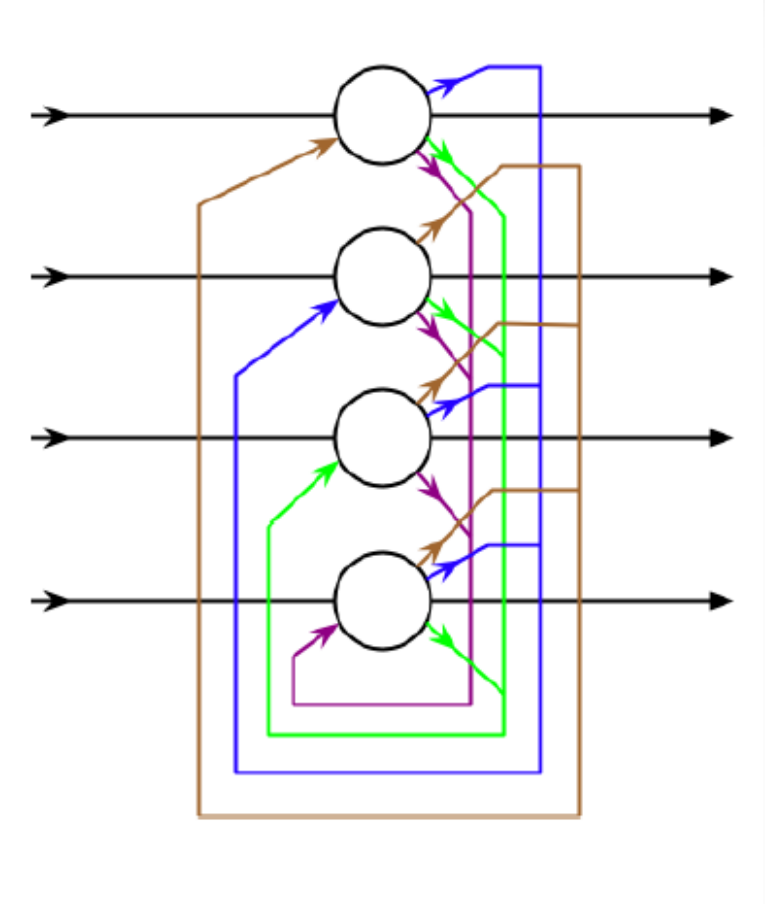
- Несколько теорем о том, что FNN может аппроксимировать произвольную функцию
- FNN можно декомпозировать для того, чтобы последовательно считать частные производные по графу
- Широко распространены

Рекуррентные нейронные сети

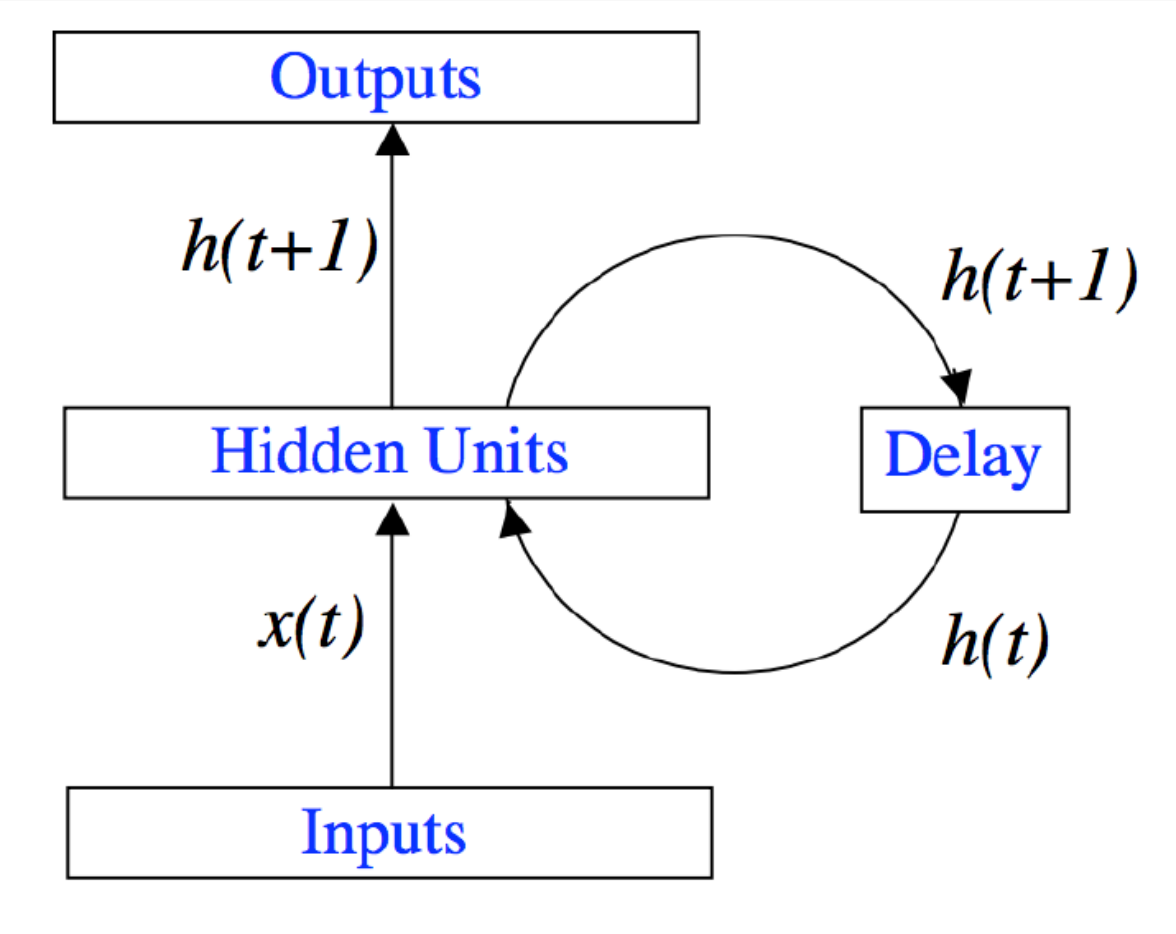
- Биологические нейронные сети рекуррентны
- RNN моделирует динамические системы
- Чуть менее распространены
- Любая машина Тьюринга может быть представлена полносвязной RNN с функцией активации сигмоида

Сеть Хопфилда

- Ассоциативная память
- Сети могут показывать стабильное поведение, осциллировать или показывать хаотическое поведение.

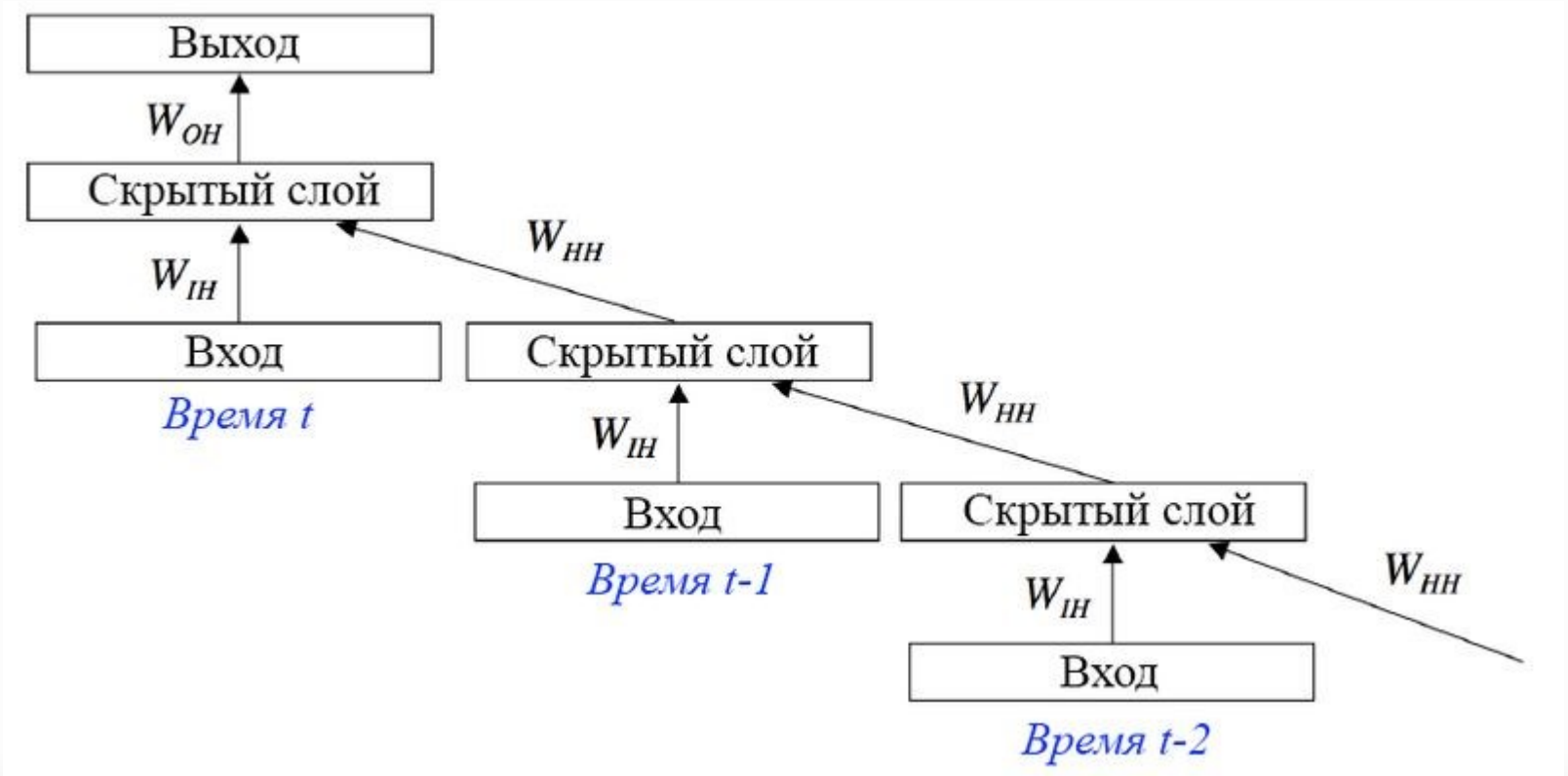


Backpropagation through time



Развернутые RNN

Ограничив максимальную длину, можно провести backpropagation through time



Совместные веса

Проблема в том, что веса должны оставаться одинаковыми

Обратное распространение ошибки можно легко изменить

Для того, чтобы $w_i = w_j$, нужно, чтобы $w_i^{(0)} = w_j^{(0)}$ и $\Delta w_i^{(k)} = \Delta w_j^{(k)}$ $\forall k$. Для этого:

$$\Delta w_i^{(k)} = \Delta w_j^{(k)} := \frac{\delta L}{\delta w_i} + \frac{\delta L}{\delta w_j}$$

Анализ RNN

Преимущества:

- Могут аппроксимировать не функции, но системы
- Является частью «экосистемы» нейронных сетей

Недостатки:

- Требует много времени для обучения
- Исчезающие / взрывающиеся градиенты
- Всегда меняют предыдущий сигнал

План лекции

- Последовательности и временные ряды
- Рекуррентные нейронные сети
- Модуль памяти
- Больше связей
- Механизм внимания
- Трансформеры
- Векторные представления слов

Долгая краткосрочная память

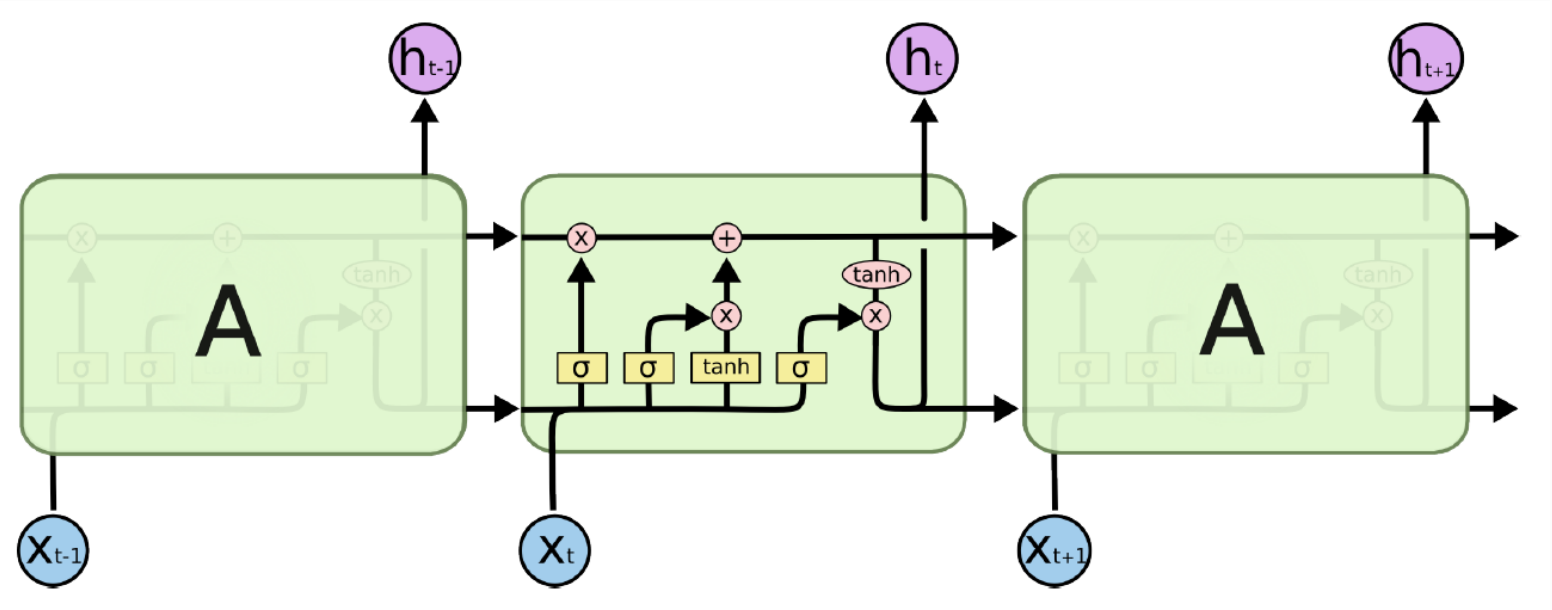
Идея: скрытые состояния не очень хорошо хранят информацию, вместо этого можно выделить отдельный блок для памяти.

Эта идея реализуется в модуле долгой краткосрочной памяти (LSTM)

Модуль LSTM

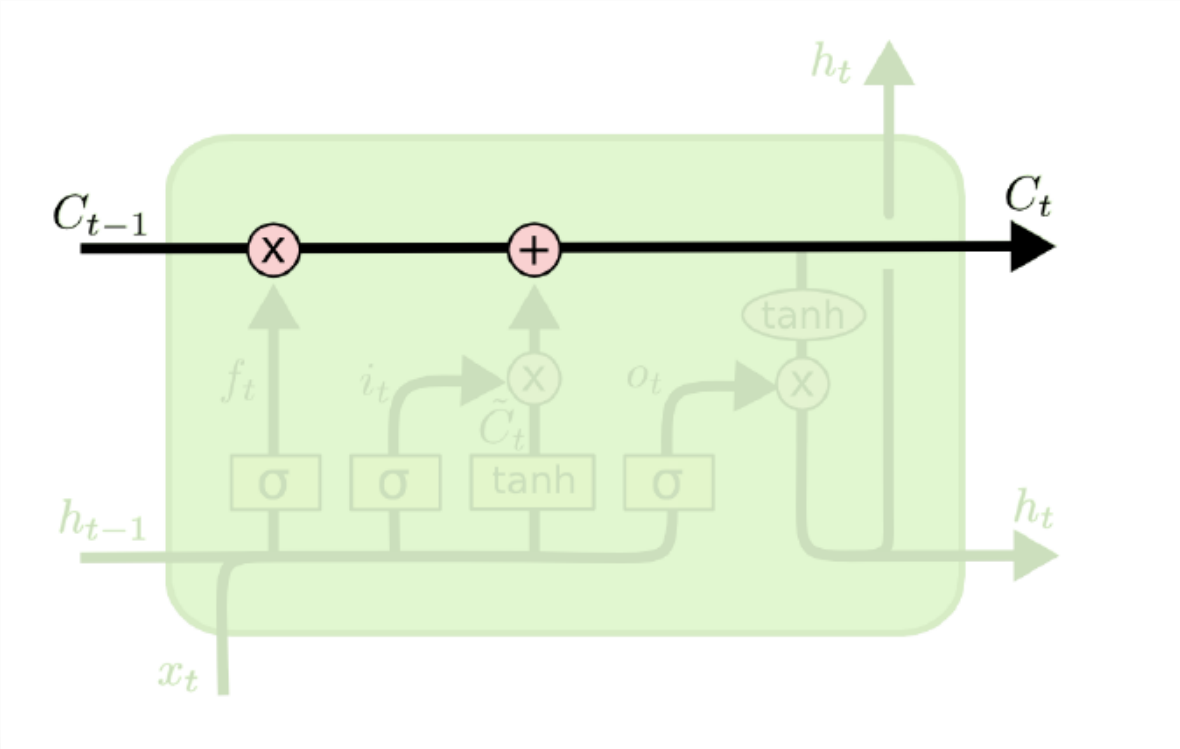
Блок памяти используется в LSTM для хранения глобального состояния

Ячейка параметрическая



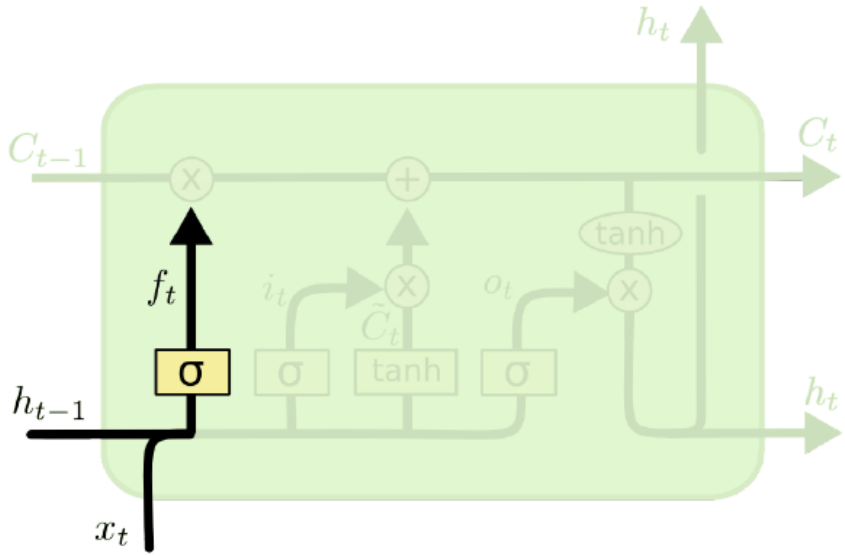
Лента

Лента (conveyor belt) используется для накопления информации



Фильтр забывания

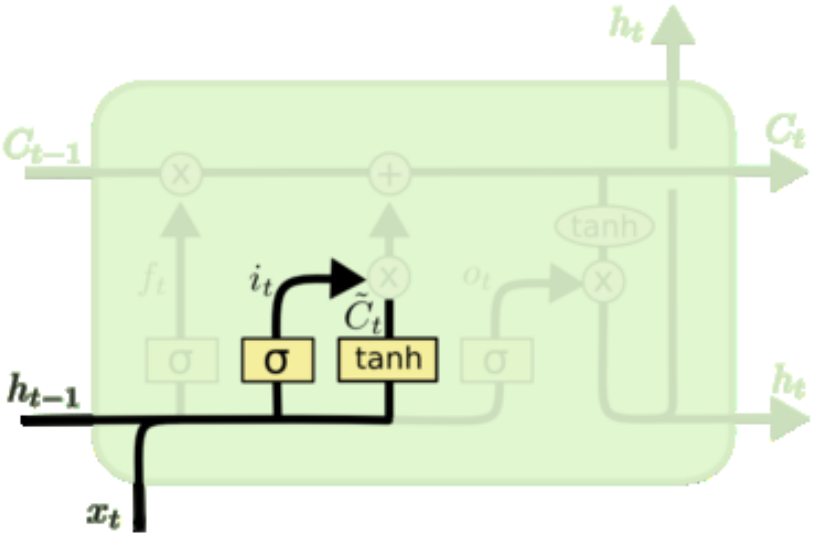
Фильтр забывания (forget layer) определяет, что нужно забыть



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

Фильтр входа

Фильтр входа (input layer) определяет, что нужно записать в память

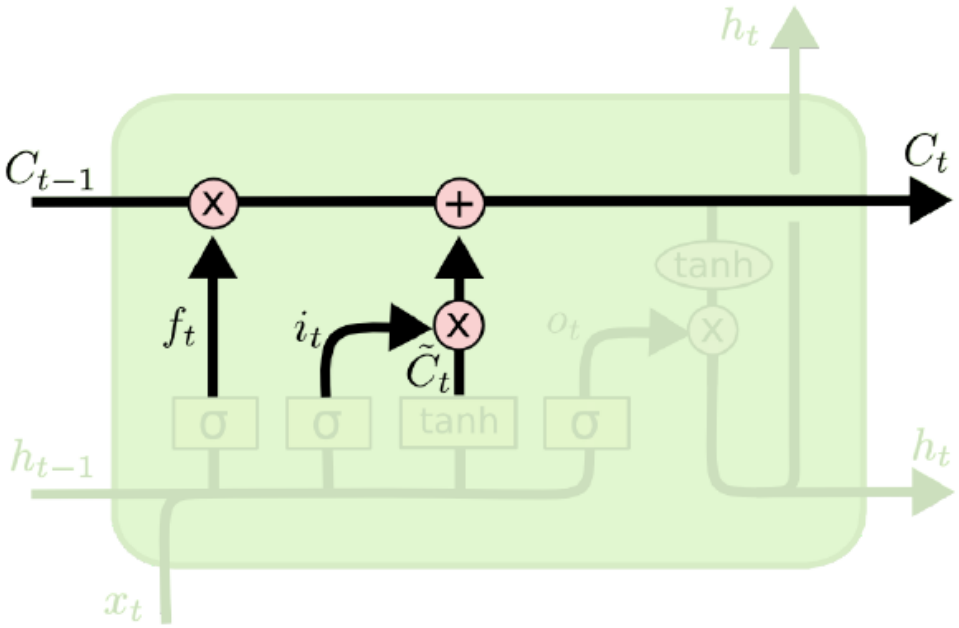


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] + b_i\right)$$

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

Обновление памяти

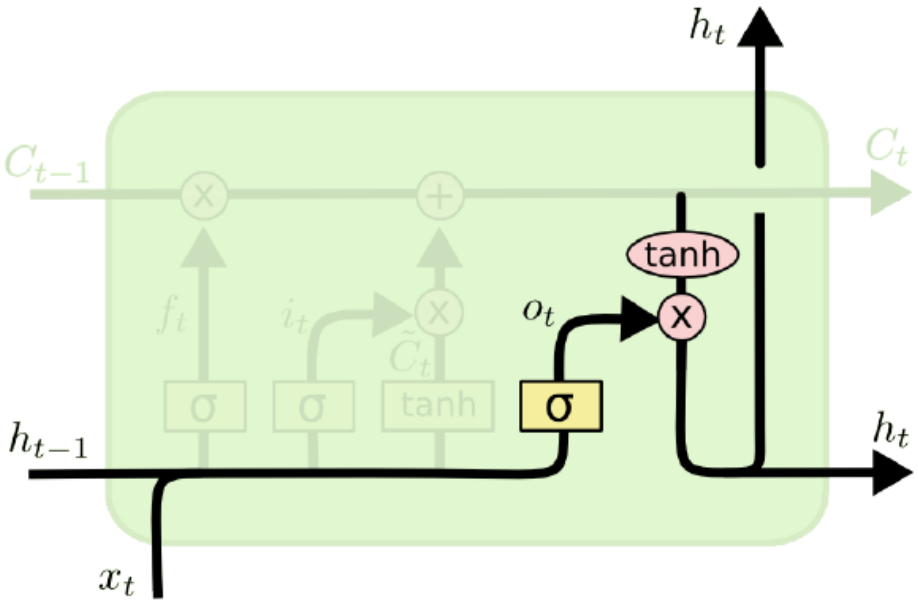
После вычислений, обновляем хранимое в памяти



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

Обновление скрытого состояния

Фильтр скрытого состояния определяет, как его изменить

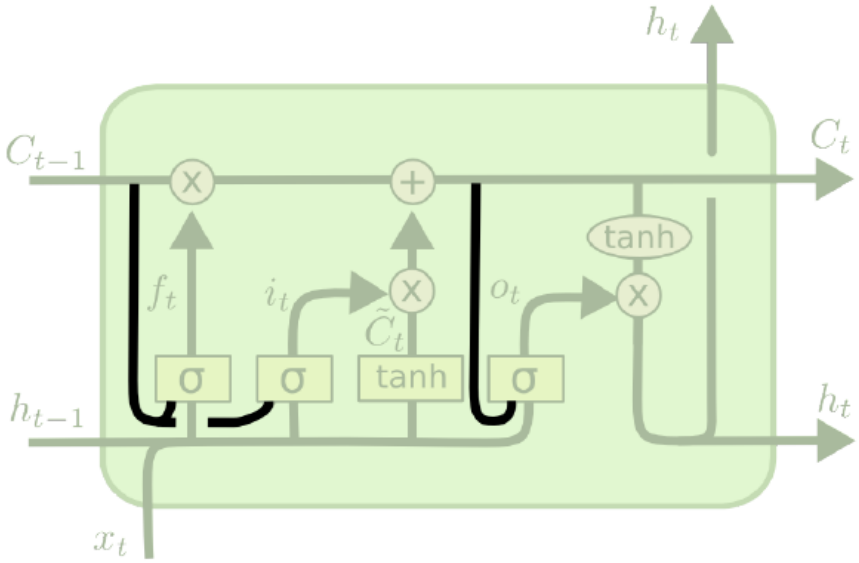


$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$

$$h_t = o_t \times \tanh\left(C_t\right)$$

LSTM со смотровыми глазками

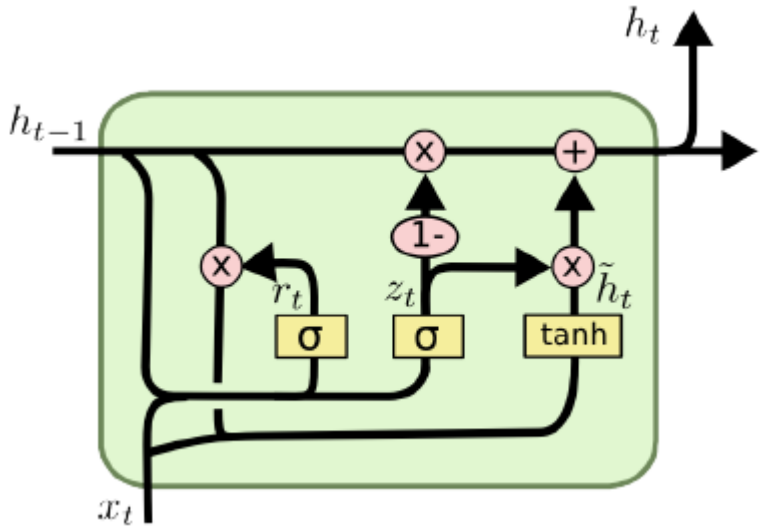
Через смотровые глазки (peephole connections) память используется для обновления себя и скрытого состояния



$$f_t = \sigma\left(W_f \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] + b_f\right)$$
$$i_t = \sigma\left(W_i \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] + b_i\right)$$
$$o_t = \sigma\left(W_o \cdot [\boldsymbol{C_t}, h_{t-1}, x_t] + b_o\right)$$

Gated restricted unit

Можно уменьшить число операций, поменяв операторы и храня все в h



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

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

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

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

Анализ ячеек памяти

Достоинства:

- Могут работать удаленными во времени зависимостями
- Нет затухания / взрыва градиента

Недостатки:

- Требуют много времени для обучения
- Все еще забывают длинные зависимости

План лекции

- Последовательности и временные ряды
- Рекуррентные нейронные сети
- Модуль памяти
- Больше связей
- Механизм внимания
- Трансформеры
- Векторные представления слов

Добавление обратного направления

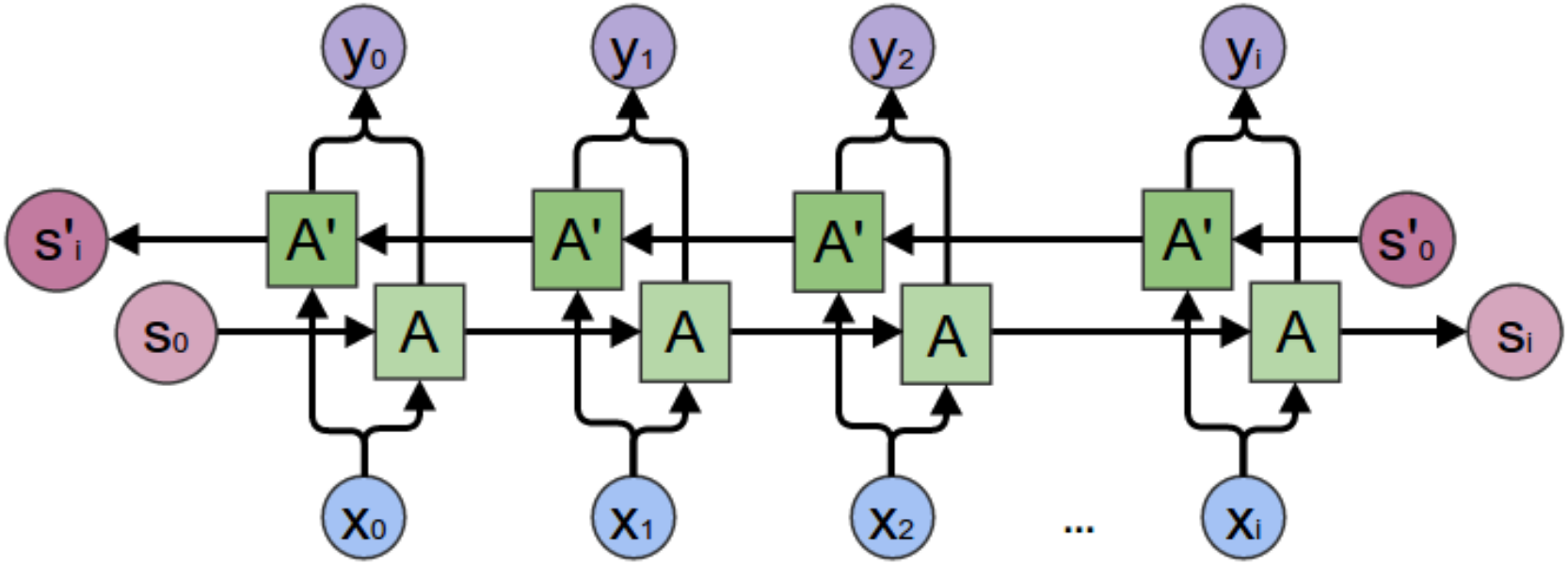
Не только предыдущая информация может быть полезна для обработки текущего сигнала

He said “Teddy bears are on sail!”

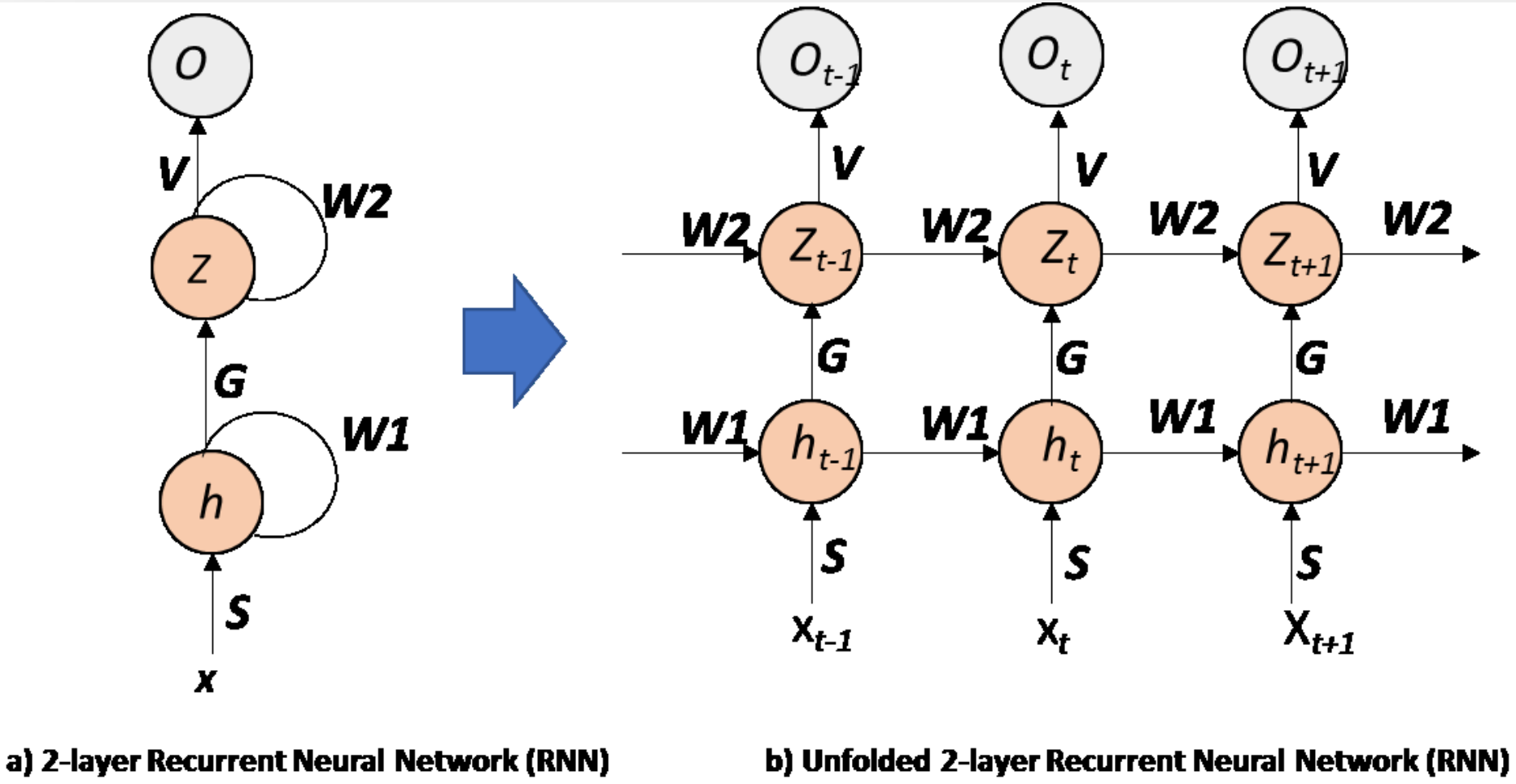
He said “Teddy Roosevelt was a great President!”

Двунаправленная рекуррентная нейронная сеть

Двунаправленная RNN (Bidirectional RNN, biRNN)



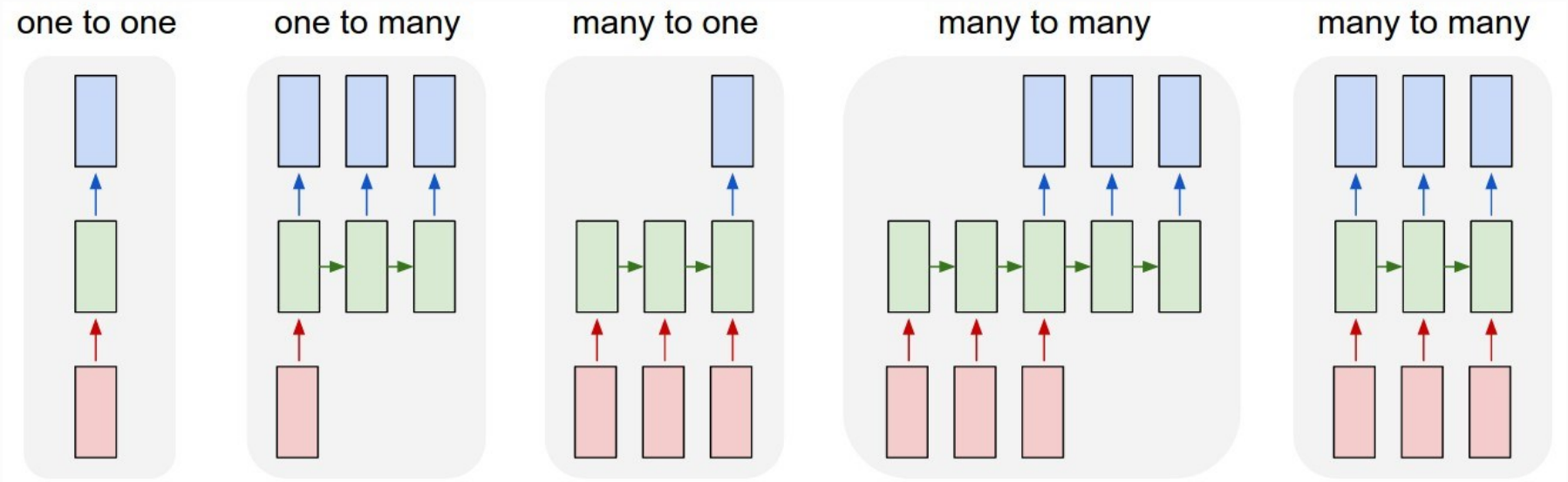
Многослойная рекуррентная нейронная сеть



a) 2-layer Recurrent Neural Network (RNN)

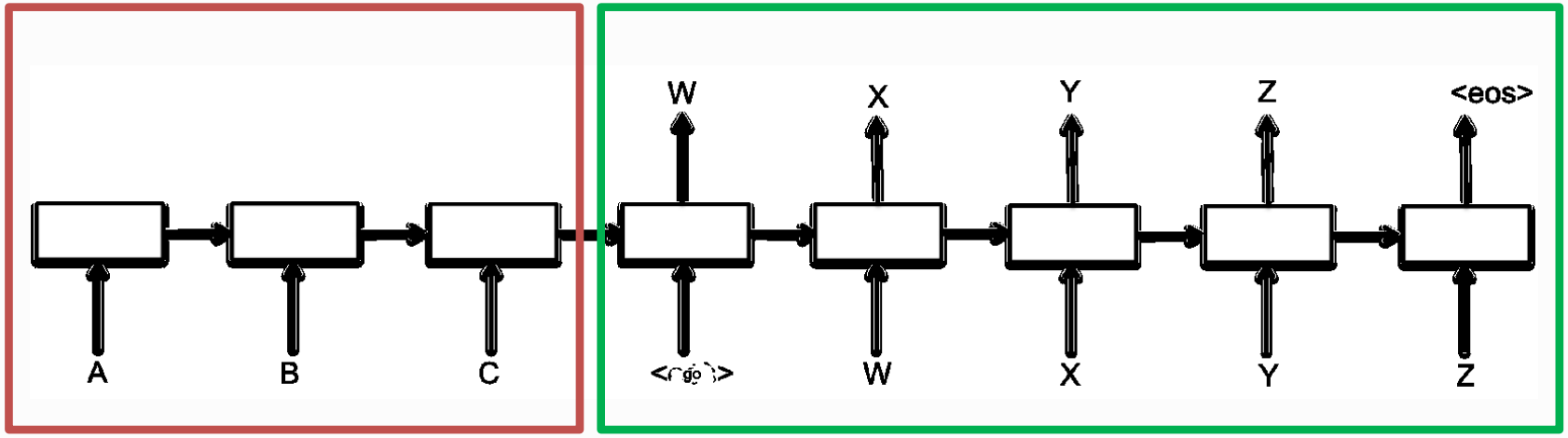
b) Unfolded 2-layer Recurrent Neural Network (RNN)

Классификация рекуррентных нейронных сетей



Seq2Seq

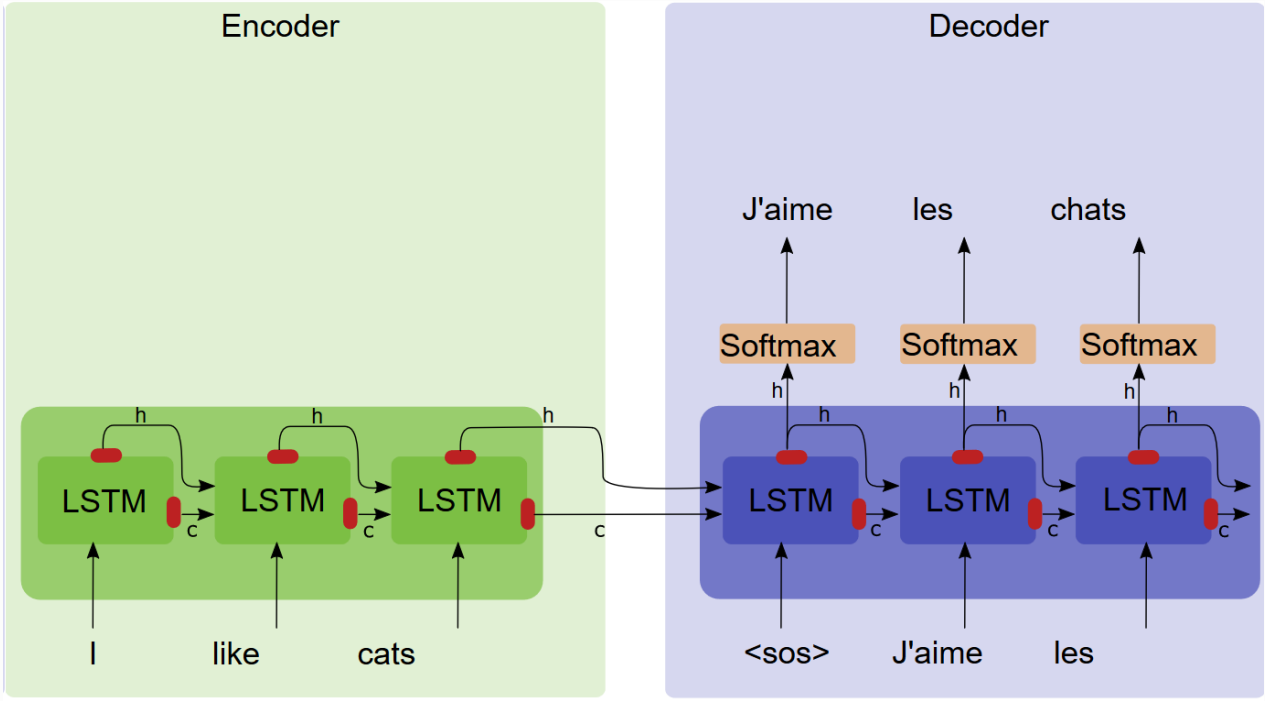
- Состоит из **кодировщика (encoder)** и **декодировщика (decoder)**



- Получает на вход одну последовательность, возвращает другую
- Ячейки могут быть любой сложности

Работа Seq2Seq модели

Между кодировщиком и декодировщиком передается вектор, кодирующий предложение



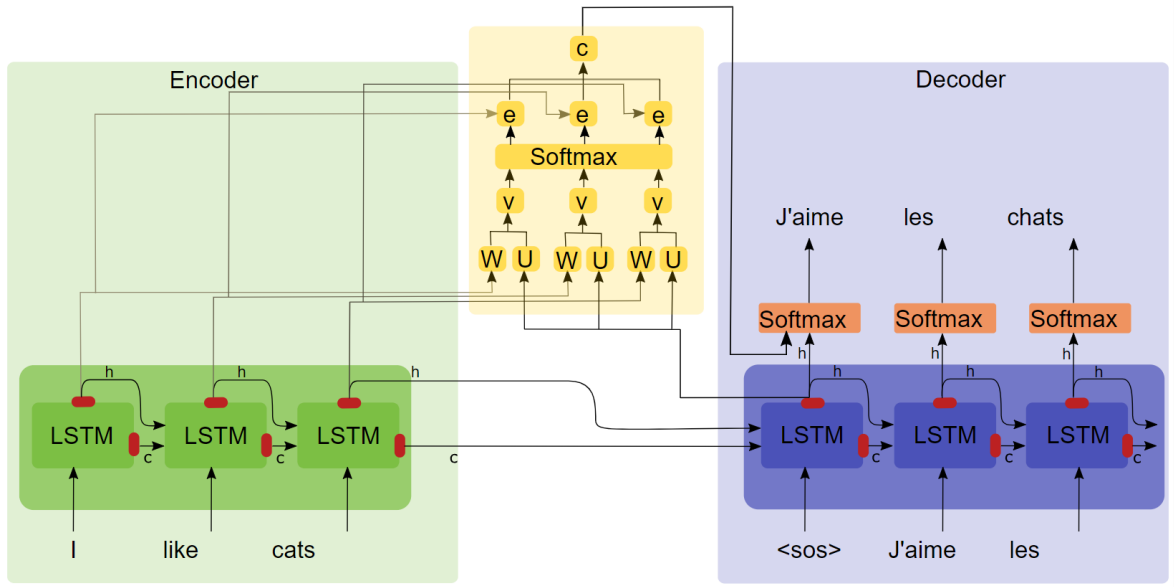
План лекции

- Последовательности и временные ряды
- Рекуррентные нейронные сети
- Модуль памяти
- Больше связей
- Механизм внимания
- Трансформеры
- Векторные представления слов

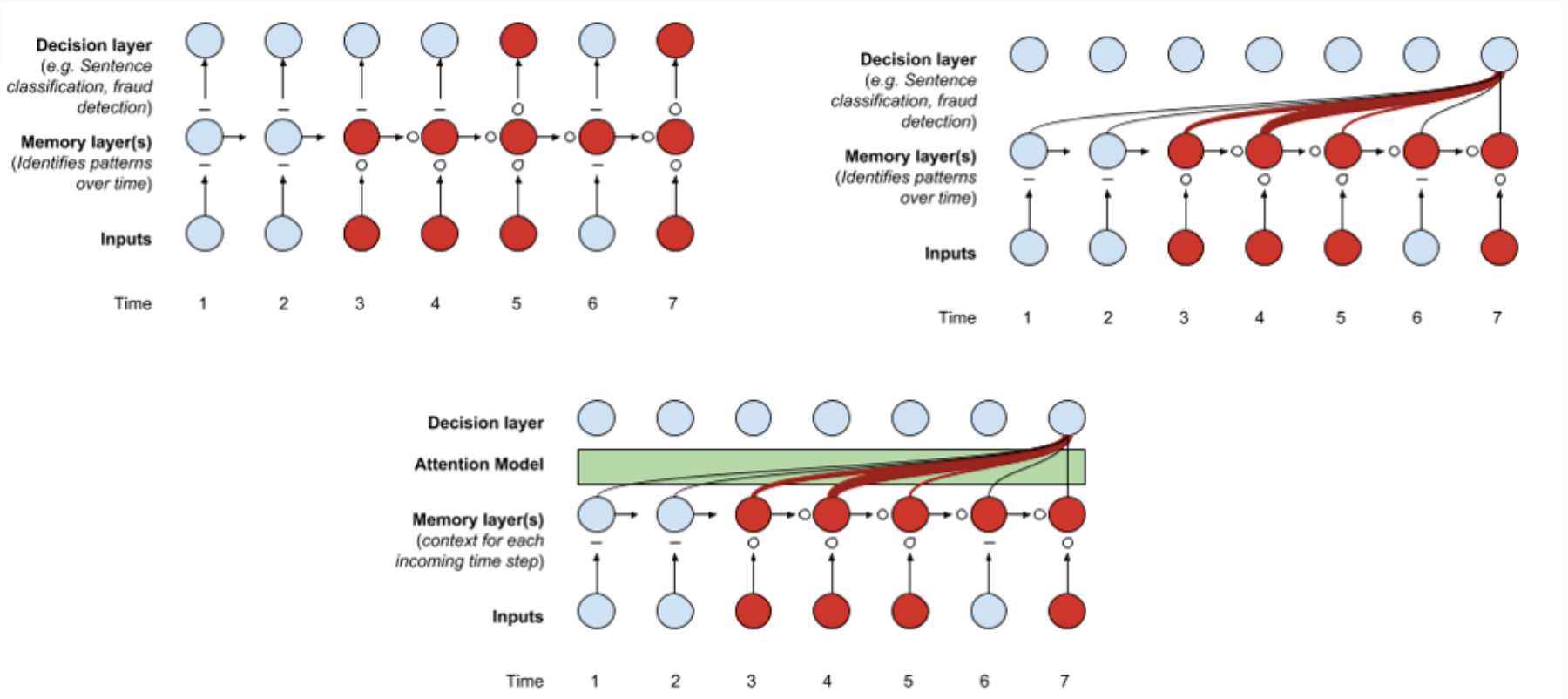
Предпосылка для механизма внимания

Предложение очень сложно (невозможно) закодировать одним вектором.

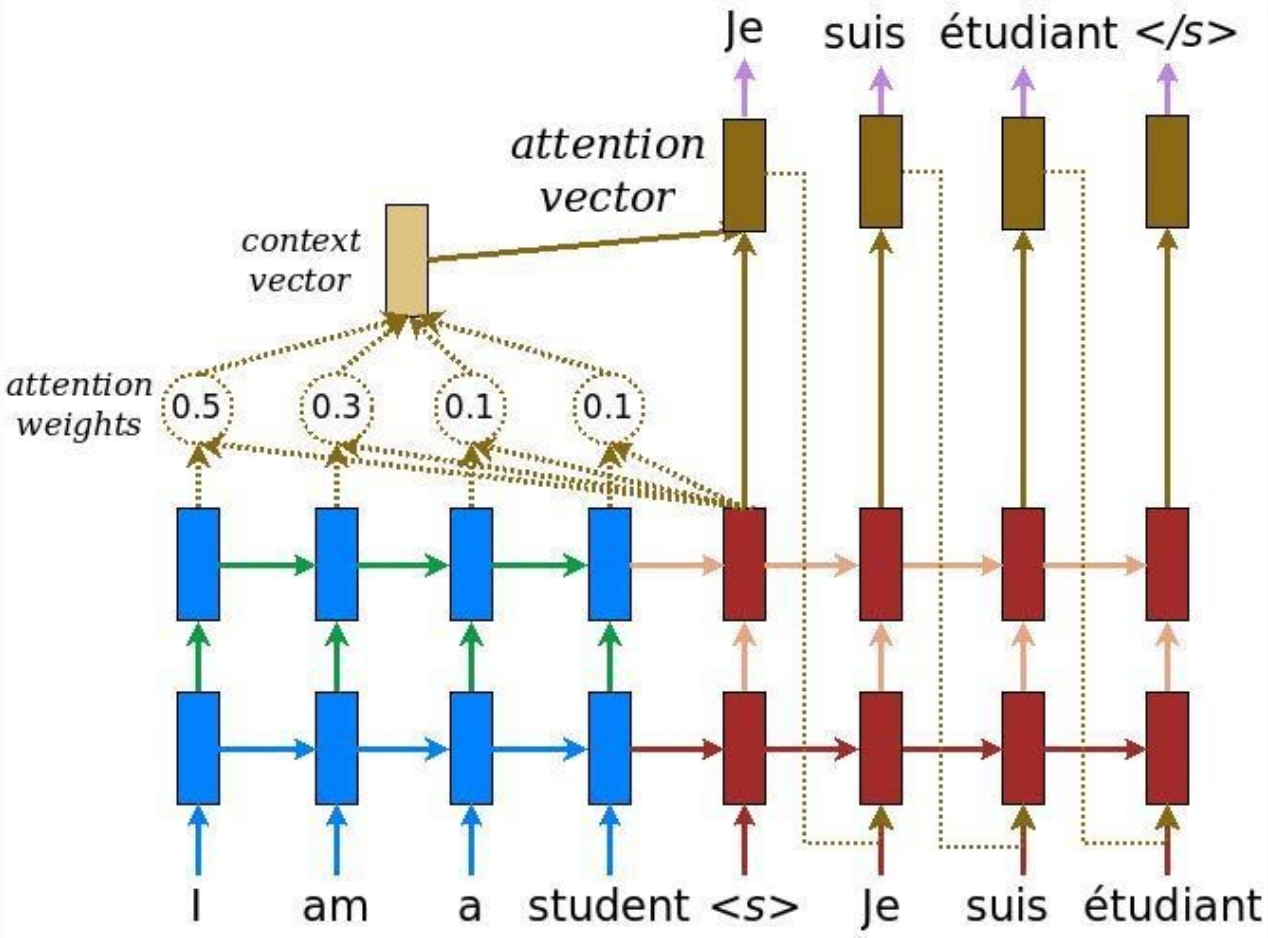
Для того, чтобы его обработать, можно смотреть не на одно скрытое состояние, а на все скрытые состояния.



Использование векторов скрытых состояний



Веса, контекст и внимание



Механизм внимания

Attention weights

$$\alpha_{ts} = \frac{\exp\left(\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s)\right)}{\sum_{s'=1}^{S} \exp\left(\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_{s'})\right)}$$

Context vector

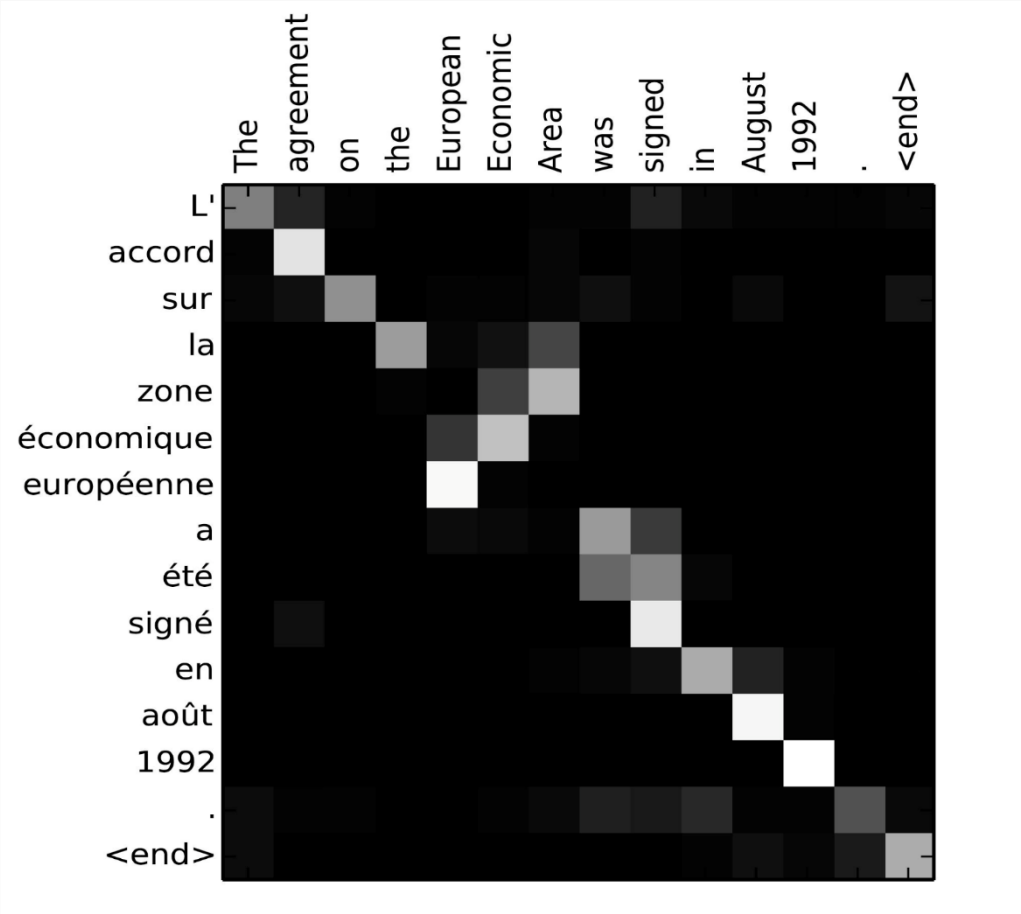
$$\boldsymbol{c}_t = \sum_s \alpha_{ts} \bar{\boldsymbol{h}}_s$$

Attention vector

$$\boldsymbol{a}_t = f(\boldsymbol{c}_t, \boldsymbol{h}_t) = \tanh(\boldsymbol{W}_c[\boldsymbol{c}_t; \boldsymbol{h}_t])$$

Name	Alignment score function
Content-base attention	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \text{cosine}[\boldsymbol{s}_t, \boldsymbol{h}_i]$
Additive(*)	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \mathbf{v}_a^\top \tanh(\mathbf{W}_a[\boldsymbol{s}_t; \boldsymbol{h}_i])$
Location-Base	$\alpha_{t,i} = \text{softmax}(\mathbf{W}_a \boldsymbol{s}_t)$ Note: This simplifies the softmax alignment to only depend on the target position.
General	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \boldsymbol{s}_t^\top \mathbf{W}_a \boldsymbol{h}_i$ where $\mathbf{W}_a$ is a trainable weight matrix in the attention layer.
Dot-Product	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \boldsymbol{s}_t^\top \boldsymbol{h}_i$
Scaled Dot-Product(^)	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \frac{\boldsymbol{s}_t^\top \boldsymbol{h}_i}{\sqrt{n}}$ Note: very similar to the dot-product attention except for a scaling factor; where n is the dimension of the source hidden state.

Похожесть слов



Обсуждение внимания

- Обучается так же, как и другие блоки
- Позволяет обрабатывать более длинные последовательности
- В целом, улучшает производительность
- Может применяться в произвольных сетях

- Добавляет больше параметров

План лекции

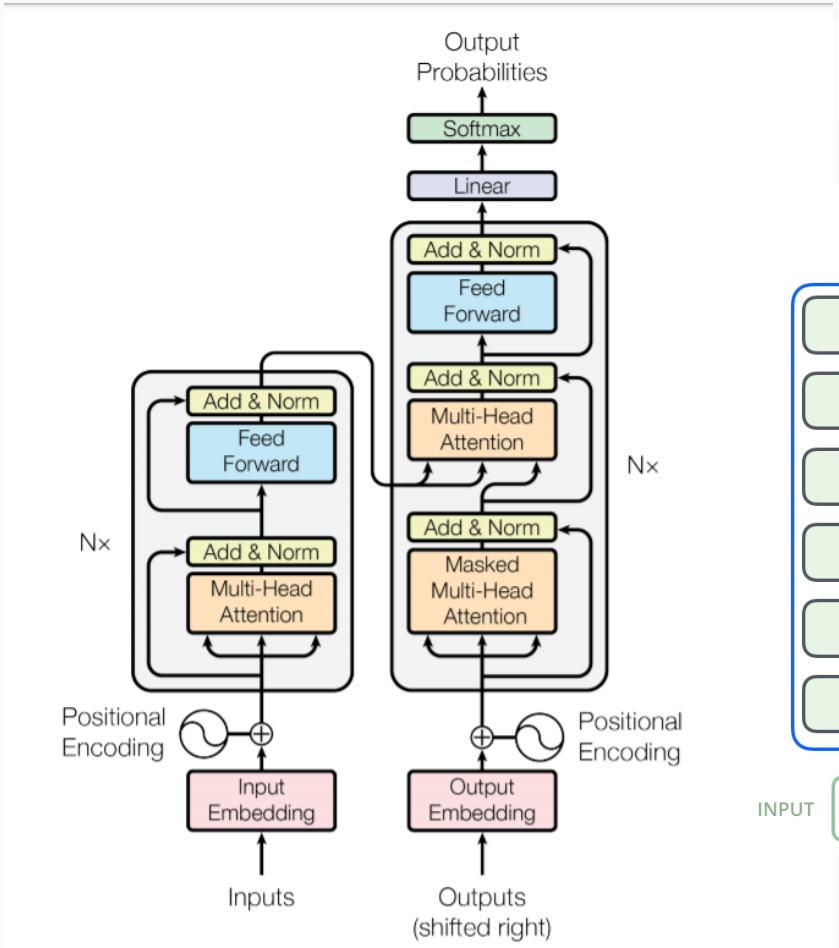
- Последовательности и временные ряды
- Рекуррентные нейронные сети
- Модуль памяти
- Больше связей
- Механизм внимания
- Трансформеры
- Векторные представления слов

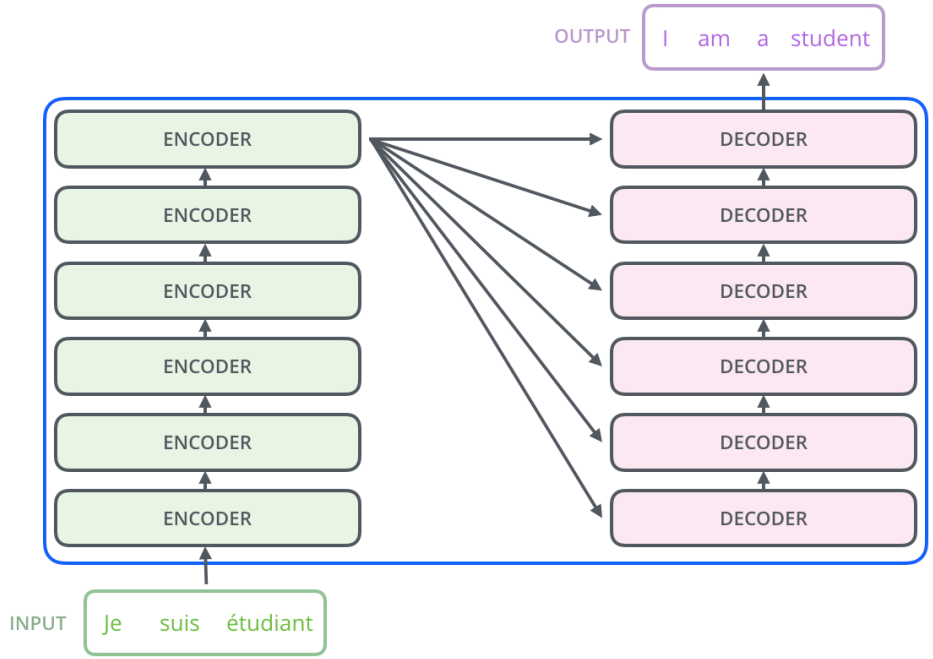
Основная идея

- Внимание отлично находит похожести между векторами
- Попробуем отказаться от скрытых состояний и просто будем искать похожесть внутри входящей последовательности

- Это называется само-внимание (self-attention)

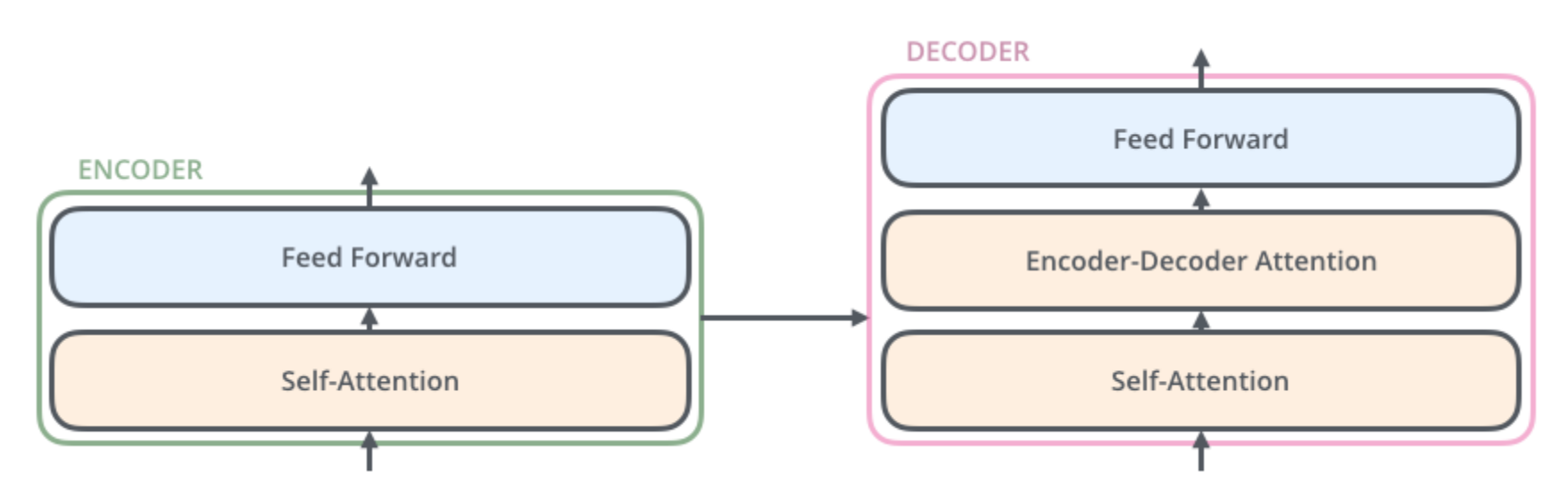
Общая архитектура





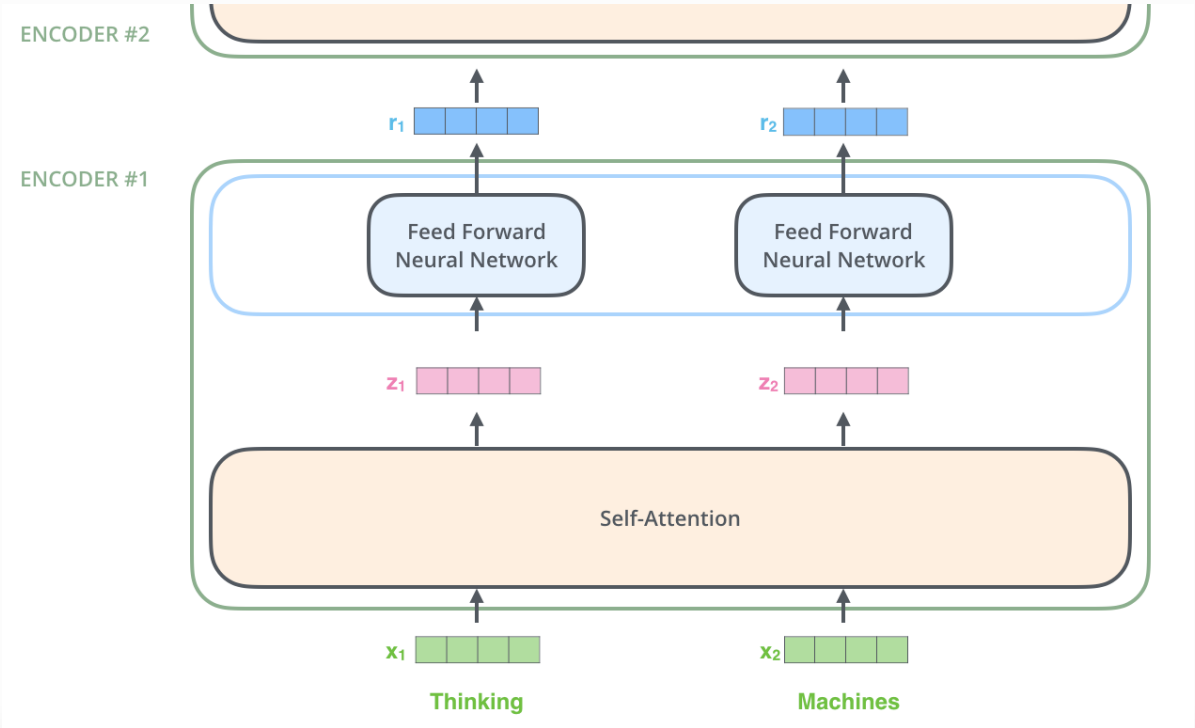
Устройство блоков внутри

- Все блоки одинакового размера, но с разными весами

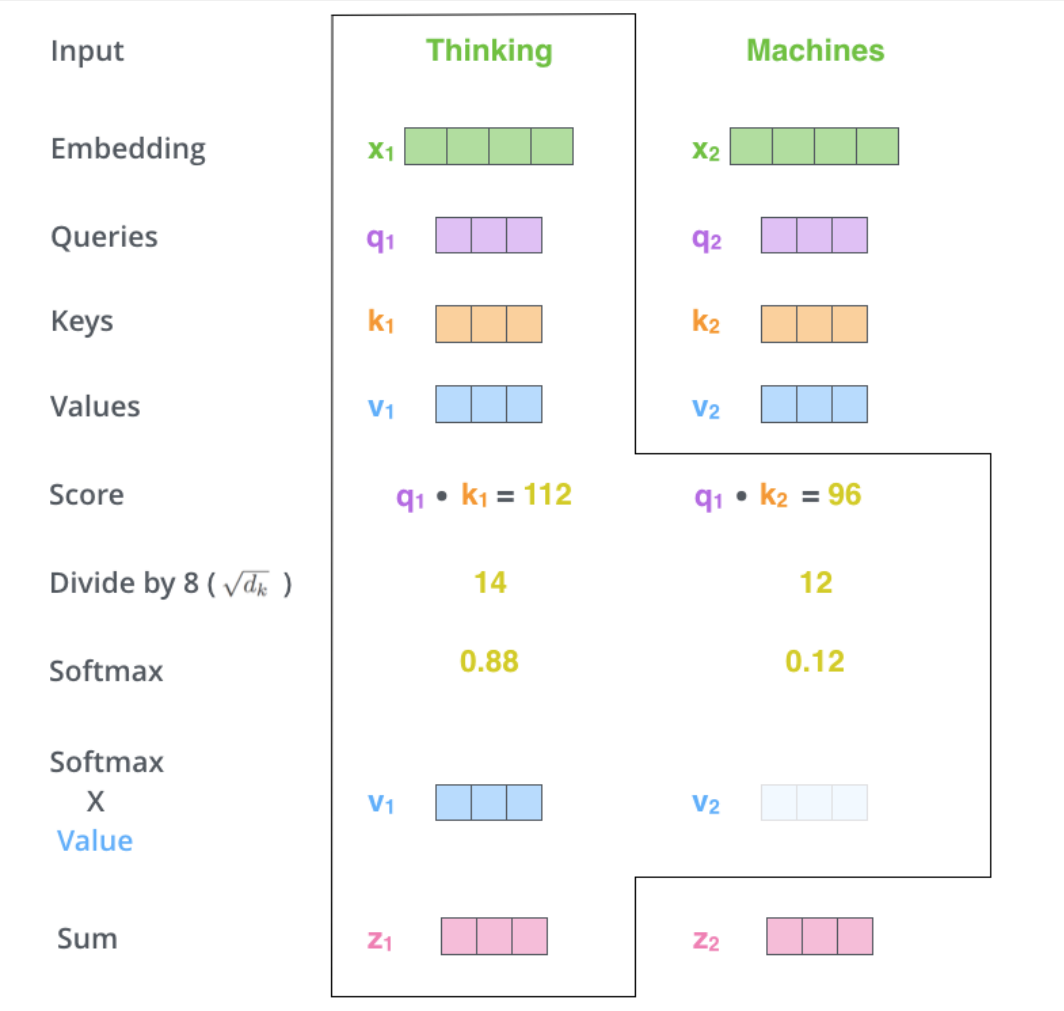


Обработка входа

- Слова обрабатываются по отдельности, но не независимо



Кратко про self-attention



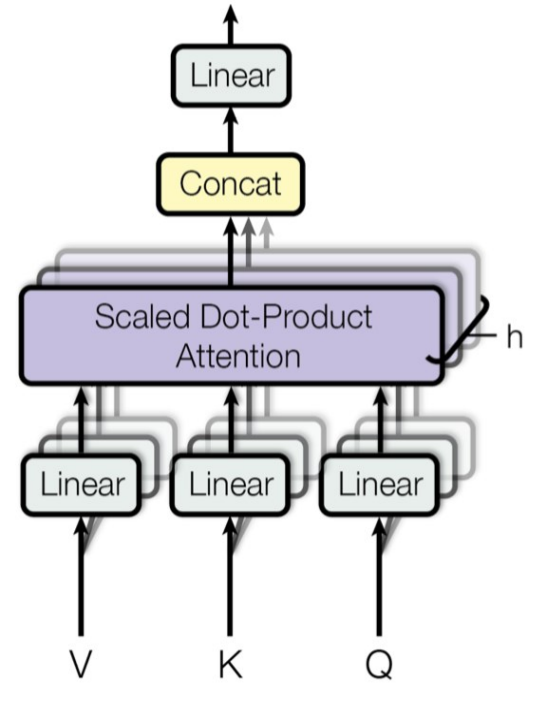
Больше голов

- На какое слово с большей вероятностью будет похоже входящее слово?

Больше голов

- На какое слово с большей вероятностью будет похоже входящее слово?
- На себя.

- Добавим больше голов



Дополнительные трюки

- Positional encoding
- Layer normalization
- Skip-connections
- Masked multi-head attention

Анализ трансформера

Достоинства:

- Распараллеливаемо
- State-of-the-art качество работы
- Потенциально интерпретируемые результаты

Недостатки:

- Очень много параметров
- Нестабильно обучается
- Фиксированная длина предложений

План лекции

- Последовательности и временные ряды
- Рекуррентные нейронные сети
- Модуль памяти
- Больше связей
- Механизм внимания
- Трансформеры
- Векторные представления слов

Вопрос о представлении текста

Модели машинного обучения работают с текстами.

Как получить векторное представление текста?

One-hot encoding

- Зафиксируем размер словаря $|V|$
- Пронумеруем все слова $i(w)$
- Будем представлять каждое слово w в виде вектора
  $\left(0_1, \ldots, 0_{i(w)-1}, 1_{i(w)}, 0_{i(w)+1}, \ldots, 0_{|V|}\right)$

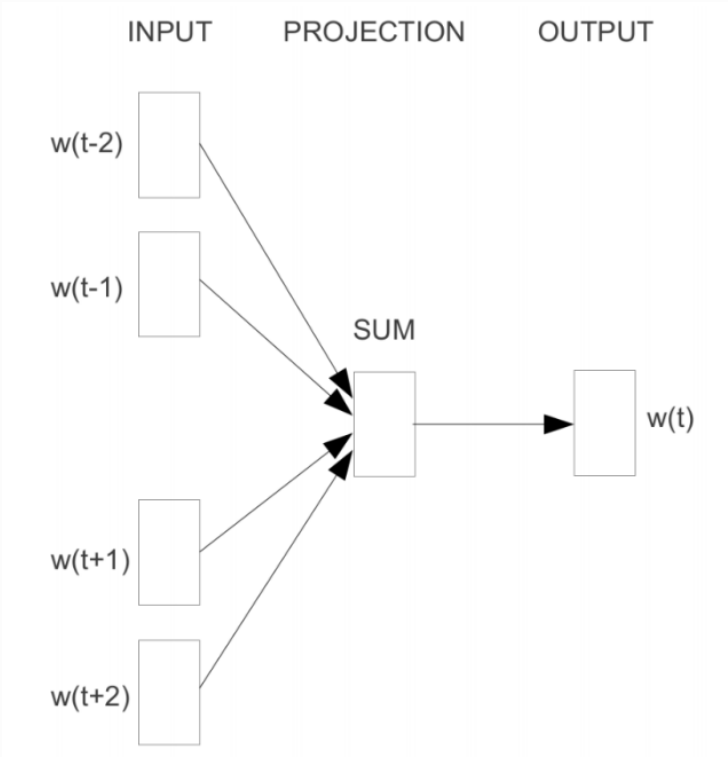
Основная идея Word2Vec

Дистрибутивная гипотеза (Harris, 1954): слова, встречающиеся в похожем контексте, вероятно, будут иметь похожий смысл

Основная идея: будет характеризовать слова контекстом, в котором они встречаются

Continuous bag of words

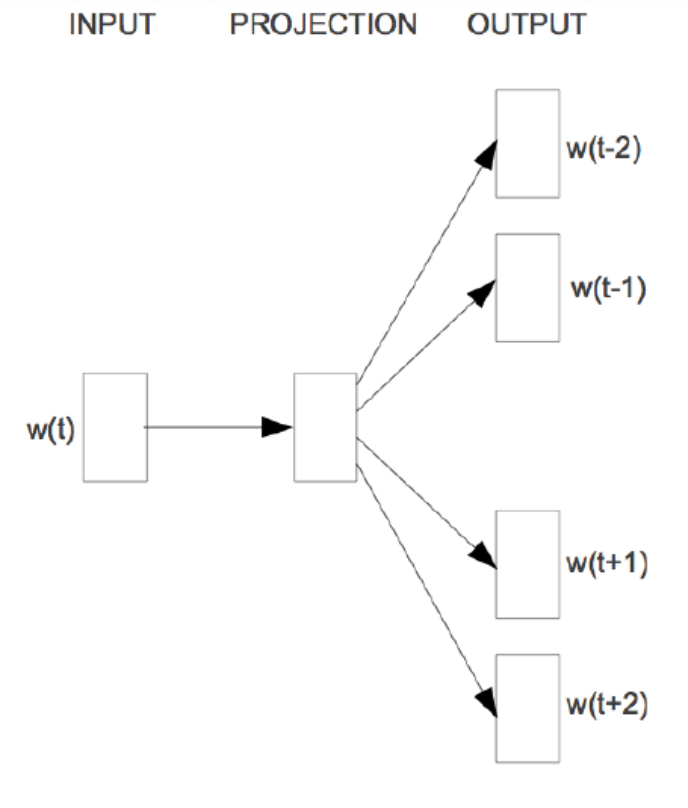
Будем предсказывать слово по контексту



Функция потерь
$-\log \Pr(w_i|\text{context}(w_i))$

Skip-gram

Будем предсказывать контекст по слову

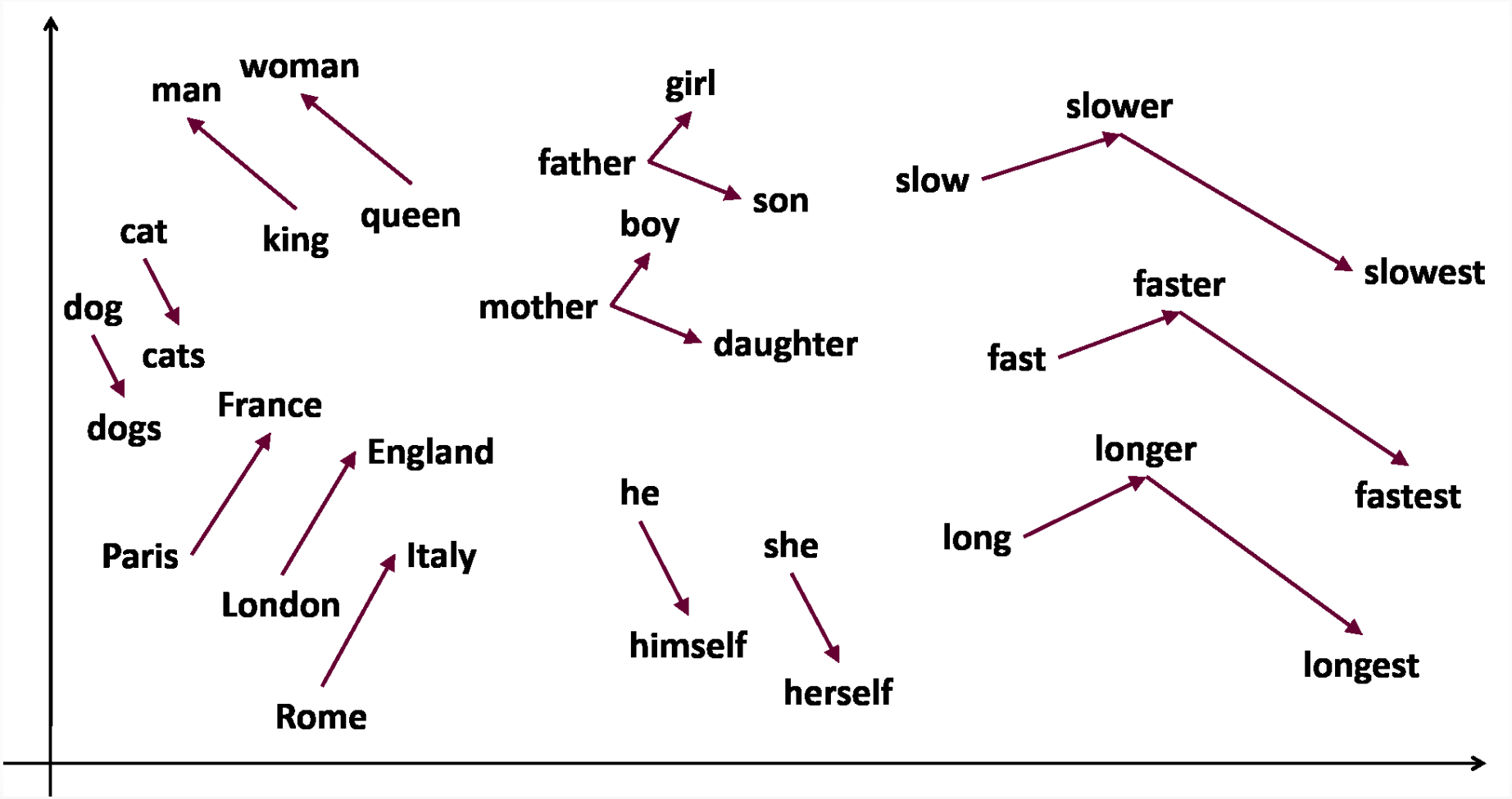


Функция потерь
$-\log \Pr(\text{context}(w_i)|w_i)$

Как обучить?

- Будем показывать пары слов в контексте, $(w_i|\text{context}_j(w_i))$
- Выберем подвыборку, чаще выкидывая часто встречаемые слова
- Негативное сэмплирование (negative sampling) для создания отрицательных примеров

Свойства Word2Vec



Современные эмбеддинги

- **BERT** и порожденные им модели
- ELMO
- FastText
- RusVectōrēs (для русского языка)