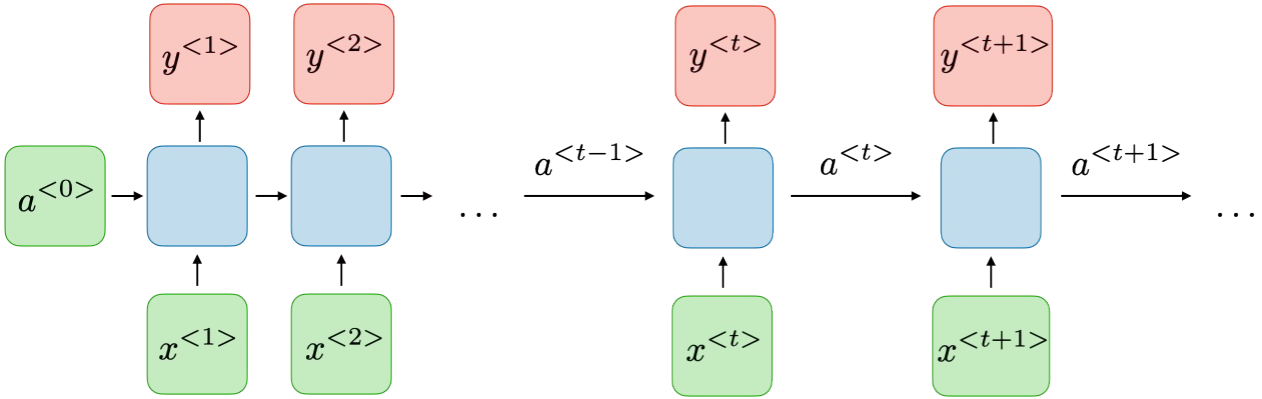
Recap: Рекуррентные нейронные сети

Рекуррентные нейронные сети

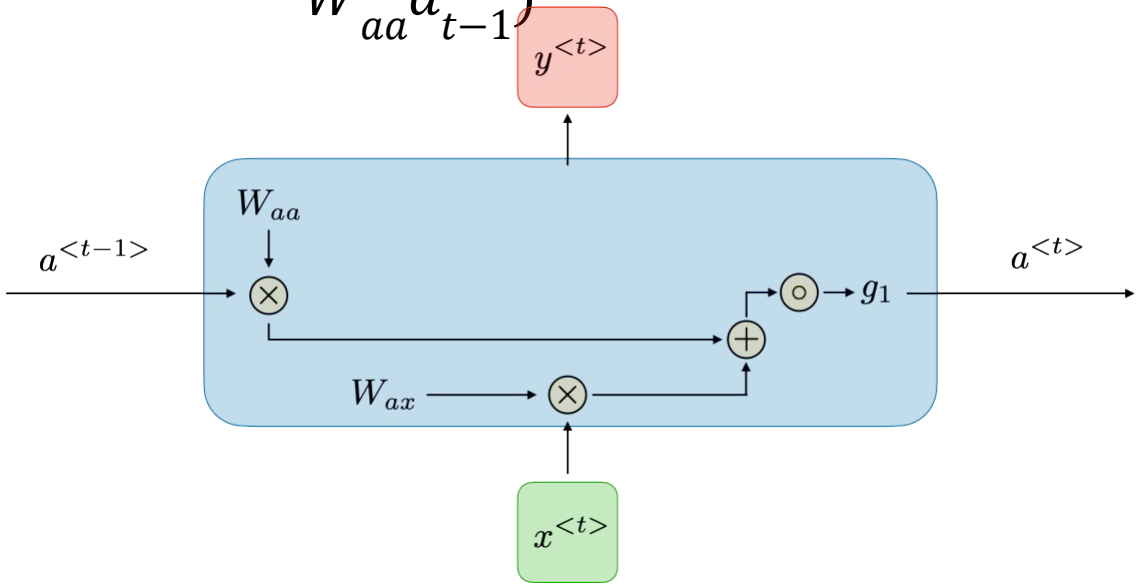


- На вход: слова (текст) $x_1, x_2, \ldots, x_n, \ldots$
- Читаем слева направо
- a_t (вектор) - накопленная информация после прочтения t элементов

Обновление состояния ячейки

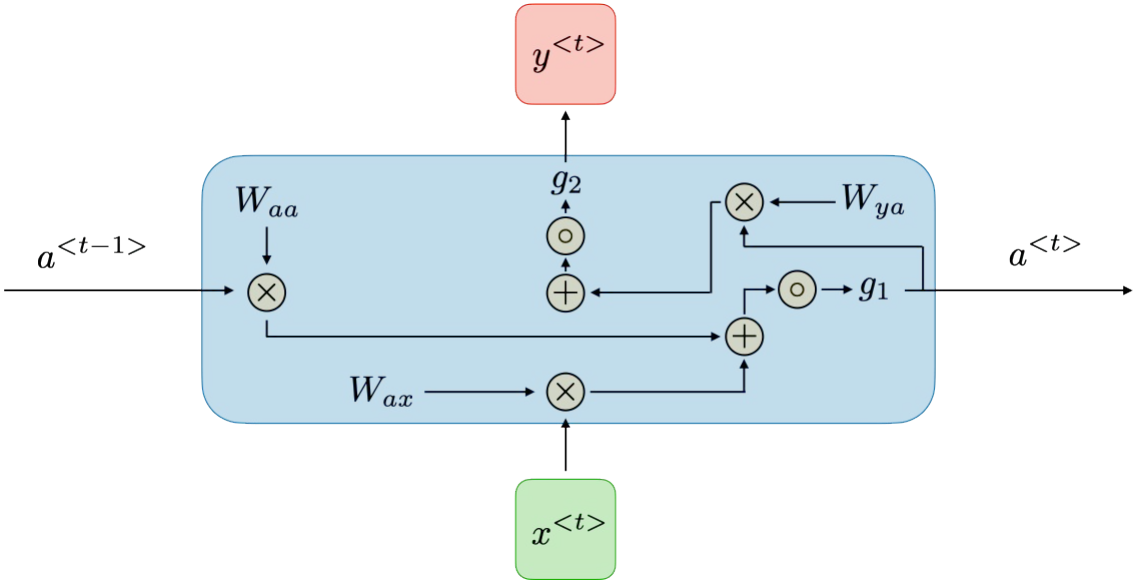
$$a_t = g_1(W_{ax}x_t + W_{aa}a_{t-1})$$

Обычно $g_1 = tanh$

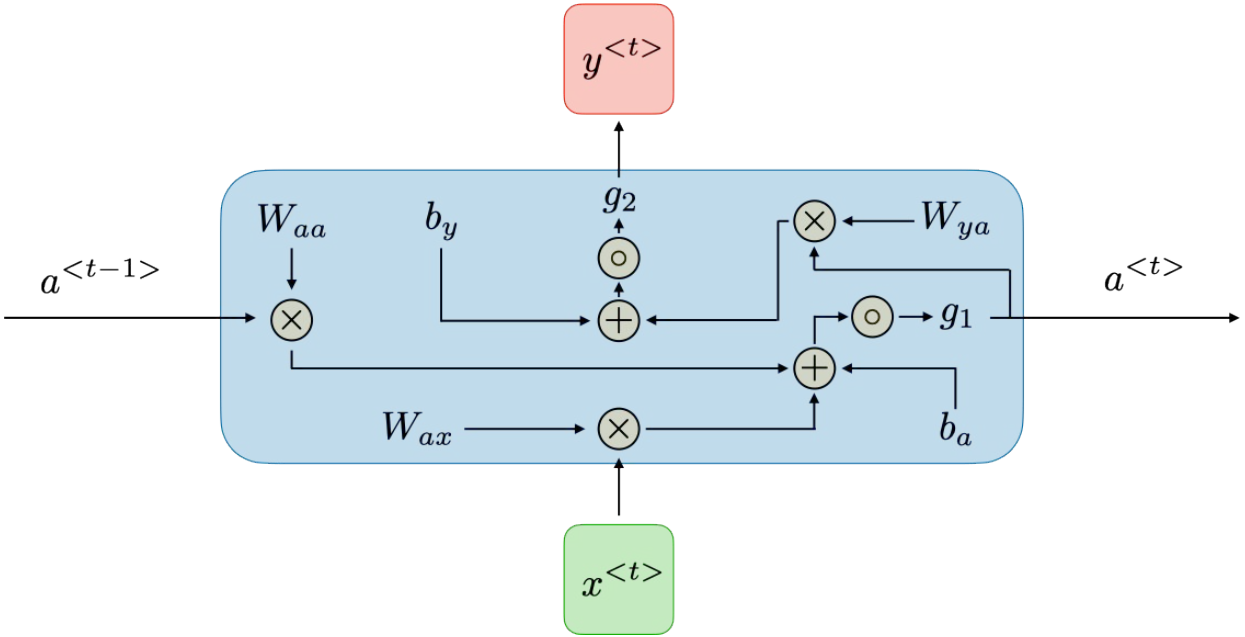


Получение прогноза

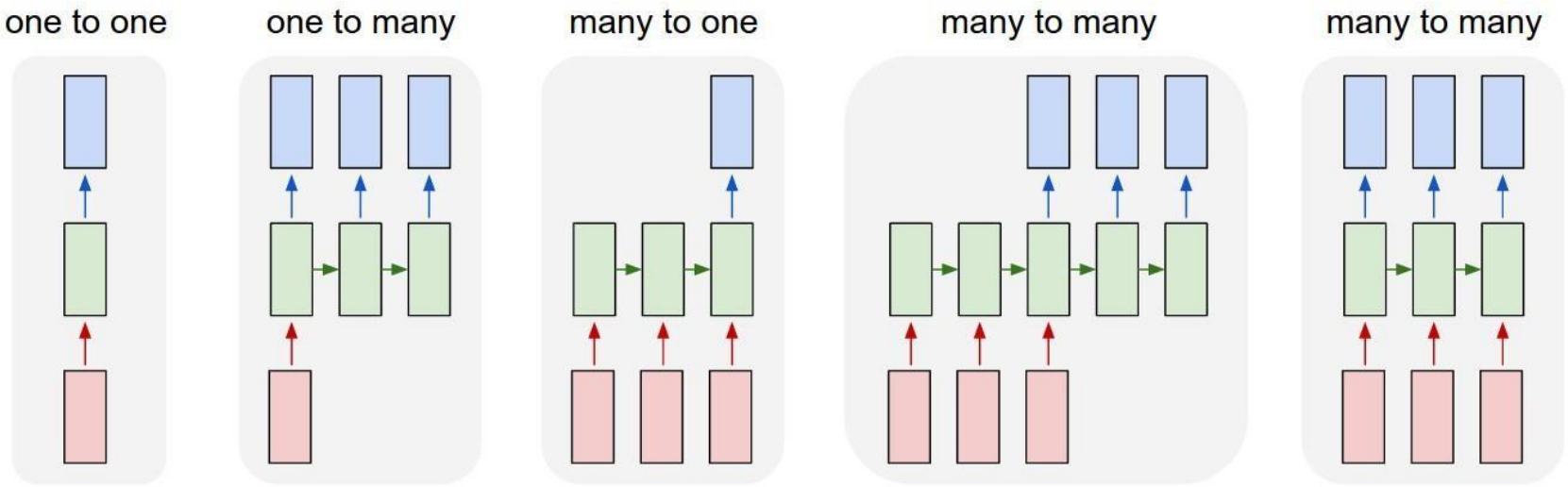
$$y_t = g_2(W_{ya} a_t)$$



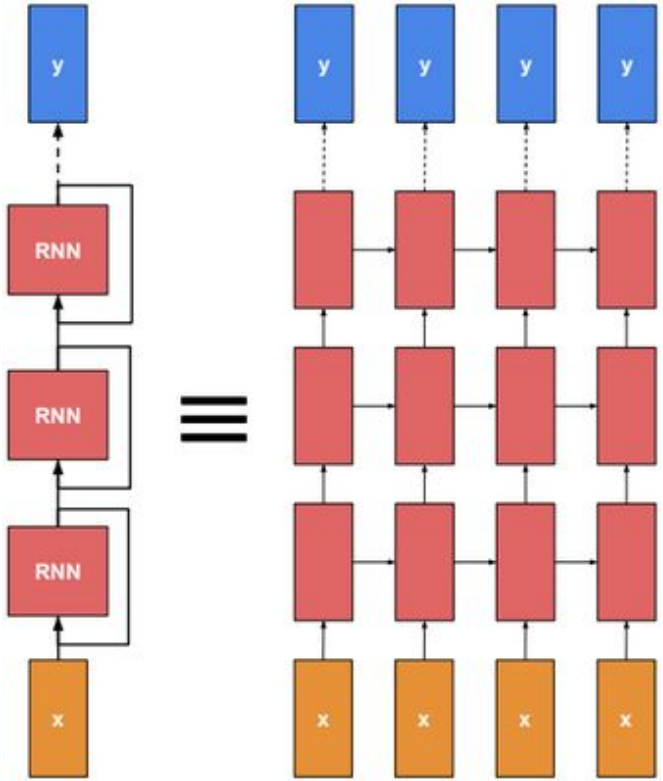
Общая картинка (с bias-term)



Типы RNN



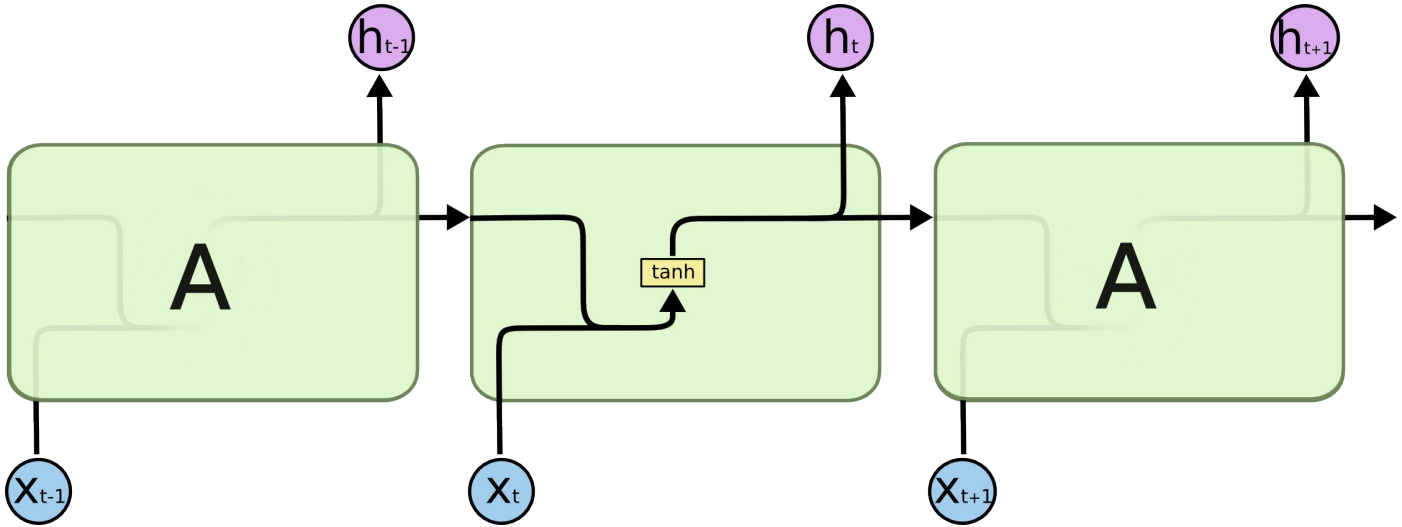
Многослойные RNN



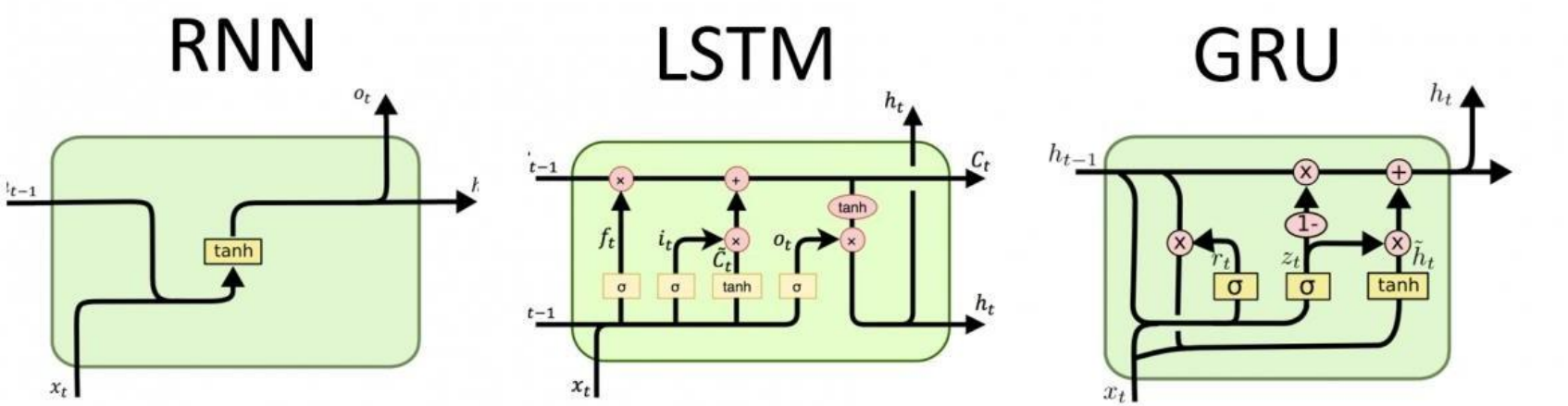
Проблемы с градиентами

- Сигнал теряется по мере прохождения
- Не факт, что получится обучить зависимость финального вектора h_n от первых слов в тексте

Классическая RNN

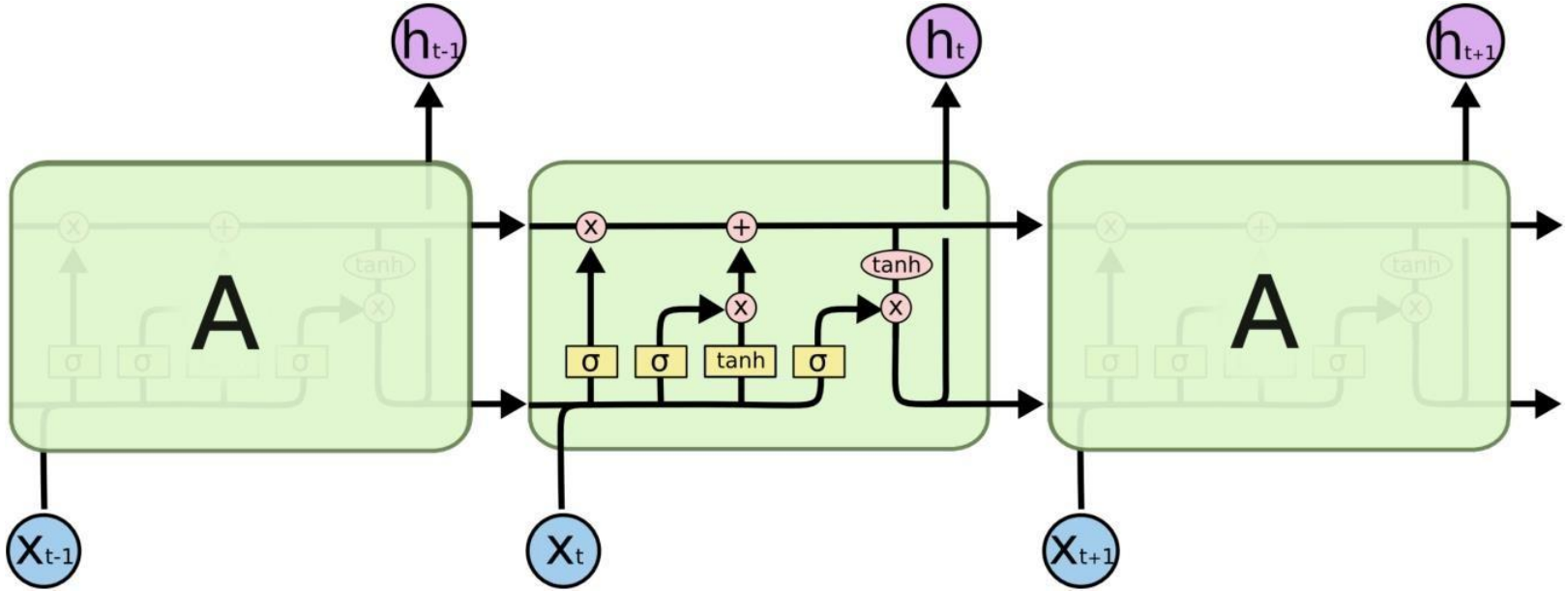


Виды рекуррентных сетей



- В LSTM 4 матрицы весов (= 4 слоя) вместо одной (в отличие от RNN)
- Матрицы весов обозначены желтыми прямоугольниками

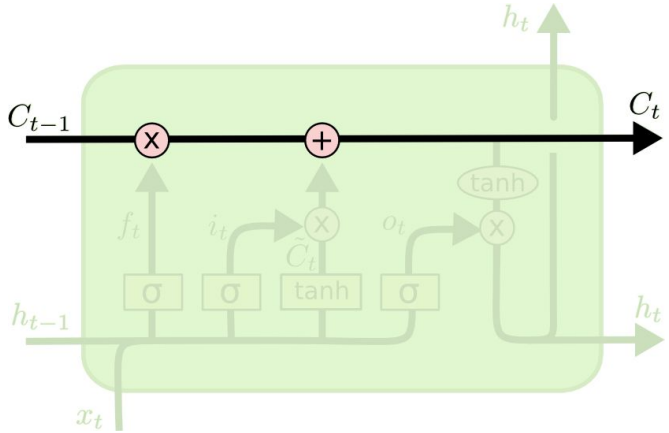
LSTM (Long Short-Term Memory)



https://colah.github.io/posts/2015-08-Understanding-LSTMs/

LSTM: C_t - состояние ячейки

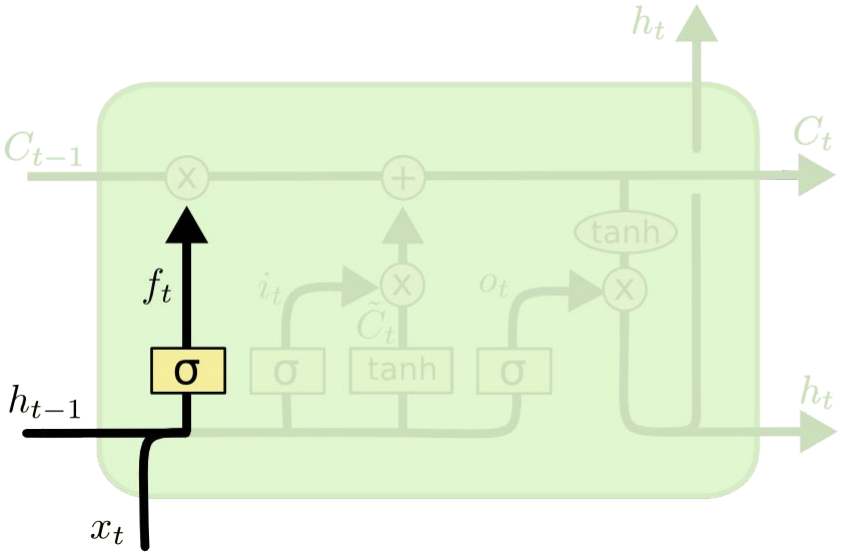
- C_t - глобальное состояние ячейки = долговременная память
- h_t - локальное состояние ячейки = кратковременная память



В него с каждым временным шагом добавляется некоторая информация, а некоторая забывается

LSTM: f_t - forget layer

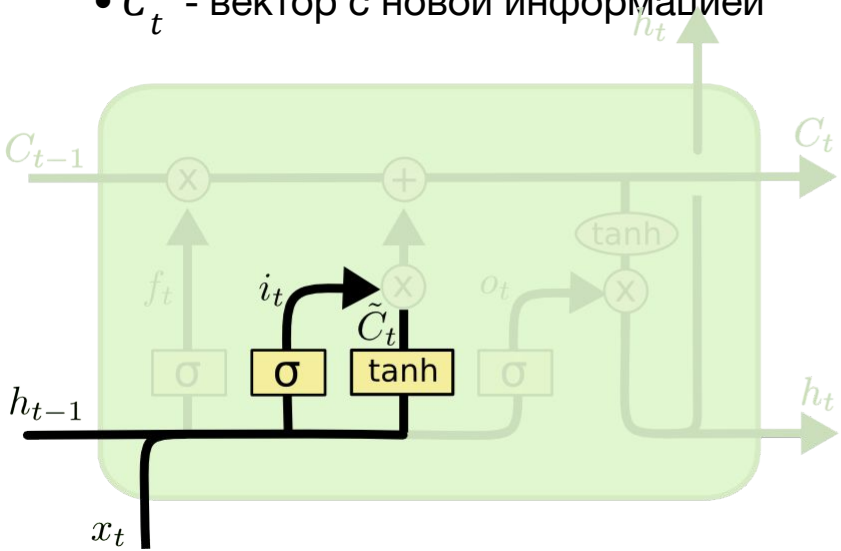
- f_t - информация с предыдущих шагов, которую хотим забыть



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

LSTM: учет новой информации

- i_t - вектор с “весами” значений, которые будем обновлять (исходя из новой информации на шаге t)
- $\tilde{C}_t$ - вектор с новой информацией

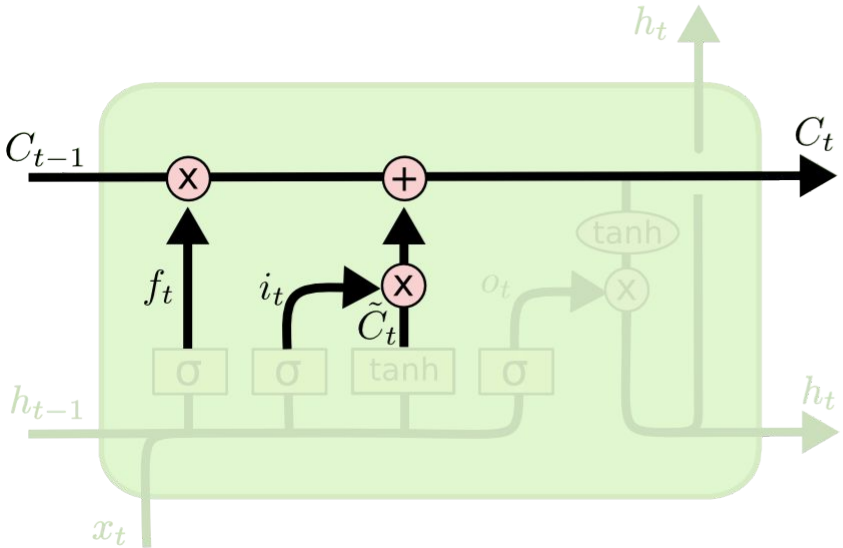


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

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

LSTM: обновление состояния ячейки

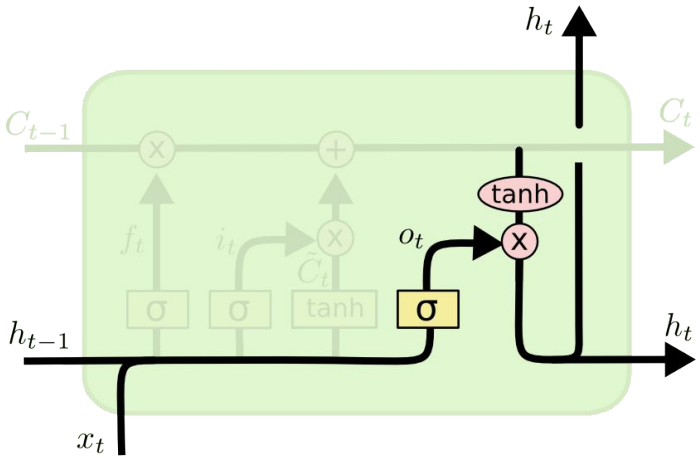
Обновляем состояние ячейки: часть забываем (первое слагаемое), часть добавляем (второе слагаемое)



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

LSTM: прогноз и обновление h_t

- Чтобы сделать прогноз o_t на текущем шаге (например, предсказываем часть речи на каждом шаге), используем поступившую на шаге t информацию и локальное состояние $h_{t\text{-}1}$
- Обновляем локальное состояние h_t с учетом прогноза o_t и обновившегося глобального состояния C_t

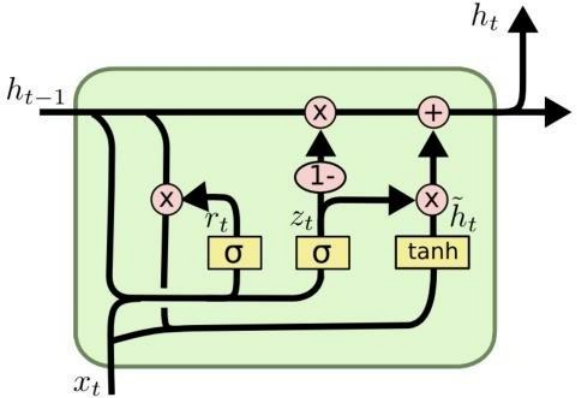


$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$

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

GRU (Gated Recurrent Unit), 2014

- Три слоя (3 матрицы весов) - логика немного отличается от логики LSTM
- Быстрее обучается, так как меньше параметров
- По качеству в большинстве задач не хуже, чем LSTM



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

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

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

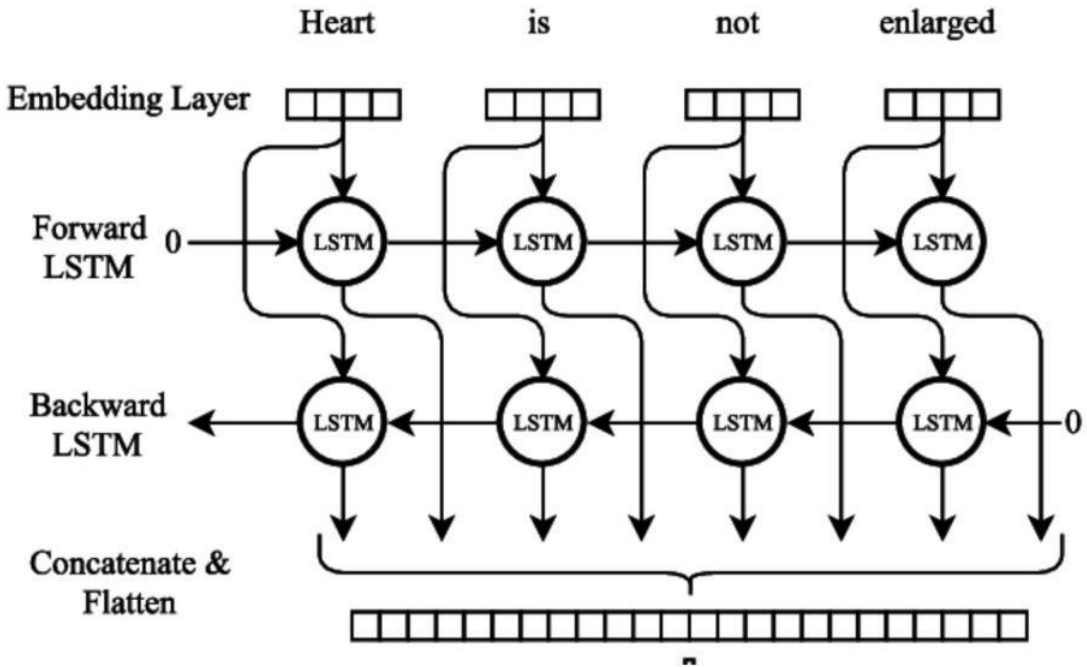
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

https://colah.github.io/posts/2015 -08 -Understanding -LSTMs/

Какой вариант рекуррентных сетей лучше?

В 2015 исследователи проводили эксперименты - по качеству все модификации LSTM показали примерно одинаковый результат.

Bidirectional LSTM



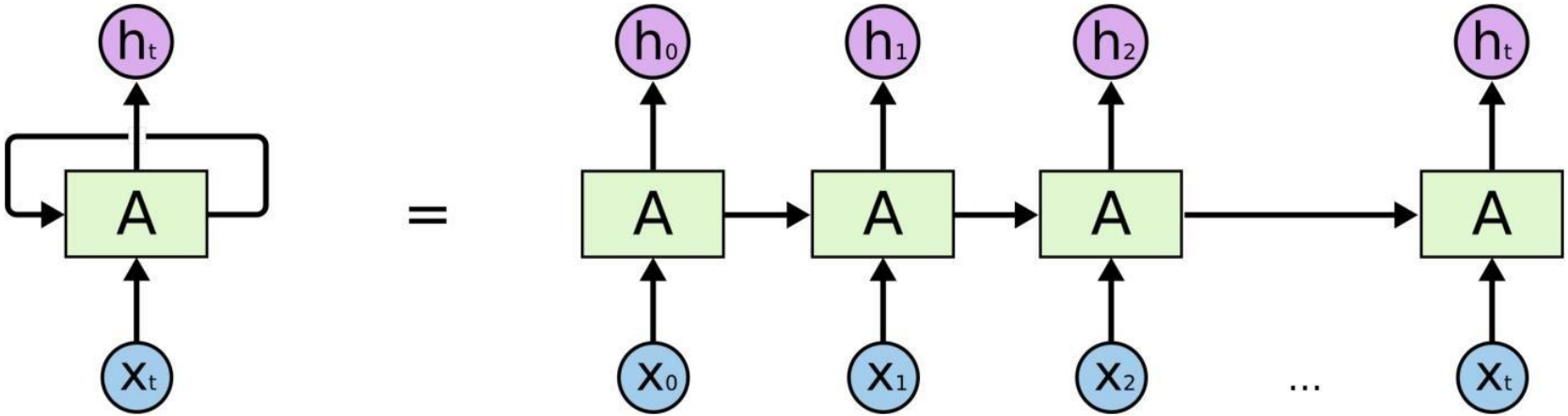
https://arxiv.org/abs/1609.08409

Seq2seq-архитектуры

Sequence to sequence

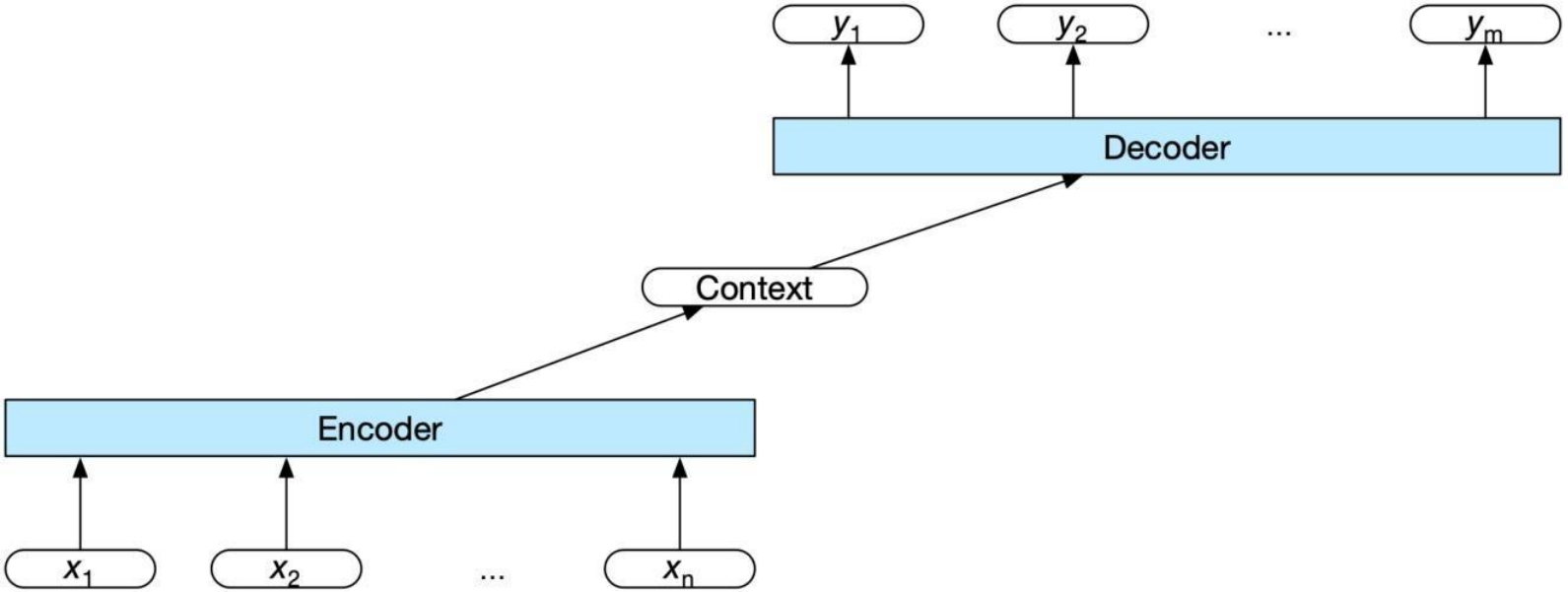
- Машинный перевод
- Суммаризация текста
- Генерация комментариев к коду
- Математические преобразования
- Смена стиля текста

Seq2seq Machine Translation



Что делать, если длины входного и выходного текстов разные?

Seq2seq Machine Translation



https://web.stanford.edu/~jurafsky/slp3/10.pdf

Seq2seq Machine Translation

- В конце входного текста ставим специальный токен <EOS>
- Прогоняем входной текст через RNN
- Скрытое состояние после всего текста — «контекст»
- Контекст передаётся в RNN, которая генерирует выходной текст
- Используется Beam Search

Seq2seq Machine Translation

- Четырёхслойные LSTM в качестве кодировщика и декодировщика
- В каждом слое — скрытые векторы размерности 1000
- Каждое слово описывается векторным представлением размерности 1000
- Входной текст подаётся «наоборот» — тогда первое слово входного текста оказывается ближе к первому слову выходного в нашей архитектуре

https://arxiv.org/abs/1409.3215

Проблемы seq2seq- архитектуры

- Нужно сжать весь текст в один вектор
- Теряется информация о первых словах
- Декодер тоже может терять информацию по мере генерации последовательности