Рекуррентные нейронные сети

Курс «Основы анализа текстовых даных»

Кафедра управления интеллектуальных технологий

НИУ «МЭИ»

2023 г.

*Рекуррентные нейронные сети**

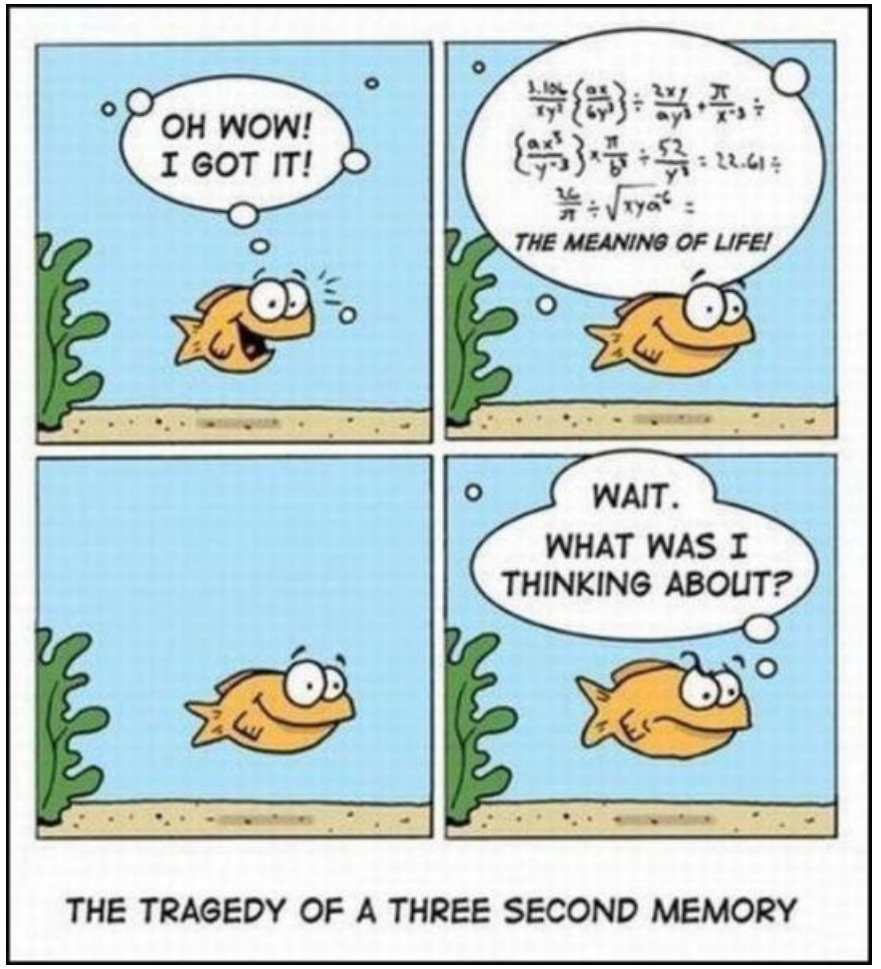
Люди не начинают думать с чистого листа каждую секунду. Читая этот текст, вы понимаете каждое слово, основываясь на понимании предыдущего слова. Мы не выбрасываем из головы все и не начинаем думать с нуля. Наши мысли обладают постоянством.

Обычные нейросети не обладают памятью, и это их главный недостаток в отличие от рекуррентых нейронных сетей (Recurrent Neural Networks, RNN, PHC).

Это сети, содержащие обратные связи и позволяющие сохранять информацию.

С RNN можем анализировать временные ряды, решать задачи генерации выходных данных – текса, музыки,...:

- One-to-many
- Many-to-one
- Many-to-many



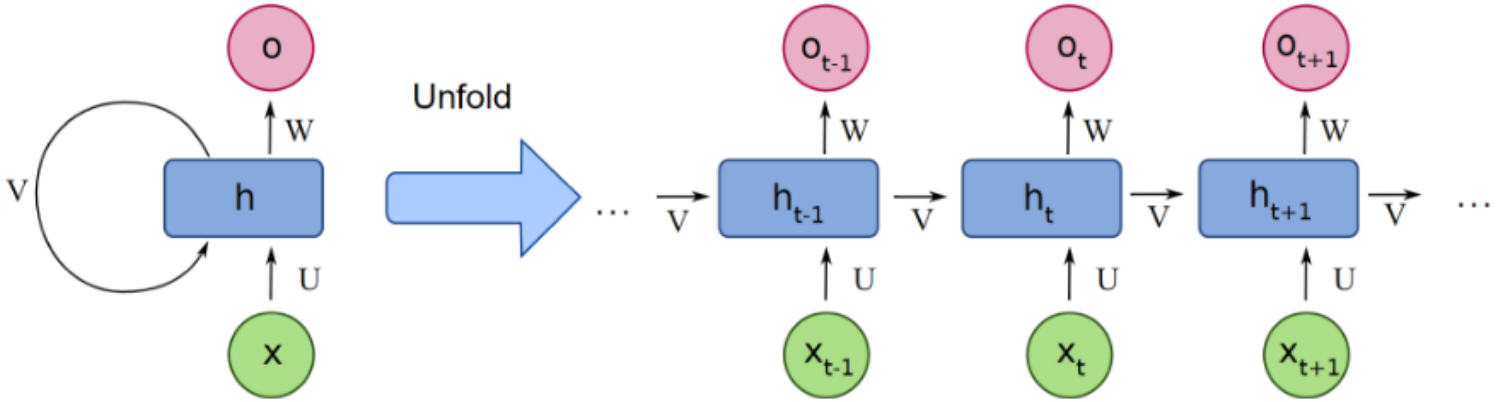
*по мотивам статей:
https://iamtrask.github.io/2015/11/15/anyone-can-code-lstm/
https://habr.com/company/wunderfund/blog/331310/

Скрытый слой RNN

Hidden state (h_t) – вектор, хранящий состояние, учитывающее глобальный и локальный контекст.

Классические нейронные сети имеют скрытые слои, и состояние скрытого слоя основывается только на входных данных.

input -> hidden -> output



Источник: Рекуррентные НС

Наличие памяти меняет эту структуру – теперь состояние скрытого слоя зависит от входных данных на текущем шаге и от состояния того же **скрытого слоя на предыдущем шаге**:

(input + prev_hidden) -> hidden -> output

Скрытый слой RNN (2)

Почему на каждом предыдущем шаге мы смотрим именно скрытый слой а не вход:

(input + prev_input) -> hidden -> output

Рассмотрим 4 шага RNN для обоих случаев:

а) смотрим скрытый слой:

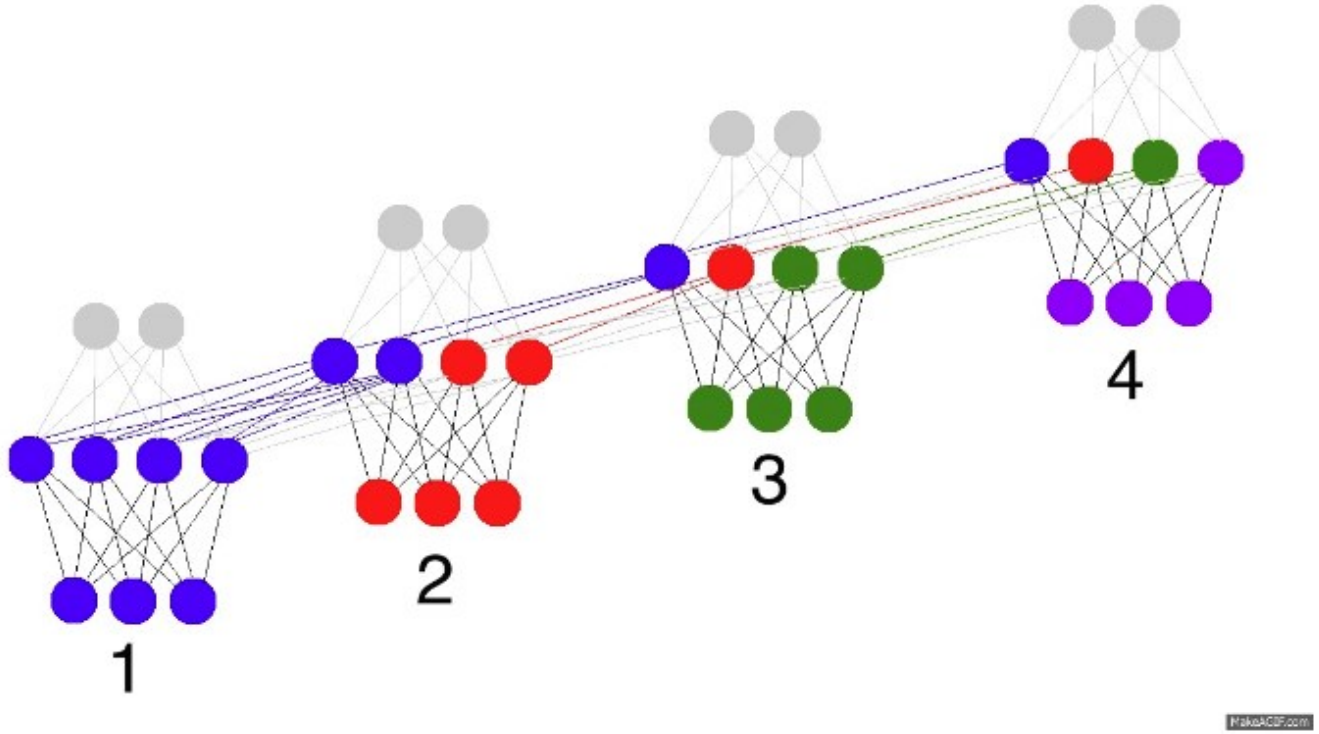
(input + empty_hidden) -> hidden -> output
(input + prev_hidden) -> hidden -> output
(input + prev_hidden) -> hidden -> output
(input + prev_hidden) -> hidden -> output

б) смотрим вход:

(input + empty_input) -> hidden -> output
(input + prev_input) -> hidden -> output
(input + prev_input) -> hidden -> output
(input + prev_input) -> hidden -> output

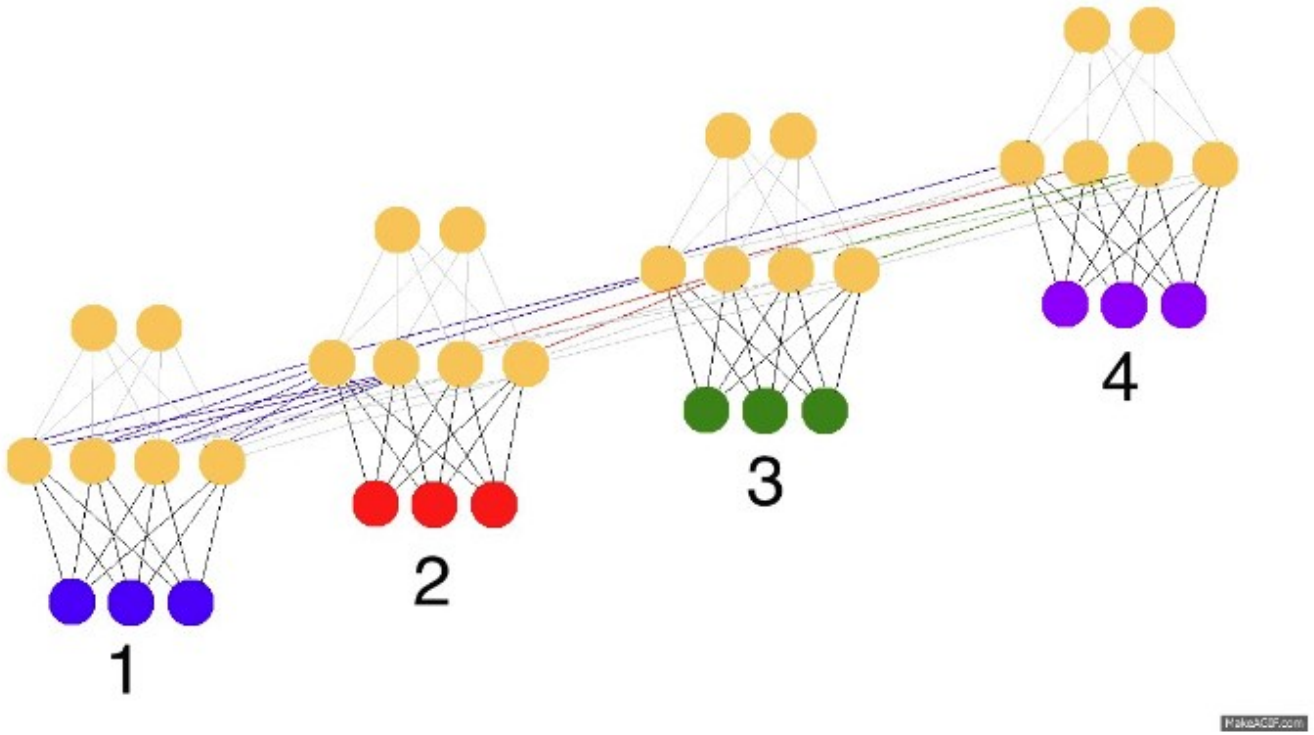
Таким образом, «память» формируется за счет комбинации входных данных и скрытого слоя на предыдущем шаге.
Такая связь реализуется с помощью весов, которые присваиваются обратной связи
$h_t = f_W(h_{t-1}, x_t)$ – функция с параметрами W, зависящая от входа и от предыдущего состояния.

Память рекуррентной нейросети (2)



- Объем памяти зависит от размера скрытого слоя.
- Выход – не является чистой функцией от входа, входной сигнал изменяет «память», а выход будет зависеть от того, что в этой «памяти» находится.
- Если на 2, 3 и 4 шагах не было бы входных сигналов, состояние скрытого слоя все равно бы менялось на каждом шаге.

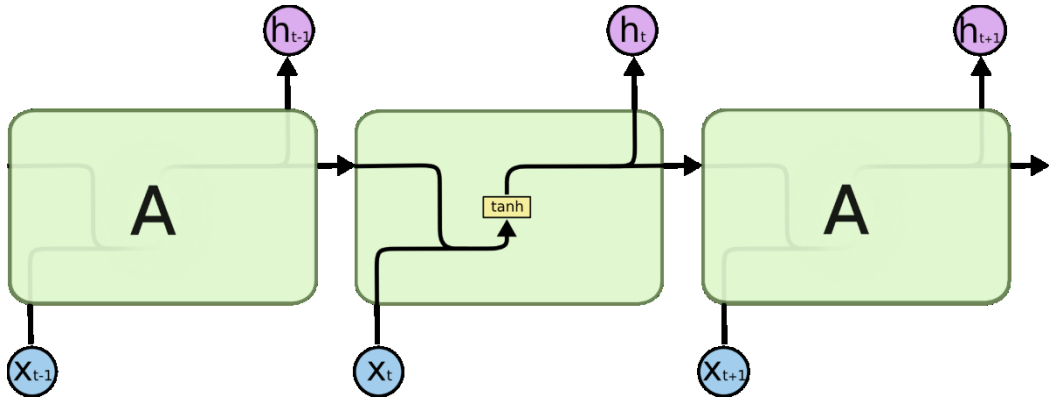
Обучение рекуррентной нейросети



- Обучение методом обратного распространения ошибки во времени (back propagation through time)
- Сначала пропускаем сигнал в прямом направлении через заданное количество последовательностей, а затем распространяем ошибку в обратную сторону через ту же последовательность.
- На gif-изображении черным цветом показаны предсказанные значения, желтым – ошибки предсказания, отклонения ошибок - оранжевым

LSTM-сети

Повторяющийся модуль в стандартной RNN состоит из одного слоя:

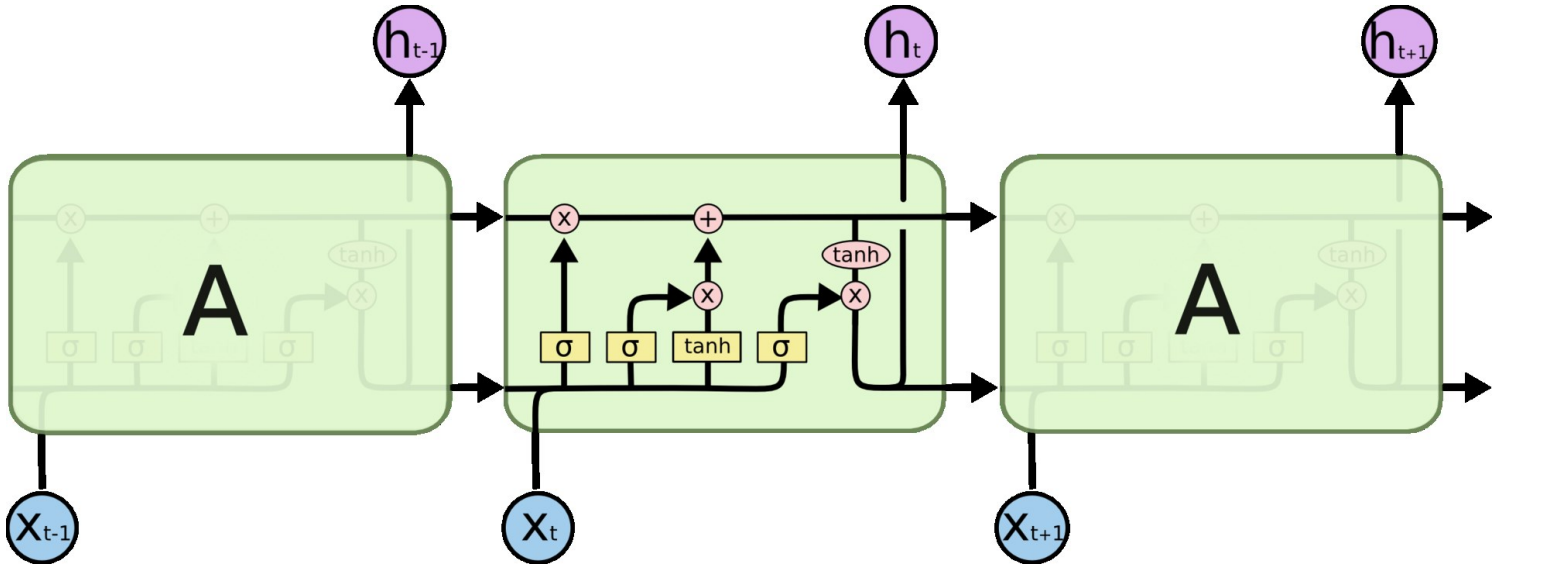


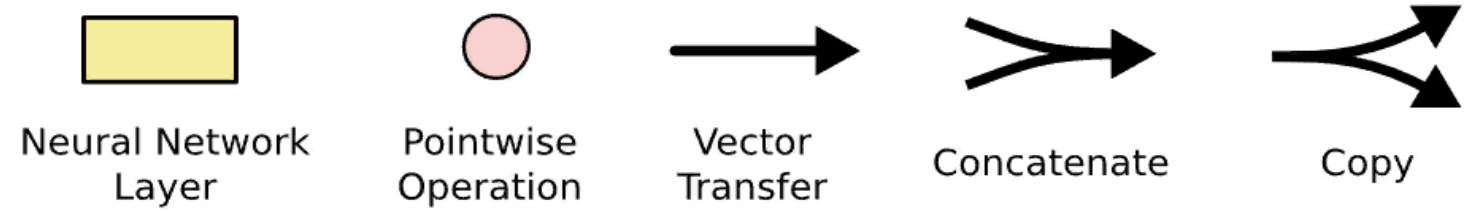
Недостаток: затухание памяти для длинных последовательностей –> теряется информация о начале предложения

LSTM - Long short-term memory, долгая краткосрочная память - особая разновидность архитектуры рекуррентных нейронных сетей, способная к обучению долговременным зависимостям.
Они были представлены Зеппом Хохрайтер и Юргеном Шмидхубером (Hochreiter & Schmidhuber (1997)) в 1997 году, а затем усовершенствованы и популярно изложены в работах многих других исследователей.

LSTM-сети

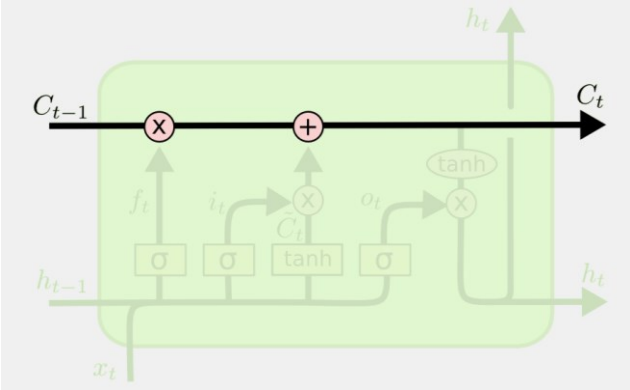
Структура LSTM также напоминает цепочку, но модули выглядят иначе. Вместо одного слоя нейронной сети они содержат целых четыре, и эти слои взаимодействуют особенным образом.



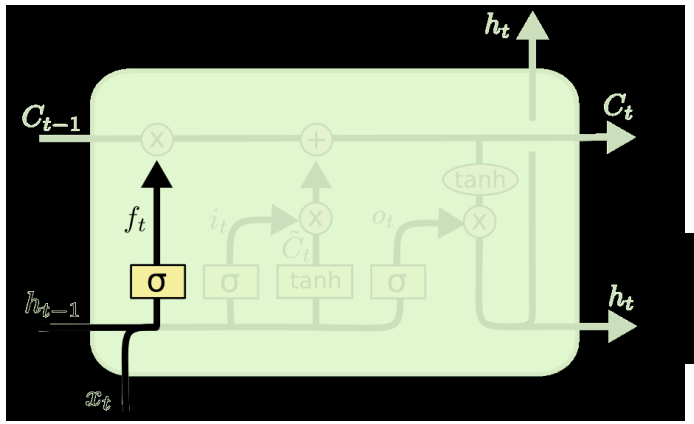


Слой нейронной сети; поточечная операция; векторный перенос; объединение; копирование.

LSTM-сети

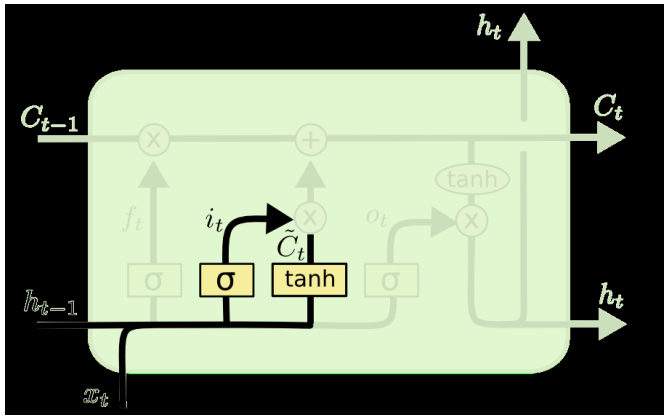


Ключевой компонент LSTM – это состояние ячейки C (cell state) – горизонтальная линия, проходящая по верхней части схемы.
Состояние ячейки напоминает конвейерную ленту. Она проходит напрямую через всю цепочку, участвуя лишь в нескольких линейных преобразованиях. Этот процесс регулируется «фильтрами» (gates).



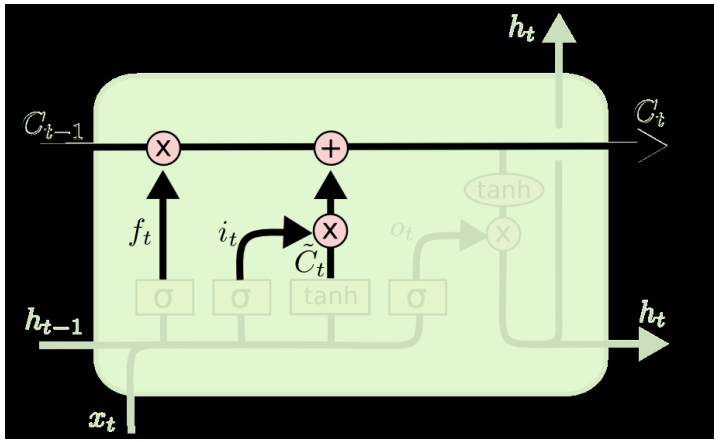
Слой фильтра забывания (forget gate layer) – определяет, какую информацию можно выбросить из состояния ячейки. Возвращает число [0;1], где 0 – полностью забыть, 1 – полностью сохранить

LSTM-сети



Следующий шаг – решить, какая новая информация будет храниться в состоянии ячейки. Этот этап состоит из двух частей. Сначала сигмоидальный слой под названием “слой входного фильтра” (input layer gate) определяет, какие значения следует обновить. Затем tanh-слой строит вектор новых значений-кандидатов которые можно добавить в состояние ячейки.

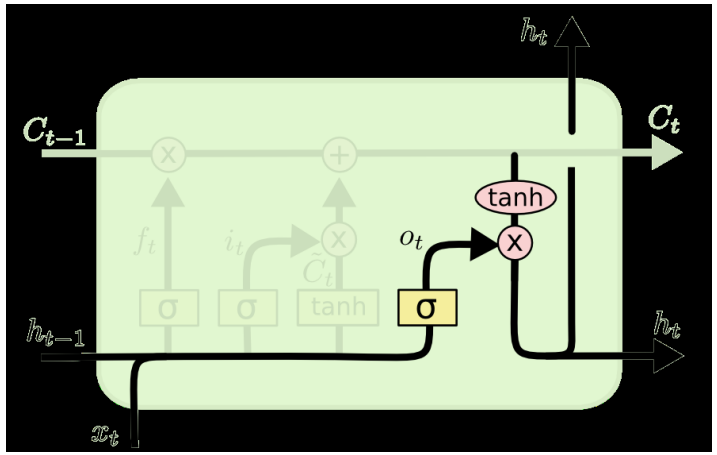
[redacted]



Меняем состояние ячейки в соответствие с предыдущими слоями forget layer, input layer и tanh layer

[redacted]

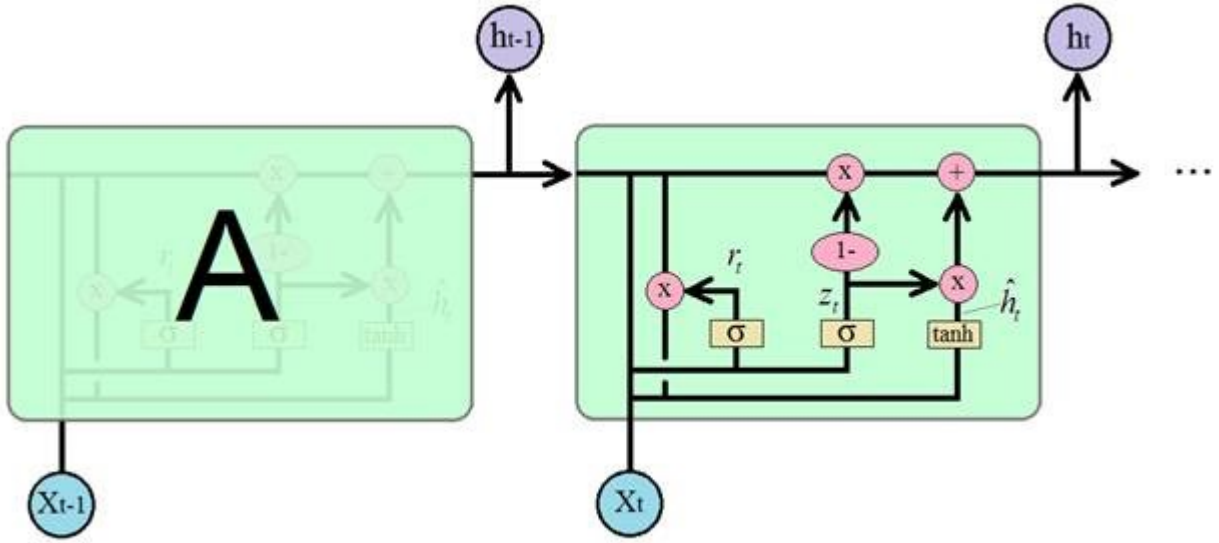
LSTM-сети



Решаем, какую информацию хотим получать на выходе h.
Выходные данные будут основаны на нашем состоянии ячейки, к ним будут применены некоторые фильтры. Сначала мы применяем сигмоидальный слой, который решает, какую информацию от входа и предыдущего шага мы будем выводить. Затем значения состояния ячейки проходят через tanh-слой, чтобы получить на выходе значения из диапазона от -1 до 1, и перемножаются с выходными значениями сигмоидального слоя, что позволяет выводить только требуемую информацию.

Gated recurrent units, GRU

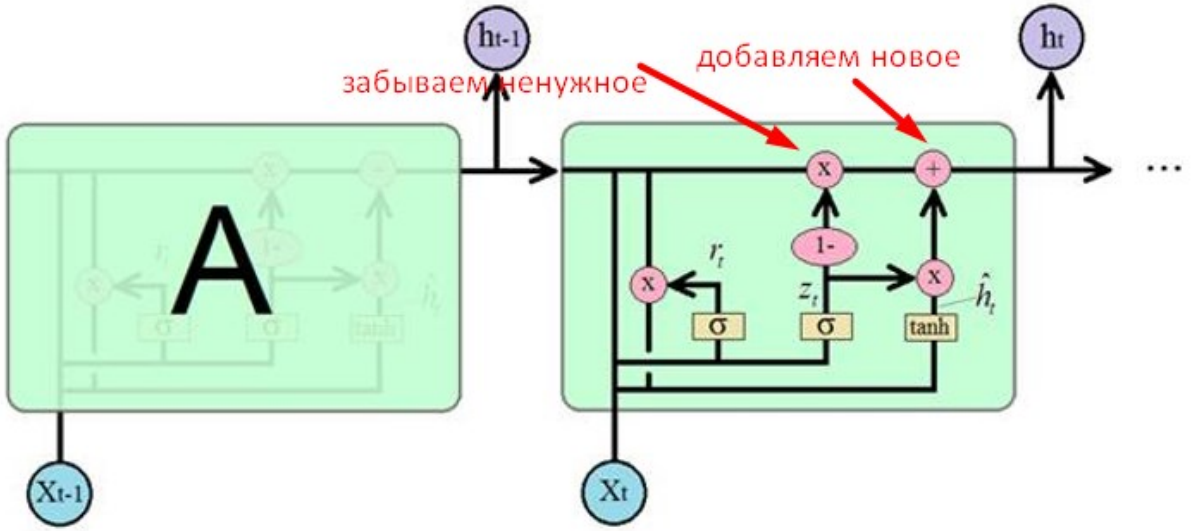
LSTM имеет существенный недостаток – большое количество настраиваемых параметров -> долго обучается, требовательна к ресурсам.
В 2014 году предложена модификация LSTM - управляемые рекуррентные нейроны (Gated recurrent units, GRU), сопоставимых по эффективности



https://proproprogs.ru/neural_network/rekurrentnye-bloki-gru-primer-realizacii-v-zadache-sentiment-analiza

Gated recurrent units, GRU

В GRU долгосрочный элемент памяти реализован не отдельным каналом, а в самом векторе скрытого состояния ht. Суть – отбросить детали и сохранить главное.



$z_t = \sigma\left(W_z \cdot \left[h_{t-1}, x_t\right] + b_z\right)$ - объединенный input и forget gate - вектор формируется из состояния на прошлом шаге и входа на текущем, который затем вычитается из 1 и перемножается с прошлым состоянием. Итоговое скрытое состояние ячейки будет рассчитываться как:

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

$$\hat{h}_t = \tanh\left(W_{xh} \cdot x_t + W_{hh} \cdot \left(r_t \otimes h_{t-1}\right) + b_h\right)$$

$$h_t = h_{t-1} \otimes \left(1 - z_t\right) + \hat{h}_t \otimes z_t$$

Пример обучения LSTM

Поставлена задача обучения сети на романе Л.Н. Толстого «Война и мир» и дальнейшего создания сетью осмысленного текста. Результаты после разного количества итераций:

100: tyntd-iafhatawiaoihrdemot lytdws e ,tfti, astai f ogoh eoase rrranbyne 'nhthnee e plia tklrgd t o idoe ns,smtt h ne etie h,hregtrs nigtike,aoaenns lng

300: "Tmont thithey" fomesscerliund Keushey. Thom here sheulke, anmerenith ol sivh I lalterthend Bleipile shuwy fil on aseterlome coaniogennc Phe lism thond hon at. MeiDimorotion in ther thize."

500: we counter. He stutn co des. His stanted out one ofler that concossions and was to gearang reay Jotrets and with fre colt otf paitt thin wall. Which das stimn

http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Пример обучения LSTM

700: Aftair fall unsuch that the hall for Prince Velzonski's that me of her hearly, and behs to so arwage fiving were to it beloge, pavu say falling misfort how, and Gogition is so overelical and ofter.

1200: "Kite vouch!" he repeated by her door. "But I would be done and quarts, feeling, then, son is people...."

2000: "Why do what that day," replied Natasha, and wishing to himself the fact the princess, Princess Mary was easier, fed in had oftened him. Pierre aking his soul came to the packs and drove up his father-in-law women.

Сначала модель выявляет общую структуру пространства слов, затем активно начинает изучать слова, начиная с коротких к более длинным. На более поздних этапах начинают выявляться зависимости появления слов и вырисовываться осмысленные предложения

Как оценить качество сгенерированного текста?

Языковая модель – статистическая модель, работающая с вероятностями слов и предложений. Мы хотим, чтобы наша модель могла предсказывать следующее слово в предложении, опираясь на предыдущие.
Будем оценивать вероятность, которую модель назначает предложению W, состоящему из последовательности слов w_i:

$$P(W) = P(w_1, w_2, \ldots, w_N)$$

При этом считаем что для логически и синтаксически корректных предложений эта вероятность будет более высокой.

Чтобы оценить качество сгенерированного текста в разных задачах могут использоваться разные подходы.

Для задачи перевода текста – прямой анализ качества перевода.

Для оценки языковой модели в целом – перплексия.

Перплексия

Перплексия – мера несоответствия модели токенам. Оценивается как обратная вероятность тестового набора, нормализованная по количеству слов. Насколько модель уверена в каждом слове.

$$PP(W) = \sqrt[N]{\frac{1}{P(w_1, w_2, \ldots, w_N)}}$$

Если тестовый набор корректных предложений «понятен» модели, модель им не «удивлена», то она будет назначать высокую вероятность этому набору, соответственно, перплексия будет малой.

Перплексия

Кроме того, перплексию можно определить через перекрестную энтропию (кроссэнтропию):

$$PP(W) = 2^{H(W)} = 2^{-\frac{1}{N}\log_2 P(w_1,w_2,\ldots,w_N)}$$

Кроссэнтропия – функция потерь, оценивающая логарифм предсказанной вероятности.
Предсказание токенов – аналог мультиклассовой классификации – предсказываем вероятность каждого токена, выбираем наиболее вероятный*.

$$CrossEntropy(p_{true}, p_{pred}) = -\sum_{i=1}^{K} p_{true}[i] * log(p_{pred}[i])$$

*На самом деле, если мы выбираем всегда наиболее вероятный токен, то генерироваться будут всегда одни и те же последовательности для одних и тех же входных данных. Чтобы генерировать разные тексты, нужно брать токены из распределения – см. лабораторную работу №4.