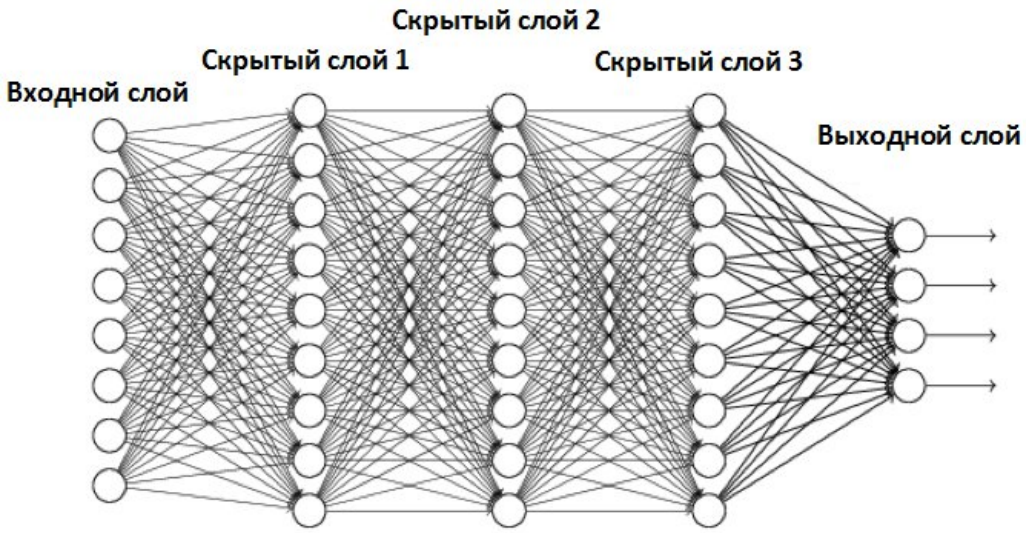
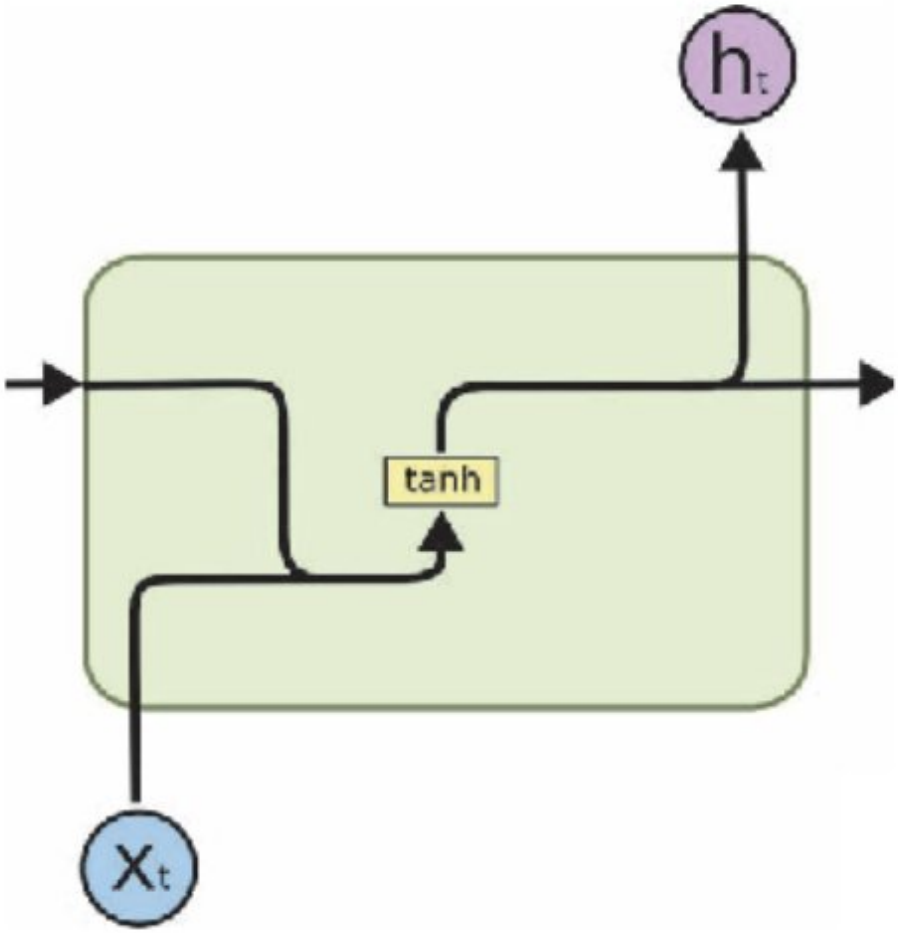


ГРУ и LSTM

Дата публикации Feb 17, 2019

Рекуррентные нейронные сети - это сети, в которых сохраняется информация. Они полезны для задач, связанных с последовательностями, таких как распознавание речи, генерация музыки и т. Д. Однако RNN страдают от кратковременной памяти. Если последовательность достаточно длинная, им будет сложно переносить

информацию с более ранних временных шагов на более поздние. Это называется исчезающей проблемой градиента. В этой статье мы рассмотрим сети Gated Recurrent Unit (GRU) и сети с кратковременной оперативной памятью (LSTM), которые решают эту проблему. Если вы не читали о RNN, вотссылканана мой пост, объясняющий, что такое RNN и как он работает.

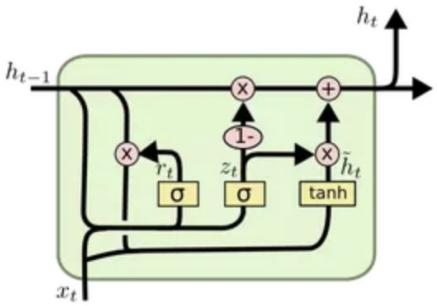


Базовая архитектура RNN Cell

Архитектура стандартного RNN показывает, что повторяющийся модуль имеет очень простую структуру, всего одну*TANH*слой. Как GRU, так и LSTM имеют повторяющиеся модули, такие как RNN, но повторяющиеся модули имеют разную структуру.

Ключевой идеей для GRU и LSTM является состояние ячейки или ячейка памяти. Это позволяет обеим сетям сохранять

любую информацию без особых потерь. В сетях также есть шлюзы, которые помогают регулировать поток информации в состояние ячейки. Эти ворота могут узнать, какие данные в последовательности важны, а какие нет. Делая это, они передают информацию в длинных последовательностях. Теперь давайте попробуем разобраться с ГРУ или Gated Recurrent Units, прежде чем мы перейдем к LSTM.



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

Базовая архитектура ячейки ГРУ

Мы можем ясно видеть, что архитектура ячейки GRU намного сложнее, чем простая ячейка RNN. Я нахожу уравнения более понятными, чем диаграмма, поэтому я все объясню, используя уравнения.

Первое, что мы должны заметить в ячейке GRU, это то, что состояние ячейки*час*равно выходу во время*Т*, Теперь давайте посмотрим на все уравнения одно за другим.

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

На каждом временном шаге у нас будет два варианта:

1. Сохранить предыдущее состояние ячейки.
2. Обновите его значение.

Вышеприведенное уравнение показывает обновленное значение или кандидата, который может заменить состояние ячейки во время*Т*, Это зависит от состояния ячейки на предыдущем шаге*час*и ворота релевантности под названием*р*,

который вычисляет релевантность предыдущего состояния ячейки при расчете текущего состояния ячейки.

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

Как мы видим, актуальность воротримеет сигмовидную активацию, значение которой находится в диапазоне от 0 до 1, которая решает, насколько релевантна предыдущая информация, а затем используется в качестве кандидата для обновленного значения.

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

Текущее состояние ячейкичасявляется отфильтрованной комбинацией предыдущего состояния ячейкичаса *также*обновленный кандидатч *(тильда)*, Ворота обновления*Z*здесь решается часть обновленного кандидата, необходимая для вычисления текущего состояния ячейки, которая, в свою очередь, также решает, какая часть предыдущего состояния ячейки сохраняется.

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

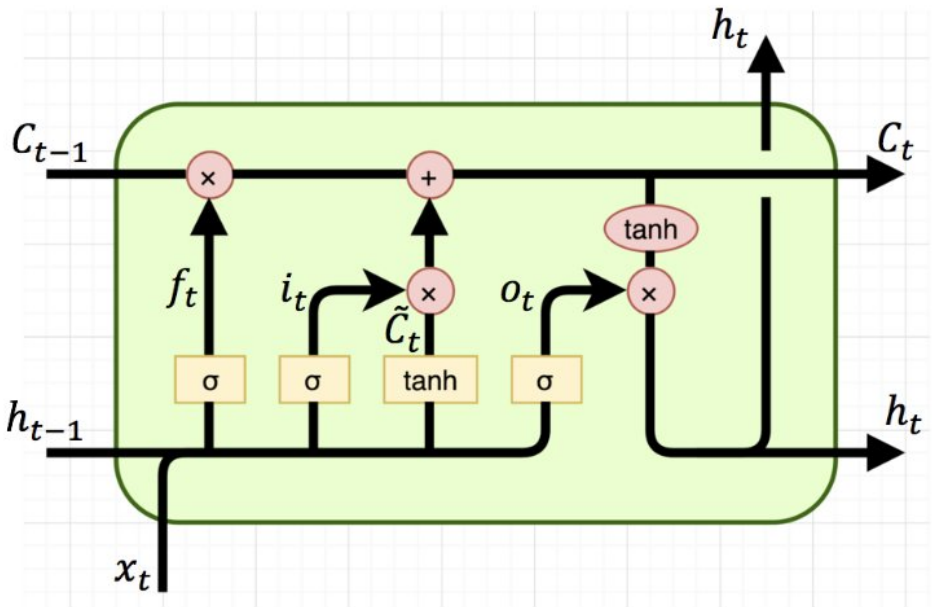
Подобно шлюзу релевантности, шлюз обновления также является сигмовидной функцией, которая помогает GRU сохранять состояние ячейки так долго, как это необходимо. Теперь давайте посмотрим на пример, который мы видели в посте RNN, чтобы лучше понять GRU.

«*Собаки, принадлежавшие миссис Смит, поняли, что в доме есть люди, и лают.*»

Слово «собаки» здесь необходимо, чтобы знать слово «были» в конце, потому что собаки множественного числа. Давайте

иметь состояние клеткис= 1 для множественного числа. Итак, когда ГРУ достигает слова «собаки», оно понимает, что речь идет о предмете предложения, и сохраняет значениес= 1 в состоянии ячейки. Это значение сохраняется до тех пор, пока не достигнет слова «были», когда оно понимает, что предметом является множественное число, и слово должно быть «были», а не «было». Здесь шлюз обновления понимает, когда сохранить значение, а когда забыть его. Поэтому, как только слово «были» закончено, оно знает, что состояние ячейки больше не является полезным, и забывает об этом. Вот как ГРУ сохраняет память и, таким образом, решает проблему исчезновения градиента.

Хотя основная идея LSTM та же, это более сложная сеть. Давайте попробуем понять это подобным образом.



Базовая единица ячейки LSTM

С первого взгляда ячейка LSTM выглядит пугающе, но давайте попробуем разбить ее на простые уравнения, как мы делали для GRU. В то время как у GRU есть два шлюза, называемых шлюзом обновления и шлюзом релевантности, у LSTM есть три шлюза, а именно шлюз забывания.*е*обновить воротая*я*и выходной вентиль*о*,

В GRU состояние ячейки было равно состоянию / выходу активации, но в LSTM они не совсем совпадают. Выход во время "t" представлен как*час*тогда как состояние ячейки представлено*с*,

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

Как и в GRU, состояние ячейки в момент времени «t» имеет значение кандидата*с (тильда)*который зависит от предыдущего вывода*час*и вход x,

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

Как и в ГРУ, текущее состояние ячейки св LSTM - отфильтрованная версия предыдущего состояния ячейки и значения кандидата. Тем не менее, фильтр здесь определяется двумя воротами, шлюзом обновления и шлюзом забывания Врата забытия очень похожи на значение (1-updateGate) в ГРУ. И забыть, и обновить ворота - сигмоидальные функции.

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

Шлюз забывания вычисляет, какая часть информации из предыдущего состояния ячейки требуется в текущем состоянии ячейки.

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

Гейт обновления вычисляет, сколько из значения кандидата*с (тильда)*требуется в текущем состоянии ячейки. И шлюз обновления, и шлюз забывания имеют значение от 0 до 1.

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

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

Наконец, нам нужно решить, что мы собираемся выводить. Эти выходные данные будут фильтрованной версией нашего состояния ячейки. Итак, мы передаем состояние клетки через*TANH*слой, чтобы выдвинуть значения между -1 и 1, затем умножить его на выходной вентиль, который имеет сигмовидную активацию, чтобы мы выводили только то, что решили.

И LSTM, и GRU очень популярны в последовательных задачах глубокого обучения. В то время как ГРУ хорошо работает для некоторых проблем, LSTM хорошо работает для других. GRU намного проще и требуют меньше вычислительных ресурсов, поэтому их можно использовать для формирования действительно глубоких сетей, однако LSTM более мощные, так как имеют большее количество шлюзов, но требуют больших вычислительных мощностей. С этим, я надеюсь, у вас есть базовое понимание LSTM и GRU, и вы готовы погрузиться в мир моделей последовательностей.

Ссылки:

1. http://colah.github.io/posts/2015-08-Understanding-LSTMs/
2. https://www.coursera.org/learn/nlp-sequence-models

Оригинальная статья

- Фреймворки и библиотеки (большая подборка ссылок для разных языков программирования)
- Список бесплатных книг по машинному обучению, доступных для скачивания
- Список блогов и информационных бюллетеней по науке о

данных и машинному обучению
- Список (в основном) бесплатных курсов машинного обучения, доступных в Интернете

map