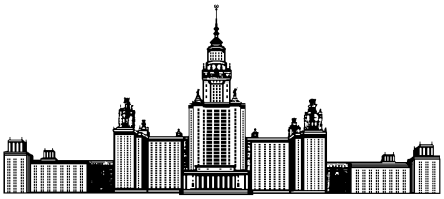


Московский государственный университет имени М. В. Ломоносова

Факультет вычислительной математики и кибернетики

Кафедра математических методов прогнозирования

Драпак Степан Николаевич

«Использование представлений слов в нейросетевых рекуррентных моделях»

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

Научный руководитель:

н.с.

Кропотов Дмитрий Александрович

Москва, 2017

Содержание

Аннотация

Данная работа посвящена анализу различных представлений слов и документов в задаче классификации с использованием рекуррентных нейронных сетей. Рассмотрены различные способы представления слов и документов, предложен способ учета представлений в LSTM сетях, проведены эксперименты, показывающие применимость данного способа.

1 Введение

Одна из самых популярных задач в современном машинном обучение — задача классификация текстов. Прошло много времени, появилось множество современных алгоритмов, которые побеждают в конкурсах, бьют все возможные бенчмарки в других задачах, например, сверточные нейронные сети в задачах распознавания изображений. В то же время большая часть экспериментов в задаче классификации текстов заканчиваются тем, что лучшие результаты в них показывает простейшая логистическая регрессия, построенная на tf-idf представлении слов. В последнее время популярными становятся рекуррентные нейронные сети, в частности, LSTM сети. Они показывают потрясающие результаты в очень большом спектре задач, в том числе и связанных с обработкой текстов.Однако, классические LSTM сети по-прежнему не могут соперничать с традиционной связкой — логистическая регрессия + tf-idf представление в задаче классификации.Цель данной работы: изучив последние исследования из области рекуррентных сетей, а так же различные современные способы представления слов, органичное соединив их и дополнив, получить представление, какая комбинация будет наилучшей и как она будет работать относительно традиционных методов.

В обзоре литературы будет дано представление о классических LSTM сетях, а так же о их последних удачных модификациях, как, например, в статье [1], где к рекуррентным слоям добавляются сверточные слои. Кроме того, рассмотрим современные способы представления слов, такие как word2vec [2] и Adaptive skip-gram [3].

В качестве новых подходов будет представлена модификация LSTM сети, которая позволит учитывать, как дополнительные представления слов, так и дополнительные представления документов. Будут описаны эксперименты на популярных наборах данных и сопоставлены с классическими подходами.

1.1 Постановка задачи

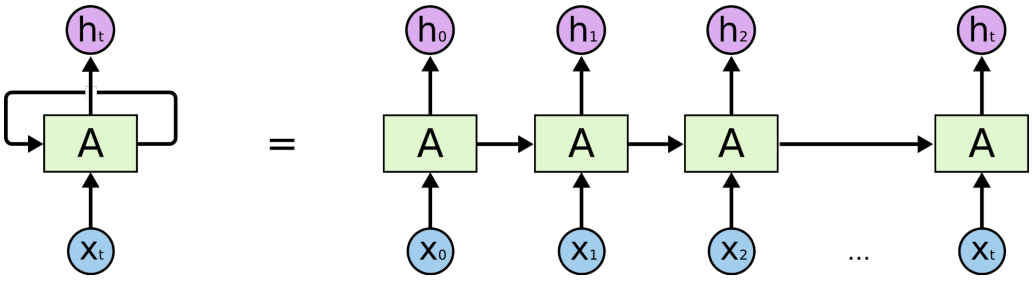
1.1.1 Задача классификации текстов

Основная задача, которая будет рассматриваться в этой работе — это задача классификации текстов. Будем рассматривать совокупность объектов $X^{train} = \{d_1, ...d_N\}$ — множество документов из обучающей выборки, заданных своими представлениями. Поскольку вопрос представления текстов и слов является одним из ключевых в данной работе, более подробно про это будет сказано в разделах Представление слов и Представление документов. Для каждого из l текстов известен класс, к которому относится данный текст. Класс текста с номером i будем обозначать y_i. Классов, вообще говоря, может быть неограничено много. То есть $y_i \in \{0, 1, ..., M\}, M \in \mathbb{N}$. Задача состоит в прогнозировании класса вновь приходящих объектов, заданных своими представлениями $X^{test} = \{d_1, ...d_q\}$.

1.1.2 Постановка задачи для рекуррентных сетей

В центре этой работы находятся рекуррентные нейронные сети. В этом разделе дадим только основные понятия и общее представление о принципах работы, подробно разберем их в обзоре рекуррентные нейронных сетей.

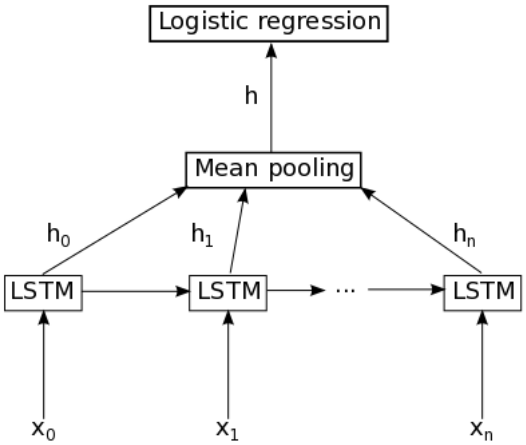
Рекуррентная нейронная сеть принимает на вход последовательность произвольных векторов. Изначально, рекуррентные сети решали задачу предсказания следующего элемента последовательности, исходя из знания о предыдущих элементах. Например, этой последовательностью может быть текст, тогда по термам $(w_1, ..., w_{k-1})$ требуется предсказать терм w_k. Проиллюстрировать это можно следующим образом:



Здесь x_t — элементы исходной последовательности. h_t — Выход сети, предполагаемый элемент, идущий следом за последовательностью $(x_1, ..., x_{t-1})$. A — совокупность слоев, которая специфична для каждой сети. Специфика этих сетей в том, что две соседних ячейки связаны и помимо выхода h_t существует еще один выход, который переносит информацию в следующую ячейку. Подробно пример будет разобран в следующем разделе.

Заметим, что, вообще говоря, в таком виде, рекуррентные сети не предполагают решение задачи классификации. Для того, чтобы свести такую задачу к задаче классификации,возможны следующие подходы:

Можно сделать некоторое преобразование над выходами последовательности h_t и получить новое представление документа. Таким преобразованием может быть одним из видов пулинга(pooling). В частности, это может быть усреднение всех выходов. Потом, можно обучать любую модель классификации.



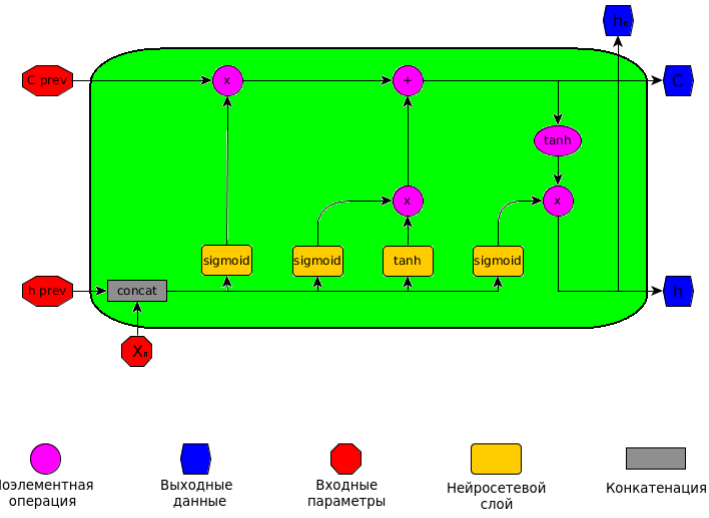
Другой вариант — это настройка сети на последний выход. То есть, если имеется последовательность длины T, то мы можем добавить в конец элемент x_{T+1}, в который будем передавать метку класса объекта. Тогда на выходе h_T для нового документа можно ожидать метку его класса. Можно над выходом h_{T+1} надстроить еще одну сеть, например, полносвязаную.

2 Рекуррентные сети

В данной работе, среди всех модификаций рекуррентных сетей нас в первую очередь будут интересовать LSTM сети. Рекуррентные сети широко используются, но у них есть ряд существенных недостатков. Остановимся на одном из них. Пусть у нас есть предложение в тексте "Я учусь на факультете ВМК *МГУ*". Здесь, для того чтобы предсказать слово *МГУ*, достаточно посмотреть на 2-3 предшествующих слова. Но, что, если у нас есть большой текст, и в нем есть пара разнесенных предложений, например: "Мой друг ездил на стажировку в Лондон. ... Кстати говоря, он свободно владеет *английским*". Здесь по нескольким предшествующим словам можно предположить, что речь идет о владение языком, но чтобы понять о каком языке идет речь необходимо вернуться на несколько предложений назад. С такими задачами помогают справиться LSTM-сети (Long Short Term Memory).

2.1 LSTM Сети

LSTM сети были призваны решить проблему выбора того, какую информацию пропускать на следующий слой, а какую оставить. Далее схематично показана структура простейшей LSTM-сети (это далеко не единственный вариант).



Здесь каждый входной элемент — вектор x, содержащий признаковое описание объекта последовательности.

h_{prev} — результат, получившийся на выходе предыдущей ячейки. $h \in \mathbb{R}^{\urcorner}$, где k — число нейронов скрытого слоя.

C_{prev} — это дополнительный выход предыдущей ячейки. Он нужен, чтобы хранить информацию о предшествующих элементах последовательности. То есть его главное отличие от h в том, что он несет информацию, главным образом, не о предыдущей ячейки, но о большом числе, идущих до нее.

Конкатенацию здесь следует понимать, как соединение двух векторов в один, размерность которого, соответственно, равна сумме размерностей исходных.

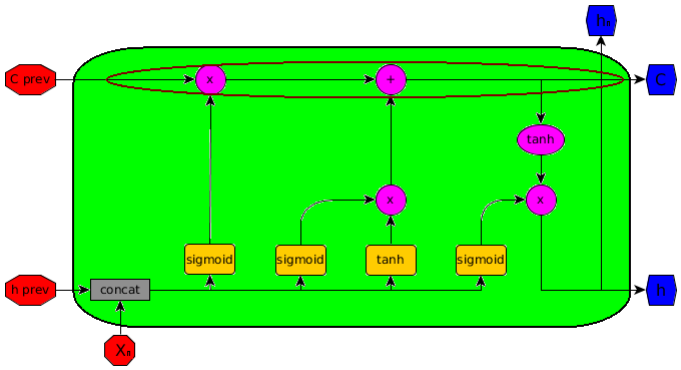
В схеме используются следующие функции активации:

$$\sigma(x) = \frac{1}{1+e^{-x}}$$
$$tanh(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$$

Подробно разберем структуру сети.

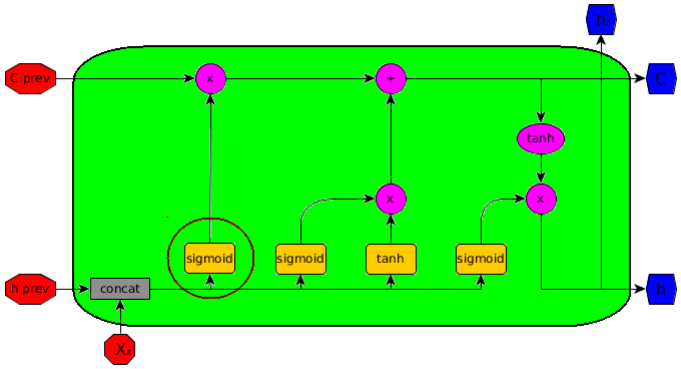
2.1.1 Первый шаг

Одно из принципиальных отличий LSTM от остальных рекурентных сетей — это последовательность слоев, которая на диаграмме изображена прямой линией в верхней части.



Через нее информация может спокойно проходить в следующую ячейку, при этом каждый новый элемент последовательности может воздействовать на нее по-разному, может поменять полностью, а может оставить без изменений.

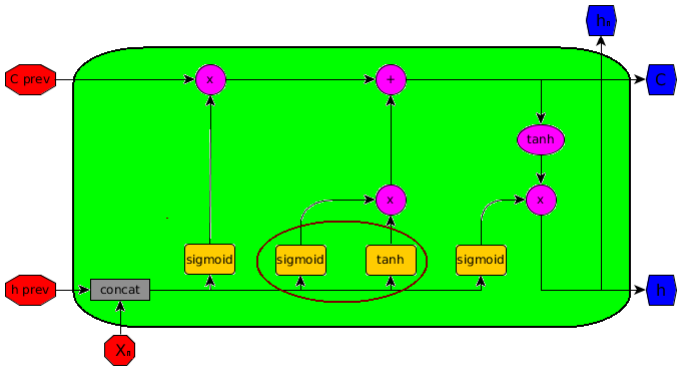
Обратимся теперь к самим нейросетевым слоям. Рассмотрим первую сигмоиду.



Здесь сеть выбирает, какая информация о предшествующем элементе должна пройти через текущую ячейку. Для этого конкатенируются вектор выхода предыдущей ячейки h_{t-1} и новый элемент последовательности x_t и в качестве одного вектора пропускаются через слой с сигмоидной функцией активации. Назовем вектор, полученный после конкатенации v_t.

2.1.2 Второй шаг

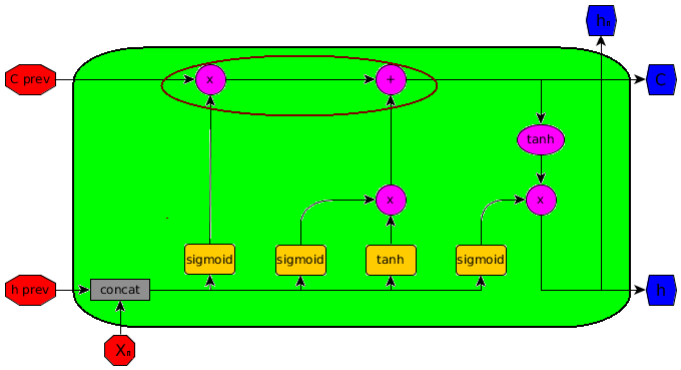
Теперь расмотрим следующие два слоя. Они отвечают за то, какую новую информацию мы привнесем в модель.



Сигмоиду в этой структуре обычно называют “input gates layer”. Она решает, какие значения вектора будут обновлены. Затем слой с гиперболическим тангенсом определяет новых “кандидатов” для C. Далее, эти вектора поэлементно перемножаются, чтобы потом прибавиться к новому значению C.

2.1.3 Третий шаг

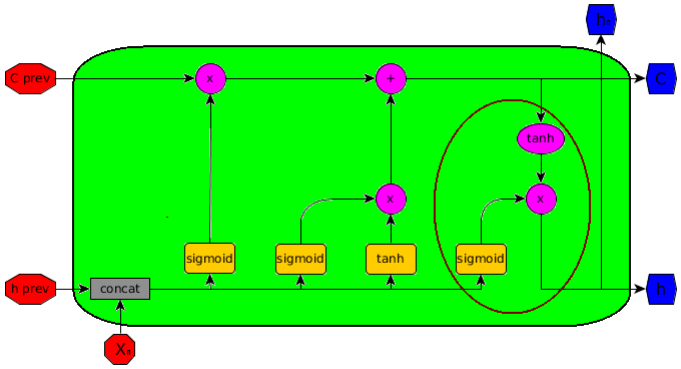
Теперь самое время обновить значения новые вектора C.



На шаге 1 и 2 уже решено, какие значения будут записаны в C, остается лишь поэлементно домножить старое значение на то, что получено на шаге 2 и прибавить новые значения, полученные на шаге 2.

2.1.4 Четвертый шаг

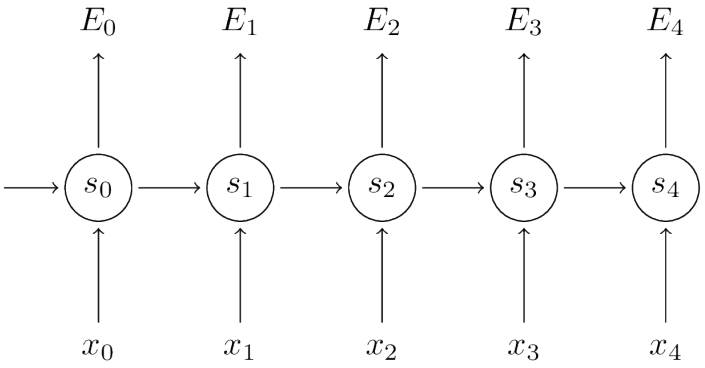
Осталось на основе информации, которая хранится в C, а также v_t получить выходное значение ячейки.



Для этого пропустим v_t через слой с сигмоидной функцией активации, чтобы решить, что из C стоит выдавать в качестве выходного значения. Потом применим слой с тангенциальной функцией активации к C и умножим на выход сигмоидного слоя, чтобы получить только нужные значения. Полученный результат и будет выходным значением.

2.2 Обучение рекуррентных сетей

При обучение рекуррентных сетей возникают некоторые сложности, потому что по сути, выход h_t зависит от элементов последовательности, которые были за долго до него. Поэтому при обучение таких сетей используется модифицированный алгоритм, получивший название Backpropagation Through Time (BPTT). Этот алгоритм используется для обучения всех рекуррентных сетей, мы рассмотрим его на примере простейшей сети, показанной ниже:



Выход этой сети можно записать следующим образом:

$$\hat{y}_t = softmax(Vs_t)$$

$$s_t = tanh(Ux_t + Ws_{t-1})$$

В качестве функции потерь можно взять, например, кросс-энтропию.

$$E_t(y_t, \hat{y}_t) = -y_t log(\hat{y}_t)$$

$$E = \sum_t E_t(y_t, \hat{y}_t)$$

Наша цель — настроить веса U, W, V. Для этого необходимо посчитать соответствующие градиенты.

Начнем с градиента по V.

$$\frac{\partial E_i}{\partial V} = \frac{\partial E_i}{\partial E_i \hat{y}_i}\frac{\partial E_i}{dV} = \frac{\partial E_i}{\partial \hat{y}_i}\frac{\partial E_i}{\partial V s_i}\frac{\partial V s_i}{\partial V} = (\hat{y}_i - y_i) \otimes s_i$$

То есть в итоге мы получили простую формулу для пересчета.

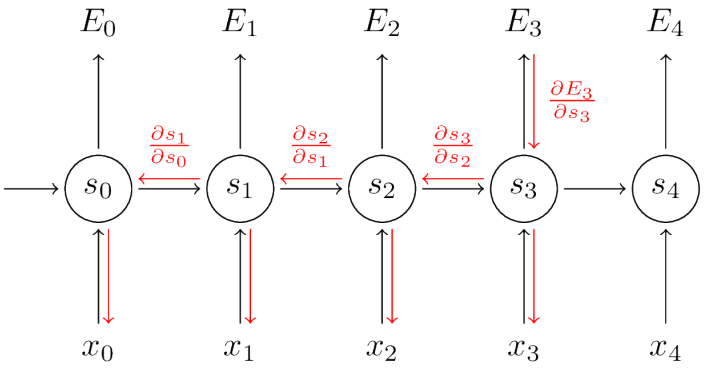
Далее рассмотрим градиент по W.

$$\frac{\partial E_i}{\partial W} = \frac{\partial E_i}{\partial \hat{y}_i}\frac{\partial \hat{y}_i}{\partial s_i}\frac{\partial s_i}{\partial W}$$

Здесь последний множитель, $\frac{\partial s_i}{\partial W}$ зависит от всей последовательности, которая была до, поэтому в итоге формула запишется в следующем виде:

$$\frac{\partial E_i}{\partial W} = \sum_{k=0}^{i} \frac{\partial E_i}{\partial \hat{y}_i} \frac{\partial \hat{y}_i}{\partial s_i} \frac{\partial s_i}{\partial s_k} \frac{\partial s_k}{\partial W}$$

Это можно проиллюстрировать следующим образом:



На практике, суммирование производят не по всем предшествующим ячейкам, а по некоторому числу последних. Это число — параметр модели.

Далее:

$$\frac{\partial E}{\partial W} = \sum_i \frac{\partial E_i}{\partial W}$$

Аналогично для V. Реально здесь нет ничего нового, по сути это все тот же backpropagation, но с небольшими особенностями. Одна из основных проблем — угасание градиента. Чем дальше элемент находится от текущего, тем меньше его вклад в градиент.

3 Представления текстов и отдельных слов

Существуют разные способы перевести текст в векторное представление. Будем считать, что изначально каждый документ представляет из себя последовательность

из n_d терминов: $(w_1, ..., w_{n_d})$ из словаря, в котором содержится W слов. Слова в словаре так же упорядочены.

3.1 Наивное представление

Начнем с самого простого варианта. Документ с номером i можно представить как вектор размера W, в котором единицы будут стоять в тех позициях, которые соответствуют словам, которые встретились в этом документе. В качестве некоторой оптимизации можно отсекать минимальное и максимальное число раз, которое слово должно встретиться в тексте, но это не так важно.

3.2 tf-idf Представление

Второй вариант связан с tf-idf преобразованием. Введем следующие обозначения:

- n_w — число документов, в которых встретился токен с номером w
- n_{wd} — число раз, которое токен с номером w встретился в документе с номером d
- $TF(w, d) = n_{wd}/n_d$ — Term-frequency, показывающая частоту встречаемости слова в данном документе
- N — число документов в коллекции
- $IDF(w) = n_w/N$ — Inverted document frequency, показывающая, насколько часто встречается слово во всей коллекции
- $TF - IDF(w, d) = TF(w, d) * IDF(w)$

В сделанных обозначениях представлением документа с номером d будет вектор, длиной W в котором на позиции с номером w будет стоять значение $TF - IDF(w, d)$.

3.3 N-граммы

Обе представленных выше моделей предполагают, что порядок слов значения не имеет, а слова генерируются независимо друг от друга. Одной из попыток ухода от такой концепции является введение n-грамм. Эти модели работают так же, как описанные выше, однако, в них в качестве терма рассматривается не одно слово, а пары, тройки и n-ки слов, идущих подряд. Например, если рассмотреть текст "Мама мыла раму то в нем будет 2 биграммы: "Мама мыла"и "мыла раму". N-граммы обычно добавляют к классическому представлению. Однако, если добавлять их достаточно много, то размерность пространства будет расти и делать какой-либо прогноз будет невозможно.

3.4 Skip-граммы

Еще одним дополнительным признаком могут быть skip-граммы. Здесь учитываются пары слов, но идущие уже на некотором расстояние друг от друга. Например, если skip-граммы берутся на расстояние одного слова, то из текста "мама мыла раму"дополнительно будет извлечен один признак, который будет соответствовать словосочетанию "мама раму". На skip-граммах будет основан один из методов, который будет активно использоваться в этой работе.

3.5 Word2vec

Это способ представления отдельных слов в векторном формате [2]. Он основан на том, что слова близкие по значению должны быть представлены векторами, близкими между собой.

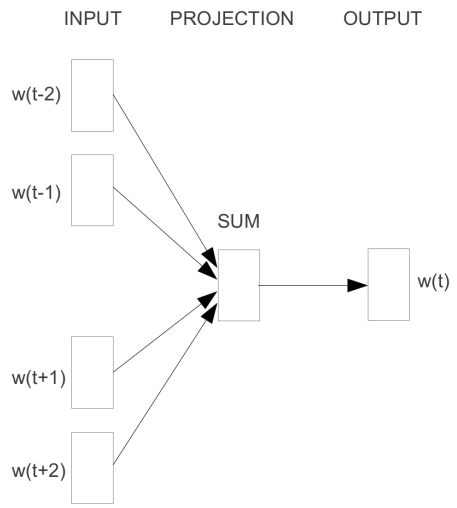
Пусть изначально у нас есть словарь размера W. Пусть каждое слово закодировано бинарным вектором с одной единицей, определяющей это слово и пусть имеется корпус документов. Введем обозначения:

Обозначим за o_t слово, которое стоит в рассматриваемом документе на месте с номером t. N — число слов в рассматриваемом тексте. Контекстом слова o_t назы-

вается совокупность слов, которая стоит на расстояние не более чем T от слова o_t. Само слово в контекст не входит. T — параметр модели.

У word2vec есть две основные модели. Обе модели в качестве входных данных принимают последовательность слов.

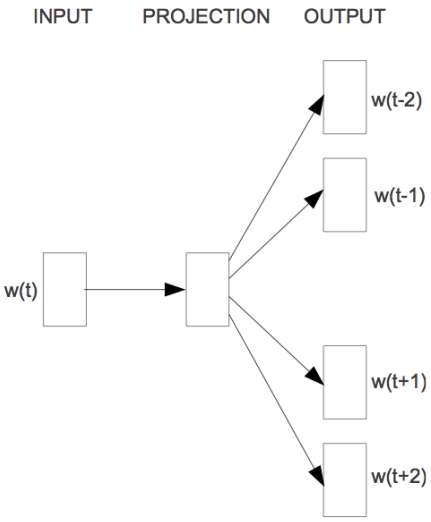
Первая модель — CBOW(Continues bag of words) имеет следующую архитектуру.:



CBOW

Задача данной сети — предсказание этого слова, исходя из контекста. Строго говоря, параметр T может не быть константой, а выбираться случайно. В простейшем случае, вектора, соответствующие словам суммируются и на этой сумме строится полносвязный слой с логистической функцией активации.

Вторая модель — Skip-gram



Skip-gram

Эта модель предсказывает слова из контекста по текущему слову. Понятно, что чем больше слов мы пытаемся предсказать, тем выше качество финальной модели и тем больше вычислений требуется.

Обозначим за y_{ij} слово с номером j в контексте слова с номером i.

C — число слов в контексте.

θ — параметры классификатора.

Тогда, итоговое правдоподобие этой модели можно записать в вероятностных терминах следующим образом:

$$p(Y|O,\theta) = \prod_{i=1}^{N} \prod_{j=1}^{C} p(y_{ij}|o_i,\theta) \tag{1}$$

В итоге, в качестве векторного представления слова берутся веса классификатора, которые соответствуют той позиции во входном векторе, на которой стоит единица в его представление.

Интересным эффектом является то, что на словах, представленных в w2v формате имеют смысл некоторые векторные операции. Например, с высокой долей вероятности, на хорошо обученной модели, если взять векторное представления слов и

провести над ними следующие операции: Москва - Россия + Германия, то результат будет очень близок к слову Берлин. Из данного подхода можно получить и представление всего текста, например, взяв и сложив все слова из данного документа. Стоит заметить, что один из основных плюсов такого подходов является то, что представление документов в таком виде значительно снижает размерность признакового пространства. Обычно w2v представление имеет размерность на несколько порядков меньше чем W.

3.6 Adaptive skip-gram

Одна из проблем не только представленных выше методов, но и вообще большей части техник, связанных с обработкой текста на естественном языке — это многозначность слов. Ни один из вышеперечисленных методов в оригинальном варианте никак не учитывает наличие нескольких значений у одного слова. Adaptive skip-gram [3](AdaGram) — это попытка расширить модель word2vec, учитывая свойство многозначности некоторых слов. В основу данного метода легли процессы Дирихле [8], которые позволяют автоматически определять число значений слов. С помощью определения процесса Дирихле через stick-breaking можно задать априорное распределение значений слов. Вся вероятность встретить слово разбивается на сумму условного бесконечного числа значений. Вероятность встретить значение номер k слова w можно записать так:

$$p(z = k|w, \beta) = \beta_{wk} \prod_{r=1}^{k-1} (1 - \beta_{wr})$$

$$p(\beta_{wk}|\alpha) = Beta(\beta_{wk}|1, \alpha)$$

Здесь $Beta$ — β распределение с плотностью

$$p(x) = \frac{1}{B(\alpha, \beta)} x^{\alpha-1}(1 - x)^{\beta-1}$$

$$B(\alpha, \beta) = \int_0^1 x^{\alpha-1}(1 - x)^{\beta-1} dx$$

Гиперпараметр α регулирует априорную информацию о том, сколько значений имеется у слова в среднем. Теперь, аналогично правдоподобию в Skip-gram модели в w2v (1) запишем полное правдоподобие AdaGram модели.

$$P(Y, Z, \beta | O, \alpha, \theta) = \prod_{w=1}^{V} \prod_{k=1}^{\infty} p(\beta_{wk}|\alpha) \prod_{i=1}^{N} \left\{ p(z_i|o_i, \beta) \prod_{j=1}^{C} p(y_{ij}|z_i, o_i, \theta) \right\}$$

Где $Z = \{z_i\}_{i=1}^{N}$ — список значений всех слов.

В итоге получаются представления похожие на извлеченных из Skip-gram w2v.

3.7 Представления из тематического моделирования

Еще одно представление документов, которое будет использовано в этой работе — это тематический профиль модели, полученный с помощью тематического моделирования [4].

Пусть D — коллекция текстовых документов, W — множество всех употребляемых в них слов. Гипотеза независимости эквивалентна предположению, что порядок слов в документах коллекции не важен для выявления тематики, то есть тематику документа можно узнать даже после произвольной перестановки слов. Таким образом, каждый документ $d \in D$ можно представить в виде последовательности слов $(w_1, \ldots, w_{n_d})$, где $w_i \in W$, $i = \overline{1, n_d}$. Число слов в документе (слова могут повторяться) обозначается n_d, число появлений слова w в документе d обозначается n_{dw}.

Предположим, что появление каждого термина w в документе d связано с некоторой скрытой переменной t из конечного множества тем T. Тогда коллекция D представляет собой выборку троек (d, w, t), взятых независимо из дискретного распределения $p(d, w, t)$ на множестве $D \times W \times T$.

Гипотезой условной независимости называется предположение, что появление слов по теме t не зависит от документа:

$$p(w|t) = p(w|d, t)$$

С учётом этой гипотезы и формулы полной вероятности, а также принятых в теории BTM обозначений $p(w|t) = \phi_{wt}$ и $p(t|d) = \theta_{td}$, тематическая модель коллекции

представляется в виде:

$$p(w|d) = \sum_{t \in T} p(w|t)p(t|d) = \sum_{t \in T} \phi_{wt}\theta_{td} \tag{2}$$

Этой вероятностной моделью описывается порождение коллекции D по известным распределениям $p(w|t)$ и $p(t|d)$. Построение тематической модели — обратная задача: по коллекции D необходимо восстановить породившие коллекцию распределения.

Задачу построения тематической модели можно трактовать как задачу поиска приближения матрицы частот F произведением двух неизвестных стохастических (с нормированными и неотрицательными столбцами) матриц Φ и Θ:

$$F \approx \Phi\Theta$$

$$F = (\hat{p}(w|d))_{|W|\times|D|} = (n_{dw}/n_d)_{|W|\times|D|}$$

$$\Phi = (\phi_{wt})_{|W|\times|T|} \quad \Theta = (\theta_{td})_{|T|\times|D|}$$

Тогда мы можем использовать столбец θ_d матрицы Θ, в качестве признакового описания документа d.

4 Современные техники классификации с помощью LSTM сетей

В качестве основы для экспериментов, будет использована модификация LSTM из статьи Rie Johnson, Tong Zhang [1], в которой подробно разобраны основные возможные подходы к классификации текстов с помощью LSTM. Далее, для удобства рассмотрения модификаций, запишем в формульном виде структуру классической LSTM сети.

$$i_t = \sigma(W^{(i)}x_t + U^{(i)}h_{t-1} + b^{(i)})$$

$$o_t = \sigma(W^{(o)}x_t + U^{(o)}h_{t-1} + b^{(o)})$$

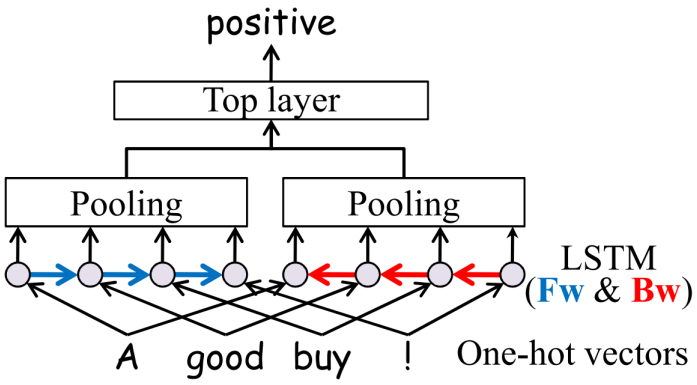
$$f_t = \sigma(W^{(f)}x_t + U^{(f)}h_{t-1} + b^{(f)})$$

$$u_t = \tanh(W^{(u)}x_t + U^{(u)}h_{t-1} + b^{(u)})$$

$$c_t = i_t \odot u_t + f_t \odot c_{t-1}$$

$$h_t = o_t \odot tanh(c_t)$$

Общая схема предсказания, предложенная в статье [1] выглядит следующим образом:



Здесь стоит выделить сразу несколько моментов

- Используются простые one-hot encoded вектора
- Целевая переменная подается не в качестве последнего выхода, а выходы каждой ячейки используются для обучения вышестоящей модели.
- Сеть работает в две стороны (bidirectional LSTM). То есть документ прогоняется не только в прямую сторону, но и в обратную.

Далее подробно рассмотрим данную архитектуру.

4.1 Word embedding

Word embedding — это линейная операция, вида Ex_t, которая переводит входной вектор в представление меньшей размерности. Такая техника часто используется при работе с LSTM сетями, причем обучают такие слои, как правило, вместе с сетью. С одной стороны, объяснимо желание снизить размерность вектора, чтобы сократить вычислительную сложность. Однако, если рассмотреть структуру LSTM сети подробно, то будет видно, что входные в LSTM данные используются в функциях вида $f(Wx_t + \theta)$, где f — сигмоида или гиперболический тангенс, W — матрица весов, а θ некоторая добавка, константная относительно x_t. Таким образом, получается, что мы сразу обучаем комбинацию весов, а не 2 матрицы отдельно. Это ускоряет оптимизацию с точки зрения вычислений на одном шаге, а так же делает задачу менее неопределенной.

4.2 Pooling

Обычно, pooling подразумевает взятие поэлементного максимума или среднего по всему документу сразу. Регионом будем называть фиксированное число слов, стоящих рядом. Авторы статьи предлагают вместо взятия максимума/среднего по всему тексту, зафиксировать число регионов и использовать max-pooling по каждому региону. Таким образом мы получим вектор фиксированной длины, который можно подавать на вход вышестоящей модели.

4.3 Удаление input/output gates

Авторы статьи обнаружили, что удаление input/output gates, при данной архитектуре, не ведет к потере точности. В свою очередь, это дает возможность значительно уменьшить требования к объему используемой памяти и объемы вычислений. Интуитивно, кажется, что применение операции max-pooling частично берет на себя output gate, которая состоит в предотвращение попадания не релевантной информации в

h_t. Принимая во внимания все изменения, формулы для полученной ячейки можно переписать в следующем виде:

$$f_t = \sigma(W^{(f)}x_t + U^{(f)}h_{t-1} + b^{(f)})$$

$$u_t = \tanh(W^{(u)}x_t + U^{(u)}h_{t-1} + b^{(u)})$$

$$c_t = u_t + f_t \odot c_{t-1}$$

$$h_t = tanh(c_t)$$

5 Использование дополнительных представлений слов и документов в LSTM сетях

Еще одна модификация LSTM сети, которую хотелось бы попробовать — это внесение дополнительной информации, в виде представлений слов и документов в сеть. Все представления слов и документов можно так или иначе использовать для улучшения качества работы сети. Например, можно дополнительно вносить информацию о конкретном слове, используя его skip-gram представление. При этом, по аналогии с моей курсовой работой, где было предложено использовать заглавные слова для улучшения качества, здесь так же можно использовать представление документа целиком. Но вместо заголовков, которые есть далеко не в каждом документе, можно использовать, например, векторы, полученные с помощью тематического моделирования. Таким образом, если мы обозначим за d — представление рассматриваемого документа, а за r_t — представление слова, которое идет в рассматриваемом документе на позиции t, то формулы из предыдущего раздела можно дополнить следующем образом:

$$f_t = \sigma(W^{(f)}x_t + U^{(f)}h_{t-1} + V^{(f)}r_t + Q^{(f)}d + b^{(f)})$$

$$u_t = \tanh(W^{(u)}x_t + U^{(u)}h_{t-1} + V^{(u)}r_t + Q^{(u)}d + b^{(u)})$$

$$c_t = u_t + f_t \odot c_{t-1}$$

$$h_t = tanh(c_t)$$

6 Эксперименты

Для экспериментов использовались два набора данных. Первый — 20 News Groups(20NG). Он содержит около восемнадцати тысяч документов, разделенных на двадцать групп. Второй набор данных — это пятьдесят тысяч отзывов к фильмам(IMDB), разделенных на позитивные и негативные. В тренировочной и тестовой выборках по двадцать пять тысяч документов. Во всех экспериментах текст приведет к нижнему регистру, удалены не литеральные символы, отброшены самые редко и часто встречающиеся слова.

6.1 Эксперименты с классификацией на исходных признаках

Для начала возьмем рассматриваемые представления слов(w2v, adagram) и получим из них представления документов с помощью пулинга, описанного в секции 4.2. Будем использовать $k-max$ пулинг, $k = 10$ для 20NG, $k = 1$, для IMDB. Кроме того, для сравнения возьмем tf-idf представление документа. В качестве классификатора будем использовать логистическую регрессию. Все параметры каждой из моделей, включая размер векторов в w2v и adagram, были подобраны на основание кросс-валидации, а финальный результат измерен на тестовой выборке. Adagram и w2v обучались на тех же выборках, классификация которых проводилась.

Ожидаемо, tf-idf представление показало лучший результат. В конце концов, размер признаковых векторов там на порядок выше, чем в остальных моделях. В adagram модели, обученной на 20NG было очень мало многозначных слов, даже слово Apple, которое обычно приводится в качестве примера, имело всего одно значение(При оптимальном, с точки зрения классификации, параметре α). Вероятно, поэтому w2v по качеству оказался впереди.

Таблица 1: Классификация. Исходные признаки. Доля правильных ответов.

Обозначения:

tf-idf + LR — Лог. регрессия на tf-idf представление.

w2v + LR — Лог. регрессия построенная на w2v представление документа

adagram + LR — Лог. регрессия, построенная на adagram представление документа

w2v + xgboost — xgboost, построенный на w2v представление

adagram + xgboost — xgboost, построенный на w2v представление

Модель	20NG	IMDB
tf-idf + LR	83.13 %	89.94 %
w2v + LR	77.31 %	85.97 %
adagram + LR	77.15 %	86.20 %
w2w + xgboost	77.1 %	86.01 %
adagram + xgboost	77.34 %	86.06 %

6.2 Эксперименты с LSTM сетью

Здесь, в качестве basline варианта будем рассматривать простую сеть из 4 и будем сравнивать ее с модификациями из 5. Кроме того, возьмем полную двунаправленную LSTM сеть, чтобы убедится, что разница в качестве с усеченной версией является незначительной. Аналогично предыдущему разделу, пропустим выходы h_t сети через k-max пулинг, описанный в 4.2. Параметры пулинга k аналогичны пункту с базовой классификацией 6.1. В качестве r_t используются w2v или adagram представления. В качестве d брались те же представления документов, что и для классификации. Так же, дополнительно возьмем результат добавления вектора, полученного из LSTM к tf-idf представления и запустим на этом логистическую регрессию. См. Таблицу 2

Получилось, что adagram и w2v представления, с точки зрения LSTM признаков, оказались почти эквивалентными. Кроме того, видим, что информация, привнесенная дополнительными представлениями в сеть, достаточно существенно подняла точность классификации. При этом, когда в качестве дополнительного описания до-

Таблица 2: Классификация с LSTM признаками. Доля правильных ответов.

Обозначения:

tf-idf + LR — Лог. регрессия на tf-idf представление.

2LSTM — двунаправленная LSTM сеть с усеченным набором весов.

full-2LSTM — двунаправленная LSTM сесть с полным набором весов.

w2v-2LSTM — Двунаправленная LSTM сеть с усеченным набором весов и дополнительным представлением слов в виде w2v

adagram-2LSTM — Двунаправленная LSTM сеть с усеченным набором весов и дополнительным представлением слов в виде adagram

w2v-TM-2LSTM — Двунаправленная LSTM сеть с усеченным набором весов и дополнительным представлением слов в виде w2v, а так же дополнительным представлением слов через тематическое моделирование

w2v-2LSTM+tf-idf+LR — Логистическая регрессия, в качестве признаков которой подается конкатенация tf-idf представления и выходах h_t LSTM сети, пропущенных через k-max пулинг.

Модель	20NG	IMDB
tf-idf + LR	83.13 %	89.94 %
2LSTM	80.13 %	85.98 %
full-2LSTM	80.21 %	85.91 %
w2v-2LSTM	81.34 %	86.90 %
adagram-2LSTM	81.28 %	86.51 %
w2v-TM-2LSTM	81.55 %	87.32
w2v-2LSTM+tf-idf+LR	83.50 %	89.97 %

кумента использовалось представление из тематических моделей, качестве оказалось выше, чем при использование простого пулинга w2v представления. Вероятно, дело в том, что когда мы делаем пулинг w2v представления и подаем его в качестве дополнительного описания документа, мы частично дублируем информацию, которая имеется в w2v представление слов. Кроме того, признаки, извлеченные из LSTM сети и поданные вместе с tf-idf подняли качество классификации на обеих выборках. Полная версия LSTM сети действительно не показала значительного прироста, при

том, что время на ее обучение уходило в несколько раз больше, чем на обучение усеченной.

Так же, если вернуться к классической задаче LSTM сети, а именно, предсказания следующего слова, то есть смысл сравнить перплексии, получаемые при обучение сетей с различными конфигурациями. См. Таблицу 3

Таблица 3: Предсказание следующего слова. Перплексия на валидационной выборке(После 10 тысяч батчей).
Обозначения:
tf-idf + LR — Лог. регрессия на tf-idf представление
2LSTM — двунаправленная LSTM сеть с усеченным набором весов (4.3)
full-2LSTM — двунаправленная LSTM сесть с полным набором весов (4)
w2v-2LSTM — Двунаправленная LSTM сеть с усеченным набором весов и дополнительным представлением слов в виде w2v (5)
adagram-2LSTM — Двунаправленная LSTM сеть с усеченным набором весов и дополнительным представлением слов в виде adagram
w2v-TM-2LSTM — Двунаправленная LSTM сеть с усеченным набором весов и дополнительным представлением слов в виде w2v, а так же дополнительным представлением слов через тематическое моделирование

Модель	20NG	IMDB
2LSTM	190.3	201.3
full-2LSTM	182.4	190.8
w2v-2LSTM	183.9	193.9
adagram-2LSTM	184.3	192.5
w2v-TM-2LSTM	183.1	192.6

Здесь результаты для усеченных сетей оказались хуже, однако следует понимать, что усеченная архитектура вводилась именно для задач классификации. Интуиция состоит в том, что необходимость в output gate пропадает из-за того, что max-pooling не дает возможности не релевантной информации пройти за пределы ячейки. Поэтому логично, что в задаче прогнозирования следующего слова, полная архитектура

показала лучший результат. Представление документа из тематического моделирования опять немного улучшило качество.

6.3 Выводы

В итоге, оказалось, что Логистическая регрессия в паре с tf-idf представлением показывает лучшие результаты в качестве одиночного алгоритма. Однако, признаки, полученные из LSTM сети могут быть полезны для улучшения качества классификации. Кроме того, если LSTM сеть используется не для классификации, а для предсказания следующего слова, то есть смысл использовать дополнительные представления слов и документов, для улучшения качества. Так же, если время обучения и предсказания имеет значение, то можно использовать усеченную версию LSTM сети и качество сильно падать не будет.

Список литературы

[1] Supervised and Semi-Supervised Text Categorization using LSTM for Region Embeddings, Rie Johnson, Tong Zhang
https://arxiv.org/abs/1602.02373

[2] Efficient Estimation of Word Representations in Vector Space, Tomas Mikolov, Kai Chen, Greg Corrado, Jeffrey Dean
https://arxiv.org/abs/1301.3781

[3] Breaking Sticks and Ambiguities with Adaptive Skip-gram, Sergey Bartunov, Dmitry Kondrashkin, Anton Osokin, Dmitry Vetrov
https://arxiv.org/abs/1502.07257

[4] Вероятностное тематическое моделирование, К. В. Воронцов, 2013
http://www.machinelearning.ru/wiki/images/2/22/Voron-2013-ptm.pdf

[5] LONG SHORT-TERM MEMORY, Sepp Hochreiter, Jurgen Schmidhuber
http://deeplearning.cs.cmu.edu/pdfs/Hochreiter97_lstm.pdf

[6] Understanding LSTM Networks
http://colah.github.io/posts/2015-08-Understanding-LSTMs/

[7] The Unreasonable Effectiveness of Recurent Neural Networks
http://karpathy.github.io/2015/05/21/rnn-effectiveness/

[8] Ferguson, T. S. A Bayesian analysis of some nonparametric problems. The Annals of Statistics, 1(2):209–230, 1973.
https://projecteuclid.org/download/pdf_1/euclid.aos/1176342360