**Содержание**

[Постановка задачи 3](#_Toc10633591)

[Введение 4](#_Toc10633592)

[Векторное представление слов 5](#_Toc10633593)

[Классический подход 5](#_Toc10633594)

[Нейросетевой подход 7](#_Toc10633595)

[Первичная обработка данных 10](#_Toc10633596)

[Рекуррентные нейронные сети. LSTM 11](#_Toc10633597)

[Выделение векторного представления слов из модели 15](#_Toc10633598)

[Заключение 17](#_Toc10633599)

[Список используемой литературы 18](#_Toc10633600)

# Постановка задачи

Изучить способ получения векторного представления слов при помощи рекуррентных нейронных сетей. Описать способ предобработки, архитектору рекуррентной нейронной сети, ее разновидностей и их преимущества, также получение векторного представления данных из нее. Сделать обзор литературы.

# Введение

Рекуррентные нейронные сети в наше время – неотъемлемый инструмент для работы с естественным языком, а так же просто с последовательностями любого рода, будь то последовательности звуков (задачи распознания речи или текста), последовательности изображений (распознание объектов на видео) и т.д.

Также актуальны в наше время векторные представления данных, особенно, слов. Само по себе векторное представление (embedding) это сопоставление некоторой произвольной сущности некоторому вектору определенной размерности. Существует множество методов создания векторных преставлений, от обычного one-hot-encoding, до более сложных нейросетевых моделей, учитывающих контекст (подробнее в разделе про векторные представления). Но даже их бывает недостаточно. Например, для случаев, когда в векторном представлении не важен контекст и метрика (любое векторное представление образует метрическое пространство, на котором можно ввести метрику – расстояние между двумя любыми элементами множества) должна отражать близость не по семантике, а по любому другому критерию.

В данной работе рассмотрен способ получения векторного представления слов из рекуррентной нейронной сети (LSTM), преимущество которого в том, что этот способ адаптируется к любым задачам с помощью составления функции потерь и подбору данных.

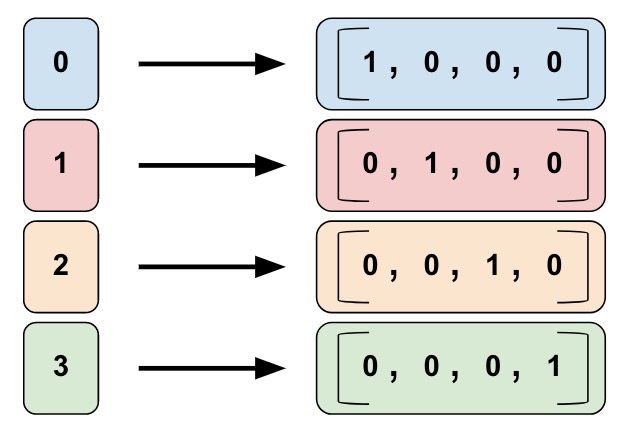
Описанный в работе подход будет применяться в дипломной работе.

# Векторное представление слов

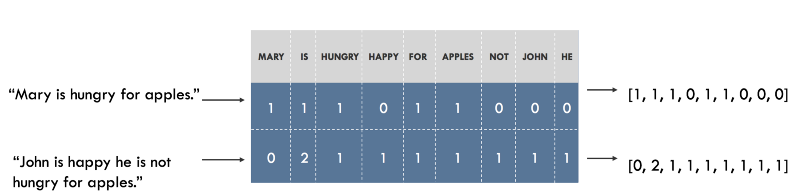
Векторное представление – подход к моделированию языка и дальнейшему обучению представлений в обработке естественного языка, суть которого в подборе оптимального метрического пространства и его размерности для сопоставления векторов этого пространства словам (фразам, буквам и т.д.) по заданной на пространстве метрикой.

## Классический подход

Самый тривиальный подход для представления – one-hot-encoding, суть которого заключается в создании векторного пространства размерности, равной мощности словаря (то есть количеству объектов, которым нужно сопоставить вектор). Каждый вектор этого пространства заполнен нулями за исключением позиции, которая совпадает с порядковым номером векторизуемого объекта (рис. 1).

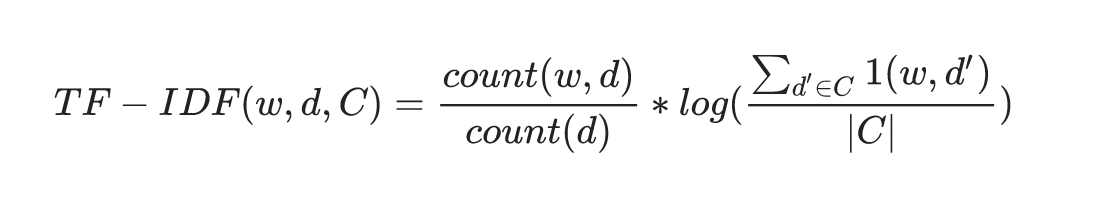


*Рис. 1. One-hot-encoding*

Значение одного слова по сути не так важно, так как речь состоит из наборов слов, которые мы называем текстами. Таким образом, можно взять one-hot-encoding вектор каждого слова в тексте и сложить вместе. То есть на выходе получится просто подсчет количества различных слов в тексте в одном векторе. Такой подход называется “мешок слов” (bag of words), потому что теряется вся информация о взаимном расположении слов внутри текста.

*Рис. 2. Bag of words*

Следующий этап - представление корпуса (набор текстов) в виде матрицы “слово-документ” (term-document). Стоит отметить, что в области информационного поиска (information retrieval) эта матрица носит название "обратного индекса" (inverted index), в том смысле, что обычный/прямой индекс выглядит как "документ-слово" и очень неудобен для быстрого поиска. Другими словами, это представление слов через документы или же просто транспонированная матрица bag of words.

Последний этап “обычного” подхода называется TF-IDF (term frequency – inverse document frequency) - статистическая мера, используемая для оценки важности слова в контексте документа, являющегося частью коллекции документов или корпуса. Вес некоторого слова пропорционален частоте употребления этого слова в документе и обратно пропорционален частоте употребления слова во всех документах коллекции. И, естественно, tf-idf вектора используются как векторные представления.

*Рис. 3. Формула tf-idf*

TF — отношение числа вхождений некоторого слова к общему числу слов документа. Таким образом, оценивается важность слова t i {\displaystyle t\_{i}} в пределах отдельного документа.

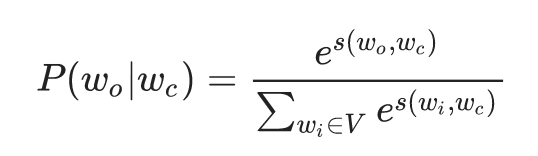
IDF - инверсия частоты, с которой некоторое слово встречается в документах коллекции. Основоположником данной концепции является Карен Спарк Джонс. Учёт IDF уменьшает вес широкоупотребительных слов. Для каждого уникального слова в пределах конкретной коллекции документов существует только одно значение IDF.

Существует также множество формул, основанных на tf-idf, являющихся улучшением этой формулы или просто изменением под конкретную задачу. Наиболее популярная формула – BM25.

## Нейросетевой подход

Описанные выше подходы были хороши для своего времени, хотя некоторые из них все еще применяются как вспомогательные методы современных подходов. Например, one-hot-encoding и bag of words применяется в end2end как простейший вариант представления каких-то сущностей (чаще букв в случае ohe и документов в случае bow), а tf-idf, которая, другими словами, является вектором важности слов внутри предложения (документа) выступает в роли набора числовых коэффициентов, использующихся при усреднении векторов более сложного представления, где одному слову уже сопоставляется вектор, а не скаляр.

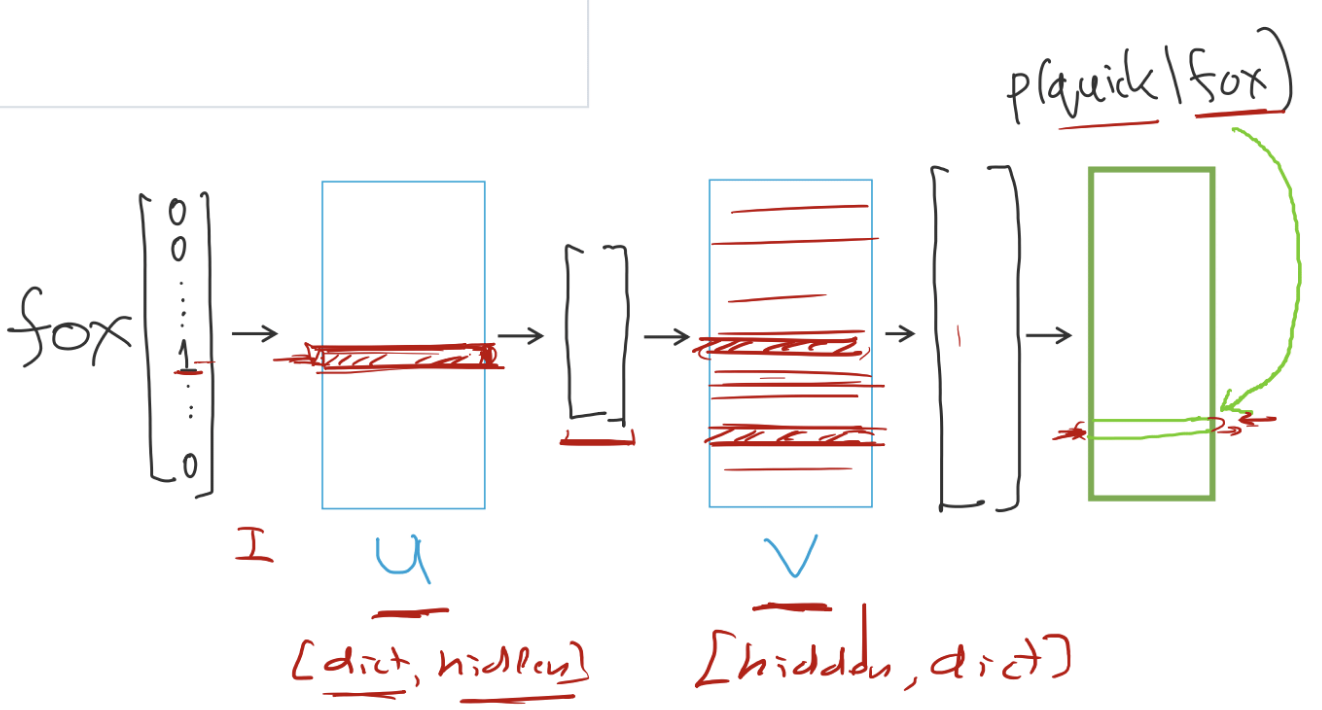
С приходом интернета и увеличением мощностей компьютеров появилась возможность “обучать” представления на больших объёмах данных. Так, благодаря гипотезе: “слова, которые встречаются в одинаковых окружениях, имеют близкие значения” появился новый подход под названием word2vec. Модель очень простая - предсказываются вероятности слов по их окружению (контексту). То есть нужно подобрать такие вектора слов, чтобы вероятность, присваиваемая моделью слову была близка к вероятности встретить это слово в этом окружении в реальном тексте.



*Рис. 4. Формула вероятности в word2vec*

Здесь — вектор целевого слова, — это некоторый вектор контекста, вычисленный (например, путем усреднения) из векторов окружающих нужное слово других слов. А — это метрика, вводимая на получившемся пространстве.

Приведенная формула называется softmax (“мягкий максимум”, мягкий — в смысле дифференцируемый). Это нужно для того, чтобы модель могла обучиться с помощью процесса обратного распространения ошибки.

Процесс тренировки устроен следующим образом: берется последовательно (2k+1) слов, слово в центре является тем, которое должно быть предсказано. А окружающие слова являются контекстом длины по k с каждой стороны. Каждому слову в модели сопоставлен уникальный вектор, который улучшается в процессе обучения модели.

*Рис. 5. Архитектура нейронной сети для тренировки word2vec*

На рис. 5 показана архитектура нейронной сети, использующаяся для тренировки word2vec, где на выходе полезны 2 слоя, обозначенные как U и V. Первый отвечает за контекст других слов относительно заданного в i-й строке (где i – номер ненулевой строки во входном one-hot транспонированном векторе), а второй является контекстом этого слова относительно других. Для конечного преображения слоев в представление можно использовать как только слой U, так и конкатенацию/усреднение векторов из слоев U и V.

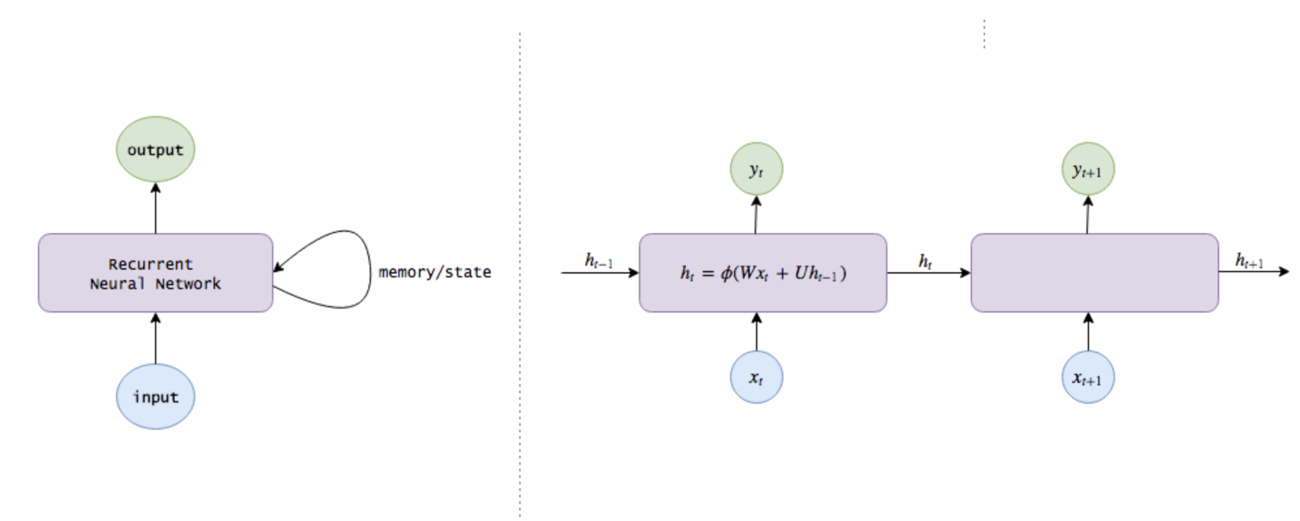
В целом, этот подход называется CBOW — continuous bag of words, continuous потому, что на вход модели подаются последовательно наборы слов из текста, a BoW потому что порядок слов в контексте не важен.

Есть также множество аналогов и улучшений этой модели, таких, как Negative Sampling, Hierarchical SoftMax, Global Vectors, Fasttext.

# Первичная обработка данных

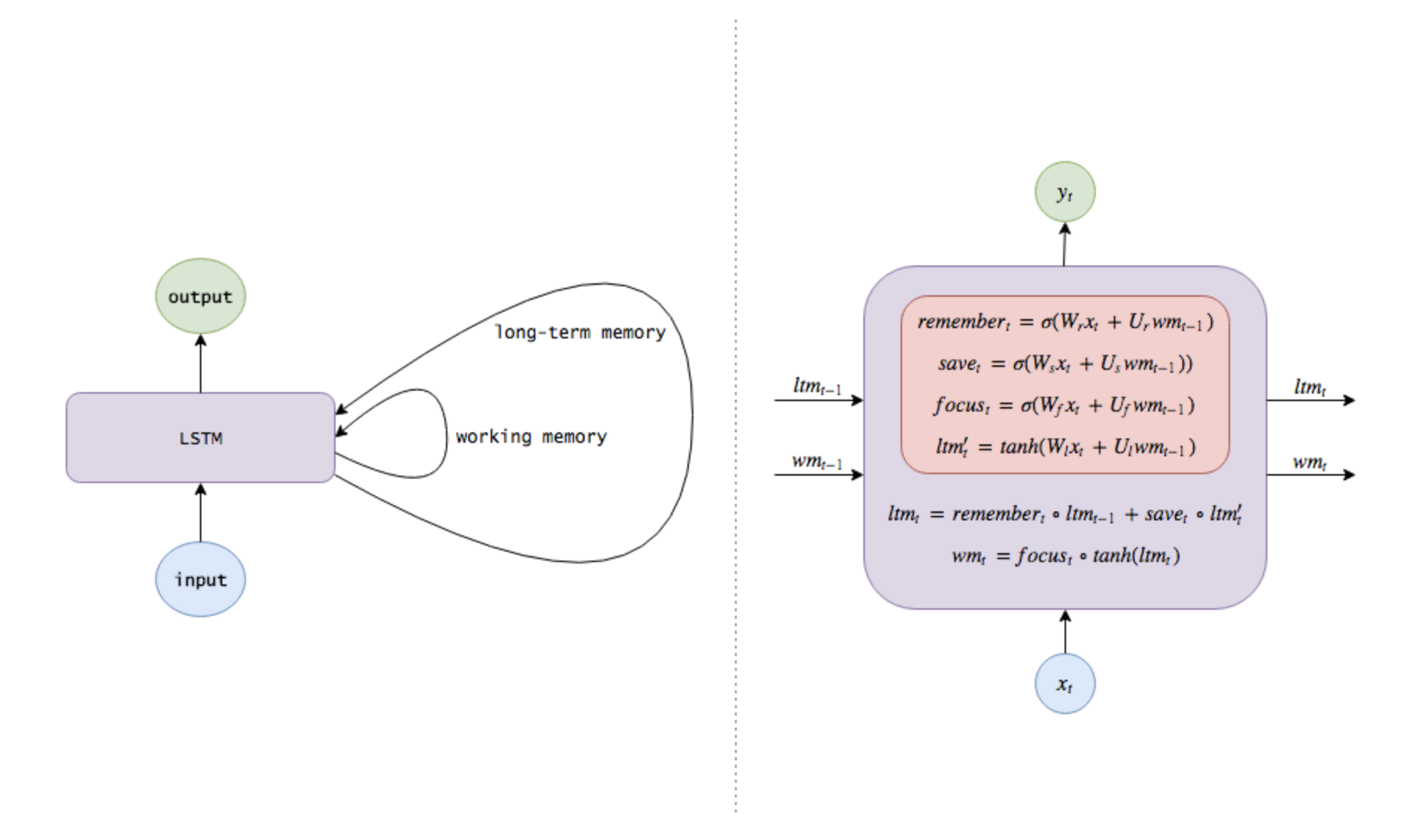
В первичной обработке данных используется простое one-hot-encoding представление букв в связке с первым линейным слоем (который будет усложнять представление в контексте задачи), в следствие двух причин: любой выбор признаков человеком будет уступать автоматическому end2end подходу и мощность множества всех букв небольшая, значит, можно не усложнять модель, уменьшая размерность представления.

# Рекуррентные нейронные сети. LSTM

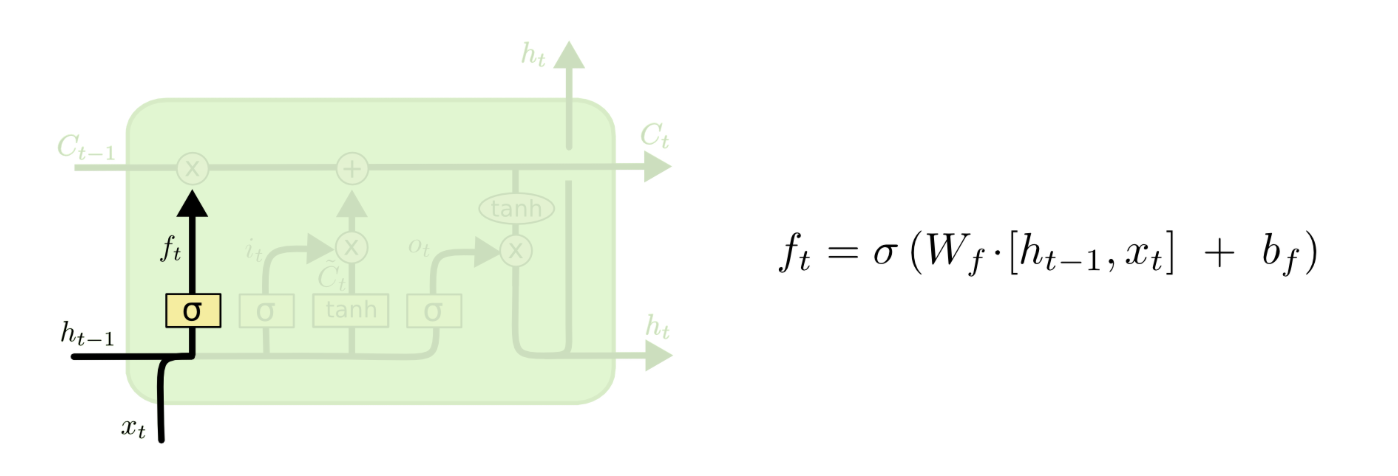
Поскольку традиционные нейронные сети не способны хранить информацию о предыдущих входах, а идея представления слова в виде связанной последовательности букв является ключевой, было решено перейти к рекуррентным сетям, а именно архитектуре LSTM. Для лучшего понимания РНС можно представить, как множество копий одной и той же нейронной сети, каждая из которых не только получает новый вход, но и передает информацию выхода последующей. Структура, имеющая вид цепочки, говорит о том, что РНС тесно связаны с последовательностями. За последние несколько лет РНС с большим успехом были использованы для решения различных задач, таких как распознавание речи, моделирование языка, перевод, создание описаний к изображениям и т.д. Большая часть успехов была достигнута с помощью особого типа РНС, называемого LSTM-сетью (long short-term memory, долгая кратковременная память), который при решении задач с более длинными последовательностями значительно превосходит стандартный вариант. LSTM-сеть представляет собой цепочку аналогичную РНС, но повторяющийся модуль имеет другую структуру. Вместо одного слоя он содержит четыре слоя, которые взаимодействуют особым образом.

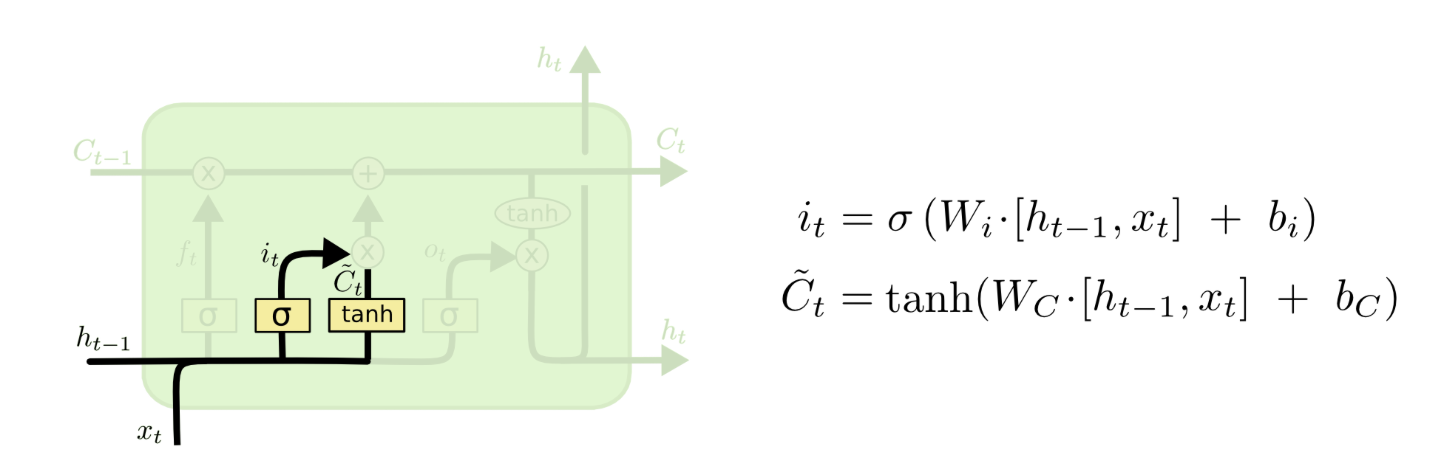
*Рис. 6. Архитектура слоя рекуррентной нейронной сети.*

Проблема РНС в том, что они не справляются с “долговременнымизависимостями”, и в обратном распространении с большой вероятностью происходит затухание градиента. Однако, LSTM-сети лишены такого недостатка. LSTM-сеть – это особый тип рекуррентных нейронных сетей, которые могут обучаться долговременным зависимостям. Такие сети прекрасно решают множество современных задач и активно применяются в данное время. LSTM-сети разработаны специально для того, чтобы решить проблему долговременных зависимостей. Их нейроны хорошо «помнят» недавно полученную информацию, но не имеют возможности надолго сохранить в памяти что-то, что обработали много циклов назад, какой бы важной та информация ни была.

Одной из главный частей LSTM сети является состояние ячейки. В этих состояниях как раз и запоминается информация из предыдущих шагов обучения. При необходимости LSTM может удалять информацию из состояния ячеек, этот механизм регулируется с помощью специальных фильтров, которые представляют из себя сигмоиданый слой нейронной сети и поточечное умножение. То есть каждый сигмоидальный слой решает какую долю информации с очередного слоя сохранить и отправить дальше по сети. Всего в LSTM три таких слоя, которые контролируют долю информации, передаваемой по сети.

*Рис. 7. Архитектура LSTM*

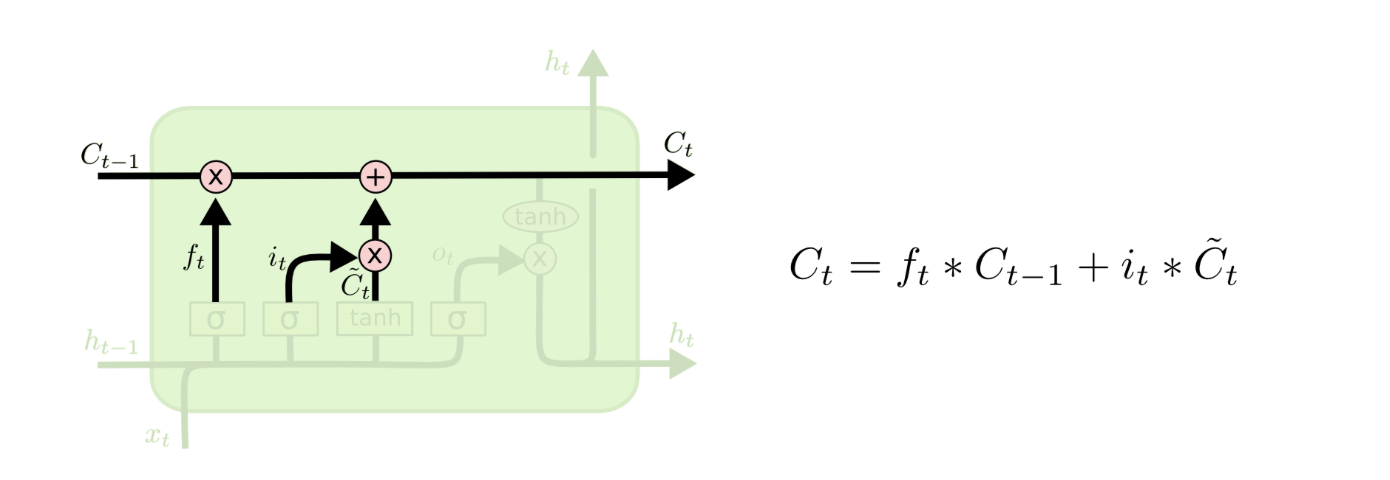
*Рис. 8. Первый шаг LSTM*

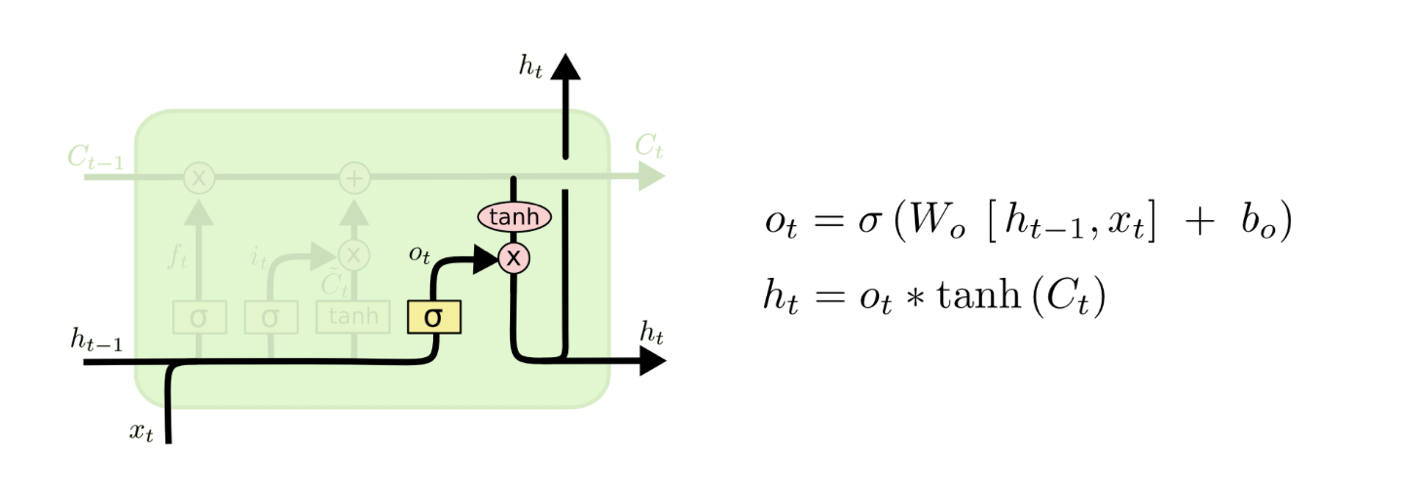
Следующий шаг – определить, какая новая информация должна храниться в состоянии ячейки. Этот этап состоит из двух частей. Сначала сигмоидальный слой под названием “слой входного фильтра” (input layer gate) определяет, какие значения следует обновить. Затем tanh-слой строит вектор новых значений-кандидатов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

*Рис. 9. Второй шаг LSTM*

Далее старое значение ячейки Ct-1  необходимо заменить на новое значение Ct.

Для этого необходимо умножить старое значение Ct- на ft, при этом отбрасывая ненужную информацию. Затем к полученному значению прибавляем  . Так мы получи новые значения, умноженные на  t, которые показывает долю обновления значений состояния.

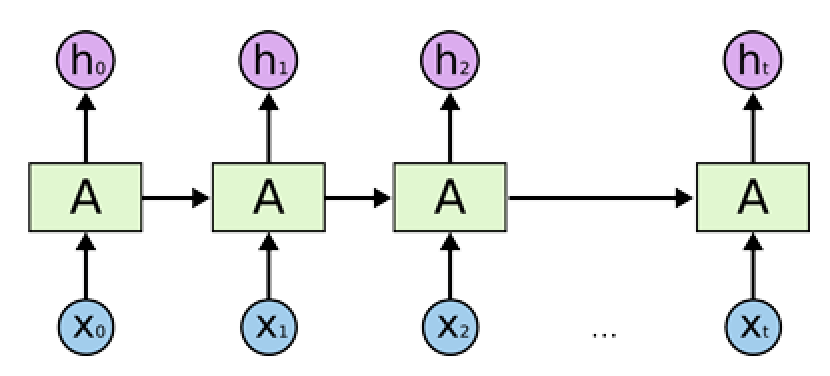
*Рис. 10. Третий шаг LSTM*

Теперь необходимо решить, какую информацию мы хотим получать на выходе. Выходные данные будут зависеть от состояния ячейки, к ним будут применены некоторые фильтры. Сначала мы применяем сигмоидальный слой, он решает, какую информацию из состояния ячейки 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.

*Рис. 11. Четвертый шаг LSTM*

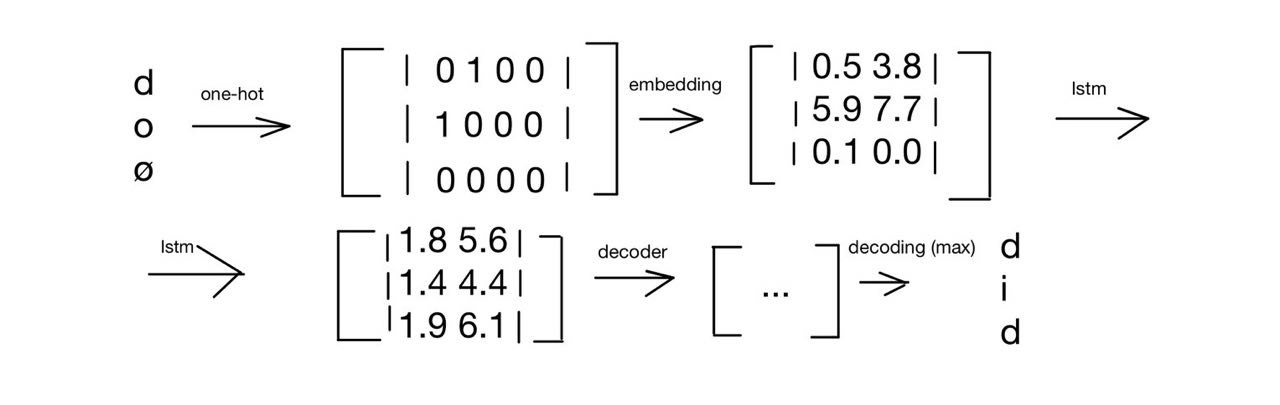
# Выделение векторного представления слов из модели

Для понимания принципа выделения векторного представления из LSTM нужно разобраться с форматом и размерностями выходных векторов.

 *Рис. 12. Пример архитектуры rnn “many to many”*

В этой работе будет рассмотрена архитектура many to many, которая от других отличается тем, что для каждого элемента входной последовательности формируется собственный выход. Так, для каждого входного вектора, описывающего букву (последовательность которых является словом) формируется вектор новой буквы. Таким образом модель является оператором, который переводит пространство слов в себя (или в другое пространство слов). В этом подходе есть проблема – размер слов на входе и выходе не обязан совпадать. Проблема решается добавлением в словарь (множество) букв “нулевого” символа и поиска слова максимальной длины с последующим заполнением “пустых” входов последовательности векторными представлениями нулевого символа.

Далее, первым слоем в нейронной сети будет слой embedding (практически обычный линейный слой), задача которого – преобразовать one-hot вектора в признаки (вектора в поле рациональных чисел с новой размерностью – гиперпараметр модели) и подать на вход LSTM слою.

В many to many архитектуре на выходе LSTM слоя будет тензор того же формата, но можно менять размерность выходных элементов последовательности (размерность скрытого слоя – один из гиперпараметров).

*Рис. 13. Пример работы модели*

В примере на рис. 13 размерность embedding слоя равна размерности скрытого слоя. К исходной размерности тензор приходит в результате работы декодера (в данном случае – это обычный линейный слой). Далее, либо в результате бинаризации, либо в результате поиска максимума по строке векторы декодируются в буквы.

Гипотетически, для векторного представления можно брать веса любого слоя, но лучшие результаты достигаются с весами из LSTM, что вполне логично, учитывая, что слои до LSTM по сути готовят батч (в случае на рис. 13 он единичный) к “проходу” по LSTM, а слои после отвечают за “развертку” тензора к исходной размерности (что не обязательно при построении классификатора).

Последний вопрос состоит в том, какой вектор брать для представления слова. Есть разные подходы, но чаще всего берется либо последний вектор (в двусторонней LSTM можно брать также и первый), либо усредняются вектора по всем буквам. Так, для рис. 13, слово “do” можно представить либо в виде вектора , либо в виде .

# Заключение

В работе были рассмотрены разные методы векторного представления слов, описана архитектура рекуррентной сети LSTM и подход к выделению векторного представления из LSTM. В дальнейшем этот подход будет использоваться в дипломной работе.

# Список используемой литературы

1. Madrugado. Чудесный мир Word Embeddings: какие они бывают и зачем нужны? <<https://habr.com/ru/company/ods/blog/329410/>>, 25.05.2019.
2. Christopher Olah. Understanding LSTM networks. <<http://colah.github.io/posts/2015-08-Understanding-LSTMs/>>, 27.05.2019.
3. Семен Козлов. Введение в NLP, word2vec.

<<https://www.dropbox.com/s/na7lpz9xhgx8gp1/Lecture%209%20-%20Intro%20to%20NLP%20-%20annotated.pdf?dl=0>>, 20.04.2019.