МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ

РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное

образовательное учреждение высшего образования

«Самарский национальный исследовательский университет

имени академика С.П. Королева»

Институт информатики и кибернетики

Факультет информатики

Кафедра программных систем

ОТЧЁТ

по лабораторной работе № 3

«Анализ тональности (sentiment analysis)»

по курсу «Нейронные сети»

Выполнил:

Портнов К.А. Хорина В.И.

гр. 6132-020402D

Проверила:

Жданова А.Н.

Самара 2022

Оглавление

[Математическая постановка задачи 3](#_Toc122517311)

[Обоснование выбора нейронной сети 3](#_Toc122517312)

[Описание архитектуры сети, параметров, метода обучения 4](#_Toc122517313)

[Структурная схема алгоритма 6](#_Toc122517314)

[Описание вычислительных экспериментов с целью оценки эффективности модели 7](#_Toc122517315)

[Результаты работы 9](#_Toc122517316)

[Заключение 9](#_Toc122517317)

[Листинг программы 10](#_Toc122517318)

# Математическая постановка задачи

Требуется реализовать нейронную сеть, позволяющую определить по тексту его тональность, то есть позитивное ли отношение несет этот текст или негативное.

Набор данных содержит 226834 строчек данных взятых и основан на сообщениях российского сегмента twitter.

Описание данных:

1. id: уникальный номер сообщения в системе twitter;
2. tdate: дата публикации сообщения (твита);
3. tmane: имя пользователя, опубликовавшего сообщение;
4. ttext: текст сообщения (твита);
5. ttype: поле в котором в дальнейшем будет указано к кому классу относится твит (положительный, отрицательный, нейтральный);
6. trep: количество реплаев к данному сообщению. В настоящий момент API твиттера не отдает эту информацию;
7. tfav: число сколько раз данное сообщение было добавлено в избранное другими пользователями;
8. tstcount: число всех сообщений пользователя в сети twitter;
9. tfol: количество фоловеров пользователя (тех людей, которые читают пользователя);
10. tfrien: количество друзей пользователя (те люди, которых читает пользователь);
11. listcount: количество листов-подписок в которые добавлен твиттер-пользователь.

# Обоснование выбора нейронной сети

Обычно другие типы нейронных сетей, такие как плотно связанные сети или сверточные сети, не имеют памяти, это означает, что каждый вход обрабатывается независимо, не имея никакого отношения к другим. Это противоположно тому, что люди обычно делают при чтении текста, а именно сохранение в памяти того, что было прочитано в предыдущих строках. В следствии чего появляется ощущение всего смысла теста, и это точно такой же принцип, который принимают RNN. RNN обрабатывают последовательности, итерируя вдоль элементов последовательности и сохраняя информацию относительно того, что она обрабатывала до сих пор.

Рекуррентные нейронные сети (RNN) - это класс нейронных сетей, который является мощным для моделирования последовательности данных, таких как временные ряды или естественный язык.

Схематически уровень RNN использует for цикл для перебора временных шагов последовательности, сохраняя при этом внутреннее состояние, которое кодирует информацию о временных шагах, которые он видел до сих пор.

# Описание архитектуры сети, параметров, метода обучения

Для выполнения работы использованы:

API Keras RNN для создания и обучения сети.

API Keras RNN разработан с акцентом на:

* Простота использования;
* Простота настройки.

Nltk – платформа для работы с данными на человеческом языке с набором библиотек обработки текста для классификации, токенизации, выделения, тегов, синтаксического анализа и т.д.

Gensim – библиотека для представления документов в виде семантических векторов

Pandas – для работы с dataFrame и т.д.

И прочие библиотеки python.

Была реализована рекуррентная нейронная сеть типа двунаправленная LSTM.

Слой встраивания – Embedding с размерность и количеством токенов.

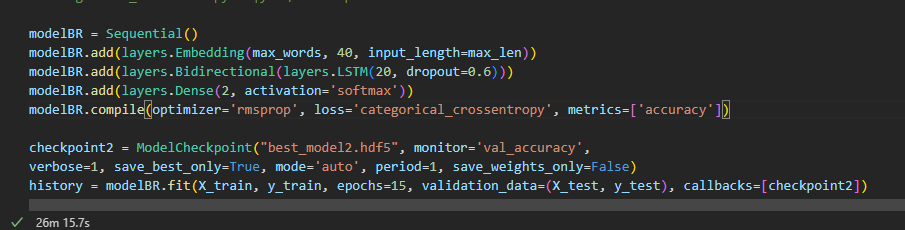
Двунаправленный слой с LSTM с 20 скрытыми единицами внутри слоя и 0,6 в качестве Dropout для регуляризации сети.

Выходной слой – Densy с 2 выходам (2 категории) и softmax в качестве функции активации.

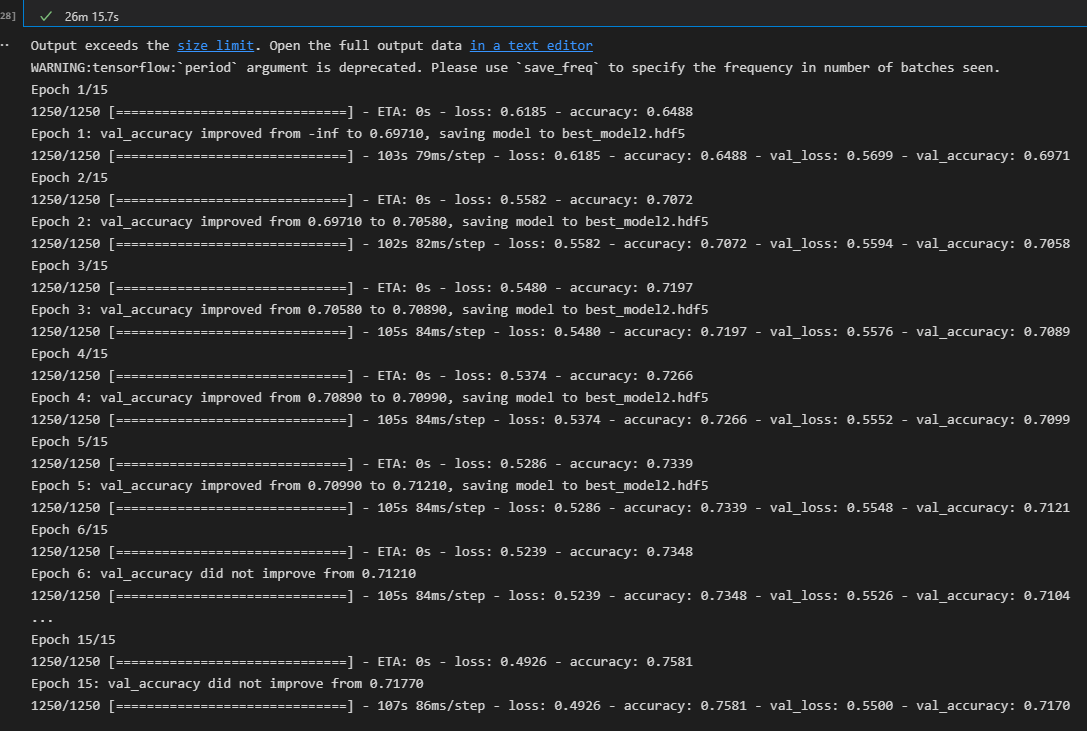
В качестве оптимизатора использован алгоритм RMSprop и categorical\_crossentropy как функция потерь.

Обучение проводилось на 15 эпохах с сохранением состояния.

Сама модель выглядит следующим образом:



И лог обучения:



# Структурная схема алгоритма

Выполнение работы начинается с импортирования всех необходимых библиотек в проект.

Импортируем датасет и превращаем его в DataFrame.

Для более быстрого обучения сети, обрезаем выборку по 25000 строк каждой категории сообщений.

Т.к. в выборке гораздо больше полей, чем нам нужно в работе, обрезаем только по необходимым полям.

Все пустые сообщения заменяем на заглушку “пусто”.

В цикле проводим очистку текста:

* Удаляем URL
* Удаляем Email
* Удаляем символы новой строки
* Удаляем одинарные кавычки

Преобразуем строки в список токенов.

Удаляем повторяющиеся слова из набора данных, следуя идее, что самые короткие и длинные обычно бесполезны.

Производим детокенизацию всех предложений – превращаем обратно в строки.

Создаём список лэйблов, состоящий из преобразованных признаков тональности (0 –негативный, 1- позитивный).

Преобразовываем текстовые данные в двумерные данные с плавающей запятой.

Разбиваем набор данных на обучающую и тестовую выборку в соотношении 4:1.

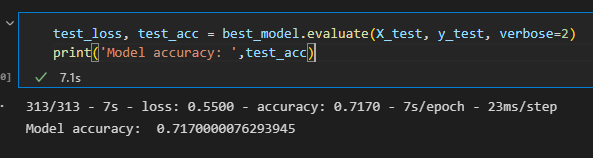
Создаём и обучаем сеть с сохранением состояния.

Запускаем сеть на тестовом наборе данных.

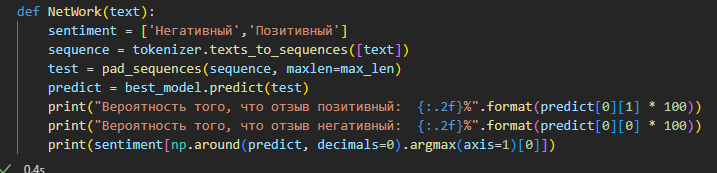
Проводим ручные проверки работы, обученной сети.

# Описание вычислительных экспериментов с целью оценки эффективности модели

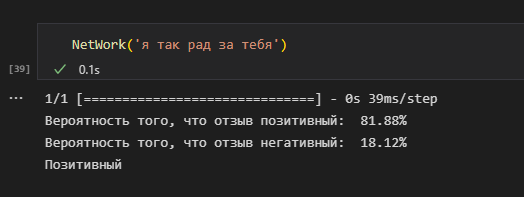
Результаты проверки обученной сети на тестовых данных представленный ниже.

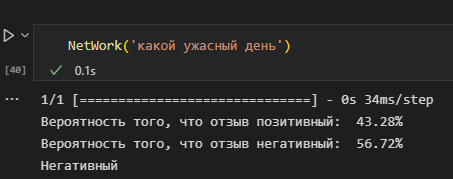


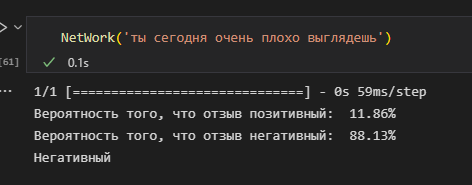
Для проверки и вывода результата прогноза сети реализована функция:

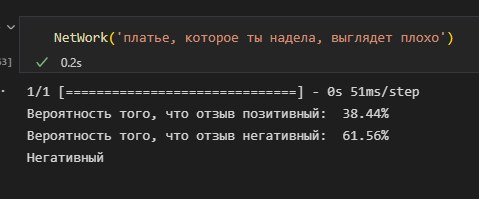


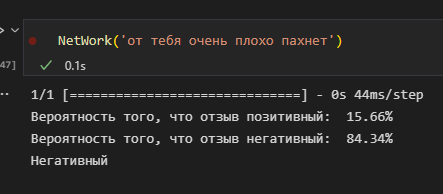
Результаты для некоторых, вручную заданных, фраз:

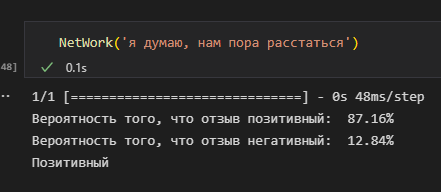


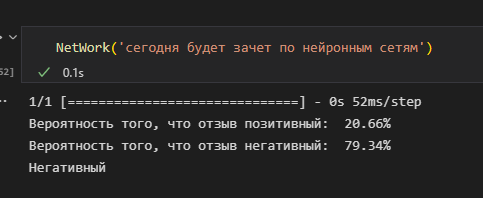




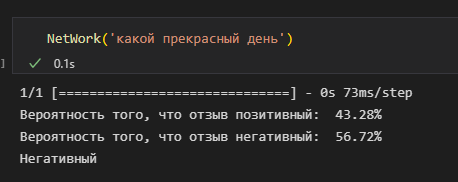


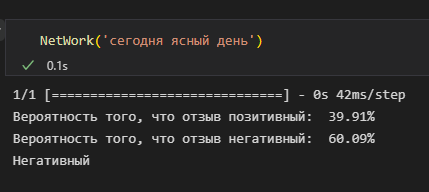






Явно ошибочнее результаты:





# Результаты работы

В результате можно смело заявить, что реализованная сеть вполне успешно справляется с поставленной задачей, хоть и не без ошибок, но увеличение выборки для обучения, увеличение числа эпох, количества слоёв и т.д. могут решить проблему с ошибочными определениями.

Так же, можно заявить, что рекуррентные сети отлично подходят для решения такого типа задач.

# Заключение

В ходе выполнения лабораторной работы была создана и обучена рекуррентная нейронная сеть, позволяющая определить по тексту его тональность, то есть позитивное ли отношение несет этот текст или негативное.

# Листинг программы

