

МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное
образовательное учреждение высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Политехнический институт

Факультет

Кафедра

вычислительной техники
(наименование)

Высшая и прикладная математика
(наименование)

Направление подготовки 01.03.04 «Прикладная математика»
(код и наименование)

Профиль Математическое моделирование в экономике и технике
(код и наименование)

БАКАЛАВРСКАЯ РАБОТА

на тему

ПРИМЕНЕНИЕ НЕЙРОННЫХ СЕТЕЙ К РАСПОЗНАВАНИЮ
НЕГАТИВНОГО ТЕКСТА В СОЦИАЛЬНЫХ СЕТЯХ

Студент 17ВБ1 Устимов 17.06.21 УСТИМОВ В. О.
(группа) (подпись, дата) (фамилия, инициалы)

Руководитель К.Т.Н., доцент Черушева 17.06.21 Черушева Т.В.
(подпись, дата) (фамилия, инициалы)

Нормоконтролер К.Т.Н., доцент Тарасов 17.06.21 Тарасов Д. В.
(подпись, дата) (фамилия, инициалы)

Работа допущена к защите (протокол заседания кафедры от 04.06.2021 № 10)

Заведующий кафедрой д.ф.-м.н., проф. Бойков 17.06.21 Бойков И. В.
(подпись, дата) (фамилия, инициалы)

Работа защищена с отметкой ВММ (протокол заседания ГЭК от 22.06.21 № 9)

Секретарь ГЭК к.пед.н., доцент Зверовщикова 22.06.21 Зверовщикова Н. В.
(подпись, дата) (фамилия, инициалы)

Пенза 2021

МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное
образовательное учреждение высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
Политехнический институт

Факультет

Вычислительной техники
(наименование)

Кафедра

Высшая и прикладная математика
(наименование)

«УТВЕРЖДАЮ»

Заведующий кафедрой

 И. В. Бойков
(подпись, инициалы, фамилия)

« 30 » ноября 2020 г.

ЗАДАНИЕ

на бакалаврскую работу

1. Студент Устимов Вадим Олегович гр. 17ВБ1
(фамилия, имя, отчество)
факультета вычислительной техники

2. Тема работы Применение нейронных сетей к распознаванию нега-
тивного текста в социальных сетях

Тема утверждена приказом по ПГУ № 0903_о от « 30 » ноября 2020 г.

3. Руководитель работы к.т.н., доцент Черушева Т. В.

4. Задание на работу (назначение разработки, исходные данные и т.п.)

1 Изучить разновидность архитектуры рекуррентных нейронных сетей: се-
ти LSTM

2 Рассмотреть бинарную задачу классификации текста

3 Исходные данные: комментарии, твиты, тип – текст

4 Выходные данные: 2 класса

5 Обучить нейронную сеть для решения задачи бинарной классификации
текста

6 На обученной модели провести сравнительную оценку текста в социальных сетях

5. Перечень подлежащих разработке вопросов

1 Рекуррентная нейронная сеть: проблема долгосрочных зависимостей

2 Сети LSTM: слой утраты, слой сохранения, новое состояние

3 Сбор данных: тренировочные и тестовые данные

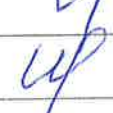
4 Программная реализация модели сети

4.1 Обучение сети

4.2 Проверка предсказаний на отдельных наборах данных

5 Анализ результатов

6. Календарный график по выполнению проекта

Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1. Анализ архитектуры нейронной сети	20 %	28.05.2021	
2. Моделирование рекуррентной нейронной сети	25 %	02.06.2021	
3. Разработка программных модулей	20 %	07.06.2021	
4. Тестирование программы	25 %	10.06.2021	
5. Оформление пояснительной записки	10 %	14.06.2021	

Дата выдачи задания « 30 » ноября 2020 г.

Руководитель бакалаврской работы



Т. В. Черушева

(подпись, инициалы, фамилия)

Задание к исполнению принял студент



В. О. Устимов

(подпись, инициалы, фамилия)

Работу к защите допустить

Декан факультета



Л. Р. Фионова

(подпись, инициалы, фамилия)

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	4
1. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ	6
1.1. Введение в нейронные сети	6
1.2. Математическая постановка задачи	8
1.3. Основные определения	9
1.4. Искусственные нейронные сети	9
1.5. Нейрон	10
1.6. Синапс	11
1.7. Функция активации	11
1.7.1. Ступенчатая функция	12
1.7.2. Линейная функция	12
1.7.3. Сигмоид	13
1.7.4. Гиперболический тангенс	14
1.7.5. Softmax	14
1.7.6. ReLU	15
1.8. Функция потерь	16
1.9. Рекуррентные нейронные сети	16
1.9.1. Полностью рекуррентная сеть	18
1.9.2. Проблема долгосрочных зависимостей	18
1.10. Сети LSTM	20
1.10.1. Слой утраты	22
1.10.2. Слой сохранения	23
1.10.3. Новое состояние	24
2. ПРОГРАММНАЯ РЕАЛИЗАЦИЯ РЕКУРРЕНТНОЙ СЕТИ	26
2.1. Загрузка данных	26
2.2. Предобработка данных	27
2.3. Обработка данных	28
2.4. Построение модели	29

2.5. Компиляция модели.....	29
2.6. Обучение модели	30
2.7. Этап тестирования	30
3. РЕЗУЛЬТАТЫ ИССЛЕДОВАНИЯ.....	32
3.1. Основные характеристики	32
3.2. Описание эксперимента	32
3.3. Исходный набор данных	33
3.4. Результаты экспериментов.....	33
ЗАКЛЮЧЕНИЕ	38
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ.....	40
ПРИЛОЖЕНИЕ А. Программа для загрузки данных	42
ПРИЛОЖЕНИЕ Б. Программа для предобработки данных	44
ПРИЛОЖЕНИЕ В. Программа для создания и тестирования модели	46

ВВЕДЕНИЕ

Все чаще в современном мире ставятся вопросы влияния сетевых ресурсов на развитие личности человека. Данная проблема актуальна и очень многогранна, ведь социальные ресурсы несут в себе большое количество информации, как позитивной, так и негативной. К сожалению, никто не застрахован от негативного контента, отрицательно влияющего на человека.

В России только в одной социальной сети «В контакте» насчитывается 73 миллиона активных пользователей в месяц [1]. Социальные сети дают пользователям возможность обмениваться информацией, общаться и создавать веб-контент.

В настоящее время нельзя недооценивать возможности ИИ. Его возможности возрастают благодаря использованию ИНС.

Искусственные нейронные сети (ИНС) – математические модели, а также их программные или аппаратные реализации, построенные по принципу организации и функционирования биологических нейронных сетей – сетей нервных клеток живого организма. Это понятие возникло при изучении процессов, протекающих в мозге, и при попытке смоделировать эти процессы. Первой такой попыткой были нейронные сети Маккалока и Питтса. После разработки алгоритмов обучения получаемые модели стали использовать в практических целях в задачах прогнозирования, для распознавания образов, в задачах управления и др.

ИНС – это система соединённых и взаимодействующих между собой простых искусственных нейронов (процессоров). Каждый нейрон сети периодически получает сигналы и периодически посылает другим процессорам. И тем не менее, будучи соединёнными в достаточно большую сеть с управляемым взаимодействием, такие локально простые процессоры вместе способны выполнять довольно сложные задачи.

В работе предлагается использовать LSTM [2] сети, которые используются в задачах распознавания текста.

Целью данной работы является построение и реализация ИНС, которая будет распознавать негативный текст.

Задачи исследования:

- построить математическую модель нейронной сети;
- обучить модель;
- на тестовом наборе данных оценить работу модели;
- проанализировать результаты правильности работы программы на наборах данных.

1. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

1.1. Введение в нейронные сети

Человеческий мозг способен обрабатывать огромные массивы данных, при этом алгоритмы обработки серьёзно отличаются от тех, что используют современные компьютеры. Именно с этим и связаны разработки искусственных нейронных сетей (в дальнейшем будем под этим подразумевать термин нейронные сети). Мозг является крайне сложным, нелинейным, параллельным суперкомпьютером.

Мозг способен создавать свои структурные части, называемые нейронами, таким образом, чтобы они были способны выполнять задачи, для которых они (нейроны) были созданы (обработка поступающих образов, сигналов различных органов чувств и др. функции) гораздо быстрее, чем любые современные компьютеры. В качестве примера такой задачи можно выделить человеческое зрение. Зрительная система позволяет составить представление об окружающем мире в том виде, который даёт возможность взаимодействия с этим миром.

И даже больше того, человеческий мозг способен последовательно выполнять несколько задач распознавания образов, например, узнавание знакомого человека в незнакомом окружении. Для выполнения подобной операции ему нужно порядка 100-200 мс, тогда как для решения такой же задачи на компьютере, даже с меньшей сложностью, способно занять куда большее время.

Почему же человеческий мозг способен добиться таких результатов?

Это объясняется тем, что структура мозга позволяет с рождения устанавливать личные правила обработки данных на основе «опыта». Опыт накапливается на протяжении всей жизни человека, причем самые резкие и сильные изменения происходят в течение первых 2 лет жизни человека. В этот период закладывается основа общей структуры мозга. В это время формируется каркас общей структуры.

Понятие «развитие нейронов» напрямую связано с пластичностью человеческого мозга - его способностью изменять параметры нервной системы в зависимости от условий окружающей среды. Вот почему пластичность - важнейшее свойство нейронов в виде единиц обработки данных в мозге человека. Также, как и человеческий мозг, ИНС используют искусственные нейроны для обработки информации. Если мы посмотрим на нейронную сеть в целом, это машина, которая использует тот же метод обработки данных, что и человеческий мозг, для решения конкретной проблемы. Такие сети обычно реализуются с использованием электронных элементов или моделируются программой, работающей на цифровом компьютере.

Элементарные вычислительные ячейки, называемые нейронами с множеством взаимосвязей между собой, используются для достижения высокой производительности нейронные сети.

Исходя из вышесказанного, дадим определение искусственным нейронным сетям: нейронная сеть - это огромный распределенный параллельный процессор, состоящий из примитивных блоков обработки данных, способных накапливать знания и передавать их для дальнейшей обработки. Сходство нейронной сети с мозгом человека выражается в следующих двух фактах:

- данные для нейронной сети поступают из внешней среды и после используются в процессе обучения;
- происходит накоплений знаний (опыта) в связях между нейронами, называемыми синаптическими весами.

Метод, используемый для обучения нейронной сети, известен как алгоритм обучения. Данный метод наращивает веса по определенному правилу нейронной сети, чтобы получить желаемую структуру связей нейронов.

Процесс изменения весов в нейронной сети является методом настройки сети. Этот подход очень близок к теории линейных адаптивных фильтров. Однако в то же время нейронные сети могут изменять свою собственную структуру. Биологически это связано с тем, что нейроны в человеческом мозге постоянно отмирают и постоянно создаются новые синаптические связи.

Нейронные сети получают свою вычислительную мощность из нескольких источников. Первый - это параллельная обработка данных, второй - способность нейронной сети обучаться при помощи входных данных. Способность учиться также известна как способность обобщать информацию. Этот термин подразумевает возможность получить обоснованный (обдуманый) результат на основе данных, которые не использовались в процессе обучения. Именно эти особенности дают возможность нейронным сетям находить решение большинства задач в разных различных областях, начиная от распознавания объектов и их поиска и заканчивая созданием музыки нейронными сетями, что другими способами сейчас затруднительно или затрачивает много времени. Однако, хоть они и кажутся в теории панацеей, которая может решить любую задачу, на практике «при самостоятельной работе» они не способны всегда получить нужное (правильное) решение. Поэтому их необходимо встраивать в сложные системы, находящиеся под контролем человека.

1.2. Математическая постановка задачи

Запишем формальную постановку задачи классификации текста.

Пусть $A^{train} = \{A_1^{train}, \dots, A_n^{train}\}$ – множество записей. Каждая запись $A_i^{train}, i = 1 \dots n$ представляет собой последовательность слов.

$B^{train} = \{B_1^{train}, B_2^{train}\}$ – набор классов (метки классов), где B_1^{train} – позитив (1), B_2^{train} – негатив (0).

Пусть каждой записи A_i^{train} сопоставлен свой класс B_j^{train} .

Для каждой входной записи H^{test} требуется восстановить метку класса B^{test} , т.е. определить класс объекта.

Для определения класса используются следующие пункты:

1. Предобработка данных: удаляются все html тэги, пунктуация, ненужные символы, производится лемматизация.

2. Обучение – это выбор конкретного атрибута, который наилучшим образом описывает шаблон.

3. Классификация – категоризирует входной объект в определенные классы с использованием подходящего метода, который сравнивает шаблоны текста с целевыми шаблонами.

1.3. Основные определения

Тренировочный набор - это последовательность данных, которыми управляет ИНС.

Эпоха в момент инициализации нейронной сети – это значение, равное нулю, и имеющее верхний предел, который задается вручную. Большее число эпох соответствует лучшей тренировке модели и улучшению результата.

Ошибка - это процент, который отражает несоответствие между ожидаемым и точным результатом.

1.4. Искусственные нейронные сети

Искусственная нейронная сеть (ИНС) – это математическая модель, имеющая аппаратное или программное воплощение, состоящая из совокупности простых связанных и взаимодействующих процессоров (искусственных нейронов). Эти процессоры, как правило, довольно просты. Каждый нейрон в этой сети имеет дело только с сигналами, которые он получает, и с сигналами, которые он периодически отправляет другим нейронам. Такая необычайно простая модель способна выполнять довольно сложные задачи.

ИНС выдает ответ вида: $y(x, w) = f\left(\sum_{i=0}^n w_i \phi_i(x)\right)$, где f - функция активации, w_i – вес, ϕ_i – базисные функции.

У каждой нейронной сети имеются входы и выходы, собственно нейроны и связь между ними. Схема простой нейронной сети изображена на рисунке 1, где $x_1 \dots x_n$ – входные данные, w_{ij} – весовые коэффициенты, y_1, y_2 – выходные данные.

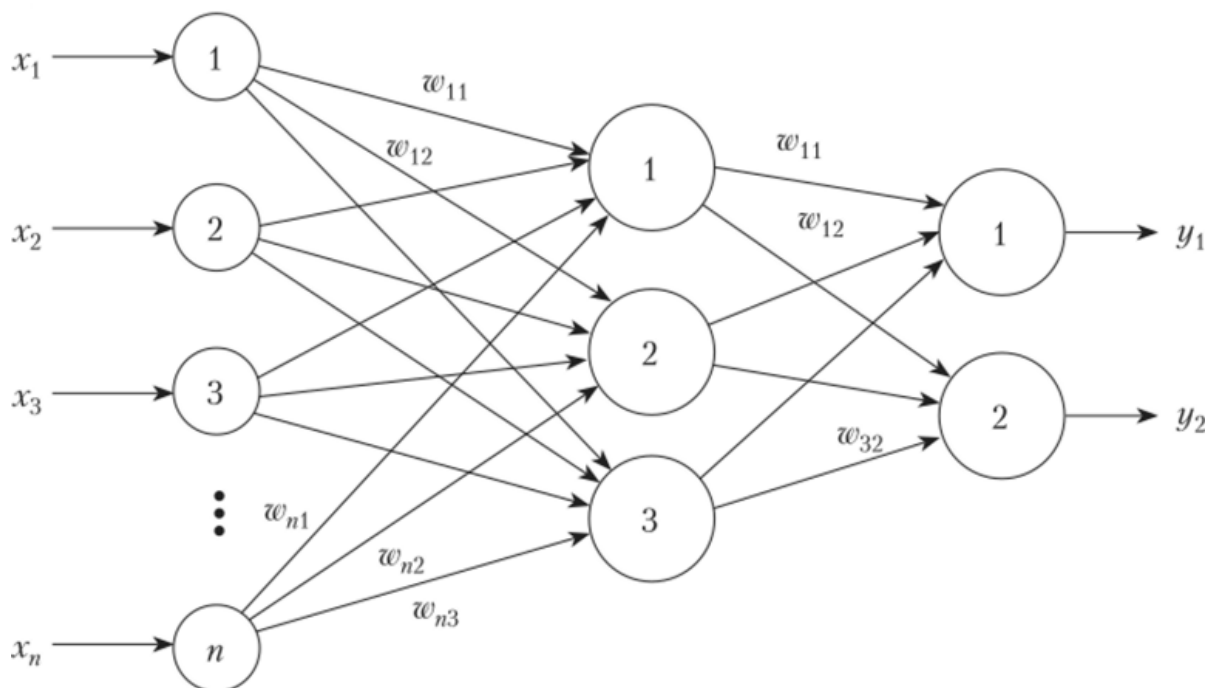


Рисунок 1 – Схема нейронной сети

Для структурной идентификации ИНС необходимо задать число нейронов в скрытом слое и характер связей между нейронами, то есть функции активации.

1.5. Нейрон

Нейрон – это вычислительная единица нейронной сети. Она получает информацию и производит над ней некоторые несложные вычисления и передает ее дальше. У каждого нейрона имеются 2 основных параметра: входные и выходные данные. Последовательность действий одного нейрона:

- прием сигналов от предыдущих элементов сети;
- комбинирование входных сигналов;
- вычисление выходного сигнала;
- передача выходного сигнала следующим элементам нейронной сети.

Обобщенная математическая модель нейрона представлена на рисунке 2. Где x_i – входные сигналы, w_i – весовые коэффициенты. Поступившие на входы

сигналы умножаются на свои веса $P = \sum_{i=1}^n x_i w_i$ и проходят через функцию активации φ .

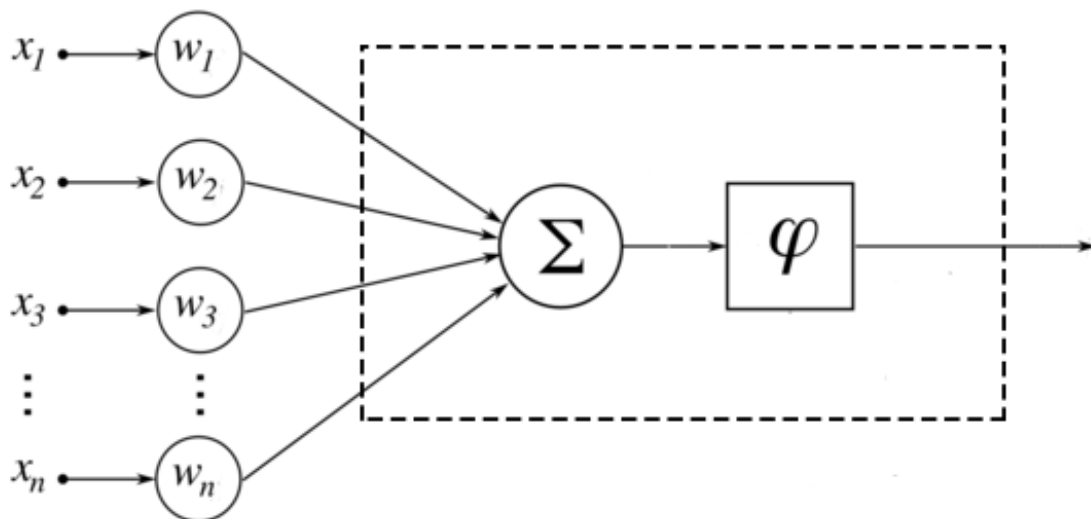


Рисунок 2 – Математическая модель нейрона

1.6. Синапс

Синапс - это связь между двумя нейронами, у которой имеется только один параметр, называемый весом. Благодаря этому весу изменяется входная информация при передаче от одного нейрона к другому. Причем у нейрона с большим весом информация будет иметь приоритет в следующем нейроне. Благодаря этому правилу входная информация обрабатывается и преобразуется в результат. Начальные веса назначаются случайным образом при инициализации ИНС.

1.7. Функция активации

Функция активации – это функция, при помощи которой вычисляется выходной сигнал нейрона. Сигнал, который получен на выходе входного сумматора, используется в качестве аргумента. Это означает, что при подаче числа на вход через функцию активации, мы получим число в желаемом диапазоне на выходе. Рассмотрим наиболее часто используемые функции активации.

1.7.1. Ступенчатая функция

Данная функция является пороговой функцией активации и имеет вид

$$f(x) = \begin{cases} 1, & x > 0, \\ -1, & x \leq 0. \end{cases}$$

Функция меняет значение скачкообразно при достижении значения аргумента функции порогового значения.

График функции представлен на 3 рисунке.

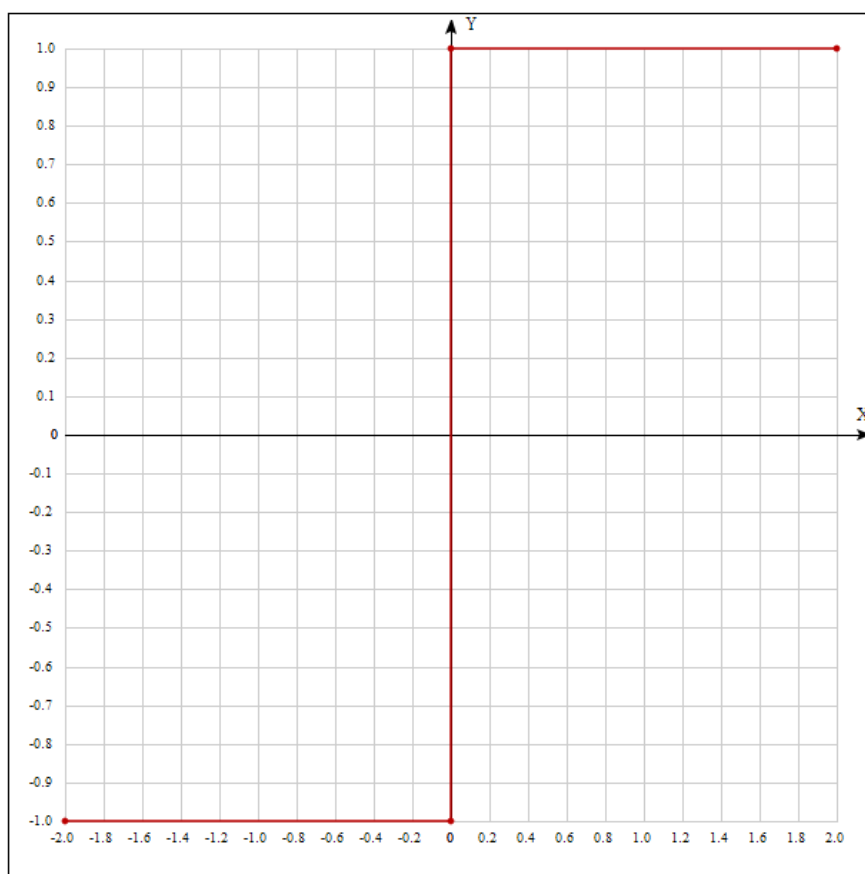


Рисунок 3 – Линейная функция

1.7.2. Линейная функция

Линейная функция активации имеет вид $f(x) = x$. У данной кусочно-линейной функции имеются два линейных участка: на первом функция активации тождественно равна минимально допустимому и на втором максимально допустимому значению, на котором функция строго монотонно возрастает.

График функции представлен на 4 рисунке.

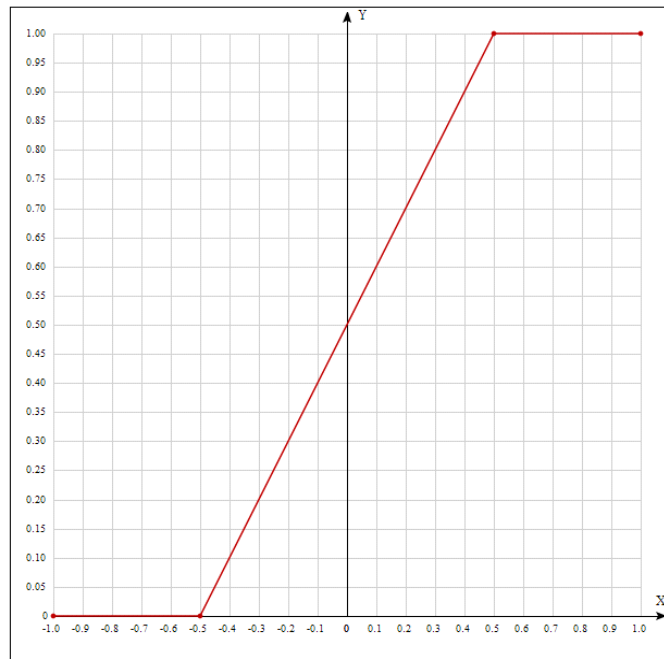


Рисунок 4 – Линейная функция

1.7.3. Сигмоид

Эта функция, относящиеся к классу непрерывных функций вида $f(x) = \frac{1}{(1 + e^x)}$. Число, прошедшее через данную функцию, будет находится в интервале от 0 до 1. График функции представлен на 5 рисунке.

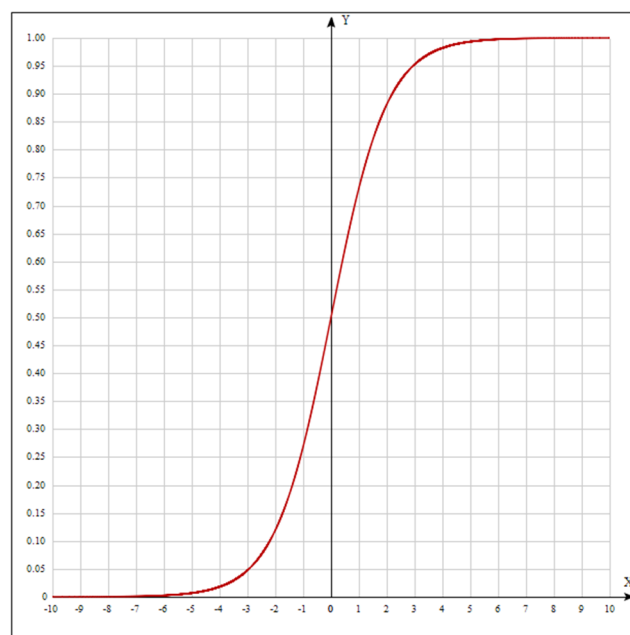


Рисунок 5 – Сигмоид

1.7.4. Гиперболический тангенс

Гиперболический тангенс (рисунок 6) очень похож на сигмоиду. И, действительно, это скорректированная сигмовидная функция:

$$f(x) = 2\text{sigmoid}(2x) - 1 \text{ или } f(x) = \frac{2}{1 + e^{-2x}} - 1.$$

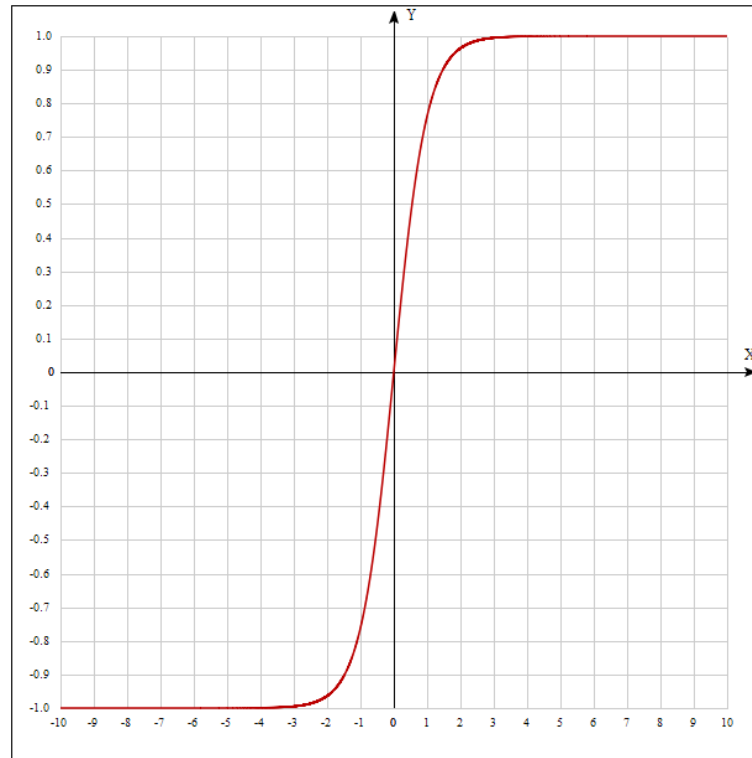


Рисунок 6 – Гиперболический тангенс

1.7.5. Softmax

Функция Softmax — это обобщение логистической функции для многомерного случая. Функция преобразует вектор z размерности K в вектор σ той же размерности, где каждая координата σ_i полученного вектора представлена вещественным числом в интервале $[0,1]$ и сумма координат равна 1. Координаты вычисляются следующим образом:

$$\sigma(z)_i = \frac{e^{z_i}}{\sum_{k=1}^K e^{z_k}}.$$

График функции представлен на рисунке 7.

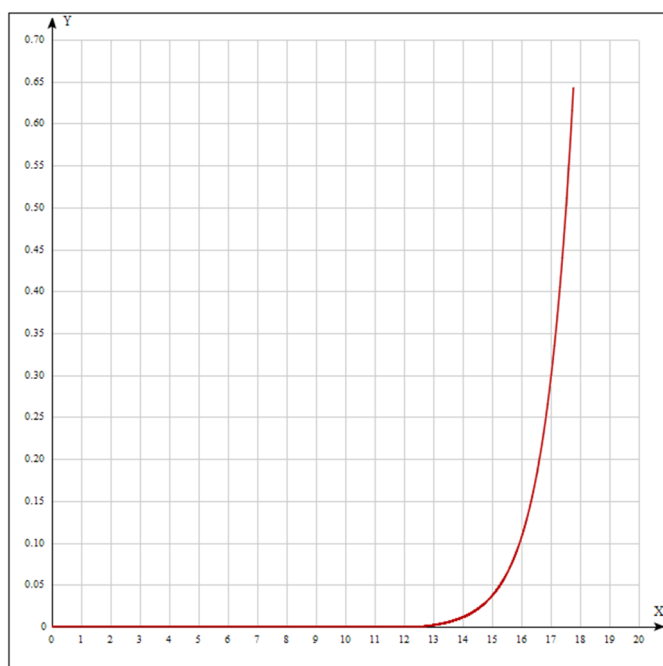


Рисунок 7 – Softmax

1.7.6. ReLU

Функция активации ReLu возвращает значение x , если x положительно, и 0 в противном случае. ReLu имеет вид: $f(x) = \max(0, x)$. График функции представлен на рисунке 8.

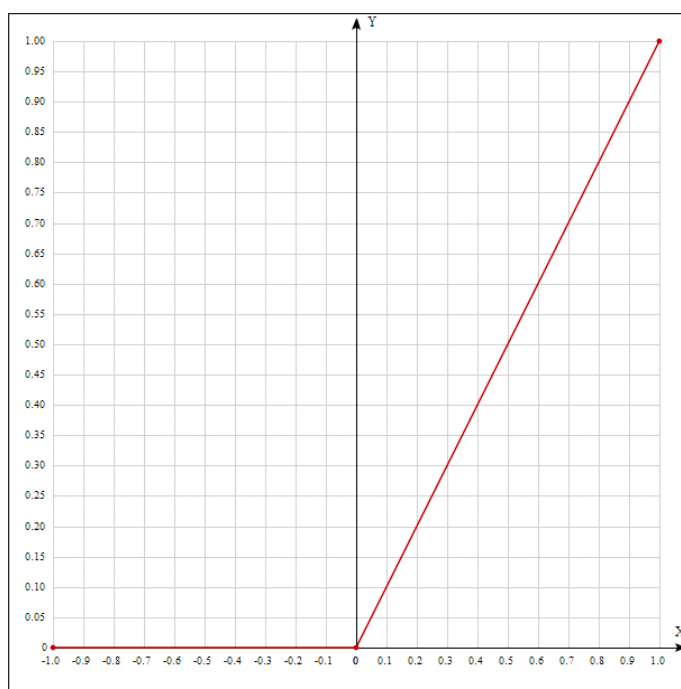


Рисунок 8 – ReLU

1.8. Функция потерь

Функция потерь является одной из важных составляющих нейронных сетей. Функция потерь – это функция, которая характеризует потери при неправильном принятии решений на основе наблюдаемых данных. Она используется для расчета ошибки между реальными и полученными ответами.

В данной работе используется *binary crossentropy (BCE)* или бинарная перекрёстная энтропия. *BCE loss* используется для задач бинарной классификации. При выборе данной функции, как функцию потерь, нужен только один выходной узел, чтобы классифицировать данные в два класса. Выходное значение должно быть передано через сигмовидную функцию активации, где диапазон выходного сигнала составляет от 0 до 1.

$$binarycrossentropyloss = -(y \log(p) + (1 - y) \log(1 - p)),$$

где y – двоичный индикатор (0 или 1) того, является ли метка класса правильной классификацией для наблюдения в текущей задаче, p – прогнозируемая вероятность модели.

1.9. Рекуррентные нейронные сети

Рекуррентные нейронные сети (RNN) — вид нейронных сетей, в которых связи между элементами образуют направленную последовательность. Это позволяет обрабатывать последовательные пространственные цепочки или серии событий во времени.

В отличие от многослойных перцептронов [3], рекуррентные сети могут использовать свою внутреннюю память для обработки последовательностей произвольной длины. Поэтому сети RNN применимы в таких задачах, где нечто целостное разбито на сегменты, например, распознавание рукописного текста или распознавание речи. В последнее время наибольшее распространение получили сети с долговременной и кратковременной памятью (LSTM) и управляемый рекуррентный блок (GRU) [4].

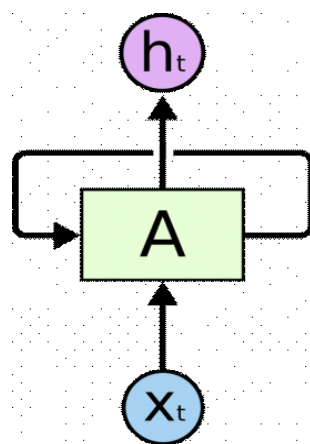


Рисунок 9 – Циклы в РНС

На рисунке 9 представлен участок нейронной сети, где блок A получает некие данные x_t на вход и подает на выход некоторое значение h_t . Такая циклическая связь позволяет передавать информацию от текущего шага сети к следующему.

Большая трудность рекуррентной сети заключается в том, что, если учитывать каждый шаг времени, то становится необходимым для каждого шага времени создавать свой слой нейронов, что вызывает серьёзные вычислительные сложности.

Рекуррентные нейронные сети не так уж сильно отличаются от обычных нейронных сетей. Их можно представить себе, как множество копий одной и той же сети, причем каждая копия передает сообщение следующей копии.

Развернутый цикл в РНС представлен на рисунке 10.

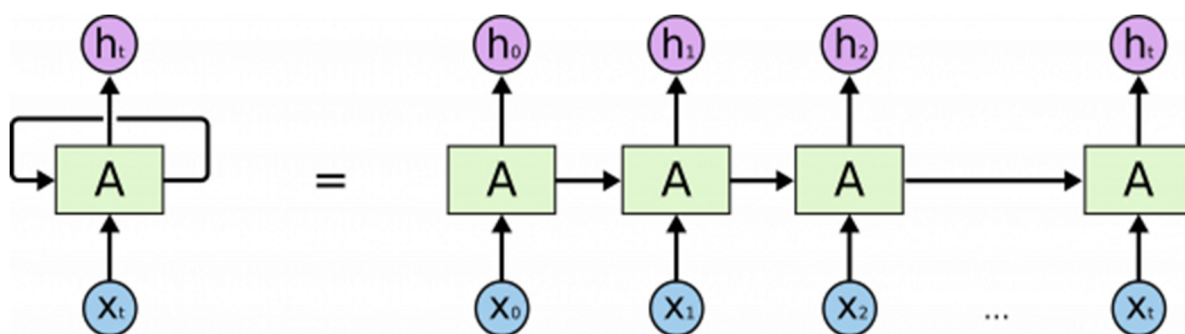


Рисунок 10 – Развертка цикла в РНС

1.9.1. Полностью рекуррентная сеть

Данная базовая архитектура была разработана в 1980-х годах. Сеть состоит из узлов, каждый из которых связан со всеми остальными узлами. Для каждого нейрона порог активации изменяется со временем и является действительным числом. Каждое соединение имеет переменный вещественный вес. Узлы делятся на входные, выходные и скрытые.

При обучении с учителем с дискретным временем каждый (дискретный) временной шаг подается на входные узлы, остальные узлы завершают свою активацию, и выходные сигналы подготавливаются для передачи нейроном на следующий уровень. Например, если сеть отвечает за распознавание речи, в результате на выходные узлы поступают уже метки (распознанные слова).

При обучении с подкреплением нет учителя, который отправлял бы целевые сигналы в сеть, но иногда использует функцию оценки [5] для оценки качества сети, начальные значения которой влияют на входное поведение сети. В частности, когда сеть реализует игру, результат измеряется количеством выигрышных очков или оценкой позиции. На каждом уровне ошибка вычисляется как сумма отклонений от выходных сигналов сети. При наличии набора обучающих выборок ошибка вычисляется с учетом ошибок каждой отдельной выборки.

1.9.2. Проблема долгосрочных зависимостей

Одна из ключевых идей, которая делает РНС настолько привлекательными, заключается в том, что они могут использовать информацию из прошлого для решения текущих задач. Например, можно использовать предыдущие кадры видео, чтобы понять следующие. Иногда для выполнения текущей задачи нам достаточно последней информации. Например, представьте языковую модель, которая пытается предсказать следующее слово на основе предыдущих. Когда мы пытаемся предсказать последнее слово в предложении «галактики в космосе», нам больше не нужен никакой контекст - очевидно, что мы говорим о кос-

мосе в конце предложения. В таких случаях, когда разрыв между требуемой информацией и тем, где она нужна, невелик. РНС может научиться использовать информацию, полученную ранее (рисунок 11).

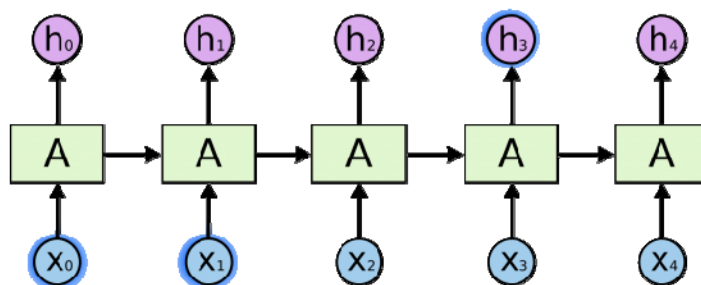


Рисунок 11 – Отсутствие потерь в РНС при краткосрочных связях

Но также бывают случаи, когда нам нужен более широкий контекст. Предположим, нужно предсказать последнее слово в тексте “Я родился и вырос в Италии ... Я могу свободно говорить на итальянском”. Недавняя информация нам может подсказать, что следующим словом, вероятно, будет название языка. Но если мы хотим уточнить какого именно языка, то нам нужен предыдущий контекст, вплоть до информации о Италии. Нередко разрыв между тем местом, где информация нужна, и местом, где она необходима, становится очень большим. К сожалению, по мере увеличения этого разрыва РНС не могут научиться связывать информацию (рисунок 12).

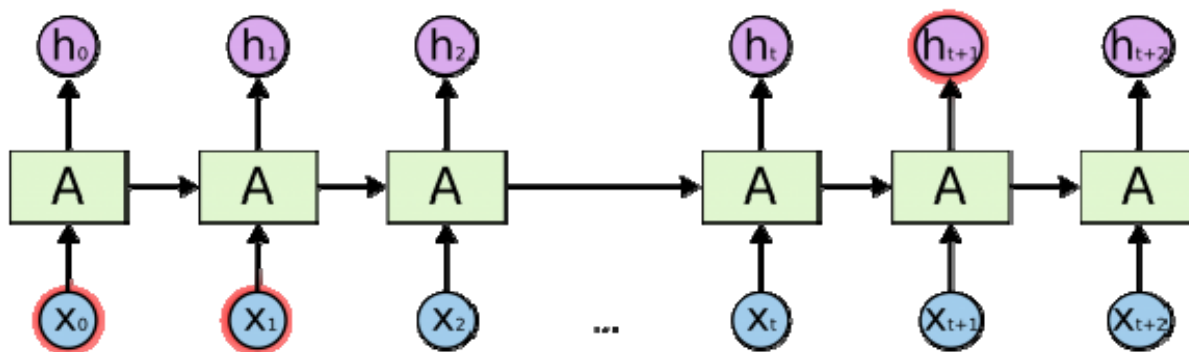


Рисунок 12 – Потеря информации в РНС при долгосрочных связях

Теоретически РНС способны справляться с такими долгосрочными зависимостями. Человек может тщательно подбирать свои параметры для решения игрушечных задач такой формы. Однако на практике РНС не могут этому научиться.

1.10. Сети LSTM

Мыслительный процесс у человека не начинается с нуля. При чтении книги, мы понимаем и осознаем смысл каждого слова, исходя из предыдущих слов. Мысли постепенно накапливаются и влияют друг на друга. Такой же принцип применяется в сетях LSTM.

LSTM (long short-term memory - долгая краткосрочная память) — тип рекуррентной нейронной сети, способный обучаться долгосрочным зависимостям.

LSTM специально разработаны для устранения проблемы долгосрочной зависимости. Их специализация – запоминание информации в течение длительных периодов времени.

Все рекуррентные нейронные сети имеют форму цепочки повторяющихся модулей нейронной сети. В стандартных РНС этот повторяющийся модуль имеет простую структуру (рисунок 13), в отличие от сетей LSTM, модуль которых состоит из четырех взаимодействующих слоев (рисунок 14).

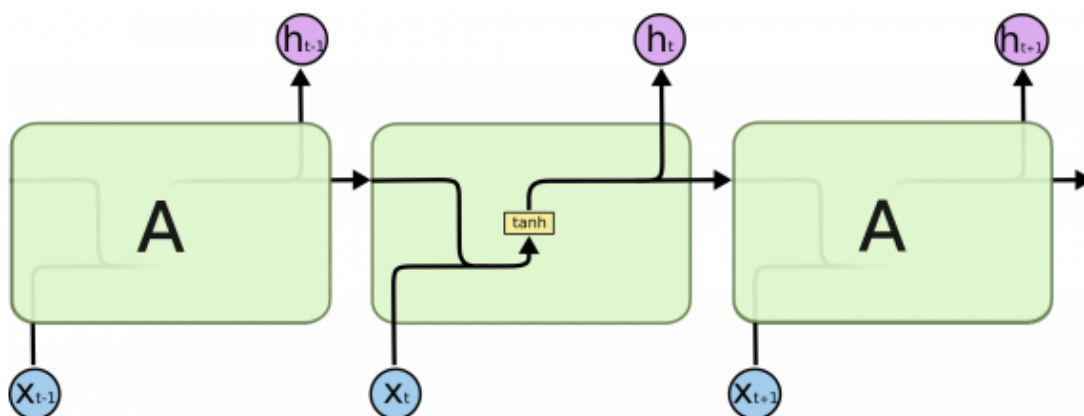


Рисунок 13 – Повторяющийся модуль стандартной РНС

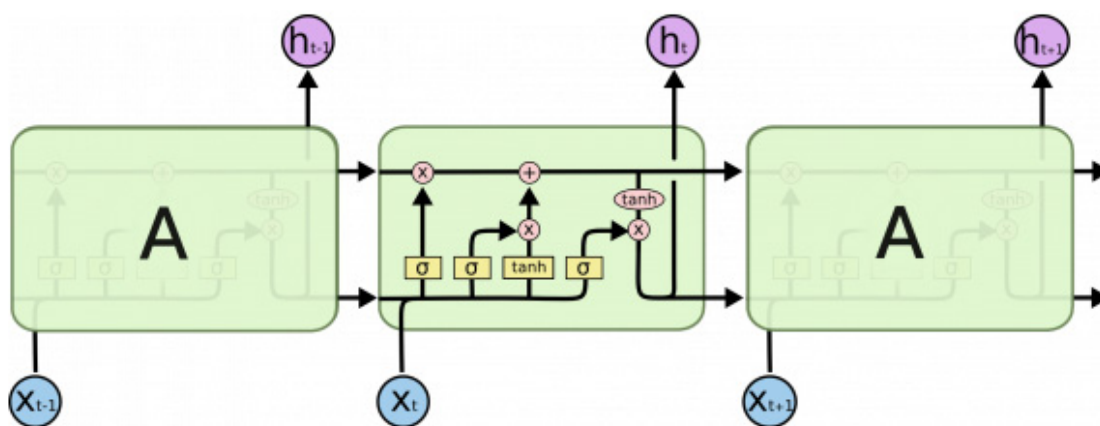


Рисунок 14 – Повторяющийся модуль LSTM

Главным и отличительным параметром сетей LSTM является канал (состояние ячейки), находящийся в верхней части диаграммы (рисунок 15).

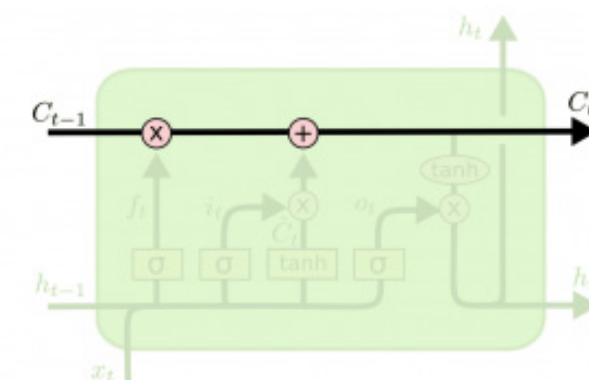


Рисунок 15 – Горизонтальная линия (канал), проходящая через верхнюю часть диаграммы

Сам канал очень похож на ленту конвейера. Он проходит через весь блок сети и подвергается небольшим линейным преобразованиям.

Сеть LSTM уменьшает или увеличивает количество информации в состоянии ячейки, в зависимости от потребностей. Для этого используются тщательно настраиваемые структуры, называемые гейтами.

Гейт — это «ворота», пропускающие или не пропускающие информацию. Гейты состоят из сигмовидного слоя нейронной сети и операции поточечного умножения (рисунок 16).

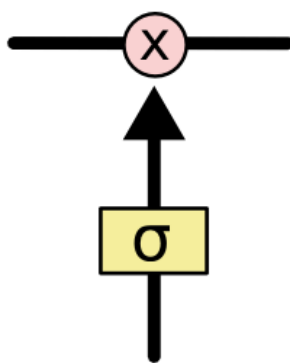


Рисунок 16 – Гейт (ворота)

Информация, проходя через гейт с функцией активации сигмоид, выдает значение от нуля до единицы, тем самым решая, сколько единиц каждой информации ей следует далее пропустить. Значение «0» означает «не пропустить ничего», значение «1» — «пропустить все».

У сети LSTM имеется три гейта для контроля состояния ячейки. Этапы работы сети LSTM представлены далее.

1.10.1. Слой утраты

Первый этап в работе сети LSTM – «забывание не нужного», то есть сеть решает какую информацию стоит удалить из состояния ячейки. Для принятия этого решения используется слой нейронной сети с сигмовидной функцией активации называемый «слоем гейта утраты» (рисунок 17). На вход он получает значения h_{t-1} и x_t , после обработки, на выходе гейта образуется вектор f_t с последовательностью чисел от «0» до «1» для каждого номера в состоянии ячейки C_{t-1} , где «1» будет означать полное сохранение информации, а «0» - «полностью забыть».

$$f_t = \sigma_g(W_f[h_{t-1}, x_t] + b_f),$$

где x_t – входной вектор, h_{t-1} – вектор стандартного скрытого состояния предыдущего блока, W_f – матрица параметров (веса), b_f – вектор смещения, σ – функция активации (сигмоид), f_t – вектор вентиля забывания, вес забывания старой информации.

Вектор смещения состоит из нейронов смещения или bias нейронов. Особенность этого типа нейронов заключается в том, что его вход и выход всегда равняются 1.

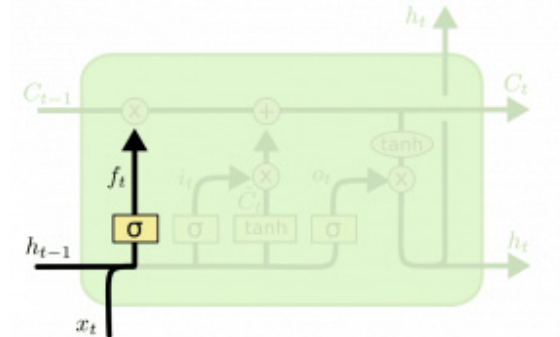


Рисунок 17 – Слой утраты

1.10.2. Слой сохранения

На текущем этапе решается задача о том, какую информацию следует добавить (сохранить) в состояние ячейки. Данную задачу можно разделить на 2 шага. На первом шаге, слой нейронной сети с сигмовидной функцией активации, который называется «слоем гейта входа» (рисунок 18), решает, какую информация следует обновить с состояние ячейки i_t (процесс «сохранения и забывания» информации аналогичен с предыдущим шагом). На втором шаге слой нейронной сети с функцией активации гиперболический тангенс подбирается вектор кандидатов на новые значения $\hat{C}_t$, который будет добавлен к состоянию ячейки. Далее происходит поэлементное умножение (объединение) векторов i_t и $\hat{C}_t$, тем самым создается обновление для состояния:

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

$$\hat{C}_t = \tanh(W_c[h_{t-1}, x_t] + b_c),$$

где b_i и b_c – векторы смещения, W_i и W_c – матрицы параметров (весовые коэффициенты), $\tanh$ – функция активации гиперболический тангенс, i_t – вектор входного вентиля, вес получения новой информации, $\hat{C}_t$ – промежуточный вектор контента.

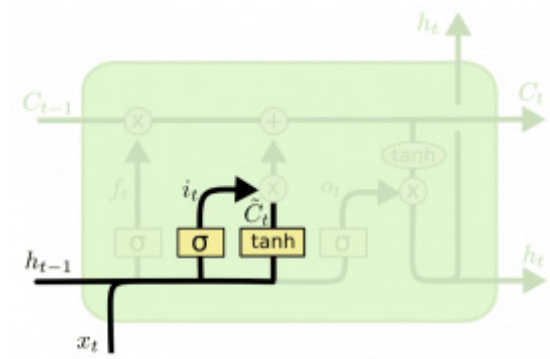


Рисунок 18 – Слой входа

1.10.3. Новое состояние

На третьем этапе обновляется состояние ячейки (рисунок 19):

- поэлементно умножается предыдущее состояние ячейки C_{t-1} на вектор f_t . Теряется информация, которую мы решили забыть;
- добавляется новая информация в вектор долгосрочного контента $i_t * \hat{C}_t$, подсчитанная на предыдущем этапе.

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

где C_t – вектор долгосрочного контента.

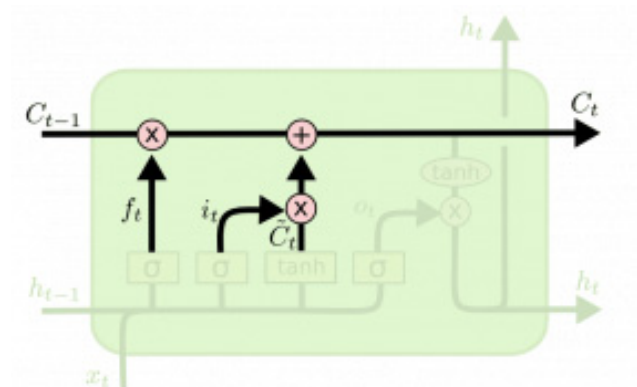


Рисунок 19 – Обновленное состояние ячейки

Наконец, последний этап работы блока. В нем формируется скрытое состояние ячейки h_t (рисунок 20). Принцип формирования скрытого состояния тот же самый, что и в предыдущих этапах. Формируется оценочный вектор o_t при помощи слоя нейронной сети с сигмовидной функцией активации. Далее вектор

долгосрочного контента C_t , который нормируется, проходя через функцию гиперболического тангенса поэлементно умножается на вектор o_t .

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

$$h_t = o_t * \tanh(C_t),$$

где W_o – матрица параметров (веса), b_o – вектор смещения, o_t – вектор выходного вентиля, кандидат на выход.

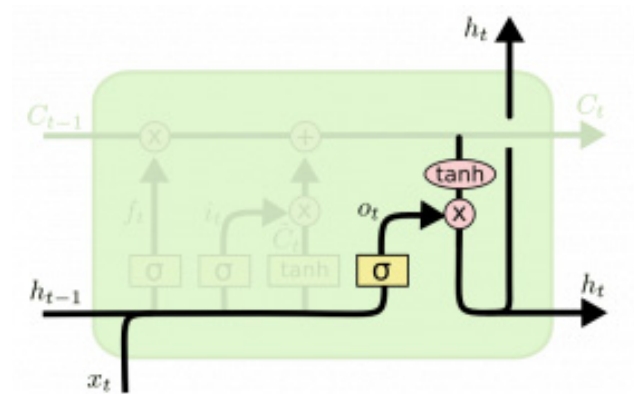


Рисунок 20 – Отфильтрованное состояние ячейки

Таким образом, суть обучения сети LSTM состоит в корректировке весов матриц W_f, W_i, W_c, W_j .

2. ПРОГРАММНАЯ РЕАЛИЗАЦИЯ РЕКУРРЕНТНОЙ СЕТИ

Для написания программной части требовался хорошо задокументированный язык программирования с наличием библиотек для классификации и высокой скоростью работы. Один из самых распространённых языков, который подходит под данное описание – это Python 3 [5].

Python – высокоуровневый язык программирования общего назначения, ориентированный на повышение производительности разработчика и читаемости кода. Весомым аргументом в пользу использования Python – наличие большого количества библиотек и расширений для работы в области машинного обучения, например: Tensorflow [6], scikit-learn [7], Theano [8].

Выбор был сделан в пользу библиотеки TensorFlow. Она подробно и понятно задокументирована, следовательно, удобна при использовании.

2.1. Загрузка данных

Для проведения эксперимента классификации текста был загружен Dataset [9] с «твитами» (короткими текстами) и конечным набором классов. Если метка класса 1 - позитив, с 0 - негатив. В Dataset'е содержится 120000 позитивных записей и 115000 негативных. Также был написан python-скрипт, который загружает комментарии из социальной сети «ВКонтакте». На вход скрипт принимает id сообщества и id записи (поста). Для загрузки комментариев использовалась функция, представленная в листинге (рисунок 21).

```
def pars_com():
    offset = 0
    while offset < counter:
        response = requests.get('https://api.vk.com/method/wall.getComments',
                                params={
                                    'access_token': token,
                                    'v': version,
                                    'owner_id': owner_id,
                                    'post_id': post_id,
                                    'count': count,
                                    'offset': offset
                                })
        data = response.json()['response']['items']
        offset += 5
        all_com.extend(data)
        time.sleep(1)
    return all_com
```

Рисунок 21 – Функция для загрузки комментариев

Полный листинг программы для загрузки данных представлен в приложении А.

2.2. Предобработка данных

Перед любой обработкой данных требуется предобработка. В этой стадии удаляются все html тэги, пунктуация, ненужные символы, производится лемматизация. Данная операции осуществляется с помощью библиотек python: re и MorphAnalyzer.

Данные загружаются в программу, и удаляются все символы, кроме кириллицы, при помощи регулярных выражений:

```
reviews[i] = re.sub(r'^А-Яа-я\s|', '', reviews[i]).
```

Далее при помощи еще одного регулярного выражения удаляются лишние пробелы, возникшие при первой обработке:

```
reviews[i] = re.sub(r'\s+', ' ', reviews[i]).
```

Чтобы избежать «засорения словаря» словами с разными падежами или склонениями, проведём лемматизацию, получив нормальную (начальную, или словарную) форму слова. Данная процедура описана в листинге (рисунок 22).

```
morph = MorphAnalyzer()
def lemmatize(doc):
    tokens = []
    for token in doc.split():
        token = token.strip()
        token = morph.normal_forms(token)[0]
        tokens.append(token)
    if len(tokens) > 2:
        return tokens
    return None
```

Рисунок 22 – Функция для лемматизации текста

Полный листинг программы для загрузки данных представлен в приложении Б.

2.3. Обработка данных

Поскольку нейронная сеть работает с числовыми значениями, необходимо сформировать числовой входной вектор значений [10]. Данный процесс называется токенизацией [11]. Токенизация - это процесс, посредством которого большое количество текста делится на более мелкие части (тексты на предложения, предложения на слова), называемые токенами.

Создается токенизатор Keras, ему подается параметр `num_words = 6000`, где `num_words` – желаемый объем словаря с самыми часто встречающимися словами в Dataset'е. Далее происходит обучение токенизатора на наборе данных из Dataset'а и создается словарь. Затем происходит процесс векторизации - процесс конвертации текста в числа. Каждая запись в Dataset'е проходит обработку. Если слово из записи есть в словаре, то это слово заменяется на число, соответствующее ему из словаря. Тем самым запись превращается в вектор. Листинг:

```
tokenizer = Tokenizer(num_words=num_words)
tokenizer.fit_on_texts(reviews)
sequences = tokenizer.texts_to_sequences(reviews)
```

Для того, чтобы каждую запись можно было подать на вход в нейронную сеть, ее нужно ограничить определенным количеством токенов:

```
max_review_len = 20.
```

Если же запись окажется больше 20 слов, то она будет обрезана, если меньше, то она будет дополнена нолями:

```
x = pad_sequences(sequences, maxlen=max_review_len).
```

После обработки данных их следует разделить на обучающий, проверочный и тестовый наборы. На тестовый набор `x_test` и `y_test` принято выделять 30%. Оставшиеся 70% `x_train` и `y_train` идут на обучающий и проверочный наборы данных. Библиотека Scikit-Learn предоставляет метод `train_test_split` [12], который разделяет данные:

```
x_train, x_test, y_train, y_test = train_test_split(x, y,
test_size = 0.3).
```

После всех подготовок следует построение, компиляция и обучение модели нейронной сети.

2.4. Построение модели

Нейронная сеть состоит из следующих 3 слоёв:

- входной слой Embedding [13];
- второй слой LSTM;
- выходной содержит один нейрон с функцией активации сигмоид.

Значение выхода из последнего слоя определяет позитив или негатив и выдает результат, где 1 – абсолютный позитив, а 0 – абсолютный негатив. Определим модель в листинге:

```
model.add(Embedding(num_words,8, input_length=max_review_len))
model.add(LSTM(32))
model.add(Dense(1, activation='sigmoid'))
```

Сначала на входном слое создается плотное векторное представление. На вход следующие параметры: словарь, длина плотного вектора, который мы хотим получить, и длина записи. Далее идет слой с 32 модулями LSTM и, в конце, слой с одним нейроном, который будет классифицировать записи.

2.5. Компиляция модели

Компиляция модели происходит при помощи метода `compile` и 3 входных параметров:

- оптимизатор (`optimizer`);
- потери (`loss`);
- показатели (`metrics`).

В данной задаче используется оптимизатор «Adam» [14]. Он сочетает в себе и идею накопления движения и идею более слабого обновления весов для типичных признаков. То, как быстро рассчитываются оптимальные веса, определяется скоростью обучения.

В качестве функции потери используется `binary_crossentropy loss` [].

А в качестве метрики используется ассигасу – доля правильных ответов алгоритма.

Для компиляции модели применим метод `compile` в листинге:

```
model.compile(optimizer='adam', loss='binary_crossentropy',  
metrics=['accuracy'])
```

2.6. Обучение модели

После компиляции, следует этап обучения модели. В библиотеке TensorFlow имеется метод `fit`, который, по сути, «тренирует» нашу модель. В течение «тренировки» находятся коэффициенты, веса и смещение. У данной функции имеются как основные параметры: тренировочные данные (`x_train`) и соответствующие им метки (`y_train`), так и необязательные. К необязательным параметрам относятся: `epochs` – количество эпох, `validation_split` – проверочный набор данных, `batch_size` – количество записей, проходящих обработку одновременно, `callbacks` – функция, при помощи которой сохранится модель с лучшими показателями на проверочном наборе. Данные из проверочного набора отличаются от набора данных для тестирования. Проверочный набор удерживается от обучения модели, он используется для выдачи объективной оценки мастерства окончательно настроенной модели при сравнении или выборе между конечными моделями. При помощи метода и его выше описанных параметров обучим модель (рисунок 23).

```
history = model.fit(x_train,  
                    y_train,  
                    epochs=2,  
                    batch_size=128,  
                    validation_split=0.1,  
                    callbacks=[checkpoint_callback])
```

Рисунок 23 – Метод `fit` и его параметры

2.7. Этап тестирования

После обучения модели следует этап её оценивания на тестовом наборе данных.

Тестирование модели предельно важно, ведь необходимо получить представление о том, как сеть ведет себя при работе с данными, на которых она не обучалась.

Для тестирования в библиотеке TensorFlow есть методы `evaluate` и `predict`. Метод `evaluate` будем использовать на тестовом наборе данных. На вход этому методу нужно подать тестовые данные (`x_test`) и соответствующие им метки (`y_test`):

```
model.evaluate(x_test, y_test, verbose=1).
```

Метод `predict`, который применяется к данным при прогнозировании вероятности результата (1 – абсолютный позитив, 0 – абсолютный негатив). Для оценки нашей модели используем данный метод:

```
predictions = model.predict(X_test).
```

3. РЕЗУЛЬТАТЫ ИССЛЕДОВАНИЯ

3.1. Основные характеристики

Для вычислительного эксперимента по классификации текста использовалась ЭВМ со следующими характеристиками (рисунок 24):

- Процессор: intel Core i3 – 4000m с 2 ядрами и 4 потоками на частоте 2.4 GHz;
- ОЗУ: 8 Гб;
- ОС: Windows 10 Pro 64-разрядная.

 Тип компьютера	Компьютер с ACPI на базе x64 (Mobile)
 Операционная система	Microsoft Windows 10 Pro
 Системная память	8105 МБ (DDR3 SDRAM)
 Видеоадаптер	Intel(R) HD Graphics 4600 (1 Гб)
 Общий размер	223.0 Гб (106.6 Гб свободно)

Рисунок 24 – Конфигурация системы

Для сбора информации о конфигурации системы использовалась программа AIDA 64 Extreme.

3.2. Описание эксперимента

Для проведения эксперимента был реализован программный модуль (приложение В) на языке программирования Python, задача которого - классифицировать текст.

Рассмотрим алгоритм работы на конкретном примере:

1. Из директории в память загружается файл Excel с позитивными и негативными записями со своими метками классов.
2. Создается токенизатор, ему подается параметр, желаемый объем словаря. От выбора объема словаря зависит дальнейшее обучение нейронной сети. Далее происходит обучение токенизатора на наборе данных из первого этапа и создается словарь.

3. Данные случайным образом разделяются на тренировочную и тестовую выборку в соотношении 70% / 30%. В результате разделения на обучение отводится 164500 записей, а для тестирования – 70500.

4. Далее следует создание, компиляция и обучение модели РНС.

5. На заключительном этапе, с помощью методов *evaluate* и *predict*, проводим оценку результатов.

3.3. Исходный набор данных

В исследовании использовалось 235000 записей. В таблице 1 приведены 10 случайных записей. В первом столбце находится метка класса (1 – позитивный твит, 0 – негативный твит), во втором столбце твиты.

Таблица 1 – Тренировочные данные

Класс	Твит
1	Заставила детей принести мне лавочку, самой то мне лень)
1	Ваааау)) классно) я после школы улеглась спать до 13:40, а потом уроки учила, но завтра буду с подругами гулять)пиццу сделала
1	Традиционно 8 декабря принимаю поздравления с днем рождения)
1	Только что узнал что наверное 13 декабря еду в санаторий в Пуховичах) я там уже был там круто.
1	Заранее спасибо)
0	Хочу что бы закрыли все социальные сети и смотреть на смерть общества
0	Мой телефон тупит ужасно, а тут трансляция важная!!:(
0	Как тяжело бросать то ,к чему ты много лет стремилась!
0	Блин,я почему то думала что завтра суббота((никогда еще так не разочаровывалась;
0	Сначала воду отключили, а потом включили холодную. Вообще красавы

3.4. Результаты экспериментов

Для того, чтобы убедиться в работоспособности РНС, необходимо взять тестовый набор данных и соответствующие им метки классов. Пропустить его через сеть, проанализировать результаты и оценить работу сети.

Для оценки качества работы нейронной сети на обучающем и на тестовом наборе данных использовалась метрика *ассурасу*. Она считает количество верно классифицированных объектов к общему количеству объектов.

Обучение нейронной сети (рисунок 25) происходит в течение 65 эпох, сохраняется модель с лучшим результатом на проверочном наборе.

```
Epoch 00060: val_accuracy did not improve from 0.77529
Epoch 61/65
1777/1777 [=====] - 23s 13ms/step - loss: 0.2485 - accuracy: 0.8854 -
val_loss: 0.8669 - val_accuracy: 0.7743

Epoch 00061: val_accuracy did not improve from 0.77529
Epoch 62/65
1777/1777 [=====] - 23s 13ms/step - loss: 0.2464 - accuracy: 0.8862 -
val_loss: 0.8835 - val_accuracy: 0.7762

Epoch 00062: val_accuracy improved from 0.77529 to 0.77624, saving model to best_model.h5
Epoch 63/65
1777/1777 [=====] - 23s 13ms/step - loss: 0.2432 - accuracy: 0.8876 -
val_loss: 0.8792 - val_accuracy: 0.7753

Epoch 00063: val_accuracy did not improve from 0.77624
Epoch 64/65
1777/1777 [=====] - 23s 13ms/step - loss: 0.2429 - accuracy: 0.8880 -
val_loss: 0.8926 - val_accuracy: 0.7780
```

Рисунок 25 – Обучение нейронной сети

На рисунке 26 представлены графики зависимости количества эпох от доли верных ответов на обучающем и проверочном наборах.

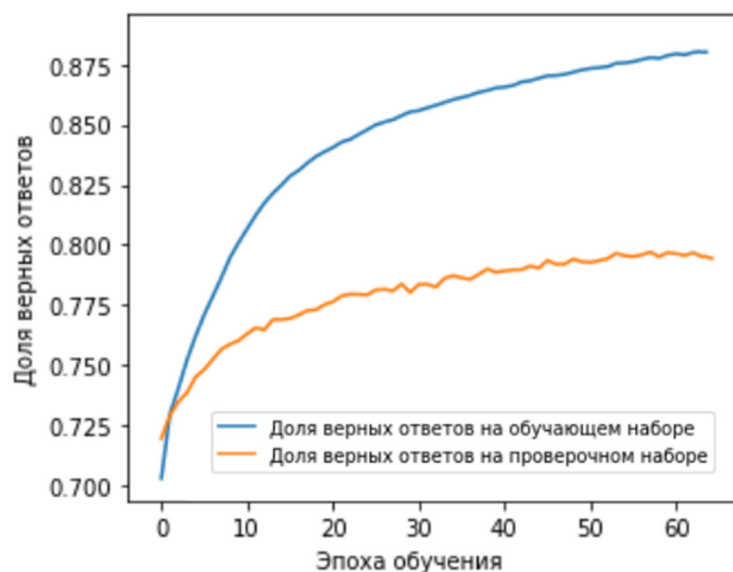


Рисунок 26 – График зависимости количества эпох от доли верных ответов на обучающем наборе, доли верных ответов на проверочном наборе

Анализируя рисунок 26, можно сказать, что, начиная с 54 эпохи, доля верных ответов остаётся неизменной и далее ухудшается, что говорит о переобучении.

На рисунках 27-32 представлены результаты оценки нейронной сети: тестовый набор данных; позитивные комментарии; негативные комментарии; нейтральные комментарии, где нейронная сеть не смогла дать правильные ответ или не была в этом уверена; набор комментариев из позитивной и негативной групп. Значения, близкие к 1, например, 0.9940543, соответствуют тому, что нейронная сеть уверена, что перед ней позитивный комментарий, а числа, близкие к 0, например, 0.01722956, соответствуют тому, что нейронная сеть уверена, что перед ней негативный комментарий.

```
Тестовый набор:
3385/3385 [=====] - 11s 3ms/step - loss:
0.8781 - accuracy: 0.7754
```

Рисунок 27 – Результаты сети на тестовом наборе данных

После 65 эпох обучения эффективность РНС приблизилась к 78% (рисунок 27), что является вполне достойным результатом в работе с тональностью текста.

```
Оригинальный комментарий: здорово)) желаю удачи
Обработанный комментарий: здорово желать удача
[[0.9940543]]
Позитивный комментарий
-----

Оригинальный комментарий: Ахаха Насть, я уже предвкушаю это!)
Обработанный комментарий: ахах настя я уже предвкушать это
[[0.91015506]]
Позитивный комментарий
-----
```

Рисунок 28 – Позитивные комментарии

```
Оригинальный комментарий: Голова болит безумно, еще эти уроки и репетитор
Обработанный комментарий: голова болеть безумно ещ этот урок и репетитор
[[0.01722956]]
Негативный комментарий
-----

Оригинальный комментарий: Гнида такая день испортил
Обработанный комментарий: гнида такой день испортить
[[0.06562123]]
Негативный комментарий
-----
```

Рисунок 29 – Негативные комментарии

По результатам рисунков 28-29 видно, что нейронная сеть уверенно определяет явно позитивные (0.9940543, 0.91015506, близкие к 1) и явно негативные отзывы (0.01722956, 0.06562123, близкие к 0).

```
Оригинальный комментарий: всякое бывает, забудь  
Обработанный комментарий: всякий бывает забыть  
[[0.34309375]]  
Негативный комментарий  
-----  
Оригинальный комментарий: ладно, еще увидимся  
Обработанный комментарий: ладный еще увидится  
[[0.82972425]]  
Позитивный комментарий  
-----  
Оригинальный комментарий: за сколько взял?  
Обработанный комментарий: за сколько взял  
[[0.63552046]]  
Позитивный комментарий  
-----
```

Рисунок 30 – Нейтральные комментарии

На нейтральных комментариях, с точки зрения человека (рисунок 30), сеть не дает таких точных результатов. Граница разделения между позитивом и негативом 0.5.

```
Набор комментариев из позитивной группы:  
[[0.68212134]]
```

Рисунок 31 – Набор комментариев из позитивной группы

```
Набор комментариев из негативной группы:  
[[0.38625973]]
```

Рисунок 32 – Набор комментариев из негативной группы

Исходя из рисунков 31-32, видно, что нейронная сеть справляется со своей задачей и верно определяет позитив и негатив для группы комментариев. Стоит учитывать, что среди тестируемых наборов из социальной сети могут попадаться как позитивные и нейтральные отзывы в негативном наборе, так и наоборот. Также стоит учитывать, что некоторые слова ей могут быть не знакомы. Оценка ведется по всем выбранным комментариям.

При обработке текста или комментария из социальных сетей нейронная сеть может не знать каких-либо слов, так как выбранный словарь для обучения ограничен набором в 6000 слов. В этом случае нужно в разработанной ИНС, позволяющей отслеживать негативную информацию в социальных сетях, использовать Dataset в 3–4 раза больше и привлекать более мощные вычислительные ресурсы.

В данном разделе была рассмотрена описательная часть эксперимента по определению негативного текста в социальных сетях. Проведены эксперименты по определению негативного контента для тестового набора данных.

ЗАКЛЮЧЕНИЕ

В данной работе была изучена задача классификации текста с применением модели рекуррентной нейронной сети. Рассмотрена актуальная архитектура сетей LSTM.

Изучена проблема долгосрочных зависимостей РНС. Рассмотрена задача бинарной классификации текста при помощи модели нейронной сети. Также была реализована актуальная архитектура сетей LSTM, которая хорошо зарекомендовала себя в машинном обучении.

На основе сети проводится четкая классификация с дальнейшим обучением в рамках разработанной системы, которая обладает такими преимуществами, как высокая скорость обработки входных данных, стабильная работа при наличии ошибочных данных и высокая эффективность при решении задач классификации.

Результатами работы являются:

- моделирование и реализация LSTM сети, которая оценивает тональность текста на python под библиотеку TensorFlow 2;
- реализация python-скрипта (парсера) для сканирования и загрузки комментариев из социальной сети «Вконтакте», которые используются для тестирования модели;
- проведен эксперимент, который показывает большие перспективы в использовании искусственных нейронных сетей для распознавания негативного текста.

Можно с уверенностью сказать, что данная архитектура нейронной сети может быть применена для решения поставленной задачи.

В процессе обучения нейронная сеть сама определяет, какие признаки в данных важны, а какие нет. Сеть может подобрать такое сложное сочетание признаков, до которого человек никогда бы не догадался. Поэтому с помощью глубоких нейронных сетей удастся решать такие задачи, с которыми не справляются традиционные алгоритмы машинного обучения.

В целом, исследование показало, что ИНС в настоящее время являются наиболее эффективными и перспективными в задачах классификации, что подтверждается результатами исследования.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Новости Вконтакте. URL: <https://vk.com/press> (дата обращения 14.05.2021).
2. Гафаров, Ф. М. Искусственные нейронные сети и их приложения: учеб. пособие / Ф. М. Гафаров, А. Ф. Галимянов. – Казань: Издательство Казанского университета, 2018. – 121 с. [Электронный ресурс]. – URL: https://kpfu.ru/staff_files/F1493580427/NejronGafGal.pdf (дата обращения: 15.05.2021).
3. Галушкин А.И. Теория нейронных сетей. Кн. 1: Учеб. пособие для вузов / Общая ред. А.И.Галушкина. – М.: ИПРЖР, 2000. – 416 с.: ил. (Нейрокомпьютеры и их применение).
4. Simeon Kostadinov Understanding GRU Networks, 16 декабря 2017 [Электронный ресурс]: - URL: <https://towardsdatascience.com/understanding-gru-networks-2ef37df6c9be> (дата обращения: 19.05.2021).
5. Python 3.7 documentation. – URL: <https://docs.python.org/3/> (дата обращения 02.05.2021).
6. TensorFlow Core. – URL: <https://www.tensorflow.org/overview> (дата обращения 02.05.2021).
7. Scikit-learn Machine Learning in Python. – URL: <https://scikit-learn.org/stable/> (дата обращения 02.05.2021).
8. Theano docs. – URL: <http://deeplearning.net/software/theano/> (дата обращения 03.05.2021).
9. Russian_twitter_sentiment. URL: <https://www.kaggle.com/thorinhood/russian-twitter-sentiment> (Дата обращения 03.05.2021).
10. Савельев, А. В. На пути к общей теории нейросетей. К вопросу о сложности / А. В. Савельев // Нейрокомпьютеры: разработка, применение. – 2006. – № 4–5. – С. 4–14. Режим доступа <http://www.radiotec.ru/catalog.php?cat=jr7>.
11. Гречачин, В. А. К вопросу о токенизации текста / В.А. Гречачин // Международный научно-исследовательский журнал. – 2016. – №6. – Часть 4. –

С. 25–27. Режим доступа <http://research-journal.org/wp-content/uploads/2011/10/6-4-48.pdf#page=25> (открытый).

12. scikit-learn stable modules. – URL: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html (дата обращения 03.05.2021).

13. Embedding layer. – URL: <https://ru-keras.com/embedding-layers> (Дата обращения 06.05.2021).

14. Diederik P. Adam: A method for stochastic optimization / Diederik P. Kingma, Jimmy Lei. Ba. // International Conference on Learning Representations, 2015. – P. 1-15.

ПРИЛОЖЕНИЕ А

Программа для загрузки данных

(обязательно)

Программа была реализована на языке программирования Python 3.9 с использованием объектно-ориентированного программирования, разработана в ФГБОУ ВО ПГУ в г. Пенза. Данная программа может быть запущена на персональных компьютерах, на которых имеется интерпретатор Python версии 3+.

При запуске программы необходимо указать id группы, id записи (поста), количество записей, необходимых для загрузки, и количество постов для отступа (необязательный параметр).

```
import requests
import csv
import time

token
'34fd219234fd219234fd219244348ac672334fd34fd219254596d8e2e65e6cea9fbde03'
owner_id = '-29544671'
version = '5.92'
post_id = '7873994'
all_com = []
count = '5'
counter = 50

def pars_com():
    offset = 0
    while offset < counter:
        response = requests.get('https://api.vk.com/method/wall.getComments',
                                params={
                                    'access_token': token,
                                    'v': version,
                                    'owner_id': owner_id,
                                    'post_id': post_id,
                                    'count': count,
                                    'offset': offset
                                })
        data = response.json()['response']['items']
        offset += 5
        all_com.extend(data)
        time.sleep(1)
    return all_com
```

```
def file_writer(all_com):
    with open('file_neg.csv','w+', encoding='utf-8') as file:
        a_pen = csv.writer(file, delimiter = ',')
        for com in all_com:
            a_pen.writerow((com['text']))

all_com = pars_com()
print(all_com)
file_writer(all_com)
```

ПРИЛОЖЕНИЕ Б

Программа для предобработки данных

(обязательно)

Программа была реализована на языке программирования Python 3.9 с использованием объектно-ориентированного программирования, разработана в ФГБОУ ВО ПГУ в г. Пенза. Данная программа может быть запущена на персональных компьютерах, на которых имеется интерпретатор Python версии 3+.

Начальными данными являются файлы `positive.xlsx`, `negative.xlsx` и `file_pars.xlsx`, в них хранятся позитивные, негативные и загруженные из социальной сети Вконтакте записи соответственно.

```
import pandas as pd
import re

from pymorphy2 import MorphAnalyzer

df_pos = pd.read_excel('positive.xlsx',
                      header=None,
                      names=['Review'])
df_neg = pd.read_excel('negative.xlsx',
                      header=None,
                      names=['Review'])
df_pars = pd.read_excel('file_pars.xlsx',
                      header=None,
                      names=['Review'])

df_pos = df_pos['Review']
df_neg = df_neg['Review']
df_pars = df_pars['Review']

morph = MorphAnalyzer()
def lemmatize(doc):
    tokens = []
    for token in doc.split():
        token = token.strip()
        token = morph.normal_forms(token)[0]
        tokens.append(token)
    if len(tokens) > 2:
        return tokens
    return None

data_pos = df_pos.apply(lemmatize)
```

```

data_neg = df_neg.apply(lemmatize)
data_pars = df_pars.apply(lemmatize)

for i in range(len(data_pos)):
    data_pos[i] = re.sub('[^А-Яа-я\s]', '', str(data_pos[i]))
    data_pos[i] = re.sub('\s+', ' ', str(data_pos[i]))

for i in range(len(data_neg)):
    data_neg[i] = re.sub('[^А-Яа-я\s]', '', str(data_neg[i]))
    data_neg[i] = re.sub('\s+', ' ', str(data_neg[i]))

for i in range(len(data_pars)):
    data_pars[i] = re.sub('[^А-Яа-я\s]', '', str(data_pars[i]))
    data_pars[i] = re.sub('\s+', ' ', str(data_pars[i]))

data_pos.to_excel('positive3.xlsx', index=False, header=False)
data_neg.to_excel('negative3.xlsx', index=False, header=False)
data_pars.to_excel('pars_pars3.xlsx', index=False, header=False)

```

ПРИЛОЖЕНИЕ В

Программа для создания и тестирования модели

(обязательно)

Программа была реализована на языке программирования Python 3.9 с использованием объектно-ориентированного программирования, разработана в ФГБОУ ВО ПГУ в г. Пенза. Данная программа может быть запущена на персональных компьютерах, на которых имеется интерпретатор Python версии 3+.

Начальными данными являются выходные данные из ПРИЛОЖЕНИЯ Б.

После запуска программы происходит обучение, вывод результатов.

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Embedding, LSTM
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.callbacks import ModelCheckpoint
from sklearn.model_selection import train_test_split
import pandas as pd
import matplotlib.pyplot as plt

num_words = 6000
max_review_len = 20

train = pd.read_excel('train_itog5.xlsx',
                      header=None,
                      names=['Review', 'Class'])

reviews = train['Review']

tokenizer = Tokenizer(num_words=num_words)
tokenizer.fit_on_texts(reviews)
sequences = tokenizer.texts_to_sequences(reviews)

x = pad_sequences(sequences, maxlen=max_review_len)
y = train['Class'] - 1

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.3)

model = Sequential()
model.add(Embedding(num_words,8, input_length=max_review_len))
model.add(LSTM(32))
model.add(Dense(1, activation='sigmoid'))
```

```

model.compile(optimizer='adam',
              loss='binary_crossentropy',
              metrics=['accuracy'])

model_save_path = 'best_model.h5'
checkpoint_callback = ModelCheckpoint(model_save_path,
                                     monitor='val_accuracy',
                                     save_best_only=True,
                                     verbose=1)

history = model.fit(x_train,
                  y_train,
                  epochs=2,
                  batch_size=128,
                  validation_split=0.1,
                  callbacks=[checkpoint_callback])

plt.plot(history.history['accuracy'],
         label='Доля верных ответов на обучающем наборе')
plt.plot(history.history['val_accuracy'],
         label='Доля верных ответов на проверочном наборе')
plt.xlabel('Эпоха обучения')
plt.ylabel('Доля верных ответов')
plt.legend()
plt.show()

print('\n\n', "Тестовый набор:")
model.evaluate(x_test, y_test, verbose=1)
print('\n')

parsing = pd.read_excel('pars3.xlsx',
                      header=None,
                      names=['obr', 'orig'])

pars_obr = parsing['obr']
pars_orig = parsing['orig']

for i in range(len(pars_obr)):
    sequense = tokenizer.texts_to_sequences([pars_obr[i]])
    data = pad_sequences(sequense, maxlen=max_review_len)
    result = model.predict(data)
    #print('Оригинальный комментарий: ', pars_orig[i])
    #print('Обработанный комментарий: ', pars_obr[i])
    #print(result)
    if result >= 0.5:
        print('Позитивный комментарий')
    else:
        print('Негативный комментарий')
    #print('-----\n')
def proverka(qwe):
    summa = 0

```

```

for i in range(len(qwe)):
    temp = model.predict(qwe[i])
    summa += temp
summa = summa / len(qwe)
print(summa)

pars = pd.read_excel('pars_poz.xlsx',
                    header=None,
                    names=['poz'])
pars_poz = pars['poz']

print('Набор комментариев из позитивной группы:')
for i in range(len(pars_poz)):
    sequense = tokenizer.texts_to_sequences([pars_poz[i]])
    data = pad_sequences(sequense, maxlen=max_review_len)
    result += model.predict(data)
print(result / len(pars_poz))
result = 0

pars = pd.read_excel('pars_neg.xlsx',
                    header=None,
                    names=['neg'])
pars_neg = pars['neg']

print('Набор комментариев из негативной группы:')
for i in range(len(pars_neg)):
    sequense = tokenizer.texts_to_sequences([pars_neg[i]])
    data = pad_sequences(sequense, maxlen=max_review_len)
    result += model.predict(data)
print(result / len(pars_neg))

```