Министерство науки и высшего образования Российской Федерации

ФГБОУ ВО «Кубанский государственный технологический университет»

(ФГБОУ ВО КубГТУ)

Институт компьютерных систем и информационной безопасности

Кафедра информационных систем и программирования

Направление подготовки 09.04.03 Прикладная информатика

Профиль Аналитические информационные системы

**КУРСОВОЙ ПРОЕКТ**

по дисциплине Нейронные сети и глубокое обучение\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

на тему Классификация текстов с помощью нейронных сетей

Выполнил студент Ручка Артем Алексеевич\_курса 2 группы 19-КМ-ПИ1

Допущен к защите\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Руководитель (нормоконтролер) работы\_\_\_\_\_\_\_\_\_\_\_ доцент, к.т.н.Шумков Е.А.

Защищен \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Члены комиссии: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Краснодар

2020

ФГБОУ ВО «Кубанский государственный технологический университет»

(ФГБОУ ВО КубГТУ)

Институт компьютерных систем и информационной безопасности

Кафедра информационных систем и программирования

Направление подготовки 09.04.03 Прикладная информатика

Профиль Аналитические информационные системы

УТВЕРЖДАЮ

Зав. кафедрой \_\_\_\_\_\_\_М.В. Янаева

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_ г.

**ЗАДАНИЕ**

на курсовое проектирование

Студенту Ручка Артему Алексеевичу группы 19-КМ-ПИ1 курса 2

Тема проекта: Классификация текстов с помощью нейронных сетей

Объем работы:

а) пояснительная записка \_\_\_ с.

Рекомендуемая литература

1. Zhang, X. Character-level convolutional networks for text classification / Xiang Zhang, Junbo Zhao, Yann LeCun // In Advances in Neural Information Processing Systems. — 2015. — Feb. — 649 - 657 p.

Срок выполнения: с «\_\_\_» \_\_\_\_\_\_\_\_\_по «\_\_\_»\_\_\_\_20\_\_г.

Срок защиты: «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_г.

Дата выдачи задания: «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_г.

Дата сдачи проекта на кафедру: «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_20\_\_г.

Руководитель работы \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_доцент, к.т.н.Шумков Е.А..

Задание принял студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Ручка А.А.

**Реферат**

Пояснительная записка курсового проекта - 26 с., 22 рис., 1 таблица, 6 формул, 5 источников.

НЕЙРОННАЯ СЕТЬ, СВЕРТОЧНЫЕ НЕЙРОННЫЕ СЕТИ, КЛАССИФИКАЦИЯ ТЕКСТА, PYTHON, PYTORCH, СЛОЙ

Объектом исследования являются сверточные нейронные сети для классификации текстов.

Целью работы является исследование применения сверточных нейронных сетей для классификации текстов.

К полученным результатам относится программа на языке программирования Python, которая реализует сверточную нейронную сеть, благодаря которой возможна классификация текста по тональности.

Содержание

[Введение 5](#_Toc56545369)

[1 Нормативные ссылки 6](#_Toc56545370)

[2 Термины и определения 7](#_Toc56545371)

[3 Описание предметной области 8](#_Toc56545372)

[3.1 Задача анализа тональности текста 8](#_Toc56545373)

[3.2 Сверточные нейронные сети 8](#_Toc56545374)

[3.2.1 Свертки 9](#_Toc56545375)

[3.2.2 Полносвязный слой 10](#_Toc56545376)

[3.2.3 Выходной слой 12](#_Toc56545377)

[3.2.4 Вычисление потерь 13](#_Toc56545378)

[3.2.5 Adam 14](#_Toc56545379)

[3.3 PyTorch 15](#_Toc56545380)

[4 Программирование нейронной сети 16](#_Toc56545381)

[4.1 Подготовка данных 16](#_Toc56545382)

[4.2 Архитектура нейронной сети 18](#_Toc56545383)

[4.3 Обучение модели 19](#_Toc56545384)

[4.4 Оценка модели 22](#_Toc56545385)

[4.5 Проверка нейронной сети 23](#_Toc56545386)

[Заключение 25](#_Toc56545387)

[Список использованных источников 26](#_Toc56545388)

Введение

Ввиду роста количества текстовой информации повсеместно, а особенно в интернете, все большую роль играет возможность классифицировать её, отбирать лишь актуальную её часть – отсюда и возникают сопутствующие задачи машинного обучения.

Одной из широко используемых задач обработки естественного языка и машинного обучения в различных бизнес-задачах является классификация текста.

Цель классификации текста - автоматически классифицировать текстовые документы по одной или нескольким предопределенным категориям.

Например, классификацию текста используют для следующих задач:

* анализ настроений аудитории в социальных сетях;
* обнаружение спама;
* классификация новостных статей по темам.

Задача классификации текстов — одна из самых актуальных для маркетологов, особенно когда речь идет об анализе мнений или отношения потребителя к какому-то товару или услуге, поэтому исследователи постоянно работают над повышением качества ее решения. Однако анализ мнений является задачей классификации скорее предложений, а не текстов — в положительном отзыве пользователь может написать одно-два отрицательно окрашенных предложения, и их тоже важно уметь определять и анализировать.

Целью данной работы является анализ структуры сверточной нейронной сети и разработка сверточной нейронной сети на языке программирования Python для классификации текста по настроению.

1 Нормативные ссылки

В данном курсовом проекте использовались следующие нормативные ссылки:

1. ГОСТ Р 1.5-2004 Стандарты национальные Российской Федерации. Правила построения, изложения, оформления и обозначения;

2. ГОСТ 2.104-68 ЕСКД. Основные надписи;

3. ГОСТ 7.80-2000 СИБИД. Библиографическая запись. Заголовок. Общие требования и правила составления;

4. ГОСТ 7.82-2001 СИБИД. Библиографическая запись. Библиографическое описание электронных ресурсов. Общие требования и правила составления;

5. ГОСТ 7.9-95 СИБИД. Реферат и аннотация. Общие требования;

6. ГОСТ 19.001-77 ЕСПД. Общие положения;

7. ГОСТ 19.104-78 ЕСПД. Основные надписи;

8. ГОСТ 19.105-78 ЕСПД. Общие требования к программным документам;

9. ГОСТ 19.404-79 ЕСПД. Пояснительная записка. Требования к содержанию и оформлению;

10. ГОСТ 19781-90. Обеспечение систем обработки информации программное. Термины и определения.

2 Термины и определения

Сверточная нейронная сеть (англ. convolutional neural network, CNN) — специальная архитектура нейронных сетей, предложенная Яном Лекуном, изначально нацеленная на эффективное распознавание изображений.

Кернел — это обычная матрица чисел, называемых весами, которые “обучаются” (подстраиваются) с целью поиска на изображениях определенных характеристик.

Те́нзор (от лат. tensus, «напряжённый») — объект линейной алгебры, линейно преобразующий элементы одного линейного пространства в элементы другого.

Adam - ​​это алгоритм оптимизации, который можно использовать вместо классической процедуры стохастического градиентного спуска для итеративного обновления весов сети на основе обучающих данных.

PyTorch —это фреймворк машинного обучения, созданный на базе Torch, для языка Python с открытым исходным кодом.

Токенизация, лексический анализ («токенизация», от англ. tokenizing) — процесс аналитического разбора входной последовательности символов на распознанные группы — лексемы — с целью получения на выходе идентифицированных последовательностей, называемых «токенами».

Токен  —  это последовательность символов в документе, имеющая значение для анализа.

3 Описание предметной области

3.1 Задача анализа тональности текста

Анализ тональности текста (сентимент-анализ, англ. Sentiment analysis, англ. Opinion mining) — класс методов контент-анализа в компьютерной лингвистике, предназначенный для автоматизированного выявления в текстах эмоционально окрашенной лексики и эмоциональной оценки авторов (мнений) по отношению к объектам, речь о которых идёт в тексте.

Основной целью анализа тональности является нахождение мнений в тексте и выявление их свойств. Какие именно свойства будут исследоваться, зависит уже от поставленной задачи.

Примеры тональных оценок:

* позитивная;
* негативная;
* нейтральная.

В данной работе классификация текста проводится для анализа пользовательских мнений к фильмам. По тональности мнения будем делить на позитивные и негативные.

3.2 Сверточные нейронные сети

Сверточные нейронные сети, первоначально не предназначались для работы с текстом, они использовались в «компьютерном зрении» и распознавании образов. Сверточная нейронная сеть — это особый вид нейронных сетей прямого распространения. Под прямым распространением понимается то, что распространение сигналов по нейронам идет по порядку, от первого слоя до последнего. Скрытых слоев в сети может быть достаточно много, всё зависит от количества данных и сложности задачи(рисунок 1).



Рисунок 1 – Структура сверточной нейронной сети

Теперь по отдельности рассмотрим компоненты, формирующие сверточную нейронную сеть. После рассмотрения каждого компонента мы запрограммируем нейронную сеть на Python с использованием библиотеки PyTorch и обучим ее.

3.2.1 Свертки

Сверточные нейронные сети работают на основе фильтров, которые занимаются распознаванием определенных характеристик изображения (например, прямых линий). Фильтр — это коллекция кернелов; иногда в фильтре используется один кернел. Фильтр перемещается вдоль изображения и определяет, присутствует ли некоторая искомая характеристика в конкретной его части. Для получения ответа такого рода совершается операция свертки, которая является суммой произведений элементов фильтра и матрицы входных сигналов.

Основной особенностью таких сетей является наличие чередующихся слоев типа «свертка — субдискретизация», которых может быть множество. Операция свертки (рисунок 2) подразумевает, что каждый фрагмент входа поэлементно умножается на небольшую матрицу весов (ядро), а результат суммируется. Эта сумма является элементом выхода, который называется картой признаков. Взвешенная сумма входов пропускается через функцию активации.

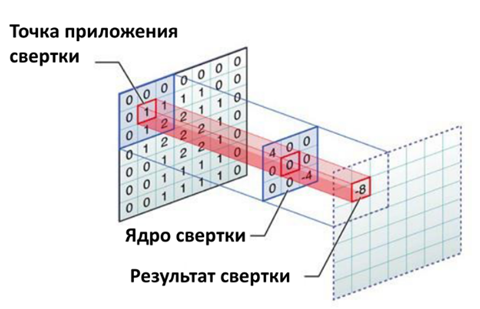


Рисунок 2 – Операция свертки

Если некоторая искомая характеристика присутствует во фрагменте изображения, операция свертки на выходе будет выдавать число с относительно большим значением. Если же характеристика отсутствует, выходное число будет маленьким.

3.2.2 Полносвязный слой

На данном этапе преобразования данных трехмерная матрица сигналов будет развернута в вектор, который будет пропущен через полносвязную нейронную сеть. Ее задача заключается в том, чтобы сообщить нам вероятности того, какую цифру представляет входной образ. Рисунок 3 демонстрирует эту операцию.

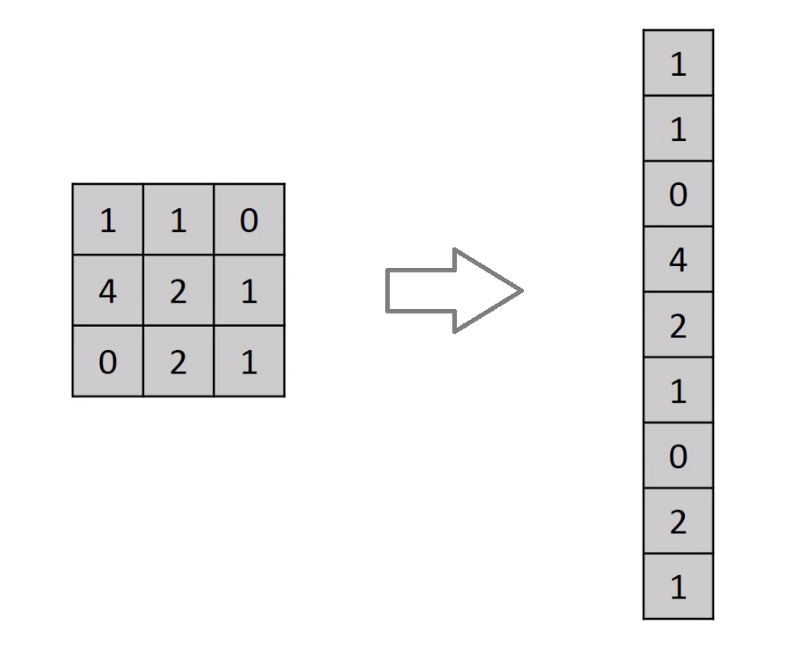


Рисунок 3 - Развертывание двумерной матрицы в вектор.

Из рисунка видно, что операция развертывания заключается в “склеивании” строк в единый — огромной длины — числовой ряд.

Принцип работы полносвязного слоя: каждый элемент вектора умножается на вес связи, эти произведения далее суммируются между собой и с некоторым смещением, после чего результат подвергается преобразованию с помощью функции активации. На рисунке 4 описанное представлено в наглядной форме.

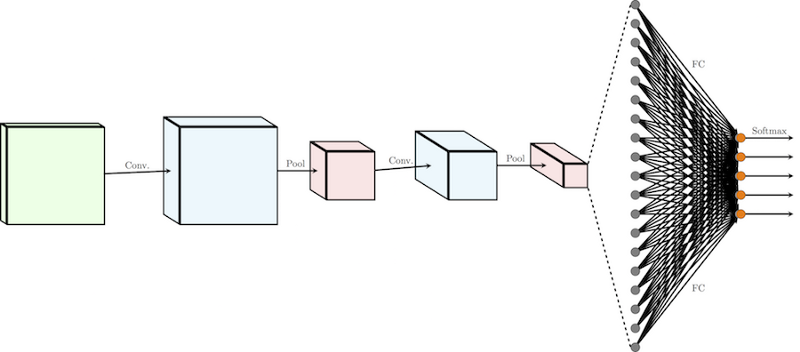


Рисунок 4 - Взаимодействие сверточных слоев с многослойным перцептроном

3.2.3 Выходной слой

Выходной слой отвечает за формирование вероятностей принадлежности входного образа тому или иному классу (некоторому числу). Чтобы добиться этого, выходной слой должен содержать количество нейронов, соответствующих количеству классов. Взвешенные и просуммированные сигналы далее модифицируются с помощью функции активации.

Нелинейность представляет из себя функцию активации. Благодаря ней картина, формируемая с помощью операции свертки, получает некоторое искажение, позволяющее нейронной сети более ясно оценивать ситуацию. Эту необходимость очень грубо можно сравнить с необходимостью людям со слабым зрением носить контактные линзы. А вообще-то такая необходимость связана с тем, что входные данные по своей природе нелинейны, поэтому нужно умышленно искажать промежуточные результаты, чтобы ответ нейронной сети был соответствующим.

В нашей работе в качестве функции активации будет использована функция ReLU (Rectified Linear Unit). Ее график изображен на рисунке 5.



Рисунок 5 - Функция ReLU

Как вы можете видеть, эта функция довольно-таки проста. Входные значения, меньшие или равные нулю, превращаются в нуль; значения, превышающие нуль, не изменяются.

3.2.4 Вычисление потерь

Чтобы определить, насколько точно нейронная сеть работает, мы будем использовать функцию потерь. Ответом функции потерь является вещественное число, характеризующие качество ответа нейронной сети. Для оценки качества классификаторов мы будем использовать кросс-энтропийную функцию потерь.

(1)

В формуле 1 ŷ — фактический ответ нейронной сети, y — желаемый ответ нейронной сети. Ответом в нашем случае является некоторое число; в более широком смысле ответ представляет категорию. Для получения общего показателя потерь, характерного для всех категорий в целом, берут среднее от значений по каждой категории.

3.2.5 Adam

В качестве алгоритма оптимизации весов в нашей работе выбран алгоритм Adam.

Авторы описывают Адама как объединение преимуществ двух других расширений стохастического градиентного спуска. В частности:

* Адаптивный Градиентный Алгоритм (AdaGrad), который поддерживает скорость обучения по параметру, которая улучшает производительность при проблемах с разреженными градиентами (например, проблемы с естественным языком и компьютерным зрением).
* Среднеквадратичное распространение (RMSProp), которое также поддерживает скорости обучения по каждому параметру, которые адаптированы на основе среднего значения последних величин градиентов для веса (например, как быстро оно изменяется). Это означает, что алгоритм хорошо справляется с онлайн и нестационарными задачами (например, с шумом).

Правило обновления весов для Adam определяется на основе использования оценок двух различных моментов (формулы 2 и 3), в первом из которых используются вычисленные ранее значения частных производных, а во втором их квадраты (как в RMSProp). Метод Adam считается довольно устойчивым к выбору значений гиперпараметров β1 и β2, и поэтому часто предлагается в качестве метода по умолчанию.

, (2)

. (3)

Вычисленные моменты корректируются по формулам 4 и 5, а затем производится пересчет весов по формуле 6.

, (4)

, (5)

. (6)

3.3 PyTorch

Для построения нейронной сети в нашей работе будет использован модуль Python под названием PyTorch. Он используется для решения различных задач: компьютерное зрение, обработка естественного языка. Разрабатывается преимущественно группой искусственного интеллекта Facebook.

PyTorch предоставляет две основные высокоуровневые модели:

* Тензорные вычисления (по аналогии с NumPy) с развитой поддержкой ускорения на GPU
* Глубокие нейронные сети на базе системы autodiff

Приступим к программированию нейронной сети.

4 Программирование нейронной сети

Для построения сверточной нейронной сети и обработки данных были использованы следующие пакеты: torch, random, numpy, spacy, time, matplotlib. Дадим им краткую характеристику:

* Torch – пакет, включающий в себя набор необходимых инструментов для построения нейронной сети и работы с нею;
* Random - предоставляет функции для генерации случайных чисел, букв, случайного выбора элементов последовательности;
* Numpy - то библиотека языка Python, добавляющая поддержку больших многомерных массивов и матриц, вместе с большой библиотекой высокоуровневых (и очень быстрых) математических функций для операций с этими массивами;
* Spacy - это библиотека для расширенной обработки естественного языка;
* Time - модуль для работы со временем в Python;
* Matplotlib -  библиотека на языке программирования Python для визуализации данных двумерной графикой.

4.1 Подготовка данных

Для обучения и тестирования модели мы использовали данные о пользовательских отзывах к фильмам с сайта IMDB. Набор данных содержит 25000 записей для обучения и 25000 записей для тестирования(рисунок 6). Все данные обучающего и тестового набора данных разделены на позитивные и негативные отзывы.

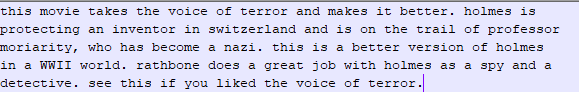


Рисунок 6 – Пример положительного пользовательского отзыва, до обработки

Обработка данных для анализа начинается с токенизации. Загружаем данные в проект и преобразовываем их в тензоры, используя токенизатор Spacy(рисунок 7). Он разделяет все тексты на отдельные слова.

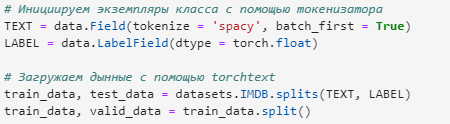


Рисунок 7 – Загрузка данных

Для того чтобы проанализировать данные с помощью нейронной сети создаем словарь(рисунок 8), в котором будут находиться слова в численном представлении. Каждое слово будет кодироваться своим числовым вектором, который будет подаваться на вход нейронной сети. Словарь состоит из всех слов, присутствующих в нашем тестовом наборе данных. В качестве векторов используются данные GloVe, которые представляют собой предварительно обученные вложения, поэтому нам не нужно обучать собственные векторы слов с нуля.

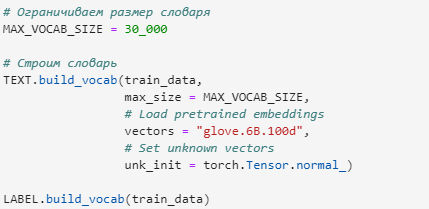


Рисунок 8 – Строим словарь

Подготовка данных завершена, теперь приступаем к созданию нейронной сети.

4.2 Архитектура нейронной сети

Создадим сверточную нейронную сеть с двоичным классификатором(рисунки 9-10). Наша модель будет основана на встроенном в PyTorch nn.Module. Он представляет собой базовый класс для всех частей нейронной сети.

В данном модуле мы задаем архитектуру сверточной нейронной сети – сверточный слой, полносвязный слой, слой активации, слой субдискретизации и слой регуляризации.



Рисунок 9 – Программируем структуру сети(часть 1)



Рисунок 10 - Программируем структуру сети(часть 2)

После того как мы описали структуру нейронной сети инициализируем предварительно обученную модель GloVe(рисунок 11).

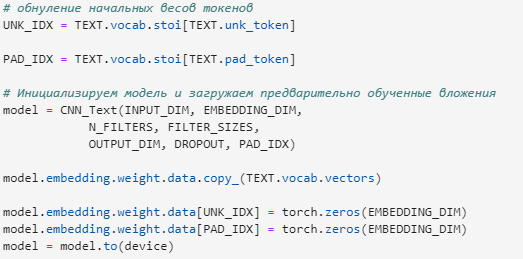


Рисунок 11 – Инициализация предварительно загруженных данных

На этом построение архитектуры сверточной нейронной сети окончено, переходим к обучению и тестированию построенной модели.

4.3 Обучение модели

Перед обучением модели нам нужно указать оптимизатор сети и функцию потерь(рисунок 12). Adam и бинарная кросс-энтропия - популярные методы при решении задач классификации. Для того чтобы обучить нашу модель, мы получаем прогнозы модели, вычисляем, насколько они точны, используя функцию потерь, и выполняем обратное распространение по сети с помощью Adam, чтобы оптимизировать веса перед следующим запуском.

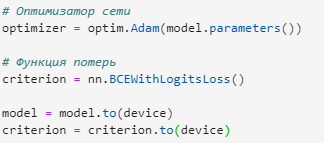


Рисунок 12 – Задаем оптимизатор сети и функцию потерь

Теперь запускаем обучение(рисунок 13). Для обучения мы задали 10 эпох. В результате завершения каждой эпохи будут выданы точность и потери для каждой эпохи(рисунок 14).



Рисунок 13 – Код для запуска обучения модели

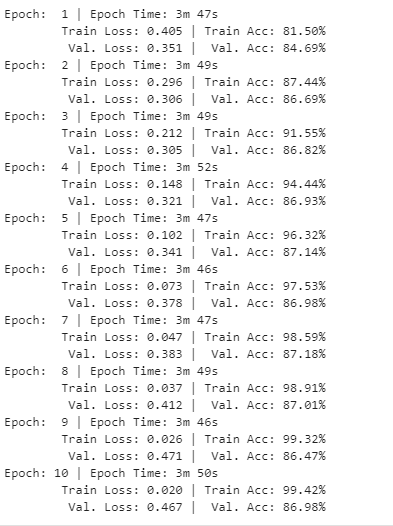


Рисунок 14 – Результаты обучения

Для обучения модели используем функцию train(рисунок 15). В ней вычисляется значение точности и потерь нейронной сети.



Рисунок 15 – Функция обучения

Для оценки используем функцию evaluate(рисунок 16). Здесь мы отключаем выпадение данных и вычисление градиента для получения верных результатов.



Рисунок 16 – Функция оценки

Теперь отобразим результат точность и потери на графике. Для этого воспользуемся matplotlib(рисунки 17-18).

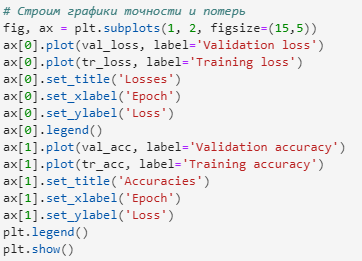


Рисунок 17 – Код для построения графика

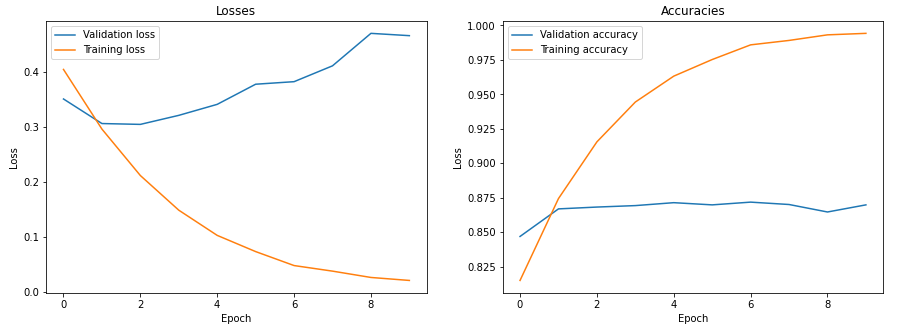


Рисунок 18 – Графиченское отображение результатов обучения

Как видно из графика наша модель показывает признаки переобучения. Чтобы избежать использования переобученной модели, мы сохраняем модель только в случае увеличения ошибки проверки. В нашем случае ошибка проверки была максимальной после третьей эпохи.

4.4 Оценка модели

Теперь оценим модель на тестовых данных(рисунок 19).

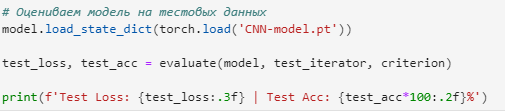


Рисунок 19 – Проведение оценки

Как видно в результате мы получили точность 85.18% что довольно неплохо(рисунок 20).



Рисунок 20 – Результаты тестирования

4.5 Проверка нейронной сети

Оценим настроения некоторых пользовательских отзывов, используя построенную модель CNN(рисунок 21). Для этого нам нужно написать функцию, которая токенизирует ввод пользователя и превращает его в тензор.

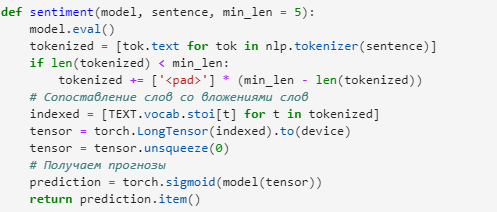


Рисунок 21 – Функция оценки

После этого проводим оценку, используя только что обученную модель(рисунок 22).

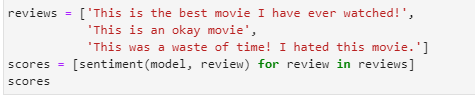


Рисунок 22 – Проведение оценки

В результате оценки получены следующие результаты, указанные в таблице 1. Пограничным критерием оценки является 0,5. Соответственно если оценка меньше 0,5, то отзыв негативный, а если выше – позитивный. Как видно из результатов модель работает довольно точно.

Таблица 1 – Результаты тестирования

|  |  |  |
| --- | --- | --- |
| **Отзыв** | **Оценка** | **Результат** |
| This is the best movie I have ever watched! | 0.9771446585655212 | Положительный |
| This is an okay movie. | 0.35399019718170166 | Отрицательный |
| This was a waste of time! I hated this movie. | 0.02346784807741642 | Отрицательный |

Заключение

В курсовой работе были рассмотрены теоретические основы сверточных нейронных сетей, и была запрограммирована сверточная нейронная сеть для решения задачи классификации текстов. Для реализации данной задачи была использована библиотека PyTorch для Python, которая позволила максимально быстро и просто создать сверточную нейронную сеть для нашей задачи.

Реализованная сверточная нейронная сеть включает себя сверточный, субдискретизирующий, полносвязный слои, а также слой активации, слой регуляризации и выходной слой. В качестве функции активации была использована функция ReLU. Для оптимизации был использован алгоритм Adam, а для вычисления оценки качества классификаторов – кросс-энтропийная функция.

Запрограммированная нейронная сеть позволяет анализировать тексты по настроению и может быть использована для любых других задач классификации текста по настроению. Для проверки работы модели были использованы несколько пользовательских отзывов. В результате проверки модель справилась со своей задачей. Полученная модель может быть использована для оценки пользовательских мнений к каким-либо продуктам.

В результате выполнения курсового проекта были решены все поставленные задачи.

Список использованных источников

1. Zhang, X. Character-level convolutional networks for text classification / Xiang Zhang, Junbo Zhao, Yann LeCun // In Advances in Neural Information Processing Systems. — 2015. — Feb. — 649 - 657 p.
2. Yann LeCun Leon Bottou, Y. B. Gradient-based learning applied to document recognition / Yoshua Bengio Yann LeCun, Leon Bottou, Patrick Haffner // IEEE. — 1998.
3. Kim, Y. Convolutional neural networks for sentence classification / Yoon Kim // IEMNLP. — 2014. — Sep. — 1746 -1751 p.
4. Методы оптимизации нейронных сетей [Электронный ресурс]. URL: <https://habr.com/ru/post/318970/>
5. Сверточные нейронные сети [Электронный ресурс]. URL: <http://neerc.ifmo.ru/wiki/index.php?title=Сверточные_нейронные_сети>