Жбанов В. А., Абарникова Е. Б.
V. A. Zhbanov, E. B. Abarnikova

ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА МОДЕЛИ НЕЙРОННОЙ СЕТИ ДЛЯ ОПРЕДЕЛЕНИЯ СХОДСТВА ДВУХ ОБРАЗЦОВ НЕСТРУКТУРИРОВАННЫХ ДАННЫХ

DESIGN AND DEVELOPMENT OF A NEURAL NETWORK MODEL FOR DETERMINING THE SIMILARITY OF TWO SAMPLES OF NON-STRUCTURED DATA

Жбанов Валерий Александрович – магистр, студент кафедры «Проектирование, управление и разработка информационных систем» Комсомольского-на-Амуре государственного университета (Россия, Комсомольск-на-Амуре).
Valery A. Zhbanov – Master's Degree Student, Design, Management and Development of Information Systems Department, Komsomolsk-na-Amure State University (Russia, Komsomolsk-on-Amur).

Абарникова Елена Борисовна – кандидат технических наук, доцент кафедры «Проектирование, управление и разработка информационных систем» Комсомольского-на-Амуре государственного университета (Россия, Комсомольск-на-Амуре).
Elena B. Abarnikova – PhD in Engineering, Associate Professor, Design, Management and Development of Information Systems Department, Komsomolsk-na-Amure State University (Russia, Komsomolsk-on-Amur).

Аннотация. Данная работа посвящена проектированию и разработке модели нейронной сети для определения сходства двух образцов неструктурированных данных.

Summary. This work is devoted to the design and development of a neural network model to determine the similarity of two samples of unstructured data.

Ключевые слова: наука о данных, глубокое обучение, нейронные сети.

Key words: data science, deep learning, neural networks.

УДК 004.4

На текущий момент существует огромное количество ресурсов, на которых люди пишут свои мысли: комментарии, публикации, статьи и многое другое.

Но в сегменте сетевой литературы существует серьёзная проблема. Если за качеством печатных книг следят редакторы издательств, то за качеством сетевой литературы следить некому – на большинстве литературных ресурсов отсутствуют редакторы и/или модераторы, поскольку владельцам ресурсов невыгодно держать отдельный редакторский штат.

В свою очередь большая часть читающей молодёжи предпочитает печатным книгам электронные ресурсы; низкое качество литературы формирует у них неправильное представление о русской литературной традиции, русском языке в целом и литературной речи в частности.

Возможным решением задачи премодерации текста является создание программного обеспечения «Классификатор текстов», которое предназначено для анализа текстов, проверки соответствия сюжета, оригинальности, обнаружения провокаций, оскорблений, нарушений речевых и стилистических норм.

Структура и функциональные возможности данного программного обеспечения подробно описаны в статье «Разработка библиотеки для обработки неструктурированных текстов» [7].

Задачи программного обеспечения решаются алгоритмами классификации текста. На практике было обнаружено, что стандартный подход к решению задач классификации текстов, описание которого было представлено в статье «Анализ моделей нейронных сетей для классификации неструктурированных текстов» [6], не может использоваться в задачах, где набор классов динами-

чески расширяется, т. к. это приводит к необходимости переобучения модели нейронной сети с учётом новых данных.

Таким образом, необходимо разработать модель, позволяющую определить сходство двух текстов и не требующую знания о конкретных классах образцов. Такая модель может быть использована для определения класса текста, если имеется образец текста, класс которого уже известен.

Для достижения поставленной цели сформулируем основные задачи исследования:

1. спроектировать архитектуру модели нейронной сети;
2. реализовать модель;
3. подготовить набор данных для обучения и тестирования;
4. обучить модель;
5. протестировать модель;
6. проанализировать результаты.

Объектом исследования является обработка неструктурированных данных при помощи технологий искусственного интеллекта.

Предметом исследования является программное обеспечение «Классификатор текстов».

Научная новизна обусловлена следующими параметрами:

1. предложен новый подход к решению задачи классификации;
2. предложена и программно реализована новая архитектура модели.

Практическая значимость определена применением результатов исследования:

1. для модерации сетевых ресурсов;
2. для автоматизации рабочих процессов цензоров или модераторов;
3. при исследованиях в других направлениях искусственного интеллекта.

Теоретической основой исследования послужили работы российских и зарубежных авторов [3; 4; 5].

Модель для решения данной задачи должна принимать на вход два векторизованных текста и возвращать единственное число с плавающей точкой – процентное сходство входных текстов.

На рис. 1 показана схема модели нейронной сети. Данные, полученные моделью, конкатенируются и попадают в слой кодирования, который превращает коды символов в векторы фиксированного размера, после чего полученная матрица попадает в рекуррентный слой, содержащий LSTM-модули, выход которого нормализуется и попадает в выходной слой.

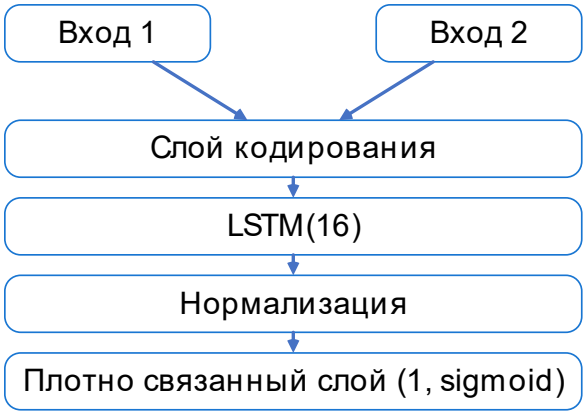

Рис. 1. Архитектура модели

Ниже приведён код на языке Python с применением библиотеки машинного обучения TensorFlow v2.8.0, реализующий описанную архитектуру:

```
# Вход для первого текста
encoder_inputs = keras.Input(shape=(sequence_length,), name="first_input")
```

```
# Вход для второго текста
decoder_inputs = keras.Input(shape=(sequence_length,), name="second_input")

# Объединение данных, полученных со входов модели
x = layers.Concatenate()([encoder_inputs, decoder_inputs])

# Кодирование данных, пребразование вектора в матрицу
x = layers.Embedding(vocab_size, 100)(x)

# Основной вычислительный слой
x = layers.LSTM(16)(x)

# Нормализация
x = layers.BatchNormalization()(x)

# Выходной слой модели
decoder_outputs = layers.Dense(1, activation="sigmoid")(x)

# Создание модели
model = keras.Model(
    [encoder_inputs, decoder_inputs], # Входы
    decoder_outputs                   # Выходы
)

# Компиляция модели
model.compile(
    optimizer=optimizers.Adam(), # Алгоритм обучения
    loss="mse",                  # Функция потерь
    metrics=["accuracy"]         # Дополнительные функции
)

# Вывод информации о модели
model.summary()
```

Для обучения модели был сгенерирован набор данных, состоящий из отрывков из трёх книг. В случае если отрывки из одной книги – они считаются схожими. Код для формирования набора данных приведён ниже:

```
# Загружаемые тексты
texts = []

# перебираем входные файлы
for file in files:

    # Открываем текущий файл для чтения
    with open(PATH + "Datasets/Fandoms/" + file, "rb") as file:

        # Получаем текст из файла
        text = file.read(-1).decode("UTF-8")
```

```
        print(len(text))

        # Сохраняем текст, обрезав его до определённой длины
        texts.append(text[:100000])

for text in texts:
    print(text[:100])

# Части набора данных
X1, X2, y = [], [], []

# Получает случайный отрывок текста
#    text: входной текст
def get_range(text):

    # Максимально возможная длина диапазона
    max_len = sequence_length

    # Минимальная длина диапазона
    min_len = int(0.5 * max_len)

    # Получаем случайную длину диапазона
    length = random.randint(min_len, max_len)

    # Вычисляем начала диапазона
    pos = random.randint(0, len(text) - length)

    # Возвращает кортеж с началом диапазона и его длиной
    return int(pos), int(length)

# Извлекает два соответствующих отрывка из двух входных текстов
#    first: первый текст
#    second: второй текст
def generate(first, second):

    # Получаем первый диапазон
    fpos, flen = get_range(first)

    # Получаем второй диапазон
    spos, slen = get_range(second)

    # Возвращаем кортеж из двух текстов
    return (first[fpos : fpos + slen],
            second[spos : spos + slen])

# Комбинации входных текстов
comb = itertools.combinations_with_replacement(range(len(texts)), 2)

# Последовательно перебираем комбинации текстов
for first, second in comb:
```

```
    # Если индексы равны, то тексты схожи – метка 1, иначе 0
    y_row = int(first == second)

    # Требуемое количество образцов
    count = 500

    # Берём требуемое количество образцов
    # из текущей комбинации текстов
    for it in range([count, (len(texts) - 1) * count][y_row]):

        # Расчёт новых данных
        x1_row, x2_row = generate(texts[first], texts[second])
        # Векторизация
        x1_row, x2_row = vectorizer(x1_row), vectorizer(x2_row)
        # Добавление новых строк в набор данных
        X1.append(x1_row)
        X2.append(x2_row)
        y.append(y_row)

# Разделение полученного набора данных
X1_train, X1_test, X2_train, X2_test, y_train, y_test = train_test_split(
    X1, X2, y, test_size=0.33, random_state=42)

# Набор данных для обучения
dataset_train = [{"first_input" : np.array(X1_train),
                  "second_input" : np.array(X2_train)},
                 np.array(y_train)]
# Набор данных для тестирования
dataset_test = [{"first_input" : np.array(X1_test),
                 "second_input" : np.array(X2_test)},
                np.array(y_test)]
```

Информация о процессе обучения приведена в табл. 1.

Таблица 1

История обучения

Этап обучения	Обучающая выборка		Тестовая выборка	
	Потери	Точность	Потери	Точность
10	0.2445	0.6133	0.2319	0.6714
20	0.1962	0.6793	0.2154	0.6552
30	0.1902	0.6706	0.2134	0.6718
40	0.1801	0.7061	0.2170	0.6751
50	0.1744	0.7118	0.1958	0.7030
60	0.1620	0.7456	0.1883	0.7145
70	0.1489	0.7695	0.1459	0.7181
80	0.1441	0.7861	0.1491	0.7377
90	0.1434	0.7900	0.1441	0.7467
100	0.1209	0.8292	0.1418	0.7809

На рис. 2 показан график изменений точности на тренировочной выборке. На рис. 3 показан график изменений функции потерь на тренировочной выборке.

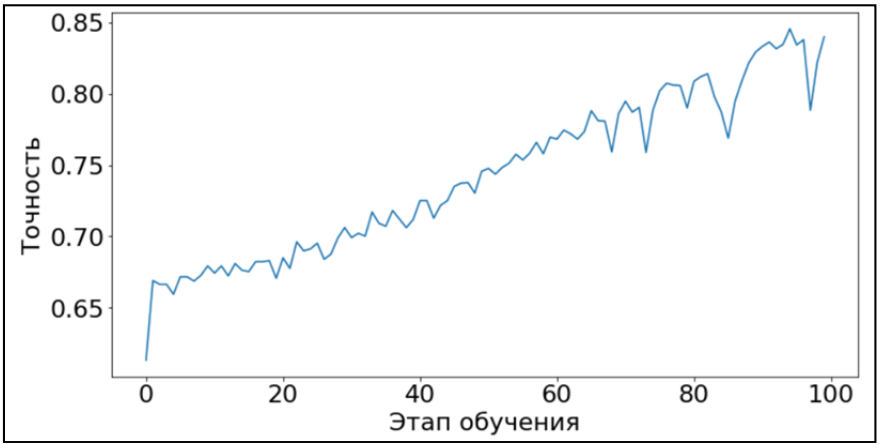

Рис. 2. График точности

Из графиков на рис. 2 и 3 видно, что в процессе обучения значения изменялись неравномерно, из чего можно сделать вывод о наличии в наборе данных выделяющихся из общего числа образцов. Для более корректной работы модели необходимо сформировать и проверить обучающий набор данных.

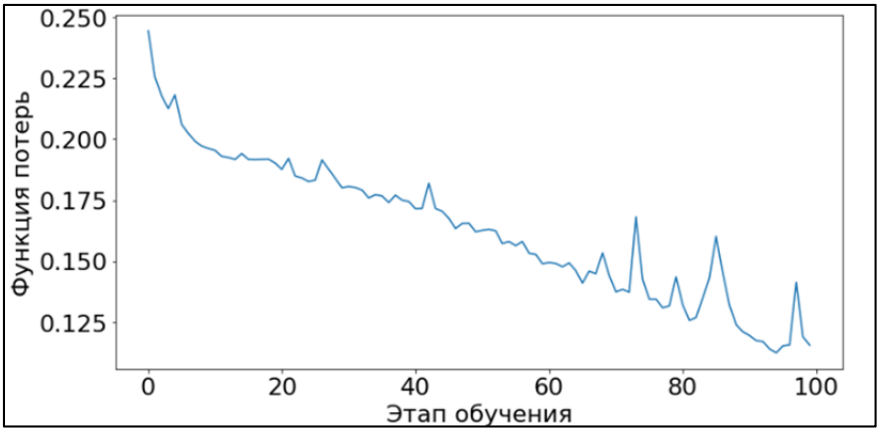

Рис. 3. График функции потерь

В результате работы была спроектирована и разработана модель, которая способна определить сходство двух неструктурированных текстов. Можно добиться большей точности модели, например применив более сложный алгоритм векторизации, такой как «Word2vec». С полным кодом проекта и результатами тестирования можно ознакомиться при помощи интерактивной облачной среды разработки Google Colab [1].

ЛИТЕРАТУРА

1. colab.research.google.com: Сравнение образцов: сайт. – United States, 2020 – . – URL: https://colab.research.google.com/drive/1GSdbbY2ODX6U7PYkW0kvA_Z5_JCK_K2b?usp=sharing (дата обращения: 06.10.2022). – Текст: электронный.
2. keras.io: Natural Language Processing: сайт. – United States, 2020 – . – URL: https://keras.io/examples/nlp/ (дата обращения: 06.10.2022). – Текст: электронный.
3. Komleva, E. V. About Digital Texts Significant In Text Classification / E. V. Komleva // The European Proceedings of Social and Behavioural Sciences, Оренбург, 19–20 сентября 2019 года. Vol. LXXXIII. – Оренбург: European Publisher, 2020. – P. 154-159. – DOI 10.15405/epsbs.2020.04.02.17. – EDN OLTUBF.
4. Potaraev, V. Analysis of relation types in semantic network used for text classification / V. Potaraev // Открытые семантические технологии проектирования интеллектуальных систем. – 2020. – № 4. – С. 305-308.

5. Бондаренко, В. И. Классификация научных текстов с помощью методов глубокого машинного обучения / В. И. Бондаренко // Вестник Донецкого национального университета. Серия Г: Технические науки. – 2021. – № 3. – С. 69-77.
6. Жбанов, В. А. Анализ моделей нейронных сетей для классификации неструктурированных текстов / В. А. Жбанов, Е. Б. Абарникова // Молодёжь и наука: актуальные проблемы фундаментальных и прикладных исследований: материалы V Всерос. нац. науч. конф. молодых учёных, Комсомольск-на-Амуре, 11-15 апреля 2022 года. В 4 ч. Ч. 1 / Редколлегия: А. В. Космынин (отв. ред.) [и др.]. – Комсомольск-на-Амуре: ФГБОУ ВО «КнАГУ», 2022. – С. 374-378.
7. Жбанов, В. А. Разработка библиотеки для обработки неструктурированных текстов / В. А. Жбанов, Е. Б. Абарникова // Наука, инновации и технологии: от идей к внедрению: материалы Международной научно-практической конференции, Комсомольск-на-Амуре, 07-11 февраля 2022 года. – Комсомольск-на-Амуре: ФГБОУ ВО «КнАГУ», 2022. – С. 17-19.
8. Омельяненко, Я. Эволюционные нейросети на языке Python: практическое руководство / Я. Омельяненко; пер. с англ. В. С. Яценкова. – Москва: ДМК Пресс, 2020. – 310 с.
9. Плас, Дж. Вандер. Python для сложных задач / Плас Дж. Вандер. – СПб.: Питер, 2018. – 576 с.