МИНОБРНАУКИ РОССИИ
федеральное государственное бюджетное образовательное учреждение высшего образования
«Национальный исследовательский университет «МЭИ»

Институт ИВТИ
Кафедра ВМСС

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (бакалаврская работа)

Направление 09.03.01 Информатика и вычислительная техника
(код и наименование)

Направленность (профиль) Вычислительные машины, комплексы, системы и сети

Форма обучения очная
(очная/очно-заочная/заочная)

Тема: Разработка информационной системы для автоматической рубрикации новостных текстов

Студент А-08-17 (группа) _____ (подпись) Челышев Э.А. (фамилия и инициалы)

Научный руководитель д.т.н. (уч. степень) доцент (должность) _____ (подпись) Оцоков Ш.А. (фамилия и инициалы)

Консультант _____ (уч. степень) _____ (должность) _____ (подпись) _____ (фамилия и инициалы)

Консультант _____ (уч. степень) _____ (должность) _____ (подпись) _____ (фамилия и инициалы)

«Работа допущена к защите»

Зав. кафедрой к.т.н. (уч. степень) доцент (звание) _____ (подпись) Вишняков С.В. (фамилия и инициалы)

Дата _____

Москва, 2021

АННОТАЦИЯ

В данной выпускной квалификационной работе выполнены проектирование и разработка информационной системы рубрикации русскоязычных текстов с использованием машинного обучения, состоящей из обученной модели классификации и веб-сайта. Представлен анализ существующих в машинной обработке естественного языка способов подготовки и векторизации текстов. Проведен ряд экспериментов по построению моделей классификации с использованием языка программирования Python. Построенные модели классификации оценены по ряду метрик. Для реализации веб-сайта были использованы язык программирования Python и фреймворк Django. Для хранения наполнения веб-сайта использовалась система управления базами данных MySQL. Работа состоит из 4 разделов, 94 страниц, 30 рисунков, 5 таблиц и 3 приложений.

ABSTRACT

This final qualifying paper describes design and development of an information system for the rubrication of Russian-language texts using machine learning, consisting of a trained classification model and a website. The analysis of the text preparation and vectorization methods, existing in natural language processing, is presented. A number of experiments have been carried out to build classification models using the Python programming language. The developed classification models are evaluated according to a number of metrics. To implement the website, the Python programming language and the Django framework were used. The database management system MySQL was used to store the content of the website. The work consists of 4 sections, 94 pages, 30 figures, 5 tables and 3 appendices.

СОДЕРЖАНИЕ

ОПРЕДЕЛЕНИЯ

В данной работе применяются следующие термины:

Анализ данных (англ. data science) — междисциплинарная область знаний, занимающаяся извлечением полезной информации, получением выводов из больших объемов данных и принятием на их основе решений [1].

Машинное обучение (англ. machine learning) — подраздел искусственного интеллекта, изучающий алгоритмы, способные обучаться, а также методы построения таких алгоритмов [2].

Машинная обработка естественного языка (англ. natural language processing, сокр. NLP) - это область машинного обучения, изучающая вопросы машинного анализа и автоматической генерации текстов на естественном языке [3].

Рубрикация (рубрицирование) – распределение некоторой информации по тематическим разделам (рубрикам).

Токен – текстовая единица (слово, предложение и т.п.) [4].

Документ – набор токенов, принадлежащих одной смысловой единице [4].

Корпус – совокупность документов [4].

Гиперпараметр в машинном обучении – параметр, посредством которого осуществляется контроль процесса обучения модели машинного обучения.

ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

ИНС – искусственная нейронная сеть.

НБК – наивный байесовский классификатор.

СУБД – система управления базами данных.

ORM – англ. Object-Relational Mapping, объектно-реляционное отображение.

URL – англ. Uniform Resource Locator, унифицированный указатель ресурса.

ВВЕДЕНИЕ

В последние десятилетия наблюдается быстрый рост объема производимых и накопляемых человечеством данных. Так, например, общемировой объем данных в 2018 году составил 33 зеттабайтов, а прогнозируемый общемировой объем данных к 2025 году составит уже 175 зеттабайтов [5], то есть вырастет более чем в пять раз.

Безусловно, с ростом накопленных данных человеку становится все сложнее ориентироваться в них, все более возрастает потребность в автоматизированной обработке информации. Именно этот факт является одной из ключевых причин роста популярности **анализа данных** и **машинного обучения**, позволяющих автоматически обрабатывать большие объемы данных и извлекать из них практическую пользу и коммерческую прибыль. Одной из отраслей машинного обучения является **машинная обработка естественного языка**. Машинная обработка естественного языка используется во многих сферах: машинный перевод, поиск информации, определение настроения и тональности, распознавание речи, прогнозирование и др. [3].

Весьма популярными становятся сейчас новостные агрегаторы, рубрикаторы научных статей и прочие решения по автоматизированной обработке информации, которые в своей работе используют автоматическую рубрикацию текстов на естественном языке. С точки зрения машинного обучения рубрикация является задачей классификации на несколько непересекающихся классов [6 - 7].

В данной работе описывается разработка информационной системы автоматической рубрикации русскоязычных новостных текстов, состоящей из обученной модели классификации, базы данных для хранения информации и веб-сайта для наглядной демонстрации полученных результатов.

Для построения и обучения моделей классификации использовался язык программирования высокого уровня **Python** и его стандартные библиотеки машинного обучения (**Pandas**, **NumPy**, **Scikit-learn** и др.), для создания веб-сайта – фреймворк **Django**, язык гипертекстовой разметки **HTML5** и каскадная таблица стилей **CSS**. В работе также использовалась система управления базами данных (СУБД) **MySQL**. Для написания программного кода на языке Python использованы интегрированные среды разработки **Jupyter Notebook** и **Spyder**.

В первой главе данной выпускной квалификационной работы ведется анализ предметной области, рассматриваются существующие новостные ресурсы, осуществляется постановка задачи.

Во второй главе излагаются критерии выбора, предъявляемые к средствам разработки, осуществляется обоснованный выбор языка программирования, проводится анализ фреймворков для создания веб-сайта, а также производится выбор системы управления базами данных.

В третьей главе приводится краткий анализ существующих методов подготовки и векторизации текстов для подачи их на вход модели классификации, осуществляется подготовка и векторизация корпуса, проводится ряд экспериментов по построению моделей классификации с дальнейшей оценкой их качества по ряду метрик классификации.

В четвертой главе описывается разработка и тестирование веб-сайта, а именно работа с базой данных, создание HTML-шаблонов, стилистическое оформление веб-сайта и его функционирование.

Заключение представляет собой подведение итогов выполненной выпускной квалификационной работы, результаты построения информационной системы автоматической рубрикации русскоязычных новостных текстов.

1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ И ПОСТАНОВКА ЗАДАЧИ

1.1. Анализ предметной области

Несмотря на обилие разнообразной информации, с которой современному человеку приходится регулярно взаимодействовать, он должен оставаться в курсе последних событий, происходящих в его стране и в мире в целом, и новостей из различных сфер общественной жизни. Безусловно, успешная ориентация человека в современном информационном поле требует современных решений. Для этих целей создаются новостные порталы и новостные агрегаторы, причем такие ресурсы должны соответствовать следующим требованиям:

- возможность предоставления доступа для широкой аудитории пользователей. Именно это требование объясняет то, что подобные ресурсы реализуются либо в виде веб-сайтов, либо в виде мобильных приложений, что позволяет любому пользователю, имеющему выход в сеть Интернет, посетить данный ресурс и ознакомиться с новостями;

- интуитивно понятный интерфейс и комфортное оформление сайта. Использование ресурса должно быть максимально удобным и не вызывать у пользователя напряжения и дискомфорта;

- удобство навигации по ресурсу и поиска в нем интересующих конкретного пользователя новостей, для чего целесообразно распределить новости по отдельным непересекающимся рубрикам. При необходимости пользователь может выбрать отдельную рубрику и ознакомиться только с теми новостями, которые относятся к данной рубрике. Учитывая неэффективность и малую скорость ручной рубрикации текстов, распределение новостей по рубрикам должно производиться автоматически;

- свежесть предоставляемой пользователю информации. Выдача новостного ресурса должна содержать на первых позициях самые свежие и актуальные новости, с уменьшением свежести и актуальности новости снижается и ее приоритет в списке выдачи ресурса.

Все приведенные выше требования являются важными при разработке любого новостного ресурса, невыполнение любого из них может обесценить всю проведенную разработку.

1.2. Обзор и анализ существующих новостных ресурсов

Рассмотрим новостной портал **LENTA.RU** [8] и новостной агрегатор **Яндекс.Новости** [9]. Оба новостных ресурса выполнены в виде веб-сайтов.

Заглавная страница новостного портала LENTA.RU представлена на рисунке 1. Как видно, на вершине выдачи веб-сайта ссылки на наиболее свежие и актуальные статьи. Интерфейс веб-сайта удобен и понятен. Для удобной навигации по порталу на каждой его странице присутствует боковая панель, содержащая меню, благодаря которому можно легко перейти на страницу, соответствующую отдельной интересующей пользователя рубрике. Среди особых достоинств данного новостного портала хочется отметить возможность регистрации на сайте с целью его персонализации.

На рисунке 2 представлена заглавная страница новостного агрегатора Яндекс.Новости. Данный новостной агрегатор аналогично предыдущему новостному ресурсу удобен в использовании и понятен, так же позволяет осуществлять навигацию по тематическим рубрикам с использованием меню. Персонализация новостного агрегатора возможна при помощи учетной записи Яндекса.

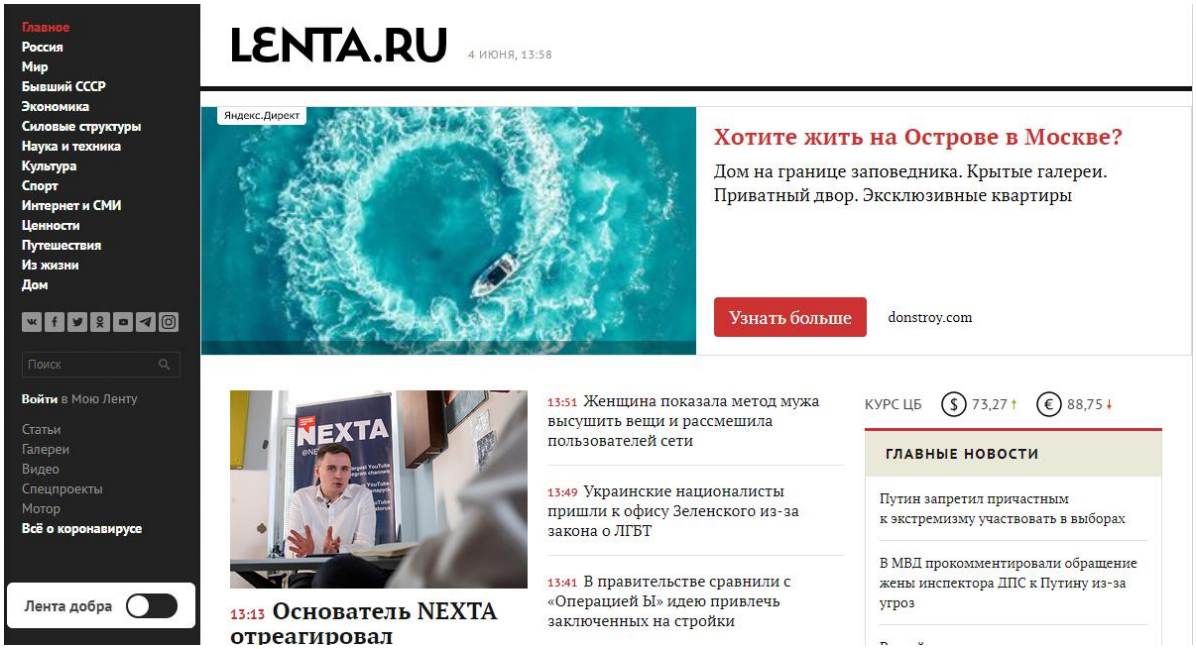

Рис. 1. Заглавная страница новостного портала LENTA.RU

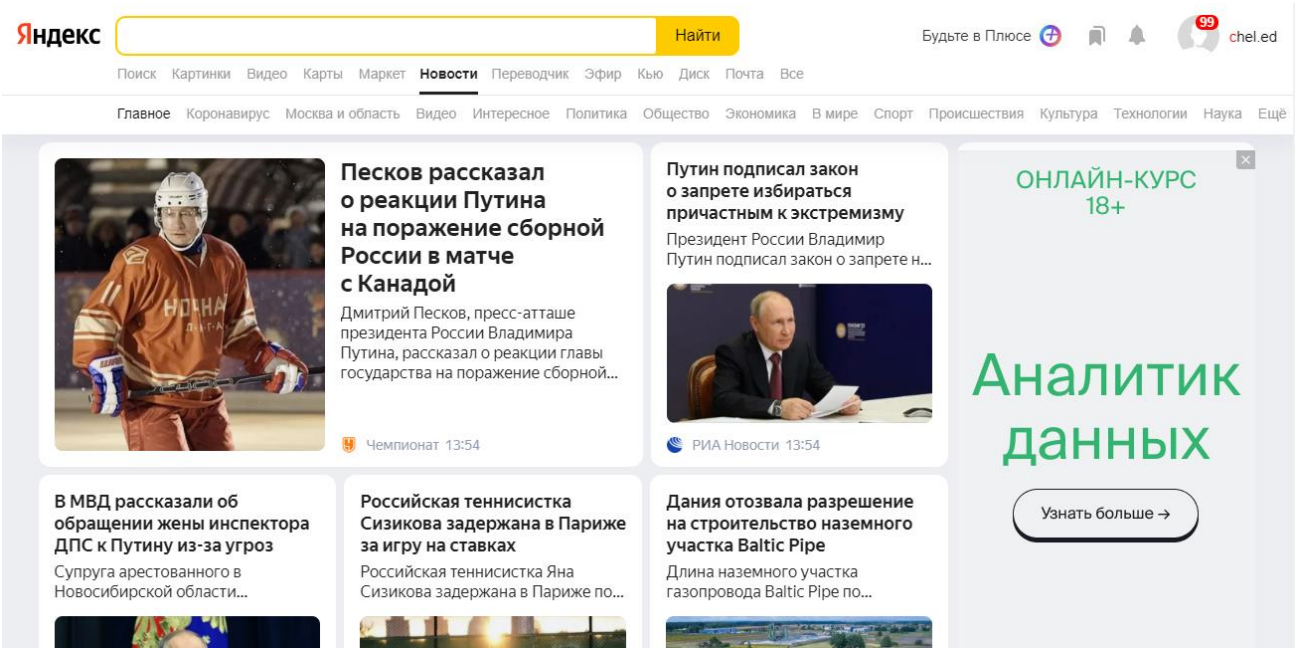

Рис. 2. Заглавная страница новостного агрегатора Яндекс.Новости

По результатам краткого обзора существующих новостных порталов можно заметить, что оба содержат возможность навигации по рубрикам. Если обратиться к другим новостным ресурсам, то в них также можно встретить аналогичный функционал.

Наличие возможности навигации по рубрикам делает использование новостного ресурса удобнее, упрощает поиск интересующей пользователя информации. При этом стоит учитывать, что ручная рубрикация малоэффективна: она занимает много времени, а люди, ее осуществляющие, не застрахованы от ошибок. Если в различных СМИ можно представить ручную рубрикацию во время написания новостной статьи, то в новостных агрегаторах рубрикация должна осуществляться «на ходу», то есть у человека может не быть достаточного количества времени для того, чтобы ознакомиться с содержимым новостной статьи. Большие задержки между появлением новости в СМИ и ее публикацией в новостном агрегаторе могут привести к оттоку пользователей из последнего.

1.3. Постановка задачи

В соответствии с техническим заданием на выпускную квалификационную работу (приложение А) основной задачей данной работы является создание информационной системы для автоматической рубрикации новостных текстов с использованием машинного обучения.

Для ее решения необходимо провести подготовку имеющегося корпуса и векторизацию новостных статей, входящих в него, а обработанный корпус использовать для построения рубрицирующих моделей классификации. Рубрикацию производить на девять непересекающихся рубрик: Дом, Интернет и СМИ, Культура, Наука и техника, Политика, Путешествия, Силовые структуры, Спорт, Экономика и бизнес. Точность построенных рубрицирующих моделей классификации необходимо оценить по ряду метрик. Для демонстрации работы системы автоматической рубрикации нужно создать веб-сайт, продумать его структуру и оформление. Предусмотреть возможность рубрикации поступающих в систему новостных статей в режиме реального времени.

Таким образом, обозначенная выше задача состоит из двух подзадач: создание системы автоматической рубрикации текстов с использованием машинного обучения и создания веб-сайта.

Создание системы автоматической рубрикации текстов с использованием машинного обучения состоит из следующих подзадач:

1) разработка программного модуля, содержащего функции подготовки и векторизации русскоязычных текстов. Данный модуль используется для подготовки и векторизации корпуса с целью построения моделей классификации;
2) проведение ряда экспериментов по построению моделей классификации. Необходимо обучить несколько различных моделей классификации, подобрать для каждой из них значения гиперпараметров, дающих наибольшее значение точности;
3) оценка качества построенных моделей классификации с использованием ряда метрик;
4) написание программы, осуществляющей рубрикацию в режиме реального времени. Данная программа должна выполняться в режиме реального времени, проводить подготовку текста новостной статьи и осуществлять его рубрикацию.

Разработка веб-сайта состоит из следующих подзадач:

1) определение структуры веб-сайта. Веб-сайт должен иметь понятную пользователю структуру, просмотр содержимого отдельных рубрик должен быть удобным;
2) работа с базой данных, которая должна содержать наполнение веб-сайта;
3) реализация веб-сайта, создание HTML-страниц;

4) внешнее оформление веб-сайта. Веб-сайт должен иметь привлекательный внешний вид и не вызывать у пользователя дискомфорта;
5) тестирование веб-сайта.

На рисунке 3 представлена наглядная схема взаимодействия компонентов разрабатываемой системы. База данных осуществляет хранение новостных статей, система автоматической рубрикации рубрицирует статьи из базы данных. Веб-сайт отображает результаты работы системы, взаимодействуя с пользователем.

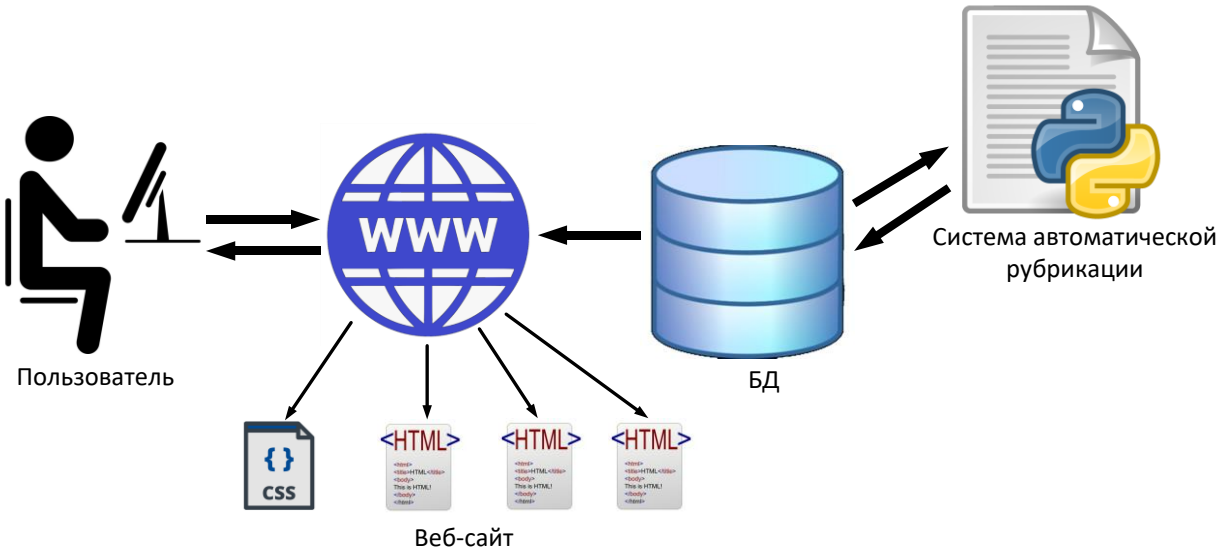

Рис. 3. Схема взаимосвязи компонентов разрабатываемой системы

Исходные данные

Для обучения моделей классификации, оценки их качества по ряду метрик и наполнения веб-сайта был использован набор данных новостного интернет-портала LENTA.RU [10]. Фрагмент данного набора данных представлен на рисунке 4.

	url	title	text	topic	tags	date
0	https://lenta.ru/news/2018/11/15/zvezda/	На «Звезде» состоялась церемония резки металла...	На судостроительном комплексе «Звезда» в город...	Экономика	Деловой климат	2018/11/15
1	https://lenta.ru/news/2018/11/15/macron_trump/	Франция отказалась раболепствовать перед США	Президент Франции Эммануэль Макрон ответил на ...	Политика	Политика	2018/11/15
2	https://lenta.ru/news/2018/11/15/bestsexever/	Названы лучшие сцены секса в кино	Издание The Independent составило список из де...	Культура	Кино	2018/11/15
3	https://lenta.ru/news/2018/11/15/metalloiskatel/	Кремль прояснил историю с металлоискателем и П...	Пресс-секретарь президента России Дмитрий Песк...	Политика	Политика	2018/11/15
4	https://lenta.ru/news/2018/11/15/zakon/	Статью об экстремизме в России смягчат до конц...	Закон о частичной декриминализации статьи об э...	Силовые структуры	Следствие и суд	2018/11/15
...	...	...	...	...	...	...
1407	https://lenta.ru/news/2018/11/30/told_you/	США помешали военным поставкам Турции	Поставка Турцией боевых вертолетов T129 Atak П...	Политика	Политика	2018/11/30
1408	https://lenta.ru/news/2018/11/30/vladimirtrump/	Американские журналисты случайно нарекли прези...	Журналисты американского издания The Wall Stre...	Интернет и СМИ	Пресса	2018/11/30
1409	https://lenta.ru/news/2018/11/30/kunzevo/	Жители Кунцево попросили Собянина переселить и...	Инициативная группа жителей 47 и 48 кварталов ...	Экономика	Бизнес	2018/11/30
1410	https://lenta.ru/news/2018/11/30/imenarf/	Настал последний день выбора названий российск...	Около пяти миллионов россиян приняли участие в...	Путешествия	Россия	2018/11/30

Рис. 4. Фрагмент используемого набора данных

Для каждой статьи приведены ее содержание и заголовок (столбцы **text** и **title** соответственно), дата публикации (столбец **date**) и URL-ссылка на нее (столбец **url**). Кроме того, указана рубрика, к которой относится данная публикация (столбец **topic**) и дополнительный поясняющий содержание статьи тэг (столбец **tags**).

Корпус новостных статей, содержащийся в рассматриваемом наборе данных, используется в работе как для построения рубрицирующих моделей классификации, так и для наполнения веб-сайта. В целях соблюдения авторского права публикация полных текстов новостных статей в данной работе производиться не может, допустима публикация лишь заголовков и начальной части текста новостной статьи. Необходимо предусмотреть возможность предоставления пользователю возможности осуществить переход на оригинальную новостную статью по ссылке.

2. ОПИСАНИЕ СРЕДСТВ РАЗРАБОТКИ

Выполнение работы состоит из двух подзадач: создание системы рубрицирования текстов с использованием машинного обучения и разработка веб-сайта. Каждая подзадача требует индивидуального подхода и отбора отдельных средств разработки.

2.1. Анализ языков программирования для машинного обучения

Для проведения исследований в области машинного обучения, а также построения и обучения моделей машинного обучения используются языки программирования со своими стандартными библиотеками машинного обучения. Такие библиотеки избавляют разработчика от необходимости каждый раз заново реализовывать тот или иной алгоритм машинного обучения. Сегодня библиотеки машинного обучения существуют у большинства языков программирования, однако в данной работе будут рассмотрены следующие: Python, C++, Java и R.

Python – объектно-ориентированный язык программирования высокого уровня. Синтаксис языка максимально прост и удобен, язык содержит большое количество стандартных библиотек, в том числе для машинного обучения: **NumPy**, **Pandas**, **Scikit-learn** и др., и машинной обработки естественного языка: **Gensim**, **PyMorphy2**, **NLTK** и др. [11]. На каждую стандартную библиотеку достаточно просто найти подробную документацию. Этот факт упрощает не только разработку проектов на языке Python, но и их дальнейшую поддержку. Кроме того, для данного языка существуют фреймворки для разработки веб-сайтов, что устраняет возможную проблему совместимости системы автоматической рубрикации текстов и веб-сайта. Эта совокупность фактов делает язык программирования Python одним из самых популярных в мире [12].

К недостаткам языка программирования Python стоит отнести менее экономное расходование памяти и менее быструю работу, чем у некоторых других языков программирования, например, C++ [13].

Язык программирования **C++** является хорошим выбором в тех проектах, где необходима высокая скорость работы и существует строгое ограничение по памяти, например, при создании программного обеспечения для высоконагруженных сервисов. В машинном обучении данный язык программирования используется в задачах, в которых требуется обрабатывать большие объемы данных и быстро принимать решения. Так, например, на языке C++ написан симулятор беспилотных летательных аппаратов и автомобилей AirSim.

Минусом данного языка программирования в отношении машинного обучения является малое количество открытых библиотек машинного обучения и более скудная документация по ним в сравнении с языком программирования Python. Кроме того, данный язык программирования, как правило, не используется в веб-разработке.

Объектно-ориентированный язык программирования высокого уровня **Java** также часто используется в машинном обучении и содержит ряд специализированных для машинного обучения библиотек: **MOA, Weka**, **MALLET**. Однако, инфраструктура в сфере машинного обучения языка программирования Java не такая богатая, а документация к ней не такая обширная и подробная, как, например, у языка программирования Python.

Основным предназначением языка программирования **R** является проведение статистических исследований. Он используется для статистической обработки данных, а также содержит ряд библиотек для машинного обучения: **CARET, benchm-ml**, **KernLab** и др. Однако данный язык программирования непрост в освоении для начинающих, так как содержит более 11 тысяч пакетов и

более сотни тысяч функций. Кроме того, данный язык программирования не предназначен для веб-разработки [14].

Таким образом, по итогам проведенного краткого обзора можно сделать вывод, что разработку в рамках данной работы целесообразно вести на языке программирования Python.

2.2. Анализ фреймворков для веб-разработки на языке Python

Для разработки веб-сайта необходимо выбрать фреймворк. Фреймворки облегчают разработку веб-сайта и предоставляют для этого удобный функционал

Для выбора фрейморвка веб-разработки необходимо провести анализ существующих фреймворков веб-разработки на языке программирования Python. Среди них самыми популярными являются фреймворки **Django** и **Flask**. Оба свободно распространяются и являются бесплатными [15]. Проведем сравнительный анализ фреймворков Django и Flask.

1) Фреймворк Flask используется, как правило, для создания небольших и простых веб-приложений. Разработка сложных веб-проектов с использованием фреймворка Flask обычно не ведется, в то время как фреймворк Django хорошо адаптирован для этой цели;
2) Фреймворк Django имеет встроенное средство ORM. **ORM** – подход в программировании, устанавливающий связь между сущностями объектно-ориентированных языков и реляционными базами данных. По сути, ORM устанавливает связь с таблицей базы данных, создавая ее объектно-ориентированный аналог. Flask не имеет собственного средства ORM, но позволяет подключить стороннее.

3) Фреймворк Django при создании проекта автоматически создает строго структурированную систему директорий, у каждой из которых имеется свое заранее определенное предназначение. Эта особенность значительно упрощает структуризацию веб-приложения.
4) При использовании фреймворка Django автоматически создается встроенная панель администратора базы данных, позволяющая определенному кругу пользователей, наделенных соответствующими правами, удобно редактировать записи в таблицах базы данных, а также управлять ею;
5) Документация к фреймворку Django [16] полнее и подробнее, чем документация к фреймворку Flask [17].

Архитектура фреймворка **Django** соответствует шаблону проектирования **MVC** (англ. model, view, controller), позволяющая разделить все файлы веб-сайта на три основные категории:

- файлы-модели (англ. model), которые предназначены для работы с базами данных;

- файлы-представления (англ. view), которые отвечают за отображение данных на HTML-страницах;

- файлы-контроллеры, предназначенные для синхронизации работы файлов-моделей в соответствии с действиями пользователя.

Преимуществом шаблона проектирования MVC является то, что он позволяет реализовывать внутри крупного Django-проекта отдельные функциональные блоки, каждый из которых может быть модифицирован независимо от других. Такая сравнительно невысокая степень интеграции отдельных функциональных блоков внутри Django-проекта облегчает его создание и поддержку.

Кроме того, в фреймворке Django есть встроенный шаблонизатор. **Шаблонизатор** – программное решение, позволяющее использовать HTML-шаблоны для генерации итоговых HTML-страниц, что существенно уменьшает дублированность кода. Шаблонизаторы также позволяют отделить друг от друга участки кода, ответственные за логику и внешний вид веб-страницы.

2.3. Анализ систем управления базами данных

Современный веб-сайт невозможно представить без хотя бы одной, а чаще нескольких таблиц внутри базы данных. Таблицы баз данных могут использоваться для хранения наполнения веб-сайта, данных о пользователях и других целей. Для работы с базами данных необходима система управления базами данных (СУБД) [18].

По умолчанию фреймворк Django настроен на работу с СУБД **SQLite**. SQLite является встраиваемой СУБД, то есть она представляет собой библиотеку. Взаимодействие веб-приложения и СУБД SQLite реализовано с использованием функций этой библиотеки, и, как следствие, такое взаимодействие оказывается быстрее и эффективнее. Существенным недостатком СУБД SQLite является отсутствие системы пользователей и прав доступа. Кроме того, у СУБД SQLite существуют ограничения на операцию записи, так как разрешен только один процесс записи в некоторый промежуток времени [19].

Все вышеназванные факты указывают на необходимость выбора другой, более совершенной и мощной СУБД в случаях разработки крупных и многопользовательских веб-приложений или при работе с большими объемами данных. Ярким примером такой СУБД является MySQL, которая уверенно входит в число самых популярных СУБД [20]. Данная СУБД поддерживает большую

часть функционала языка запросов SQL, предоставляет возможность удобной работы с большими объемами данных. Кроме того, **MySQL Workbench** предоставляет удобный пользовательский интерфейс для администрирования баз данных и работы с ними.

Подводя итог всему вышесказанному, можно констатировать, что использование СУБД MySQL в данной работе представляется наиболее обоснованным.

2.4. Анализ средств разработки

Интегрированная среда разработки (англ. Integrated Development Environment, сокр. IDE) – это специализированная программа, предназначенная для разработки программного обеспечения с использованием заранее оговоренных средств разработки (языки программирования, фреймворки, API и т.п.). Интегрированная среда разработки включает в себя редактор программного кода, инструменты сборки, отладки и прочий функционал.

Для комфортной работы к интегрированным средам разработки предъявляются следующие требования:

- возможность запуска разрабатываемой программы из интегрированной среды разработки, что значительно упрощает разработку и отладку программы;

- поддержка отладки, включающая в себя возможность пошагово выполнять программный код;

- возможность сохранения файлов, что позволяет продолжить работу над проектом после его закрытия. Для обеспечения надежности приветствуется также промежуточное сохранение без участия разработчика;

- подсветка синтаксиса и выделение синтаксических ошибок. Данная особенность позволяет разработчику удобно читать программный код и находить в нем нужные участки;

- автоматическое форматирование кода. Правильно отформатированный программный код легче читать, форматирование выделяет иерархию и вложенность участков программного кода друг в друга.

Безусловно, можно назвать еще много требований, а также индивидуальных пожеланий к функционалу интегрированных сред разработки, но указанные выше являются своеобразным минимумом, которому должна отвечать всякая интегрированная среда разработки.

Как уже обозначалось выше, разработка в рамках данной работы разделяется на две подзадачи: создание системы рубрикации русскоязычных текстов и создание веб-сайта. В связи с тем, что эти задачи разные, они требуют и различных средств разработки.

В качестве среды разработки для проведения экспериментов по машинному обучению была выбрана интегрированная среда разработки **Jupyter Notebook**, которая является открытой и свободно распространяемой. Данная интегрированная среда разработки генерирует файлы с расширением .ipynb, называемые ноутбуками (записными книжками Jupyter). Такой ноутбук построен по принципу последовательности ячеек. Программный код в каждой из ячеек выполняется отдельно, что весьма удобно при проведении экспериментов по машинному обучению. Кроме того, данная интегрированная среда разработки предоставляет возможность создавать заголовки и отображать графику прямо в ноутбуке, что значительно повышает его читаемость. Пользовательский интерфейс интегрированной среды разработки Jupyter Notebook представлен на рисунке 5. Так как работа с большими объемами данных и обучение моделей машинного обучения являются процессами, требующими сравнительно больших

вычислительных ресурсов, целесообразно для проведения экспериментов по машинному обучению использовать облачную реализацию рассматриваемой интегрированной среды разработки **Google Colaboratory**.

Данная интегрированная среда разработки доступна вместе с дистрибутивом языка Python **Anaconda**, основной направленностью которого являются анализ данных и машинное обучение [21].

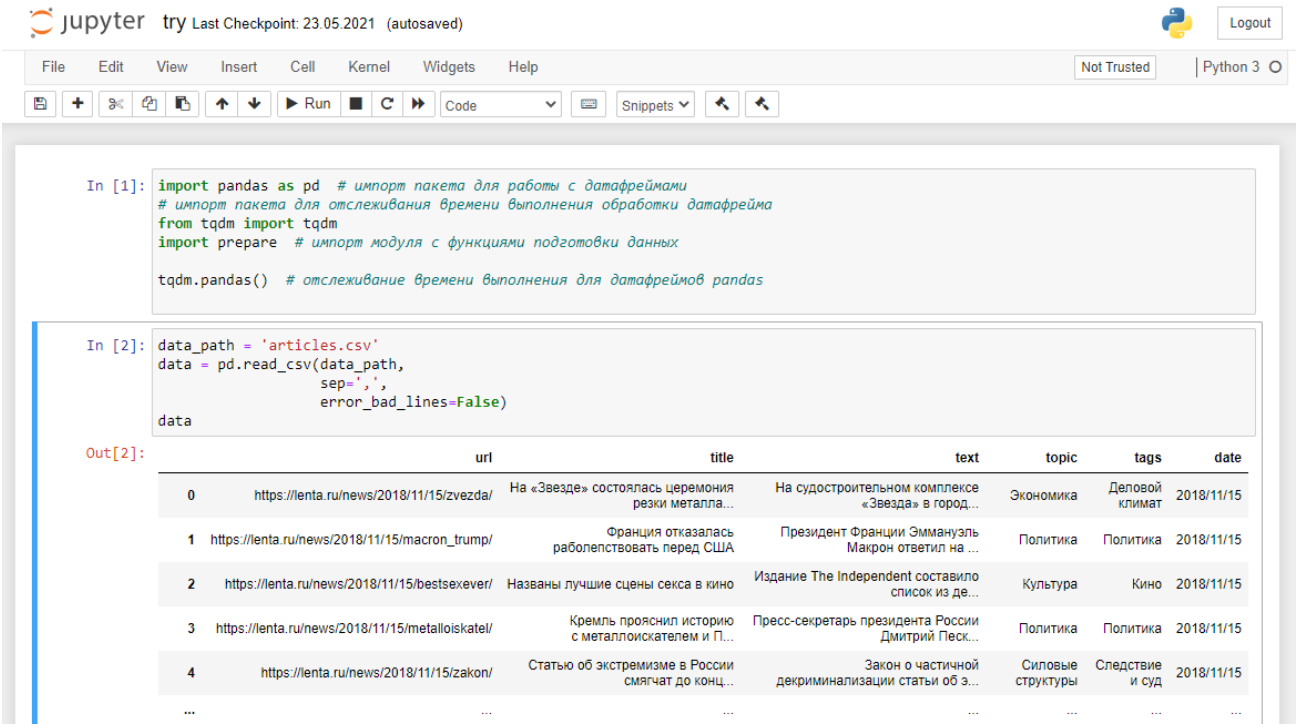

Рис. 5. Пользовательский интерфейс Jupyter Notebook

В составе дистрибутива Anaconda доступна также интегрированная среда разработки **Spyder**, пользовательский интерфейс которой представлен на рисунке 6. Данная интегрированная среда разработки была использована для разработки веб-сайта.

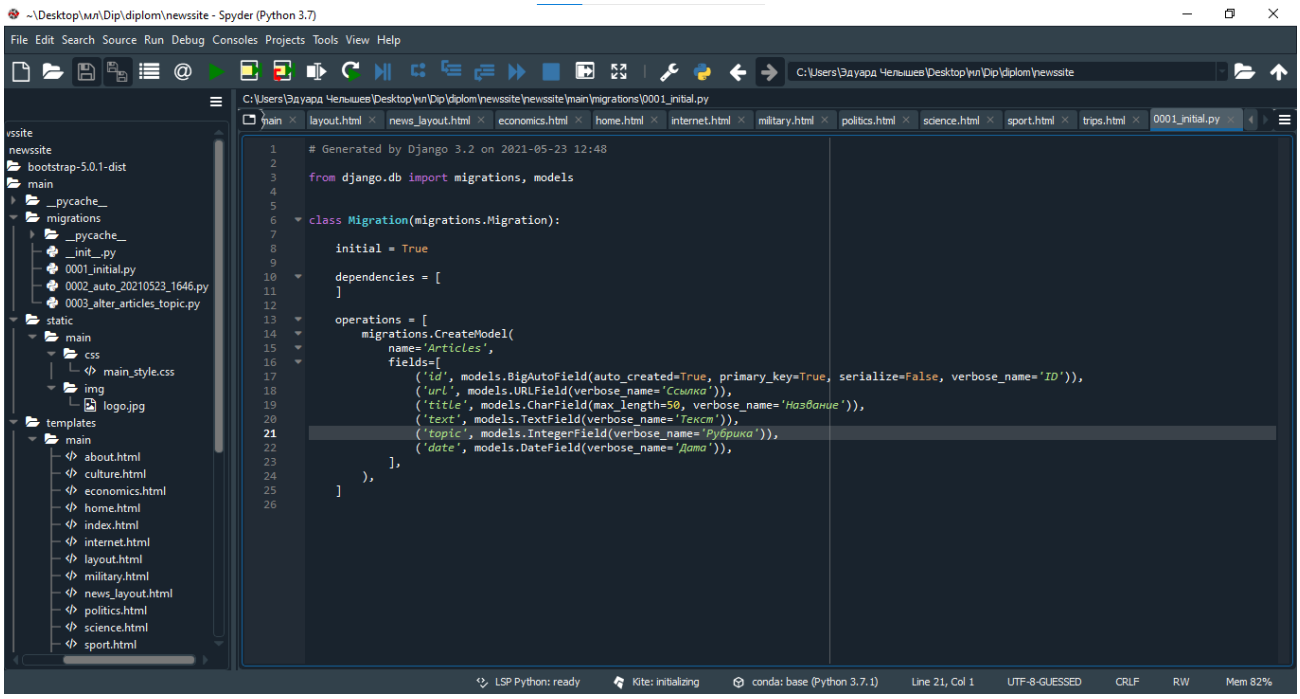

Рис. 6. Пользовательский интерфейс Spyder

3. РАЗРАБОТКА СИСТЕМЫ РУБРИКАЦИИ ТЕКСТОВ

С разработанными в рамках данного раздела программными файлами можно ознакомиться в приложении Б.

3.1. Обзор задачи машинной обработки естественного языка

Для рубрикации новостных текстов в данной работе используется машинное обучение. Методы машинного обучения разделяются на три большие группы: методы машинного обучения по прецедентам (с учителем), методы машинного обучения без учителя и методы машинного обучения с подкреплением.

Машинное обучение по прецедентам (машинное обучение с учителем, контролируемое машинное обучение, англ. supervised learning) предполагает наличие размеченного набора данных, в котором каждому элементу набора данных (объекту) поставлено в соответствие некоторое значение из множества допустимых значений – ответов. Иными словами, задана целевая функция (англ. target function) (1).

$$f: X \to Y \tag{1}$$

X – множество объектов (набор данных);

Y – множество допустимых значений – ответов.

Пары $(x \in X, y \in Y)$ называются **прецедентами**. Совокупность прецедентов называется обучающей выборкой (англ. training sample).

Целью машинного обучения по прецедентам является восстановление целевой функции f по обучающей выборке с как можно большей точностью, то есть построение решающей функции (англ. decision function), приближающей целевую [6].

Машинное обучение без учителя (неконтролируемое машинное обучение, англ. unsupervised learning) отличается от машинного обучения по прецедентам тем, что в обучающей выборке отсутствуют ответы. Целью машинного обучение без учителя является обнаружение внутренних взаимосвязей обучающей выборки, выявление ее скрытой структуры [22].

Машинное обучение с подкреплением (англ. reinforcement learning) подразумевает наличие обучающейся системы и среды. В случае, если система справляется с поставленной перед ней задачей, то среда направляет в систему сигнал вознаграждения. Система постепенно находит именно ту последовательность действий, которая приносит ей наибольшее вознаграждение [22].

Задачи машинного обучения по прецедентам в зависимости от вида множества допустимых ответов Y подразделяется на задачи классификации (когда множество допустимых значений состоит из конечного количества значений – меток) и задачи восстановления регрессии (когда множество допустимых значений является непрерывным множеством, то есть $Y \subset \mathbb{R}$) [6]. Как уже отмечалось выше, рубрикация является задачей многоклассовой классификации.

Тексты на естественных языках отличаются неоднозначностью, которая может значительно усложнять их машинную обработку. Например, одно и то же слово в различных контекстах может обозначать различные понятия. Нередко в естественных текстах встречаются омонимы, то есть слова с одинаковым звучанием и/или написанием, но абсолютно разными значениями [3]. Если рассматривать неформальные тексты, например, бытовую переписку или

публикации в социальных сетях, то часто можно встретить различного рода сокращения слов, жаргонизмы и даже орфографические ошибки.

Безусловно, описанные выше существенные неоднозначности естественного языка делают его машинную обработку непростой задачей, требующей специфичных подходов. В данной работе проводится машинная обработка новостных русскоязычных текстов. Выбор новостных текстов объясняется в том числе тем немаловажным фактом, что их рубрикацию легче осуществить, так как новостные тексты четко структурированы, относятся к конкретной тематике, включают большое количество специфических для данной тематики терминов. Язык, используемый при написании новостных текстов, почти лишен метафор и орфографических ошибок.

3.2. Разработка программного модуля

Одним из важнейших этапов решения задач машинной обработки естественного языка является подготовка данных. Подготовка данных в рассматриваемой задаче включает в себя как общие для всех задач машинного обучения этапы (анализ набора данных, удаление выбросов и пропусков) [1], которые были реализованы мною отдельно, так и специфичные для машинной обработки естественного языка этапы: **удаление нерелевантных символов и приведение к общему регистру, токенизация, удаление стоп-слов, приведение к начальной форме и векторизация** [3, 23], для осуществления которых был разработан программный модуль на языке Python **prepare.py**.

Модуль содержит ряд функций, осуществляющих вышеназванные этапы подготовки, а также функцию **prepare_text**, включающую в себя все этапы подготовки и осуществляющую их путем вызова прочих функций данного

модуля. Таблица 1 содержит спецификации всех входящих в данный программный модуль функций.

Таблица 1.

Спецификации функций программного модуля prepare.py

Имя функции	**Назначение**	**Входные данные**	**Выходные данные**
load_model	Данная функция загружает предобученную модель векторизации FastText, расположенную по пути **path**	строка пути к модели векторизации: **path**	нет
preprocess_text	Данная функция производит приведение текста к одному регистру и удаляет из него нерелевантные символы	строка, содержащая текст, подлежащий обработке: **text**	строка, содержащая обработанный текст: **preprocessed_text**
preprocess_text	Данная функция производит приведение текста к одному регистру и удаляет из него нерелевантные символы	строка, содержащая текст, подлежащий обработке: **text**	строка, содержащая обработанный текст: **preprocessed_text**
lemmatize	Данная функция осуществляет лемматизацию токенов	список токенов: **text_list**	список лемматизированных токенов: **result**
delete_stop_words	Данная функция удаляет из списка токенов стоп-слова	список токенов: **text_list**	список токенов с удаленными стоп-словами: **result**
build_article_vector	Данная функция формирует вектор по списку токенов	список токенов: **t**	массив NumPy: **res**
prepare_text	Данная функция осуществляет все этапы подготовки данных.	строка текста: **text**	массив NumPy: **vector**

Первый этап подготовки, а именно **удаление нерелевантных символов и приведение символов к общему регистру**, позволяет исключить из текста несущественную или даже вредную для обучения модели машинного обучения

информацию. Его производит функция **preprocess_text**. Данная функция осуществляет замену всех символов входной строки на строчные, а также заменяет символ «ё» на «е». Далее с использованием регулярных выражений удаляются присутствующие в тексте URL-ссылки, а также все небуквенные символы, исключая пробелы. Для работы с регулярными выражениями необходимо подключить стандартную библиотеку языка Python **re**.

Токенизация – процесс, в ходе которого осуществляется разбиения текста на токены, то есть текстовые единицы (символ, слово, предложение и т.д.). Как правило, токены отделены друг от друга разделительными символами. В данной работе токенизация осуществляется с использованием встроенного метода **word_tokenize** стандартной библиотеки **NLTK** [24].

Как известно, морфологическая структура русского языка очень богата. Слова могут иметь различные падежные и временные формы. Это придает текстам на естественном языке связность, однако, мешает машинной обработке русскоязычных текстов, так как одному и тому же значению соответствуют сразу несколько форм одного слова, которые в дальнейшем при машинной обработке будут восприниматься как различные токены [25]. Этот факт приводит к необходимости **приведения слов к их начальной**, то есть словарной, **форме**. Для решения данной задачи используются два метода: стемминг и лемматизация.

Стемминг (англ. stemming) – метод приведения слов к начальной форме, в ходе которого от слов отсекаются окончания. Данный метод в сравнении с лемматизацией, рассмотренной ниже, менее требователен к вычислительным ресурсам. Однако стемминг является достаточно грубым методом и не гарантирует, что слова будут приведены к начальной форме: отсечение окончаний может оказаться не совсем корректным, могут быть удалены или, наоборот, оставлены лишние буквы. Проблема так же возникает со словами, у которых в

некоторых формах имеется изменение не только окончания, но и основной части, например, при чередовании букв.

Лемматизация – метод приведения слов к начальной форме, при котором проводится морфологический анализ слова, по результатам которого оно приводится к своей начальной форме. В отличие от стемминга, лемматизация гарантирует, что различные формы будут в итоге приведены к одной, начальной форме. Однако, несмотря на это достоинство, у лемматизации есть недостаток: она более требовательна к вычислительным ресурсам и занимает больше времени.

В рамках данной работы для решения задачи приведения к начальной форме была выбрана лемматизация, для чего была реализована функция **lemmatize**, которая, получая на вход список токенов, приводит их к начальной форме с использованием русскоязычного морфологического анализатора, реализованного в библиотеке **pymorphy2**. Подробнее ознакомиться с данной библиотекой и ее возможностями можно в [26] и [27].

Стоп-слова – это часто встречающиеся в текстах слова, которые играют большую роль в обеспечении связности предложения, однако при машинной обработке естественного языка являются шумом и мешают обучению модели, так как не несут практически никакой информации. К ним можно отнести частицы, предлоги, союзы, местоимения и прочие слова-связки. Пакет **NLTK** имеет в своем составе список стоп-слов для русского языка, насчитывающий 151 слово.

Для решения задачи удаления стоп-слов в программном модуле prepare.py реализована функция **delete_stop_words**. Получая на вход список токенов, она возвращает список, очищенный от стоп-слов.

Векторизация – процесс, в результате которого каждому токену ставится в соответствие его векторное представление, то есть вектор v некоторого n-мерного векторного пространства $\mathbb{R}^n$. Для получения векторного представления слов

используются различные методы векторизации, которые могут как сохранять семантическую, то есть смысловую, близость слов, так и игнорировать ее. Например, модель мешка слов никак не учитывает семантическую близость [23]. Модели, сохраняющие семантическую близость, подразделяются на статические модели векторизации (например, Word2Vec, FastText) и динамические (контекстуализированные) модели [28]. Последние называются так же языковыми моделями (ELMO, BERT и др.).

Мешок слов – один из наиболее простых методов для получения векторного представления слов. Данный метод на этапе обучения выделяет N наиболее часто встречаемых токенов, из них формируется словарь мощности N. Каждому токену словаря ставится в соответствие некоторый уникальный индекс i от 1 до N. При работе для документа подсчитывается количество вхождений каждого токена из словаря, полученное количество вхождений присваивается соответствующей этому слову i-той компоненте векторного представления документа. Таким образом, каждая компонента векторного представления документа содержит в себе число, равное количеству вхождений соответствующего ей токена в данный документ. Достоинством данного метода можно назвать простоту реализации. Недостатком его является очень большой размер получаемых векторов и, как следствие, замедление операции их сравнения. Однако наиболее существенным недостатком является тот факт, что метод никак не учитывает порядок слов в тексте и семантику документа, а лишь подсчитывает количество токенов, что существенно уменьшает информативность полученных векторных представлений в сравнении с текстом на естественном языке и понижает точность машинной обработки [23].

Методы, сохраняющие семантическую близость, как правило, строятся на основе искусственных нейронных сетей. Основываются такие методы на идее дистрибутивной семантики: слова, встречающиеся в аналогичных контекстах и имеющие близкую частоту употребления, как правило, семантически (т.е. по

смыслу) близки. Важно отметить, что, в отличие от метода мешка слов, такие методы ставят токенам в соответствие вектора размерности, значительно меньшей, чем размер словаря [29]. Такой подход в английском языке получил название **embedding**.

Word2Vec – статическая модель векторизации, сохраняющая семантическую близость [28]. Данный подход был предложен Томашом Миколовым в 2013 году. На этапе обучения Word2Vec получает на вход текстовый корпус большого размера, формирует словарь полученного корпуса и строит векторные представления для его слов. Данное векторное представление учитывает контекстную близость: слова, которые встречаются рядом с одинаковыми словами и имеют похожий смысл, будут иметь близкие векторные представления.

Достоинством данного метода является то, что он учитывает контекстную близость слов, что значительно увеличивает информативность векторных представлений слов в сравнении с методом мешка слов. Однако данный метод более требователен к объему обучающей выборки и дольше обучается, чем метод мешка слов.

Модель векторизации **FastText**, разработанная компанией **Facebook**, является статической моделью векторизации, сохраняющей семантическую близость. Показывая результаты, схожие с результатами аналогичных моделей векторизации, FastText имеет более высокую скорость обучения. Другим достоинством данного метода является то, что он аналогично Word2Vec учитывает контекстную близость слов, что значительно увеличивает информативность векторных представлений слов. На рисунке 7 видно, что модель векторизации способна определять семантически (т.е. по смыслу) близкие слова. Именно по этим причинам для выполнения работы мною была выбрана эта модель векторизации.

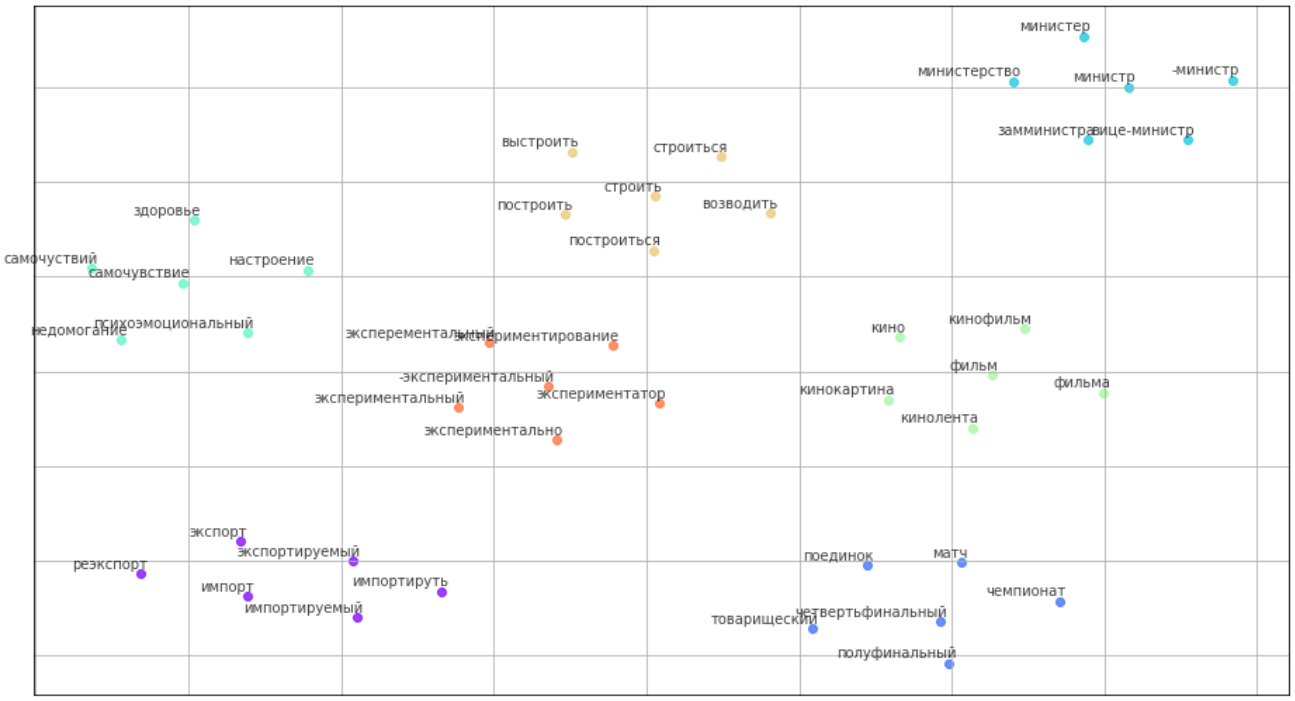

Рис. 7. Пример определения моделью векторизации семантически близких слов

Недостатком метода является то, что для обучения он требует большого объема данных. Кроме того, процесс обучения все же занимает немало времени, несмотря на то, что производится быстрее других статических моделей векторизации. Однако сейчас нетрудно найти ресурсы, содержащие уже предобученные модели векторизации. Одним из таких ресурсов является [30]. Оттуда была выбрана модель векторизации **geowac_lemmas_none_fasttextskipgram_300_5_2020**. Для работы с данной моделью векторизации необходимо подключить пакет **gensim** [31]. Загрузка модели векторизации осуществляется функцией **load_model** модуля prepare.py.

Языковые модели (англ. language models) являются еще более мощным инструментом машинной обработки естественных языков. Они не только сохраняют семантическую близость, но и способны определить, с какой вероятностью в тексте может встретиться некоторая последовательность слов, то есть определить вероятность встретить такую последовательность

$P(w_1, w_2, \dots, w_N,)$, где w_i – некоторое слово. Также языковые модели используются при языковом моделировании (англ. language modeling). Языковое моделирование – задача определения слова, которое с наибольшей вероятностью продолжает имеющуюся последовательность слов. Примером практического применения языкового моделирования является автоматическая генерация текстов на естественном языке. Подробнее о языковых моделях изложено в [28].

Функция **build_article_vector** данного модуля, получая на вход список токенов, соответствующий некоторому тексту, вычисляет его вектор как среднее арифметическое векторов отдельных токенов, входящих в данный список. Для корректной работы данной функции необходимо подключить стандартную библиотеку **NumPy**, позволяющую осуществлять эффективную работу с массивами [33].

Схема алгоритма функции **prepare_text** представлена на рисунке 8. Данная функция последовательно осуществляет удаление нерелевантных символов, токенизацию, приведение токенов к начальной форме, удаление стоп-слов и векторизацию.

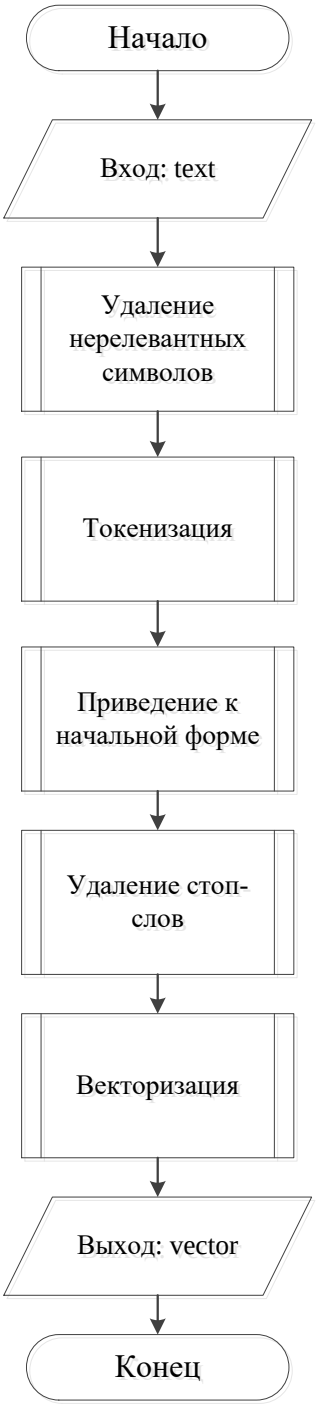

Рис. 8. Схема алгоритма функции prepare_text

3.3. Подготовка корпуса

Подготовка данных для построения моделей классификации была осуществлена в файле **corpus.py**.

Использованный в данной работе набор данных представлен в виде файла в формате CSV. В дальнейшей работе, в соответствии с принятой в машинной обработке естественного языка терминологией, набор данных будет именоваться **корпусом**.

Для чтения данного корпуса был использован стандартный метод библиотеки **Pandas read_csv**. Подробнее с библиотекой Pandas можно ознакомиться в [34]. Возвращаемым данным методом результатом является объект класса **pandas.DataFrame**, который в среде Jupyter Notebook отображается в виде таблицы, как показано на рисунке 9.

	text	topic	tags
0	Бои у Сопоцкина и Друскеник закончились отступ...	Библиотека	Первая мировая
1	Министерство народного просвещения, в виду про...	Библиотека	Первая мировая
2	Штабс-капитан П. Н. Нестеров на днях, увидев в...	Библиотека	Первая мировая
3	Фотограф-корреспондент Daily Mirror рассказыва...	Библиотека	Первая мировая
4	Лица, приехавшие в Варшаву из Люблина, передаю...	Библиотека	Первая мировая
...	...	...	...
800970	Певец Сергей Шнуров раскритиковал свою коллегу...	NaN	ТВ и радио
800971	Министерство юстиции России предложило изменит...	NaN	Все
800972	Испытание США ранее запрещенной Договором о ли...	NaN	Политика
800973	В ближайшие дни в европейской части России пог...	NaN	Общество
800974	Ведущие футбольные чемпионаты ушли на зимние к...	NaN	Английский футбол

800975 rows × 3 columns

Рис. 9. Пример отображения объекта класса pandas.DataFrame

При анализе корпуса выяснилось, что он включает в себя 23 рубрики. Из их числа были выделены девять рубрик, которые вошли в итоговый корпус. Из корпуса также были удалены пропуски. Названия рубрик и соответствующее каждой из них количество статей в корпусе представлены в таблице 2. Далее словесные названия рубрик были заменены их числовыми эквивалентами, которые также указаны в таблице 2.

Таблица 2.

Содержимое корпуса

Рубрика	Числовой эквивалент	Количество статей
Дом	0	21734
Интернет и СМИ	1	44663
Культура	2	53796
Наука и техника	3	53136
Политика	4	40716
Путешествия	5	6408
Силовые структуры	6	19596
Спорт	7	64413
Экономика и бизнес	8	86926

Подготовка корпуса была осуществлена с использованием программного модуля prepare.py.

Корпус после приведения к общему регистру и удаления нерелевантных символов представлен на рисунке 10.

	text	topic
0	с января года все телеканалы будут оплачивать ...	8
1	германский автопромышленный концерн volkswagen...	8
2	нераспределенная прибыль оао тюменнефтегаз доч...	8
3	две крупнейших телекоммуникационных компании с...	8
4	оао газ и нижегородский банк сбербанка россии ...	8
...	...	...
372817	законопроект о борьбе с домашним насилием одно...	4
372818	основатель американской частной военной компан...	4
372819	пресс секретарь президента россии дмитрий песк...	4
372820	белый дом ограничил число лиц которые имеют до...	4
372821	испытание сша ранее запрещенной договором о ли...	4

Рис. 10. Изображение очищенного от нерелевантных символов корпуса

Корпус после проведенной токенизации, лемматизации и удаления стоп-слов изображен на рисунке 11.

	text	topic
0	[январь, год, всё, телеканал, оплачивать, услу...	8
1	[германский, автопромышленный, концерн, volksw...	8
2	[нераспределённый, прибыль, оао, тюменнефтегаз...	8
3	[крупный, телекоммуникационный, компания, сша,...	8
4	[оао, газ, нижегородский, банк, сбербанк, росс...	8
...	...	...
372817	[законопроект, борьба, домашний, насилие, одно...	4
372818	[основатель, американский, частный, военный, к...	4
372819	[пресс, секретарь, президент, россия, дмитрий,...	4
372820	[белый, дом, ограничить, число, лицо, который,...	4
372821	[испытание, сша, ранее, запретить, договор, ли...	4

Рис. 11. Изображение корпуса после лемматизации

Обработка корпуса в виду его большого размера производилась частями, для чего в библиотеке Pandas есть удобный функционал. Для осуществления над корпусом действий, предусмотренных функциями модуля prepare.py, был использован метод класса Pandas.DataFrame **progress_apply,** для работы которого необходимо подключение библиотеки **tqdm.** Данный метод принимает функцию, которая должна быть применена к записям объекта класса Pandas.DataFrame, в качестве аргумента. Преобразуя корпус в соответствии с этой функцией, метод также в наглядной форме отображает текущий прогресс операции преобразования и выводит оценку оставшегося времени выполнения.

Подготовленный для построения моделей классификации корпус был сохранен в виде файла **dataset.csv**.

3.4. Проведение экспериментов по построению моделей классификации

Для построения моделей классификации и определения гиперпараметров моделей, дающих наибольшие значения точности, были использованы стандартные библиотеки языка программирования Python **Scikit-learn** и **Keras**. С документацией на нее можно ознакомиться в [35].

Подготовленные ранее данные были разделены на обучающую и тестовую выборки с использованием стандартного метода библиотеки Scikit-learn **train_test_split**. Размер тестовой выборки составил 25% от общего числа статей в корпусе.

Для построения моделей классификации были испробованы четыре метода классификации: гауссовский наивный байесовский классификатор, логистическая регрессия, случайный лес решающих деревьев и искусственная нейронная сеть.

Для определения значений гиперпараметров был использован алгоритм решетчатого поиска, заключающийся в последовательном обучении некоторой модели машинного обучения на одних и тех же данных, но при различных значениях гиперпараметров [36].

3.4.1. Наивный байесовский классификатор

Наивный байесовский классификатор (НБК) – вероятностный классификатор, в основе которого лежит теорема Байеса. Являясь довольно простой вероятностной моделью машинного обучения, он использует «наивное» предположение о независимости признаков используемого набора данных. Схема алгоритма наивного байесовского классификатора для случая n-мерных входных данных и k непересекающихся классов представлена на рисунке 12.

Получая на вход вектор признаков объекта, подлежащего классификации, НБК формирует ряд гипотез H_i, $i = 1 \dots k$ о принадлежности данного объекта к отдельным классам, после чего для каждого класса в соответствии с формулой (2) вычисляются значение вероятности $P(X|H_i)$, $i = 1 \dots k$, которую можно интерпретировать как вероятность встретить объект X в i-том классе. Именно в формуле (2) используется предположение о независимости признаков.

$$P(X|H_i) = \prod_{j=1}^{n} P\left(x_j \middle| H_i\right) \quad (2)$$

$P\left(x_j \middle| H_i\right), j = 1 \dots n$ – вероятность конкретного значения отдельного признака в i-том классе. Данная вероятность определяется на этапе обучения;

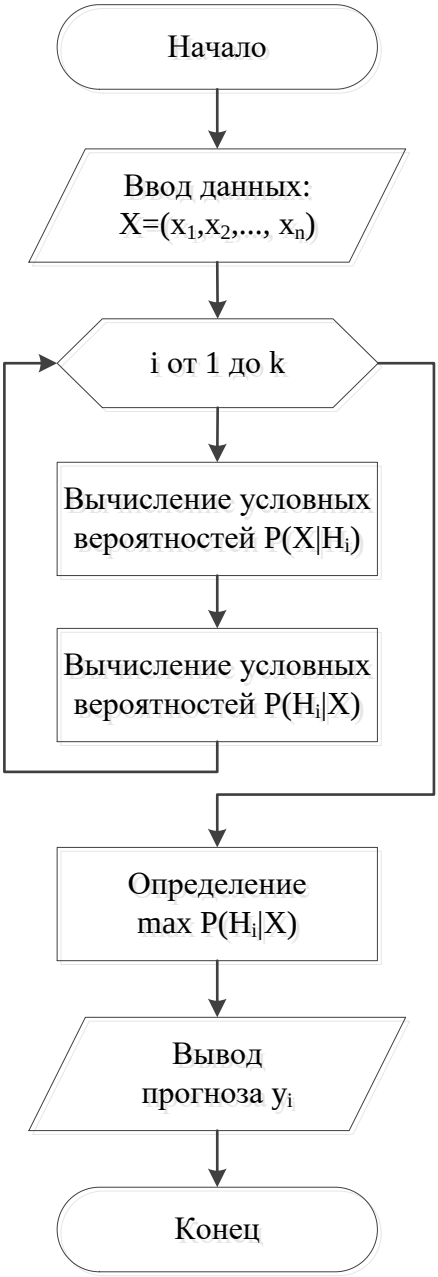

Рис. 12. Схема алгоритма наивного байесовского классификатора

Затем в соответствии с формулой (3) вычисляются вероятности $P(H_i|X), i = 1 \dots k$, имеющие смысл вероятности принадлежности объекта X i-тому классу.

$$P(H_i|X) = \frac{P(X|H_i)P(H_i)}{P(X)} = \frac{P(X|H_i)P(H_i)}{\prod_{j=1}^{n} P(x_j)} \quad (3)$$

$P(X)$ – вероятность объекта $X = (x_1, x_2, \dots, x_n)$;

$P(x_j),\ j = 1 \dots n$ – вероятность конкретного значения отдельного признака.

Затем определяется наибольшая из всех вероятностей $P(H_i|X), i = 1 \dots k$, на основе чего выдается прогноз о принадлежности объекта к некотором классу.

Одной из реализаций НБК является **гауссовский НБК**. Гауссовский НБК предполагает, что входные данные являются непрерывными и распределены в соответствии с нормальным вероятностным распределением [37]. В работе был использован гауссовский НБК, реализованный в библиотеке Scikit-learn.

3.4.2. Случайный лес решающих деревьев

Решающее дерево представляет собой древовидную структуру условий (тестов), которая разделяет набор данных на отдельные классы. Однако применять одно решающее дерево может быть нецелесообразным, так как у данного метода есть недостаток, а именно склонность к переобучению [22].

Переобучение – явление, при котором модель машинного обучения слишком точно повторяет зависимости обучающей выборки, что приводит к потере моделью обобщающей способности. Данное явление нежелательно, для борьбы с ним среди прочих используются ансамблевые методы, которые сочетают в себе множество моделей машинного обучения [36]. Примером ансамблевого метода является **случайный лес решающих деревьев**. Схема алгоритма случайного леса для *N* решающих деревьев приведена на рис. 13.

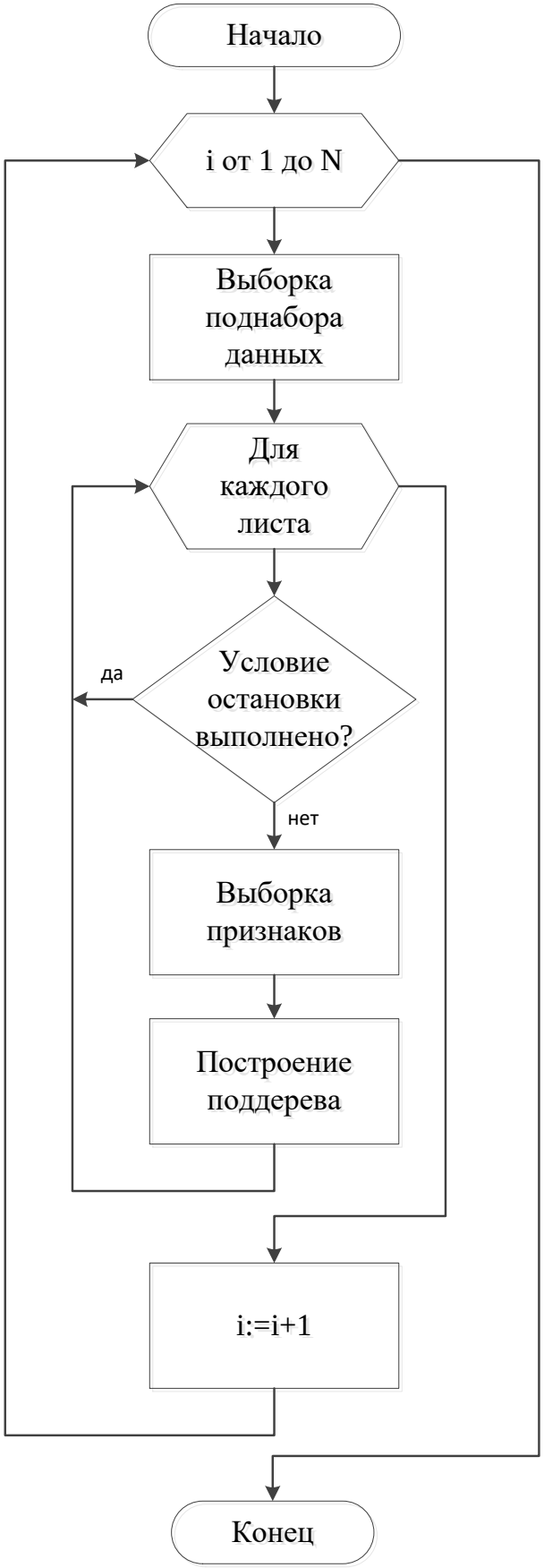

Рис. 13. Схема алгоритма случайного леса решающих деревьев

Данный метод заключается в построении множества решающих деревьев, которые в некоторой степени отличаются друг от друга. При построении каждого из деревьев в отдельности модель случайным образом выбирает объекты обучающей выборки, на которых будет обучаться. Для каждого теста модель случайным образом также выбирает значимые признаки. Таким образом, возможное переобучение каждого из деревьев в отдельности на своем наборе данных компенсируется за счет усреднения по множеству различных деревьев. Оптимизируемыми гиперпараметрами для данной модели машинного обучения в данной работе выступают количество решающих деревьев, и максимальное число признаков, учитываемых моделью на каждом тесте.

В работе был использован стандартный класс **RandomForestClassifier** библиотеки Scikit-learn. С использованием алгоритма решетчатого поиска были вычислены значения числа деревьев *n_estimators = 150* и максимального числа признаков *max_features = 30.*

3.4.3. Логистическая регрессия

Несмотря на то, что данный метод имеет в своем названии слово «регрессия», он является методом бинарной классификации. Данный метод вычисляет линейную границу двух классов. В общем случае, это гиперплоскость. Ее уравнение имеет вид (4).

$$f(x_1, x_2, \ldots, x_n) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_n x_n = 0 \quad (4)$$

где β_i, $i = 0 \ldots n$ – вычисляемые коэффициенты;

x_i, $i = 1 \ldots n$ – признаки объектов.

При классификации для объекта вычисляется значение $z = f(x_1, x_2, \ldots, x_n)$, если результат z оказался отрицательным, то объект относится к классу с меткой «0», если положительным, то – к классу с меткой «1». Для нормировки полученному результату применяется логистическая функция (5).

$$P(z) = \frac{1}{1+e^{-z}} \quad (5)$$

Значение логистической функции можно интерпретировать как вероятность того, что объект относится к классу с меткой «1». Объекту присваивается метка класса с наибольшей вероятностью принадлежности. Использование моделей бинарной классификации (например, логистической регрессии) для многоклассовой классификации заключается в построении набора бинарных классификаторов, каждый из которых определяет вероятность принадлежности объекта к отдельному классу, после чего находится класс с наибольшей вероятностью принадлежности. Такой подход в машинном обучении получил название «один против многих» [36].

С целью недопущения переобучения в логистической регрессии используется регуляризация, которая «штрафует» модель за переобучение [38]. Параметр регуляризации C является для данной модели гиперпараметром. На рисунке 14 представлен график зависимости точности модели от параметра регуляризации. График был построен с использованием программных средств языка Python [39]. Хорошо видно, что начиная с некоторого значения *C* точность модели становится константной. В качестве итогового значения параметра регуляризации было выбрано значение *C = 50*.

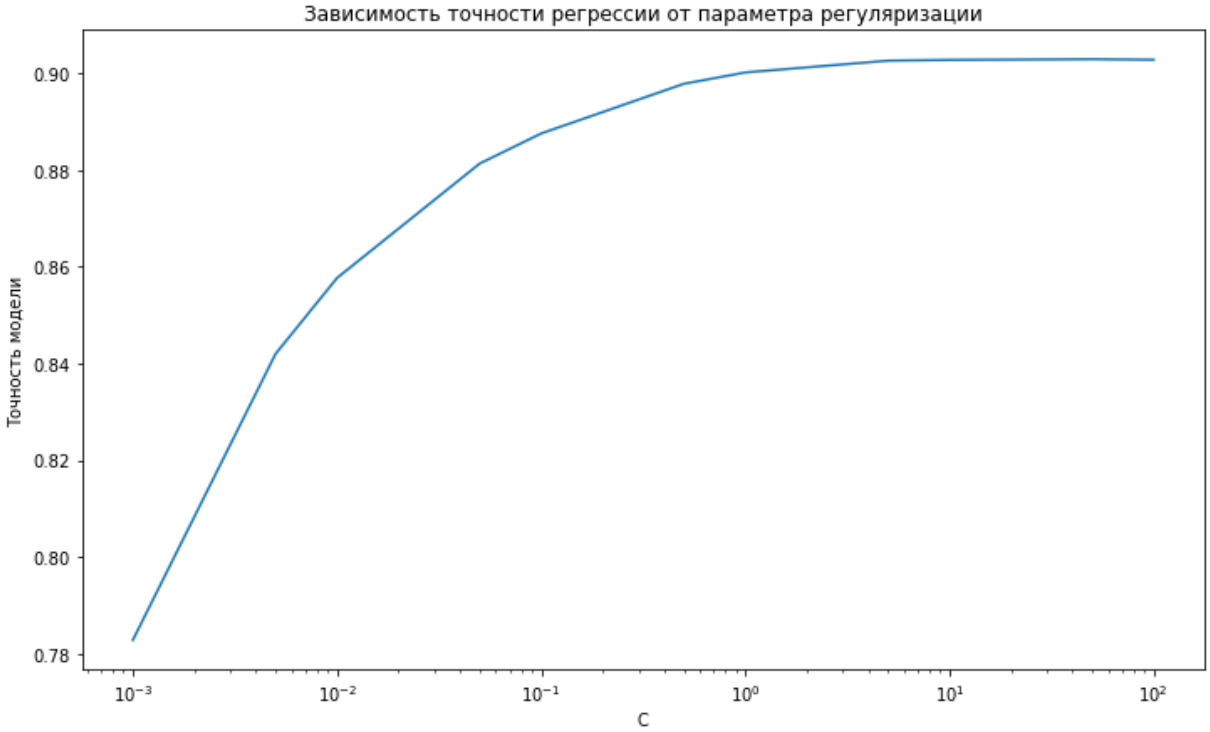

Рис. 14. Зависимость точности модели логистической регрессии от значения параметра регуляризации

3.4.4. Искусственная нейронная сеть

Искусственные нейронные сети (ИНС) – мощный инструмент машинного обучения, строящийся по подобию с естественными нейронными сетями живых существ. Искусственные нейронные сети успешно применяются для распознавания образов, машинной обработки естественного языка и др [40].

Библиотека **Keras** предоставляет разработчику удобный и гибкий функционал для создания ИНС и их обучения [41]. Кроме того, существует ряд других библиотек глубокого обучения для языка программирования Python, которые в данной работе рассматриваться не будут.

В рамках данной работы для решения поставленной задачи была разработана полносвязная четырехслойная ИНС, структура которой представлена на рисунке 15. Входной полносвязный слой содержит 300 нейронов. Первый

внутренний слой реализован с использованием строенного класса **Dropout** библиотеки Keras. Принцип работы Dropout-слоя (слоя исключения) прост: при работе с каждым из объектов обучающей выборки случайным образом отключаются отдельные нейроны этого слоя. Доля отключаемых нейронов определяется коэффициентом исключения, который указывается в конструкторе класса Dropout. Как правило, данное значение лежит в диапазоне от 0,2 до 0,5. Отключение отдельных нейронов позволяет снизить вероятность переобучения. Второй скрытый слой разработанной ИНС реализован при помощи встроенного класса **BatchNormalization** библиотеки Keras и предназначен для статистической нормализации значений, получаемых на выходах предыдущих слоев. Выходной слой является полносвязным и содержит 9 нейронов, каждый из которых соответствует определенной рубрике.

В качестве функции активации для скрытых слоев была использована активационная функция **ReLu** *A(x)*, определяемая формулой (6). Активационная функция выходного слоя – **Softmax**, которая является многомерным обобщением логистической функции (5).

$$A(x) = \max(0, x) \tag{6}$$

где *x* – аргумент активационной функции, преобразуемое значение.

В обучении ИНС большую роль играет оптимизатор, то есть специализированный алгоритм, осуществляющий модификацию весовых коэффициентов и смещений для связей между нейронами. В данной работе был использован оптимизатор **Adam** (англ. adaptive moment estimation).

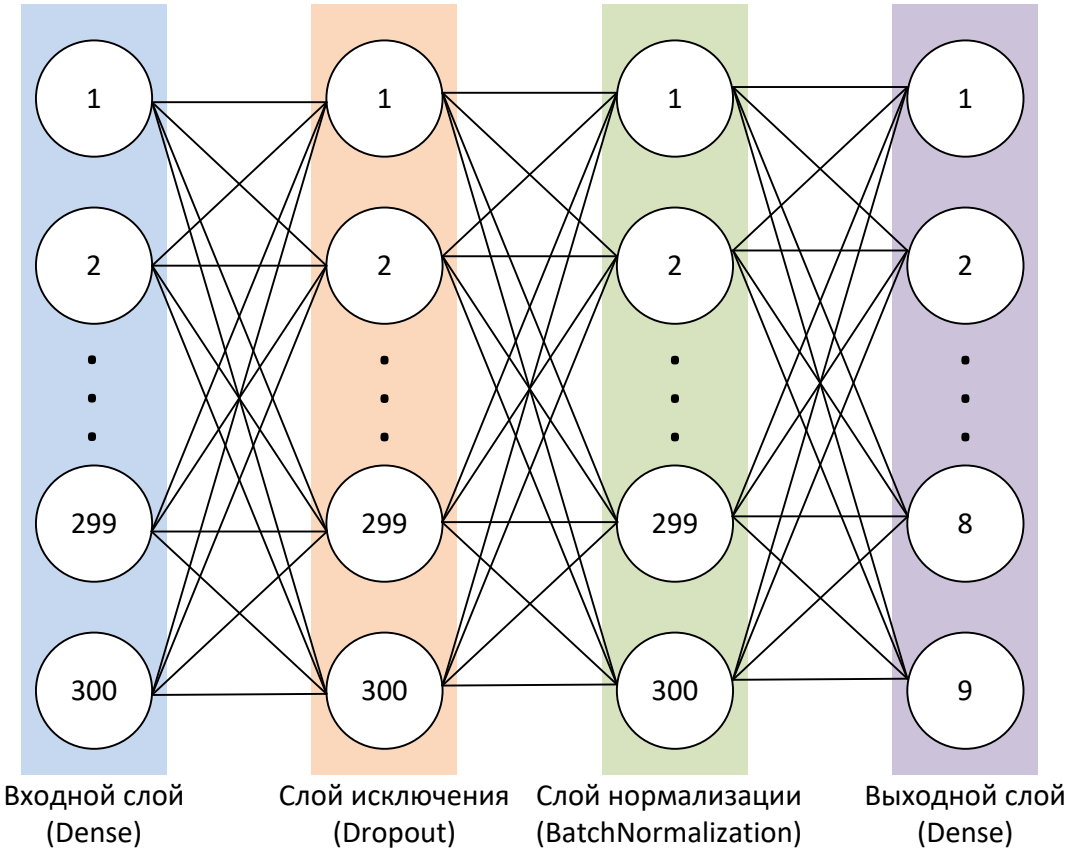

Рис. 15. Структура искусственной нейронной сети

В соответствии с алгоритмом решетчатого поиска было построено несколько искусственных нейронных сетей с различными значениями коэффициента исключения Dropout-слоя при различных значениях параметра оптимизатора. Обучение велось при 50 эпохах, то есть ИНС обучалась на одном тренировочном наборе 50 раз подряд. На каждой эпохе значение точности ИНС фиксировалось. С программным кодом разработанной ИНС можно ознакомиться в файле **network.py**, содержимое которого приведено в приложении Б.

Наивысшее значение точности было получено при значении коэффициента оптимизатора 0,0001 и коэффициенте исключения Dropout-слоя 0,25. Зависимость точности модели от числа эпох для названного выше случая приведена на рисунке 16. Обученная нейронная сеть была сохранена в виде файла **network.json**, а ее веса – в файле **network.h5**.

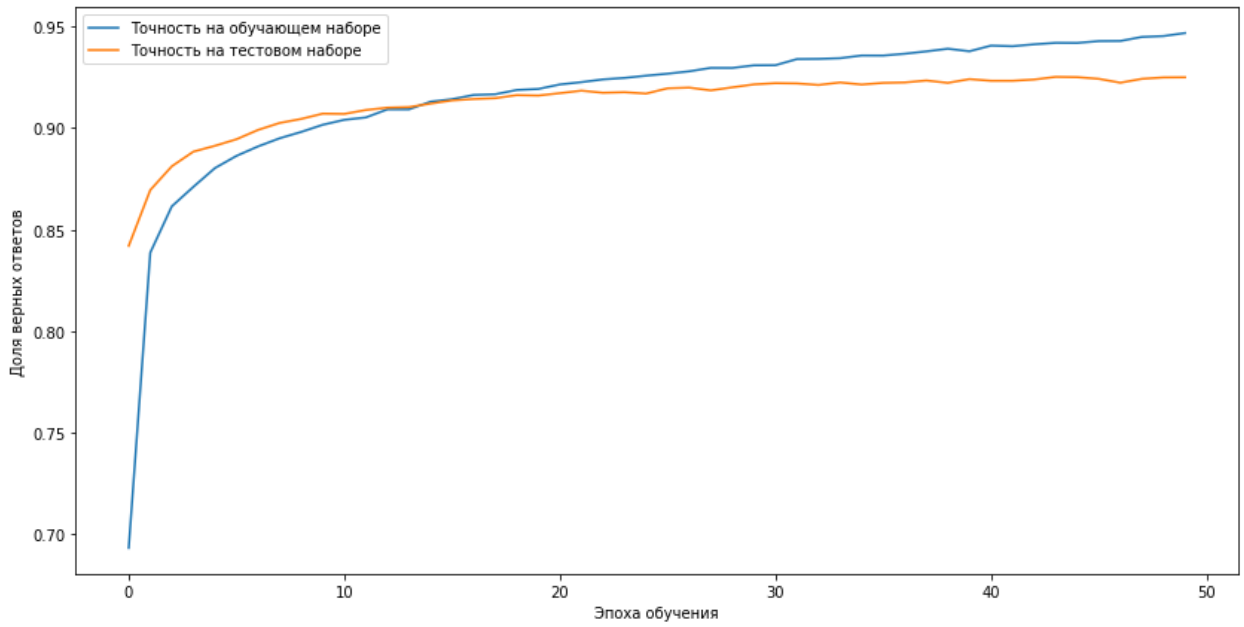

Рис. 16. Зависимость точности ИНС от числа эпох

По графику видно, что при количестве эпох больше 15 точность ИНС на обучающем наборе продолжает возрастать, в то время как на тестовом наборе практически не изменяется.

3.5. Оценка качества моделей классификации

Для оценки качества моделей классификации используются различные метрики, которые позволяют определить, насколько велика обобщающая способность построенной модели классификации. Рассмотрим некоторые из них.

Матрица ошибок – матрица, у которой на пересечении *i*-ой строки и *j*-ого столбца расположено число, равное числу объектов, которые относятся к *j*-ому классу, но были классифицированы моделью машинного обучения как относящиеся к *i*-му классу [42].

В качестве метрик оценки качества модели классификации были использованы метрики **точности** (англ. precision) и **полноты** (англ. recall), а также **F-мера**, которые рассматриваются для каждого класса в отдельности [36, 43].

Метрики точности $\boldsymbol{precision}$ и полноты $\boldsymbol{recall}$ определяются по формулам (7) и (8) соответственно.

$$precision = \frac{TP}{TP+FP} \tag{7}$$

$$recall = \frac{TP}{TP+FN} \tag{8}$$

TP – количество объектов, которые были правильно классифицированы как относящие к данному классу;

TN – количество объектов, которые были правильно классифицированы как не относящиеся к данному классу;

FP – количество объектов, которые были ошибочно классифицированы как относящиеся к данному классу;

FN – количество элементов, которые были ошибочно классифицированы как не относящиеся к данному классу.

Значения описанных выше метрик может лежать в отрезке от 0 до 1. Чем ближе значения полученных метрик точности и полноты к 1, тем выше точность классификации.

В качестве комбинированной метрики классификации используется F-мера, которая определяется в соответствии с формулой (5), где параметр β имеет смысл веса метрики точности.

$$F_\beta = (1 + \beta^2) \frac{precision \cdot recall}{(\beta^2 \cdot precision) + recall} \quad (8)$$

Частным случаем F-меры является F_1-мера, в которой $\beta=1$.

На рисунках 17-24 для каждой модели классификации представлены матрица ошибок и значения метрик точности и полноты, определенные с использованием метода **classification_report** (отчет о классификации) библиотеки Scikit-learn.

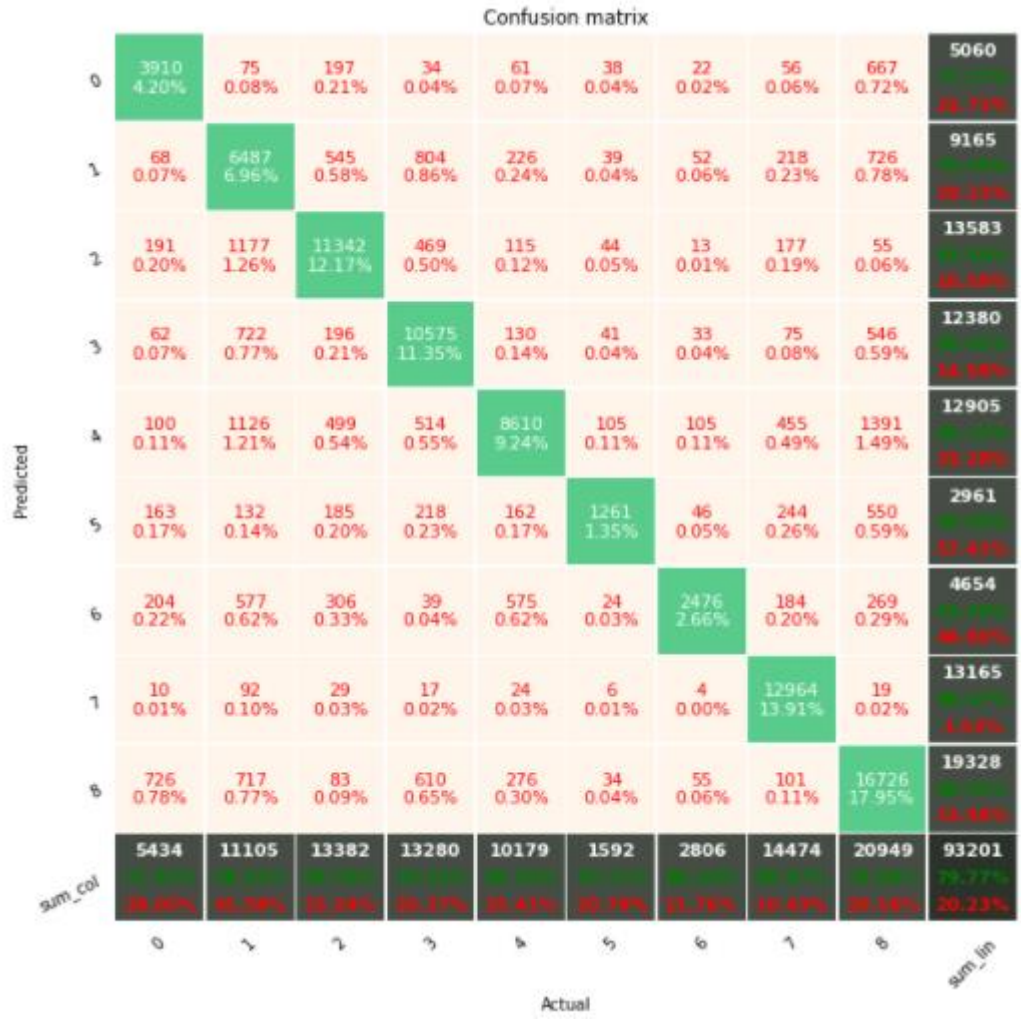

Рис. 17. Матрица ошибок для наивного байесовского классификатора

	precision	recall	f1-score	support
0	0.77273	0.71954	0.74519	5434
1	0.70780	0.58415	0.64006	11105
2	0.83501	0.84756	0.84124	13382
3	0.85420	0.79631	0.82424	13280
4	0.66718	0.84586	0.74597	10179
5	0.42587	0.79209	0.55392	1592
6	0.53202	0.88239	0.66381	2806
7	0.98473	0.89568	0.93809	14474
8	0.86538	0.79842	0.83055	20949
accuracy			0.79775	93201
macro avg	0.73832	0.79578	0.75367	93201
weighted avg	0.81459	0.79775	0.80123	93201

Рис. 18. Отчет о классификации для наивного байесовского классификатора

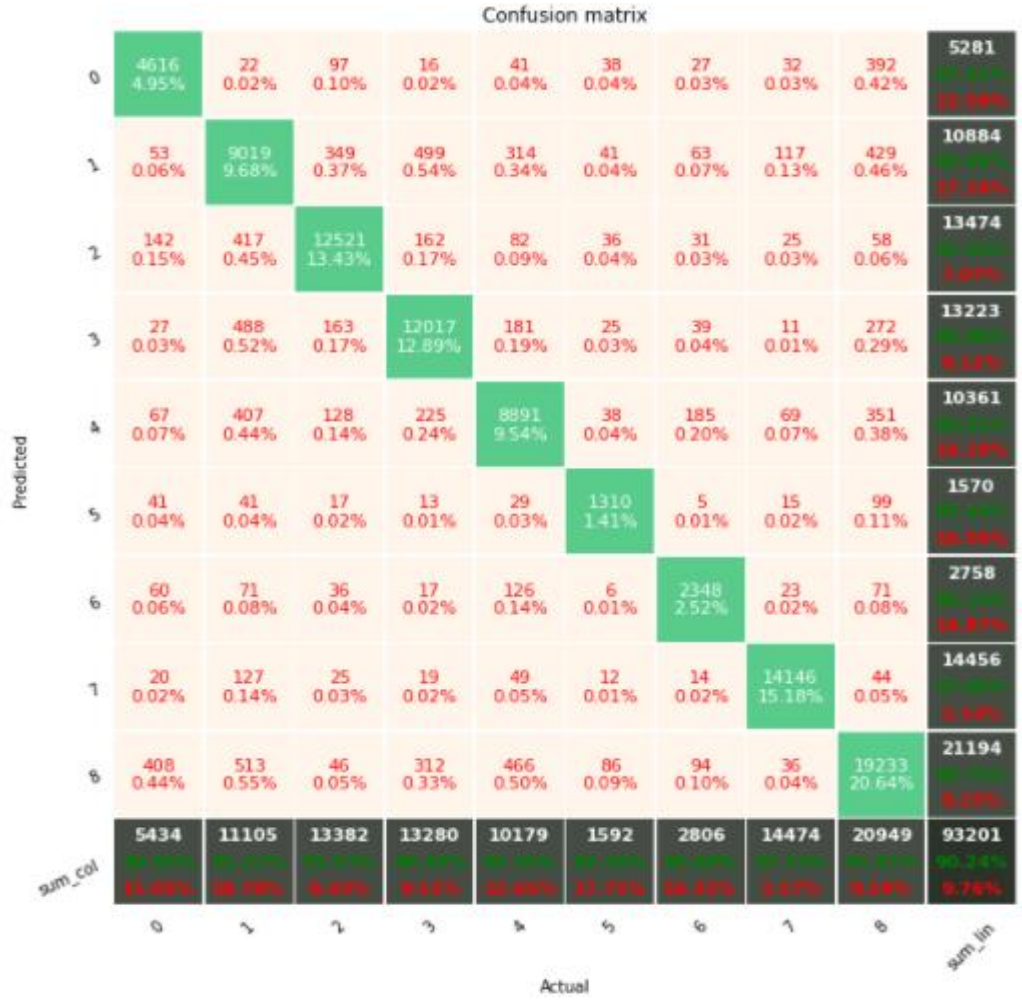

Рис. 19. Матрица ошибок для логистической регрессии

	precision	recall	f1-score	support
0	0.87408	0.84947	0.86160	5434
1	0.82865	0.81216	0.82032	11105
2	0.92927	0.93566	0.93245	13382
3	0.90880	0.90489	0.90684	13280
4	0.85812	0.87346	0.86573	10179
5	0.83439	0.82286	0.82859	1592
6	0.85134	0.83678	0.84400	2806
7	0.97856	0.97734	0.97795	14474
8	0.90747	0.91809	0.91275	20949
accuracy			0.90236	93201
macro avg	0.88563	0.88119	0.88336	93201
weighted avg	0.90216	0.90236	0.90222	93201

Рис. 20. Отчет о классификации для логистической регрессии

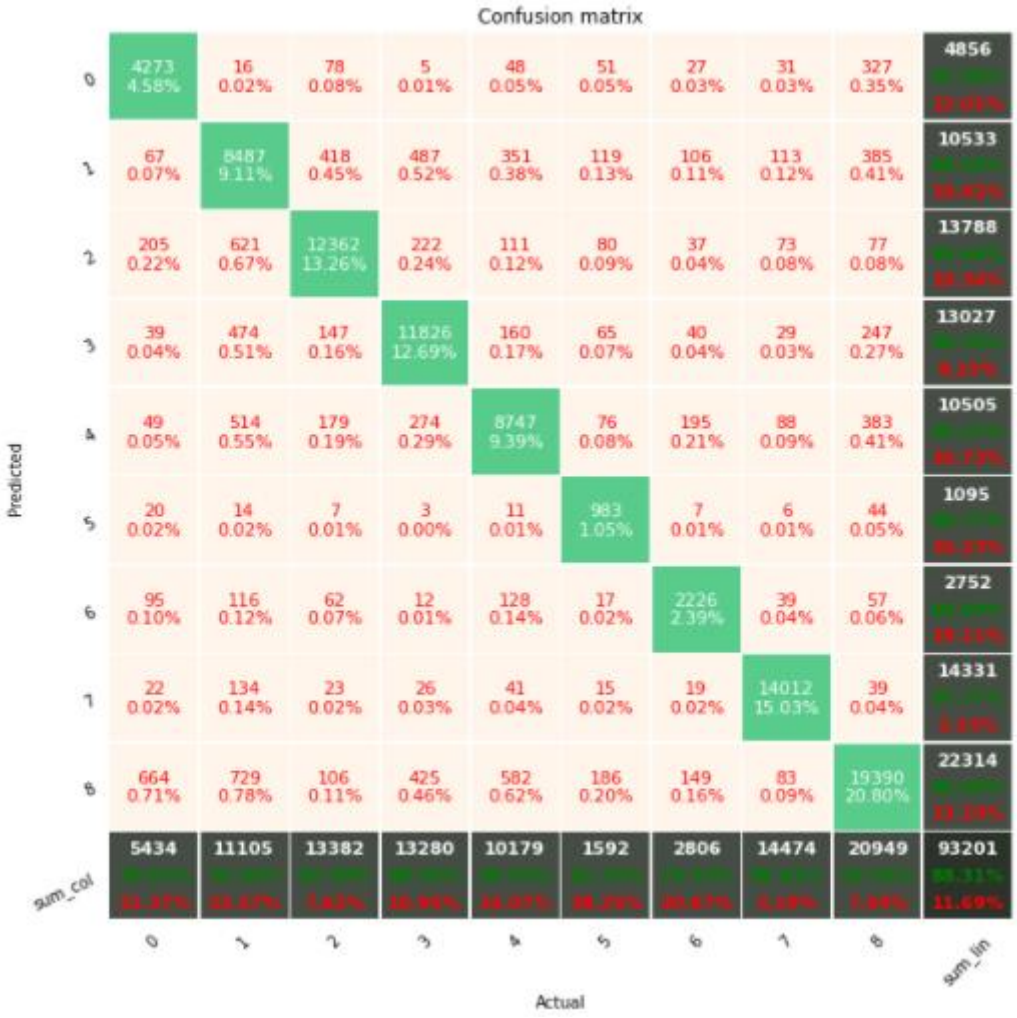

Рис. 21. Матрица ошибок для случайного леса решающих деревьев

	precision	recall	f1-score	support
0	0.87994	0.78635	0.83052	5434
1	0.80575	0.76425	0.78445	11105
2	0.89658	0.92378	0.90997	13382
3	0.90781	0.89051	0.89908	13280
4	0.83265	0.85932	0.84577	10179
5	0.89772	0.61746	0.73167	1592
6	0.80887	0.79330	0.80101	2806
7	0.97774	0.96808	0.97289	14474
8	0.86896	0.92558	0.89638	20949
accuracy			0.88310	93201
macro avg	0.87511	0.83651	0.85242	93201
weighted avg	0.88318	0.88310	0.88221	93201

Рис. 22. Отчет о классификации для случайного леса решающих деревьев

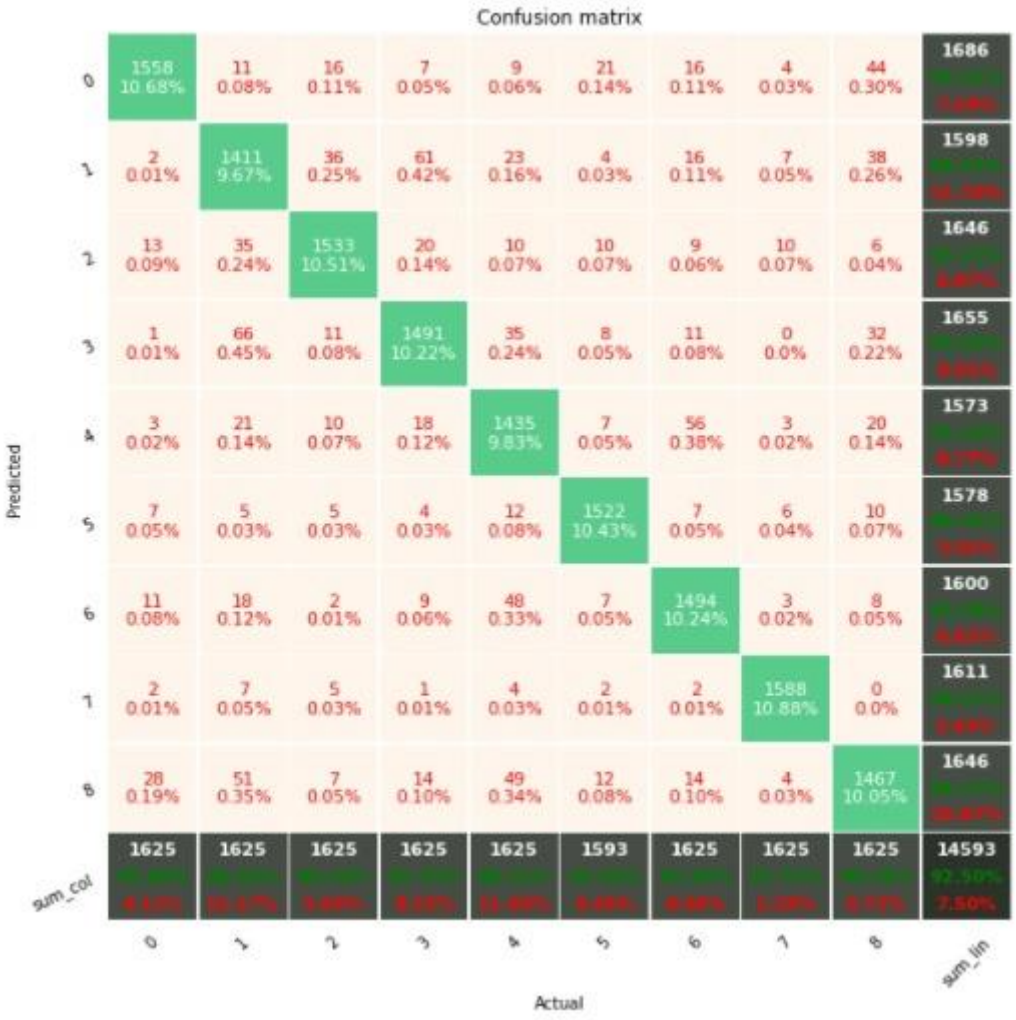

Рис. 23. Матрица ошибок для ИНС

	precision	recall	f1-score	support
0	0.9588	0.9241	0.9411	1686
1	0.8683	0.8830	0.8756	1598
2	0.9434	0.9313	0.9373	1646
3	0.9175	0.9009	0.9091	1655
4	0.8831	0.9123	0.8974	1573
5	0.9554	0.9645	0.9599	1578
6	0.9194	0.9337	0.9265	1600
7	0.9772	0.9857	0.9815	1611
8	0.9028	0.8913	0.8970	1646
accuracy			0.9250	14593
macro avg	0.9251	0.9252	0.9251	14593
weighted avg	0.9253	0.9250	0.9251	14593

Рис. 24. Отчет о классификации для ИНС

В таблице 3 представлены сводные результаты оценки качества построенных моделей классификации

Таблица 3.

Сводная таблица метрик для построенных моделей классификации

Модель классификации	**Среднее взвешенное значение метрики точности**	**Среднее взвешенное значение метрики полноты**	**Среднее взвешенное значение F_1-меры**
Наивный байесовский классификатор	0,81459	0,79775	0,75367
Логистическая регрессия	0,90216	0,90236	0,90222
Случайный лес решающих деревьев	0,88318	0,88310	0,88221
Искусственная нейронная сеть	0,9253	0,9250	0,9251

По результатам проведенной оценки качества построенных моделей классификации можно сделать следующие выводы:

- все разработанные в рамках данной работы модели классификации проверены и работают в штатном режиме;

- все построенные модели классификации обладают обобщающей способностью, что означает, что данные модели классификации были успешно обучены;

- наилучшие значения метрик качества имеет модель классификации на основе искусственной нейронной сети;

- случайный лес решающих деревьев и логистическая регрессия показывает значения метрик качества классификации ниже искусственной нейронной сети;

- наивный байесовский классификатор как самая простая из всех построенных моделей показывает наихудшие результаты, хотя даже в этом случае можно признать, что обучение данной модели является успешным;

- для дальнейшего использования наиболее пригодной является модель классификации на основе искусственной нейронной сети.

4. РАЗРАБОТКА ВЕБ-САЙТА

Для разработки веб-сайта был использован фреймворк **Django** языка программирования Python. Для задания разметки веб-страниц был использован язык гипертекстовой разметки **HTML5**, для задания стиля веб-страниц была использована каскадная таблица стилей **CSS**. Разработка велась в среде **Spyder 4.1.3**. Для хранения данных была выбрана СУБД **MySQL 8.0**. Для работы с СУБД был использован **MySQL Workbench 8.0**.

4.1. Разработка структуры веб-сайта

При создании Django-проекта автоматически создается структурированная система директорий с определенным набором файлов. Правильная структура Django-проекта необходима для его корректной работы. Коротко рассмотрим некоторые необходимые в Django-проекте программные файлы: manage.py, settings.py, urls.py.

Программный файл **manage.py** – скрипт, включенный в Django-проект и предназначенный для управления им. Именно в данный файл в качестве передаваемых при вызове консоли параметров разработчик подает команды управления локальным сервером Django.

Программный файл **settings.py** содержит все глобальные настройки для Django-проекта, например: полный путь к директории проекта, секретный ключ веб-сайта, встроенные в проект приложения, используемые шаблоны, используемая база данных, язык веб-сайта и панели администратора и др.

Программный файл **urls.py** помогает отслеживать URL-адреса внутри проекта.

Django-проект состоит из приложений. **Приложение** в Django – это отдельная функциональная и структурная единица веб-сайта. Приложения используются для декомпозиции проекта, разделения на отдельные функциональные блоки. Каждое приложение реализует некоторую функционально-структурную категорию веб-сайта.

Разработанный Django-проект содержит единственное приложение **main**. Рассмотрим коротко некоторые программные файлы, которые по умолчанию должны входить в Django-приложение: models.py и view.py.

Программный файл **models.py** содержит информацию о моделях. **Модель** – это, по сути, объектно-ориентированный аналог базы данных. Такой подход в полной мере соответствует методологии объектно-реляционного отображения Django ORM.

Программный файл **view.py** является файлом представлений и содержит функции, которые осуществляют переход на некоторую HTML-страницу. Внутри данных функций также возможно сформировать передаваемое в HTML-страницу содержимое в виде стандартного словаря языка Python. Содержимое данного словаря может быть отображено на HTML-странице при помощи стандартного функционала шаблонизатора Django.

Разработанный веб-сайт содержит заглавную страницу, страницу «О проекте», а также ряд HTML-страниц, каждая из которых относится к некоторой рубрике. В таблице 4 представлен список HTML-страниц разработанного веб-сайта, их предназначение, а также их URL-адреса относительно адреса в веб-сайта.

Таблица 4.

Структура веб-сайта

HTML-страница	**Предназначение HTML-страницы**	**Относительный URL-адрес**
Заглавная	Заглавная страница веб-сайта	
О проекте	Содержит сведения об ответственности и целях разработки	/about
Дом	Новости, отнесенные моделью классификации к рубрике «Дом»	/home
Интернет и СМИ	Новости, отнесенные моделью классификации к рубрике «Интернет и СМИ»	/internet
Культура	Новости, отнесенные моделью классификации к рубрике «Культура»	/culture
Наука и техника	Новости, отнесенные моделью классификации к рубрике «Наука и техника»	/science
Политика	Новости, отнесенные моделью классификации к рубрике «Политика»	/politics
Путешествия	Новости, отнесенные моделью классификации к рубрике «Путешествия»	/trips
Силовые структуры	Новости, отнесенные моделью классификации к рубрике «Силовые структуры»	/military
Спорт	Новости, отнесенные моделью классификации к рубрике «Спорт»	/sport
Экономика и бизнес	Новости, отнесенные моделью классификации к рубрике «Экономика и бизнес»	/economics

Обработка URL-запроса внутри Django-проекта производится в соответствии со следующей схемой: вначале в соответствии с информацией, хранящейся в файле urls.py, определяется Django-приложение, в которое должен быть передан запрос. Получив некоторый URL-запрос от пользователя, Django-приложение в соответствии с информацией, хранящейся в его файле urls.py, осуществит вызов соответствующей запросу функции, которая хранится в файле представлений views.py, а уже сама функция, в свою очередь, производит отображение содержимого HTML-шаблона с использованием функции **render** фреймворка Django. При этом HTML-шаблон должен содержаться в директории **templates**, включенной в директорию приложения, в противном случае добиться корректной работы невозможно.

4.2. Работа с базой данных

Для хранения новостных статей, которые в дальнейшем распределяются по тематическим рубрикам и, соответственно, по веб-страницам, была создана база данных **newssite**, содержащая в себе ряд служебных таблиц фреймворка Django, а также созданную в процессе разработки таблицу **main_articles**. Таблица была создана при помощи инструментария фреймворка Django: вначале была описана модель базы данных и ее структура, а затем были произведены миграции, то есть отображения объектно-ориентированных моделей на реляционную базу данных. Подробнее с принципом миграции можно ознакомиться в [15].

На рисунке 25 представлена структура таблицы main_articles. Поле **id** содержит порядковый номер новостной статьи, он же выступает первичным ключом, поле **url** – URL-ссылку на оригинальную новостную статью. Заголовок статьи хранится в поле **title**, а ее полный текст – в поле **text**. Поле **date** содержит информацию о дате новостной публикации, а поле **topic** хранит числовой идентификатор рубрики. По умолчанию в поле topic хранится значение -1, которое изменяется после автоматической рубрикации.

В таблице 5 содержатся сведения о типах данных для каждого поля созданной таблицы main_articles базы данных newssite и соответствующий им тип данных внутри модели фреймворка Django, указанный при создании данной модели.

Таблица 5.

Типы данных полей базы данных

Поле	Тип данных MySQL	Тип данных Django
id	bigint	models.BigAutoField
url	varchar(200)	models.URLField
title	varchar(100)	models.CharField
text	longtext	models.TextField
topic	int	models.IntegerField
date	date	models.DateField

	id	url	title	text	topic	date
▶	1	https://lenta.ru/news/2018/11/15/zvezda/	На «Звезде» состоялась церемония резки ме…	На судостроительном комплексе «Звезда» в …	8	2018-11-15
	2	https://lenta.ru/news/2018/11/15/macron_trump/	Франция отказалась раболепствовать пере…	Президент Франции Эммануэль Макрон отве…	4	2018-11-15
	3	https://lenta.ru/news/2018/11/15/bestsexever/	Названы лучшие сцены секса в кино	Издание The Independent составило список и…	2	2018-11-15
	4	https://lenta.ru/news/2018/11/15/metalloiskatel/	Кремль прояснил историю с металлоискател…	Пресс-секретарь президента России Дмитри…	4	2018-11-15
	5	https://lenta.ru/news/2018/11/15/zakon/	Статью об экстремизме в России смягчат до …	Закон о частичной декриминализации стать…	4	2018-11-15
	6	https://lenta.ru/news/2018/11/15/aurus/	Чиновников пересадят на лимузины проекта…	Управделами президента планирует при обн…	4	2018-11-15
	7	https://lenta.ru/news/2018/11/15/instagramchik/	Возвращение Кадырова в Instagram было ош…	Аккаунт главы Чечни Рамзана Кадырова в I…	1	2018-11-15
	8	https://lenta.ru/news/2018/11/15/tribunal/	Россиянин сел на 14 лет за отрубленные рук…	Отрубивший руки жене 27-летний Дмитрий …	6	2018-11-15
	9	https://lenta.ru/news/2018/11/15/fsb/	Сибирячка перевела миллион боевикам ИГ	Жительница Томской области перевела окол…	6	2018-11-15
	10	https://lenta.ru/news/2018/11/15/garant_kach…	Объявили победителей конкурса «Гарантия…	8 ноября в Совете Федерации состоялось на…	8	2018-11-15
	11	https://lenta.ru/news/2018/11/15/rosn_prim/	«Роснефть» и администрация Приморья дого…	«Роснефть» и администрация Приморского к…	8	2018-11-15
	12	https://lenta.ru/news/2018/11/15/luxusy_amb…	Трамп назначит послом США в Южной Африк…	Президент США Дональд Трамп намерен наз…	1	2018-11-15
	13	https://lenta.ru/news/2018/11/15/investment/	Российская недвижимость привлечет четыр…	В 2018 году объем инвестиций в недвижимос…	0	2018-11-15
	14	https://lenta.ru/news/2018/11/15/jim/	Кадыров подарил Mercedes отжавшемуся че…	Глава Чечни Рамзан Кадыров встретился с п…	7	2018-11-15
	15	https://lenta.ru/news/2018/11/15/neutrality/	Австрия отказалась от участия в европейск…	Австрия не сможет стать частью армии Евро…	4	2018-11-15
	16	https://lenta.ru/news/2018/11/15/gaz/	«Газпром» столкнулся с угрозой огромного …	Госкомпании «Газпром» грозит многомиллиа…	8	2018-11-15
	17	https://lenta.ru/news/2018/11/15/signaldome/	В Москве на Signal Dome выступят Ancient Me…	В Москве пройдет вечеринка Signal Dome — …	2	2018-11-15
	18	https://lenta.ru/news/2018/11/15/razan/	Освобождавший зеков за взятки российский…	В суд направлено дело в отношении бывшег…	6	2018-11-15
	19	https://lenta.ru/news/2018/11/15/besiktas/	Футболист подкрепился в перерыве матча и…	Немецкий полузащитник турецкого «Бешикт…	7	2018-11-15
	20	https://lenta.ru/news/2018/11/15/repost/	Законопроект о репостах приняли в «пример…	Госдума приняла в первом чтении пакет пре…	4	2018-11-15

Рис. 25. Структура таблицы main_articles

Модель классификации на основе искусственной нейронной сети была использована для разработки программного файла **rubricate.py**, осуществляющего рубрикацию в режиме реального времени. Данная программа с интервалом в 100 секунд считывает еще нерубрицированные записи таблицы main_articles и проводит их рубрикацию, по результатам которой модифицирует поле topic. Данная программа может быть остановлена пользователем вызовом прерывания **KeyboardInterrupt**, причем переопределение обработчика данного прерывания, осуществленное в программе, обеспечивает не аварийную, как обычно, а корректную остановку выполнения программы.

Любой Django-проект по умолчанию содержит встроенную панель администратора, которая является удобным инструментом для работы с базой данных. Пользовательский интерфейс панели администратора позволяет читать содержимое таблиц базы данных, редактировать имеющиеся в ней записи и удалять их. Панель администратора Django-проекта представлена на рисунке 26. С ее помощью можно добавлять новых пользователей и удалять уже существующих, добавлять и удалять группы пользователей, работать с другими таблицами. На рисунке 27 представлен графический интерфейс страницы панели

администратора, позволяющей осуществить редактирование записи таблицы базы данных main_articles.

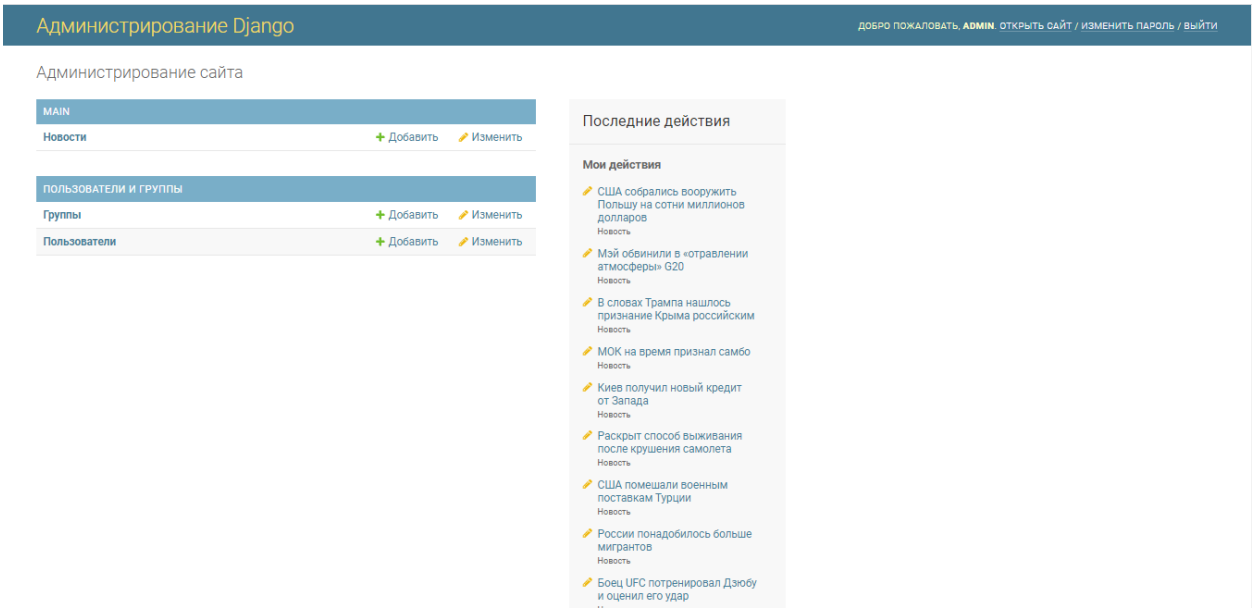

Рис. 26. Пользовательский интерфейс панели администратора

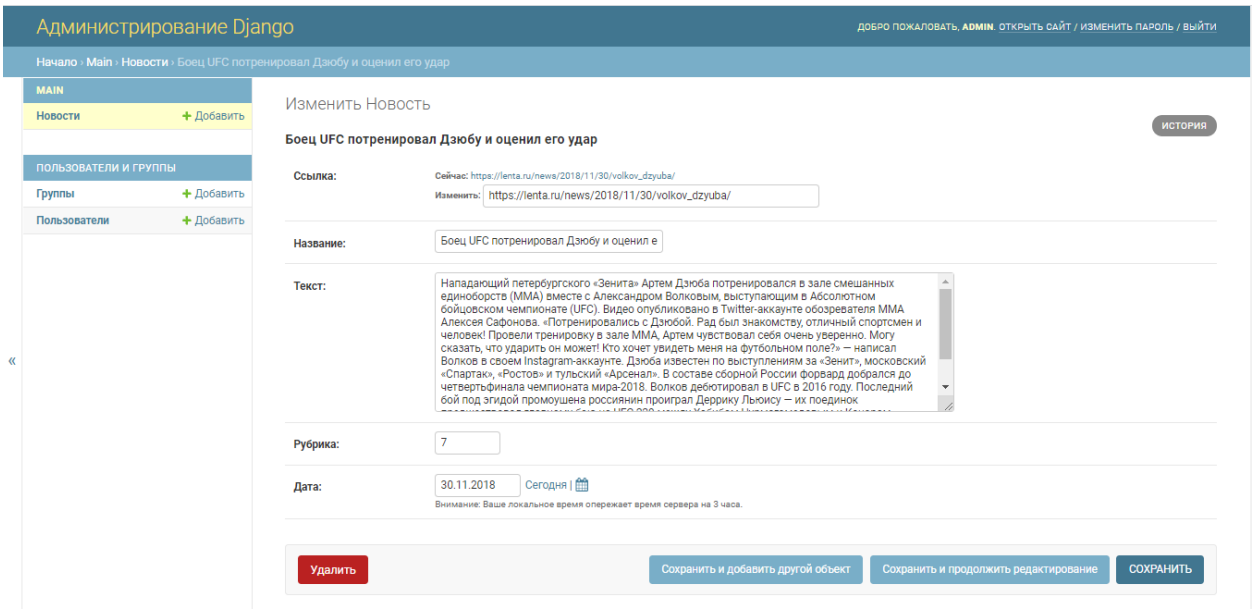

Рис. 27. Редактирование базы данных через панель администратора

4.3. Разработка веб-сайта и пользовательского интерфейса

Веб-страницы, содержащие информацию по соответствующим рубрикам, идентичны по своему функционалу и оформлению, а различаются лишь наполнением. В целях избежания дублирования кода HTML-страниц был использован шаблонизатор фреймворка Django. Были созданы два HTML-шаблона: **layout.html**, наследниками которого являются заглавная страница сайта и страница «О проекте», и **news_layout.html**, от которого наследуются веб-страницы отдельных рубрик. Внутри шаблонов-родителей имеются динамически изменяемые секции, которые в шаблонах-наследниках заменяются на конкретный HTML-код. За счет этого удается при минимальном дублировании кода добиться функционального и стилистического единства, обеспечив при этом различные HTML-страницы различным наполнением.

Для задания стиля веб-страниц была использована каскадная таблица стилей **CSS**, а также фреймворк **Bootstrap**, разработанный компанией **Twitter** [44]. **Bootstrap** – свободный CSS-фреймворк, который используется для создания дизайна веб-сайтов. Он включает в себя HTML- и CSS-шаблоны, которые разработчик может использовать для задания оформления форм, кнопок и других элементов веб-страницы. В данной работе для задания стилей веб-сайта был создан файл **main_style.css**, с содержанием которого можно ознакомиться в приложении В.

На рисунке 28 изображена заглавная страница веб-сайта. Рассмотрим интерфейс веб-сайта. Боковая панель 1 присутствует на всех веб-страницах и содержит пункты меню, которые используются для навигации по веб-сайту. Каждый пункт меню носит название, соответствующее тематике своей веб-страницы, например, «Политика», «Спорт», «О проекте» и т.д., при нажатии на пункт меню осуществляется переход на соответствующую ему веб-страницу. Пример веб-страницы отдельной рубрики представлен на рисунке 29. При

наведении курсора на пункт меню при помощи стандартных средств CSS происходит визуальное выделение данного пункта меню, как показано на рисунке 29. Кроме того, боковая панель 1 содержит надпись 2 «Новости», при нажатии на которую происходит переход на заглавную страницу веб-сайта. По нажатии на кнопку 3, оформленную с использованием CSS-стиля **btn-warning** , происходит переход на веб-страницу «О проекте».

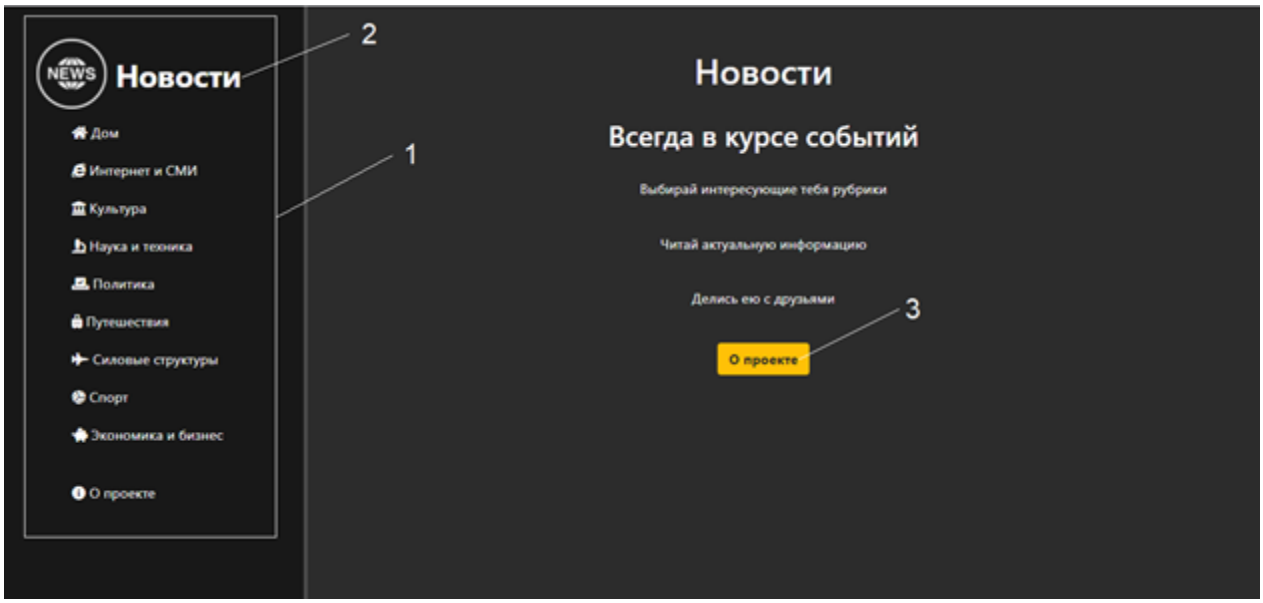

Рис. 28. Заглавная страница веб-сайта

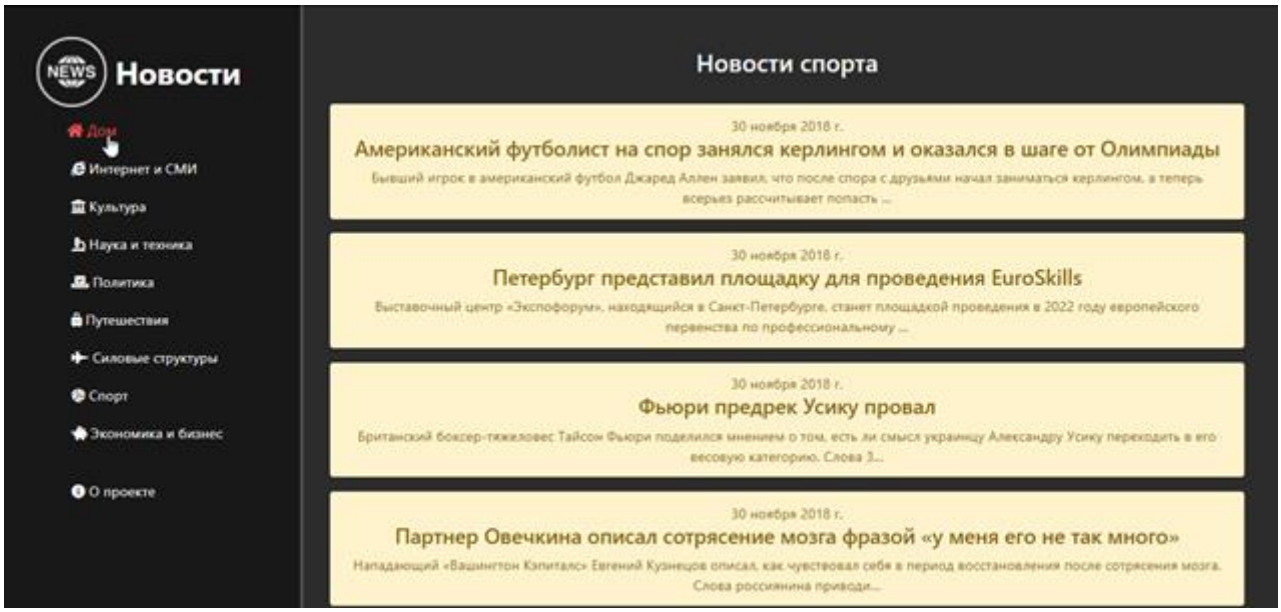

Рис. 29. Страница "Новости спорта"

Изображенная на рисунке 29 веб-страница содержит набор блоков, каждый из которых посвящен отдельной новостной статье и содержит: дату публикации, заголовок статьи и начальную часть ее текста. При нажатии на блок происходит переход на страницу новостного ресурса с оригинальными статьями.

4.4. Тестирование веб-сайта

Тестирование разработанного в рамках данной работы веб-сайта производилось в соответствии с ГОСТ 19.301-79.

Объект испытания: разработанный в рамках данной работы веб-сайт «Новости».

Цель испытания: выявление ошибок функционирования веб-сайта и проверка соответствия работы веб-сайта требованиям технического заданиями на выпускную квалификационную работу (приложение А).

Средства испытания: испытание проводилось на персональном компьютере с ОС Windows 10 с использованием трех различных веб-браузеров: Google Chrome 90.0.4430.212, Яндекс.Браузер 21.5.1.330, Mozilla Firefox 70.0.1.

Методы испытания:

- проверка корректности отображения отдельных элементов веб-сайта и их визуальных эффектов;

- проверка корректности переходов по ссылкам как внутри веб-сайта, так и за его пределы.

Для проверки корректности отображения элементов веб-страниц в вышеназванных трех веб-браузерах были просмотрены все веб-страницы разработанного веб-сайта на предмет корректности цветопередачи элементов, их визуальных эффектов и стилей.

Для проверки корректности переходов по ссылкам были проверены все пункты меню, ссылка «mailto» на странице «О проекте», а также выборочно были проверены переходы при нажатии на некоторые блоки новостных статей. Пример перехода на сторонний новостной ресурс с оригинальной статьей изображен на рисунке 30.

Рис. 30. Пример перехода на сторонний ресурс

Вывод: по итогам тестирования ошибок функционирования веб-сайта не выявлено, все его элементы работают в нормальном режиме. Все пункты технического задания были выполнены.

ЗАКЛЮЧЕНИЕ

В ходе выпускной квалификационной работы были изучены подходы к автоматической рубрикации русскоязычных текстов с использованием машинного обучения. Результатом проведенной разработки является информационная система автоматической рубрикации русскоязычных новостных текстов, состоящая из рубрицирующей модели классификации и веб-сайта для демонстрации полученных результатов.

Для создания рубрицирующей модели классификации были осуществлены подготовка и векторизация корпуса новостных статей, проведен ряд экспериментов по построению рубрицирующих моделей классификации, в рамках которых был обучены модели классификации на основе гауссовского наивного байесовского классификатора, случайного леса решающих деревьев, логистической регрессии и искусственной нейронной сети. Проведена оценка качества построенных моделей классификации по ряду метрик, в результате оценки определено, что модель классификации на основе искусственной нейронной сети обладает наибольшей обобщающей способностью.

Веб-сайт был разработан с использованием фреймворка Django языка программирования Python, языка гипертекстовой разметки HTML, каскадной таблицы стилей CSS, в качестве СУБД использована MySQL. Разработанный веб-сайт имеет интуитивно понятный интерфейс, нацелен на максимально удобную работу пользователя. Веб-сайт был протестирован в соответствии с ГОСТ 19.301-79, по итогам тестирования ошибок функционирования веб-сайта не выявлено.

Полученные в данной работе практические результаты можно использовать как часть более крупного новостного ресурса. Кроме того, разработка может быть продолжена в направлении создания новостного агрегатора реального времени.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Поручиков, М.А. Анализ данных: учеб. пособие / М.А. Поручиков. – Самара: Изд-во Самарского университета, 2016. – 88 с.: ил. – ISBN 978-5-7883-1085-5.

2. Машинное обучение: [Электронный ресурс] // Машинное обучение. URL: http://www.machinelearning.ru/wiki/index.php?title=Машинное_обучение. (Дата обращения: 12.05.2021).

3. Обработка естественного языка (NLP) для машинного обучения: [Электронный ресурс] // Машинное обучение. URL: https://www.machinelearningmastery.ru/natural-language-processing-nlp-for-machine-learning-d44498845d5b/. (Дата обращения: 21.02.2021).

4. Что такое NLP: история, терминология, библиотеки: [Электронный ресурс] // Python School. URL: https://python-school.ru/wiki/nlp/. (Дата обращения: 11.03.2021).

5. Reinsel, D. The Digitalization of the World / D. Reinsel, J. Gantz, J. Rydning – 2018. – 28 с.: [Электронный ресурс]. – URL: https://www.seagate.com/files/www-content/our-story/trends/files/idc-seagate-dataage-whitepaper.pdf. (Дата обращения: 11.03.2021).

6. Воронцов, К.В. Математические методы машинного обучения по прецедентам (теория обучения машин) / К.В. Воронцов. – 141 с.: [Электронный ресурс]. – URL: http://www.machinelearning.ru/wiki/images/6/6d/Voron-ML-1.pdf. (Дата обращения: 14.01.2021).

7. Шаграев, А.Г. Модификация, разработка и реализация методов классификации новостных текстов: дисс. … канд. технических наук: 05.13.17. – НИУ «МЭИ», Москва, 2014. – 108 с.

8. LENTA.RU: [Электронный ресурс]. URL: https://lenta.ru/. (Дата обращения: 08.04.2021).

9. Яндекс.Новости: [Электронный ресурс]. URL: https://yandex.ru/news/. (Дата обращения: 11.03.2021).

10. News dataset from Lenta.ru [Электронный ресурс] // Kaggle: Your Home for Data Science. URL: https://www.kaggle.com/yutkin/corpus-of-russian-news-articles-from-lenta. (Дата обращения 08.02.2021)

11. Список библиотек Python: [Электронный ресурс] // Python Дайджест. URL: https://pythondigest.ru/view/1041/. (Дата обращения: 11.03.2021).

12. TIOBE Index for June 2021: [Электронный ресурс] // TIOBE. URL: https://tiobe.com/tiobe-index/. (Дата обращения: 11.03.2021).

13. Лутц, М. Изучаем Python. В 2 тт. Т.1 / М. Лутц; пер. с англ. – 5 изд. – СПб.: Диалектика, 2019. – 832 с. – ISBN 978-5-907144-52-1.

14. Какой язык программирования выбрать специалисту по машинному обучению?: [Электронный ресурс] // Skillbox. URL: https://skillbox.ru/media/code/kakoy_yazyk_programmirovaniya_vybrat_spetsialistu_po_mashinnomu_obucheniyu/. (Дата обращения: 11.03.2021).

15. Дронов, В.А. Django: практика создания Web-сайтов на Python / В.А. Дронов. – СПб.: БХВ-Петербург, 2016. 516 с. – ISBN 978-5-9775-0421-8.

16. Official site of Django framework: [Электронный ресурс]. URL: https://www.djangoproject.com/start/overview/. (Дата обращения: 17.03.2021).

17. Official site of Flask framework: [Электронный ресурс]. URL: http://flask.pocoo.org/. (Дата обращения: 17.03.2021).

18. Тарасов, С.В. СУБД для программиста. Базы данных изнутри / С.В. Тарасов. – М: СОЛОН-Пресс, 2015. 321 с.:ил. – ISBN 978-2-7466-7383-0.

19. SQLite vs MySQL vs PostgreSQL: сравнение систем управления базами данных: [Электронный ресурс] // DevAcademy. URL: https://devacademy.ru/article/sqlite-vs-mysql-vs-postgresql/. (Дата обращения: 17.03.2021).

20. Грубер, М. Понимание SQL / М. Грубер; пер. с англ. В.Н. Лебедева, ред. В.Н. Булычева. – М, 1993. – 291 с.

21. Anaconda: [Электронный ресурс]. URL: https://www.anaconda.com/. (Дата обращения: 11.03.2021).

22. Рашка, С. Python и машинное обучение / С. Рашка; пер. с англ. А. В. Логунов; под ред. Д. А. Мовчан. — М.: ДМК Пресс, 2017. — 418 с.: ил. — ISBN: 978-5-97060-409-0.

23. Как решить 90% задач NLP: пошаговое руководство по обработке естественного языка: [Электронный ресурс] // Хабр. URL: https://habr.com/ru/company/oleg-bunin/blog/352614/. (Дата обращения: 12.02.2021).

24. NLTK 3.6.2 documentation: [Электронный ресурс]. URL: https://www.nltk.org/. (Дата обращения 14.04.2021).

25. Предобработка текста в NLP: [Электронный ресурс] // Python School. URL: https://python-school.ru/nlp-text-preprocessing/. (Дата обращения: 02.03.2021).

26. Korobov, M.: Morphological Analyzer and Generator for Russian and Ukrainian Languages // Analysis of Images, Social Networks and Texts, pp. 320-332 (2015).

27. Морфологический анализатор pymorphy2: [Электронный ресурс]. URL: https://pymorphy2.readthedocs.io/en/stable/. (Дата обращения: 03.05.2021).

28. Чудесный мир Word Embeddings: какие они бывают и зачем нужны: [Электронный ресурс] // Хабр. URL: https://habr.com/ru/company/ods/blog/329410/. (Дата обращения: 15.04.2021).

29. Жеребцова, Ю.А., Чижик А.В. Сравнение моделей векторного представления текстов в задаче создания чат-бота. // Вестник НГУ. 2020. Т.18. URL: https://cyberleninka.ru/article/n/sravnenie-modeley-vektornogo-predstavleniya-tekstov-v-zadache-sozdaniya-chat-bota/viewer. (Дата обращения: 19.03.2021).

30. RusVectores: семантические модели для русского языка: [Электронный ресурс]. URL: https://rusvectores.org/ru/. (Дата обращения: 14.02.2021).

31. Gensim – Руководство для начинающих: [Электронный ресурс] // Еще один блог веб-разработчика. URL: https://webdevblog.ru/gensim-rukovodstvo-dlya-nachinajushhih/. (Дата обращения: 16.03.2021).

32. Куратов, Ю.М. Специализация языковых моделей для применения к задачам обработки естественного языка: дисс. … канд. физико-математических наук: 05.13.17. – МФТИ, Москва, 2020. – 121 с.

33. Varoquaux, G. Scipy Lecture Notes / G. Varoquaux, E. Gouillart, O. Vahtras, P. Buyl. и др. – 2020. – 674 с.: [Электронный ресурс]. – URL: http://scipy-lectures.org/. (Дата обращения: 08.10.2021).

34. Pandas – Python Data Analysis Library: [Электронный ресурс]. URL: https://pandas.pydata.org/. (Дата обращения: 05.02.2021).

35. Scikit-learn. Machine Learning in Python: [Электронный ресурс]. URL: https://scikit-learn.org/stable/. (Дата обращения: 12.04.2021).

36. Мюллер, А. Введение в машинное обучение с помощью Python / А. Мюллер, С. Гвидо; пер. с англ. – М.: Альфа-книга, 2018. – 480 с.: ил. – ISBN: 978-1-449-36941-5.

37. Наивный байесовский классификатор: [Электронный ресурс] // NOP::Nuances of programming. URL: https://nuancesprog.ru/p/10732/. (Дата обращения: 14.05.2021).

38. Флах, П. Машинное обучение. Наука и искусство построения алгоритмов, которые извлекают знания из данных / П. Флах; пер. с англ. А.А. Слинкина. – М.: ДМК Пресс, 2015. – 400 с.: ил.

39. Абдрахманов, М.И. Библиотека matplotlib / М.И. Абдрахманов. – 2019. – 124 с.: [Электронный ресурс]. URL: https://devpractice.ru/files/books/python/Matplotlib.book.pdf. (Дата обращения: 03.05.2021).

40. Типы нейронных сетей. Принцип их работы и сфера применения: [Электронный ресурс]. URL: https://otus.ru/nest/post/1263/. (Дата обращения 04.05.2021).

41. Keras: the Python deep learning API: [Электронный ресурс]. URL: https://keras.io/. (Дата обращения: 18.05.2021).

42. Что такое матрица путаницы в машинном обучении: [Электронный ресурс] // Машинное обучение. URL: https://www.machinelearningmastery.ru/confusion-matrix-machine-learning/. (Дата обращения: 14.05.2021).

43. Метрики в задачах машинного обучения: [Электронный ресурс] // Хабр. URL: https://habr.com/ru/company/ods/blog/328372/. (Дата обращения: 11.05.2021).

44. Bootstrap: [Электронный ресурс]. URL: https://getbootstrap.com/. (Дата обращения: 02.06.2021).

Приложение А. Техническое задание

МИНОБРНАУКИ РОССИИ
федеральное государственное бюджетное образовательное
учреждение высшего образования
«Национальный исследовательский университет «МЭИ»

Институт ИВТИ
Кафедра ВМСС

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
(бакалаврскую работу)

Направление 09.03.01 Информатика и вычислительная техника
(код и наименование)

Направленность (профиль) Вычислительные машины, комплексы, системы и сети

Форма обучения очная
(очная/очно-заочная/заочная)

Тема: Разработка информационной системы для автоматической рубрикации новостных текстов

Студент А-08-17 _____ Челышев Э.А.
группа подпись фамилия и инициалы

Научный руководитель д.т.н. доцент _____ Оцоков Ш.А.
уч. степень должность подпись фамилия и инициалы

Зав. кафедрой к.т.н. доцент _____ Вишняков С.В.
уч. степень звание подпись фамилия и инициалы

Место выполнения работы каф. ВМСС

СОДЕРЖАНИЕ РАЗДЕЛОВ ЗАДАНИЯ И ИСХОДНЫЕ ДАННЫЕ

Рубрикация (рубрицирование) – распределение некоторой информации по тематическим разделам (рубрикам).

В рамках данной выпускной квалификационной работы необходимо разработать систему автоматической рубрикации русскоязычных новостных текстов с использованием алгоритмов машинного обучения, а также Web-сайт для демонстрации полученных результатов.

Для автоматической рубрикации текстов использовать язык программирования Python, а также стандартные библиотеки машинного обучения (Pandas, NumPy, Scikit-learn и др.). Для обучения моделей машинного обучения использовать набор данных новостного Интернет-ресурса lenta.ru. Необходимо провести обработку данных, затем выделить ряд рубрик, включенных в вышеназванный набор данных («Экономика», «Культура», «Наука» и др.), разделение по которым будет проводиться в дальнейшей работе. Обработку данных проводить в том числе с ипользованием регулярных выражений.

Для рубрикации использовать стандартные классификаторы, включенные в библиотеку Scikit-learn (линейные методы классификации, случайный лес решающих деревьев, байесовский метод). Рассмотреть возможность рубрикации при помощи искусственных нейронных сетей. Провести поиск оптимальных параметров для классификаторов, оценить качество классификации для каждого классификатора по ряду метрик. В дальнейшей работе использовать классификатор, дающий наиболее точные результаты.

Для создания Web-сайта использовать язык программирования Python и фреймворк Django, а для разметки страниц – язык гипертекстовой разметки HTML, для графического оформления страниц использовать каскадную таблицу стилей CSS. Web-сайт должен содержать удобный пользовательский интерфейс и включать в себя отдельные страницы, соответствующие отдельным рубрикам.

Для хранения содержимого сайта и обрабатываемых новостных статей использовать систему управления базами данных MySQL.

На Web-сайте должны содержаться заголовки оригинальных новостных статей и ссылки на них, публикация полных текстов статей не допускается.

ПЕРЕЧЕНЬ ГРАФИЧЕСКОГО МАТЕРИАЛА

Количество листов _____

Количество слайдов в презентации не менее 8

1. Схема алгоритма случайного леса решающих деревьев.
2. Схема алгоритма наивного байесовского классификатора.
3. Пользовательский интерфейс разработанного Web-сайта.
4. Иллюстрация взаимосвязей компонентов разрабатываемой системы.

РЕКОМЕНДУЕМАЯ ЛИТЕРАТУРА

1. Шаграев, А.Г. Модификация, разработка и реализация методов классификации новостных текстов: дисс. … канд. технических наук: 05.13.17. – НИУ «МЭИ», Москва, 2014. – 108 с.
2. Мюллер, А. Введение в машинное обучение с помощью Python / А. Мюллер, С. Гвидо; пер. с англ. – М.: Альфа-книга, 2018. – 480 с. - ISBN: 978-1-449-36941-5.
3. Лутц, М. Изучаем Python. В 2 тт. Т.1 / М. Лутц; пер. с англ. – 5 изд. – СПб.: Диалектика, 2019. – 832 с. – ISBN 978-5-907144-52-1.
4. Рашка, С. Python и машинное обучение / С. Рашка; пер. с англ. А. В. Логунов; под ред. Д. А. Мовчан. — М.: ДМК Пресс, 2017. — 418 с. — ISBN: 978-5-97060-409-0.

Примечания:

1. Задание брошюруется вместе с выпускной работой после титульного листа (страницы задания имеют номера 2, 3).
2. Отзыв руководителя, рецензия(и), отчет о проверке на объем заимствований и согласие студента на размещение работы в открытом доступе вкладываются в конверт (файловую папку) под обложкой работы.

Приложение Б. Листинг программ для подготовки корпуса и построения моделей классификации

Программный модуль prepare.py

```
#!/usr/bin/env python
# coding: utf-8


"""
Программный модуль, содержащий функции для
обработки текстов на естественном языке.

Разработчик: студент группы А-08-17 "НИУ МЭИ"
Челышев Э.А.
e-mail: ChelyshevEA@mpei.ru
Место разработки: каф. ВМСС НИУ "МЭИ"
Дата разработки модуля: 15 мая 2021 г.

Входные и выходные данные описаны для каждой функции в отдельности.
См. поясняющую информацию далее.
Для корректной и полнофункциональной работы необходимо наличие
файла предобученной модели векторизации FastText (файл model.model)
"""

import numpy as np  # импорт пакета для работы с массивами
import re  # импорт пакета для работы с регулярными выражениями
import gensim  # импорт пакета для работы с моделями векторизации
import nltk  # импорт пакета для работы с естественным языком
from nltk import word_tokenize  # импорт функции токенизации
from nltk.corpus import stopwords  # импорт стоп-слов
import pymorphy2  # импорт морфологического анализатора

# создание объекта класса морфологического анализатора
morph = pymorphy2.MorphAnalyzer()
# список стоп-слов русского языка
stop_words = stopwords.words("russian")
# объявление переменной для модели векторизации
w2v_model = None

"""
Данная функция устанавливает путь к внешему файлу модели векторизации
и загружает ее.
На вход функция получает строку - путь к файлу. По умолчанию путь
'model.model'
Каких-либо результатов функция не возвращает.
"""
def load_model(path='model.model'):
    try:
        # загрузка модели векторизации
        global w2v_model
        w2v_model = gensim.models.KeyedVectors.load(path)
    except:  # если не удалось загрузить модель векторизации
        print('Не удалось загрузить модель векторизации. Проверьте
правильность пути')
    pass
```

```
"""
Данная функция осуществляет приведение строки текста к общему регистру
(строчные буквы), а также
производит удаление нерелевантных символов и встречающиеся в тексте URL-
ссылки.
На вход функция получает text - строку текста, подлежащую обработке.
Результатом является обработанная строка
"""
def preprocess_text(text):
    # приведение к общему регистру и удаление буквы "ё"
    text = text.lower().replace("ё", "е")
    # удаление URL-ссылок
    text = re.sub('((www\.[^\s]+)|(https?://[^\s]+))', 'URL',
                  text)
    # удаление небуквенных символов
    text = re.sub('[^a-zA-Za-яА-Я]+', ' ', text)
    # несколько подряд идущих пробелов заменяются на один
    text = re.sub(' +', ' ', text)
    # удаление пробелов с обоих концов строки, если таковые имеются
    preprocessed_text = text.strip()
    return preprocessed_text

"""
Данная функция осуществляет приведение токенов к начальной форме путем их
лемматизации.
На вход функция получает text_list - список строк, каждая из которых
является отдельным токеном, подлежащим лемматизации.
Результатом функции является список токенов, прошедших лемматизацию
"""
def lemmatize(text_list):
    result = list()  # подготовка списка для хранения результата
    for token in text_list:  # цикл по токенам списка
        # получение начальной формы
        modified_token = morph.parse(token)[0].normal_form
        result.append(modified_token)  # добавление результата к списку
    return result

"""
Данная функция осуществляет удаление стоп-слов.
На вход функция получает список строк, являющихся отдельными токенами.
Результатом является список токенов, откуда удалены стоп-слова.
Для корректной работы данной функции токены должны быть приведены
к начальной форме с использованием функции lemmatize()
"""
def delete_stop_words(text_list):
    # подготовка списка для хранения результата
    result = list()
    # цикл по токенам входного списка
    for token in text_list:
        # если токен не является стоп-словом
        if token not in stop_words:
            # он добавляется к результирующему списку
            result.append(token)
    return result
```

```
"""
Данная функция осуществляет построение вектора статьи.
На вход функция получает список токенов.
Результатом выполнения функции является вектор размерности 300,
в формате массива NumPy.
Для корректной работы функции входной список должен быть обработан
с использованием функции prepare_text
"""
def build_article_vector(t):
    # счетчик числа векторизованных токенов
    count = 0
    # подготовка списка для хранения результата
    res = np.zeros(300)
    # цикл по токенам
    for word in t:
        # если токен содержится в модели
        if (word in w2v_model.vocab):
            # прибавляем полученный для токена вектор к результирующему
            res += w2v_model.get_vector(word)
            # увеличиваем счетчик векторизованных токенов
            count += 1
    # определяем среднее по статье
    res /= count
    return res

"""
Данная функция осуществляет полную подготовку текста на естественном языке,
а именно: токенизацию, лемматизацию и удаление стоп-слов.
На вход функция получает строку текста, подлежащую обработке.
Результатом выполнения функции является список токенов, прошедший этапы
подготовки
"""
def prepare_text(text):
    # приведение к одному регистру и удаление нерелевантных символов
    text = preprocess_text(text)
    #  токенизация с использованием пакета nltk
    text_list = word_tokenize(text, language = "russian")
    # лемматизация с использованием функции lemmatize
    text_list = lemmatize(text_list)
    # удаление стоп-слов с использованием функции delete_stop_words
    text_list = delete_stop_words(text_list)
    # векторизация
    vector = build_article_vector(text_list)
    return vector
```

Программный файл corpus.py

```
#!/usr/bin/env python
# coding: utf-8

"""
Данная программа подготавливает корпус для экспериментов по построению
моделей машинного обучения.

Разработчик: студент группы А-08-17 "НИУ МЭИ"
Челышев Э.А.
e-mail: ChelyshevEA@mpei.ru
Место разработки: каф. ВМСС НИУ "МЭИ"
Дата разработки: 16 мая 2021 г.

Входными данными является корпус статей новостного портала lenta.ru (файл
lentanews.csv).
Результатом выполнения данной программы являются подготовленный для
построения моделей машинного обучения
корпус в виде файла dataset.csv.
Для работы необходимо подключить модуль prepare.py
"""

import pandas as pd  # импорт пакета для работы с датафреймами
# импорт пакета для отслеживания времени выполнения обработки датафрейма
from tqdm import tqdm
import prepare  # импорт модуля с функциями подготовки данных

tqdm.pandas()  # отслеживание времени выполнения для датафреймов pandas

# Считываем данные
data_path = 'C:/Users/Эдуард Челышев/Desktop/мл/Dip/diplom/lentanews.csv'
data = pd.read_csv(data_path,
                   sep=',',
                   error_bad_lines=False,
                   usecols=['text', 'topic', 'tags'])

# подсчет количества статей в рубриках
print(data.groupby(['topic']).count())
print(data.groupby(['tags']).count())

# формирование рубрики Политика
data['topic'] = np.where((data.tags == 'Политика'), 'Политика', data.topic)
# проверим наличие рубрики Политика
print(data.groupby(['topic']).count())

# удаление из корпуса статей, принадлежащих некоторым рубрикам
data = data.drop(
    data[(data['topic'] == '69-я параллель') | (data['topic'] ==
'Библиотека')
         | (data['topic'] == 'Крым') | (data['topic'] == 'МедНовости') |
         (data['topic'] == 'Оружие') | (data['topic'] == 'Сочи') |
         (data['topic'] == 'ЧМ-2014') | (data['topic'] == 'Мир') |
```

```
        (data['topic'] == 'Культпросвет ') | (data['topic'] == 'Легпром')
|
        (data['topic'] == 'Россия') | (data['topic'] == 'Бывший СССР') |
        (data['topic'] == 'Из жизни') | (data['topic'] ==
'Ценности')].index)
# объединение рубрик Бизнес и Экономика
data = data.replace({'topic': {'Бизнес': 'Экономика'}})
# изучение оставшихся рубрик
print(data.groupby(['topic']).count())
# удаление строк, содержащих пропуски
data = data.dropna(axis='index', how='any')

# Построение списка из названий имеющихся рубрик
topics = np.sort(data['topic'].unique())
print(topics)
# построение словаря соответствия названий рубрик и их цифровых
эквивалентов
dct = dict()  # объявление пустого словаря
label = 0  # нумерация рубрик начинается с 0
for item in topics:
    dct[item] = label  # добавляем новую рубрику
    label += 1
# замена названий рубрик на цифровые эквиваленты
data = data.replace({'topic': dct})

# рассмотрим структуру имещихся данных
print('Размерность датафрейма: {}'.format(data.shape))
print('Столбцы датафрейма: {}'.format(data.columns))
print('Информация: {}'.format(data.info()))
print('Статистические характеристики:\n')
data.describe()

# указываем модулю prepare адрес к модели векторизации
model_path = 'model/model.model'
prepare.load_model(model_path)

# произведем подготовку с использованием функции
# prepare_text модуля prepare
# обработка датафрейма осуществляется частями
chunk_list = []
for chunk in data:
    # для каждой части датафрейма осуществляем обработку
    chunk['text'] = chunk['text'].progress_apply(prepare.prepare_text)
    chunk_list.append(chunk)
# объединяем обработанные части датафрейма
data = pd.concat(chunk_list)

# создаем копию датафрейма
vectors = data.copy()
# разбиваем список по колонкам
vectors = pd.DataFrame(vectors['text'].to_list(), columns=range(0, 300))
# объединяем со столбцом рубрик
df = pd.concat([vectors, data['topic']], axis=1)
df = df.dropna(axis='index', how='any')  # удаляем пропуски
df.to_csv('dataset.csv', sep=',', header=True,
          index=False)  # сохранем итоговый файл
```

Программный файл build_models.py

```
#!/usr/bin/env python
# coding: utf-8

"""
Данная программа содержит ряд экспериментов по построению моделей машинного обучения.

Разработчик: студент группы А-08-17 "НИУ МЭИ"
Челышев Э.А.
e-mail: ChelyshevEA@mpei.ru
Место разработки: каф. ВМСС НИУ "МЭИ"
Дата разработки: 17 мая 2021 г.

Входными данными является заранее подготовленный набор данных (файл dataset.csv).
Результатом работы являются обученные модели машинного обучения для задачи рубрикации текстов.
"""

# отчет о классификации
from sklearn.metrics import classification_report
# функция разделения на тренировочный и тестовый набор
from sklearn.model_selection import train_test_split
# импорт пакета для работы с датафреймами
import pandas as pd
import numpy as np  # для работы с массивами
import pickle  # для сохранения моделей
# логистическая регрессия
from sklearn.linear_model import LogisticRegression
# решетчатый поиск
from sklearn.model_selection import GridSearchCV
# для отображения матрицы путаницы
import confusion_matrix_pretty_print as cmpp
# гауссовский наивный байесовский классификатор
from sklearn.naive_bayes import GaussianNB
# предобработка данных
from sklearn import preprocessing
# случайный лес решающих деревьев
from sklearn.ensemble import RandomForestClassifier

"""
Данная функция выводит метрики качества модели классификации: отчет о классификации и матрицу путаницы.
На вход функция получает два массива: достоверный массив меток и предсказание модели.
Функция выводит отчет о классификации и отображает матрицу путаницы.
"""
def evaluate_quality(y_test, predicted):
    # отчет о классификации
    print(classification_report(y_test, lr_predicted, digits=5))
    # матрица путаницы
    cmpp.plot_confusion_matrix_from_data(y_test,
```

```
                                        lr_predicted,
                                        columns=range(0, 9),
                                        figsize=[7, 7])
    pass


# чтение данных
path = 'dataset.csv'
df = pd.read_csv(path)
# выделение меток классов в отдельный датафрейм
targets = df.filter(['topic'], axis=1)
df.drop(['topic'], axis='columns', inplace=True)
vectors = df  # содержит только векторы статей
# разделение на тренировочную и тестовую выборки

x_train, x_test, y_train, y_test = train_test_split(vectors,
                                                    targets,
                                                    test_size=0.25,

stratify=targets['topic'],
                                                    random_state=2)
# сжатие массивов меток до одной оси и приведение их к целочисленному типу
y_train = np.ravel(y_train)
y_train = np.array(y_train, dtype='int8')
y_test = np.ravel(y_test)
y_test = np.array(y_test, dtype='int8')

# приведение к типу float32
x_train = x_train.astype('float32')
x_test = x_test.astype('float32')

""" Наивный байесовский классификатор """
# нормализация данных
x_train_norm = preprocessing.normalize(x_train)
x_test_norm = preprocessing.normalize(x_test)

# обучение гауссовского наивного байесовского классификатора
bayes = GaussianNB().fit(x_train_norm, y_train)
bayes_predicted = bayes.predict(x_test_norm)
# оценка качества
evaluate_quality(y_test, bayes_predicted)
# Сохранить модель в файле
pickle.dump(bayes, open('bayes.pkl', 'wb'))

""" Логистическая регрессия """
# создание объекта логистической регрессии
lr = LogisticRegression(random_state=0, max_iter=10000)
# поиск гиперпараметров методом решетчатого поиска
log_params = {'C': [0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1, 5, 10, 50, 100]}
log_grid = GridSearchCV(lr, log_params, cv=3)
log_grid.fit(x_train, y_train)
# вывод оптимальных гиперпараметров
print(log_grid.best_params_)
print(log_grid.cv_results_)

# обучение регрессии при оптимальных параметрах
```

```
lr_best = LogisticRegression(C=50, max_iter=10000, random_state=0)
lr_best.fit(x_train, y_train)

# оценка качества
lr_predicted = lr_best.predict(x_test)
evaluate_quality(y_test, lr_predicted)
# Сохранить модель в файле
pickle.dump(lr_best, open('log_reg.pkl', 'wb'))

""" Случайный лес решающих деревьев """
# поиск оптимальных параметров
rfc = RandomForestClassifier(random_state=42)
forest_params = {
    'n_estimators': [50, 100, 150, 200],
    'max_features': [5, 10, 15, 20, 25, 30, 35, 40]
}
forest_grid = GridSearchCV(rfc, forest_params, cv=3)
forest_grid.fit(x_train, y_train)
print(forest_grid.best_params_)
print(forest_grid.cv_results_)

# обучение для оптимальных параметров
rfc = RandomForestClassifier(n_estimators=150, max_features=30)
rfc.fit(x_train, y_train)
rfc_predicted = rfc.predict(x_test)

# оценка качества
evaluate_quality(y_test, rfc_predicted)
# Сохранить модель в файле
pickle.dump(rfc, open('forest.pkl', 'wb'))
```

Программный файл network.py

```
"""
Данная программа содержит эксперименты по построению классификатора на
основе ИНС
Разработчик: Челышев Э.А. каф. ВМСС НИУ "МЭИ" 25 мая 2021 г.
"""
# для разделения на обучающую и тестовую выборки
from sklearn.model_selection import train_test_split
import seaborn as sns
# слои для нейросети
from tensorflow.keras.layers import Dense, Dropout, BatchNormalization
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.models import Sequential  # полносвязная нейронная
сеть
from tensorflow.keras import utils  # для работы с категориальными данными
import numpy as np  # для работы с данными
import pandas as pd  # для работы с таблицами
import matplotlib.pyplot as plt  # для работы с графиками
%matplotlib inline

df = pd.read_csv('/content/drive/MyDrive/diplom/dataset.csv')  # чтение
файла
```

```
# балансировка классов
maxim = 6500  # максимальное число объектов в классе
for i in df['topic'].unique():  # по каждому классу
    if (df[df['topic'] == i].shape[0] > maxim):
        # удаление лишних объектов
        df = df.drop(df[df['topic'] == i].index[maxim:])
targets = df.filter(['topic'], axis=1)  # выделение меток классов
# выделение признаков
df.drop(['topic'], axis='columns', inplace=True)
vectors = df
# выделение обучающей и тестовой выборок
x_train, x_test, y_train, y_test = train_test_split(vectors,
                                                    targets,
                                                    test_size=0.25,

stratify=targets['topic'],
                                                    random_state=1)
# преобразование типов
y_train = np.ravel(y_train)
y_train = np.array(y_train, dtype='int32')
y_test = np.ravel(y_test)
y_test = np.array(y_test, dtype='int32')
x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
# one-hot encoding меток классов
y_train = utils.to_categorical(y_train, 9)
n_classes = 9
# Создание нейросети
# полносвязная ИНС
net2 = Sequential()
# входной полносвязный слой
net2.add(Dense(300, input_dim=300, activation='relu'))
# слой регуляризации Dropout
net2.add(Dropout(0.25))
# слой пакетной нормализации
net2.add(BatchNormalization())
# выходной полносвязный слой
net2.add(Dense(n_classes, activation='softmax'))
acc = []
val_acc = []
# Алгоритм решетчатого поиска
for param in [0.0001, 0.0005, 0.001, 0.005, 0.01, 0.05]:
    net2.compile(optimizer=Adam(param),
                 loss='categorical_crossentropy',
                 metrics=['accuracy'])
    history = net2.fit(x_train,
                       y_train,
                       epochs=50,
                       batch_size=128,
                       validation_data=(x_test, y_test))
    acc = np.append(acc, history.history['accuracy'])
    val_acc = np.append(val_acc, history.history['val_accuracy'])
acc = acc.reshape(6, 50)
val_acc = val_acc.reshape(6, 50)
print(val_acc)
```

Приложение В. Листинг веб-сайта

Программный файл rubricate.py

```
"""
Данная программа предназначена для работы в фоновом режиме и осуществляет
автоматическую рубрикацию текстов.

Разработчик: студент группы А-08-17 "НИУ МЭИ"
Челышев Э.А.
e-mail: ChelyshevEA@mpei.ru
Место разработки: каф. ВМСС НИУ "МЭИ"
Дата разработки модуля: 10 июня 2021 г.

Программа считывает статьи из базы данных, которые еще не были отнесены ни
к одной рубрике и обновляет в БД прочтенные записи,
устанавливая значение для рубрики по итогам классификации с использованием
предобученной модели
"""

import prepare # импорт модуля для подготовки данных
import pickle # импорт библиотеки для работы с готовыми моделями машинного
обучения
import numpy as np # импорт библиотеки для работы с массивами
import time # импорт библиотеки для работы со временем
import signal # импорт библиотеки для работы с сигналом прерывания
# импорт слоев для работы с ИНС
from tensorflow.keras.layers import Dense, Dropout, BatchNormalization
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.models import Sequential  # полносвязная нейронная
сеть
from tensorflow.keras.models import model_from_json
# импорт методов библиотеки для работы с БД
from sqlalchemy import create_engine
from sqlalchemy import update
from sqlalchemy.orm import Session

# переопределение обработчика прерывания KeyboardInterrupt
signal.signal(signal.SIGINT, handler)

# параметры подключения к БД
hostname = "localhost"
dbname = "newssite"
uname = "root"
pwd = "passw0rd"
# глобальный флаг окончания работы
terminate = False

# обработчик прерывания KeyboardInterrupt
def handler(signum, frame):
    global terminate
    print("Termination requested by user")
    terminate = True

# подключение к БД
```

```
engine = create_engine("mysql+pymysql://{user}:{pw}@{host}/{db}".format(
    host=hostname, db=dbname, user=uname, pw=pwd))
session = Session(bind=engine)

# загрузка модели векторизации
prepare.load_model('model/model.model')
# загрузка нейронной сети из формата JSON
model=Sequential()
model_file=open('network.json', 'r')
model=model_from_json(model_file.read())
model_file.close()
model.load_weights('network.h5')
model.compile(optimizer=Adam(0.0001),
                loss='categorical_crossentropy',
                metrics=['accuracy'])

# цикл рубрикации
while not terminate:
    res = engine.execute('select * from main_articles where topic=-1') #
выборка нерубрицированных статей из БД
    for row in res:
        if (terminate):
            break # выход из цикла, если установлен глобальный флаг
окончания работы
        text = prepare.prepare_text(row['text']) # подготовка данных
        topic = model.predict(text.reshape(1, -1)) # предсказание модели
        query = 'update main_articles set topic={value} where
id={id}'.format(
            value=topic[0], id=row['id']) # запрос на обновление БД
        engine.execute(query) # выполнение запроса на обновление БД
    for i in range(0,100):
        if (terminate):
            break # выход из цикла, если установлен глобальный флаг
окончания работы
        time.sleep(1)
```

Листинг серверной части веб-сайта

Файл urls.py

```
from django.conf import settings # настройки settings
from django.urls import path, include # для работы с URL-адресами
from django.contrib import admin # панель администратора
from django.conf.urls.static import static

# задаются URL-адреса
urlpatterns = [
       # заглавная страница
       path('', include('main.urls')),
       # панель администратора
```

```
       path('admin/', admin.site.urls),
] +
static(settings.STATIC_URL, document_root=settings.STATIC_ROOT)
```

Файл main/urls.py

```
# импорт библиотек
from django.urls import path
from . import views
# URL-адреса
urlpatterns = [
    path('', views.index, name='main'),
    path('culture', views.culture, name='culture'),
    path('economics', views.economics, name='economics'),
    path('home', views.home, name='home'),
    path('internet', views.internet, name='internet'),
    path('military', views.military, name='military'),
    path('politics', views.politics, name='politics'),
    path('science', views.science, name='science'),
    path('sport', views.sport, name='sport'),
    path('trips', views.trips, name='trips'),
    path('about', views.about, name='about'),

]
```

Файл models.py

```
from django.db import models

# описание ORM-модели


class Articles(models.Model):
    url = models.URLField('Ссылка')  # URL-адрес стаьи
    title = models.CharField('Название', max_length=100)  # заголовок
статьи
    text = models.TextField('Текст')  # текст статьи
    topic = models.IntegerField('Рубрика', default=-1)  # рубрика
    date = models.DateField('Дата')  # дата публикации

    # конструктор класса

    def __str__(self):
        return self.title

    # задаем имя таблиц в панели администратора

    class Meta:
        verbose_name = 'Новость'# единственное число
        verbose_name_plural = 'Новости'# множественное число
```

Листинг клиентской части веб-сайта

Файл news_layout.html

```
<!-- Шаблонный файл для HTML-страниц -->
{% load static %}
<!doctype html>
<html lang="ru">
<head>
  <meta charset="utf-8" />
  <title>{% block title %}
  <!-- динамически изменяемый заголовок -->
  {% endblock %}</title>
  <!-- подключение файлов стилей -->
  <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@4.6.0/dist/css/bootstrap.min.c
ss" />
  <link rel="stylesheet" href="{% static 'main/css/main_style.css' %}" />
  <link rel="stylesheet"
href="https://use.fontawesome.com/releases/v5.8.2/css/all.css" />
</head>
<body>
    <!-- боковая панель -->
    <aside>
        <img src="{% static 'main/img/logo.png' %}" alt="Лого">
        <a href="{% url 'main' %}"><span class="logo">Новости</span></a>
        <!-- меню -->
        <ul>
            <br>
            <a href="{% url 'home' %}"><li><i class="fas fa-home"></i>
Дом</li></a>
            <a href="{% url 'internet' %}"><li><i class="fab fa-internet-
explorer"></i> Интернет и СМИ</li></a>
            <a href="{% url 'culture' %}"><li><i class="fas fa-
landmark"></i> Культура</li></a>
            <a href="{% url 'science' %}"><li><i class="fas fa-
microscope"></i> Наука и техника</li></a>
            <a href="{% url 'politics' %}"><li><i class="fas fa-vote-
yea"></i> Политика</li></a>
            <a href="{% url 'trips' %}"><li><i class="fas fa-suitcase-
rolling"></i> Путешествия</li></a>
            <a href="{% url 'military' %}"><li><i class="fas fa-fighter-
jet"></i> Силовые структуры</li></a>
            <a href="{% url 'sport' %}"><li><i class="fas fa-volleyball-
ball"></i> Спорт</li></a>
            <a href="{% url 'economics' %}"><li><i class="fas fa-piggy-
bank"></i> Экономика и бизнес</li></a>
            <br>
            <a href="{% url 'about' %}"><li><i class="fas fa-info-
circle"></i> О проекте</li></a>
        </ul>
    </aside>
    <!-- основная часть страницы -->
    <main>
        <div class="main_page">
```

```
            <!-- динамически изменяемый заголовок страницы -->
            <h3>{% block head %}{% endblock %}</h3>
            <br>
            <!-- вывод новостей соответствующей рубрики -->
            {% for el in news %}
                <a href={{el.url}}>
                    <div class="alert alert-warning">
                        <p>{{el.date}}</p>
                        <h4>{{el.title}}</h4>
                        <p>{{el.text|truncatechars:150}}</p>
                    </div>
                </a>
            {% endfor %}
        </div>
    </main>
</body>
</html>
```

Файл main_style.css

```
/* Файл стилей веб-сайта */
html {
     height: 100%;
}
/* стиль для тэга body */
body {
     background: #2c2c2c;
     height: 100%;
}
aside {
     float: left;
     background: #181818;
     width: 23%;
     padding: 2.5%;
     min-height: 100%;
     color: #fff;
     border-right: 5px solid #4d4d4d;
     position: fixed;
     bottom: 0;
}
/* боковая панель */
aside img {
     width: 80px;
     float: left;
}
/* логотип */
aside .logo {
     font-size: 35px;
     margin-left: 10px;
     font-weight: bold;
     position: relative;
     top: 15px;
     color: #fff;
}
/* выделение пунктов меню при наведении курсора */
```

```
aside a:hover {
      text-decoration: none;
      color: #eb5959;
      transform: scale(1.05);
}
/* заголовок */
aside h3 {
      margin-top: 50px;
      font-size: 30px;
}
/* меню */
aside ul {
      list-style: none;
}
aside ul li {
      display: block;
      margin-top: 20px;
      color: #fff;
      transition: transform 0.6s ease;
}
aside ul li:hover, aside ul a:hover {
      text-decoration: none;
      color: #eb5959;
      transform: scale(1.05);
}
/* основная часть страницы */
main .main_page {
      margin-top: 50px;
      padding-left: 25%;
      text-align: center;
      width: 95%;
      float: left;
      color: #fff;
}
main .main_page h1 {
      font-size: 50px;
}

main .main_page p {
      margin: 0px auto;
      max-width: 100%;
      }
main .index {
      margin-top: 50px;
      text-align: center;
      float: left;
      padding-left: 25%;
      width: 90%;
      color: #fff;
}
main .index p {
      margin: 20px auto;
      max-width: 400px;
}
```