



Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К КУРСОВОЙ РАБОТЕ
НА ТЕМУ:

«Библиотечная система бронирования»

Студент ИУ7-23М
(Группа)

(Подпись, дата)

Н. Р. Трошкин

(И.О. Фамилия)

Руководитель курсовой работы

(Подпись, дата)

А. А. Ступников

(И.О. Фамилия)

РЕФЕРАТ

Расчетно-пояснительная записка 28 с., 8 рис., 1 табл., 17 ист.

Курсовая работа представляет собой разработку модели, позволяющей по тексту статьи на английском языке подобрать к ней один или несколько тегов. Проводится обзор и сравнение методов векторизации текста и моделей многозначной классификации для решения поставленной задачи. Разрабатывается программное обеспечение, реализующее методы векторизации TF-IDF и Word2Vec, а также два разных варианта ансамбля бинарных классификаторов. Выбираются функционалы качества и на основе разработанного ПО сравниваются несколько моделей.

КЛЮЧЕВЫЕ СЛОВА

Машинное обучение, обработка текстов, многозначная классификация, векторизация, подбор тегов, Word2Vec, TF-IDF, ансамбль классификаторов.

СОДЕРЖАНИЕ

РЕФЕРАТ	3
ВВЕДЕНИЕ	6
1 Аналитический раздел	7
1.1 Постановка задачи	7
1.2 Описание системы	7
1.3 Общие требования к системе	7
1.4 Требования к функциональным характеристикам	8
1.5 Функциональные требования к системе с точки зрения пользо- вателя	8
1.6 Входные данные	9
1.7 Выходные параметры	11
1.8 Топология Системы	12
1.9 Требования к программной реализации	14
1.10 Функциональные требования к подсистемам	15
2 Конструкторский раздел	19
2.1 Описание датасета	19
2.1.1 Подготовка данных	19
2.2 Описание алгоритмов	20
3 Технологический раздел	23
3.1 Выбор языка и среды программирования	23
3.2 Код программы	24
3.2.1 Обработка текста	24
3.2.2 Работа с моделями	25
4 Исследовательский раздел	29
4.1 Исследование моделей	29
4.2 Демонстрация работы программы	33

ЗАКЛЮЧЕНИЕ	34
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	35

ВВЕДЕНИЕ

Фитнес-индустрия в России продолжает стремительно развиваться, фитнес перестал быть просто модным хобби и стал неотъемлемой частью здорового образа жизни. По данным исследования, подготовленного аналитической компанией FitnessData, занимающейся исследованием рынков фитнес-услуг в России и за рубежом, объем рынка фитнес-услуг в РФ за 2023 год увеличился на 18% и составил не менее 214 млрд рублей. Этот рост связан как с повышением цен на данный вид услуг (на 11-12%), так и с повышением численности занимающихся (с 6 млн человек в 2022 г. до 6.5 млн человек в 2023 г.).

В Москве насчитывается более тысячи фитнес-клубов, и потенциальным клиентам требуется много времени, чтобы подобрать клуб, который бы лучше всего подходил под их запросы. По этой причине есть необходимость в агрегаторе, содержащем информацию о клубах и упрощающем поиск нужного для пользователя.

Целью данной работы является разработка распределенной системы бронирования книг в библиотеках. Для достижения поставленной цели необходимо решить следующие задачи:

- сформулировать основные требования к системе;
- описать требования к подсистемам;
- описать архитектуру системы, а так же сценарии её функционирования;
- выбрать и обосновать инструменты для разработки программного обеспечения;
- реализовать программный продукт.

1 Аналитический раздел

1.1 Постановка задачи

Разрабатываемая система должна предоставлять пользователям информацию о наличии книг в библиотеках, возможность найти интересующую книгу и взять ее в библиотеке. Если у пользователя на руках есть уже N книг, то он не может взять новую, пока не сдал старые. Если пользователь возвращает книги в хорошем состоянии и сдает их в срок, то максимальное возможное количество книг у него на руках увеличивается.

1.2 Описание системы

Разрабатываемое программное обеспечение должно представлять собой распределённую систему для поиска и бронирования книг в библиотеках.

Авторизованные пользователи могут просмотреть список библиотек в городе, список книг в выбранной библиотеке, информацию по всем книгам, которые были взяты в прокат данным пользователем, свой рейтинг, а также забронировать и вернуть книгу в библиотеке. Неавторизованные пользователи могут только зарегистрироваться или войти в систему, если учетная запись уже существует. Для регистрации необходимо указать следующую информацию о себе: фамилия, имя, отчество, дата рождения, номер телефона, электронная почта.

1.3 Общие требования к системе

Требования к системе следующие:

1. Разрабатываемое ПО должно поддерживать функционирование системы в режиме 24/7/365 (24 часа, 7 дней в неделю, 365 дней в году) со среднегодовым временем доступности не менее 99.9%. Допустимое время, в течении которого система недоступна, за год должна составлять $24 \cdot 365 \cdot 0.001 = 8.76$ ч.
2. Время восстановления системы после сбоя не должно превышать 15 минут.
3. Каждый узел должен восстанавливаться после сбоя автоматически.

4. Система должна поддерживать возможность «горячего» переконфигурирования. Необходимо предусмотреть поддержку добавления нового узла во время работы системы без рестарта.
5. Обеспечить безопасность работоспособности за счёт отказоустойчивости узлов.

1.4 Требования к функциональным характеристикам

1. По результатам работы модуля сбора статистики среднее время отклика системы на запросы пользователя на получение информации не должна превышать 3 секунд.
2. По результатам работы модуля сбора статистики медиана времени отклика системы на запросы, добавляющие или изменяющие информацию на портале не должна превышать 5 секунд.
3. Медиана времени отклика системы на действия пользователя должна быть менее 0.8 секунд при условии работы на рекомендованной аппаратной конфигурации, задержках между взаимодействующими сервисами менее 0.2 секунды и одновременном числе работающих пользователей менее 100 на каждый сервер, обслуживающий внешний интерфейс.

1.5 Функциональные требования к системе с точки зрения пользователя

Система должна обеспечивать реализацию следующих функций.

1. Система должна обеспечивать регистрацию и авторизацию пользователей с валидацией вводимых данных как через интерфейс приложения, так и через социальные сети.
2. Система должна обеспечивать аутентификацию пользователей.
3. В системе должно быть обеспечено разделение всех пользователей на три роли:
 - Пользователь (неавторизованный пользователь);
 - Клиент (авторизованный пользователь);

- Администратор.

4. Предоставление возможностей **Пользователю, Клиенту, Администратору** представленных в таблице 1.

Таблица 1: Функции пользователей

Пользователь	<ol style="list-style-type: none"> 1. регистрация в системе; 2. авторизация в системе;
Клиент	<ol style="list-style-type: none"> 1. получение и изменение информации текущего аккаунта; 2. просмотр списка доступных библиотек; 3. просмотр списка книг, доступных в конкретной библиотеке; 4. просмотр всех своих бронирований; 5. просмотр информации о своем рейтинге; 6. бронирование конкретной книги в конкретной библиотеке; 7. возврат забронированной данным клиентом книги в библиотеку;
Администратор	<ol style="list-style-type: none"> 1. просмотр списка всех клиентов, зарегистрированных в системе; 2. создание новых учетных записей; 3. получение отчетов о выполненных действиях от сервиса статистики; 4. просмотр и редактирование списка доступных библиотек; 5. просмотр и редактирование списка доступных книг в конкретной библиотеке; 6. бронирование книги в библиотеке на имя любого клиента; 7. возврат книги в библиотеку от имени любого клиента; 8. изменение рейтинга любого клиента; 9. просмотр списка бронирований любого клиента;

1.6 Входные данные

Входные параметры системы представлены в таблице 2.

Таблица 2: Входные данные

Сущность	Входные данные
Клиент / Администратор	<ol style="list-style-type: none"> 1. <i>фамилия, имя и отчество</i> не более 256 символов каждое поле; 2. <i>дата рождения</i> в формате дд.мм.гггг; 3. <i>логин</i> не более 80 символов; 4. <i>пароль</i> не менее 8 символов и не более 128, как минимум одна заглавная и одна строчная буква, только латинские буквы, без пробелов, как минимум одна цифра; 5. <i>номер телефона</i>; 6. <i>электронная почта</i>; 7. <i>роль пользователя</i> — клиент или администратор;
Рейтинг	<ol style="list-style-type: none"> 1. <i>идентификатор</i> пользователя; 2. <i>логин</i> пользователя не более 80 символов; 3. <i>рейтинг</i> — целое число от 0 до 100;
Библиотека	<ol style="list-style-type: none"> 1. <i>идентификатор</i> библиотеки; 2. <i>UUID</i> библиотеки; 3. <i>название</i> библиотеки не более 80 символов; 4. <i>город</i>, в котором находится библиотека, не более 255 символов; 5. <i>адрес</i> библиотеки не более 255 символов;
Книга	<ol style="list-style-type: none"> 1. <i>идентификатор</i> книги; 2. <i>UUID</i> книги; 3. <i>название</i> книги не более 255 символов; 4. <i>автор</i> книги не более 255 символов; 5. <i>жанр</i> книги не более 255 символов; 6. <i>состояние</i> книги, одно из трех: EXCELLENT, GOOD, BAD;

Продолжение на следующей странице

Сущность	Входные данные
Связь книги и библиотеки	1. <i>идентификатор</i> книги; 2. <i>идентификатор</i> библиотеки; 3. <i>количество</i> экземпляров книги в библиотеке;
Бронирование	1. <i>идентификатор</i> бронирования; 2. <i>UUID</i> бронирования; 3. <i>логин</i> пользователя, совершившего бронирование, не более 80 символов; 4. <i>UUID</i> забронированной книги; 5. <i>UUID</i> библиотеки, в которой совершено бронирование; 6. <i>статус</i> бронирования, одно из трех: RENTED, RETURNED, EXPIRED; 7. <i>дата и время</i> совершения бронирования; 8. <i>дата и время</i> , когда истекает бронирование.

1.7 Выходные параметры

Выходными параметрами системы являются web-страницы. В зависимости от запроса и текущей роли пользователя они содержат следующую информацию (таблица 3)

Таблица 3: Выходные параметры

Клиент	1. список доступных библиотек в городе, в списке указывается: <ul style="list-style-type: none"> ● <i>название</i> библиотеки; ● <i>полный адрес</i> библиотеки; ● <i>город</i>;
	2. список книг в выбранной библиотеке, в котором указывается: <ul style="list-style-type: none"> ● <i>название</i> книги; ● <i>автор</i> книги; ● <i>жанр</i> книги; ● <i>состояние</i> книги; ● <i>количество экземпляров</i> книги в выбранной библиотеке.

Администратор	3. список взятых пользователем в прокат книг, в котором указывается: <ul style="list-style-type: none"> • <i>статус</i> бронирования; • <i>дата начала</i> бронирования; • <i>дата окончания</i> бронирования; • <i>название</i> книги; • <i>автор</i> книги; • <i>жанр</i> книги; • <i>название</i> библиотеки; • <i>адрес</i> библиотеки; • <i>город</i> библиотеки;
	4. информация о рейтинге пользователя: <ul style="list-style-type: none"> • <i>рейтинг</i> пользователя.
	Страницы, аналогичные страницам, доступным клиентам, а также: 5. список клиентов, зарегистрированных в системе: <ul style="list-style-type: none"> • <i>имя</i> клиента; • <i>логин</i> клиента; • <i>дата рождения</i> клиента; • <i>почта</i> клиента; • <i>телефон</i> клиента.
	6. отчет о пришедших данных из сервиса статистики.

1.8 Топология Системы

На рисунке 1 изображён один из возможных вариантов топологии разрабатываемой распределенной Системы.

Система будет состоять из фронтенда и 6 подсистем:

- сервис-координатор;
- сервис авторизации;
- сервис объектов библиотек;
- сервис бронирования;
- сервис рейтингов;

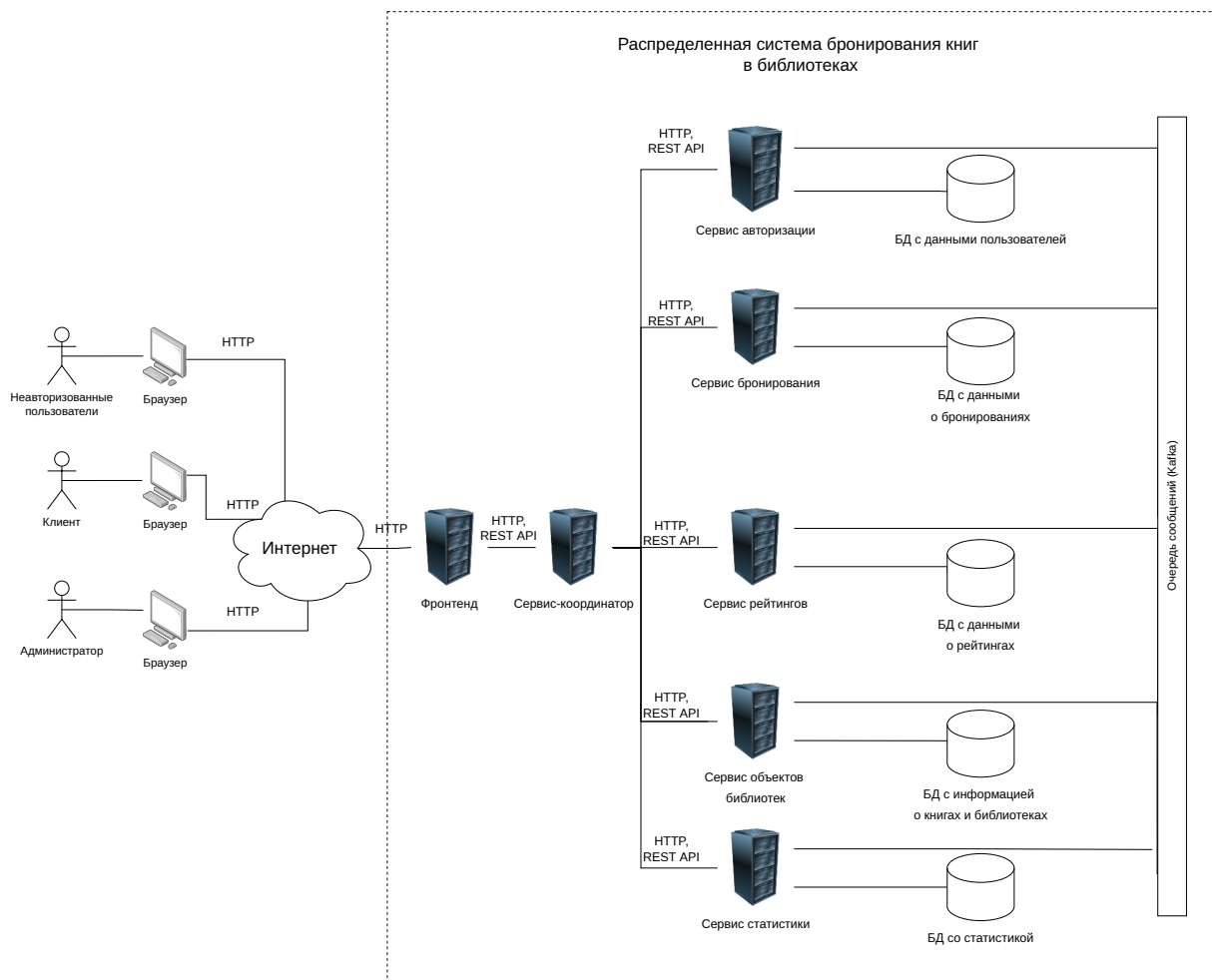


Рисунок 1: Топология системы

— сервис статистики.

Фронтенд принимает запросы от пользователей по протоколу HTTP и анализирует их. На основе проведенного анализа выполняет запросы к микросервисам бекенда, агрегирует ответы и отправляет их пользователю.

Сервис-координатор – единая точка входа и межсервисной коммуникации.

Сервис авторизации отвечает за:

- возможность регистрации нового клиента;
- аутентификацию пользователя (клиента и администратора);
- авторизацию пользователя;
- изменение персональных данных пользователя;

— выход из сессии.

Сервис объектов библиотек отвечает за хранение информации обо всех библиотеках и книгах в них.

Сервис бронирования реализует следующие функции:

- хранение и передача информации о книгах, взятых в прокат пользователем;
- бронирование новой книги в библиотеке;
- возврат книги в библиотеку.

Сервис рейтингов реализует хранение рейтингов пользователей системы.

Сервис статистики получает через Kafka информацию о выполненных действиях и по этим данным строит отчет для администратора.

1.9 Требования к программной реализации

1. Система состоит из микросервисов. Каждый микросервис отвечает за свою область логики работы приложения. Сервисы должны быть запущены изолированно друг от друга.
2. При необходимости, каждый сервис имеет своё собственное хранилище, запросы между базами запрещены.
3. Необходимо реализовать один web-интерфейс для фронтенда в виде SPA или мобильного клиента с обязательным использованием CSS. Интерфейс должен быть доступен через тонкий клиент (браузер).
4. Взаимодействие между сервисами осуществляется посредством HTTP-запросов и/или очереди сообщений по технологии REST.
5. Требуется выделить Gateway Service как единую точку входа и межсервисной коммуникации. В системе не должно осуществляться горизонтальных запросов.
6. На Gateway Service для всех операций чтения реализовать паттерн Circuit Breaker. Накапливать статистику в памяти, и если система не

ответила N раз, то в $N + 1$ раз вместо запроса сразу отдавать fallback. Через небольшой timeout выполнить запрос к реальной системе, чтобы проверить ее состояние.

7. При недоступности подсистем должна осуществляться деградация функциональности (если подсистема некритичная) или выдача пользователю сообщения об ошибке и полный откат операции (либо постановка запроса в очередь для повторного выполнения и возврат пользователю сообщения об успехе).
8. Все методы `/api/**` (кроме `/api/v1/authorize` и `/api/v1/callback`) на всех сервисах должны быть закрыты token-based авторизацией.
9. Если авторизация некорректная (отсутствие токена, ошибка валидации JWT токена, закончилось время жизни токена (поле `exp` в `payload`)), то отдавать 401 ошибку.
10. Развертывание приложения выполнить с использованием Managed Kubernetes Cluster и настроить Ingress Controller (для публикации сервисов наружу можно использовать только Ingress). Для деплоя использовать helm charts, они должны быть универсальными для всех сервисов и отличаться лишь набором параметров запуска. Сервисы должны быть завернуты в docker.

1.10 Функциональные требования к подсистемам

Подсистемы: фронтенд, бекенд-координатор, бекенд авторизации, бекенд объектов библиотеки, бекенд бронирования, бекенд рейтингов, бекенд статистики.

Фронтенд – серверное приложение, предоставляет пользовательский интерфейс и внешний API системы, при разработке которого нужно учитывать следующее:

- должен принимать запросы по протоколу HTTP и формировать ответы пользователям в формате HTML;

- в зависимости от типа запроса должен отправлять последовательные запросы в соответствующие микросервисы;
- запросы к микросервисам необходимо осуществлять по протоколу HTTP;
- данные необходимо передавать в формате JSON.

Сервис-координатор – серверное приложение, через которое проходит весь поток запросов и ответов, должен соответствовать следующим требованиям разработки:

- принимать и возвращать данные в формате JSON по протоколу HTTP;
- для всех операций чтения реализовать паттерн Circuit Breaker — накапливать статистику в памяти, и если система не ответила N раз, то в N + 1 раз вместо запроса сразу отдавать fallback, а через небольшой timeout выполнить запрос к реальной системе, чтобы проверить ее состояние.
- выполнять валидацию JWT токена с помощью JWKS;
- реализовывать запросы, связанные с получением информации о книгах, библиотеках, бронированиях, пользователях и рейтинге пользователей, а также запросы статистики от администратора;
- реализовывать бизнес-логику бронирования книги в библиотеке, в которую включается:
 - получение информации от пользователя о выбранной библиотеке, книге и дате окончания бронирования;
 - проверка количества книг у пользователя на руках (в состоянии RENTED);
 - проверка рейтинга пользователя (рейтинг определяет максимальное возможное количество книг у пользователя на руках);
 - если у пользователя меньше книг на руках, чем его рейтинг, запрос в сервис бронирований о создании нового бронирования;

- запрос на изменение числа доступных экземпляров взятой книги в библиотеке, где произошло бронирование;
 - возврат пользователю информации об успешном или неуспешном бронировании.
- реализовывать бизнес-логику возврата книги в библиотеку, в которую включается:
- получение информации от пользователя о бронировании, которое он собирается закрыть, состоянии книги и фактической дате возврата;
 - запрос на изменение статуса бронирования (возвращено или просрочено);
 - запрос на изменение числа доступных экземпляров возвращенной книги в библиотеке, куда произошёл возврат;
 - если у книги ухудшилось состояние или возврат просрочен, запрос на уменьшение рейтинга пользователя на 10 за каждое условие;
 - в противном случае запрос на увеличение рейтинга пользователя на 1.

Сервис авторизации должен реализовывать следующие функциональные возможности:

- принимать и возвращать данные в формате JSON по протоколу HTTP;
- возможность регистрации нового клиента и обновление данных уже существующего;
- выполнять функцию Identity Provider с использованием протокола OpenID Connect (scope: openid, profile, email);
- для получения токена на Identity Provider должен быть реализован Authorization Flow (UI рассматривается как 3rd party application и авторизация пользователя так же выполняется через Authorization Flow).

Сервис объектов библиотеки реализует следующие функции:

- получение и отправка данных в формате JSON по протоколу HTTP;
- получение таких данных, как:
 - список доступных библиотек в заданном городе;
 - список книг в конкретной библиотеке;
- валидация JWT токена с помощью JWKs.

Сервис рейтингов реализует функции:

- получение и отправка данных в формате JSON по протоколу HTTP;
- предоставления информации об рейтинге по идентификатору пользователя;
- валидация JWT токена с помощью JWKs.

Сервис бронирования должен реализовывать представленные такие функциональные возможности, как:

- получение и отправка ответов на запросы в формате JSON по протоколу HTTP;
- валидация JWT токена с помощью JWKs;
- предоставление информации о списке книг, взятых в прокат пользователем;
- создание и обновление записей о бронировании в базе данных при взятии книги из библиотеки или возврате в библиотеку.

Сервис статистики получает через Kafka информацию о выполненных действиях и по этим данным строит отчет для администратора и валидирует его JWT-токен.

2 Конструкторский раздел

2.1 Описание датасета

Для обучения и тестирования модели выбран датасет, содержащий 192368 статей на английском языке с платформы Medium [7].

Данные, содержащиеся в датасете, описаны в таблице 4.

Таблица 4: Лучшие параметры для случая срочных и важных задач

Название поля	Тип данных	Описание
title	Строка	Заголовок статьи
text	Строка	Текст статьи
url	Строка	URL статьи
authors	Список строк	Авторы статьи
timestamp	Строка	Дата публикации
tags	Список строк	Список тегов, связанных со статьями

2.1.1 Подготовка данных

В первую очередь, в датасете были убраны все поля, кроме текста статьи и тегов, так как теги требуется назначать, основываясь исключительно на тексте статьи.

Помимо этого, в изначальном датасете было более 40 тысяч уникальных тегов, так как у Medium нет стандартного набора тегов и к каждой статье может быть придуман любой тег. При этом многие теги встречаются в очень малом числе статей и могли бы не попасть в тренировочную выборку, а если бы попали, то такого объема недостаточно, чтобы модель верно предсказывала соответствующий тег. Более того, для каждого тега пришлось бы обучать классификатор, что заняло бы слишком много времени, особенно в случае их цепочки. Поэтому было принято решение отфильтровать все теги, встречающиеся в статьях менее 1000 раз, оставив тем самым наиболее популярные. Статьи, у которых не было оставшихся тегов, были удалены. Таким образом, в датасете осталось 127316 статей с 98 различными тегами.

Также из текстов всех статей убраны так называемые «стоп-слова» — слова, которые очень распространены, однако не влияют на смысл текстов (артикли, предлоги, союзы и т. д.).

Входными данными при обучении являются только векторизованные тексты без тегов. Выходными данными для каждого текста является массив из 98 значений «истина» или «ложь», каждое из которых отвечает за то, присваивается некоторый тег статье или нет. На основе поля tags датасета для обучения создана матрица тегов, в которой хранятся такие строки для каждого текста.

2.2 Описание алгоритмов

Разрабатываемое программное обеспечение может работать в двух режимах:

1. Режим обучения. Из файлов формата csv загружается предобработанный набор текстов на естественном языке и матрица тегов. На основе них выполняется обучение модели с заданными алгоритмами векторизации, классификации и долей тестовых данных. Обученная модель сохраняется в файл.
2. Пользовательский режим. Из заданного файла загружается готовая модель, далее из файла или из командной строки вводится текст, и на основе модели программа выводит для него подходящие теги.

Общая структура ПО изображена на диаграмме в нотации IDEF0 [8] на рисунке 2.

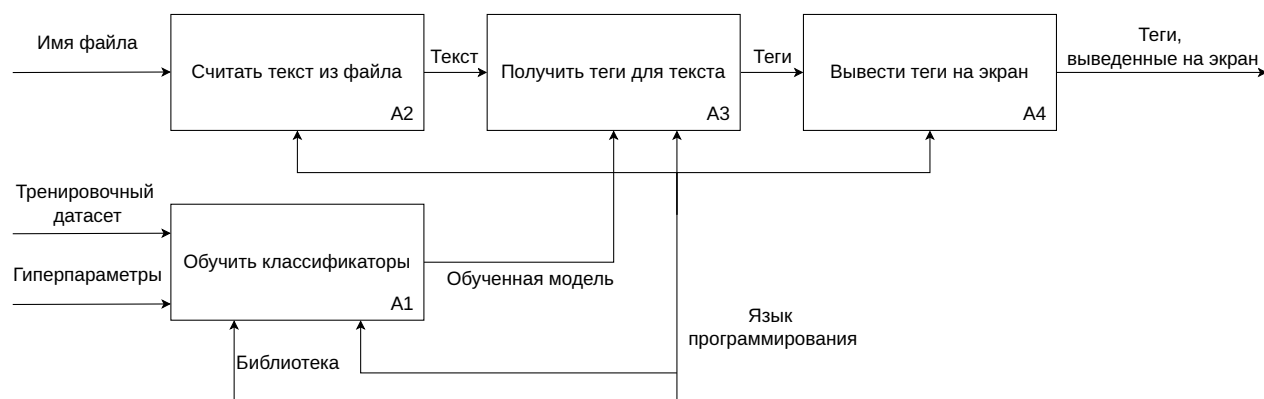


Рисунок 2: IDEF0-диаграмма разрабатываемого ПО

Алгоритм работы программы в пользовательском режиме приводится на рисунке 3. Пользователю требуется ввести путь к файлу, в котором лежит текст, либо ввести его с командной строки (без символов перевода строки),

а также указать путь к файлу, в котором сохранена модель, на вход которой нужно подать текст. Если пользователь не указывает путь к файлу, то используется путь по умолчанию. Помимо этого, необходимо ввести метод векторизации *Vec*, которым требуется предобработать текст (в зависимости от того, какую модель пользователь выбрал).

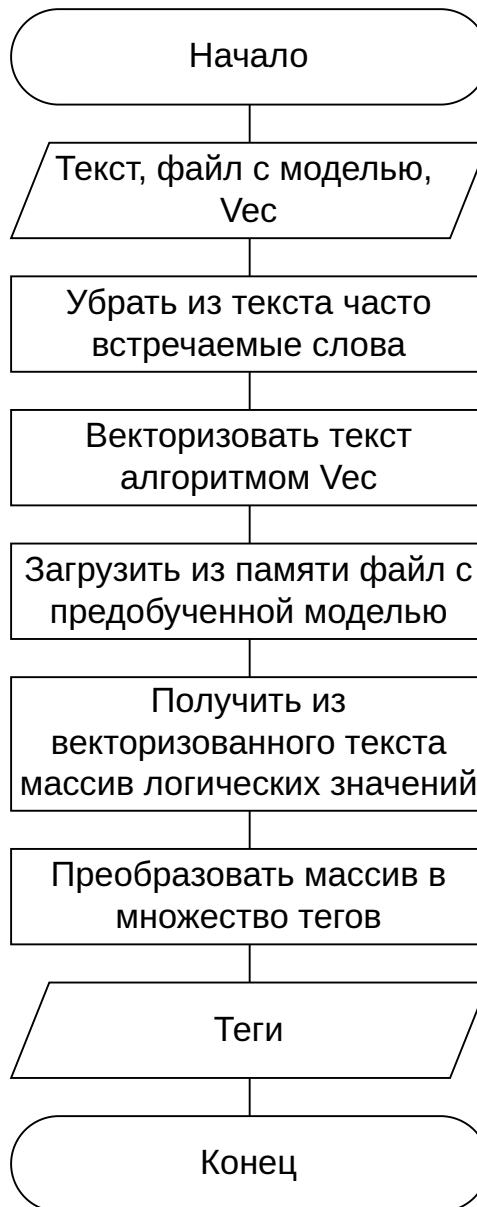


Рисунок 3: Алгоритм работы программы в пользовательском режиме

На рисунке 4 показана схема работы программы в режиме обучения модели. Пользователь может выбрать алгоритм *Vec* векторизации текста, алгоритм *Class* для обучения модели, а также долю тестовой выборки в датасете (*test_size*). Также можно указать имя и расположение файла, в который будет сохранена новая модель, и путь к файлу с датасетом.

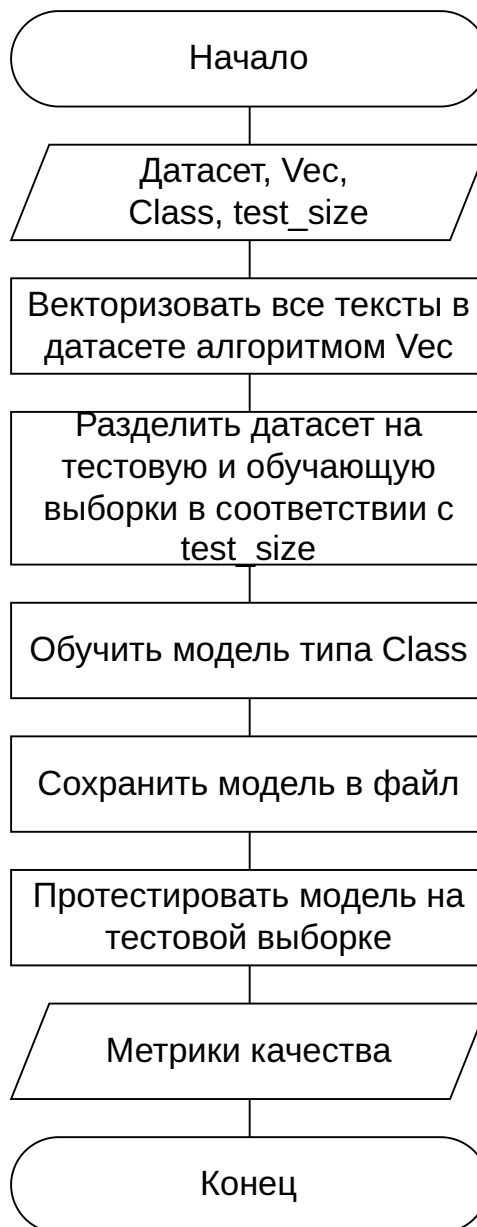


Рисунок 4: Алгоритм работы программы в режиме обучения

3 Технологический раздел

3.1 Выбор языка и среды программирования

В качестве языка программирования был выбран язык Python [9]. Язык пользуется популярностью у исследователей и разработчиков моделей машинного обучения благодаря своей простоте и лаконичности, что позволяет быстро разрабатывать сложные алгоритмы. Это привело к появлению множества библиотек и фреймворков, значительно упрощающих разработку программного обеспечения.

В качестве среды разработки была выбрана Visual Studio Code [10]. Она бесплатная, кроссплатформенная, также предоставляет возможность установки дополнений, например, подсветку синтаксиса и статический анализатор кода.

В программе были использованы следующие библиотеки:

- scikit-learn;
- gensim;
- pandas;
- matplotlib;
- numpy;
- skops.

Библиотека scikit-learn [11] используется для обучения модели, так как в ней присутствуют встроенные алгоритмы для многозначной классификации, а также встроенные метрики для оценки качества полученной модели.

Библиотека gensim [12] необходима для реализации векторизации методом Word2Vec. Помимо инструментов для данного типа векторизации, библиотека предоставляет доступ к предобученным нейронным сетям, которые можно использовать в Word2Vec.

Библиотека pandas [13] предназначена для работы с данными. Она предоставляет высокоуровневые структуры данных, которые облегчают об-

работку, манипуляцию и анализ данных, а также включает в себя широкое множество функций для работы с ними.

Matplotlib [14] предоставляет инструменты для построения графиков, диаграмм и других видов визуализации данных.

Skops [15] — это библиотека, предназначенная для работы с моделями машинного обучения, созданными с использованием scikit-learn. Она используется для сохранения и загрузки моделей.

Библиотека numpy [16] создана для работы с массивами и матрицами, а также для выполнения математических и логических операций над ними. Она используется, так как поддерживается многими другими библиотеками, в том числе описанными выше.

3.2 Код программы

3.2.1 Обработка текста

В листингах 1-2 приводится код модуля, отвечающего за векторизацию текста. Реализованы два метода векторизации — TF-IDF и Word2Vec.

Листинг 1: Модуль обработки текста

```
1 import nltk
2 import re
3 import numpy as np
4 import pandas as pd
5
6 from sklearn.feature_extraction.text import TfidfVectorizer
7 from sklearn.preprocessing import normalize
8 from nltk.corpus import stopwords
9 from gensim.models import Word2Vec, KeyedVectors
10 from gensim import downloader
11 import skops.io as sio
12
13 stopWords = stopwords.words("english")
14
15 def delete_stop_words(text):
16     return [word for word in text if word not in stopWords]
17
18 def format_data(text):
19     fmtTextStr = re.sub(r'[^\\w\\s]', ' ', text.lower(), flags=re.UNICODE)
20     tokenTextArr = nltk.word_tokenize(fmtTextStr)
21     cleanTextArr = delete_stop_words(tokenTextArr)
22     return " ".join(cleanTextArr)
```

```

1 def vectorize_tf_idf(data):
2     vocabulary = None
3     try:
4         vocabulary = sio.load("model/vocab.skops")
5     except:
6         pass
7     vectorizer = TfidfVectorizer(sublinear_tf=True, min_df=0.01, max_df=0.95,
8         stop_words=stopWords, vocabulary=vocabulary)
9     x = vectorizer.fit_transform(data['text'])
10    sio.dump(vectorizer.vocabulary_, "model/vocab.skops")
11    return x
12
13 def _preprocess(s):
14     return [i for i in s.split()]
15
16 def _get_vector(s, model):
17     try:
18         return np.sum(np.array([model[i] for i in _preprocess(s) if i in model]),
19             axis=0)
20     except:
21         return 0
22
23 def vectorize_word2vec(data):
24     w2v_model = downloader.load('word2vec-google-news-300')
25     x = [_get_vector(text, w2v_model) for text in data["text"]]
26     return normalize(x)
27
28 def vectorize(data, mode):
29     if isinstance(data, str):
30         data = pd.DataFrame({"text": [data]})
31
32     data["text"] = data.text.apply(lambda text: format_data(text))
33
34     if mode == "tf-idf":
35         return vectorize_tf_idf(data)
36
37     return vectorize_word2vec(data)

```

3.2.2 Работа с моделями

В листингах 3-6 приводятся фрагменты кода модуля, отвечающего за модели. Помимо кода ниже, в модуле реализованы экспорт и импорт моделей из файлов.


```

1 import skops.io as sio
2 import numpy as np
3 import pandas as pd
4
5 from sklearn.multioutput import MultiOutputClassifier, ClassifierChain
6 from sklearn.linear_model import LogisticRegression
7 from sklearn.metrics import jaccard_score, precision_score, recall_score,
    accuracy_score
8 from sklearn.model_selection import train_test_split
9
10 PRED_OK = 0.5
11
12 def import_model(filename):
13     unknown_types = sio.get_untrusted_types(file=filename)
14     return sio.load(filename, trusted=unknown_types)
15
16 def export_model(model, filename):
17     obj = sio.dump(model, filename)
18
19 def train_moc(data, labels, test_size):
20     X_train, X_test, Y_train, Y_test = train_test_split(data, labels, test_size=
        test_size)
21     clf = MultiOutputClassifier(LogisticRegression())
22     model = clf.fit(X_train, Y_train)
23
24     predicted_y = model.predict(X_test)
25     Y_test_arr = np.array(Y_test).reshape(1, -1)[0]
26     predicted_y_arr = predicted_y.reshape(1, -1)[0]
27
28     return {
29         "model": model,
30         "precision": precision_score(Y_test_arr, predicted_y_arr),
31         "recall": recall_score(Y_test_arr, predicted_y_arr),
32         "accuracy": accuracy_score(Y_test_arr, predicted_y_arr),
33         "jaccard": jaccard_score(Y_test_arr, predicted_y_arr)
34     }
35
36 def train(data, labels, test_size, chains_amt=0):
37     if chains_amt == 0:
38         return train_moc(data, labels, test_size)
39
40     return train_chain(data, labels, test_size, chains_amt)

```

Листинг 4: Обучение цепочки классификаторов

```

1 def train_chain(data, labels, test_size, chains_amt):
2     X_train, X_test, Y_train, Y_test = train_test_split(data, labels, test_size=
        test_size)
3     chains = [ClassifierChain(LogisticRegression(), order="random", random_state
        =i) for i in range(chains_amt)]
4     for chain in chains:
5         chain.fit(X_train, Y_train)
6
7     Y_pred_chains = np.array([chain.predict_proba(X_test) for chain in chains])
8     Y_pred_ensemble = Y_pred_chains.mean(axis=0).reshape(1, -1)[0]
9     Y_test_arr = np.array(Y_test).reshape(1, -1)[0]
10
11     ensemble_jaccard_score = jaccard_score(
12         Y_test_arr, Y_pred_ensemble >= PRED_OK
13     )
14
15     ensemble_precision_score = precision_score(
16         Y_test_arr, Y_pred_ensemble >= PRED_OK
17     )
18
19     ensemble_recall_score = recall_score(
20         Y_test_arr, Y_pred_ensemble >= PRED_OK
21     )
22
23     ensemble_accuracy_score = accuracy_score(
24         Y_test_arr, Y_pred_ensemble >= PRED_OK
25     )
26
27     return {
28         "model": chains,
29         "precision": ensemble_precision_score,
30         "recall": ensemble_recall_score,
31         "accuracy": ensemble_accuracy_score,
32         "jaccard": ensemble_jaccard_score
33     }

```

Листинг 5: Применение моделей

```

1 def to_tags_list(labels):
2     tags = []
3     i = 0
4     for key in KEYS:
5         if labels[0][i]:
6             tags.append(key)
7         i += 1
8     return tags

```

```
1 def predict_tags(text, model):  
2     if isinstance(model, list):  
3         Y_pred_chains = np.array([chain.predict_proba(text) for chain in model])  
4         return to_tags_list(Y_pred_chains.mean(axis=0).reshape(1, -1) >= PRED_OK  
5                               )  
6     return to_tags_list(model.predict(text))
```

Помимо описанных выше функций в программе реализовано меню, посредством которого пользователь осуществляет ввод данных, а также управляющий модуль, который на основе введенных данных вызывает соответствующие функции.

4 Исследовательский раздел

4.1 Исследование моделей

Так как результатом многозначной классификации является последовательность логических выражений «истина» и «ложь», то для оценки ее точности можно использовать те же параметры, основанные на матрице ошибок, что и в классической бинарной классификации.

К этим показателям относятся [17]:

- точность (precision) — показывает, сколько объектов, предсказанных положительными, действительно таковыми являются ($TP/(TP + FP)$);
- полнота (recall) — показывает, сколько объектов, являющихся положительными, были правильно обнаружены моделью ($TP/(TP + FN)$);
- общая точность (ассигасу) — показывает, как часто модель предсказывает любой класс верно (доля правильных ответов, $(TP + TN)/(TP + TN + FN + FP)$);
- f1-мера (F1-score) — гармоническое среднее точности p и полноты r ($2pr/(p + r)$).

В данной задаче выходной массив можно назвать разреженным, так как в исходном датасете из 98 тегов одной статье соответствует не более 5 тегов. По этой причине, за счет того что подавляющее число значений в выходном векторе являются значениями «ложь», значение ассигасу, вероятнее всего, будет высоким, но при этом недостаточно хорошо отражающим качество модели.

На рисунке 5 продемонстрирован график зависимости общей точности классификации от объема обучающей выборки для разных моделей. На нем можно видеть, что эта метрика везде приблизительно составляет 98%.

Ниже будут рассмотрены другие показатели, которые являются более репрезентативными для данной задачи.

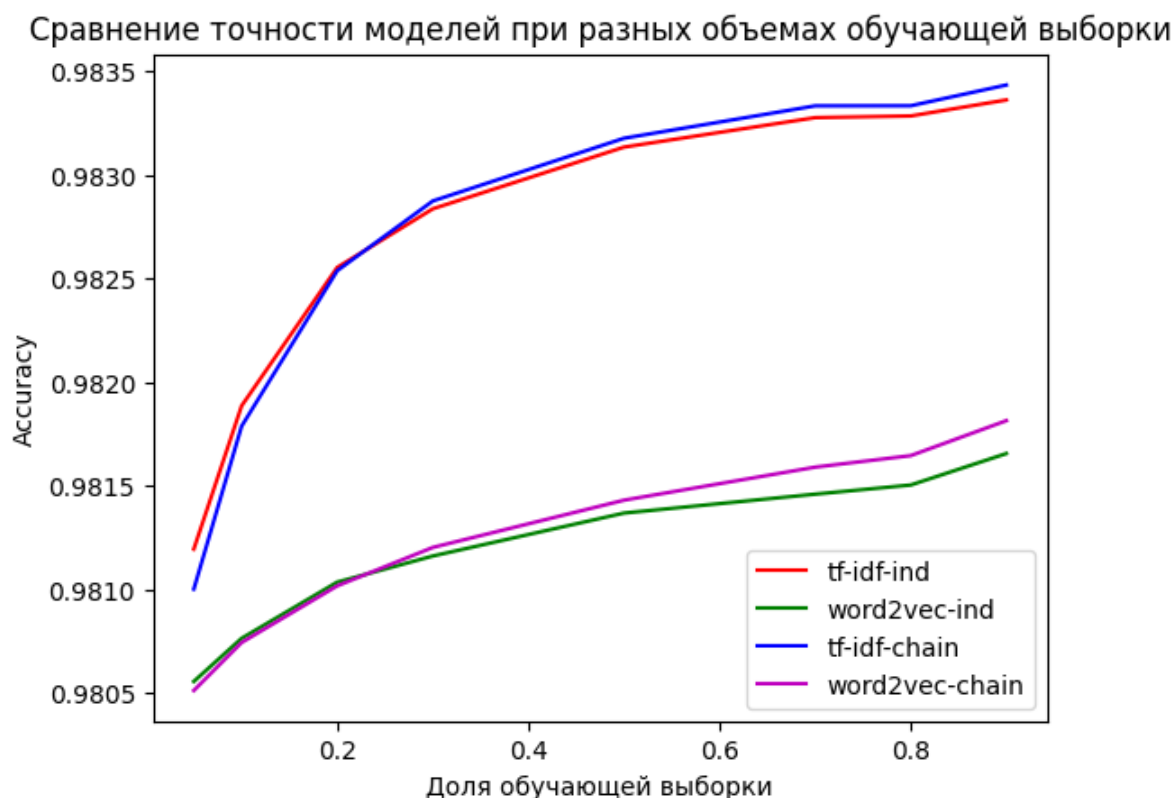


Рисунок 5: Значение метрики ассурасу для разных моделей

Показатель *recall* при многозначной классификации подходит в задачах, где очень важно обнаружить все релевантные метки и пропуск любой метки может привести к значительным негативным последствиям. В случае определения тегов для текста вполне допустимо пропустить одну или несколько меток, сходных по значению, присвоив только некоторые.

С другой стороны, показатель *precision* позволяет понимать, как часто модель ошибается, присваивая теги, которые на самом деле отсутствуют. С точки зрения определения тегов этот показатель более информативен, так как важно присваивать метки, соответствующие материалам текста.

Так как ранее было выяснено, что для цепочки классификаторов требуется строить несколько моделей и усреднять их для получения лучшего качества, сперва необходимо исследовать влияние числа этих моделей на результат с целью подбора оптимального. На рисунке 6 приводится график зависимости точности результатов, предсказанных моделью, от числа цепочек в этой модели.



Рисунок 6: Зависимость метрики precision от количества классификаторов в цепочке

Из графика следует, что примерно при 5-10 цепочках значение точности выравнивается и дальнейшее увеличение их числа уже не приводит к повышению качества модели. Поэтому далее при сравнении моделей будет указываться результат для модели с 10 цепочками.

На рисунке 7 продемонстрирован график зависимости точности классификации от объема обучающей выборки для разных моделей.

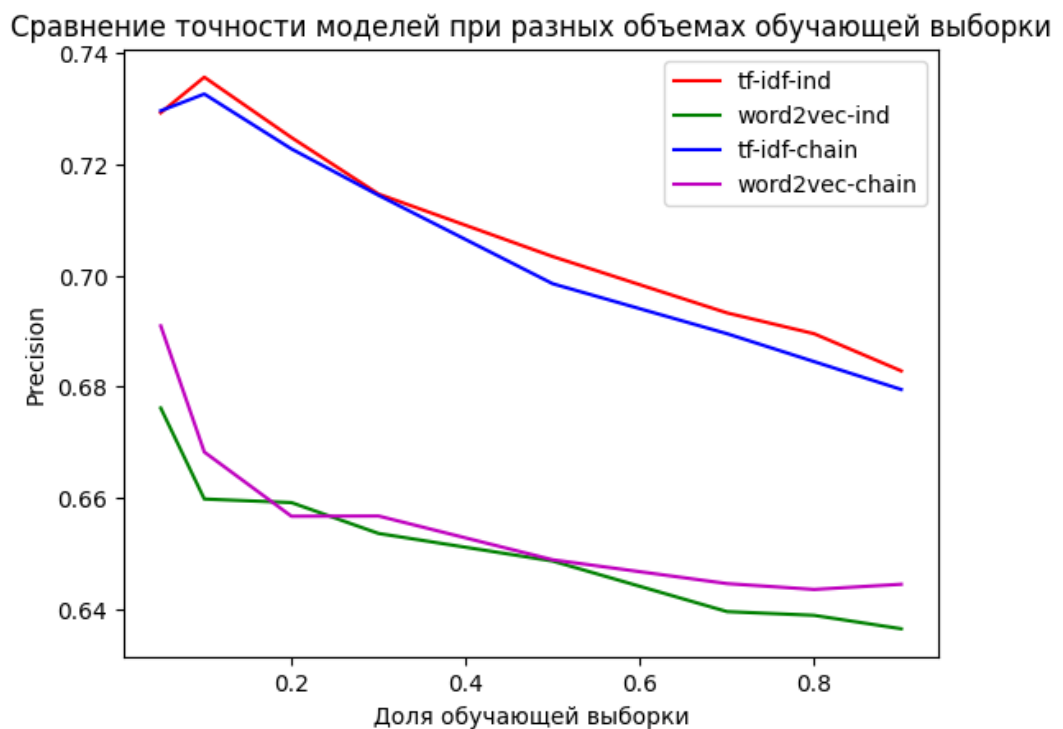


Рисунок 7: Значение метрики precision для разных моделей

На графике можно видеть, что при выборе метода векторизации TF-IDF исследуемая метрика получается примерно на 5 процентных пунктов (или на 7.5%) выше, чем при векторизации методом Word2Vec, и составляет примерно 70-73%. Также можно отметить, что наивысшая точность получается при небольшой тестовой выборке, примерно 5-10% от общего объема датасета. Далее происходит переобучение и точность снижается. В то же время разница между использованием независимых классификаторов и их цепочки незначительна.

Также было вычислено значение метрики, аналогичной recall и обозначающей долю данных тестовой выборки, для которых модель верно предсказала хотя бы один тег. Среднее гармоническое этой метрики и precision (аналог f1-меры) приводится на рисунке 8. При этом сохраняется тенденция метода векторизации TF-IDF к более высоким показателям качества в данной задаче, чем у Word2Vec.

Сравнение точности моделей при разных объемах обучающей выборки

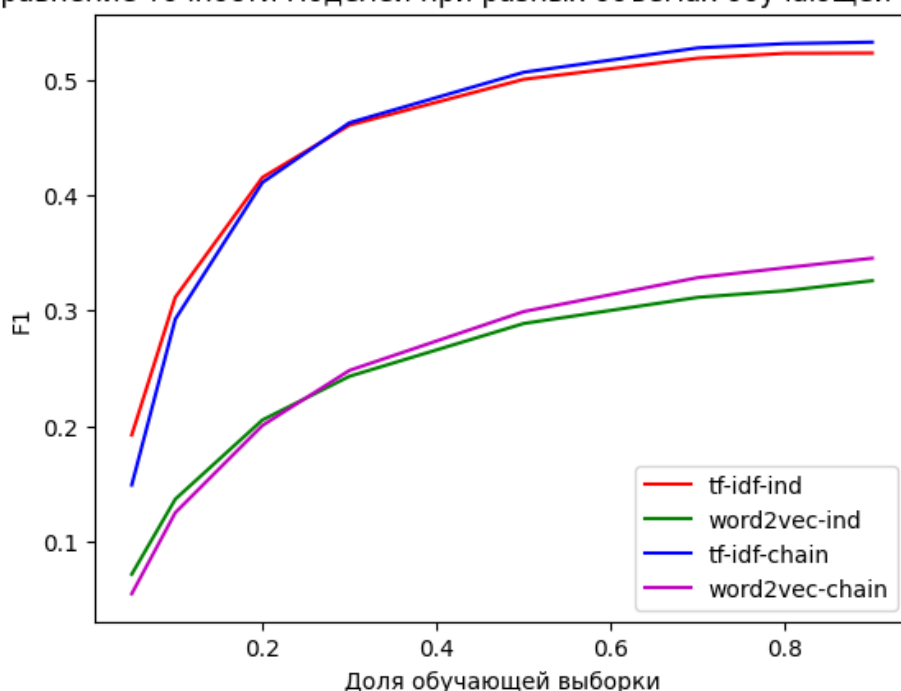


Рисунок 8: Значение аналога метрики f1 для разных моделей

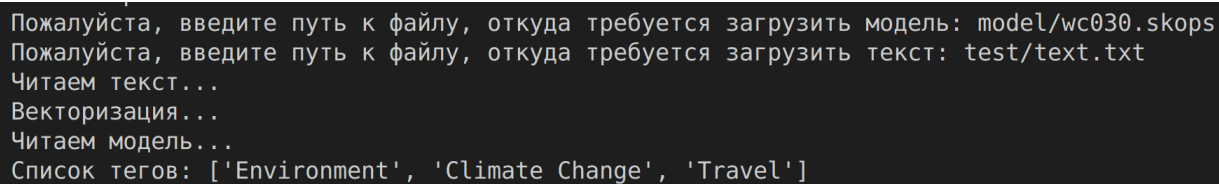
Таким образом, наилучшей из исследованных моделей является модель на основе векторизации TF-IDF, построенная на обучении независимых классификаторов, так как он обеспечивает тот же уровень качества, что и цепочка

классификаторов, но работает быстрее за счет возможности параллелизации и отсутствия дополнительных входных параметров.

Объем тренировочной выборки можно варьировать в зависимости от целей: если не так важно определить все теги для текста, но определять их точнее, то объем тренировочной выборки может составлять около 10% от всего датасета. Если же важно получить более широкое множество релевантных тегов, то для повышения значения recall можно увеличить долю тренировочной выборки вплоть до 90%.

4.2 Демонстрация работы программы

На рисунке 9 изображен пример вывода программы при вводе текста, взятого не из датасета и не с источника Medium. В тексте рассказывается об особенностях экотуризма, его преимуществах и отличиях от массового туризма. Как можно видеть, модель распознала теги «Окружающая среда», «Изменение климата» и «Путешествия», что соответствует тематике текста.



```
Пожалуйста, введите путь к файлу, откуда требуется загрузить модель: model/wc030.skops
Пожалуйста, введите путь к файлу, откуда требуется загрузить текст: test/text.txt
Читаем текст...
Векторизация...
Читаем модель...
Список тегов: ['Environment', 'Climate Change', 'Travel']
```

Рисунок 9: Демонстрация работы программы

ЗАКЛЮЧЕНИЕ

В результате выполнения курсовой работы:

- проанализированы и выбраны методы векторизации и многозначной классификации, необходимые для решения поставленной задачи;
- описан набор данных, на которых проводилось обучение модели;
- разработано программное обеспечение, реализующее обучение модели, а также определение тегов к произвольному тексту на основе предобученной модели;
- выбраны метрики качества и сравнены различные реализации модели по этим метрикам.

В результате сравнения был выбран метод векторизации TF-IDF в сочетании с обучением независимых бинарных классификаторов.

Таким образом, была разработана и протестирована модель для подбора тегов к англоязычным текстам.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Абрамов, Е. Г. Подбор ключевых слов для научной статьи / Е. Г. Абрамов // Научная периодика: проблемы и решения. — 2011. — № 2(2). — С. 35-40.
2. Кравченко, Ю. А. Векторизация текста с использованием методов интеллектуального анализа данных / Ю. А. Кравченко, А. М. Мансур, Ж. Х. Мохаммад // Известия ЮФУ. Технические науки. — 2021. — № 2(219). — С. 154-167.
3. Le Q., Mikolov T. Distributed Representations of Sentences and Documents // Proceedings of the 31st International Conference on Machine Learning. T. 32 / под ред. Е. Р. Xing, Т. Jebara. — Beijing, China : PMLR, 22–24 Jun.2014. — С. 1188–1196.
4. Mikolov T. Efficient Estimation of Word Representations in Vector Space / Т. Mikolov [и др.]. — 2013.
5. Jesse Read, Bernhard Pfahringer, Geoff Holmes. Classifier Chains for Multilabel Classification. — 2009.
6. Multiclass and multioutput algorithms — scikit-learn 1.5.0 documentation [Электронный ресурс]. Режим доступа: <https://scikit-learn.org/stable/modules/multiclass.html> (дата обращения: 26.05.2024).
7. 190k+ Medium Articles | Kaggle [Электронный ресурс]. Режим доступа: <https://www.kaggle.com/datasets/fabiochiusano/medium-articles> (дата обращения: 26.05.2024).
8. РД IDEF0-2000. Методология функционального моделирования IDEF0: руководящий документ. // М.: Госстандарт России. — 2000. — 75 с.
9. Our Documentation | Python.org [Электронный ресурс]. Режим доступа: <https://www.python.org/doc/> (дата обращения: 06.06.2024).

10. Visual Studio Code — Code Editing. Redefined [Электронный ресурс]. Режим доступа: <https://code.visualstudio.com/> (дата обращения: 06.06.2024).
11. scikit-learn: machine learning in Python — scikit-learn 1.5.0 documentation [Электронный ресурс]. Режим доступа: <https://scikit-learn.org/stable/index.html> (дата обращения: 08.06.2024).
12. Gensim: Topic modelling for humans [Электронный ресурс]. Режим доступа: <https://radimrehurek.com/gensim/> (дата обращения: 08.06.2024).
13. pandas — Python Data Analysis Library [Электронный ресурс]. Режим доступа: <https://pandas.pydata.org/> (дата обращения: 08.06.2024).
14. Matplotlib — Visualization with Python [Электронный ресурс]. Режим доступа: <https://matplotlib.org/> (дата обращения: 08.06.2024).
15. Welcome to skops's documentation! — skops 0.9 documentation [Электронный ресурс]. Режим доступа: <https://skops.readthedocs.io/en/stable/> (дата обращения: 08.06.2024).
16. NumPy [Электронный ресурс]. Режим доступа: <https://numpy.org/> (дата обращения — 08.06.2024).
17. Metrics for Multilabel Classification | Mustafa Murat ARAT [Электронный ресурс]. Режим доступа: https://mmuratarat.github.io/2020-01-25/multilabel_classification_metrics (дата обращения: 06.06.2024).