МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение

высшего образования

**«Северный (Арктический) федеральный университет имени М.В. Ломоносова»**

Высшая школа информационных технологий и автоматизированных систем

(наименование высшей школы / филиала / института / колледжа)

**КУРСОВОЙ ПРОЕКТ**

|  |  |  |  |
| --- | --- | --- | --- |
| По модулю | | Разработка AI проекта | |
|  | | | |
|  | | | |
| На тему | Разработка сервиса оценки привлекательности северного региона по | |
| данным социальных медиа на примере Архангельской области | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Ф.И.О.  обучающихся | Наименование направления подготовки / специальности | Курс | Группа | Ф.И.О. руководителя (-ей) должность / уч. степень / звание |
| Скок А.С. | 09.03.02 Информационные системы и технологии | 4 | 351018 | Васендина И.С., доцент, к.т.н. |
| Сивков И.Г. | 09.03.02 Информационные системы и технологии | 4 | 351018 | Васендина И.С., доцент, к.т.н. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Признать, что проект выполнен и защищен с отметкой |  |  |  |  |
|  |  | (отметка прописью) |  | (дата) |
| Руководитель |  |  |  | И.С. Васендина |
|  |  | (подпись руководителя) |  | (инициалы, фамилия) |

Архангельск 2024

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ | | | | | | | | | | | | | | | | | | | | | |
| федеральное государственное автономное образовательное учреждение | | | | | | | | | | | | | | | | | | | | | |
|  | высшего образования  **«Северный (Арктический) федеральный университет имени М.В. Ломоносова»** | | | | | | | | | | | | | | | | | | |  | |
|  |  | | | | | | | | | | | | | | | | | | |  | |
|  | Высшая школа информационных технологий и автоматизированных систем | | | | | | | | | | | | | | | | | | |  | |
|  | (наименование высшей школы / филиала / института / колледжа) | | | | | | | | | | | | | | | | | | |  | |
|  |  | | | | | | | | | | | | | | | | | | |  | |
|  | **ЗАДАНИЕ НА КУРСОВОЙ ПРОЕКТ** | | | | | | | | | | | | | | | | | | |  | |
|  |  | | | | | | | | | | | | | | | | | | |  | |
|  | ппо | Разработка AI проекта | | | | | | | | | | | | | | | | | |  | |
|  |  | | | | (наименование дисциплины) | | | | | | | | | | | | | | |  | |
|  | студенту | | ВШИТиАС | | высшей школы | | | | 4 | | | | | курса | 351018 | | | | группы |  | |
|  | Скоку Андрею Сергеевичу, Сивкову Ивану Георгиевичу | | | | | | | | | | | | | | | | | | |  | |
|  | (фамилия, имя, отчество студента) | | | | | | | | | | | | | | | | | | |  | |
|  | |  | | --- | | 09.03.02 «Информационные системы и технологии» | | (код и наименование направления подготовки/специальности) | | | | | | | | | | | | | | | | | | | |  | |
|  | ТЕМА: | | | Разработка сервиса оценки привлекательности северного региона по | | | | | | | | | | | | | | | |  | |
|  | данным социальных медиа на примере Архангельской области | | | | | | | | | | | | | | | | | | |  | |
|  |  | | | | | | | | | | | | | | | | | | |  | |
|  | ИСХОДНЫЕ ДАННЫЕ: | | | | | | | Необходимо решить задачу по разработке | | | | | | | | | | | |  | |
|  | сервиса оценки привлекательности северных регионов по данным социальных медиа | | | | | | | | | | | | | | | | | | |  | |
|  | на примере Архангельской области. Описать | | | | | | | | | | | | | | | | | | |  | |
|  | проанализировать целевую переменную, провести предобработку изображений | | | | | | | | | | | | | | | | | | |  | |
|  | актуальность, цель, задачи, описать набор данных или как собирали, сформулировать | | | | | | | | | | | | | | | | | | |  | |
|  | задачу, (если требуется), построить несколько моделей по архитектуре (или с разными | | | | | | | | | | | | | | | | | | |  | |
|  | параметрами) для решения задачи. Обучить модели. Подобрать параметры. Оценить | | | | | | | | | | | | | | | | | | |  | |
|  | качество работы моделей. Обосновать выбор модели. Проверить работу модели на | | | | | | | | | | | | | | | | | | |  | |
|  | тестовых данных. Сделать выводы о качестве работы. Разработать веб сервис. | | | | | | | | | | | | | | | | | | |  | |
|  |  | | | | | | | | | | | | | | | | | | |  | |
|  | Срок выполнения с 12.02.2024 г. по 09.03.2024 г. | | | | | | | | | | | | | | | | | | |  | |
|  | Руководитель работы | | | | | | доцент / ст. | | | |  | |  | | | |  | И.С. Васендина | |  | |
|  |  | | | |  | (должность) | | | |  | | (подпись) | | | |  | | (инициалы, фамилия) | | |

Архангельск 2024

лист для замечаний

Оглавление

[Введение 6](#_Toc161098519)

[1 Описание оценки качества жизни северных регионов 8](#_Toc161098520)

[2 Анализ вариантов и проектирование решения задачи 10](#_Toc161098521)

[2.1 Возможные решения и существующие подходы к классификации текста 10](#_Toc161098522)

[2.2. Примеры коммерческих или открытых решений 10](#_Toc161098523)

[2.3. Выбор и обоснование архитектурного решения 11](#_Toc161098524)

[3 Сбор и подготовка набора данных 12](#_Toc161098525)

[3.1 Источники данных 12](#_Toc161098526)

[3.2 Описание структуры набора данных 13](#_Toc161098527)

[3.3 Процесс сбора данных 15](#_Toc161098528)

[3.4 Процесс разметки данных 17](#_Toc161098529)

[3.4.1 Описание задачи 17](#_Toc161098530)

[3.4.2 Проектирование сервиса разметки 18](#_Toc161098531)

[3.4.3 Размещение на хостинге 20](#_Toc161098532)

[3.4.4 Скрипты для обработки данных 21](#_Toc161098533)

[3.4.5 Результаты разметки данных 22](#_Toc161098534)

[3.5 Описание характеристик данных необходимых для обучения нейронной сети 23](#_Toc161098535)

[3.5.1 Описание классификации тональности 23](#_Toc161098536)

[3.5.2 Описание классификации категорий 24](#_Toc161098537)

[3.5.3 Описание классификации мусор 24](#_Toc161098538)

[4 Обучение нейронной сети 25](#_Toc161098539)

[4.1 Реализация нейронной сети 25](#_Toc161098540)

[4.2 Эксперименты обучения 25](#_Toc161098541)

[4.3 Параметры обучения 26](#_Toc161098542)

[4.4 Подача данных и обучение 27](#_Toc161098543)

[5 Оценка качества нейронной сети и решения задачи 28](#_Toc161098544)

[5.1 Метрики оценки качества нейронной сети 28](#_Toc161098545)

[5.2 Получение предсказаний 28](#_Toc161098546)

[5.3 Результаты оценки качества модели 28](#_Toc161098547)

[6 Разработка web-приложения 31](#_Toc161098548)

[6.1 Проектирование веб приложения 31](#_Toc161098549)

[6.1.1 Планирование и выбор технологий 31](#_Toc161098550)

[6.1.2 Фронтенд 31](#_Toc161098551)

[6.1.3 Бекенд 31](#_Toc161098552)

[6.1.4 Использование Celery и Redis 32](#_Toc161098553)

[6.1.5 Разметка данных с использованием моделей глубокого обучения 32](#_Toc161098554)

[6.1.6 Проектирование дизайна с помощью Figma 33](#_Toc161098555)

[6.2 Разработка веб приложения 33](#_Toc161098556)

[6.2.1 Фронтенд 33](#_Toc161098557)

[6.2.2 Бекенд 35](#_Toc161098558)

[6.2.3 Реализация парсеров через Celery и Redis 35](#_Toc161098559)

[6.2.4 Разметка данных при помощи моделей глубокого обучения 36](#_Toc161098560)

[6.3 Контейнеризация веб приложения 37](#_Toc161098561)

[Заключение 40](#_Toc161098562)

[Список использованных источников 42](#_Toc161098563)

[Приложение А (обязательное) Листинг подготовки данных 43](#_Toc161098564)

[Приложение Б (обязательное) Листинг кода классификатора 46](#_Toc161098565)

[Приложение В (обязательное) Листинг кода интеграции моделей 49](#_Toc161098566)

[Сведения о самостоятельности выполнения работы 51](#_Toc161098567)

Введение

В современном информационном обществе обработка и анализ текстовых данных с использованием технологии глубокого обучения становятся все более актуальными и неотъемлемыми компонентами различных прикладных задач. Одной из ключевых областей применения этой технологии является классификация текстов, позволяющая систематизировать и структурировать большие объемы информации.

Настоящий курсовой проект посвящена решению прикладной задачи классификации текстов с применением глубокого обучения, причем основной упор делается на анализе уровня качества жизни в северных регионах. Текстовая классификация в данном контексте представляет собой эффективный инструмент для выделения и категоризации факторов, влияющих на уровень благосостояния в данных регионах.

Целью данного исследования является разработка и реализация системы, способной автоматически классифицировать текстовую информацию, связанную с жизнью в северных регионах, с использованием передовых методов глубокого обучения. Для достижения этой цели необходимо решить следующие задачи:

1. Определить предметную область и выявить проблемы, связанные с анализом уровня качества жизни в северных регионах.

2. Проанализировать существующие подходы и варианты решения задачи классификации текстов, включая различные архитектуры нейронных сетей, используемые в подобных задачах.

3. Подготовить и описать набор данных, необходимых для обучения и тестирования модели, включая процессы сбора, разметки [5] и аугментации данных.

4. Разработать и реализовать архитектуру нейронной сети для классификации текстов, провести эксперименты с различными параметрами обучения.

5. Оценить качество разработанной модели, используя соответствующие метрики, и сравнить результаты на обучающем и тестовом наборах данных.

6. Сделать выводы о результатах исследования, предложить возможные улучшения разработанной модели, выделить ограничения и предложить пути их преодоления.

Решение поставленных задач позволит не только эффективно классифицировать текстовую информацию, но и предоставит основу для более глубокого анализа факторов, влияющих на качество жизни в северных регионах.

**1 Описание оценки качества жизни северных регионов**

Современное общество сталкивается с неотвратимой потребностью в анализе информации для принятия обоснованных решений в различных областях. Однако, в условиях растущего объема данных, особенно в сфере социально-экономического анализа, становится крайне сложным обработать и структурировать текстовую информацию. В этом контексте, использование технологии глубокого обучения для классификации текстов становится важным инструментом для автоматизации этого процесса.

Одной из актуальных задач, требующих тщательного анализа и систематизации текстов, является оценка уровня качества жизни в северных регионах. Северные территории часто сталкиваются с уникальными вызовами, такими как суровый климат, ограниченные ресурсы и особенности социальной инфраструктуры. Понимание и оценка факторов, влияющих на качество жизни в этих регионах, требуют комплексного анализа множества текстовых данных, включая новостные статьи, обзоры, социальные медиа и другие источники.

Проблемы данной предметной области включают в себя сложность обработки больших объемов неструктурированной информации, а также необходимость учета множества переменных, влияющих на качество жизни. Традиционные методы анализа данных сталкиваются с ограничениями в плане точности и эффективности при работе с текстовой информацией.

Целью настоящего исследования является разработка и реализация системы глубокого обучения для классификации текстов, связанных с уровнем качества жизни в северных регионах. Данная система должна обеспечивать точную и эффективную классификацию текстовых данных, позволяя выделить ключевые факторы, влияющие на качество жизни в данных регионах.

Для достижения поставленной цели перед исследованием стоят следующие задачи:

1. Выполнить обзор существующих методов классификации текстов и архитектур нейронных сетей, применяемых в подобных задачах.

2. Идентифицировать и анализировать источники данных, необходимые для обучения и тестирования модели, включая их структуру и характеристики.

3. Разработать архитектуру нейронной сети, оптимально сочетающую в себе высокую точность классификации и эффективность обработки текстовых данных.

4. Провести эксперименты по обучению модели, настроить гиперпараметры и определить оптимальные условия для достижения высоких показателей качества классификации.

5. Оценить разработанную модель с использованием соответствующих метрик на различных наборах данных, включая обучающий и тестовый наборы.

6. Разработать веб приложение для удобного визуального анализа данных, а также для их интерпретации простым пользователям не имеющих глубоких знаний в данной области.

Анализ и решение данных задач позволят создать эффективную систему классификации текстов, способную применяться для анализа уровня качества жизни в северных регионах и предоставлять ценные выводы для социально-экономических исследований в данных областях.

**2 Анализ вариантов и проектирование решения задачи**

2.1 Возможные решения и существующие подходы к классификации текста

В задаче классификации текстов существует множество подходов и архитектур, используемых для достижения высокой производительности. Рассмотрим несколько ключевых направлений:

- Сверточные нейронные сети (CNN): Этот подход успешно применяется для анализа последовательностей данных. CNN может извлекать локальные признаки из текста, что полезно для задач классификации.

- Рекуррентные нейронные сети (RNN): RNN позволяют учитывать контекст и последовательность в тексте. Однако, они могут сталкиваться с проблемой затухающих градиентов.

- Трансформеры: Архитектура трансформера стала преобладающей в области обработки естественного языка. Модели, такие как BERT [6] (Bidirectional Encoder Representations from Transformers), показали выдающиеся результаты в задачах классификации текстов.

2.2. Примеры коммерческих или открытых решений

Существует ряд решений, основанных на глубоком обучении или предназначенных для обработки текстов и социальных медиа:

SurveyMonkey: Для проведения социальных опросов существуют платформы, такие как SurveyMonkey, которые предоставляют средства для создания, распространения и анализа опросов онлайн.

Hootsuite: Это инструмент для управления социальными медиа, который предоставляет аналитику, мониторинг и планирование публикаций. Позволяет отслеживать и анализировать обсуждения в социальных сетях.

Данные сервисы хоть и присутствуют на рынке уже довольно продолжительное время, однако подобных решений в Российском сегменте не так уж и много и зачастую это обычные опросники или же крупные «сборщики» данных на подобии Яндекса. Для подобных решений не характерно целиком и полностью погружаться в анализ социальных медиа в связи с этим наше решение кажется довольно неплохим и уникальным.

2.3. Выбор и обоснование архитектурного решения

В данной работе мы выбрали для решения задачи классификации текстов модель ruBERTa (Russian RoBERTa). Это обосновано следующими причинами:

- Предобучение на русскоязычных данных: ruBERTa обучен на обширных корпусах русскоязычных текстов, что позволяет ему лучше понимать специфику русского языка.

- Мощность представлений: Благодаря своей архитектуре, ruBERTa способен создавать высокоинформативные векторные представления для текстов, что важно для точной классификации.

- Открытость и доступность: ruBERTa является открытым решением, что обеспечивает удобство в использовании и модификации в соответствии с требованиями конкретной задачи.

В рамках проектирования решения предполагается использование ruBERTa в качестве основной модели для извлечения признаков из текстов и последующей классификации. Потенциально можно рассмотреть вариант использования fine-tuning для более точной адаптации модели к конкретной задаче классификации. Помимо этого, предлагается рассмотреть возможность построения pipeline решения, включая этапы предобработки данных, аугментации и оптимизации гиперпараметров.

**3 Сбор и подготовка набора данных**

3.1 Источники данных

В рамках нашего исследования, одним из важных аспектов стал выбор ресурсов для сбора информации о качестве жизни в регионе. Мы приняли решение использовать социальные сети VK и Telegram в качестве основных источников данных. Это решение обусловлено несколькими факторами, которые делают эти платформы оптимальными для наших целей.

Во-первых, VK и Telegram предоставляют широкий доступ к информации и общению между пользователями, что делает их популярными среди жителей регионов. Это обеспечивает более полное представление о мнениях, проблемах и потребностях населения. Мы можем получить разнообразную информацию от различных групп и сообществ, что способствует формированию более объективной картины качества жизни в регионе.

Во-вторых, VK и Telegram предоставляют доступ к открытым данным через API, что позволяет автоматизировать процесс сбора информации. Это позволяет нам обрабатывать большие объемы данных и получать более точные результаты. Структурированный формат данных на этих платформах также упрощает анализ.

Тем не менее, наше решение не ограничивается только социальными сетями. Для разметки исходных данных мы использовали информацию, собранную из различных источников, предоставляющих данные о событиях и новостях в городе. Эти источники включают новостные паблики в VK, Telegram каналы, новостные сайты, и данные из Яндекс Карт.

Новостные паблики и каналы предоставили широкий спектр информации о различных аспектах жизни в городе. Новостные веб-ресурсы, такие как news29.ru и 29.ru, также внесли свой вклад, обеспечив более полное покрытие событий и разнообразие источников. Данные о отзывах о конкретных местах были извлечены из Яндекс Карт, предоставляя более конкретные данные.

Общий подход к сбору данных включает использование социальных сетей, таких как VK и Telegram, для анализа мнений и обсуждений в регионе, а также использование различных новостных и информационных ресурсов для обеспечения широкого охвата событий и мнений. Этот комбинированный подход позволяет получить комплексную и объективную оценку качества жизни в регионе, а также выявить ключевые проблемы, требующие внимания и решения.

3.2 Описание структуры набора данных

В данной главе рассмотрена структура данных, разработанная для анализа новостных источников. Определены целевые метки, обоснован выбор столбцов и рассмотрены категории для классификации данных. Структура данных включает следующие столбцы: таблица 1.

Таблица 1 – Структура данных

|  |  |
| --- | --- |
| Название колонки | Тип данных, формат |
| Data (Дата создания) | String (строка), формат «yyyy-mm-dd» |
| Time (Время создания) | String (строка), формат «hh:mm:ss» |
| www.news29.ru | Uint8 (целое число) |
| otkrytiiarkhangelsk | Uint8 (целое число) |
| arkhangelsk\_\_online | Uint8 (целое число) |
| Header (Заголовок) | String (строка) |
| Text (Текст поста) | String (строка) |
| Comments\_Count (Количество комментариев) | Uint8 (целое число) |
| Views (Количество просмотров) | Float64 (число с плавающей запятой) |
| Rating (Рейтинг) | Float64 (число с плавающей запятой) |
| Count\_Positive\_Reactions (Количество позитивных реакций) | Float64 (число с плавающей запятой) |

Окончание Таблицы 1

|  |  |
| --- | --- |
| Count\_Negative\_Reactions (Количество негативных реакций) | Float64 (число с плавающей запятой) |
| Reposts (Количество репостов) | Float64 (число с плавающей запятой) |
| Comment\_Text (Текст комментария) | String (строка) |
| Post (является ли постом) | Bool (0 или 1) |

Данная структура позволяет описать каждую собранную нашей командой запись или комментарий. Обоснование выбора столбцов:

- Дата и Время: позволяют анализировать динамику публикаций со временем;

- Ресурсы: идентификация источника помогает определить доверительность информации;

- Заголовок и Текст: содержат основную информацию для анализа тематики;

- Статистика поста: позволяет оценить вовлеченность аудитории;

- Комментарии и Репосты: отражают общественное восприятие;

- Текст комментария: дополнительная информация для анализа;

- Пост и Мусор: определение целевых меток для классификации.

Следующим шагом в разработке структуры данных было выделение целевых меток, таким образом чтобы их можно было использовать в дальнейшем исследовании. В таблице 2 представлены подготовленные метки.

Таблица 2 – Целевые метки

|  |  |
| --- | --- |
| Название колонки | Тип данных, формат |
| Garbage (является ли мусором) | Bool (0 или 1) |
| Healthcare (Здравоохранение) | Bool (0 или 1) |
| Housing\_and\_Public\_Utilities (ЖКХ) | Bool (0 или 1) |
| Education (Сфера образования) | Bool (0 или 1) |
| Infrastructure (Инфраструктура) | Bool (0 или 1) |

Окончание Таблицы 2

|  |  |
| --- | --- |
| Culture (Культура) | Bool (0 или 1) |
| Environmental\_Conditions (Экологическая обстановка) | Bool (0 или 1) |
| Social\_Security (Безопасность) | Bool (0 или 1) |
| Politics (Политика) | Bool (0 или 1) |
| Safety (Социальное обеспечение) | Bool (0 или 1) |
| Availability\_of\_Goods\_and\_Services (Доступность товаров и услуг) | Bool (0 или 1) |
| Official\_Statements (Официальные заявления) | Bool (0 или 1) |
| Tourism (Туризм) | Bool (0 или 1) |
| Facts (Факты) | Bool (0 или 1) |
| Positive (Позитивная) | Bool (0 или 1) |
| Negative (Негативная) | Bool (0 или 1) |
| Neutral (Нейтральная) | Bool (0 или 1) |

Данные метки отражают необходимые для исследования данные, такие как: категорию собранного текста (Туризм, Политика и т.д.) и реакцию людей на новость по одной из данных тематик.

3.3 Процесс сбора данных

Система для сбора данных о качестве жизни в регионе была реализована на языке Python с использованием различных библиотек и инструментов, среди которых:

1. VK и Telegram API: [1],[2]

- requests: для отправки HTTP-запросов к API ВКонтакте;

- pyrogram: библиотека Python для работы с Telegram API.

2. Обработка данных:

- pandas: для создания и обработки данных в формате DataFrame и сохранения данных в CSV файл.

3. Работа с датами и временем:

- datetime и pytz: для работы с датами и временем, полученным из данных ВКонтакте.

4. Управление временными задержками:

- time: для создания задержек между запросами к API, предотвращая превышение лимита количества запросов.

5. Обработка текстовых сообщений Telegram:

- MessageEntityTextUrl из pyrogram.raw.types: для работы с различными сущностями в текстовых сообщениях Telegram, включая ссылки (URL).

6. Дополнительные инструменты:

- glob: стандартный модуль Python для поиска путей файлов, совпадающих с определенным шаблоном в файловой системе;

- dateutil.relativedelta из python-dateutil: для работы с датами и временем в Python, выполняя математические операции с датами;

- nest\_asyncio: для запуска asyncio событий во вложенных циклах событий;

- os: стандартная библиотека Python для работы с операционной системой, предоставляющая функции для работы с файлами, путями и другими системными операциями;

- json: для кодирования и декодирования данных в формате JSON;

- asyncio: стандартная библиотека Python для асинхронного программирования;

- csv: для работы с файлами CSV, чтения и записи данных в CSV формате.

Для парсинга данных с различных ресурсов, включая социальные сети и другие источники, были использованы следующие библиотеки и инструменты:

1. Парсинг веб-страниц:

- requests: для отправки HTTP-запросов к серверу и обработки полученных ответов;

- beautiful soup 4: для парсинга HTML- и XML-документов, облегчая обход дерева DOM и извлечение нужных элементов;

- selenium: для автоматизации действий в веб-браузере;

- fake user agent: для генерации случайных User-Agent'ов для HTTP-запросов.

2. Обработка данных:

- pandas: для создания и обработки данных в формате DataFrame и сохранения данных в CSV файл.

3. Стандартизация формата даты и времени:

- time: для стандартизации формата даты и времени, а также визуализации оставшегося времени работы парсинга.

Обе системы предоставляют удобные и эффективные инструменты для сбора, обработки и анализа данных, а также управления временными задержками при взаимодействии с API различных платформ.

3.4 Процесс разметки данных

3.4.1 Описание задачи

Нашей команде была поставлена сложная задача – разметить огромное количество данных. В стремлении упростить и ускорить этот процесс, мы приняли решение разработать инновационный сервис, предоставляющий несравненно удобный интерфейс для выполнения данной масштабной задачи.

Excel, несмотря на свою широкую распространенность, оказался неэффективным средством для разметки такого объема информации. Поэтому было решено создать веб-платформу с передовым интерфейсом, который позволил бы нашей команде более эффективно и продуктивно осуществлять процесс разметки данных. Новый веб-сайт предоставляет интуитивно понятные инструменты и функциональность, обеспечивая команде максимальное удобство в работе с огромным объемом данных.

3.4.2 Проектирование сервиса разметки

Для того чтобы создать пользовательский интерфейс, который не только удобен, но и привлекателен, мы разработали дизайн с использованием инструмента Figma (см. рисунок 1 и рисунок 2). Данный дизайн включает в себя современные элементы управления, которые значительно облегчают процесс разметки данных. Наши усилия по созданию дизайна направлены на то, чтобы обеспечить пользователей максимальным комфортом при взаимодействии с интерфейсом, что, в свою очередь, способствует более эффективному выполнению задач по разметке.

Изображение выглядит как текст, снимок экрана, программное обеспечение, веб-страница

Автоматически созданное описание

Рисунок 1 – Внешний вид главной страницы

Изображение выглядит как текст, снимок экрана, Операционная система, дизайн

Автоматически созданное описание

Рисунок 2 – Внешний вид страницы авторизации

Для обеспечения функциональности веб-сайта мы разработали бэкэнд, используя фреймворк Django[3]. Этот подход дал нам возможность эффективно управлять данными, обрабатывать запросы пользователей и взаимодействовать с базой данных. Создание бэкэнда на основе Django стало ключевым шагом, обеспечивающим надежную и плавную работу функциональных аспектов нашего сайта. Наш бэкэнд обеспечивает не только стабильность, но и высокую производительность при обработке различных операций, что значительно улучшает общий опыт пользователей во взаимодействии с сайтом. Структура моделей Django представлено на рисунке 3.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 3 – Структура моделей Django

Для разработки фронтенда мы создали представления с использованием HTML[4] и осуществили стилизацию с применением CSS. Этот подход гарантировал не только приятный визуальный дизайн нашего сайта, но и обеспечил его простоту и удобство в использовании. HTML-представления обеспечивают структурированное отображение контента, а CSS-стили добавляют эстетику и согласованность в интерфейсе, делая его более привлекательным для пользователей. Этот комплексный подход к разработке фронтенда дал возможность создать не только функциональный, но и эстетически приятный пользовательский интерфейс и представлено это на рисунке 4.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 4 – Пример визуализации HTML и CSS

3.4.3 Размещение на хостинге

Для размещения нашего веб-сайта мы выбрали хостинг "pythonanywhere.com" (см. рисунок 3). Этот выбор обеспечил нам стабильную работу, легкость в развертывании и быстрый доступ к сервису. Процесс размещения включал в себя загрузку файлов на сервер и настройку необходимых параметров для обеспечения корректной работы приложения. Выбор данного хостинга позволил нам сосредоточиться на разработке функциональности, не тратя лишнего времени на сложности в обслуживании сервера. Надежность и простота использования "pythonanywhere.com" значительно упростили процесс размещения и поддержания нашего веб-сайта в активном состоянии.

Изображение выглядит как текст, снимок экрана, Шрифт, веб-страница

Автоматически созданное описание

Рисунок 3 – Хостинг pythonanywhere.com

3.4.4 Скрипты для обработки данных

С целью обеспечения удобства загрузки данных на сайт, мы разработали скрипт, который преобразует CSV-файлы в базу данных Django. Этот инструмент позволяет нам быстро и эффективно добавлять большие объемы информации для последующей разметки. Разработанный скрипт автоматизирует процесс импорта данных, обеспечивая точность и оперативность в обновлении нашей базы данных. Это важное дополнение к функциональности сайта, улучшающее процесс работы с данными и повышающее общую эффективность нашего проекта.

В дополнение к функциональности сайта, мы разработали скрипт для выгрузки размеченных данных. Этот скрипт осуществляет конвертацию таблиц из базы данных Django в CSV-формат, что обеспечивает удобство использования результатов разметки в других системах. Эта возможность выгрузки данных в формате CSV придает гибкость и максимальную переносимость информации, что полезно при интеграции результатов разметки с различными платформами и приложениями. Разработанный скрипт дополняет функциональность нашего проекта, обеспечивая команде удобство в работе с размеченными данными и их последующее использование.

3.4.5 Результаты разметки данных

Благодаря созданному сервису для разметки, мы успешно разметили более 11 000 строк данных (см. Рисунок 4). Удобный интерфейс, в сочетании с функциональностью бэкэнда, обеспечил высокую производительность нашей команды. Разработка сервиса для разметки данных существенно улучшила процесс выполнения поставленной задачи. Эффективность и удобство использования позволили команде успешно завершить поставленную задачу, превысив ожидаемые объемы разметки. Созданный сервис стал незаменимым инструментом, существенно улучшившим рабочий процесс и результативность работы команды.

Изображение выглядит как текст, снимок экрана, Шрифт, Цвет электрик

Автоматически созданное описание

Рисунок 4 – Результаты разметки данных на сайте

3.5 Описание характеристик данных необходимых для обучения нейронной сети

В данном пункте мы осуществляем предобработку данных для обучения нейронной сети, рассматривая три различных классификатора: "Тональность", "Категории" и "Мусор". Каждый классификатор требует специфической обработки данных для оптимального обучения. Ниже представлено описание характеристик данных для каждого классификатора.

3.5.1 Описание классификации тональности

Для классификации текстов по тональности мы использовали данные, представленные в файле `test4.csv`. Первоначально, избавившись от ненужных признаков, таких как "Time", "Data", "Comments\_Count", "Views" и др., мы создали новый признак "Sentiment". Этот признак объединяет информацию о позитивных, негативных и нейтральных отзывах в единый столбец, где:

- Позитивный отзыв представлен значением 1;

- Негативный отзыв представлен значением 2;

- Нейтральный отзыв представлен значением 0.

Для обеспечения корректности исследования, мы также устранили возможное наличие пропущенных значений в данных. Затем текстовые данные были обработаны с использованием регулярных выражений для удаления спецсимволов и приведения текста к единому формату. Наконец, данные были разделены на обучающий и тестовый наборы. Листинг кода можно увидеть в приложении А-1.

3.5.2 Описание классификации категорий

Для классификации текстов по категориям мы использовали два набора данных: `categories\_train.csv` и `categories\_test.csv`. Каждый текст был отнесен к одной из 13 категорий, представленных в виде текстового значения. Для обучения нейронной сети необходимо было преобразовать текстовые категории в числовые значения. Для этого мы использовали отображение, сопоставив каждую категорию уникальному числовому идентификатору. Далее, данные были подготовлены аналогично предыдущему случаю: обработка текста и разделение на обучающий и тестовый наборы. Листинг кода можно увидеть в приложении А-2.

3.5.3 Описание классификации мусор

Для классификации текстов на предмет наличия "мусора" мы использовали набор данных, представленный в файлах `garbage\_train.csv` и `garbage\_test.csv`. Каждый текст был отнесен к категории "мусор" или "не мусор". Подготовка данных включала в себя аналогичные этапы: обработку текста, подготовку обучающего и тестового наборов.

Таким образом, обработка данных в каждом классификаторе была направлена на корректное представление текстовой информации для обучения нейронных сетей, а также на обеспечение единообразия в формате и структуре данных, что является ключевым аспектом успешного обучения моделей. Листинг кода можно увидеть в приложении А-3.

**4 Обучение нейронной сети**

4.1 Реализация нейронной сети

Для обучения нейронной сети использовалась предварительно обученная модель BertForSequenceClassification [7] из библиотеки Transformers. Мы использовали русскоязычную модель ruBERT (Russian BERT) исходя из задачи классификации текстов. Код для инициализации модели и токенизатора представлен в листинге 1.

Листинг 1 – Инициализации модели и токенизатора

model = BertForSequenceClassification.from\_pretrained('DeepPavlov/rubert-base-cased-sentence', num\_labels=3).to("cuda")

tokenizer = BertTokenizer.from\_pretrained('DeepPavlov/rubert-base-cased-sentence')

Мы загрузили предварительно обученные веса модели, настроенные для задачи классификации с тремя классами. num\_labels=3 указывает на количество классов, в данном случае, это число тональностей (позитивная, негативная, нейтральная).

4.2 Эксперименты обучения

Для обучения модели мы использовали размеченный набор данных, подготовленный в предыдущем этапе. Код для создания датасета и загрузки данных представлен в листинге 2.

Листинг 2 – Создание датасета и загрузки данных

train\_dataset = Data(tokens\_train, train\_labels)

train\_dataloader = torch.utils.data.DataLoader(train\_dataset, batch\_size=8, shuffle=True)

Мы создали DataLoader для обеспечения эффективного чтения данных в процессе обучения. Важно отметить, что мы провели проверку на наличие индекса в self.labels и преобразовали self.labels[idx] в целое число в методе getitem.

4.3 Параметры обучения

Для обучения модели использовались следующие параметры, которые показаны в таблице 3.

Таблица 3 – Параметры для обучения модели

|  |  |  |
| --- | --- | --- |
| Переменная | Значение | Пояснение |
| output\_dir | /kaggle/working/results' | Выходной каталог |
| num\_train\_epochs | 7 | Кол-во эпох для обучения |
| er\_device\_train\_batch\_size | 8 | Размер пакета для каждого устройства во время обучения |
| per\_device\_eval\_batch\_size | 8 | Размер пакета для каждого устройства во время валидации |
| weight\_decay | 0.01 | Понижение весов |
| logging\_dir | '/kaggle/working/logs' | Каталог для хранения журналов |
| load\_best\_model\_at\_end | True | Загружать ли лучшую модель после обучения |
| learning\_rate | 1e-5 | Скорость обучения |
| evaluation\_strategy | 'epoch' | Валидация после каждой эпохи (можно сделать после конкретного кол-ва шагов) |
| logging\_strategy | 'epoch' | Логирование после каждой эпохи |
| save\_strategy | 'epoch' | Сохранение после каждой эпохи |
| save\_total\_limit | 1 | Ограничение на сохранение |
| seed | 21 | Кол-во сидов |

Эти параметры определяют количество эпох обучения, размер пакета данных, скорость обучения, стратегии сохранения и логирования, а также другие настройки для оптимизации процесса обучения.

4.4 Подача данных и обучение

Данные были подготовлены с использованием токенизатора, а затем загружены в DataLoader для эффективного обучения. Процесс обучения запускался при помощи trainer.train(). Код представлен в листинге 4.

Листинг 4 – Процесс обучения

trainer = Trainer(model=model,

tokenizer=tokenizer,

args=training\_args,

train\_dataset=train\_dataset,

eval\_dataset=train\_dataset,

compute\_metrics=compute\_metrics)

trainer.train()

Этот код инициирует процесс обучения, включая сохранение лучшей модели, журналы, и другие параметры, определенные в training\_args.

Обученная модель теперь может использоваться для классификации текстов в соответствии с заданными классами.

**5 Оценка качества нейронной сети и решения задачи**

5.1 Метрики оценки качества нейронной сети

Для оценки качества нейронной сети мы использовали метрику F1, precision и recall так как она подходит для задач с несбалансированными классами. Функция compute\_metrics принимает предсказания модели и реальные метки, а затем вычисляет F1-меру. Код представлен в листинге 5.

Листинг 5 – Оценка качества нейронной сети

def compute\_metrics(pred):

labels = pred.label\_ids

preds = pred.predictions.argmax(-1)

# Выберите подходящее значение для параметра average

average\_type = 'weighted'

f1 = f1\_score(labels, preds, average=average\_type)

return {'F1': f1}

5.2 Получение предсказаний

Для получения предсказаний модели на тестовом наборе данных была использована функция get\_prediction(). Код представлен в листинге 6.

Листинг 6 – Оценка качества нейронной сети

def get\_prediction():

test\_pred = trainer.predict(test\_dataset)

labels = np.argmax(test\_pred.predictions, axis=-1)

return labels

pred = get\_prediction()

5.3 Результаты оценки качества модели

Результаты модели классификации текста на мусор показаны на рисунке 5.

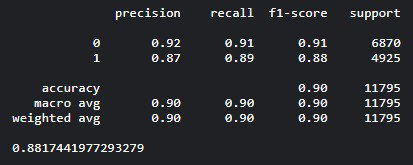


Рисунок 5 - Результаты модели классификации текста на мусор

Результаты модели классификации на категории представлены на рисунке 6.

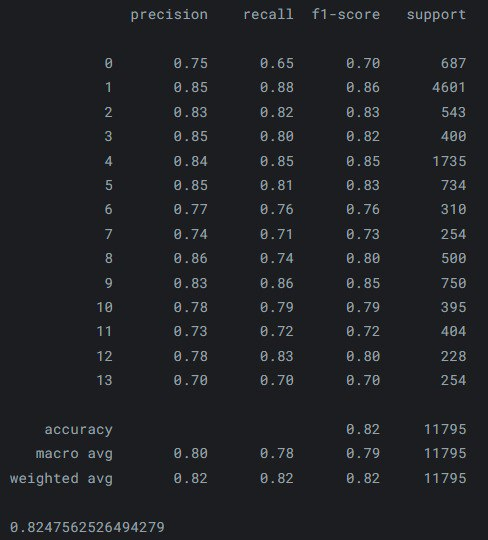


Рисунок 6 - Результаты модели классификации текста на категории

Результаты модели классификации текста на тональность показаны на рисунке 7.

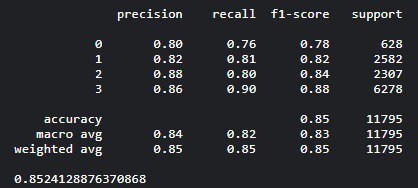


Рисунок 7 - Результаты модели классификации текста на тональность

Модель показала хорошие результаты по классификации текста на мусор, достигнув взвешенной F1-меры в районе 0.88. Однако, в задаче классификации по категориям и тональности, результаты оказались ниже с взвешенной F1-мерой около 0.82 и 0.85 соответственно.

Основываясь на оценках, можно заключить, что модель эффективна в задачах классификации текста на мусор, но требует дополнительных улучшений для более точной классификации по категориям и тональности. Возможные улучшения могут включать в себя тщательную настройку гиперпараметров, использование более сложных архитектур, и улучшение разнообразия и качества обучающего набора данных.

**6 Разработка web-приложения**

6.1 Проектирование веб приложения

6.1.1 Планирование и выбор технологий

При проектировании веб-приложения мы определили его основной целью создание дашборда для удобного отображения и анализа данных. Основная архитектурная идея заключается в том, чтобы разделить приложение на две основные части: фронтенд и бекенд.

6.1.2 Фронтенд

Для фронтенд-части нашего веб-приложения мы выбрали Vue.js. Этот фреймворк предоставляет эффективные инструменты для создания интерактивных пользовательских интерфейсов. Его преимущества включают в себя:

1. Простота в освоении и использовании, что позволит нам быстро разрабатывать интерфейс.

2. Гибкость и масштабируемость, что обеспечит возможность добавления новых компонентов и функций в будущем без больших изменений в коде.

3. Активное сообщество и обширная документация, что обеспечит нам поддержку и ресурсы для решения любых проблем, с которыми мы можем столкнуться.

6.1.3 Бекенд

В качестве бекенд-части нашего приложения мы выбрали Django с использованием его API Framework. Django предлагает широкий набор инструментов для быстрой разработки безопасных и масштабируемых веб-приложений. Его преимущества включают в себя:

1. Комплексность и полнота, включая аутентификацию, авторизацию, работу с базой данных и многое другое, что упростит разработку и улучшит безопасность приложения.

2. Высокая производительность и эффективность, что позволит обрабатывать большие объемы данных и запросов от пользователей.

3. Широкие возможности по настройке и расширению, что позволит нам создать уникальное и функциональное приложение согласно нашим требованиям.

6.1.4 Использование Celery и Redis

Одним из ключевых аспектов нашего проекта является автоматизация процессов сбора данных социальных медиа. Для этого мы планируем использовать Celery и Redis.

Celery — это инструмент для асинхронной работы, который позволяет нам создавать задачи, которые могут быть выполнены в фоновом режиме. Мы используем его для планирования и запуска ежедневных парсеров данных.

Redis — это быстрая база данных, которая используется для кэширования результатов парсинга и обмена сообщениями между различными частями приложения. Его высокая производительность и поддержка различных структур данных делают его идеальным выбором для наших потребностей.

6.1.5 Разметка данных с использованием моделей глубокого обучения

Для разметки данных, полученных из социальных медиа, мы планируем использовать модели глубокого обучения. Это позволит нам автоматически анализировать и классифицировать контент, что улучшит качество и точность данных, сохраняемых в нашей базе данных Django.

6.1.6 Проектирование дизайна с помощью Figma

Для разработки дизайна пользовательского интерфейса (UI) и пользовательского опыта (UX) мы планируем использовать инструмент Figma. Figma предоставляет возможность создавать прототипы, макеты и макеты пользовательского интерфейса в реальном времени.

Использование Figma позволит нам создать современный и удобный пользовательский интерфейс, который будет соответствовать требованиям нашего проекта и ожиданиям пользователей.

Пример прототипа личной карточки региона представлен на рисунке 8.

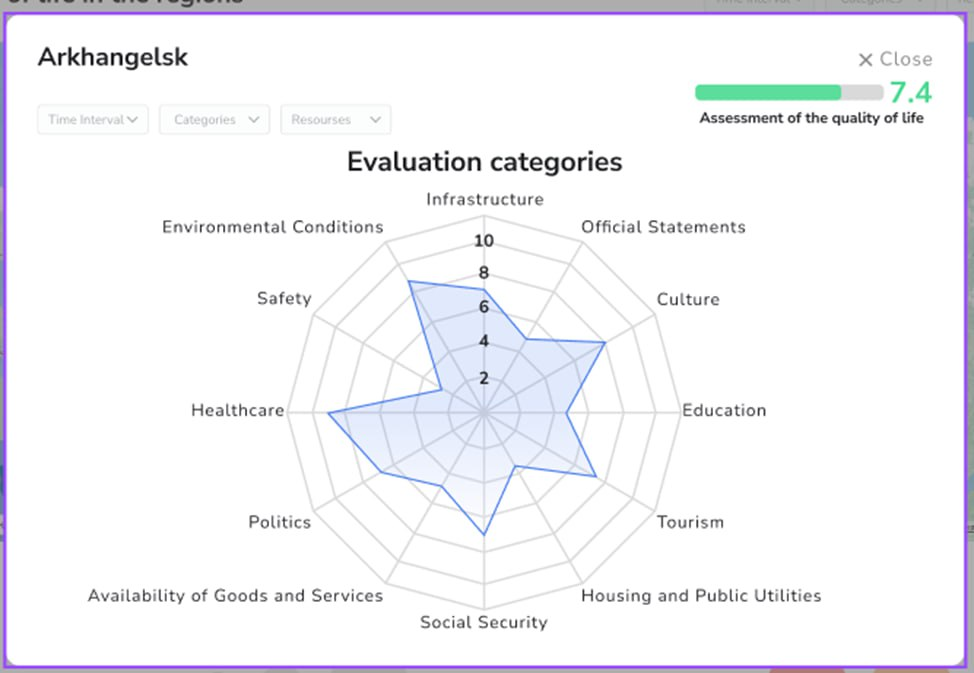


Рисунок 8 – Прототип личной карточки региона

6.2 Разработка веб приложения

6.2.1 Фронтенд

В процессе разработки фронтенд-части нашего веб-приложения мы создали компоненту личного кабинета региона, ориентированную на данные по одному конкретному региону - Архангельску. Этот компонент представляет собой важную часть нашего приложения, обеспечивая пользователей информацией о качестве жизни в данном регионе и представлен он на рисунке 9.

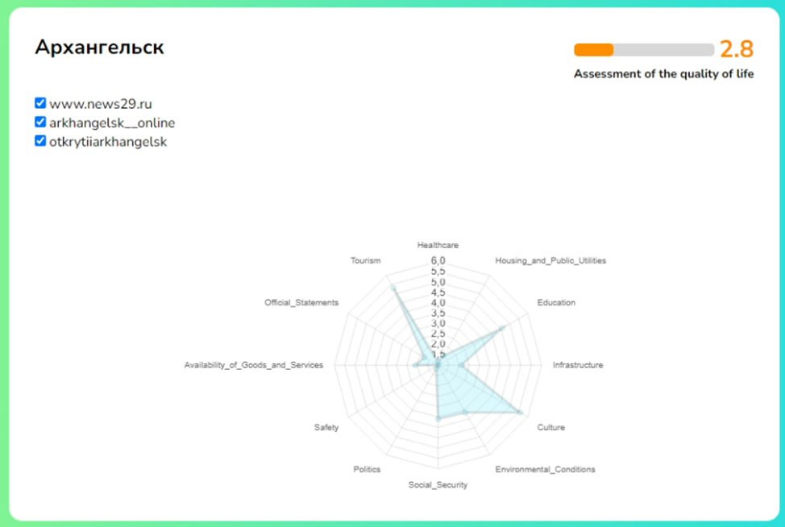


Рисунок 9 – Компонента личной карточки региона

Мы настроили обращение к API Django для получения информации о регионе Архангельску. Это включает в себя запросы к базе данных Django для получения данных о качестве жизни, которые будут отображаться в личном кабинете пользователя.

В компоненте личного кабинета региона мы представили оценки качества жизни для различных категорий, таких как экономика, экология, здравоохранение и т. д. Эти оценки представлены в виде радиальной диаграммы, что позволяет пользователям быстро визуализировать и сравнить данные.

Радиальная диаграмма является эффективным способом визуализации многомерных данных, где каждая ось представляет собой одну из категорий, а расстояние от центра диаграммы отображает значение оценки качества жизни в данной категории. Это обеспечивает интуитивно понятное представление данных и помогает пользователям быстро сориентироваться в информации.

6.2.2 Бекенд

В процессе разработки бекенд-части нашего веб-приложения мы реализовали API-запрос и соответствующий сериализатор для обработки данных из базы данных Django и предоставления их в формате JSON для фронтенда.

Мы создали API запрос, который обрабатывает запросы от фронтенда и возвращает необходимые данные о качестве жизни в регионе Архангельску. Этот запрос является ключевым элементом взаимодействия между фронтендом и бекендом, обеспечивая передачу данных от сервера к клиенту.

Сериализатор отвечает за преобразование данных из базы данных Django в формат JSON, который будет передан клиентской части приложения. Мы разработали специальный сериализатор, который выбирает необходимые данные и предоставляет их в удобном для фронтенда формате.

Сериализатор обрабатывает данные из базы данных Django и предоставляет их в виде JSON файла, который затем отправляется на фронтенд. Этот JSON файл содержит информацию о качестве жизни в регионе Архангельску, включая оценки по различным категориям и другие важные данные.

6.2.3 Реализация парсеров через Celery и Redis

Для реализации парсеров данных социальных медиа мы воспользовались инструментами Celery и Redis для асинхронного выполнения задач и управления очередями.

Celery позволяет нам создавать и планировать асинхронные задачи, которые могут выполняться независимо от основного процесса приложения. Мы определили несколько задач для парсинга данных социальных медиа и настроили их выполнение через Celery.

Для хранения очередей задач и обмена сообщениями между различными процессами мы использовали Redis. Redis обеспечивает высокую производительность и надежность при работе с очередями задач, что делает его идеальным выбором для наших потребностей.

Мы создали задачи в Celery для парсинга данных из различных источников социальных медиа, таких как Вконтакте, Телеграм и новостные сайты. Каждая задача отвечает за сбор данных из конкретного источника и их обработку для дальнейшего анализа.

6.2.4 Разметка данных при помощи моделей глубокого обучения

Для автоматической разметки данных, полученных из социальных медиа, мы использовали модели глубокого обучения. Эти модели позволяют нам анализировать и классифицировать контент автоматически, что улучшает качество и точность данных, сохраняемых в нашей базе данных Django и представлено это на рисунке 10.



Рисунок 10 – Разметка спаршеных данных моделями

6.3 Контейнеризация веб приложения

Для обеспечения легкости развертывания и масштабирования нашего веб-приложения мы применили подход контейнеризации. Контейнеризация позволяет упаковать все компоненты приложения и их зависимости в единый исполняемый объект, называемый контейнером. Это обеспечивает консистентность среды выполнения и изолирует приложение от окружающей среды, что делает его переносимым и управляемым.

Мы использовали Docker - одну из наиболее популярных платформ для контейнеризации приложений. Docker предоставляет простой и эффективный способ упаковки, доставки и запуска приложений в контейнерах.

Мы создали Docker-контейнеры для фронтенд- и бекенд-частей нашего веб-приложения. В каждом контейнере упакованы все необходимые зависимости и компоненты приложения, включая код, библиотеки и настройки.

Для удобного управления множеством контейнеров мы использовали Docker Compose. Docker Compose позволяет определить и управлять многоконтейнерными приложениями с помощью простого YAML-файла конфигурации и представлено это на рисунке 11.

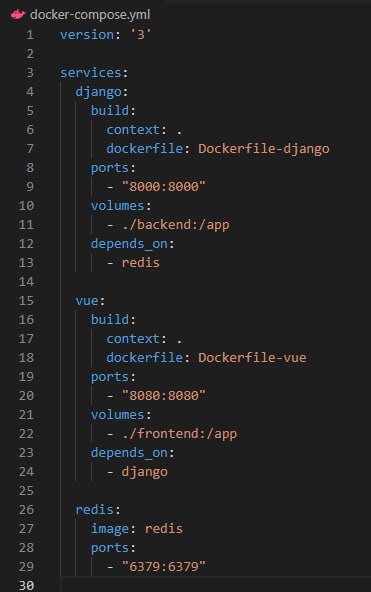


Рисунок 11 - YAML-файл Docker Compose

Контейнеризация веб-приложения обеспечивает ряд преимуществ:

1. Портабельность: Контейнеры могут быть легко переносимы между различными средами выполнения, что обеспечивает единое и надежное развертывание приложения.

2. Изолированность: Каждый контейнер изолирован от остальной среды, что обеспечивает надежную и безопасную работу приложения.

3. Масштабируемость: Контейнеры могут быть легко масштабируемы в зависимости от потребностей приложения, что обеспечивает гибкость и эффективность работы.

Контейнеризация веб-приложения является важным шагом для обеспечения его устойчивости, надежности и масштабируемости. Мы успешно применили Docker для упаковки и развертывания нашего приложения в контейнерах, что обеспечивает его гладкую и эффективную работу в любой среде выполнения.

Заключение

В ходе выполнения данной работы мы успешно реализовали комплексный проект, направленный на сбор, обработку и визуализацию данных из социальных медиа. Весь процесс был разделен на несколько этапов, каждый из которых вносил свой вклад в достижение общей цели - создание полноценного веб-приложения для визуализации и анализа данных.

Мы начали со сбора данных из различных источников социальных медиа, используя специальные парсеры. Этот этап позволил нам создать обширную базу данных, содержащую информацию о различных аспектах жизни региона Архангельску.

Далее мы разработали сервис для разметки этих данных - Narfu Toloka. Этот сервис позволил нам эффективно обрабатывать и классифицировать данные, что было важным шагом в подготовке данных к последующему анализу.

Следующим этапом была предобработка данных, включающая в себя очистку, преобразование и структурирование данных для подготовки к обучению моделей.

Затем мы обучили модели глубокого обучения для автоматической разметки данных. Это позволило нам автоматизировать процесс анализа данных и обеспечить высокую точность и эффективность в разметке.

Наконец, мы разработали веб-приложение для визуализации данных и автоматического парсинга новых данных, а также их автоматической разметки. Это приложение предоставляет пользователям удобный интерфейс для работы с данными, а также возможность получать актуальную информацию о качестве жизни в регионе Архангельску в режиме реального времени.

В итоге наш проект не только демонстрирует применение современных методов и технологий в области анализа данных, но и предоставляет практический инструмент для мониторинга и улучшения качества жизни в конкретном регионе. Надеемся, что наша работа будет полезна для различных групп интересов и способствует развитию области анализа данных и информационных технологий в целом.

Список использованных источников

1. VK API : сайт. – URL: https://dev.vk.com/guide (дата обращения: 20.01.2024)
2. Telegram API : сайт. – URL: https://core.telegram.org/ (дата обращения: 20.01.2024)
3. Django documentation : сайт. – URL: https://docs.djangoproject.com/en/5.0/ (дата обращения: 20.01.2024)
4. Htmlbook : сайт. – URL: https://htmlbook.ru/ (дата обращения: 20.01.2024)
5. Разметка данных в машинном обучении: процесс, разновидности и рекомендации : сайт. – URL: https://habr.com/ru/articles/678524/ (дата обращения: 20.01.2024)
6. РruT5, ruRoBERTa, ruBERT: как мы обучили серию моделей для русского языка : сайт. – URL: https://habr.com/ru/companies/sberbank/articles/567776/ (дата обращения: 20.01.2024)
7. Text Classification Using NLP : сайт. – URL: https://www.codersarts.com/post/text-classification-using-nlp (дата обращения: 20.01.2024)

Приложение А

(обязательное)

Листинг подготовки данных

Листинг А.1 – Подготовка классификации текстов по тональности

df = pd.read\_csv('/kaggle/input/kurswork1/test4.csv')

df=df.drop(["Time","Data", "Comments\_Count","Views","Rating","Count\_Positive\_Reactions","Count\_Negative\_Reactions","Reposts","Comment\_Text","Garbage","Healthcare","Housing\_and\_Public\_Utilities","Type", "Resource\_Name","id","Header","Education","Infrastructure","Culture","Environmental\_Conditions","Social\_Security","Politics","Safety","Availability\_of\_Goods\_and\_Services","Official\_Statements","Tourism","Facts"], axis = 1)

df['Sentiment'] = df['Positive'].map({True: 1, False: 0}) + \

df['Negative'].map({True: 2, False: 0}) + \

df['Neutral'].map({True: 0, False: 0})

df=df.drop(["Positive","Negative","Neutral"], axis=1)

column\_name='Sentiment'

# Assuming 'self.dataset' is a pandas DataFrame

if np.isnan(df[column\_name]).any():

print("There are NaN values in the dataset.")

else:

print("There are no NaN values in the dataset.")

import re

df = df.applymap(lambda x: re.sub(r'[^\w\s]', '', str(x)))

test\_size = 0.2

test\_rows = int(test\_size \* len(df))

random\_rows = np.random.choice(df.index, test\_rows, replace=False)

test\_df = df.loc[random\_rows]

train\_df = df.drop(random\_rows)

test\_df

train\_text = train\_df['Text']

train\_labels = train\_df['Sentiment']

test\_text = test\_df['Text']

test\_labels = test\_df['Sentiment']

Листинг А.2 – Подготовка классификации текстов по категориям

train\_df = pd.read\_csv('/kaggle/input/categories/categories\_train.csv')

test\_df = pd.read\_csv('/kaggle/input/categories/categories\_test.csv')

class\_to\_numeric = {

'Healthcare': 0,

'Housing\_and\_Public\_Utilities': 1,

'Education': 2,

'Infrastructure': 3,

'Culture': 4,

'Environmental\_Conditions': 5,

'Social\_Security': 6,

'Politics': 7,

'Safety': 8,

'Availability\_of\_Goods\_and\_Services': 9,

'Official\_Statements': 10,

'Tourism': 11,

'Facts': 12

}

# Преобразуйте колонку 'target' в числовые значения, используя отображение

train\_df['Target'] = train\_df['Target'].map(class\_to\_numeric)

test\_df['Target'] = test\_df['Target'].map(class\_to\_numeric)

train\_text = train\_df['Text']

train\_labels = train\_df['Target']

test\_text = test\_df['Text']

test\_labels = test\_df['Target']

Листинг А.3 – Подготовка классификации текстов по мусору

train\_df = pd.read\_csv('/kaggle/input/garbage-text-classification/garbage\_train.csv')

test\_df = pd.read\_csv('/kaggle/input/garbage-text-classification/garbage\_test.csv')

train\_text = train\_df['Text']

train\_labels = train\_df['Garbage']

test\_text = test\_df['Text']

test\_labels = test\_df['Garbage']

Приложение Б

(обязательное)

Листинг кода классификатора

Листинг Б.1 – Код классификатора

def seed\_all(seed\_value):

random.seed(seed\_value)

np.random.seed(seed\_value)

torch.manual\_seed(seed\_value)

if torch.cuda.is\_available():

torch.cuda.manual\_seed(seed\_value)

torch.cuda.manual\_seed\_all(seed\_value)

torch.backends.cudnn.benchmark = True

torch.backends.cudnn.deterministic = False

seed\_all(42)

model = BertForSequenceClassification.from\_pretrained('DeepPavlov/rubert-base-cased-sentence', num\_labels=13).to("cuda")

tokenizer = BertTokenizer.from\_pretrained('DeepPavlov/rubert-base-cased-sentence')

seq\_len\_train = [len(str(i).split()) for i in train\_df['Text']]

seq\_len\_test = [len(str(i).split()) for i in test\_df['Text']]

max\_seq\_len = max(max(seq\_len\_test), max(seq\_len\_train))

seq\_len\_train = [len(str(i).split()) for i in train\_df['Text']]

seq\_len\_test = [len(str(i).split()) for i in test\_df['Text']]

max\_seq\_len = max(max(seq\_len\_test), max(seq\_len\_train))

class Data(torch.utils.data.Dataset):

def \_\_init\_\_(self, encodings, labels):

self.encodings = encodings

self.labels = labels

def \_\_getitem\_\_(self, idx):

item = {k: torch.tensor(v[idx]) for k, v in self.encodings.items()}

item["labels"] = torch.tensor([self.labels[idx]])

return item

Продолжение листинга Б.1

def \_\_len\_\_(self):

return len(self.labels)

train\_dataset = Data(tokens\_train, train\_labels)

from sklearn.metrics import f1\_score, precision\_recall\_fscore\_support

def compute\_metrics(pred):

labels = pred.label\_ids

preds = pred.predictions.argmax(-1)

# Выберите подходящее значение для параметра average

average\_type = 'weighted'

f1 = f1\_score(labels, preds, average=average\_type)

return {'F1': f1}

training\_args = TrainingArguments(

output\_dir = '/kaggle/working/results', #Выходной каталог

num\_train\_epochs = 3, #Кол-во эпох для обучения

per\_device\_train\_batch\_size = 8, #Размер пакета для каждого устройства во время обучения

per\_device\_eval\_batch\_size = 8, #Размер пакета для каждого устройства во время валидации

weight\_decay =0.01, #Понижение весов

logging\_dir = '/kaggle/working/logs', #Каталог для хранения журналов

load\_best\_model\_at\_end = True, #Загружать ли лучшую модель после обучения

learning\_rate = 1e-5, #Скорость обучения

evaluation\_strategy ='epoch', #Валидация после каждой эпохи (можно сделать после конкретного кол-ва шагов)

logging\_strategy = 'epoch', #Логирование после каждой эпохи

save\_strategy = 'epoch', #Сохранение после каждой эпохи

save\_total\_limit = 1,

seed=21)

Окончание листинга Б.1

trainer = Trainer(model=model,

tokenizer = tokenizer,

args = training\_args,

train\_dataset = train\_dataset,

eval\_dataset = train\_dataset,

compute\_metrics = compute\_metrics)

trainer.train()

model\_path = "fine-tune-bert3"

model.save\_pretrained(model\_path)

tokenizer.save\_pretrained(model\_path)

def get\_prediction():

test\_pred = trainer.predict(test\_dataset)

labels = np.argmax(test\_pred.predictions, axis = -1)

return labels

pred = get\_prediction()

average\_type = 'weighted'

print(classification\_report(test\_labels, pred))

print(f1\_score(test\_labels, pred, average=average\_type))

Приложение В

(обязательное)

Листинг кода интеграции моделей

Листинг В.1 – Код интеграции моделей

from celery import Celery

from celery.schedules import crontab

import torch

from transformers import BertForSequenceClassification, BertTokenizer

# Путь к сохраненной модели

model\_path = "../fine-tune-bert\_tonality.pth"

# Загрузка модели

model = BertForSequenceClassification.from\_pretrained(

'DeepPavlov/rubert-base-cased-sentence',

num\_labels=4

)

model.load\_state\_dict(torch.load(model\_path))

# Токенизация текста

tokenizer = BertTokenizer.from\_pretrained('DeepPavlov/rubert-base-cased-sentence')

# Создание экземпляра Celery

celery = Celery('tasks', broker='redis://localhost:6379/0')

# Определение задачи для проверки мусора

@celery.task

def check\_garbage(text):

with torch.no\_grad():

outputs = model(\*\*inputs)

predicted\_class = torch.argmax(outputs.logits).item()

return is\_garbage

# Определение задачи для определения категории

Окончание листинга В.1

@celery.task

def determine\_category(text):

with torch.no\_grad():

outputs = model(\*\*inputs)

predicted\_class = torch.argmax(outputs.logits).item()

return category

# Определение задачи для определения тональности

@celery.task

def determine\_tonality(text):

with torch.no\_grad():

outputs = model(\*\*inputs)

predicted\_class = torch.argmax(outputs.logits).item()

return tonality

@celery.task

def process\_text(text):

is\_garbage = check\_garbage.delay(text).get()

category = determine\_category.delay(text).get()

tonality = determine\_tonality.delay(text).get()

# Настройка периодического выполнения задачи

@celery.on\_after\_configure.connect

def setup\_periodic\_tasks(sender, \*\*kwargs):

# Запуск задачи process\_text каждый день в полночь

sender.add\_periodic\_task(

crontab(hour=0, minute=0),

process\_text.s(parse\_text)

)

Сведения о самостоятельности выполнения работы

Работа «Разработка сервиса оценки привлекательности северного региона по данным социальных медиа на примере Архангельской области.»

«09» февраля 2024г. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Сивков И.Г.

(Подпись) (Фамилия, инициалы)

«09» февраля 2024г. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Скок А.С.

(Подпись) (Фамилия, инициалы)