AI

РЕАЛИЗАЦИЯ ПРОЕКТА МАШИННОГО ОБУЧЕНИЯ ОТ А ДО Я

на примере приложения для обобщения текста

АЛЕКСЕЙ МИХНИН

Алексей Михнин

Реализация проекта машинного обучения от А до Я на примере приложения для обобщения текста

http://www.litres.ru/pages/biblio_book/?art=69563596
SelfPub; 2023

Аннотация

Книга посвящена практической реализации проекта машинного обучения. Рассматривается весь жизненный цикл создания продукта на основе моделей машинного обучения, от формулировки бизнес-задачи до развертывания веб-приложения.Автор на конкретном кейсе демонстрирует процесс исследования проблемы, поиска алгоритмов, разработки и обучения AI моделей. Особое внимание уделяется вопросам проектирования кода и архитектуры, позволяющим создавать гибкие и масштабируемые системы искусственного интеллекта.Читатель получает ценные практические навыки по модульной разработке, тестированию, контейнеризации моделей и их интеграции через веб-интерфейсы. Книга содержит примеры кода и инструкции для создания собственных приложений машинного обучения.Это издание станет полезным как для

начинающих, так и для опытных разработчиков в области искусственного интеллекта.

Содержание

Алексей Михнин
Реализация проекта машинного обучения от А до Я на примере приложения для обобщения текста

Введение

В эпоху стремительного развития технологий искусственного интеллекта всё больше компаний и разработчиков пытаются применить машинное обучение в своих продуктах и проектах. Однако зачастую процесс создания работающего продукта на основе моделей машинного обучения представляет собой «черный ящик» для новичков в этой сфере.

Эта книга ставит своей целью максимально подробно и пошагово рассказать о том, как создать полноценный проект в сфере искусственного интеллекта – от исследования идеи до готового веб-приложения с моделью машинного обучения. Мы возьмем за основу конкретный проект по текстово-

му обобщению (рафинированию) – это процесс создания более короткой версии длинного текста или диалога и увидим, как он был реализован от начала и до конца.

Эта книга станет настоящей «библией» для всех, кто хочет разобраться в прикладном применении машинного обучения и понять весь процесс от А до Я.

После ее прочтения вы получите бесценные знания о том, как подходить к разработке подобных проектов, что позволит вам:

Структурировать код проекта с использованием передовых практик

Организовать процесс исследования и поиска решения

Разрабатывать и обучать эффективные модели машинного обучения

Создавать тренировочные и прогнозирующие конвейеры

Развертывать модели в виде готовых веб-приложений

Автоматизировать процесс непрерывной интеграции и доставки моделей

Книга содержит реальный код, примеры и шаги по созданию проекта от начала до конца. Это позволит вам не только изучить, но и применить на практике паттерны и подходы разработки проектов в сфере ИИ.

После прочтения вы сможете использовать полученные знания как шаблон для создания собственных приложений и сервисов с машинным обучением.

Начало проекта по

Data
Science

Бизнес-постановка – основа любого коммерческого проекта по Data science

В 90% случаев коммерческие проекты по Data science начинаются с бизнес-постановки от заказчика. Это означает, что заказчик четко определяет, какую задачу необходимо решить с помощью данных.

Бизнес-постановка включает в себя следующие элементы:

Цель проекта. Что заказчик хочет достичь с помощью данных?

Задачи проекта. Какие шаги необходимо предпринять для достижения цели?

Данные. Какие данные необходимы для выполнения проекта?

Ожидаемые результаты. Что заказчик ожидает получить в результате проекта?

Дано:

Заказчик обратился с потребностью автоматически суммаризировать большие объемы текста, в особенности длинные диалоги. Основная цель состояла в том, чтобы пользователи могли быстро понять основное содержание предоставленного текста без необходимости читать его полностью. Это особенно актуально для быстрого анализа новостей, длинных документов или корпоративных диалогов.

Кроме того, заказчик предоставил специфические данные

для обучения, чтобы модель лучше понимала и адаптировалась к уникальной специфике и структуре диалогов в компании заказчика.

Ожидание заказчика – Web API интерфейс для решения задач по суммаризации текста

Заказчик ожидает, что исполнитель предоставит Web API интерфейс для решения задач по суммаризации текста. Этот интерфейс должен соответствовать следующим требованиям:

Легкость использования. Интерфейс должен быть простым и понятным в использовании. Он должен быть доступен через стандартные методы HTTP, такие как POST, GET и PUT.

Производительность. Интерфейс должен быть производительным. Он должен обеспечивать быстрое и эффективное выполнение запросов.

Надежность. Интерфейс должен быть надежным. Он должен поддерживать высокую доступность и отказоустойчивость.

Цели заказчика:

Эффективность:

Сократить время, затрачиваемое на анализ и понимание больших объемов текста, предоставляя краткие и точные резюме.

Адаптация к специфике:

Улучшить качество и точность суммаризации, адаптируя

модель к уникальным особенностям и структуре диалогов в компании заказчика.

Интеграция:

Возможность легко интегрировать решение в существующие корпоративные системы для автоматизации процесса суммаризации.

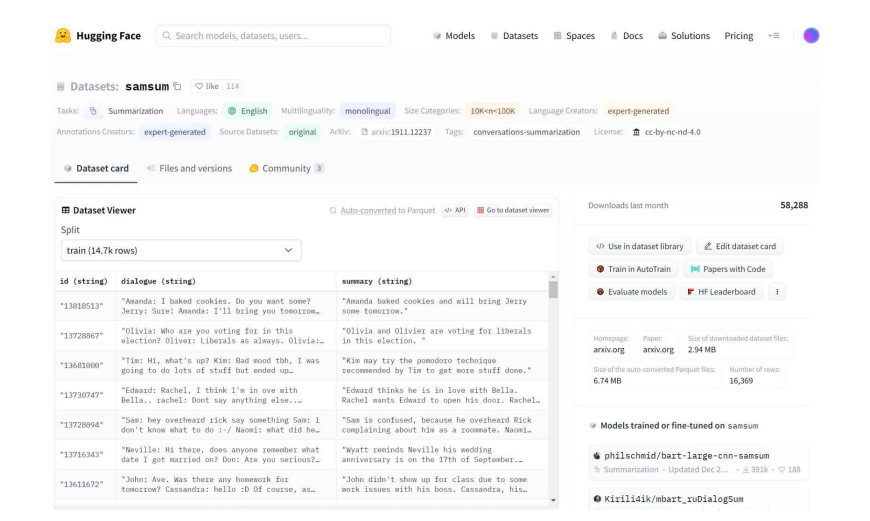
Улучшение взаимодействия:

Помочь сотрудникам быстрее и эффективнее взаимодействовать с информацией, улучшая таким образом коммуникации и принятие решений в компании.

Данные:

Для наглядности обсуждения примем ситуацию, где заказчик предоставил, среди прочего, размеченный датасет для дополнительного обучения нашей предстоящей модели.

В целях демонстрации, мы взяли обучающий датасет с платформы Hugging Face. После получения всей необходимой информации от заказчика, включая данные, исполнитель переходит к этапу прототипирования решения. Если прототип удовлетворяет требованиям заказчика, следуют действия по внедрению решения в рабочую среду. Данный процесс будет описан в деталях в нашей книге.


Hugging Face
Search models, datasets, users...
Models
Datasets
Spaces
Docs
Solutions
Pricing
Datasets: samsum
like 114
Tasks: Summarization
Languages: English
Multilinguality: monolingual
Size Categories: 10K<n<100K
Language Creators: expert-generated
Annotations Creators: expert-generated
Source Datasets: original
ArXiv: arxiv:1911.12237
Tags: conversations-summarization
License: cc-by-nc-nd-4.0
Dataset card
Files and versions
Community 3
Dataset Viewer
Auto-converted to Parquet
API
Go to dataset viewer
Split
train (14.7k rows)
id (string)
dialogue (string)
summary (string)
"13818513"
"Amanda: I baked cookies. Do you want some? Jerry: Sure! Amanda: I'll bring you tomorrow…
"Amanda baked cookies and will bring Jerry some tomorrow."
"13728867"
"Olivia: Who are you voting for in this election? Oliver: Liberals as always. Olivia:…
"Olivia and Olivier are voting for liberals in this election. "
"13681000"
"Tim: Hi, what's up? Kim: Bad mood tbh, I was going to do lots of stuff but ended up…
"Kim may try the pomodoro technique recommended by Tim to get more stuff done."
"13730747"
"Edward: Rachel, I think I'm in ove with Bella.. rachel: Dont say anything else.…
"Edward thinks he is in love with Bella. Rachel wants Edward to open his door. Rachel…
"13728894"
"Sam: hey overheard rick say something Sam: i don't know what to do :-/ Naomi: what did he…
"Sam is confused, because he overheard Rick complaining about him as a roommate. Naomi…
"13716343"
"Neville: Hi there, does anyone remember what date I got married on? Don: Are you serious?…
"Wyatt reminds Neville his wedding anniversary is on the 17th of September.…
"13611672"
"John: Ave. Was there any homework for tomorrow? Cassandra: hello :D Of course, as…
"John didn't show up for class due to some work issues with his boss. Cassandra, his…
Downloads last month
58,288
Use in dataset library
Edit dataset card
Train in AutoTrain
Papers with Code
Evaluate models
HF Leaderboard
Homepage:
arxiv.org
Paper:
arxiv.org
Size of downloaded dataset files:
2.94 MB
Size of the auto-converted Parquet files:
6.74 MB
Number of rows:
16,369
Models trained or fine-tuned on samsum
philschmid/bart-large-cnn-samsum
Summarization · Updated Dec 2… · 391k · 188
Kirili4ik/mbart_ruDialogSum

Прототип решения задачи по обобщению текста

Предположим, что исполнитель успешно справился с реализацией представленной бизнес-постановки. В качестве результата работы он предоставил прототип решения задачи по обобщению текста в формате Jupyter Notebook. (**см. Приложение №1**)

Основные этапы работы, представленные в файле Jupyter Notebook:

Подготовка рабочей среды:

Установка всех необходимых библиотек и пакетов, проверка доступности графического процессора для ускоренных вычислений.

Загрузка данных:

Импорт предоставленных заказчиком данных и их предварительная обработка.

Выбор и загрузка модели:

Выбрана модель PEGASUS из библиотеки Hugging Face's Transformers для задачи обобщения.

Дообучение модели:

Используя предоставленные заказчиком данные, произведено дообучение модели для лучшей адаптации к специфике диалогов заказчика.

Оценка качества:

Проведена валидация и оценка качества модели на отложенной выборке.

Демонстрация работы:

Представлены примеры обобщения различных текстов с использованием обученной модели.

Этот Jupyter Notebook служит как детальное руководство по реализации решения, так и демонстрацией его эффективности.

Приложение №1 Прототип по обобщению текста в формате Jupyter Notebook

Пошаговая инструкция по запуску в реализацию проекта

По итогам предоставления прототипа заказчиком принято решение о запуске данного решения в продакшен.

Это открывает новый этап работы для исполнителя.

Модульное кодирование:

Необходимо структурировать код из Jupyter Notebook, разделив его на модули и функции, что облегчит последующую интеграцию, тестирование и поддержку решения.

Создание Web-API интерфейса:

Разработка пользовательского интерфейса, который позволит конечным пользователям легко и удобно использовать решение для обобщения текстов.

Контейнеризация:

Все компоненты решения, включая зависимости, модель и интерфейс, необходимо упаковать в Docker-контейнер. Это обеспечит портативность, масштабируемость и надежность при развертывании решения.

Разворачивание контейнера на облачной инфраструктуре заказчика:

После тестирования и упаковки решения в Docker-контейнер, оно должно быть развернуто на облачной инфраструктуре заказчика, обеспечив тем самым доступность для

конечных пользователей.

Эти этапы являются ключевыми для успешного перехода от прототипа к полноценному продакшен-решению, способному обслуживать множество пользователей и интегрироваться с другими системами заказчика.

Шаг 1. Подготовка проекта

Подготовка проекта включает в себя ряд действий, направленных на настройку инфраструктуры и кода для обеспечения качественной и надежной разработки. Это важный этап в жизненном цикле проекта, который помогает избежать ошибок и сложностей на последующих этапах.

Все необходимые шаги, которые необходимо выполнить перед тем как приступить к модульному кодированию проекта, подробно со скриншотами кода, представлены в следующих разделах.

Настройка и клонирование репозитория GitHub на ПК

Первым делом в разработке любого проекта должно стать создание его «дома» – репозитория на GitHub. Эта платформа позволит нам не только хранить и версионировать код, но и настроить процесс непрерывной интеграции.

Мы задаем имя репозитория, совпадающее с названием нашего будущего проекта. Это поможет коллегам сразу понять его суть и назначение. Далее определяемся с уровнем доступа – сделать репозиторий публичным или приватным.

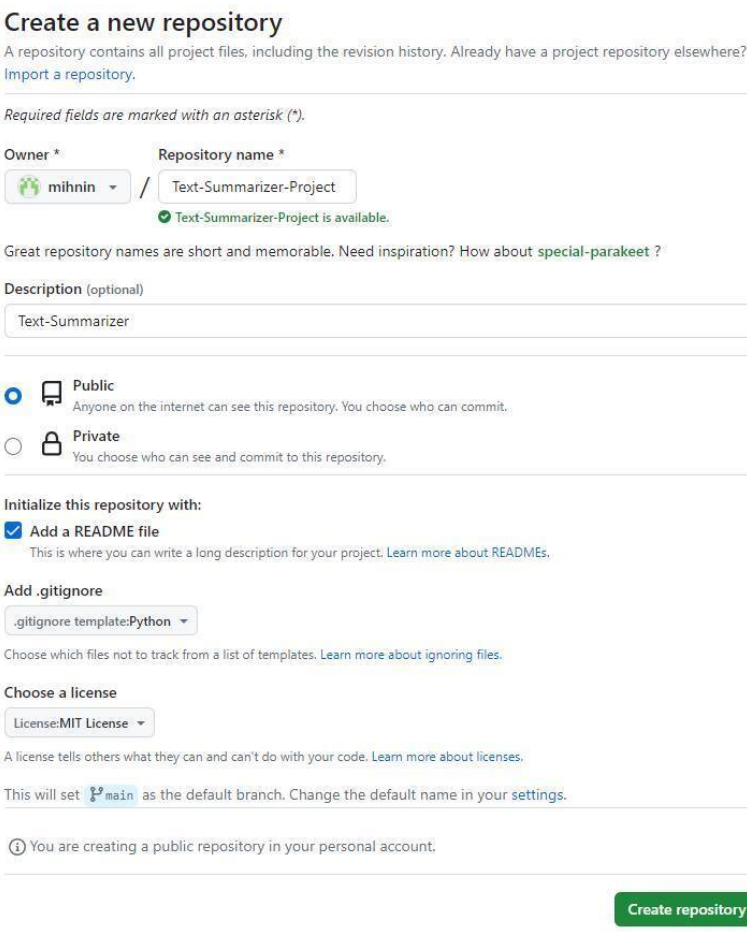
Как только репозиторий создан, добавляем в него файл README.md – это своего рода «паспорт» проекта с описанием его функционала и инструкциями по запуску.

Также важный шаг – добавление .gitignore и указание там Python как языка разработки. Это позволит исключить лишние промежуточные файлы из репозитория.

Завершающим аккордом станет выбор лицензии. Для открытых проектов отлично подходит лаконичная и ненавязчивая MIT – она позволит любому использовать код, указав авторство разработчиков.

Теперь у нашего проекта есть дом с просторными кодовыми хранилищами, настроен охранник в лице .gitignore и определены правила проживания благодаря выбранной лицензии.

Можно приступать к активной фазе – наполнению репозитория полезным кодом!



Получив доступ к репозиторию, созданному на GitHub, мы можем приступить к его клонированию – процессу создания полной локальной копии удаленного репозитория. Это позволит нам в дальнейшем работать с кодом на своем компьютере с последующей синхронизацией изменений обратно в удаленный репозиторий.

Для клонирования репозитория необходимо выполнить следующие действия:

Шаг1: перейти на страницу созданного репозитория на GitHub.

Шаг2: нажать на кнопку "Code", после чего скопировать предложенную ссылку в формате HTTPS. Эта ссылка указывает на расположение репозитория.

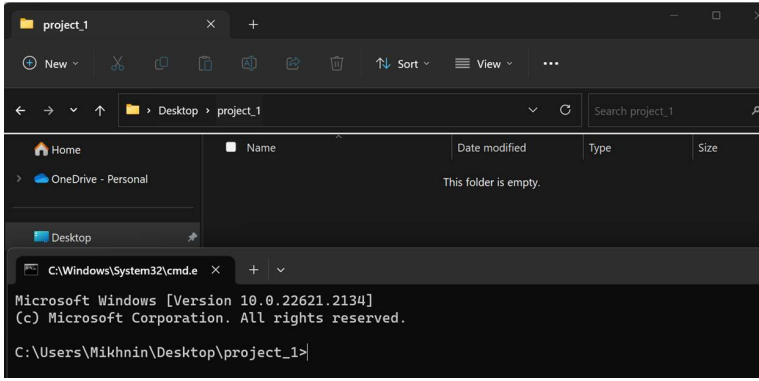
Шаг3: в директории на локальном компьютере, куда необходимо поместить клонируемый репозиторий, открыть окно терминала, набрав команду ***cmd*** *в адресной строке.*

Шаг4: выполнить команду git clone <ссылка>. Git использует указанную ссылку для скачивания всех файлов и данных репозитория.

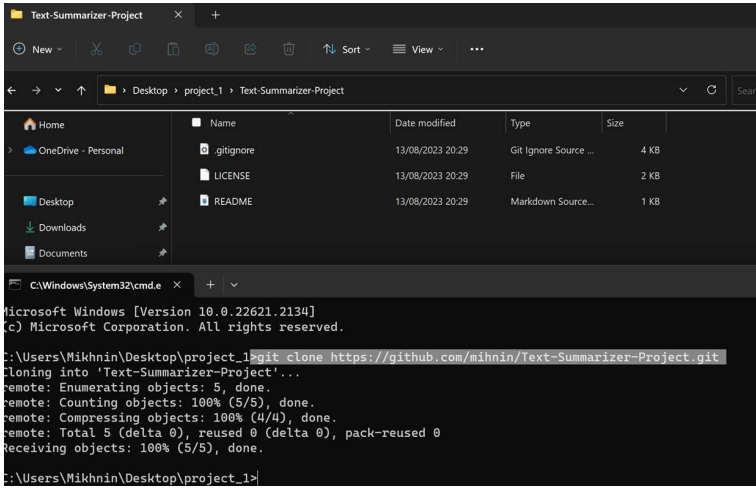
После завершения клонирования в выбранной директории появится полная копия репозитория со всеми файлами и версиями на вашем диске.

Теперь репозиторий готов к использованию в локальной разработке. Все изменения можно будет синхронизировать обратно в удаленный репозиторий с помощью команд git

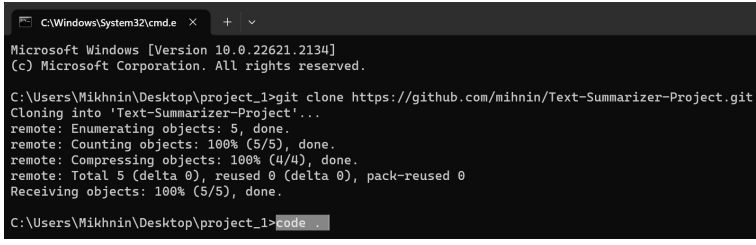
push и git pull.



Создали директорию project_1 на рабочем столе и открыли терминал в этой же директории

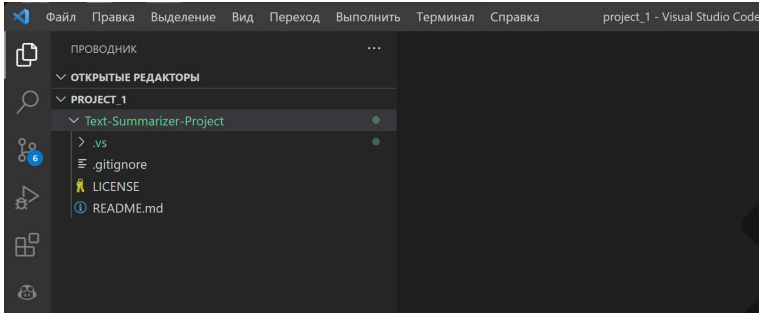


Клонируем ранее созданный репозиторий на GitHub на локальный компьютер. В результате копирования на компьютере создаётся новая папка с названием репозитория.



Запускаем VSC (Visual studio code) среды разработки –

для последующей работы над нашим проектом



Открыв VSC, в проводнике мы видим все наши файлы, которые мы клонировали с репозитория. Теперь приступаем к шагу 2: созданию скрипта на Python, содержащего будущую структуру папок и файлов нашего проекта.

Создание шаблона структуры папок и файлов

Text-Summarizer-Project > template.py > ...

You, 5 дней назад | 1 author (You)

```python
# Импортируем необходимые модули
import os
from pathlib import Path
import logging

# Настраиваем логирование
logging.basicConfig(level=logging.INFO, format='[%(asctime)s]: %(message)s:')

# Задаем имя проекта
project_name = "textSummarizer"

# Задаем список файлов, которые необходимо создать
list_of_files = [
    ".github/workflows/.gitkeep",
    f"src/{project_name}/__init__.py",
    f"src/{project_name}/conponents/__init__.py",      You, 5 дней назад • cr
    f"src/{project_name}/utils/__init__.py",
    f"src/{project_name}/utils/common.py",
    f"src/{project_name}/logging/__init__.py",
    f"src/{project_name}/config/__init__.py",
    f"src/{project_name}/config/configuration.py",
    f"src/{project_name}/pipeline/__init__.py",
    f"src/{project_name}/entity/__init__.py",
    f"src/{project_name}/constants/__init__.py",
    "config/config.yaml",
    "params.yaml",
    "app.py",
    "main.py",
    "Dockerfile",
    "requirements.txt",
    "setup.py",
    "research/trials.ipynb",
]

# Создаем файлы
for filepath in list_of_files:
    filepath = Path(filepath)
    filedir, filename = os.path.split(filepath)
```

Краткое пояснение кода:

Данный код является простым и эффективным способом создания списка папок и файлов, связанных с проектом, для их последующего автоматического создания.

Он легко читается и понимается, и может быть легко настроен для разных проектов, изменив значение переменной `project_name`.

`src` – это сокращение от английского слова "source" (исходный код). В контексте данного кода, `src` – это директория, в которой хранится исходный код проекта. В этой директории могут находиться поддиректории, содержащие модули и компоненты проекта. Обычно, исходный код проекта хранится в директории `src`, чтобы отделить его от других файлов и директорий, связанных с проектом, таких как документация, конфигурационные файлы, тесты и т.д.

Если в будущем необходимо досоздать новые папки и файлы, то необходимо отредактировать список `list_of_files`, добавив новые пути к файлам и папкам, которые необходимо создать. При этом необходимо убедиться, что пути к файлам и папкам заданы корректно и соответствуют структуре проекта.

Структура папок, представленная в данном коде, напоминает внутреннюю библиотеку, которую можно использовать в текущем проекте. В данном случае, проект разделен на модули и компоненты, которые хранятся в соответствую-

щих папках. Это позволяет легко организовать код проекта и упростить его поддержку и развитие.

Например, папка `conponents` содержит компоненты проекта, такие как модуль для предобработки данных или модуль для обучения модели.

Папка `utils` содержит утилиты, которые могут быть использованы в разных компонентах проекта.

Папка `config` содержит файлы конфигурации, которые используются для настройки параметров проекта.

Папка `pipeline` содержит модули, которые отвечают за обработку данных в конвейере.

Такая структура папок позволяет легко найти нужный модуль или компонент проекта, а также упрощает его тестирование и отладку. Кроме того, такая структура папок может быть использована для создания библиотеки, которую можно переиспользовать в других проектах.

Создание виртуального окружения

`conda create` – это команда менеджера пакетов Conda, которая создает новое виртуальное окружение и устанавливает в него пакеты и зависимости.

`-n project_1` – это опция команды `conda create`, которая задает имя нового виртуального окружения. В данном случае, имя виртуального окружения – `project_1`.

`python=3.11.4` – это опция команды `conda create`, которая указывает на установку пакета Python версии 3.11.4 в новое виртуальное окружение.

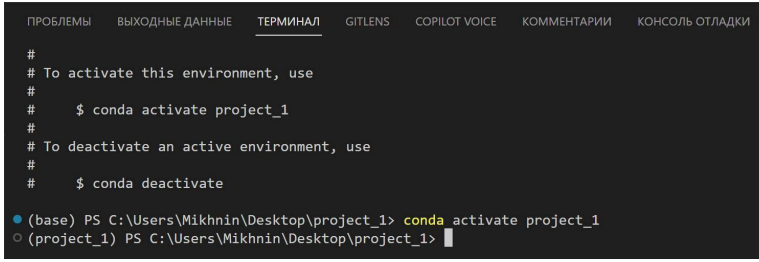
`-y` – это опция команды `conda create`, которая указывает на автоматическое подтверждение установки без запроса подтверждения пользователя.

Таким образом, данная команда создает новое виртуальное окружение с именем `project_1` и устанавливает в него Python версии 3.10с помощью менеджера пакетов Conda.

Опция `-y` указывает на автоматическое подтверждение установки без запроса подтверждения пользователя.

В результате выполнения этой команды будет создано новое виртуальное окружение, которое можно активировать с помощью команды `conda activate project_1`.

После активации виртуального окружения можно устанавливать необходимые пакеты и зависимости для проекта, не затрагивая другие проекты и системные настройки.



Создание структуры шаблона папок и файлов

Данный код является последовательностью команд, которые выполняются в командной строке операционной системы Windows.

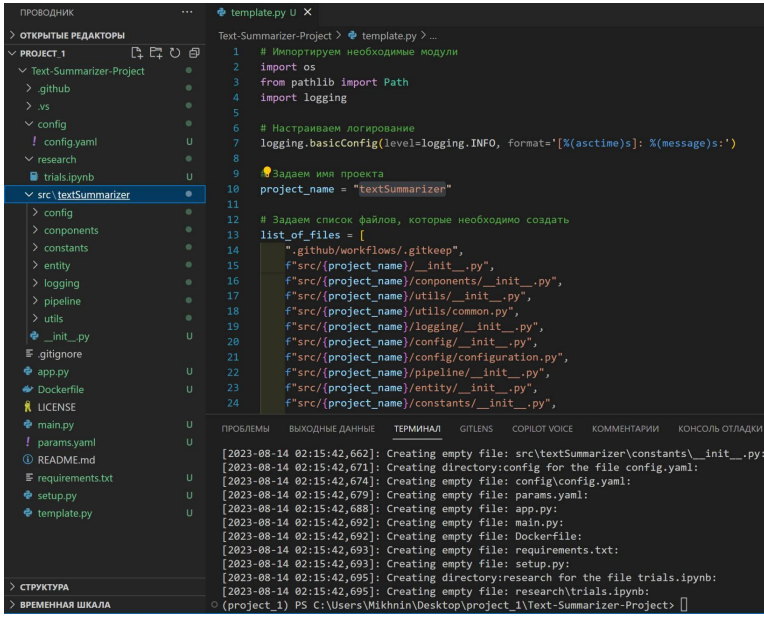
Первая команда `cd Text-Summarizer-Project` переходит в директорию `Text-Summarizer-Project`, которая находится в текущей директории `project_1`.

Затем выполняется команда `python template.py`, которая запускает скрипт `template.py` на языке Python.

В данном случае, скрипт `template.py` содержит код для работы с модулем по автоматическому созданию краткого содержания текста (text summarization).

При выполнении данной команды в командной строке будет запущен интерпретатор Python, который выполнит код из файла `template.py`.

В результате выполнения скрипта будет создано краткое содержание текста, которое будет выведено в консоль.



Итого: Шаблон нашего проекта создан. Шаблон содержит все необходимые папки и файлы, которые на последующих шагах мы будем заполнять.

Лог в терминале – содержит информацию о создании файлов и директорий в проекте. Каждая строка лога содержит дату и время создания файла или директории, а также путь к созданному файлу или директории.

Например, первая строка лога `[2023-08-14 02:15:42,662]: Creating empty file: src\textSummarizer\constants\__init__.py:`

указывает на создание пустого файла `__init__.py` в директории `constants`, которая находится в директории `textSummarizer`, которая в свою очередь находится в директории `src`.

Аналогично, остальные строки лога указывают на создание пустых файлов и директорий в проекте. Например, строка `[2023-08-14 02:15:42,674]: Creating directory:config for the file config.yaml:` указывает на создание директории `config`, в которой будет создан файл `config.yaml`.

Таким образом, логирование создания файлов и директорий позволяет отслеживать процесс создания файлов и директорий в проекте и выявлять возможные проблемы в процессе создания.

Конец ознакомительного фрагмента.