**Документация к коду**

**Общая информация**

**Название:** CopyDetector.

**Описание:** Решение, предназначенное для поиска смысловых копий изображения**.**

**Используемые библиотеки:**

* pillow: библиотека для обработки изображений в Python, основанная на проекте PIL
* transformers: библиотека для работы с моделями трансформеров
* torch: основная библиотека для машинного обучения и глубокого обучения
* numpy: основная библиотека для работы с многомерными массивами и матрицами
* pandas: библиотека для обработки и анализа данных
* sentence\_transformers: библиотека для создания и использования трансформеров на уровне предложений
* faiss-cpu: библиотека для эффективного поиска по векторным пространствам
* flask: микрофреймворк для создания веб-приложений

**Информация по коду**

**\_fit\_tusur\_func.py**

**Описание**

Код для обработки изображений с последующим хранением и поиском по базе данных с описаниями изображений.

**Константы**

**device –** определяет, будет ли использование GPU для модели (если доступно).

**db\_name –** название базы данных SQLite, в которой хранятся изображения и их описания.

**Основные функции**

1. **generate\_caption(image\_path):**

* Принимает путь к изображению, открывает его, конвертирует в формат RGB и использует модель BLIP для генерации описания изображения. Возвращает описание в виде строки.

1. **add\_image\_to\_db(image\_path):**

* Генерирует описание изображения с помощью generate\_caption().
* Извлекает имя изображения из пути.
* Добавляет описание и информацию о файле в базу данных SQLite в таблицу images.

1. **initialize\_db(db\_name):**

* Подключается к базе данных SQLite.
* Создает таблицу images (если она еще не существует) с полями для идентификатора, описания, источника и имени изображения.

1. **load\_data\_from\_db(cursor):**

* Загружает все данные (описания, источник и имя изображений) из базы данных.

1. **filter\_data\_by\_phrase(cursor, search\_phrase):**

* Выполняет запрос к базе данных для поиска записей, где описание изображения содержит определенную фразу.

1. **create\_embeddings\_and\_index(descriptions, model):**

* Принимает список описаний и использует модель для создания эмбеддингов этих описаний.
* Нормализует эмбеддинги и создает индекс FAISS для эффективного поиска по этим эмбеддингам.

1. **add\_similar\_results(search\_phrase, model, index, descriptions, source\_files, image\_names, filtered\_data):**

* Добавляет в список filtered\_data дополнительные результаты, если их недостаточно, запрашивая наиболее похожие изображения из индекса FAISS.

1. **search\_top\_20\_similar(search\_phrase, db\_name='image\_data.db'):**

* Основная функция поиска.
* Загружает данные из базы данных, фильтрует их по поисковой фразе, генерирует эмбеддинги для описаний и выполняет поиск похожих изображений.
* Если найдено менее 40 результатов, добавляет дополнительные похожие изображения с помощью индекса FAISS.
* Возвращает топ-20 наиболее похожих результатов, отсортированных по схожести, в виде DataFrame.

**Пояснение к структуре кода**

* Работа с изображениями: Код использует модель BLIP для автоматической генерации текстовых описаний изображений. Это позволяет добавить изображения в базу данных вместе с описанием, не требуя ручной работы.
* SQLite: Все изображения и их описания хранятся в базе данных SQLite, что позволяет хранить большие объемы данных и выполнять быстрые запросы для поиска изображений по описанию.
* Индексы и FAISS: Для эффективного поиска похожих изображений используется библиотека FAISS. Это ускоряет поиск похожих изображений по векторным эмбеддингам и позволяет расширять результаты поиска с помощью схожих данных.
* Sentence Embeddings: Используется модель sentence-transformers для создания эмбеддингов текстовых описаний. Это необходимо для вычисления схожести между фразами и изображениями.

**Потоки данных**

* Изображение загружается и передается в модель для генерации описания.
* Описание добавляется в базу данных вместе с именем изображения и исходной папкой.
* Для поиска по фразе генерируются эмбеддинги описаний изображений и выполняется поиск по схожести. Если данных меньше 40, добавляются дополнительные похожие результаты с помощью FAISS.
* Результаты сортируются по схожести и возвращаются в виде DataFrame.

**Пример использования**

Для поиска 20 наиболее похожих изображений по введенной фразе:

results = search\_top\_20\_similar(‘dog’) # Ищем изображения с описанием, содержащим слово 'пес'

print(results)

Результатом будет таблица с 20 наиболее похожими изображениями и их описаниями, отсортированными по схожести с запросом.

**App.py**

**Описание**

Код для веб-приложение на Flask, которое позволяет пользователям загружать изображения или вводить текст для поиска схожих изображений. Он включает два основных функционала: генерацию описаний изображений с использованием модели и поиск схожих изображений по текстовому запросу или загруженному изображению.

#### **Константы**

* **UPLOAD\_FOLDER**: переменная для хранения пути, куда будут загружаться изображения. Папка называется uploads.
* **app.secret\_key**: используется для защиты сессий и флеш-сообщений Flask. Это случайный секретный ключ, который должен быть уникальным для каждого приложения.

#### **Маршрут** / **(Главная страница)**

Этот маршрут обрабатывает как GET-запросы (для рендеринга страницы), так и POST-запросы (для обработки отправленных данных):

* **GET-запрос**: просто рендерит страницу, показывая список результатов (если они есть).
* **POST-запрос**: выполняет следующие шаги в зависимости от данных, отправленных пользователем.

#### **Логика POST-запроса**

1. **Если изображение загружено**:
   * Изображение сохраняется в папку uploads.
   * Для загруженного изображения вызывается функция generate\_caption(image\_path), которая генерирует описание изображения.
   * Если описание создано успешно, оно отображается в сообщении пользователю.
   * Затем вызывается функция search\_top\_20\_similar(description), которая ищет 20 схожих изображений по описанию изображения.
   * Если схожие изображения найдены, они конвертируются в список словарей для отображения в HTML-шаблоне.
2. **Если введен текст для поиска**:
   * Текст из формы поиска передается в функцию search\_top\_20\_similar(search\_text), которая ищет схожие изображения по введенному запросу.
   * Результаты также отображаются в виде списка словарей, если изображения найдены.
3. **Если не выбрано изображение или не введен текст**:
   * Отображается предупреждающее сообщение, что необходимо выбрать изображение или ввести текст.

#### **HTML-шаблон** index.html

* В render\_template("index.html", results=results, search\_text=search\_text) передаются:
  + results: список найденных схожих изображений, который будет отображен в шаблоне.
  + search\_text: текст, введенный в форму поиска, для отображения в поле поиска.

**Флеш-сообщения**

* **flash** используется для вывода сообщений пользователю в зависимости от результата:
  + **"success"**: успешная генерация описания.
  + **"error"**: ошибка при создании описания или поиске.
  + **"info"**: информация о результатах поиска.
  + **"warning"**: предупреждение, если не выбрано изображение или не введен текст.

### Структура приложения

1. **Загрузка изображений**:
   * Пользователь может загрузить изображение через форму на главной странице. Это изображение сохраняется в папке uploads, и для него генерируется описание с помощью модели.
2. **Поиск схожих изображений**:
   * Пользователь может ввести текстовый запрос, и приложение выполнит поиск схожих изображений по этому запросу.
   * Также можно загрузить изображение, для которого будет выполнен поиск схожих изображений.
3. **Обработка результатов**:
   * Найденные схожие изображения (по тексту или по изображению) отображаются в виде списка на веб-странице.
   * При успешных или неудачных операциях отображаются флеш-сообщения, информирующие пользователя о текущем состоянии.

**Script.js**

**Описание**

Код обеспечивает взаимодействие пользовательского интерфейса веб-страницы с сервером для загрузки изображений, генерации описаний и поиска похожих изображений. Он выполняет несколько функций, включая обработку событий на кнопках, управление элементами интерфейса, загрузку и показ результатов.

#### **Обработка событий**

* **Отслеживание выбора файла**:
  + **fileInput.addEventListener('change')**: при выборе файла запускается функция, которая создает URL для выбранного файла и устанавливает его как источник изображения для предпросмотра.
* **Перехват отправки формы**:
  + **uploadForm.addEventListener('submit')**: при отправке формы загружается индикатор и затеняется блок с результатами, пока не завершится обработка.
* **Кнопка generateCaptionButton**:
  + Отправляет изображение на сервер для генерации описания.
  + **Метод**: делает POST-запрос к маршруту /generate\_caption с файлом изображения в formData.
  + **Ответ**: сервер возвращает описание изображения и похожие результаты, которые выводятся на странице. Путь к каждому изображению добавляется в imagePaths для использования при загрузке zip-файла.
* **Кнопка searchSimilarButton**:
  + Выполняет текстовый поиск похожих изображений.
  + **Метод**: делает POST-запрос к маршруту /search\_similar с текстом запроса в formData.
  + **Ответ**: сервер возвращает список схожих изображений, которые отображаются на странице. Пути изображений сохраняются для последующего скачивания.
* **Кнопка downloadZipButton**:
  + Скачивает zip-файл с результатами.
  + **Метод**: делает POST-запрос к маршруту /download\_zip с путями изображений в формате JSON.
  + **Ответ**: получает zip-файл с изображениями. При клике на скрытую ссылку файл скачивается на устройство пользователя.

#### **Функции**

* **displayImages(images)**:
  + Обрабатывает и отображает изображения, полученные с сервера.
  + Очищает контейнер результатов, затем создает HTML-элементы для каждого изображения.
  + Каждое изображение сопровождается описанием.
  + Убирает индикатор загрузки и возвращает прозрачность для блока.

#### **Пути и маршруты**

* **/generate\_caption**: принимает изображение и возвращает его описание и похожие изображения.
* **/search\_similar**: принимает текст и возвращает изображения, схожие по значению с этим текстом.
* **/download\_zip**: принимает массив путей изображений, создает zip-файл и возвращает его для скачивания.

**Index.html**

**Описание**

Код представляет собой веб-страницу для поиска изображений по схожести, используя как изображение, так и текст, и содержит элементы интерфейса, которые позволяют пользователю загружать файлы, просматривать результаты, скачивать архивы с изображениями, а также видеть информацию о проекте и команде.

**Заголовок и подключение ресурсов**

* Включает настройки HTML-документа, заголовок страницы и ссылки на внешние файлы стилей и скриптов:
  + **<link rel="stylesheet" href="../static/css/style.css">**: подключает CSS-файл для стилизации страницы.
  + **<script src="..."></script>**: подключает библиотеки JSZip и FileSaver.js для работы с zip-файлами, а также пользовательский скрипт script.js.

**Основной контейнер для формы и пояснений**

* **<div class="alert-info">**: выводит краткое описание двух доступных режимов поиска изображений.
* **Форма загрузки**:
  + Элемент <form id="uploadForm"> предоставляет поля для выбора файла и ввода текста, который используется для поиска изображений.
  + Поля для **загрузки изображения** и **ввода текста** включены с инструкциями и атрибутами доступности.
  + Кнопка **Поиск** отправляет форму на сервер для обработки на маршруте /.

**Предварительный просмотр загружаемого изображения**:

* **<div id="imagePreview">**: отображает блок предпросмотра изображения, если оно выбрано пользователем, позволяя увидеть загруженное изображение перед отправкой.

**Сообщения о результатах и оповещения**

* **<div class="message" id="message">**: выводит сообщения (успех, ошибка и т.д.) с использованием механизма Flask flash для отображения статуса загрузки или поиска.

**Результаты поиска**

* **<div class="results">**: отображает список найденных изображений, сгенерированный сервером по результатам поиска.
* Используется **{% for result in results %}** для отображения каждого результата в виде изображения с описанием (используется серверный шаблонизатор Jinja2).

**Кнопки действий**

* **<button id="downloadZip">**: кнопка для скачивания найденных изображений в виде zip-архива.
* **<button id="addData">**: кнопка для добавления новых данных (предполагает дополнительную функциональность, не описанную в коде).

**Футер с информацией о проекте и команде**

* **Раздел информации о кейсе**:
  + Содержит название проекта и контактную информацию.
* **Информация о команде**:
  + Представлены участники команды с именами и контактной информацией.

**Подключение скриптов**

* Скрипты для работы с zip-файлами и пользовательские скрипты (script.js) для обработки событий на странице (например, обработка кнопок и загрузка изображений).

**Информация по работе решения**

**Описание работы**

Это решение представляет собой полноценное веб-приложение для поиска смысловых копий изображений на основе загруженного изображения или введенного текста. Оно использует Python (Flask) на серверной стороне для обработки запросов и предоставления результатов поиска, а также JavaScript на клиентской стороне для управления пользовательским интерфейсом.