**Техническое задание на разработку системы VideoRAG**

**1. Общие сведения**

**Название проекта:** VideoRAG – Система поиска сегментов видео с использованием CLIP и FAISS

**Цель проекта:**  
Создать автоматизированную систему для семантического поиска интересующих сегментов в видео по изображению-запросу. Система должна извлекать кадры из видео, вычислять их эмбеддинги с помощью модели CLIP, индексировать векторы с помощью FAISS, проводить поиск наиболее похожих кадров и на основе найденных результатов вырезать клипы из исходного видео.

**2. Область применения**

Проект предназначен для:

* Поиска сцен или персонажей в видео.
* Автоматического редактирования и создания клипов из длинных видео.
* Анализа и категоризации видеоконтента по семантическому содержимому.
* Интеграции в системы видеонаблюдения, медиаархивы и аналитические платформы.

**3. Функциональные требования**

**3.1. Подключение и организация данных**

* **Монтирование Google Drive:**
  + Система должна монтировать Google Drive для доступа к исходным данным и сохранения результатов.
* **Создание структуры каталогов:**  
  При инициализации проекта автоматически создаются следующие каталоги:
  + raw\_videos/ – для хранения исходных видео (например, example.mp4).
  + extracted\_frames/ – для сохранения извлечённых кадров (каждое видео имеет отдельную папку с кадрами и CSV-файл с временными метками).
  + features/ – для хранения эмбеддингов (PyTorch-тензоров) и меток времени (например, example\_features.pt и example\_timestamps.pt).
  + index/ – для сохранения FAISS-индекса (например, example.faiss).
  + query\_images/ – для изображений-запросов (например, query.jpg).
  + processed\_videos/ – для сохранения вырезанных сегментов видео.
  + tmp/ – для временных файлов (например, сконвертированных видео).

**3.2. Обработка видео**

* **Конвертация видео:**
  + Если исходное видео не в формате MP4, система должна сконвертировать его с помощью ffmpeg.
* **Извлечение кадров:**
  + С использованием библиотеки decord из видео извлекаются кадры с адаптивной частотой (определяемой длительностью видео).
  + Кадры сохраняются с именами вида frame\_XXXXXX.jpg.
  + Генерируется CSV-файл (timestamps.csv) с временными метками каждого кадра (формат HH:MM:SS.mmm).

**3.3. Векторизация кадров**

* **Используемая модель:**
  + Модель CLIP (ViT-B/32) используется для получения эмбеддингов кадров.
* **Процесс векторизации:**
  + Кадры проходят стандартный препроцессинг (ресайз, нормализация) согласно требованиям CLIP.
  + Вычисление эмбеддингов выполняется пакетами (batch processing) с использованием GPU.
  + При необходимости используется half precision (float16) через torch.cuda.amp.autocast() для экономии памяти.
  + Результаты сохраняются в файлы:
    - example\_features.pt – тензор эмбеддингов размером [N, D], где *N* — число кадров, а *D* — размерность эмбеддинга.
    - example\_timestamps.pt – метки времени для каждого кадра.

**3.4. Построение индекса FAISS**

* **Построение индекса:**
  + Из эмбеддингов создаётся FAISS-индекс (например, IndexFlatIP) после L2-нормализации векторов.
  + Индекс сохраняется в файл example.faiss для последующего быстрого поиска.

**3.5. Поиск по изображению-запросу**

* **Векторизация запроса:**
  + Изображение-запрос (например, query.jpg) обрабатывается той же моделью CLIP для получения эмбеддинга.
* **Поиск похожих кадров:**
  + Эмбеддинг запроса сравнивается с эмбеддингами из FAISS-индекса.
  + Возвращается топ-N (например, 5) наиболее похожих кадров с их временными метками и коэффициентами сходства.

**3.6. Вырезание сегментов видео**

* **Автоматическое вырезание клипов:**
  + На основе временных меток найденных кадров с помощью ffmpeg вырезаются сегменты видео (например, по 10 секунд).
  + Клипы сохраняются в каталоге processed\_videos для дальнейшего использования.

**4. Нефункциональные требования**

**4.1. Производительность**

* **Скорость обработки:**
  + Обработка видео длительностью до 1 часа должна укладываться в разумные рамки (например, до 30 минут при оптимальных настройках).
* **Использование ресурсов:**
  + Использование GPU (например, NVIDIA T4 в Google Colab) для ускорения обработки и векторизации.

**4.2. Масштабируемость**

* Возможность обработки нескольких видео.
* Настройка частоты извлечения кадров (FPS) в зависимости от длительности видео.

**4.3. Надёжность и обработка ошибок**

* Обработка ошибок при нехватке памяти, ошибках конвертации видео, проблемах при поиске.
* Логирование и информирование пользователя о возникших проблемах.

**4.4. Удобство использования**

* Автоматическая генерация всех необходимых каталогов.
* Интерактивная визуализация результатов поиска.
* Генерация текстового отчёта с временными метками и коэффициентами сходства.
* Возможность автоматического вырезания клипов по найденным результатам.

**5. Окружение и зависимости**

* **Платформа:** Google Colab с активированным GPU (например, NVIDIA T4)
* **Язык:** Python 3.x
* **Основные библиотеки:**
  + torch, torchvision
  + faiss-cpu
  + decord
  + clip
  + pandas
  + matplotlib
  + tqdm
  + ffmpeg (через apt)
* **Хранение данных:** Google Drive (смонтированное через Colab)

**6. Входные и выходные данные**

**6.1. Входные данные**

* **Видео:** Файл (например, example.mp4) размещается в каталоге raw\_videos.
* **Изображение-запрос:** Файл (например, query.jpg) размещается в каталоге query\_images.

**6.2. Выходные данные**

* **Извлечённые кадры и метки времени:** Каталог extracted\_frames содержит изображения и CSV с временными метками.
* **Эмбеддинги и метки времени:** Файлы example\_features.pt и example\_timestamps.pt в каталоге features.
* **FAISS-индекс:** Файл example.faiss в каталоге index.
* **Вырезанные сегменты видео:** Клипы сохраняются в каталоге processed\_videos.
* **Отчёт по поиску:** Текстовый вывод с временными метками и коэффициентами сходства найденных кадров.

**7. Этапы разработки и тестирования**

1. **Настройка окружения:**
   * Монтирование Google Drive, создание файловой структуры, установка необходимых библиотек.
2. **Разработка модулей:**
   * Модуль извлечения кадров из видео.
   * Модуль векторизации кадров с использованием CLIP.
   * Модуль построения FAISS-индекса.
   * Модуль поиска по изображению-запросу.
   * Модуль вырезания сегментов видео.
3. **Интеграционное тестирование:**
   * Тестирование на коротком видео (1-2 минуты) для проверки корректности работы всех этапов.
4. **Оптимизация:**
   * Настройка параметров обработки (batch size, использование float16, FPS).
   * Обработка ошибок и оптимизация использования памяти.
5. **Документация:**
   * Составление README.md и технического задания для пользователей и разработчиков.

**8. Критерии приемки**

* **Корректное извлечение кадров:** Все кадры и их временные метки корректно извлекаются из видео.
* **Точность векторизации:** Эмбеддинги кадров получаются с использованием модели CLIP и сохраняются в требуемом формате.
* **Работа FAISS-индекса:** Индекс строится и обеспечивает быстрый поиск по векторным представлениям.
* **Поиск по запросу:** Система возвращает корректные временные метки и коэффициенты сходства для изображения-запроса.
* **Вырезание клипов:** На основе найденных временных меток автоматически вырезаются видео-сегменты заданной длительности.
* **Интеграция и производительность:** Система работает в Google Colab с использованием GPU, обработка происходит в пределах установленных временных рамок, а логирование ошибок информирует пользователя о возможных проблемах.