МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
Российской Федерации
федеральное государственное автономное
образовательное учреждение высшего образования
«Самарский национальный исследовательский университет
имени академика С.П. Королева»

(Самарский университет)

Институт информатики, математики и электроники
Кафедра технической кибернетики

Отчет

по лабораторной работе № 2

«Применение n8n для реализации пайплайна обработки видео с использованием LLM моделей»

по дисциплине

«Инженерия данных»

Обучающийся: Фролова А. С.

6231-010402D

Преподаватель: Паринger Р. А.

Самара 2025

1. ***Архитектура:***

Реализован пайплан для автоматической обработки видео, поступающего в Telegram Bot с последующей загрузкой, извлечением аудио, распознаванием речи, переводом субтитром оригинального языка, формирование файлов субтитров (SRT) и добавлением их в исходное видео. В задании было представлено описание пайплайна. Описание по шагам:

1) Опрашиваем нашего телеграм бота, узнаем, есть ли новое сообщение. Значение offset предотвращает повторное получение сообщения. Получили сообщение? Идем дальше, определяем его тип.
2) Если в сообщении ссылка – скачиваем видео файл по url, если файл, то запрашиваем путь и скачиваем бинарные данные. Таким образом получаем данные входного видео, которое мы сохраняем, чтобы потом добавить субтитры.
3) Генерируем английские субтитры при помощи аудиодорожки и сервиса, реализованного через ffmpeg и auto_subtitle, создающий файл SRT, возвращаемый в данные.
4) Далее мы разбиваем файл SRT с субтитрами на объекты с тайм кодом и текстом, а также индексом, то есть каждый становится отдельным объектом.
5) Перебираем объекты, переводим через Hugging Face API и модель Helsinki-NLP opus-mt-en-ru, но можно настроить на другую, просто поменяв ссылку (выбор обусловлен нахождением данной модели на Trending в Hugging Face, когда ищем модели для переводов: https://huggingface.co/models?pipeline_tag=translation). Потом объединяем оригинальный объект и результат перевода, переводим в удобный вид.
6) Собираем итоговый SRT на русском языке из пар блоков, сохраняем его локально.
7) Встраиваем субтитры в видео через execute command с использованием ffmpeg и с указанием расположения субтитров, начального видео, получаем итоговое видео, которое сохраняем.
8) Отправляем результат пользователю (готовое видео с русскими субтитрами)
9) Очищаем временные файл, такие как: srt, финальное и начальное видео.

Инструменты и почему именно они:

- n8n Workflow (последняя версия): оркестрация триггеров, работаем с логикой и узлами, а также бинарными данными (https://n8n.io/).
- LLM переводчик (Hugging Face API): используем для перевода сегментов субтитров оригинального видео (https://huggingface.co/docs/inference-providers/tasks/index).
- ffmpeg: извлечение аудио, встраивание переведенных субтитров обратно в видео.
- auto_subtitle — генерация SRT (STT)., удобно для быстрого получения субтитров; имеет как локальные, так и контейнерные варианты (скачиваем через requirements).
- Telegram Bot API: удобная выдача сообщений, прием и отправка видео.
- Базовые библиотеки для реализации.

Главная причина – требование для выполнения лабораторной. n8n имеет удобный визуальный стиль и контроль, однако сложно для localhost с учетом особенностей, например, Telegram Trigger. ffmpeg – классический выбор, а auto_subtitle и LLM в виде API дали гибкость для обработки, GPU возможности, удаленное API это приятное легковесное упрощение.

Вход: mp4 файлы, которые загружает пользователь в Telegram, или же ссылка на видео.

Выход: готовое видео mp4 с русскими субтитрами, полученными с помощью нейронки.

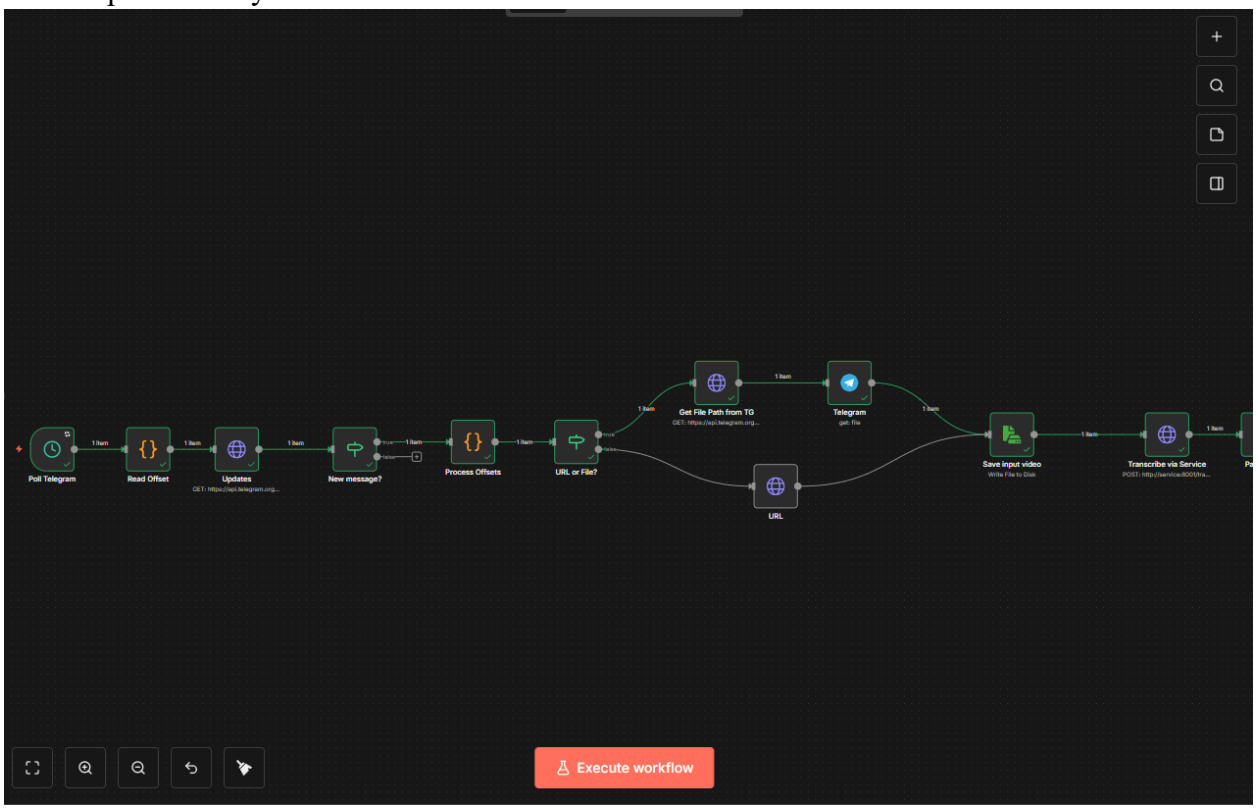
2. ***Качество данных: какие проверки качества данных реализованы и возможные точки сбоя.***

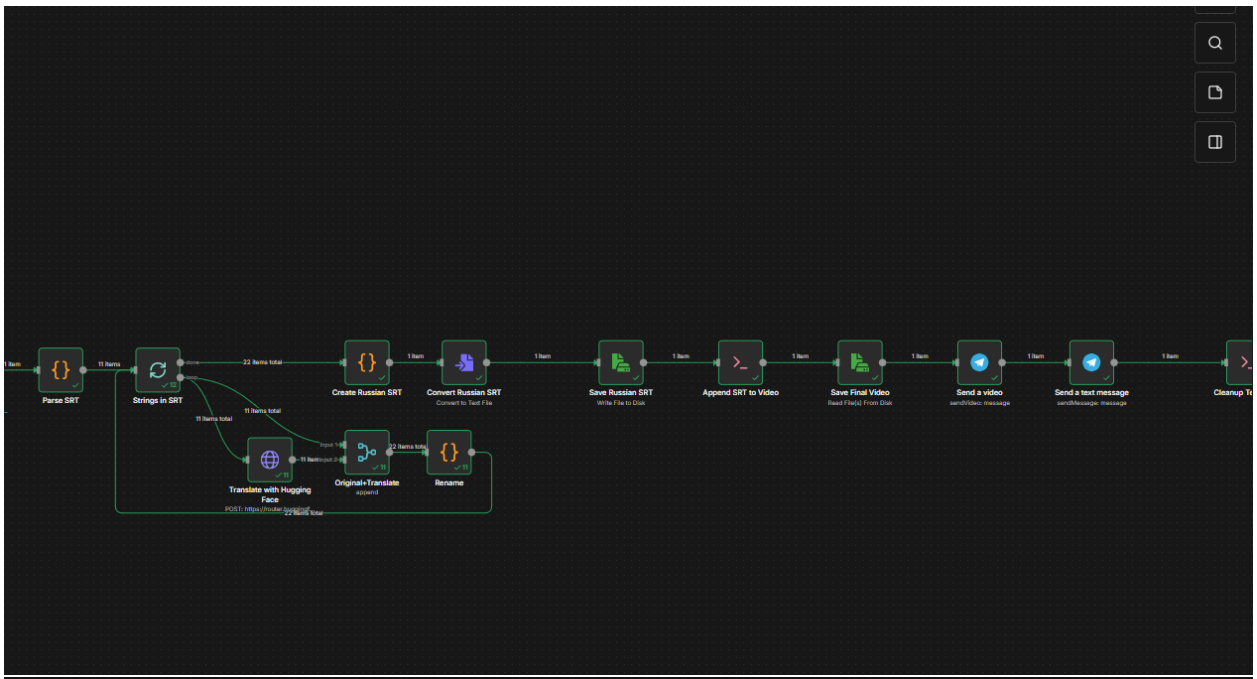
Добавлено несколько проверок данных, обработок точек сбоя и т. д на разных этапах. Приведем их:

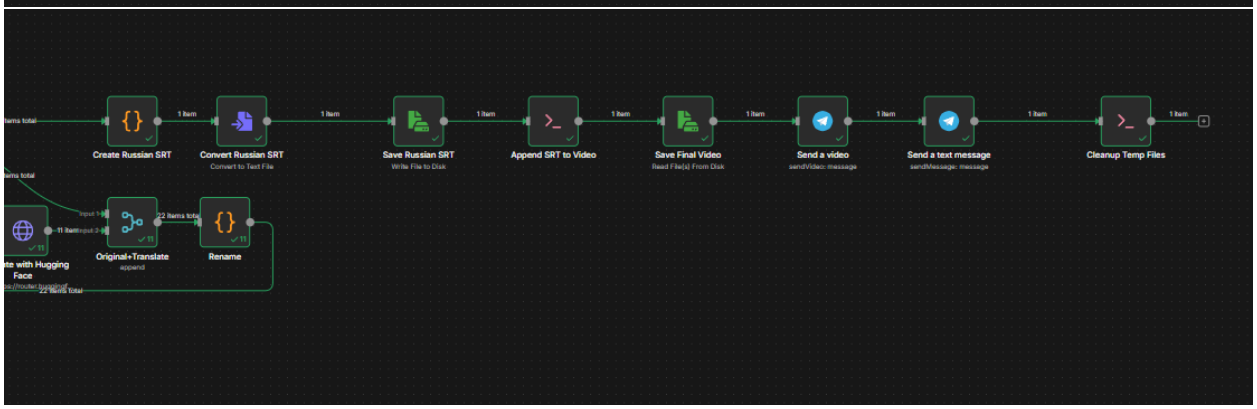
- Проверка наличия новых сообщения.
- Проверка формата файла.
- Проверка на mime type.
- Проверка доступ и работу сервиса.
- Ошибки в случае отсутствия файла в сервисе для перевода.
- Ошибки, возникающие при переводе.
- Гарантированная очистка в случае успешного завершения работы workflow

3. ***Результаты.***

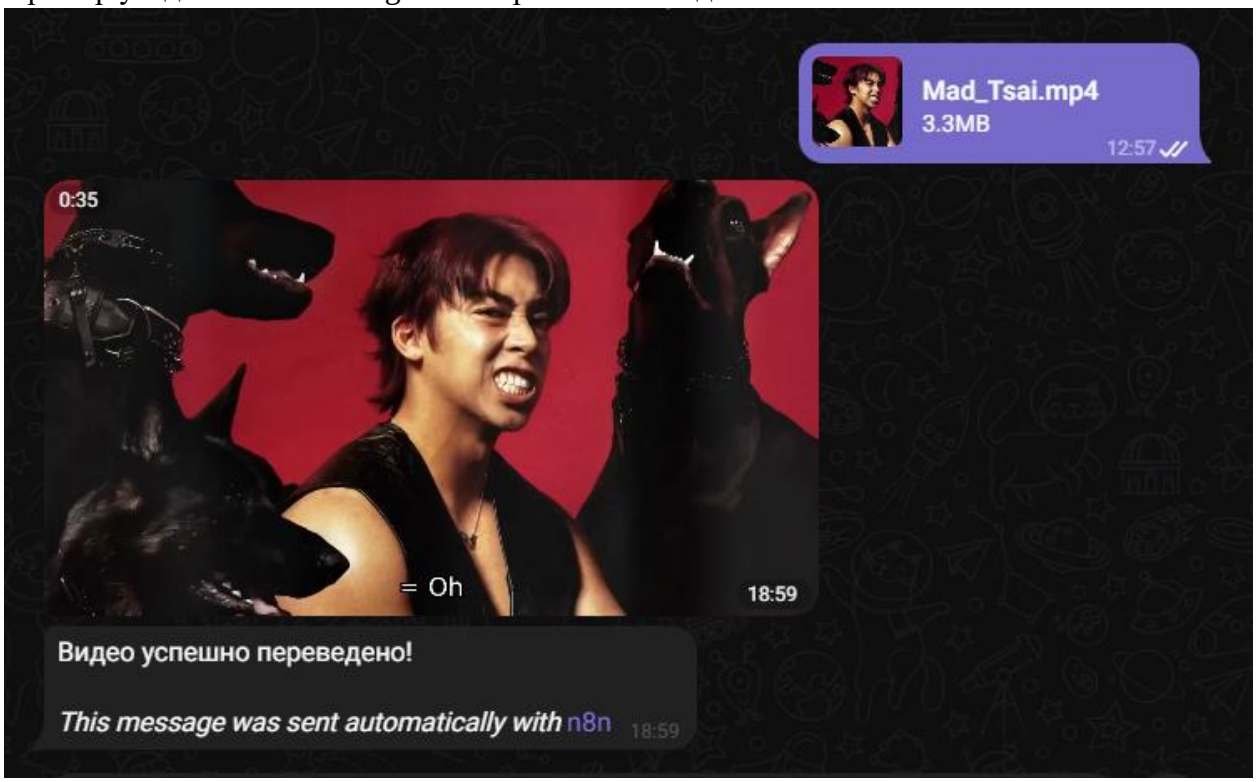
Также представлен в workflow в виде файла с названием Telegram Bot. Скриншоты с n8n с отображением узлов:







Пример уведомления в Telegram и скриншот из видео:



4. Выводы: что было сложным, что бы улучшили.

Первоначальная попытка выполнить проект без FastAPI (version 0.121.3) оказалась неудачной из-за ограничений и особенностей интеграции с n8n. Попытки использовать Whisper напрямую приводили к сбоям контейнера, что потребовало перехода на auto_subtitle

Интеграция auto_subtitle и локальная установка зависимостей (в частности GPU-версии torch) привели к большим размерам образов (~20 ГБ) и усложнили развёртывание на локальном сервере. Попытки вручную заменить сборку на CPU-версию оказались затруднительными, поэтому пришлось работать с данным размером контейнеров.

Переход на локальную LLM (vLLM/llama) потребовал значительных ресурсов; в условиях ограниченных ресурсов показалось оправданым использование Hugging Face Inference API, вместо локальной LLM. auto_subtitle и torch кажутся и так увеличили размеры контейнеров.

Проблемы с Telegram trigger на localhost — работал нестабильно в случае локального запуска, потребовалось использовать scheduled trigger как альтернативу.

По улучшениям:

- Автоматическое определение языка входных субтитров и многоязычная поддержка.
- Ограничение по длительности и размеру видео на входе.
- Поддержка дополнительных форматов видео и возможно ссылок.
- Добавление механизма очереди для обработки большого количества видео.
- Добавление механизма очереди для обработки нескольких видео.
- Отображение прогресса обработки в Telegram.
- Добавление возможности отправки итогового .srt файла отдельно от видео.

Лабораторная работа оказалась требующей значительных вычислительных и технических ресурсов, особенно, если сравнивать с первой, но позволила глубже изучить автоматизацию рабочих процессов, интеграцию с внешними сервисами и обработку медиаконтента.