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 и кибернетики
Кафедра технической кибернетики

ЛАБОРАТОРНАЯ РАБОТА № 2

по курсу
Инженерия данных

Группа 6232-010402D
Студент_____ Л.А. Абакумов
(*подпись*)

Преподаватель,
к.т.н. _____ Р.А. Паринге
(*подпись*)

Самара 2025

1. ЗАДАНИЕ

1. Telegram Bot принимает либо ссылку на видео, либо сам файл видео.
2. Если пришла ссылка – скачать видео; если пришёл файл – использовать его напрямую. В обоих случаях извлечь аудиодорожку с помощью ffmpeg.
3. Сгенерировать субтитры (EN), используя auto_subtitle.
4. Выполнить перевод EN→RU с помощью LLM из HuggingFace (допустим API; предпочтительно – собственный сервер vLLM/LLama).
5. Добавить полученные (RU) субтитры в исходное видео.
6. Отправить результат обратно пользователю в Telegram.

2. ОПИСАНИЕ РАБОТЫ

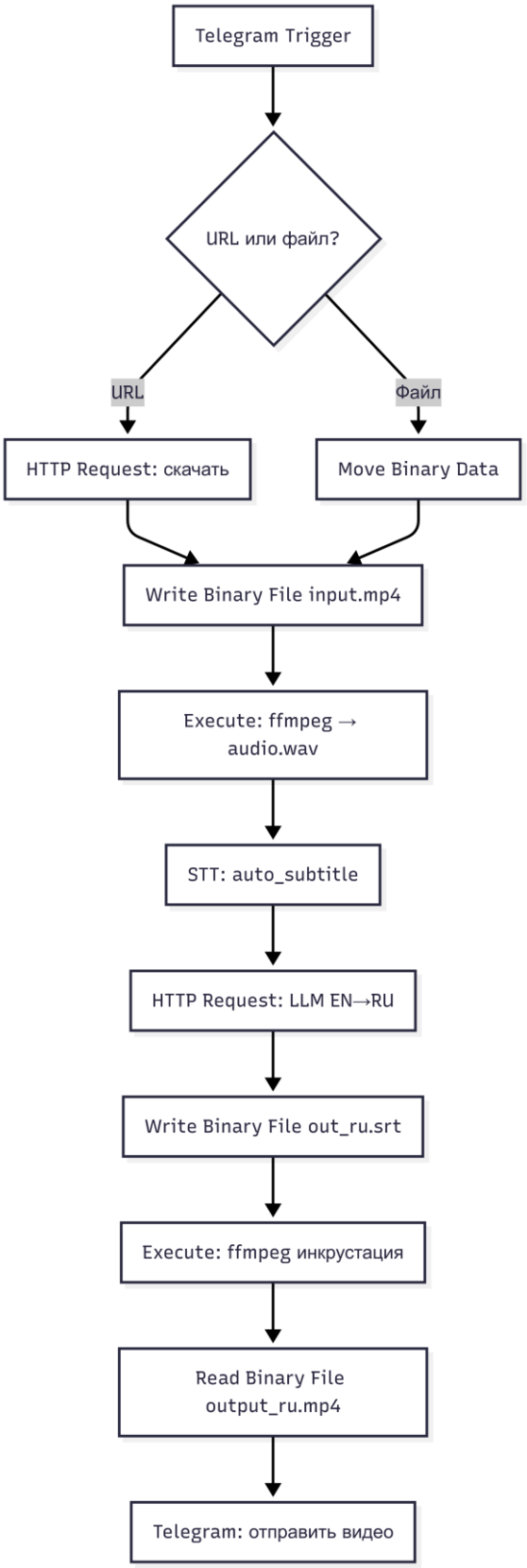

Рисунок 1 – Схема пайплайна

Решение реализовано на базе n8n в Docker. Workflow принимает сообщение из Telegram через webhook, сохраняет входное видео во временной каталог /data/temp, извлекает аудио (ffmpeg), генерирует английские субтитры (Whisper → SRT), переводит их на русский через сервис FastAPI с дополнительной обработкой LLM через HuggingFace Router (HF_MODE=llm), после чего встраивает русские субтитры в видео с помощью ffmpeg и отправляет готовый файл пользователю. Для контроля выполнения на ключевых этапах отправляются статусные уведомления.

Telegram Trigger (telegramTrigger). Узел инициирует запуск workflow при поступлении нового сообщения в Telegram-бот. На вход поступает webhook-событие Telegram (update типа message), а на выходе формируется JSON, содержащий идентификатор чата (message.chat.id), текст сообщения (message.text) и данные вложений (message.video / message.document). В конфигурации включена опция загрузки вложений, что позволяет получать файл также в бинарном виде.

Set chat id (set). Узел используется как техническая точка после триггера и в текущей конфигурации не выполняет преобразования данных. Его основная роль – служить удобной точкой подключения последующих уведомлений и ветвлений, повышая читаемость схемы.

Status: start (telegram). На данном шаге пользователю отправляется уведомление о начале обработки. Сообщение направляется в чат, идентификатор которого берется из данных, полученных на входе от Telegram Trigger.

If (if). Узел определяет тип входных данных: выполняется проверка наличия полей message.document или message.video. При наличии вложения выбирается ветка обработки файла (TRUE). При отсутствии вложения выбирается ветка обработки ссылки (FALSE), где URL ожидается в message.text.

Telegram Download File (telegram), ветка TRUE. Если пользователь отправил видеофайл, узел скачивает его из Telegram по file_id. Результатом

является бинарное содержимое видео, которое далее сохраняется во временный каталог.

HTTP Request (httpRequest), ветка FALSE. Если вход представлен ссылкой, узел загружает видео по URL, полученному из текста сообщения (message.text). Результатом также является бинарный файл, эквивалентный по формату данным ветки скачивания из Telegram.

Read/Write Files from Disk (readWriteFile). На данном этапе входной файл (из любой ветки) приводится к единому виду и сохраняется по пути /data/temp/input.mp4. Это обеспечивает одинаковую дальнейшую обработку независимо от источника видео.

Status: saved input (telegram). После успешного получения и сохранения входного видео пользователю отправляется уведомление о том, что видео принято и начинается извлечение аудио.

Execute Command (executeCommand) – извлечение аудио. Узел запускает ffmpeg для извлечения аудиодорожки из видеоролика. В результате формируется файл /data/temp/audio.wav в формате WAV (16 kHz, mono), который является оптимальным входом для распознавания речи.

Status: audio ready (telegram). Пользователь получает уведомление о завершении извлечения аудио и запуске этапа распознавания речи.

Execute Command1 (executeCommand) – STT (Whisper). Узел выполняет распознавание английской речи и генерирует субтитры в формате SRT. Результатом является файл /data/temp/subs_en.srt.

Status: subs en ready (telegram). После формирования английских субтитров пользователю отправляется уведомление о готовности результата и о переходе к этапу перевода.

Read/Write Files from Disk1 (readWriteFile). Узел читает файл /data/temp/subs_en.srt и используется как контрольная точка, обеспечивающая явный переход к шагу перевода и удобство трассировки при отладке.

HTTP Request1 – перевод EN в RU через сервис video-translation (FastAPI). На данном шаге выполняется перевод субтитров через сервис video-translation (эндпоинт POST /translate). Внутри узла запускается скрипт, который читает /data/temp/subs_en.srt, отправляет запрос на сервис перевода и сохраняет результат в /data/temp/subs_ru.srt.

Status: ru subs ready (telegram). Пользователь получает уведомление о готовности русских субтитров и запуске этапа встраивания их в видео.

Execute Command2 (executeCommand) – встраивание (burn-in) субтитров. Узел выполняет встраивание SRT в видеоряд с использованием ffmpeg (фильтр subtitles=/data/temp/subs_ru.srt и кодирование libx264/aac). Для повышения надежности предусмотрена проверка результата: если файл /data/temp/subs_ru.srt отсутствует либо имеет некорректно малый размер, используется резервный вариант (английские субтитры), что предотвращает выпуск пустого результата. На выходе формируется итоговый файл /data/temp/output_ru.mp4.

Read/Write Files from Disk3 (readWriteFile). Итоговое видео /data/temp/output_ru.mp4 считывается с диска и преобразуется в бинарные данные для отправки пользователю.

Status: sending (telegram). Перед отправкой результата пользователю направляется уведомление о том, что итоговое видео сформировано и начинается отправка в чат.

Telegram (telegram) – отправка результата. Узел выполняет операцию sendVideo и отправляет пользователю итоговое видео в исходный чат. В качестве данных используется бинарный файл, сформированный на предыдущем шаге.

Execute Command3 (executeCommand) – очистка. Заключительный узел удаляет временные артефакты обработки: /data/temp/input.mp4, /data/temp/audio.wav, /data/temp/subs_en.srt, /data/temp/subs_ru.srt, /data/temp/output_ru.mp4. На этапе отладки данный шаг целесообразно отключать, чтобы сохранить промежуточные файлы для анализа.

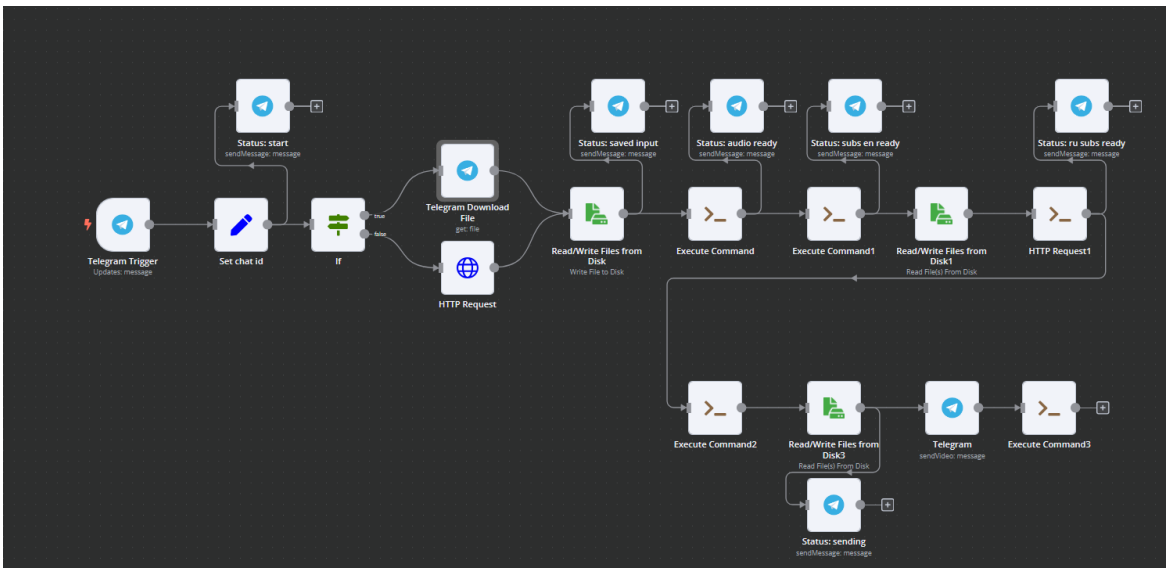

Рисунок 2 – n8n web UI

Для работы Telegram Trigger требуется публичный HTTPS-webhook с валидным TLS. При локальном запуске n8n используется туннелирование через tuna: внешний HTTPS-адрес проксируется на локальный http://127.0.0.1:5678. Публичный URL задается переменной окружения WEBHOOK_URL, после чего n8n регистрирует его в Telegram как webhook.

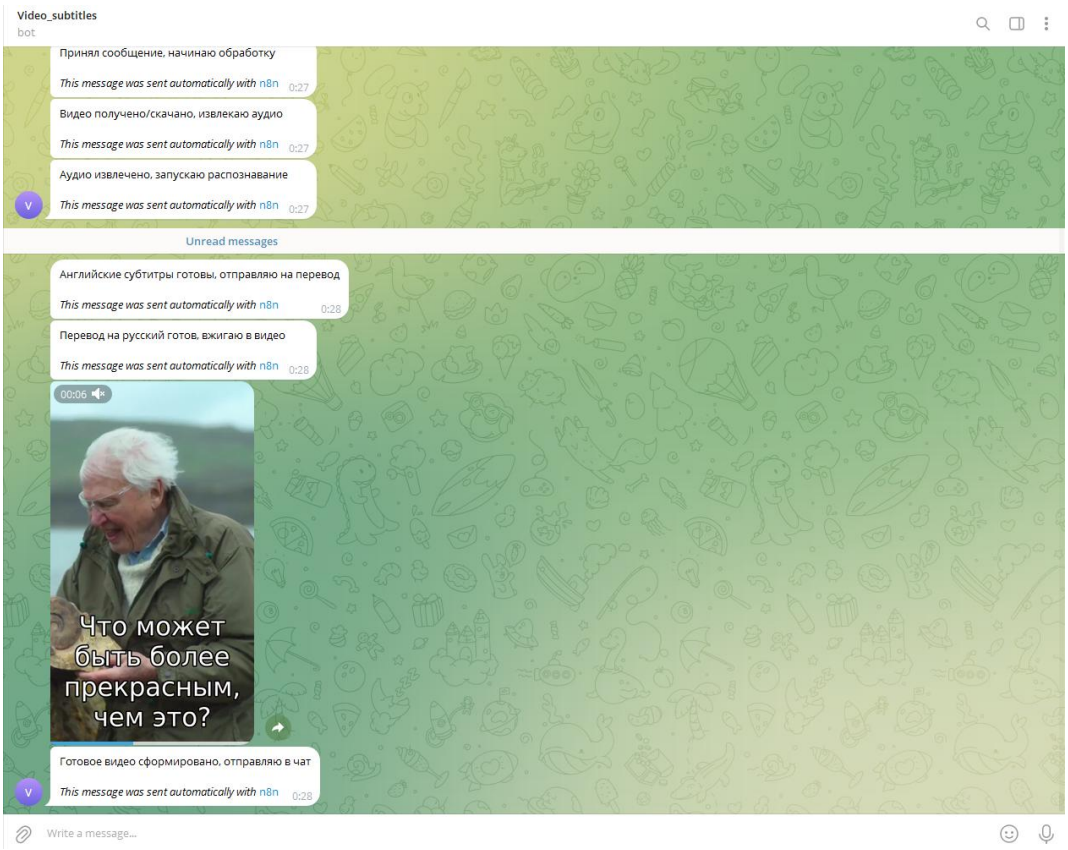

Рисунок 3 – Чат Telegram-бота

На выходе пользователь получает видеофайл с русскими субтитрами. Временные артефакты (mp4/wav/srt) формируются в общей директории и удаляются после выполнения.

ЗАКЛЮЧЕНИЕ

В результате выполнения лабораторной работы был разработан n8n-workflow для автоматической обработки видеоматериалов и формирования итогового видео с русскими субтитрами. Реализация включает локально развернутый инстанс n8n в Docker, обработку входных данных из Telegram (ссылка или файл), извлечение аудиодорожки с помощью ffmpeg, генерацию английских субтитров на базе Whisper в формате SRT, перевод субтитров через сервис video-translation (FastAPI) с использованием LLM (HuggingFace Router), а также последующее встраивание русских субтитров в исходное видео и отправку результата пользователю через Telegram-бота.

Основные трудности возникли на этапах освоения n8n и настройки публичного HTTPS-webhook для Telegram Trigger при локальном запуске. Поскольку Telegram требует доступный из интернета адрес с валидным TLS, потребовалось использование туннелирования.