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

Отчет по лабораторной работе №2

Дисциплина: «Инженерия данных»

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

Выполнил: Миролюбов В.Б.

Группа: 6233-010402D

Самара 2025

Данный отчет будет строиться на описании пайплайна из n8n, потому что он является центральным элементом данной системы.

Небольшая пометка: в данной работе не использовались специальные телеграмм-ноды, потому что n8n, запущенный на локалхосте, не может их использовать, поэтому взаимодействие с телеграмм было реализовано через API и ноды HttpRequests.

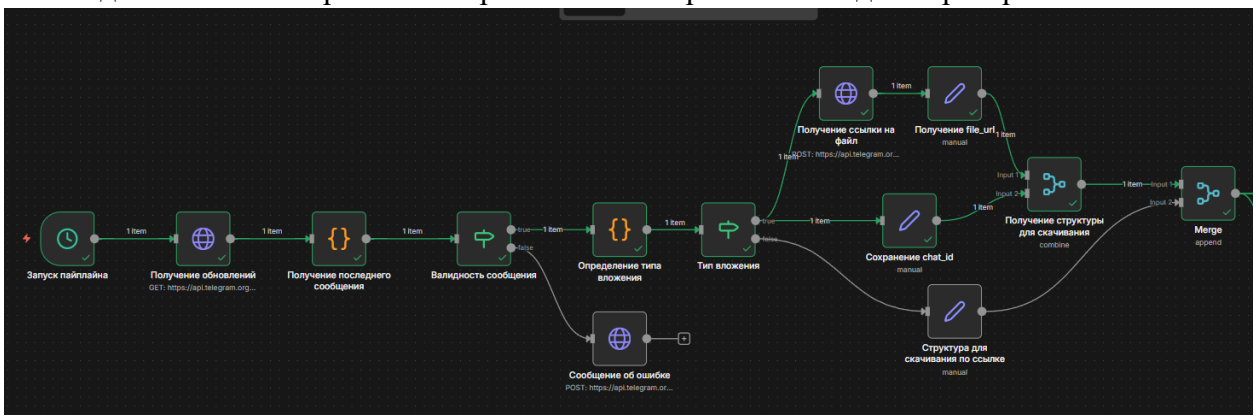

Рисунок 1 – Первая часть пайплайна

Первая часть пайплайна проверяет входящее сообщение от пользователя, что оно соответствует критериям: либо ссылка, либо видео-файл. В противном случае пользователь получает сообщение об ошибке.

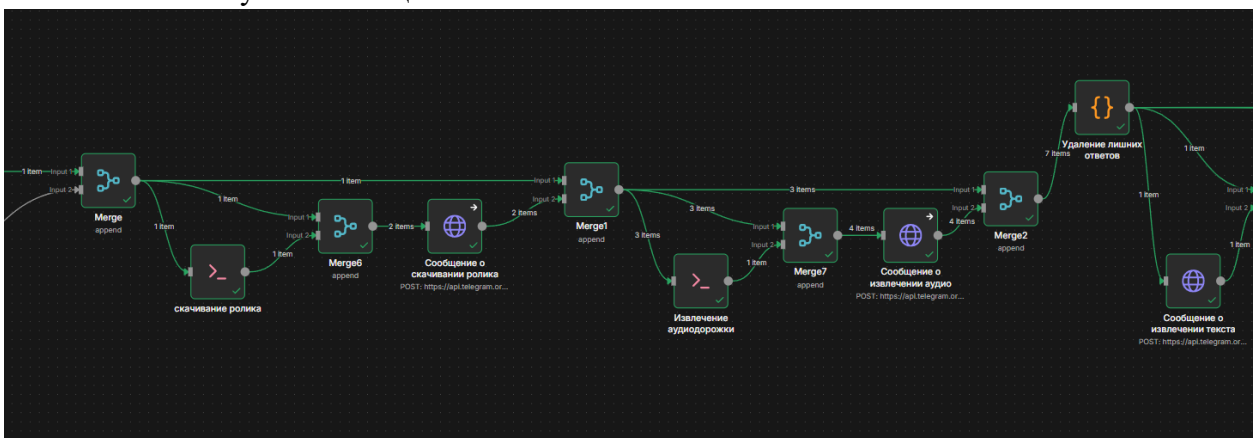

Рисунок 2 – Вторая часть пайплайна

Вторая часть пайплайна предназначена для скачивания ролика, извлечения из него аудиодорожки.

Скачивание производилось при помощи команды curl. Извлечение аудио при помощи ffmpeg.

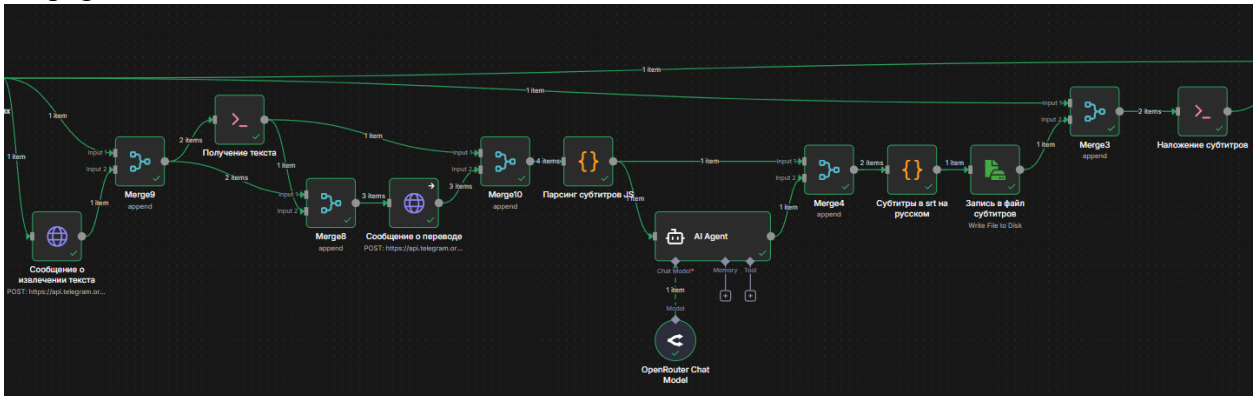

Рисунок 3 – Третья часть пайплайна

Третья часть пайплайна предназначена для получения и перевода текста, формированию и наложению субтитров.

Получение текста было реализовано при помощи модели Whisper. Для работы модели был реализован скрипт, который принимал запросы от n8n. Было реализовано взаимодействие n8n с моделью Whisper по технологии REST API.
Парсинг полученных субтитров был произведен с помощью кода JS.
Перевод текста был реализован при помощи Open-Router API.
Наложение субтитров на ролик было реализовано при помощи ffmpeg.

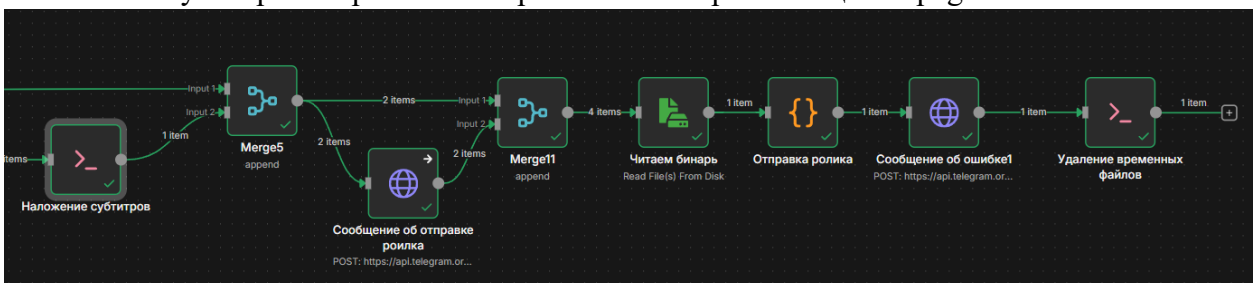

Рисунок 4 – Отправка ролика

Последняя часть пайплайна предназначена для отправки ролика с субтитрами пользователю и удалению временных файлов (2 видео, аудио, 2 файла с субтитрами).
Сама отправка ролика была реализована через чистый JS код.

Ключевые моменты реализации пайплайна:

1. Сервис для модели Whisper был поднят отдельно от n8n, поэтому его нет в d-c.yml
2. Команда для скачивания ролика (пример): curl -L {{ $input.first().json.file_url }} -o /tmp/video_{{ $input.first().json.chat_id }}_{{ $json.update_id }}.mp4

Имя файла содержит те же chat_id и update_id, чтобы:
- избежать коллизий между разными пользователями,
- легко сопоставить аудио с исходным видео и запросом.

3. Команда для извлечения аудиодорожки: ffmpeg -i /tmp/video_{{ $input.first().json.chat_id }}_{{ $json.update_id }}.mp4 -vn -acodec mp3 -ar 16000 /tmp/audio_{{ $input.first().json.chat_id }}_{{ $json.update_id }}.mp3

   Ключевые параметры
   -vn - отключает видео-поток (от англ. video none). Так как требуется получиться только аудио, видео не нужно.
   -acodec mp3 - указывает аудиокодек для выходного файла.
   mp3 - широко поддерживаемый формат, совместимый со многими системами распознавания речи (ASR).
   -ar 16000 - устанавливает частоту дискретизации (sample rate) аудио в 16 000 Гц.
   Почему именно 16 kHz?
   - Это стандарт для большинства систем распознавания речи (ASR), включая OpenAI Whisper (рекомендует 16 kHz для английского и многих языков),
   - Речь человека содержит почти всю полезную информацию в диапазоне до 8 kHz, а по теореме Найквиста для корректного восстановления нужна частота ≥16 kHz.
4. Команда для извлечения текста: curl -X POST "http://itkubrik.ru:11788/transcribe_srt" -F file=@/tmp/audio_{{ $input.first().json.chat_id }}_{{ $json.update_id }}.mp3

   Отправляется POST запрос на сервер, который принимает аудио-файл и возвращает текст в формате субтитров.
5. Модель LLM: openai/gpt-oss-20b:free

   Системный промпт модели:
   You are a professional subtitler. Translate the following dialogue lines into Russian.
   - Preserve the meaning and tone.

- Keep line breaks exactly as in input.
- NEVER add extra text, notes, or formatting.
- Output ONLY the translated lines, separated by \n|||\n.
6. Команда для наложения готовых субтитров на видео:

```
ffmpeg -i "/tmp/video_{{ $json.chat_id }}_{{ $json.update_id }}.mp4" \
  -vf "subtitles=/tmp/translated_srt.srt:force_style='Fontsize=16,PrimaryColour=&Hffffff&'"
\
  -c:v libx264 \
  -crf 28 \
  -preset fast \
  -c:a aac \
  -b:a 96k \
  -movflags +faststart \
  -max_muxing_queue_size 1024 \
  -y "/tmp/video_with_subs_{{ $json.chat_id }}_{{ $json.update_id }}.mp4"
```

Ключевые параметры
-vf "subtitles=..." - видеофильтр для субтитров
- subtitles=/tmp/translated_srt.srt — путь к файлу субтитров
- force_style='...' - принудительно задаёт стиль субтитров:
- Fontsize=16 — размер шрифта
- PrimaryColour=&Hffffff& - белый цвет в формате BGR в шестнадцатеричном виде с альфа-каналом

-c:v libx264 - кодек для видео: H.264 через библиотеку libx264.
Это самый совместимый видеоформат для MP4 - проигрывается везде: браузеры, Telegram, мобильные устройства.

-crf 28 - Constant Rate Factor - управляет качеством видео:
- Чем ниже CRF, тем выше качество (и больше размер).
- Диапазон: 0–51 (0 = без потерь, 51 = почти чёрный экран)
- 23–28 — стандарт для web: 28 — ниже среднего, но маленький размер подходит для Telegram (ограничение 50 МБ)

-preset fast - баланс между скоростью кодирования и сжатием: fast - быстрое кодирование с умеренным сжатием

-c:a aac - кодек для аудио: AAC, стандарт для MP4. Поддерживается везде, включая Telegram.

-b:a 96k - битрейт аудио: 96 кбит/с.
- Достаточно для речи (музыка требует 128–192 кбит/с)
- Экономит место без заметной потери качества

-movflags +faststart - перемещает метаданные в начало файла. Telegram и мобильные плееры требуют faststart для быстрого предпросмотра.

-max_muxing_queue_size 1024 - увеличивает размер внутренней очереди мультиплексирования. Нужно, если FFmpeg ругается: Too many packets buffered for output stream. Часто возникает при обработке длинных видео или сложных фильтров (как subtitles). 1024 - безопасное значение.

-y - перезаписывает выходной файл без подтверждения. Важно в автоматизированных пайплайнах, иначе команда зависнет в ожидании ввода.

В качестве примера был прочитал небольшой простой текст на английском языке, ролик снят на камеру телефона Apple Iphone 12, изначально ролик был чуть длиннее минуты, как требовалось по заданию, но размер был слишком большой, поэтому он был обрезан в два раза.
Пример работы: