|  |
| --- |
| Федеральное государственное бюджетное образовательное учреждение высшего образования  **«РОССИЙСКАЯ АКАДЕМИЯ НАРОДНОГО ХОЗЯЙСТВА И ГОСУДАРСТВЕННОЙ СЛУЖБЫ ПРИ ПРЕЗИДЕНТЕ РОССИЙСКОЙ ФЕДЕРАЦИИ»**  КОЛЛЕДЖ МНОГОУРОВНЕВОГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ |

Утверждается преподавателем по дисциплине ОП.14 Основы объектно-ориентированного программирования

\_\_\_\_\_\_\_\_\_\_\_\_\_Гусятинер Л. Б.

«\_\_\_»\_\_\_\_\_\_\_\_20\_\_г.

**Документация**

к программе для транскрибирования аудио из видео

Москва, 2024

**Введение**

Эта программа предназначена для извлечения аудио из видеофайла, транскрибирования аудио в текст с использованием модели Whisper от Hugging Face, удаления повторений слов и разделения текста на абзацы.

**Используемые библиотеки**

* MoviePy: Библиотека для редактирования видео.
* Transformers: Библиотека от Hugging Face для работы с моделями машинного обучения.
* Torch: Библиотека для работы с тензорами и нейронными сетями.
* Librosa: Библиотека для анализа аудио.
* OS: Библиотека для работы с операционной системой.
* Tempfile: Библиотека для работы с временными файлами.
* Re: Библиотека для работы с регулярными выражениями.

**Установка**

Для установки всех необходимых библиотек выполните следующие команды:

pip install moviepy tr

**Основные функции**

**1. MoviePy**

MoviePy — это библиотека для редактирования видео в Python. Она позволяет легко загружать, редактировать и сохранять видеофайлы.

Основные функции:

1. **VideoFileClip:** Класс для загрузки и редактирования видеофайлов.

from moviepy.editor import VideoFileClip

video = VideoFileClip("path/to/video.mp4")

1. **audio:** Атрибут для извлечения аудио из видео.

audio = video.audio

1. **write\_audiofile:** Метод для сохранения аудио в файл.

audio.write\_audiofile("path/to/audio.mp3", codec='libmp3lame')

1. **close:** Метод для закрытия видеофайла.

video.close()

**Пример использования:**

from moviepy.editor import VideoFileClip

video = VideoFileClip("path/to/video.mp4")

audio = video.audio

audio.write\_audiofile("path/to/audio.mp3", codec='libmp3lame')

video.close()ansformers torch librosa

**2. Transformers**

Transformers — это библиотека от Hugging Face для работы с моделями машинного обучения, включая модели для обработки естественного языка (NLP).

Основные функции:

1. **WhisperProcessor:** Класс для загрузки и использования процессора модели Whisper.

from transformers import WhisperProcessor

processor = WhisperProcessor.from\_pretrained("openai/whisper-tiny")

1. **WhisperForConditionalGeneration:** Класс для загрузки и использования модели Whisper.

from transformers import WhisperForConditionalGeneration

model = WhisperForConditionalGeneration.from\_pretrained("openai/whisper-tiny")

1. **to:** Метод для перемещения модели на GPU или CPU.

model.to("cuda" if torch.cuda.is\_available() else "cpu")

1. **generate:** Метод для генерации текста на основе аудио.

with torch.no\_grad():

generated\_ids = model.generate(inputs["input\_features"], attention\_mask=inputs["attention\_mask"])

1. **batch\_decode:** Метод для декодирования генерированных идентификаторов в текст.

transcription = processor.batch\_decode(generated\_ids, skip\_special\_tokens=True)[0]

**Пример использования:**

from transformers import WhisperProcessor, WhisperForConditionalGeneration

import torch

processor = WhisperProcessor.from\_pretrained("openai/whisper-tiny")

model = WhisperForConditionalGeneration.from\_pretrained("openai/whisper-tiny")

model.to("cuda" if torch.cuda.is\_available() else "cpu")

# Пример использования модели для генерации текста

inputs = processor(audio\_chunk, sampling\_rate=16000, return\_tensors="pt")

inputs = {k: v.to("cuda" if torch.cuda.is\_available() else "cpu") for k, v in inputs.items()}

with torch.no\_grad():

generated\_ids = model.generate(inputs["input\_features"], attention\_mask=inputs["attention\_mask"])

transcription = processor.batch\_decode(generated\_ids, skip\_special\_tokens=True)[0]

**3. Torch**

Torch (или PyTorch) — это библиотека для работы с тензорами и нейронными сетями. Она широко используется для машинного обучения и глубокого обучения.

**Основные функции:**

1. **torch.no\_grad:** Контекстный менеджер для отключения вычисления градиентов.

import torch

with torch.no\_grad():

# Вычисления без градиентов

1. **to:** Метод для перемещения тензоров на GPU или CPU.

inputs = {k: v.to("cuda" if torch.cuda.is\_available() else "cpu") for k, v in inputs.items()}

**Пример использования:**

import torch

with torch.no\_grad():

generated\_ids = model.generate(inputs["input\_features"], attention\_mask=inputs["attention\_mask"])

**4. Librosa**

Librosa — это библиотека для анализа аудио и музыки. Она позволяет легко загружать, анализировать и обрабатывать аудиофайлы.

**Основные функции:**

**librosa.load:** Функция для загрузки аудиофайлов и преобразования их в массив numpy.

import librosa

speech, sampling\_rate = librosa.load("path/to/audio.mp3", sr=16000)

**Пример использования:**

import librosa

speech, sampling\_rate = librosa.load("path/to/audio.mp3", sr=16000)

**5. OS**

OS — это стандартная библиотека Python для работы с операционной системой. Она позволяет выполнять различные операции с файлами и директориями.

**Основные функции:**

1. **os.environ:** Словарь, содержащий переменные окружения.

import os

os.environ["HF\_HUB\_DISABLE\_SYMLINKS\_WARNING"] = "1"

1. **os.remove:** Функция для удаления файла.

import os

os.remove("path/to/file.txt")

**Пример использования:**

import os

os.environ["HF\_HUB\_DISABLE\_SYMLINKS\_WARNING"] = "1"

os.remove("path/to/file.txt")

**6. Tempfile**

Tempfile — это стандартная библиотека Python для работы с временными файлами и директориями.

**Основные функции:**

1. **tempfile.NamedTemporaryFile:** Функция для создания временного файла.

import tempfile

with tempfile.NamedTemporaryFile(delete=False, suffix=".mp3") as temp\_audio\_file:

temp\_audio\_path = temp\_audio\_file.name

**Пример использования:**

import tempfile

with tempfile.NamedTemporaryFile(delete=False, suffix=".mp3") as temp\_audio\_file:

temp\_audio\_path = temp\_audio\_file.name

**7. Re**

Re — это стандартная библиотека Python для работы с регулярными выражениями. Она позволяет выполнять поиск и замену текста на основе регулярных выражений.

**Основные функции:**

**re.split:** Функция для разбивки строки по регулярному выражению.

import re

sentences = re.split(r'(?<=[.!?]) +', text)

**Пример использования:**

import re

sentences = re.split(r'(?<=[.!?]) +', text)

**Основные функции программы**

**1. extract\_audio**

Описание: Функция для извлечения аудио из видеофайла.

**Пример использования:**

temp\_audio\_path = extract\_audio("path/to/video.mp4")

Параметры:

* video\_path: Путь к видеофайлу.

Возвращаемое значение:

* Путь к временному аудиофайлу.

**2. load\_audio**

Описание: Функция для загрузки аудио из файла.

**Пример использования:**

audio = load\_audio("path/to/audio.mp3")

Параметры:

* audio\_path: Путь к аудиофайлу.

Возвращаемое значение:

* Массив numpy, содержащий аудиоданные.

**3. split\_audio**

Описание: Функция для разбивки аудио на части.

**Пример использования:**

audio\_chunks = split\_audio(audio, chunk\_size=30)

Параметры:

* audio: Массив numpy, содержащий аудиоданные.
* chunk\_size: Длина каждой части в секундах (по умолчанию 30).

Возвращаемое значение:

* Список массивов numpy, содержащих части аудио.

**4. remove\_repetitions**

Описание: Функция для удаления повторений слов в тексте.

**Пример использования:**

text = remove\_repetitions("This is a test. This is a test.")

Параметры:

* text: Строка, содержащая текст.

Возвращаемое значение:

* Строка, содержащая текст без повторений слов.

**5. split\_into\_paragraphs**

Описание: Функция для разделения текста на абзацы.

**Пример использования:**

text = split\_into\_paragraphs("This is a test. This is another test.", max\_length=500)

Параметры:

* text: Строка, содержащая текст.
* max\_length: Максимальная длина абзаца (по умолчанию 500).

Возвращаемое значение:

* Строка, содержащая текст, разделенный на абзацы.

Основная функция программы

**main**

Описание: Основная функция программы, которая выполняет все шаги от извлечения аудио до сохранения транскрипции в файл.

**Пример использования:**

main("path/to/video.mp4", "path/to/transcription.txt")

Параметры:

* video\_path: Путь к видеофайлу.
* output\_file\_path: Путь для сохранения текста транскрипции.

Возвращаемое значение:

* Нет возвращаемого значения.