Министерство науки и высшего образования Российской Федерации
ФГБОУ ВО «АЛТАЙСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
Институт цифровых технологий, электроники и физики
Кафедра вычислительной техники и электроники (ВТиЭ)

Отчёт по практике

Автоматизация решения CAPTCHA в аудиоформате

Выполнил: студент гр. 5.306М:

_____ Лаптев А. В.

Проверил: доц. каф. ВТиЭ

_____ Калачев А. В.

Оценка _____

«_____» _____2024 г.

Барнаул 2024

РЕФЕРАТ

Полный объем работы составляет 22 страницы, включая 2 рисунка, 0 таблиц, 10 источников.

Данная работа посвящена разработке программного сервиса для автоматизации решения Audio CAPTCHA на web-ресурсах для облегчения автоматизации тестирования web-приложений с использованием средств автоматизации.

Ключевые слова: автоматизация, CAPTCHA, Selenium, Python, Audio CAPTCHA, распознавание речи.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

На сегодняшний день большинство web-ресурсов внедряют различные меры для противодействия спаммерам и ботам, защищая пользователей от несанкционированного доступа, фишинговых атак и автоматизированного извлечения данных. Одним из широко используемых методов защиты является CAPTCHA. Этот механизм представлен в разных формах: текстовые, графические, аудио и интерактивные тесты, которые помогают отличить человека от машины.

С развитием технологий и усложнением CAPTCHA, автоматизация её решения становится полезной для решения следующих задач:

1. Автоматизированное тестирование web-приложений: Интеграция CAPTCHA в сайты требует проверки её работоспособности в рамках автоматизированных тестов, что позволяет выявлять ошибки на ранних стадиях тестирования.
2. Обход CAPTCHA для исследовательских целей: Разработка решений для автоматического распознавания CAPTCHA находит применение для анализа слабых мест этих систем и повышения их безопасности.
3. Сбор данных и анализ: Некоторые задачи требуют автоматического извлечения данных с web-ресурсов, в которые интегрирована CAPTCHA.

Автоматизация распознавания CAPTCHA с помощью современных средств и языков программирования, таких как Python и библиотека Selenium, позволяет сократить временные затраты на ручное тестирование или анализ web-приложений, делая процессы более эффективными. Эти инструменты дают возможность моделировать действия пользователей и обрабатывать сложные запросы, включая распознавание текста, изображений или аудиофайлов.

Цель работы: разработать, реализровать и протестировать программу для автоматизации решения Audio CAPTCHA.

Задачи, которые нужно решить для достижения цели:

1. Ознакомиться с наиболее часто встречающимися реализациями CAPTCHA на web-ресурсах.
2. Выбрать язык программирования и инструменты для автоматизации сценариев для взаимодействия с CAPTCHA.
3. Реализовать программу для автоматизации решения одной из реализаций CAPTCHA, предоставляемой в аудиоформате.

1. СОВРЕМЕННЫЕ МЕТОДЫ ЗАЩИТЫ ОТ БОТОВ И СПАМА НА ОСНОВЕ CAPTCHA

1.1. Что такое CAPTCHA

Проверочный код CAPTCHA – это метод защиты, основанный на принципе аутентификации «вызов-ответ». Он предназначен для предотвращения автоматических действий, таких как спам или попытки взлома учетных записей, путем выполнения пользователем простого теста, подтверждающего, что он человек, а не программа [1].

CAPTCHA является важной мерой безопасности, так как предотвращает автоматические атаки, например, массовую регистрацию ботов, и защищает данные пользователя. Современные системы CAPTCHA используют не только текст, но и изображения, аудио, поведенческие анализы и другие инновационные подходы, чтобы сделать тесты удобными для людей, но сложными для программ.

На сегодняшний день наиболее распространенные виды CAPTCHA включают:

1. reCAPTCHA – разработанная Google система, которая предлагает тесты на основе распознавания объектов, анализа поведения или текстовых символов.
2. hCAPTCHA – альтернатива reCAPTCHA, фокусирующаяся на защите конфиденциальности пользователей.
3. Capy – система CAPTCHA, предлагающая пользователю головоломки, например, сборку изображения или взаимодействие с элементами интерфейса [2].

1.2. reCAPTCHA

reCAPTCHA – система защиты от автоматизированных действий, разработанная Google, которая помогает различать человека и бота. Она объединяет несколько подходов, делая проверку удобной для пользователей, но сложной для автоматических систем [3].

reCAPTCHA включает в себя следующие версии:

1. reCAPTCHA v1 (устарела в 2018 году):
   1.1. Пользователи вводили текст, состоящий из искаженных слов, отображаемых на изображении.
   1.2. Использовала слова из книг и документов, которые не могли быть распознаны OCR.
2. reCAPTCHA v2:
   2.1. Клик по флажку: пользователи подтверждают, что они не роботы, нажимая на флажок «Я не робот».
   2.2. Выбор объектов на изображениях: пользователи идентифицируют заданные объекты на сетке из картинок.
   2.3. Аудио CAPTCHA: для пользователей с ограничениями зрения, предлагается прослушать запись и ввести услышанные символы.
3. reCAPTCHA v3:
   3.1. Полностью работает в фоновом режиме, анализируя поведение пользователя на странице.
   3.2. Не требует явных действий, если пользователь считается низкорискованным [4].

1.3. hCAPTCHA

hCAPTCHA – это альтернативная система CAPTCHA, разработанная для защиты сайтов от ботов и спама, при этом уделяющая особое внимание конфиденциальности пользователей. Она стала популярной благодаря своей гибкости и ориентации на защиту данных [5].

Основные особенности hCAPTCHA:

1. Конфиденциальность:
   1.1. В отличие от reCAPTCHA, hCAPTCHA не собирает данные о пользователях для рекламных целей, что делает ее привлекательной с точки зрения соблюдения конфиденциальности.
2. Простота интеграции:
   2.1. Легко интегрируется с web-сайтами через API.
   2.2. Совместима с большинством популярных платформ, таких как WordPress, и может быть настроена для разных типов взаимодействия.
3. Модели монетизации:
   3.1. Владельцы сайтов могут зарабатывать, разрешая hCAPTCHA использовать проверочные задачи, связанные с машинным обучением, например, разметку данных.

Виды взаимодействия с пользователями:

1. Графическая CAPTCHA: выбор изображений, соответствующих запросу.
2. Текстовая CAPTCHA: ввод символов (редко используется).
3. Аудио CAPTCHA: для пользователей с ограниченными возможностями, предлагается прослушать и ввести услышанные символы.
4. Клик CAPTCHA: нажатие на флажок «Я не робот» (для низкорискованных пользователей).

1.4. Capy

Capy CAPTCHA – это инновационная система CAPTCHA, разработанная с акцентом на удобство для пользователей и адаптацию к современным web-средам. Она предлагает интерактивные методы проверки,

направленные на минимизацию раздражения пользователей при сохранении высокого уровня защиты от ботов [6].

Основные особенности Capy CAPTCHA:

1. Интерактивность:
   1.1. Capy использует методы проверки, которые требуют не просто ввода текста или выбора картинок, а выполнения задач, таких как перемещение объектов.
   1.2. Простые задачи делают процесс проверки менее раздражающим и более интуитивным.
2. Гибкость настройки:
   2.1. Система может быть адаптирована под конкретные нужды сайта, включая выбор сложности задач и дизайн интерфейса.
3. Доступность:
   3.1. Подходит для пользователей с различными потребностями, включая мобильные устройства.

Виды взаимодействия с пользователями:

1. Головоломки (Puzzle CAPTCHA): сборка пазла с перемещением недостающих элементов в нужное место.
2. Тесты на логику и распознавание: выбор нужного объекта или логического варианта из предложенных.
3. Текстовая CAPTCHA (редко используется).

Capy CAPTCHA используется на сайтах, где важны как защита от ботов, так и положительный пользовательский опыт. Особенно популярна в проектах с высоким акцентом на дизайн и пользовательское взаимодействие.

2. АВТОМАТИЗАЦИЯ РЕШЕНИЯ AUDIO CAPTCHA НА САЙТЕ

2.1. Выбор языка программирования и инстументов для разработки

Для разработки сервиса по автоматизации распознавания CAPTCHA был выбран язык программирования Python и библиотека для автоматизации тестирования web-приложений Selenium.

Python обладает рядом преимуществ для решения подобных задач:

1. Простота и читаемость кода: Python легко использовать благодаря лаконичному синтаксису, что ускоряет разработку и упрощает поддержку проекта.
2. Широкая экосистема библиотек: Для работы с CAPTCHA можно использовать специализированные библиотеки, а также сторонние инструменты для машинного обучения.
3. Поддержка скриптового и объектно-ориентированного подхода: Это делает Python гибким для создания как небольших сценариев автоматизации, так и сложных систем.

Selenium выделяется следующими преимуществами среди других инструментов автоматизации:

1. Кросс-браузерная поддержка: Selenium поддерживает тестирование во всех популярных браузерах, таких как Chrome, Firefox, Edge и Safari.
2. Возможности для взаимодействия с динамическими элементами: Selenium может эмулировать действия пользователя, включая ввод текста, клики и работу с выпадающими меню, что полезно для работы с CAPTCHA.
3. Поддержка различных языков программирования: Хотя Python удобен для автоматизации, Selenium можно использовать с Java, C#, Ruby и другими языками.

4. Интеграция с другими библиотеками и инструментами: Selenium легко интегрируется с фреймворками для тестирования или системами для распознавания изображений.

Audio CAPTCHA представляет собой элемент, встроенный в web-страницу, который содержит в себе ссылку на отрезок звуковой дорожки, которая содержит шум и запись голоса.

Подобная запись хорошо поддается распознаванию с использованием современных библиотек для распознавания речи, одна из которых была использована для решения Audio CAPTCHA в данной работе.

В Python существует множество библиотек для распознавания человеческой речи, таких как Google Speech Recognition, Pocketsphinx, DeepSpeech и других [7]. Среди них была выбрана библиотека SpeechRecognition по следующим причинам:

1. Удобство использования: Простота в освоении и интеграции благодаря интуитивно понятному API.
2. Поддержка нескольких API: Библиотека предоставляет интерфейсы для работы с несколькими сервисами, включая Google Web Speech API, IBM Watson, Microsoft Azure и другие [8].
3. Кроссплатформенность: SpeechRecognition работает на Windows, macOS и Linux, что обеспечивает гибкость разработки.
4. Встроенные функции обработки звука: Возможность работы с различными форматами аудио, включая wav, и встроенные методы для улучшения качества записи перед отправкой на обработку.
5. Локальная и облачная обработка: Поддержка локальных движков, таких как PocketSphinx, и облачных сервисов, таких как Google Speech API, что делает библиотеку универсальной для различных задач.
6. Открытый исходный код: Это бесплатная библиотека с открытым кодом, что позволяет исследователям и разработчикам адаптировать её под свои нужды.

2.2. Описание алгоритма работы программы

Процесс получения аудиофайла, содержащего CAPTCHA, с web-сайта для последующего распознавания и отправку готового решения с использованием Selenium можно представить как следующую последовательность шагов:

1. Инициализация настроек браузера.
2. Открытие web-страницы, содержащей CAPTCHA.
3. Переключение на фрейм с чекбоксом CAPTCHA на web-странице и нажатие на чекбокс.
4. Переключение на фрейм с CAPTCHA в виде картинки или набора картинок и нажатие на кнопку доступа к аудиофайлу.
5. Переключение на фрейм с аудиозаписью и поиск элемента, который содержит ссылку на аудиозапись.
6. Создание запроса на получение файла по ссылке.
7. Передача файла в решатель для последующей обработки.
8. Вставка результата распознавания в текстовое поле и подтверждение ввода для успешного решения CAPTCHA.

Для создания запроса на получение файла используется встроенный модуль Python – requests.

Блок-схема, иллюстрирующая приведенный алгоритм представлена на рис. 2.1.

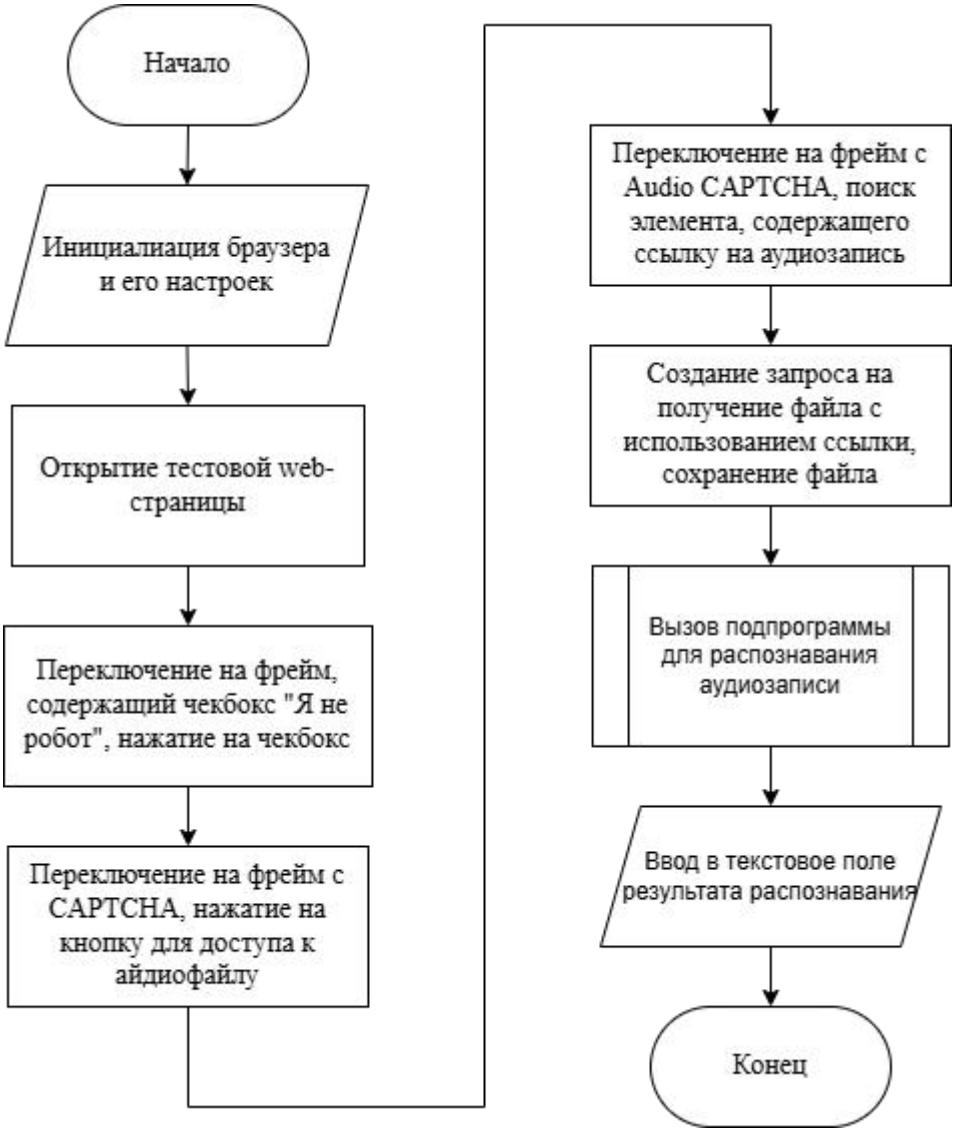

Рис. 2.1 Блок-схема алгоритма тестирования решателя Audio CAPTCHA.

Процесс обработки аудиофайла, полученного с web-страницы можно разделить на следующие этапы:

1. Преобразование полученного аудиофайла в другой формат, который является подходящим для распознавания.
2. Распознавание речи в перекодированном файле.
3. Сохранение полученного результата распознавания для последующей вставки в тектстовое поле.

На первом этапе исходный файл всегда имеет формат mp3, которое не пригодно для распознавания с использованием SpeechRecognition, поскольку данный формат использует сжатие с потерями, поэтому исходный файл

необходимо перекодировать в формат wav. Для перекодирования файла используется библиотека с открытым исходным кодом – ffmpeg, которая предоставляет обширные возможности для работы с любыми мультимедиа файлами [9].

На следующем этапе создается объект для распознавания, который содержит перекодированный файл. Распознавание происходит с использованием Google Web Speech API, поскольку данный API обеспечивает более высокую точность распознавания среди прочих [10].

Результатом распознавания является текстовое сообщение, которое сохраняется для последующих действий в браузере.

Описанный алгоритм можно представить в виде следующей блок-схемы (рис. 2.2):

Рис. 2.2 Блок-схема алгоритма решателя Audio CAPTCHA.

ЗАКЛЮЧЕНИЕ

В ходе выполнения работы были решены следующие задачи:

1. Осуществлено обзорное знакомство с наиболее часто встречающимися реализациями CAPTCHA на web-ресурсах.
2. Подобраны язык программирования и инструменты для автоматизации сценариев удобного взаимодействия с CAPTCHA.
3. Реализована программа для автоматизации решения одной из реализаций CAPTCHA, предоставляемой в аудиоформате.

В результате выполнения работы была реализована программа для автоматизации решения Audio CAPTCHA и создан автоматизированный сценарий для тестирования работоспособности программы на реальном сайте.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Что такое CAPTCHA? [Электронный ресурс] Сайт support.google.com. Режим доступа: https://support.google.com/a/answer/1217728?hl=ru.
2. Я не робот: 10 альтернатив reCAPTCHA [Электронный ресурс] Сайт tproger.ru. Режим доступа: https://tproger.ru/articles/recaptcha-alternatives.
3. reCAPTCHA (ru) [Электронный ресурс] Сайт wikipedia.org. Режим доступа: https://ru.wikipedia.org/wiki/ReCAPTCHA.
4. reCAPTCHA (en) [Электронный ресурс] Сайт wikipedia.org. Режим доступа: https://en.m.wikipedia.org/wiki/ReCAPTCHA.
5. hCAPTCHA [Электронный ресурс] Сайт hcaptcha.com. Режим доступа: https://www.hcaptcha.com/.
6. Capy CAPTCHA [Электронный ресурс] Сайт corp.capy.me. Режим доступа: https://corp.capy.me/en/product/captcha.
7. The State of Python Speech Recognition in 2021 [Электронный ресурс] Сайт assemblyai.com. Режим доступа: https://www.assemblyai.com/blog/the-state-of-python-speech-recognition-in-2021/.
8. SpeechRecognition [Электронный ресурс] Сайт pypi.org. Режим доступа: https://pypi.org/project/SpeechRecognition/.
9. FFMPEG [Электронный ресурс] Сайт ffmpeg.org. Режим доступа: https://www.ffmpeg.org/.
10. Как использовать Python для работы с распознаванием речи [Электронный ресурс] Сайт sky.pro. Режим доступа: https://sky.pro/media/kak-ispolzovat-python-dlya-raboty-s-raspoznavaniem-rechi/.

ПРИЛОЖЕНИЕ

Исходный код программы для тестирования решателя с использованием Selenium

```
from selenium import webdriver
from selenium.webdriver.remote.webdriver import WebDriver
from selenium.webdriver.common.by import By

import logger
from random import randint
import time
import requests
import os


from audiocaptcha import AudioCaptchaSolver



logger = logger.ConfigLogger(__name__)


class MainWorker():
    '''
    Основной класс проекта, который управляет вызовом дочерних классов для решения определенных видов captcha
    '''


    def __init__(self, browser: WebDriver):
        '''Конструктор класса'''
        super().__init__()
        self.browser = browser



    def get_captcha(self, link: str) -> str:
        '''Метод получения captcha со страницы'''
        # Проходим по ссылке
        self.browser.get(link)
        time.sleep(randint(3, 5))


        # Переключаемся на фрейм с чекбоксом captcha
        self.browser.switch_to.frame(self.browser.find_element(By.XPATH, '//*[@id="g-recaptcha"]/div/div/iframe'))
        # Кликаем по чекбоксу "Я не робот"
        self.browser.find_element(By.XPATH, '/html/body/div[2]/div[3]/div[1]/div/div/span').click()
```

```
                time.sleep(randint(3, 5))

                # Переключаемся на обычную web-страницу
                self.browser.switch_to.default_content()
                # Переключаемся на фрейм с картинкой captcha
                self.browser.switch_to.frame(self.browser.find_element(By.XPATH,
'/html/body/div[2]/div[4]/iframe'))
                # Кликаем на кнопку для перехода к audiocaptcha
                self.browser.find_element(By.XPATH, '//*[@id="recaptcha-audio-
button"]').click()
                time.sleep(randint(3, 5))

                # Переключаемся на обычную web-страницу
                self.browser.switch_to.default_content()
                # Переключаемся на фрейм с айдиозаписью
                self.browser.switch_to.frame(self.browser.find_element(By.XPATH,
'/html/body/div[2]/div[4]/iframe'))
                # Находим элемент, содержащий ссылку на аудиозапись
                audio = self.browser.find_element(By.XPATH, '//*[@id="audio-
source"]').get_attribute('src')
                # Делаем запрос для получения файла
                response = requests.get(audio)
                response.raise_for_status()


                # Создаем папку для хранения временных файлов
                if not os.path.isdir('./audio'):
                    os.mkdir('./audio')
                path_to_file = './audio/audiocaptcha.mp3'
                # Сохраняем файл
                with open(f'{path_to_file}', 'wb') as audioCaptcha:
                    audioCaptcha.write(response.content)


                return path_to_file



            def paste_response(self, response_message):
                '''Метод для вставки результата распознавания'''
                browser.find_element(By.XPATH, '//*[@id="audio-
response"]').send_keys(f'{response_message}')
                time.sleep(randint(3, 5))
                browser.find_element(By.XPATH, '//*[@id="recaptcha-verify-
button"]').click()



        if __name__ == '__main__':
            '''Запуск программы'''
```

```
            list_of_links = [
                'https://rucaptcha.com/demo/recaptcha-v2'
            ]


            for link in list_of_links:
                # Настройки user agent
                USER_AGENT = "Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/127.0.0.0 Safari/537.36"


                select_browser = randint(1, 10)


                # Выбор браузера и опций характерных для него
                if select_browser < 5:
                    options = webdriver.ChromeOptions()
                else:
                    options = webdriver.EdgeOptions()

                options.add_experimental_option("excludeSwitches", ["enable-
automation"])
                options.add_experimental_option('useAutomationExtension', False)
                options.add_argument(f"user-agent={USER_AGENT}")
                options.add_argument("--disable-blink-
features=AutomationControlled")


                # Передача параметров
                if select_browser < 5:
                    browser = webdriver.Chrome(options=options)
                else:
                    browser = webdriver.Edge(options=options)
                browser.implicitly_wait(30)


                # Создаем аудиофайл по указанному пути с captcha
                solver = MainWorker(browser)
                path_to_audio = solver.get_captcha(link)


                # Запускаем распознавание
                captcha_solver = AudioCaptchaSolver()
                response = captcha_solver.recognition_audio(path_to_audio)


                # Вставляем результат распознавания в поле ввода
                solver.paste_response(response)
                time.sleep(randint(10, 15))
```

Исходный код программы для распознавания Audio CAPTCHA

```
'''Файл с классаом для решения audiocaptcha'''
import speech_recognition as sr
import subprocess
import logger
import os


logger = logger.ConfigLogger(__name__)


class AudioCaptchaSolver():
    '''Класс решателя audio captcha'''


    def __init__(self):
        '''Конструктор класса'''
        # Создаем объект распознавателя речи
        self.recognizer = sr.Recognizer()
        # Распознанное текстовое сообщение
        self.text_message = None



    def recognition_audio(self, path_to_audio: str) -> str:
        '''
        Метод распознавания аудиофайла
        Файлы сохраняются в формате mp3 (обычно содержат шум, кроме мест,
где слышен голос)
        '''


        # Преобразование mp3 файла в формат, который подходит для
распознавания
        mp3_file = path_to_audio
        wav_file = './audio/audiocaptcha.wav'


        if os.name == 'nt':
            subprocess.run(['C:/ffmpeg/bin/ffmpeg.exe', '-i', mp3_file,
wav_file])
        else:
            subprocess.run(['ffmpeg', '-i', mp3_file, wav_file])

        try:
            # Загружаем аудио файл
            audio_captcha = sr.AudioFile(wav_file)


            # Распознаем речь из аудио файла
            with audio_captcha as voice:
                audio_data = self.recognizer.record(voice)
```

```
                    text_message = self.recognizer.recognize_google(audio_data,
language='en-US')
                logger.log_info('Распознавание речи завершено успешно!')
            except Exception as e:
                logger.log_warning(f'Распознавание завершилось с ошибкой: {e}')

            if text_message:
                self.text_message = text_message
                os.remove(mp3_file)
                os.remove(wav_file)

            return self.text_message
```