

Интерактивная цифровая книга для профессионалов

Анатолий Постоли́т

Компьютерное зрение и искусственный
интеллект в приложениях на
Python и Django



Postolit Press

Оглавление

ПРЕДИСЛОВИЕ	20
ГЛАВА 1	23
ОСНОВЫ КОМПЬЮТЕРНОГО ЗРЕНИЯ.....	23
1.1. Пиксели и цветовые пространства	23
1.2. Библиотеки для обработки изображений, видео файлов и видео потоков	27
1.3. Чтение изображений из файлов с выводом в шаблоны DJANGO.....	28
1.4. Чтение видео файлов с выводом в шаблоны DJANGO	30
1.5. Трансляция потоковых данных с видео камер в шаблоны DJANGO	32
1.5.1. Прямой вывод видеопотока в окно браузера с использованием API MediaStream	32
1.5.2. Вывод видеопотока в шаблон Django с использованием API MediaStream	37
1.5.3. Трансляция потоковых данных с видео камер в шаблоны Django	39
1.6. РАСПОЗНАВАНИЕ ОБЪЕКТОВ НА ОСНОВЕ КАСКАДОВ ХААРА.....	43
1.6.1. Распознавание лиц в изображениях на основе каскадов Хаара	47
1.6.2. Распознавание лиц в видео файлах на основе каскадов Хаара	50
1.6.3. Распознавание лиц в потоковом виде с видео камер на основе каскадов Хаара	55
1.7. РАСПОЗНАВАНИЕ ЛИЦ НА ОСНОВЕ ДЕТЕКТОРА SSD (SINGLE-SHOT MULTIBOX DETECTION)	58
1.7.1. Распознавание лиц в изображениях на основе детектора SSD (Single-Shot multibox Detection)	59
1.7.2. Распознавание лиц в видео файлах на основе детектора SSD (Single-Shot multibox Detection)	63
1.7.3. Распознавание лиц в видео потоке с видеокамеры на основе детектора SSD (Single-Shot multibox Detection)	67
1.8. РАСПОЗНАВАНИЕ ЛИЦЕВЫХ ОРИЕНТИРОВ С БИБЛИОТЕКОЙ DLIB	70
1.8.1. Распознавание лицевых ориентиров в изображениях с библиотекой Dlib	74
1.9. ДИАГРАММА ВОРОНОГО И ТРИАНГУЛЯЦИЯ ДЕЛОНЕ	76
1.9.1. Создание диаграммы Вороного и триангуляции Делоне для лица	78
1.10. КРАТКИЕ ИТОГИ	80
ГЛАВА 2	82
СТИЛИЗАЦИЯ ИЗОБРАЖЕНИЙ С ИСПОЛЬЗОВАНИЕМ НЕЙРОННЫХ СЕТЕЙ	82
2.1. СТИЛИЗАЦИЯ ИЗОБРАЖЕНИЙ С БИБЛИОТЕКОЙ OPENCV И ПРЕДВАРИТЕЛЬНО ОБУЧЕННЫХ НЕЙРОННЫХ СЕТЕЙ	82
2.1.1. Базовый модуль Python быстрого переноса нейронного стиля для настольных приложений	83
2.1.2. Использование функции быстрого переноса нейронного стиля в web приложениях.....	84
2.1.3. Модуль стилизации видеопотока с web камеры в настольных приложениях	86
2.1.4. Модуль создания видео файла с плавным изменением стиля изображения в настольных приложениях.....	88
2.1.5. Использование модуля плавного переноса нейронного стиля в web приложениях.....	89

2.2. СТИЛИЗАЦИЯ ИЗОБРАЖЕНИЙ С ИСПОЛЬЗОВАНИЕМ БИБЛИОТЕКИ	
PYTORCH И НЕЙРОННЫХ СЕТЕЙ	91
2.2.1. Базовый модуль Python для переноса нейронного стиля	91
2.2.2. Использование базового модуля переноса нейронного стиля в web приложениях ..	92
2.3. КРАТКИЕ ИТОГИ	95
ГЛАВА 3	96
ИНТЕГРАЦИЯ НЕЙРОННЫХ СЕТЕЙ С TELEGRAM	
БОТАМИ НА PYTHON	96
3.1. СОЗДАНИЕ TELEGRAM БОТА.....	97
3.2. ПРОСТЕЙШИЙ TELEGRAM БОТ «ПОПУГАЙ»	100
3.3. ИНТЕГРАЦИЯ ТЕЛЕГРАММ БОТА С НЕЙРОННЫМИ СЕТЯМИ	101
3.4. ИСПОЛЬЗОВАНИЕ КНОПОК В TELEGRAM БОТАХ	106
3.5. БОТ – ХУДОЖНИК ДЛЯ СТИЛИЗАЦИИ ФОТОГРАФИЙ РАЗНЫМИ СТИЛЯМИ	
(СИНХРОННЫЙ РЕЖИМ)	112
3.6. БОТ – ХУДОЖНИК ДЛЯ СТИЛИЗАЦИИ ФОТОГРАФИЙ (АСИНХРОННЫЙ	
РЕЖИМ)	119
3.7. КРАТКИЕ ИТОГИ	131
ГЛАВА 4	132
ОБРАБОТКА ИЗОБРАЖЕНИЙ С ИСПОЛЬЗОВАНИЕМ	
НЕЙРОННЫХ СЕТЕЙ	132
4.1. БЕСШОВНОЕ КЛОНИРОВАНИЕ С ИСПОЛЬЗОВАНИЕМ OPENCV	132
4.2. ТРАНСФОРМАЦИЯ ЧЕРНО-БЕЛЫХ ИЗОБРАЖЕНИЙ В ЦВЕТНЫЕ С	
ИСПОЛЬЗОВАНИЕМ НЕЙРОННЫХ СЕТЕЙ	136
4.2.1. Класс для трансформации черно-белых изображений в цветные	137
4.2.2. Трансформации черно-белых изображений в цветные в веб приложениях	138
4.3. ТРАНСФОРМАЦИЯ ЧЕРНО-БЕЛЫХ ВИДЕО ФАЙЛОВ В ЦВЕТНЫЕ С	
ИСПОЛЬЗОВАНИЕМ НЕЙРОННЫХ СЕТЕЙ	140
4.3.1. Класс для трансформации черно-белых видео файлов в цветные.....	140
4.3.2. Трансформации черно-белых видео файлов в цветные в веб приложениях	142
4.4. ПРИМЕНЕНИЕ МЕТОДОВ ДИЗЕРИНГА ДЛЯ ОБРАБОТКИ ИЗОБРАЖЕНИЙ	145
4.4.1. Функция для дизеринга изображений	145
4.4.2. Применение методов дизеринга для обработки изображений в веб приложениях ..	147
4.5. ГЕНЕРАЦИЯ ИЗОБРАЖЕНИЙ НА KERASCV С ПОМОЩЬЮ STABLE	
DIFFUSION	148
4.5.1. Функция для генерации изображений с помощью Stable Diffusion	149
4.5.2. Генерация изображений с помощью Stable Diffusion в веб приложениях	150
4.6. КРАТКИЕ ИТОГИ	153
ГЛАВА 5	154
ОБРАБОТКА PDF ДОКУМЕНТОВ В ПРОГРАММАХ НА	
PYTHON.....	154
5.1. БИБЛИОТЕКИ ДЛЯ РАБОТЫ С PDF ФАЙЛАМИ	154

5.2. Модуль РYТНОН для создания PDF файлов с библиотекой REPORTLAB	155
5.3. Загрузка PDF файлов из представлений в шаблоны DJANGO.....	159
5.4. Создание PDF файлов в приложениях на DJANGO с библиотекой REPORTLAB	161
5.5. Модуль РYТНОН для чтения PDF файлов с библиотекой PyPDF2	163
5.6. Модуль РYТНОН для формирования многостраничного файла PDF с использованием списков.....	167
5.7. Формирование многостраничного файла PDF с использованием списков в приложениях на DJANGO	170
5.8. Модуль РYТНОН для формирования многостраничного файла PDF с использованием класса CANVAS	173
5.9. Модуль РYТНОН для формирования многостраничного файла PDF с текстом и рисунками	176
5.10. Формирование многостраничного файла PDF с текстом и рисунками в веб приложениях на DJANGO.....	179
5.11. Модуль РYТНОН для защиты файла PDF паролем с библиотекой PyPDF2	182
5.12. Защита PDF файлов паролем в приложениях на DJANGO	185
5.13. Программная дешифровка PDF файлов в приложениях на DJANGO.....	188
5.14. Дешифровка PDF файлов путем ввода пароля в шаблоне DJANGO	189
5.15. Краткие итоги	191
ГЛАВА 6	192
МОДУЛИ РYТНОН для распознавания речи и перевода текстов.....	192
6.1. Распознавание речи и обработка аудио файлов на РYТНОН ...	193
6.1.1. Библиотеки Python для распознавания речи.....	193
6.1.2. Установка библиотек для распознавания речи и обработки аудио файлов	194
6.1.3. Модуль Python для распознавания речи в аудио файлах	194
6.1.4. Модуль Python для распознавания речи с микрофона	196
6.2. ПРЕОБРАЗОВАНИЕ ТЕКСТА В РЕЧЬ С ПАКЕТОМ GTTS API	198
6.2.1. Библиотеки для преобразования текста в речь	198
6.2.2. Модуль Python для преобразования текста в речь	198
6.2.3. Преобразование текста в речь с библиотекой pyttsx3	201
6.3. Распознавание речи с микрофона с записью в текстовый файл	202
6.4. Перевод текстов на РYТНОН.....	205

6.4.1. Модуль Python перевода текстов с использованием пакета API Google Translate	205
6.4.2. Модуль Python для перевода текстов с библиотекой deep-translator	207
6.4.3. Модуль Python для перевода многостраничного файла PDF с библиотекой deep_translator.....	209
6.5. Модуль Python озвучивания текста PDF файлов с библиотекой PyPDF2.....	214
6.6. Модуль Python перевода и озвучивания текста PDF файлов с библиотекой GOOGLETRANS.....	217
6.7. Модуль Python перевода и озвучивания текста PDF файлов с библиотекой DEEP-TRANSLATOR	221
6.8. Голосовое управление компьютером	223
6.9. Краткие итоги	226
ГЛАВА 7	227
ПЕРЕВОД И ОЗВУЧИВАНИЕ PDF ФАЙЛОВ В ПРИЛОЖЕНИЯХ DJANGO	227
7.1. ПРЕОБРАЗОВАНИЕ ТЕКСТА В РЕЧЬ С БИБЛИОТЕКОЙ DJANGO-GTTS ..	227
7.2. ПРЕОБРАЗОВАНИЕ ТЕКСТА В РЕЧЬ С БИБЛИОТЕКОЙ GTTS	230
7.3. ОЗВУЧИВАНИЕ PDF ФАЙЛОВ С ПАКЕТОМ DJANGO-GTTS	232
7.4. ОЗВУЧИВАНИЕ ТЕКСТА PDF ФАЙЛОВ С БИБЛИОТЕКОЙ GTTS	234
7.5. ПЕРЕВОД И ОЗВУЧИВАНИЕ PDF ФАЙЛОВ С GOGLETRANS И DJANGO-GTTS	236
7.6. ПЕРЕВОД И ОЗВУЧИВАНИЕ PDF ФАЙЛОВ С GOOGLETRANS И GTTS	240
7.7. ПЕРЕВОД И ОЗВУЧИВАНИЕ МНОГОСТРАНИЧНЫХ PDF ФАЙЛОВ С ПАКЕТОМ DJANGO-GTTS	243
7.8. ПЕРЕВОД И ОЗВУЧИВАНИЕ МНОГОСТРАНИЧНОГО ТЕКСТА PDF ФАЙЛОВ С БИБЛИОТЕКОЙ GTTS	248
7.9. ПЕРЕВОД И ОЗВУЧИВАНИЕ ТЕКСТА PDF ФАЙЛОВ С ПЕРЕНОСОМ ИЗОБРАЖЕНИЙ С БИБЛИОТЕКОЙ GTTS.....	254
7.10. Краткие итоги	263
ГЛАВА 8	265
ТРАНСКРИБАЦИЯ МЕДИА КОНТЕНТА В ПРИЛОЖЕНИЯХ НА PYTHON И DJANGO	265
8.1. Библиотеки Python для транскрипции медиа контента	265
8.2. Модуль Python для транскрипции аудио контента	267
8.3. Транскрипция аудио контента в приложениях на Django.....	269
8.4. Модуль Python для транскрипции видео контента.....	271
8.5. Транскрипция видео контента в приложениях на Django.....	274
8.6. Краткие итоги	276
ПОСЛЕСЛОВИЕ.....	278
СПИСОК ИСТОЧНИКОВ И ЛИТЕРАТУРЫ	279

ПРИЛОЖЕНИЕ 1. ЛИСТИНГ 2.13 ПРОГРАММНОГО МОДУЛЯ NET.PY (ВЫЗЫВАЕТСЯ ИЗ МОДУЛЯ M_S_21.PY)	281
ПРИЛОЖЕНИЕ 2. ЛИСТИНГ 3.2 (КОД МОДУЛЯ M_S_21.PY С ВКЛЮЧЕННЫМ В НЕГО КОДОМ МОДУЛЯ NET)	282
ПРИЛОЖЕНИЕ 3. ЛИСТИНГ 3.5 (МОДУЛЬ BOT_BUTON_1.PY)	283
ПРИЛОЖЕНИЕ 4. ЛИСТИНГ 3.6 (МОДУЛЬ ASIN_BOT.PY)	284

Постолит А.В.

Компьютерное зрение и искусственный интеллект в приложениях на Python и Django (интерактивная цифровая книга). – М.: Postolit Press, 2025. -408 с.: 294 ил.

Интерактивная цифровая книга предназначена для практикующих программистов, имеющих опыт работы с Python и Django. В книге раскрываются возможности использования методов компьютерного зрения и нейронных сетей для обработки медиа контента, для распознавания объектов в изображениях и видео потоке. Книга знакомит с методами синтеза и распознавания речи, перевода текстовой информации и содержимого PDF документов на различные языки с озвучиванием переведенного текста. Приводятся примеры создания телеграмм ботов, их интеграции с нейронными сетями в синхронном и асинхронном режиме работы программных модулей. Рассмотрено применение генеративных сетей для создания изображений на основе их текстового описания.

В книге читатель найдет множество примеров программного кода, который можно не только просмотреть и скачать, но и выполнить в рамках самой интерактивной цифровой книги, так как она содержит весь необходимый инструментарий для выполнения программных модулей с отображением результатов их работы в окне интернет браузера. Все примеры программного кода можно скачать и использовать в своих проектах.

Книга предназначена для программистов, имеющих опыт работы с **Python и Django**. Она может быть использована как лабораторный практикум для студентов ВУЗов и слушателей компьютерных курсов при изучении практических приемов программирования приложений на Python и Django.

Предисловие

Специалисты IT-технологий уже много лет занимают первые позиции в рейтингах кадровых агентств, как представители наиболее востребованных профессий на отечественном и мировом рынке труда. Постоянная нехватка квалифицированных программистов дает высокий шанс молодым и целеустремленным новичкам для входа в эту профессию. В ближайшие несколько лет дефицит IT-специалистов будет только нарастать, так как они нужны повсеместно. Если раньше ими интересовались сугубо профильные компании (разработчики софта и онлайн-платформ), то сейчас квалифицированные кадры IT-профилей требуются практически везде. И речь здесь идет не о дефиците, а о катастрофическом дефиците. Уже много лет подряд сервис по поиску работы hh.ru фиксирует огромный неудовлетворенный спрос на IT-специалистов.

Значительный прогресс в развитии систем компьютерного зрения и искусственного интеллекта привел к проникновению этих технологий практически во все области повседневной жизни. Медицина, транспорт, промышленность, изобразительное и музыкально искусство, сельское хозяйство, органы безопасности и МЧС, экономика, образование — это далеко не полный перечень сфер, работу которых уже трудно представить без систем на основе искусственного интеллекта. Возможности машинного зрения на основе искусственного интеллекта не ограничиваются обработкой статичных изображений. Видео аналитика на базе искусственного интеллекта теперь может отслеживать движение беспилотных автомобилей, поездов и летательных аппаратов, на производстве такие системы контролируют качество комплектующих деталей и работу сборочных линий.

Язык программирования Python с его обширными библиотеками является сам популярным и востребованным языком, на котором создаются системы машинного зрения и искусственного интеллекта. Он используется при машинном обучении, реализации нейронных сетей, для написания веб-сайтов и мобильных приложений. В целом это универсальный язык, входящий в тройку языков для анализа больших данных. Еще одна причина, по которой следует обратить внимание на язык Python, это его использования для веб-разработки совместно с фреймворками Django и Bootstrap. Здесь Django обеспечивает разработку серверной части, а Bootstrap — клиентской части веб-приложений. На Django разработаны такие сайты, как YouTube, Google, Instagram, а Bootstrap это детище компании Twitter. С использованием Django можно практически «бесшовно сшивать» между собой программные модули, написанные на Python, с нейронными сетями, что обеспечивает возможность использование элементов искусственного интеллекта в сети интернет

Эта книга предназначена не для начинающих программистов, а для специалистов с опытом, которые уже занимались разработкой приложений с использованием Python и Django. Для начинающих программистов сначала нужно получить базовые знания, изучив материалы книги «Python, Django и Bootstrap для начинающих» (Анатолий Постоит, «БХВ-Петербург», 2023 г.), «Мастер веб разработки на Python, Django и Bootstrap» (Анатолий Постоит, «БХВ-Петербург», 2025 г.). Сразу следует отметить, что веб-программирование требует от разработчиков больших знаний, умений и усилий, чем программирование традиционных приложений. Здесь, кроме основного языка, который реализует логику приложения, необходимо знание структуры HTML-документов, основ работы с базами данных, принципов взаимодействия удаленных пользователей с веб-приложением.

Практикующие программисты уверяют, что самый правильный, а главное практически бесплатный способ повысить квалификацию в области программированию — это освоить но-

вовведения самостоятельно. Так можно не только избежать серьезных расходов, но и получить только те практические навыки, которые пригодятся в будущей работе. Данная книга как раз и позволит самостоятельно освоить приемы интеграции нейронных сетей с настольными приложениями на Python и с веб приложениями на Django.

Некоторые фрагменты программного кода данной книги могут показаться сложными, но они взяты из проверенных и работающих приложений. В них гарантированно нет ошибок, использованы приемы большого количества действующих программистов, о которых ничего не сказано в официальной документации. Абсолютно все примеры интерактивной цифровой книги можно запустить на выполнение в рамках самой книги, так как она содержит весь необходимый инструментарий. Когда вы увидите результаты работы программного кода, появится ясность — как работает тот или иной элемент изучаемого программного модуля. Мало того, зарегистрированные пользователи могут скачать листинги программных кодов интерактивной цифровой книги и использовать их в своих проектах.

Книга, как уже отмечалось, предназначена для программистов, имеющих практику создания приложений на Python, Django и Bootstrap. В этом издании раскрываются некоторые особенности интеграции библиотек компьютерного зрения и нейронных сетей модулями Python и веб приложениями на Django и Bootstrap.

Первая глава книги посвящена основам компьютерного зрения. Из этой главы вы узнаете, как компьютер хранит и обрабатывает черно-белые и цветные изображения, какие библиотеки используются для обработки изображений, видео файлов и потокового видео. Как в шаблоны Django можно вывести изображения, видео файлы и потоковое видео. Как можно реализовать распознавание объектов на основе нейронных сетей в изображениях, видео файлах и потоковом видео. Что такое лицевые ориентиры, и как можно выполнить разбивку поверхности изображения на области с использованием диаграммы Вороного и триангуляцией Делоне. Как в приложениях на Django можно использовать API MediaStream для вывода медиа потока с видео камер в интернет браузер.

Вторая глава книги посвящена описанию возможностей стилизаций изображений с использованием нейронных сетей. Из материалов данной главы вы узнаете, как создать базовый модуль стилизации изображений на Python и отладить его работу на локальном компьютере, а также как подключить этот модель к веб приложению. Как можно создать видеофайл с плавным изменением стиля изображения с реализацией этого подхода в веб приложениях. Как можно использовать библиотеку PyTorch и модель нейронной сети 21styles.model для переноса любого стиля на любое изображение с использованием методов машинного обучения.

В *третьей главе* рассматриваются технологии создания и использования телеграмм ботов на Python. Как можно интегрировать телеграмм бот с нейронными сетями. Научитесь создавать кнопки, которые облегчат взаимодействие пользователей с телеграмм ботом. А также узнаете, как создать бота художника для стилизации изображений с использованием синхронного и асинхронного режима взаимодействия программных модулей с нейронными сетями.

Четвертая глава посвящена особенностями обработки изображений с использованием нейронных сетей. Вы узнаете, что такое бесшовное клонирование изображений, обеспечивающее объединение нескольких изображений в одно целое. Как можно черно-белое изображение (например, старое фото) превратить в цветное изображение, или как можно калоризовать (раскрасить) черно-белый видео файл. Как с использованием дизеринга можно цветное изображение трансформировать в черно-белое изображение (нарисованное точка-

ми). А так же научитесь генерировать изображения на основе их текстового описания с использованием машинного обучения на основе библиотеки PyTorch.

В *пятой главе* приведены примеры обработки PDF документов. Известно, что в сети интернет основная масса документов хранится в PDF формате, а для обработки содержимого этих документов, например, нейронными сетями, нужно сначала автоматизировать процесс извлечения из этих документов текста и рисунков. Из данной главы вы узнаете, какие библиотеки Python используются для обработки PDF файлов. Научитесь извлекать данные из PDF файлов, выводить PDF документы в шаблоны Django. Узнаете, как программным способом можно защитить PDF документы паролем (зашифровать) и открыть зашифрованный документ в шаблоне Django.

В *шестой главе* приведен базовый набор программных модулей Python для распознавания речи и перевода текстов с подключением открытых облачных сервисов и нейронных сетей. Вы узнаете, какие библиотеки Python используются для распознавания речи в аудио файлах и аудио потоке с микрофона. Как можно преобразовать текст в речь с пакетами gTTS API и pyttsx3. Как можно реализовать распознавание речи с микрофона с записью результатов в текстовый файл. Научитесь выполнять перевод текстов с использованием пакета API Google Translate и библиотекой deep-translator. Узнаете, как можно озвучить тексты, извлеченные из PDF документов, а также как можно реализовать голосовое управление компьютером.

В *седьмой главе* приведены материалы, связанные с переводом и озвучиванием PDF файлов в веб приложениях. Из материалов данной главы вы узнаете, как преобразовать текстовую переменную в речь с пакетами Django-Gtts и gTTS. Как озвучить текст PDF файлов с пакетами Django-gTTS и gTTS. Научитесь выполнять перевод и озвучивание текста PDF файлов с библиотекой gogletrans, пакетом Django-Gtts и gTTS. Узнаете, как можно выполнять перевод и озвучивание текстов PDF файлов с переносом изображений.

На протяжении всей книги раскрываемые вопросы сопровождаются упрощенными, но полностью законченными примерами. Ход решения той или иной задачи показан на большом количестве иллюстративного материала. Желательно изучение разделов книги осуществлять, сидя непосредственно за компьютером, — тогда вы сможете увидеть результаты работы любого примера книги в браузере вашего компьютера. А затем, последовательно повторять выполнение тех шагов, которые описаны в примерах, и тут же видеть результаты своих действий на своем локальном компьютере, что в значительной степени облегчит восприятие материала книги. Наилучший способ обучения — это практика. Все листинги программ приведены на языке Python с элементами java скриптов, а шаблоны веб-страниц — в виде HTML-файлов. Любой пример, приведенный в интерактивной цифровой книге, можно запустить на выполнение на публичном сервере и посмотреть, как выглядит конечный результат. Книга содержит 294 рисунка и 343 листинга программного кода. Программный код можно скачать и использовать в своих проектах.

Примечание.

Работать с интерактивной цифровой книгой можно по следующему адресу gumnik.ru. Демо-версию книги можно найти в главном меню в опции «Каталог». Полная версия книги доступна только для зарегистрированных пользователей после ее приобретения в магазине сайта. Книга будет доступна в личном кабинете, в разделе кабинета «Мои книги».

Итак, если вас заинтересовали вопросы использования элементов компьютерного зрения и нейронных сетей в приложениях на Python и Django, то самое время перейти к изучению материалов этой цифровой интерактивной книги.

ГЛАВА 1

Основы компьютерного зрения



Компьютерное зрение (computer vision) или машинное зрение - это набор программно-технических средств, позволяющих компьютерам понимать и обрабатывать визуальную информацию, такую как изображения или видео. Являясь составной частью систем искусственного интеллекта, системы компьютерного зрения позволяют не только читать, хранить и обрабатывать изображения, но и позволяют обучить компьютеры принимать решения на основе увиденного. Построенные на основе компьютерного зрения модели помогают смартфонам узнавать своих владельцев, дорожным камерам распознавать номера автомобилей, медикам диагностировать болезни, а роботам и беспилотным автомобилям объезжать препятствия.

Перед тем как перейти к решению практических задач, нужно разобраться с некоторыми теоретическими аспектами, в частности как компьютер получает, хранит и обрабатывает различные виды изображений. Из материалов данной главы вы узнаете:

- ☐ что такое матрица изображения;
- ☐ как компьютер хранит и обрабатывает черно-белые изображения, а вернее изображения в оттенках серого цвета;
- ☐ как компьютер хранит и обрабатывает цветные (трехканальные) изображения;
- ☐ как компьютер хранит и обрабатывает четырехканальные изображения;
- ☐ какие библиотеки используются для обработки изображений, видео файлов и потокового видео;
- ☐ как в шаблоны Django вывести изображение, видео файл и потоковое видео;
- ☐ как можно реализовать распознавание объектов на основе каскадов Хаара в изображениях, видео файлах и потоковом видео;
- ☐ как можно реализовать распознавание объектов на основе нейронных сетей в изображениях, видео файлах и потоковом видео;
- ☐ как можно реализовать распознавание лицевых ориентиров на основе библиотеки dlib;
- ☐ как можно выполнить разбивку поверхности изображения на области с использованием диаграммы Вороного и триангуляцией Делоне;
- ☐ как в приложениях на Django можно использовать API MediaStream для вывода медиа потока с видео камер.

1.1. Пиксели и цветовые пространства

Каждое изображение состоит из набора пикселей. Пиксель — это основной строительный блок изображения. Если представить изображение в виде сетки, то каждый квадрат в сетке

содержит один пиксель, где точке с координатой (0, 0) соответствует верхний левый угол изображения. Иными словами любое цифровое изображение — это матрица (а в случае Python массив NumPy), в которой положение пикселей (составных частей изображения) задано координатами (рис.1.1).

(0, 0)	(0, 1)	(0, 2)
(1, 0)	(1, 1)	(1, 2)
(2, 0)	(2, 1)	(2, 2)

Рис. 1.1. Матрица изображения с указанием координат каждого пикселя

Цифровое значение, расположенное в каждой ячейки такой матрицы, будет соответствовать определенному оттенку цвета. Сочетание пикселей разных оттенков в такой матрице и создает цифровое изображение. Представим, что у нас есть изображение с разрешением 400x300 пикселей. Это означает, что наша сетка (матрица) состоит из 400 строк (h) и 300 столбцов(w). В совокупности такое изображение будет состоять из $400 \cdot 300 = 120000$ пикселей.

Черно-белое изображение. В случае черно-белого изображения, или правильнее сказать, изображения в оттенках серого (grayscale image), речь идет об одном слое матрицы, в каждой ячейки которой содержится определенное число. В изображениях в оттенках серого каждый пиксель имеет значение в интервале от 0 до 255, где 0 соответствует чёрному цвету, а 255 соответствует белому цвету. А значения между 0 и 255 принимают различные оттенки серого, где значения ближе к 0 более тёмные, а значения ближе к 255 более светлые (рис.1.2).



Рис. 1.2. Числовые значения пикселей оттенков серого цвета в однослойной матрице изображений

Схематично однослойная матрица изображения в оттенках серого представлена на рис.1.3.

	0	1	2
0	0...255	0...255	0...255
1	0...255	0...255	0...255
2	0...255	0...255	0...255

Рис. 1.3. Матрица изображения в оттенках серого с указанием интервала значения цвета каждого пиксела

Таким образом, любое изображение можно представить в виде двумерной матрицы, в которой значения яркости каждого пикселя представлено некоторым числом. Пример такой матрицы представлен на рис. 1.4.

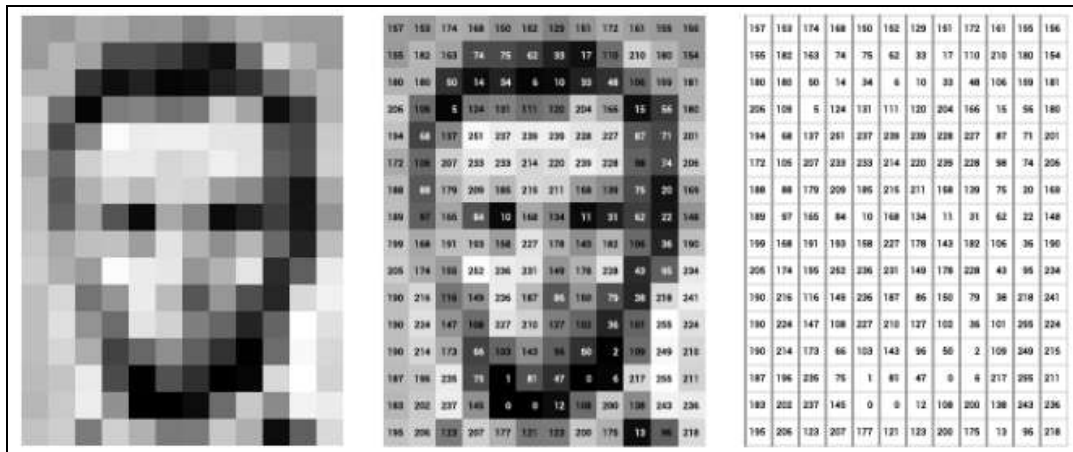


Рис. 1.4. Пример представления яркости пикселей изображения в виде матрицы

Цветное изображение RGB. Цветное изображение состоит не из одной, а из трех матриц, наложенных друг на друга. Иными словами у цветного изображения (color image) таких матриц, или правильнее говорить каналов (channels), три: красный, зелёный и синий (red, green, blue) или RGB). Схематично структура трехканального цветного изображения представлена на рис. 1.5.

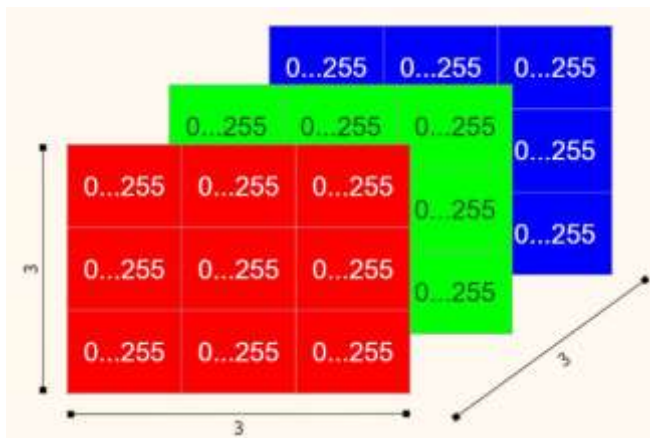


Рис. 1.5. Матрица 3-х канального цветного изображения с указанием интервала значения цвета каждого пиксела

Комбинация интенсивности этих цветов и определяет итоговый цвет (оттенок) каждого пикселя. Насыщенность цвета каждой ячейки матрицы представлена целым числом в диапазоне от 0 до 255 включительно. Исходя из того, что каждая компонента представлена в диапазоне $[0, 255]$, то для того, чтобы представить насыщенность каждого цвета, нам будет

достаточно 8-битного целого без знакового числа. Затем мы объединяем значения всех трех компонент в кортеж вида (красный, зеленый, синий). К примеру, чтобы получить белый цвет, каждая из компонент должна равняться 255 (255, 255, 255). Тогда, чтобы получить черный цвет, каждая из компонент должна быть равной 0 (0, 0, 0). Некоторые распространенные цвета, представленные в виде RGB кортежей, приведены на рис.1.6.

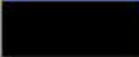
Название	Код RGB	Цвет
Красный	(255, 0, 0)	
Зеленый	(0, 255, 0)	
Синий	(0, 0, 255)	
Черный	(0, 0, 0)	
Белый	(255, 255, 255)	
Желтый	(255, 255, 0)	
Оливковый	(128, 128, 0)	
Фиолетовый	(128, 0, 128)	

Рис. 1.6. Примеры различных цветов пикселей в формате RGB

Цветное изображение RGBA. Цветное изображение RGBA — это по сути та же трехслойная матрица RGB, которая имеет дополнительный слой или канал (альфа-канал) для формирования уровня прозрачности изображения. Цвет RGBA состоит из четырех каналов, где каждый канал представлен 8 битами. В RGBA информация о прозрачности хранится в альфа-канале, а информация о цвете хранится в трех каналах RGB.

Альфа-канал может быть представлен изображением в оттенках серого. Это означает, что альфа-канал состоит из 256 уровней прозрачности, где 0 представляет максимальный уровень прозрачности, а 255 представляет максимальный уровень непрозрачности. Пример изображения с альфа-каналом в виде шахматной доски представлен на рис. 1.7.



Рис. 1.7. Пример 4-х канального цветного изображения RGBA

Из данного рисунка видно, что в верхней его части альфа-канал является не прозрачным, а в нижней части полностью прозрачным. Добавление альфа-канала к традиционной модели RGB значительно расширяет творческие возможности. В отличие от RGB, которая может создавать только сплошные цвета, RGBA может создавать такие эффекты, как прозрачность и полупрозрачность. Это особенно важно в веб-дизайне и разработке программного обеспечения, где решающее значение имеет возможность накладывать изображения, создавать градиентные эффекты и разрабатывать визуально привлекательные интерфейсы с полупрозрачными элементами. Альфа-канал эффективно позволяет изображению сливаться с фоном или другими изображениями, обеспечивая плавную интеграцию.

1.2. Библиотеки для обработки изображений, видео файлов и видео потоков

Для обработки изображений в виртуальную среду проекта Django необходимо установить несколько специализированных библиотек:

```
pip install opencv-python
pip install pillow
pip install numpy
```

Эти библиотеки используются для обработки изображений и кадров видеопотоков и видео файлов, где каждый кадр, по сути, является изображением. Импорт необходимых библиотек выполним в самых первых строках модуля `views.py` (листинг 1.1).

Листинг 1.1. Импорт библиотек и пакетов для обработки изображений

Примечание.

В демо-версии книги удалены листинги программ.

Так как данные пакеты будут использоваться в различных представлениях `views`, то мы их импорт в самом начале программного модуля `views.py`. Кроме того здесь было задано несколько глобальных переменных, которые будут доступны для всей группы представлений:

- ❑ `base_dir = settings.BASE_DIR` – путь к базовой директории проекта;
- ❑ `media_dir = settings.MEDIA_ROOT` - путь к файлам в папке `media`;
- ❑ `static_dir = settings.STATICFILES_DIRS` - путь к файлам в папке `static`;
- ❑ `media_url = settings.MEDIA_URL` – URL адрес к файлам в папке `media`.

Эти глобальные переменные будут использованы в представлениях для загрузки различных изображений и видеофайлов.

1.3. Чтение изображений из файлов с выводом в шаблоны Django

Для начала напишем простой программный модуль, который позволяет прочитать изображение из файла, получить его параметры и показать в шаблоне Django. Для этого будем использовать библиотеку `OpenCV`.

На первом шаге в одном из шаблонов или элементе меню необходимо создать ссылку, которая активирует нужный URL маршрут (листинг 1.2).

Листинг 1.2. Ссылка активации маршрута `show_image` для вывода изображения

Примечание.

В демо-версии книги удалены листинги программ.

По данной ссылке в приложении `api_vision` будет активирован URL маршрут с именем `show_image` (листинг 1.3).

Листинг 1.3. Маршрут URL с именем `show_image`

Примечание.

В демо-версии книги удалены листинги программ.

По этому маршруту будет активировано представление `show_image` (листинг 1.4).

Листинг 1.4. Представление (view) `show_image`

Примечание.

В демо-версии книги удалены листинги программ.

Здесь в строковый объект `image_path` помещаем путь к файлу с изображением, для чего используем модуль `Path`. Необходимость использования данного модуля вызвано тем, что в разных операционных системах (`Windows` и `Linux`) для указания директории с нужным файлом используются разные слешы («\» или «/»). Модуль `Path` сам определяет тип операционной системы и формирует путь к файлу с использованием «правильных» слешей. Та-

ким образом, если вы разрабатывали проект под Windows на локальном компьютере, то он будет работать без ошибок и на сервере под Linux.

На следующем шаге в объект `image` загружается изображение из файла с именем `Moto.jpg`, который находится в папках проекта `media/images`. Для чтения файла с диска используется метод `imread` библиотеки `OpenCV`. Этот объект нам нужен только для того, чтобы получить и показать параметры загруженного изображения (высоту `h`, ширину `w` и цветность или количество слоев `d`).

Для того чтобы показать изображение в шаблоне Django здесь создан другой объект - `img_url`. В данном объекте прописан путь к файлу с изображением - `'/media/images/Moto.jpg'`.

Затем в словарь `context` загружаются полученные параметры: `«img_url»`, `«h»`, `«w»` и `«d»`, а сам словарь передается в шаблон с именем `show_image.html`. Этот шаблон имеет следующий код (листинг 1.5).

Листинг 1.5. Шаблон `show_image.html`

Примечание.

В демо-версии книги удалены листинги программ.

В данном шаблоне создан контейнер, в теле которого находятся строки и колонки для вывода текстовой информации и самого изображения. В интерактивной цифровой книге можно посмотреть, как будет показано изображение в шаблоне `show_image.html`.

Примечание.

В интерактивной цифровой книге для запуска данного примера нужно на вкладке «Показ примеров» нажать на кнопку «Примеры главы 1» и активировать ссылку «Показать изображение».

В меню на сервере – Глава 1/Вывод изображения в шаблон

Страница на сервере - http://kalinkashop.tw1.ru/api_vision/show_image

Внешний вид шаблона `show_image.html` с загруженным изображением представлен на рис.1.8.

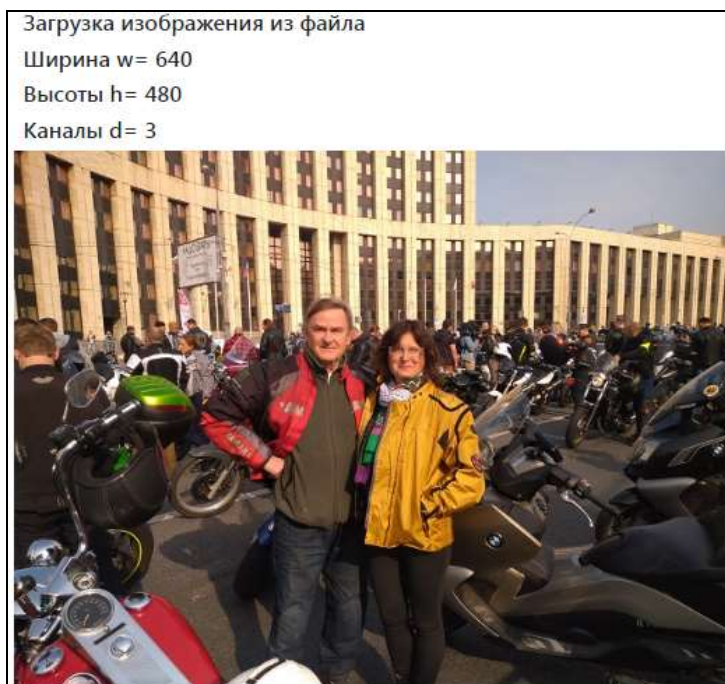


Рис. 1.8. Пример вывода 3-х канального цветного изображения из файла в шаблон Django

1.4. Чтение видео файлов с выводом в шаблоны Django

Для начала напишем простой программный модуль, который позволяет прочитать видео файл, получить его параметры и показать в шаблоне Django. Для этого будем использовать библиотеку OpenCV.

На первом шаге в одном из шаблонов или элементе меню необходимо создать ссылку, которая активирует нужный URL маршрут (листинг 1.6).

Листинг 1.6. Ссылка активации маршрута `show_video` для вывода видео файла

Примечание.

В демо-версии книги удалены листинги программ.

По данной ссылке в приложении `api_vision` будет активирован URL маршрут с именем `show_video` (листинг 1.7).

Листинг 1.7. Маршрут URL с именем `show_video`

Примечание.

В демо-версии книги удалены листинги программ.

По этому маршруту будет активировано представление `show_video` (листинг 1.8).

Листинг 1.8. Представление (view) `show_video`

Примечание.

В демо-версии книги удалены листинги программ.

Здесь в строковый объект `video_path` помещаем путь к видео файлу, для чего используем модуль `Path`. Необходимость использования данного модуля вызвано тем, что в разных операционных системах (`Windows` и `Linux`) для указания директории с нужным файлом используются разные слешы («\» или «/»). Модуль `Path` сам определяет тип операционной системы и формирует путь к файлу с использованием «правильных» слешей. Таким образом, если вы разрабатывали проект под `Windows` на локальном компьютере, то он будет работать без ошибок и на сервере под `Linux`.

На следующем шаге создается объект `sar`, который позволяет захватывать видео из разных источников и выполнять с ним различные задачи: читать, обрабатывать и записывать видео файлы на диск. В данном случае объект `sar` получает данные из видео файла с именем `Dictor.mp4`, который находится в папках проекта `media/video`. Затем, с использованием методов объекта `sar`, получаем следующие свойства видео файла:

- ❑ `w` - ширина кадра;
- ❑ `h` - высота кадра;
- ❑ `fps` - частота кадров;
- ❑ `total_frames` - количество кадров.

Для того чтобы проиграть видео файл в шаблоне `Django` здесь создан другой объект - `video_url`. В данном объекте прописан путь к видео файлу - `'/media/video/ Dictor.mp4'`.

Затем в словарь `context` загружаются полученные параметры: `«video_url»`, `«w»`, `«h»`, `«fps»`, и `«total_frames»`, а сам словарь передается в шаблон с именем `show_video.html`. Этот шаблон имеет следующий код (листинг 1.9).

Листинг 1.9. Шаблон `show_image.html`

Примечание.

В демо-версии книги удалены листинги программ.

В данном шаблоне создан контейнер, в теле которого находятся строки и колонки для вывода текстовой информации и проигрыватель для просмотра самого видео файла. В интерактивной цифровой книге можно посмотреть, как выглядит шаблон `show_image.html`.

Примечание.

В интерактивной цифровой книге для запуска данного примера нужно на вкладке «Показ примеров» нажать на кнопку «Примеры главы 1» и активировать ссылку «Показать видео файл».

Внешний вид шаблона `show_video.html` представлен на рис.1.9.

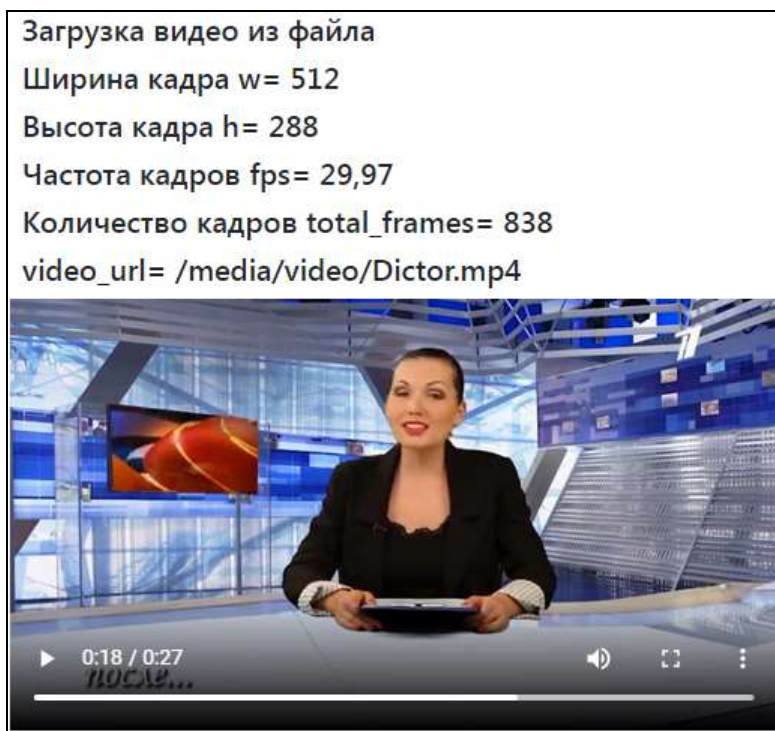


Рис. 1.9. Пример вывода видео файла в шаблон Django

1.5. Трансляция потоковых данных с видео камер в шаблоны Django

В Python и Django имеется несколько вариантов трансляции потоковых данных с видео камер в интернет браузер:

- ☐ вывод видеопотока в окно браузера с использованием API MediaStream;
- ☐ вывод видеопотока в шаблон Django с использованием API MediaStream;
- ☐ вывод видеопотока в шаблон Django из представления view.

Первые два метода позволяют просто показать потоковое видео на html странице, третий метод позволяет выполнить обработку кадров видео потока перед тем, как они будут показаны в окне браузера (например, выполнить распознавание объектов в потовом видео, стилизовать кадры, изменить цветность, яркость и т.п.).

1.5.1.Прямой вывод видеопотока в окно браузера с использованием API MediaStream

Чтобы подключиться к веб-камере и получить доступ к ее видеопотоку раньше приходилось использовать плагины, встроенные в интернет браузер. Это не давало возможность получить доступ к веб-камере без использования сторонних компонентов, например, на

мобильных устройствах. По крайней мере, так было до недавнего времени. Однако с появление API MediaStream и HTML5 для взаимодействия с веб-камерой на интернет страницу достаточно добавить минимальное количество кода на JavaScript. Хорошей новостью является то, что, различные браузеры уже поддерживают эту технологию доступа к веб-камере, и теперь достаточно просто видеопоток с веб-камеры вывести на html страницу.

Если на стороне клиента используется последние версии Google Chrome, то нужно учитывать, что было внесено изменение в систему безопасности, согласно которому доступ к веб-камере возможен только в том случае, если соединение выполнено через протокол HTTPS. Можно по-прежнему разрабатывать и тестировать приложения в локальном режиме, но вы не сможете протестировать их «в реальных условиях», если с публичным сервером отсутствует защищенное соединение HTTPS. Нужно также учесть, что разные браузеры по-разному запрашивают разрешение на использование веб-камеры.

При использовании данного технологического подхода задействовано всего два компонента, которые выполняют всю тяжелую работу по получению данных с веб-камеры и вывода видеопотока в браузер - это тег HTML страницы `<video>` и функция JavaScript с именем `getUserMedia` (рис.1.10).

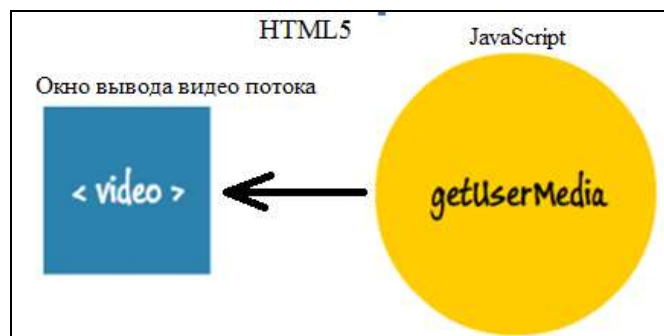


Рис. 1.10. Элементы API MediaStream

В этой схеме элемент `<video>` довольно прост, он отвечает только за прием и отображение видеопотока в окне браузера. Функция `getUserMedia` выполняет следующую работу: получает видеоданные с веб-камеры, получает аудиоданные с микрофона, или выполняет эти два действия одновременно. В теле данной функции с использованием инструкции `if` реализовано разветвление:

- ☐ если пользователь дал разрешение на доступ к веб-камере, и она подключена, то выполняется трансляция видеопотока:
- ☐ если пользователь не предоставляет разрешение на доступ к веб-камере или с подключением камеры имеются проблемы, то функция возвращает сведения о такой ошибке.

Функции `getUserMedia` можно указать, транслировать только видео поток, или необходимо дополнить его аудио данными.

Этот технологический подход выглядит очень уж простым только потому, что вся сложные моменты спрятаны от разработчика и работают «под капотом».

Начнем с создания простой HTML страницы с именем `cam_stream.html` (листинг 1.10).

Листинг 1.10. Шаблон `cam_stream.html`

Примечание.

В демо-версии книги удалены листинги программ.

В этом программном коде с использованием стилей CSS заданы параметры рамки или контейнера, в который будет выведен видеопоток. После этого следует тег HTML, в котором буде отображаться видеопоток с web камеры:

```
<div id="container">
  <video autoplay="true" id="videoElement"> </video>
</div>
```

Тег <video> имеет идентификатор id=videoElement, а для атрибута autoplay установлено значение true. Задание атрибуту autoplay значения true гарантирует, что видео изображение в данном теге начнет отображаться автоматически, как только появится видеопоток с веб-камеры.

Теперь проанализируем код JavaScript:

```
var video = document.querySelector("#videoElement");
if (navigator.mediaDevices.getUserMedia) {
  navigator.mediaDevices.getUserMedia({video: true, audio: true })
    .then(function (stream) {
      video.srcObject = stream;
    })
    .catch(function (error) {
      console.log("Что то пошло не так!");
    });
}
```

Разберем первую строку:

```
var video = document.querySelector("#videoElement");
```

Здесь была объявлена переменная video, и она через идентификатор связана с элементом video, который находится в коде HTML. Мы получаем доступ к элементу video, используя querySelector и указывая идентификатор id, который на него нацелен.

Далее следует код для доступа к API getUserMedia. Здесь сначала в инструкции if проверяется условие, поддерживает ли текущий браузер метод getUserMedia. Если браузер пользователя позволяет отображать потоковое видео, то активируется метод getUserMedia (получить медиа данные от пользователя), в который передаются параметры выводимых данных:

```
{video: true, audio: true }).
```

В этом словаре указаны ограничения, которые накладываются на пользовательские данные. В данном случае указано, что от пользователя можно получать видео поток (video:true) и аудио поток (audio:true).

Если видеочамера пользователя подключена и находится в рабочем состоянии, то иницируется трансляция медиа данных:

```
.then(function (stream) {
  video.srcObject = stream;
})
```

Если по какой-то причине происходит сбой в передаче медиа данных, то ошибка передачи данных обрабатывается и трансляция прерывается:

```
.catch(function (error) {  
    console.log("Something went wrong!");  
});
```

Если вам потребуется принудительная остановка трансляции с веб-камеры, то понадобится добавить код, который выглядит следующим образом:

```
function stop(e) {  
    var stream = video.srcObject;  
    var tracks = stream.getTracks();  
  
    for (var i = 0; i < tracks.length; i++) {  
        var track = tracks[i];  
        track.stop();  
    }  
  
    video.srcObject = null;  
}
```

На локальном компьютере корректность работы данного программного модуля можно проверить следующим образом:

- ☐ войдите в директорию, где находится файл с именем `cam_stream.html`;
- ☐ дважды кликните левой кнопкой мыши на имя данного файла.

Примечание.

В рамках интерактивной цифровой книги вы не можете выполнить данный пример, это можно сделать только на локальном компьютере. Однако вы можете скопировать код страницы `cam_stream.html` и запустить его на своем компьютере

Если видео камера не подключена или есть проблемы с передачей данных, то в окне браузера будет показана рамка с серой прямоугольной областью (рис. 1.11).

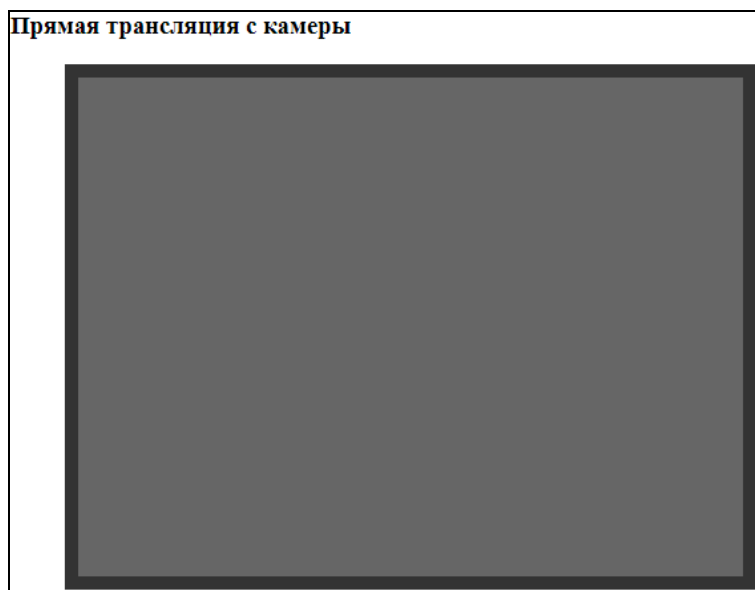


Рис. 1.11. Вид окна браузера при отсутствии видеокамеры

Если видео камера подключена и нет проблем с передачей медиа данных, то в окне браузера будет показана окно с предложением активировать (разрешить) трансляции медиа данных (рис.1.12).

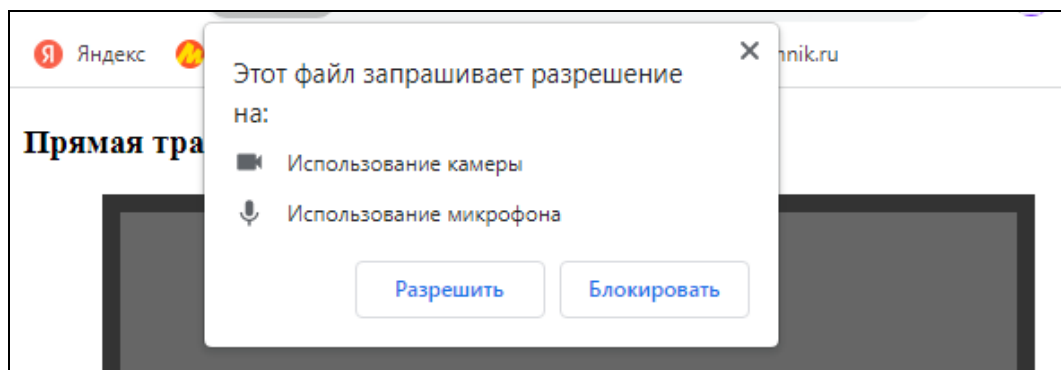


Рис. 1.12. Вид окна браузера с запросом на разрешение трансляции с видеокамеры

Если пользователь нажимает на кнопку «разрешить», то медиа поток с видеокамеры будет выведен в окно браузера (рис.1.13).

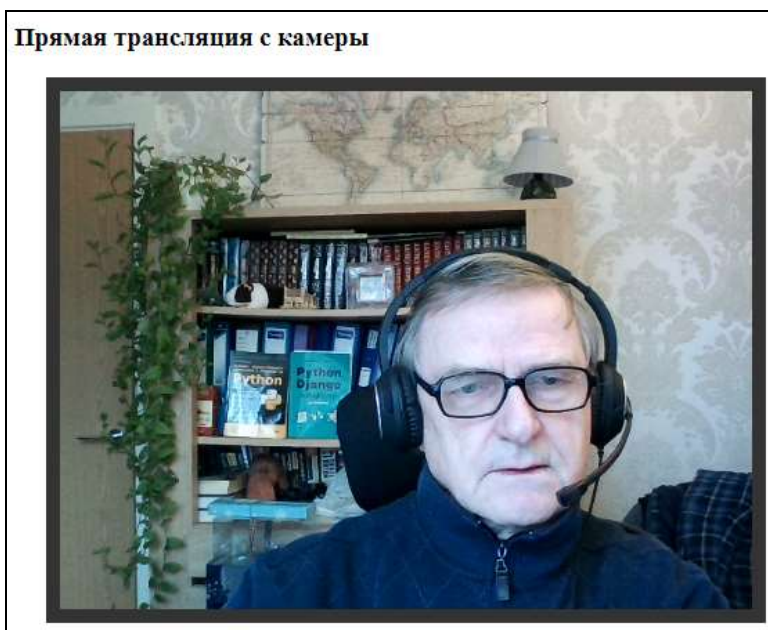


Рис. 1.13. Вид окна браузера в режиме трансляции медиа данных с видеокамеры

Как видно из данного рисунка, видео поток успешно выведен в окно браузера, кроме того происходит передача и аудио потока с микрофона.

1.5.2. Вывод видеопотока в шаблон Django с использованием API MediaStream

В предыдущем разделе была показана возможность использования API MediaStream для вывода медиа потока с видеокамеры в окно браузера из обычного HTML файла. Теперь реализуем пример вывода медиа потока с видеокамеры в шаблон Django с использованием API MediaStream. Для этого в одном из шаблонов или элементе меню необходимо создать ссылку, которая активирует нужный URL маршрут (листинг 1.11).

Листинг 1.11. Ссылка активации маршрута `cam_stream`

Примечание.

В демо-версии книги удалены листинги программ.

По данной ссылке в приложении `api_vision` будет активирован URL маршрут с именем `cam_stream` (листинг 1.12).

Листинг 1.12. Маршрут URL с именем `cam_stream`

Примечание.

В демо-версии книги удалены листинги программ.

По этому маршруту будет активировано представление `cam_stream` (листинг 1.13).

Листинг 1.13. Представление (view) `cam_stream`

Примечание.

В демо-версии книги удалены листинги программ.

Это достаточно простой программный код, в котором вызывается шаблон с именем `cam_stream.html`. Этот шаблон имеет следующий код (листинг 1.14).

Листинг 1.14. Шаблон `cam_stream.html`

Примечание.

В демо-версии книги удалены листинги программ.

Это достаточно простой шаблон, в котором имеется два контейнера: первый обеспечивает вывод текстовой строки, второй – вывод медиа контента. Кроме того, здесь в теге `<style>` заданы стили для данной HTML страницы, а в теге `<script>` находится код JavaScript, необходимый для работы с API MediaStream. Построчный анализ этого кода приведен в предыдущем разделе.

Примечание.

В интерактивной цифровой книге для запуска данного примера нужно на вкладке «Показ примеров» нажать на кнопку «Примеры главы 1» и активировать ссылку «Трансляция с камеры с API MediaStream». Эта опция может не работать на стороне сервера из-за отсутствия SSL сертификата.

Внешний вид шаблона `cam_stream.html` представлен на рис.1.14.

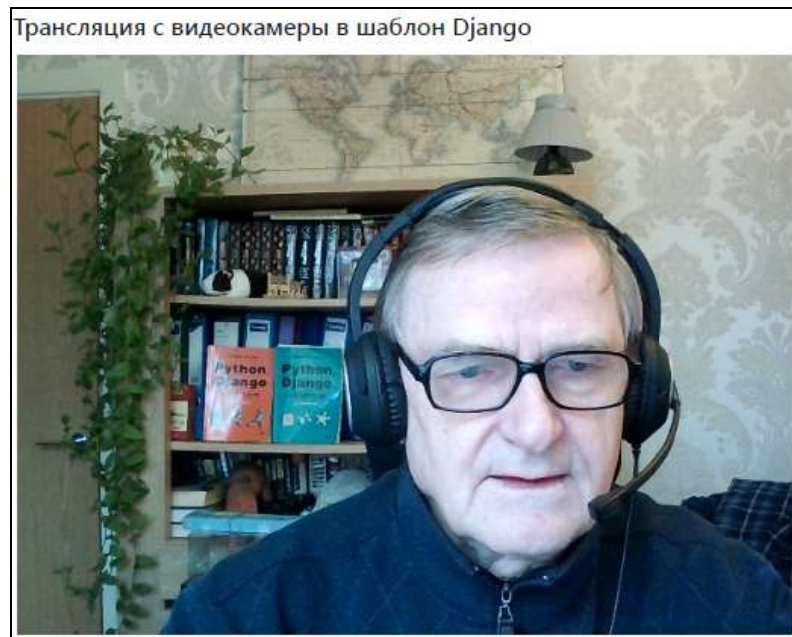


Рис. 1.14. Вид окна браузера в режиме трансляции медиа данных в шаблон Django

Примечание.

В режиме разработки на локальном компьютере данные программные модули будут работать без ошибок. Однако после их переноса на публичный сервер приложение может перестать выводить медиа поток в шаблон Django. Это может произойти из-за того, что связь с удаленным сервером осуществляется по протоколу HTTP. В документации на API MediaStream указано, что для обеспечения безопасности большинство современных браузеров обеспечивают трансляцию медиа данных только по протоколу HTTPS. В связи с этим на вашем удаленном сервере должен быть установлен SSL сертификат и связь с клиентами должна выполняться по протоколу HTTPS. Если такого сертификата нет, или имеются проблемы с видеокamerой, то пользователь получит страницу с сообщением «**На вашем компьютере отсутствует камера или камера занята другим приложением. Проверьте подключение, или повторите ее инициализацию.**».

1.5.3. Трансляция потоковых данных с видео камер в шаблоны Django

Теперь рассмотрим более сложный программный код, который обеспечит возможность перехвата кадров из потокового видео для их обработки перед вводом в шаблон Django. Такой подход обеспечит возможность, например, выполнить распознавание объектов в потовом видео, стилизовать кадры, изменить цветность, яркость и т.п.

На Python с использованием библиотеки OpenCV достаточно просто принять входной поток с видеокamerы и вывести его на экран локального компьютера. Базовый программный код выглядит следующим образом:

```
import cv2
cap = cv2.VideoCapture(0)
while(True):
    ret, frame = cap.read()
    if ret is None:
        break
    cv2.imshow('frame', frame)

    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
cap.release()
cv2.destroyAllWindows()
```

Здесь объект cap захватывает видео поток с видеокamerы, а затем в цикле while можно выполнить обработку кадров. В частности объект frame получает очередной кадр из видеопотока (а это обычное изображение image), и выводит его в окно компьютера с использованием метода imshow библиотеки OpenCV. В этом месте, перед отображением кадра пользователю, его можно изменить, передав в пользовательскую функцию. При нажатии на клавишу «q» цикл обработки видео потока прерывается с закрытием всех активных окон. Такой простой код можно использовать в программах, работающих на локальном компьютере, однако он не позволит вывести видеопоток в окно браузера через сеть интернет. Тем не менее, на Python реализована возможность вывода видеопотока на HTML страницы в шаблоны Django. Для начала рассмотрим схему работы группы программных модулей, обеспечивающих вывод видеопотока с камеры в шаблон Django (рис. 1.15).

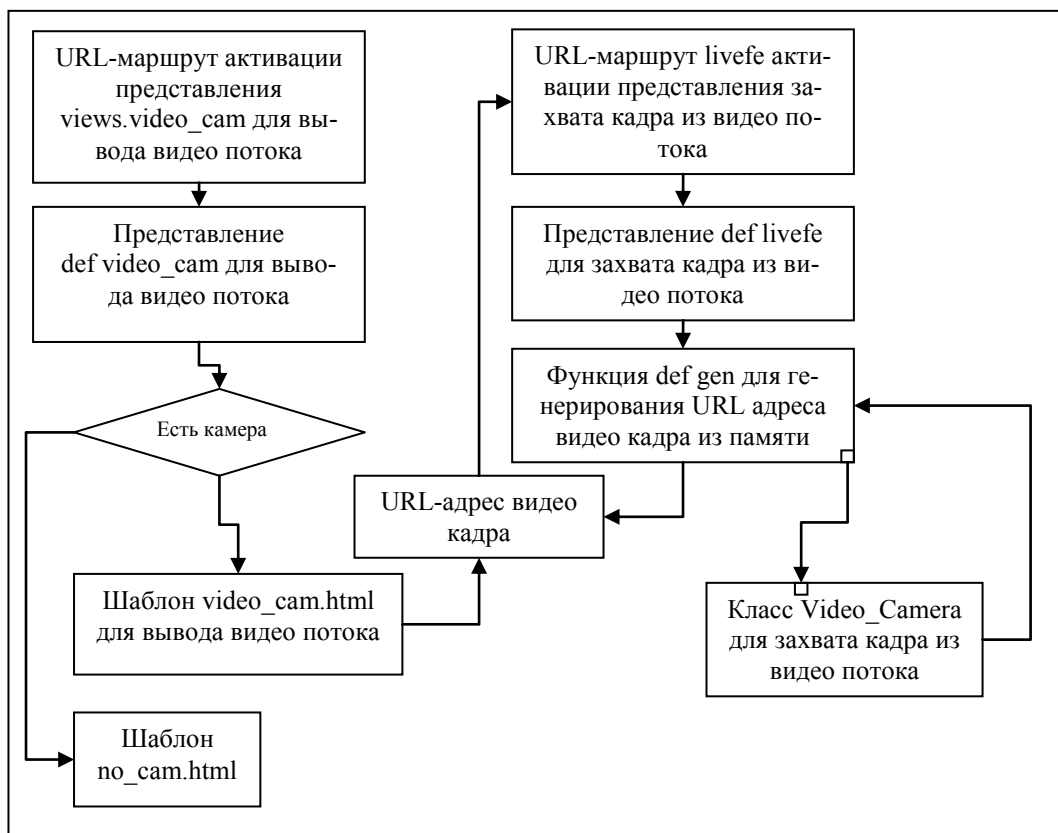


Рис. 1.15. Схема работы группы программных модулей для вывода видео потока в шаблон Django

Согласно приведенной схеме на первом этапе необходимо создать URL-маршрут для активации представления (`views.video_cam`), которое обеспечит проверку подключенной видеокамеры и, при ее наличии, вывод видеопотока в шаблон Django - `video_cam.html`. Если на компьютере отсутствует камера, то пользователю будет выдано соответствующее предупреждение через шаблон `no_cam.html`.

При наличии работающей видео камеры из представления `video_cam` будет вызван шаблон (`video_cam.html`), в котором будет указан URL-адрес на кадры видеопотока с видеокамеры. Если бы мы получали кадры из видеофайла, то здесь было бы достаточно указать на папку, в которой находится видео файл. Но в нашем случае такого файла на диске нет, и все данные находятся в памяти компьютера. То есть нужно сгенерировать ссылку на кадры потокового видео, которые находятся в памяти. Для этих целей будет использована дополнительная функция (`gen`) и класс (`Video_Camera`).

Итак, из шаблона `video_cam.html` фактически нужно вызывать URL-маршрут с именем `livefe`, который активирует представление захвата кадра из видео потока (представление `def livefe`). Данное представление обратится к функции с именем `gen()`, которое генерирует URL адрес видеокadra, находящегося в памяти компьютера. Эта функция вызывает метод `get_frame` (получить кадр) класса `Video_Camera`. В теле данного класса создается объект `self.video` с использованием метода `VideoCapture` библиотеки `OpenCV`, который примет видео поток от видеокамеры. А метод `get_frame` данного класса вернет очередной кадр видео

потока в виде изображения, преобразованного из формата jpg в поток байтов. Именно здесь можно организовать обработку кадров видеопотока. Функция `get`, приняв очередной кадр, вернет его в шаблон `video_cam.html`, где он и будет показан в окне браузера. Теперь реализуем эту схему в виде набора программных модулей.

На первом шаге в одном из шаблонов или элементе меню необходимо создать ссылку, которая активирует нужный URL маршрут для вывода видеопотока с камеры (листинг 1.15).

Листинг 1.15. Ссылка активации маршрута `video_cam` для вывода видеопотока с камеры

Примечание.

В демо-версии книги удалены листинги программ.

По данной ссылке в приложении `api_vision` будет активирован URL маршрут с именем `video_cam` (листинг 1.16).

Листинг 1.16. Маршрут URL с именем `video_cam`

Примечание.

В демо-версии книги удалены листинги программ.

По данному маршруту будет активировано представление `views.video_cam` (листинг 1.17).

Листинг 1.17. Представление (view) `video_cam`

Примечание.

В демо-версии книги удалены листинги программ.

Это достаточно простой программный код. Данное представление принимает набор входных параметров, проверяет наличие на компьютере подключенной камеры путем вызова функции `test_cam()`, и, при наличии камеры, вызывает шаблон с именем `video_cam.html`. Если подключенная камера отсутствует, то будет вызван шаблон с именем `no_cam.html`.

Код функции проверки наличия видеокамеры, подключенной к компьютеру, приведен в листинге 1.18.

Листинг 1.18. Код функции `test_cam`

В этой функции с использованием метода `cv2.VideoCapture` делается попытка открыть видео поток с камеры компьютера. Если у компьютера имеется подключенная камера, то создается объект «сар» для приема видео потока с камеры и признаку наличия камеры присваивается значение `cam=1`. Если у компьютера нет подключенной камеры, то признаку наличия камеры присваивается значение `cam=0`.

Код шаблона `no_cam.html` имеет следующий код (листинг 1.19).

Листинг 1.19. Шаблон no_cam.html

Примечание.

В демо-версии книги удалены листинги программ.

В этом шаблоне выдается текстовое сообщение о том, что у компьютера отсутствует подключенная камера.

Код шаблона video_cam.html имеет следующий код (листинг 1.20).

Листинг 1.20. Шаблон video_cam.html

Примечание.

В демо-версии книги удалены листинги программ.

Это достаточно простой шаблон, который имеет всего один тег, отвечающий за вывод изображения из URL адреса api_vision:livefe:

```
<div class="col-4">
  
</div>
```

По сути, из данного шаблона активируется URL-маршрут с именем «livefe» (листинг 1.21).

Листинг 1.21. Маршрут URL с именем livefe

Примечание.

В демо-версии книги удалены листинги программ.

По данному маршруту будет активировано представление views.livefe. Работу это представление сопровождает функция gen() и класс Video_Camera(). Программный код этих модулей приведен в листинге 1.22.

Листинг 1.22. Представление (view) livefe, функция gen() и класс Video_Camera()

Примечание.

В демо-версии книги удалены листинги программ.

Представление livefe представляет собой функцию - декоратор, в которой создается объект cam на основе класса VideoCamera(). Этот объект возвращает данные, полученные из функции с именем gen(). В свою очередь функция gen() обращается в метод get_frame класса VideoCamera(), откуда получает текущий кадр видеопотока от видеокамеры.

Если обратиться к коду класса VideoCamera(), то мы увидим, что там, с использованием библиотеки OpenCV принимается поток с видеокамеры в объект self.video. При этом текущий кадр из видеопотока принимает объект self.frame с использованием метода self.video.read(). Метод get_frame возвращает текущий кадр из видеопотока в виде изображения формата «jpg», преобразованного в массив байтов. В данном случае мы не храним

изображение на диске в виде файла, оно находится в памяти компьютера, а класс VideoCamera() в конечном итоге позволяет нам получить URL-адрес текущего кадра из памяти компьютера. При этом представление с именем livefe как раз и формирует URL-адрес последовательности кадров видео потока с видео камеры.

Подводя итог разбора этих программных модулей можно сделать следующие выводы:

- ❑ представление с именем livefe и сопровождающие его работу программные модули обеспечивают получение URL адреса последовательности кадров из видеопотока;
- ❑ представление с именем video_cam и шаблон video_cam.html обеспечивают вывод кадров видеопотока на HTML страницу.

В интерактивной цифровой книге можно запустить на выполнение набор программных модулей данного раздела и посмотреть, как будут отображаться кадры видеопотока с камеры в шаблоне video_cam.html.

Примечание.

В интерактивной цифровой книге для запуска данного примера нужно на вкладке «Показ примеров» нажать на кнопку «Примеры главы 1» и активировать ссылку «Трансляция видео потока в шаблон Django». Эта опция может не работать на стороне сервера из-за отсутствия SSL сертификата.

Внешний вид шаблона video_cam.html представлен на рис.1.16.

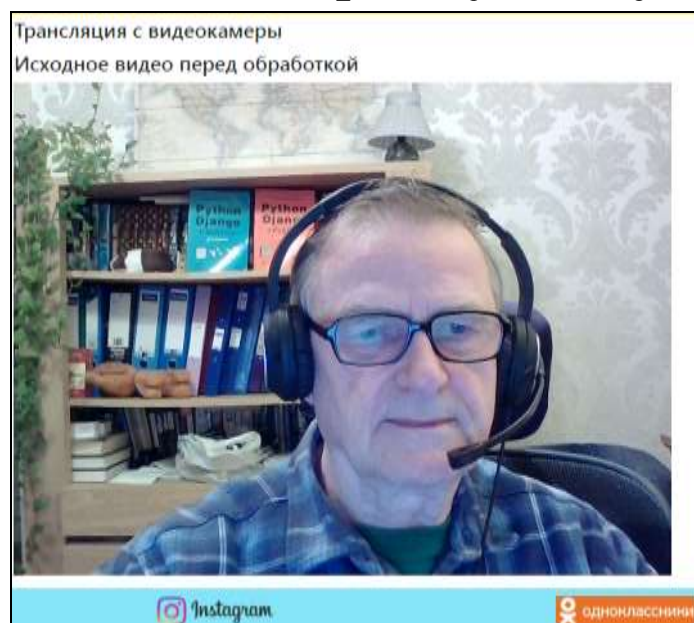


Рис. 1.16. Пример вывода видео потока с камеры в шаблон Django

1.6. Распознавание объектов на основе каскадов Хаара

Метод поиска и распознавания объектов на основе признаков (каскадов) Хаара был разработан и представлен в 2001 году, однако он до сих пор является основополагающим для

поиска объектов на изображениях в реальном времени. Основные принципы, на которых основан данный метод, таковы:

- ❑ используются изображения в интегральном представлении, что позволяет быстрее находить необходимые объекты;
- ❑ используются признаки Хаара, с помощью которых происходит поиск нужного объекта;
- ❑ используется бустинг (от англ. boost – улучшение, усиление) для выбора наиболее подходящих признаков для искомого объекта на некоторой части изображения;
- ❑ все признаки поступают на вход классификатора, который дает результат либо «верно», либо «ложь»;
- ❑ используются каскады признаков для быстрого отбрасывания окон, где не найден искомый объект.

Чтобы эффективно определить наличие и отсутствие сотен функций Хаара в изображении используется технология «интегрального изображения», где под термином «интеграция» понимается сложение маленьких блоков изображения. Здесь интегральное значение для каждого пикселя – это сумма всех пикселей над ним и слева от него. Начиная с левого верхнего угла и совершая обход вправо и вниз, все изображение может быть интегрировано с несколькими целочисленными операциями на пиксель (его значение). Схема процесса преобразования исходного изображения в интегральное изображение представлено на рис.1.17.

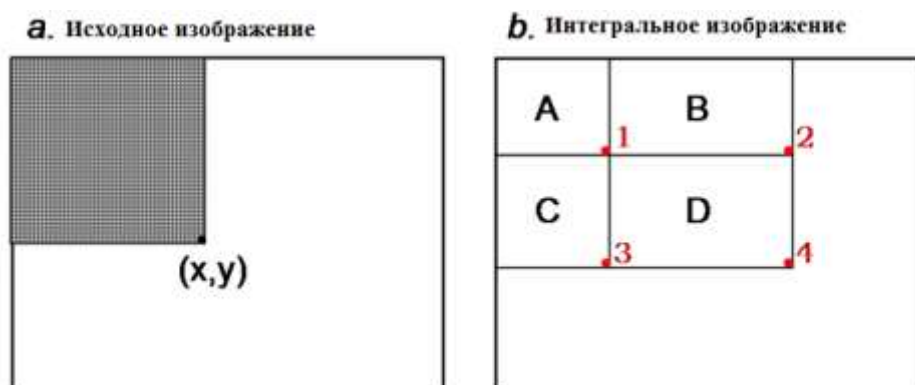


Рис.1.17. Схема процесса преобразования исходного изображения в интегральное изображение

Особенность «интегрального изображения» заключается в том, что:

- ❑ после интеграции пиксель с координатами (x, y) содержит сумму всех пиксельных значений в заштрихованном прямоугольнике (рис.3.9);
- ❑ сумма пиксельных значений в прямоугольнике D есть $(x_4, y_4) - (x_2, y_2) - (x_3, y_3) + (x_1, y_1)$.

При таком подходе удобно то, что сумма прямоугольников $A+B+C+D$ является значением «интегрального изображения» в положении 4, прямоугольников $A+B$ есть значение в положении 2, прямоугольников $A+C$ – значение в положении 3, и прямоугольника A – значение в положении 1 (рис.3.9). Так, с использованием ИНТЕГРАЛЬНОГО ИЗОБРАЖЕНИЯ, можно найти сумму пиксельных значений для любого прямоугольника в первоначальном изображении всего тремя целочисленными операциями: $(x_4, y_4) - (x_2, y_2) - (x_3, y_3) + (x_1, y_1)$. Иными

словами, скорость обработки интегрального изображения некоторыми специальными функциями на порядок выше, чем скорость обработки исходного изображения.

Интегральные изображения обрабатываются функциями с использованием каскадов Хаара, или примитивов Хаара. Пример примитивов Хаара приведен на рис.1.18.

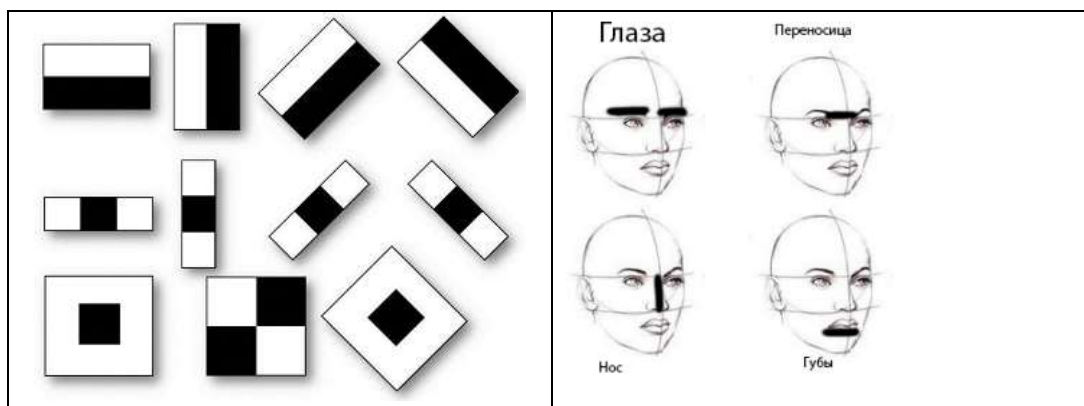


Рис.1.18. Примитивы Хаара

Примитивы Хаара в основном используются для поиска в изображениях некоторых объектов (например, лица человека, элементов фигуры человека, автомобилей, пешеходов и т.п.). При этом некое бегущее окно пробегает по исходному изображению и как бы сканирует его. В общем виде алгоритм обнаружения объекта (а возможно и характеристик объекта) на цифровом изображении выглядит так:

Есть исходное изображение, на котором имеется один (или несколько) искомых объектов. Такое изображение представлено двумерной матрицей пикселей размером $w \times h$, в которой каждый пиксель имеет значение:

- от 0 до 255, если это одноканальное черно-белое изображение;
- от (0 до 255) $\times$ 3, если это трехканальное цветное изображение (компоненты цветов R, G, B).

По исходному изображению начинает скользить другое окно меньшего размера. Окно сканирования начинает последовательно двигаться по изображению с шагом в 1 ячейку окна. В процессе сканирования изображения в каждом окне вычисляется до 200 000 вариантов расположения различных признаков, за счет изменения масштаба признаков и их положения в окне сканирования (при этом масштабируется не исходное изображение, а сканирующее окно - изменяется размер ячейки).

В результате этой работы, алгоритм находит в исходном изображении фрагменты, которые совпадают с признаками в скользящем окне сканирования.

Происходит группировка найденных в изображении признаков. Если такая группа пикселей совпадает с чертами искомых объектов (например, с лицом человека, автомобилем, пешеходом, светофором и т.п.), то они обрамляются прямоугольником с параметрами $rectangle_i = \{x, y, w, h, a\}$, где:

- x, y – координаты центра i -го прямоугольника;
- w – ширина прямоугольника;
- h – высота прямоугольника;

□ α – угол наклона прямоугольника к вертикальной оси изображения.

Иными словами, применительно к изображениям или фотографиям используется *подход на основе сканирующего окна (scanning window)*: сканируется изображение окном поиска (так называемое, окно сканирования), а затем происходит сопоставление с классификатором объекта к каждому положению. Одной особенностью данного подхода заключается в том, что для каждого класса объектов предварительно необходимо создать и обучить соответствующую модель.

Обучение классификаторов идет очень медленно, но **поиск объектов на основе обученных моделей идет очень быстро**, именно поэтому данный метод распознавания объектов на изображениях получил достаточно широкое распространение, особенно при обработке потокового видео. Данный алгоритм является одним из лучших по соотношению показателей эффективности распознавания - скорость работы. Также этот детектор обладает крайне низкой вероятностью ложного обнаружения объектов. Алгоритм хорошо работает и распознает объекты при их повороте под небольшим углом, примерно до 30 градусов. Система обучения и выбора наиболее значимых признаков *полностью автоматизирована* и не требует вмешательства человека, поэтому данный подход работает быстро.

Использование данного метода может дать достаточно хорошие результаты и для решения одной из задач – обнаружение и распознавания различных объектов. Однако для этого необходимо иметь предварительно обученную модель обнаружения и распознавания этих объектов. Такие готовые модели имеются в различных библиотеках компьютерного зрения. Они достаточно хорошо распознают людей в фас, в профиль, с боку, сзади, на видах сверху под углом до 60 градусов. Эти модели позволяют распознавать не только контуры людей, но и различные части тела: лицо, руки, ноги, фрагменты лица и даже выражения на лицах.

Обученный каскад Хаара, принимая на вход изображение, определяет, есть ли на нем искомый объект, т.е. выполняет задачу классификации, разделяя входные данные на два класса (есть искомый объект, нет искомого объекта). Собственно сам алгоритм работы сводится к тому, что определяются признаки Хаара. Это, по сути, набор прямоугольных областей, которые проходят по изображению и находят в нем детали искомого объекта. Создать и обучить собственный классификатор для распознавания объектов на основе каскадов Хаара – это достаточно трудоемкая и дорогая процедура. Однако для библиотеки **OpenCV** имеется набор уже обученных классификаторов, которые можно использовать в собственных приложениях.

Готовые (обученные) классификаторы на основе каскадов Хаара представлены в виде xml-файлов. Каждый из таких классификаторов обучен находить и распознавать в изображениях только ограниченный класс объектов. Примеры обученных классификаторов Хаара приведены в табл.1.1.

Таблица 1.1

Набор обученных классификаторов Хаара для распознавания объектов в изображениях

№ пп	Наименование файла	Распознаваемые объекты
1	haarcascade_eye.xml	Глаза
2	haarcascade_eye_tree_eyeglasses.xml	Глаза в очках

3	haarcascade_frontalcatface.xml	Фронтальная кошачья морда
4	haarcascade_frontalcatface_extended.xml	Фронтальная кошачья морда 2
5	haarcascade_frontalface_alt.xml	Фронтальное лицо
6	haarcascade_frontalface_alt2.xml	Фронтальное лицо 2
7	haarcascade_frontalface_alt_tree.xml	Фронтальное лицо каскад
8	haarcascade_frontalface_default.xml	Фронтальное лицо
9	haarcascade_fullbody.xml	Фронтальное тело
10	haarcascade_lefteye_2splits.xml	Левый глаз
11	haarcascade_licence_plate_rus_16stages.xml	Российские номерные знаки
12	haarcascade_lowerbody.xml	Нижняя часть тела
13	haarcascade_profileface.xml	Профиль лица
14	haarcascade_righteye_2splits.xml	Правый глаз
15	haarcascade_russian_plate_number.xml	Российские номерные знаки
16	haarcascade_smile.xml	Улыбка
17	haarcascade_upperbody.xml	Верхняя часть тела

1.6.1. Распознавание лиц в изображениях на основе каскадов Хаара

Для распознавания лиц будем использовать XML-файл каскадов Хаара со следующим именем:

```
haarcascade_frontalface_default.xml
```

Данный файл можно скачать из сети интернет и поместить в одну из папок вашего проекта, а можно использовать те XML-файлы, которые включены в библиотеку OpenCV и, после подключения данной библиотеки к проекту Django, находятся в папке: cv2/data/haarcascades.

Для запуска модулей распознаваний лиц на изображениях в одном из шаблонов или элементе меню необходимо создать ссылку, которая активирует нужный URL маршрут (листинг 1.23).

Листинг 1.23. Ссылка активации маршрута spot_face_image для распознавания лиц

Примечание.

В демо-версии книги удалены листинги программ.

По данной ссылке в приложении api_vision будет активирован URL маршрут с именем spot_face_image (листинг 1.24).

Листинг 1.24. Маршрут URL с именем spot_face_image

Примечание.

В демо-версии книги удалены листинги программ.

По этому маршруту будет активировано представление `spot_face_image`, которое использует дополнительные функции (листинг 1.25).

Листинг 1.25. Представление (view) `spot_face_image` и дополнительные функции

Примечание.

В демо-версии книги удалены листинги программ.

В данном представлении создано два объекта:

- ❑ `img_url` - URL- адрес исходного изображения в папке `media`;
- ❑ `image_path` - полный путь к файлу с исходным изображением.

Объект `img_url` нужен для того, чтобы показать исходное изображение в шаблоне Django, а объект `image_path` нужен для того, чтобы загрузить исходное изображение из файла и передать его для обработки в каскады Хаара.

С использованием метода `imread` библиотеки OpenCV исходное изображение читается из файла и загружается в объект `img`. По умолчанию изображения в библиотеке OpenCV представляют собой массив Numpy в цветовом пространстве BGR. Перед тем, как передать это изображение для обработки классификатором с каскадами Хаара, оно преобразовано в цветовое пространство RGB с использованием метода `cvtColor` библиотеки OpenCV.

На следующем шаге в объект `classifier` из библиотеки OpenCV загружается классификатор Хаара для распознавания лиц с именем `haarcascade_frontalface_default.xml`. Загрузка классификатора может быть выполнена и из локального файла (в данном примере эта строчка закомментирована).

Сама процедура распознавания выполняется с использованием метода `detectMultiScale`, которому передается исходное изображение:

```
# выполнение распознавания лиц
bboxes = classifier.detectMultiScale(img)
```

Результаты распознавания (по сути это координаты прямоугольников вокруг обнаруженных лиц) сохраняются в объекте `bboxes`.

На следующем шаге необходимо обрезать прямоугольными рамками все обнаруженные на изображении лица. Для этого с использованием инструкции `for` организован цикл, в котором эти рамки вставляются в исходное изображение.

```
# формирование прямоугольника вокруг каждого обнаруженного лица
for box in bboxes:
    # формирование координат
    x, y, width, height = box
    x2, y2 = x + width, y + height
    # рисование прямоугольников
    cv2.rectangle(img, (x, y), (x2, y2), (0, 255, 0), 4)
```

Для рисования рамок используется метод `rectangle` библиотеки OpenCV, в который передается исходное изображение (`img`), координаты левого верхнего и правого нижнего угла ра-

мок (x, y, x2, y2), цвет рамок зеленый в формате RGB (0, 255, 0), толщина рамки в пикселах (4).

Для того чтобы обработанное изображение показать в шаблоне Django нужно сохранить его на диске в виде файла, а затем получить URL-адрес этого файла, который передать в шаблон. Однако можно сформировать URL-адрес изображения, которое находится в памяти компьютера без записи его на диск. Для этих целей было создано две функции:

- ❑ `def to_image` – преобразовать массив Numpy в изображение цветового пространства RGB;
- ❑ `def to_data_uri` – получить изображение в виде строки с перекодировкой двоичных данных в символы ASCII (с использованием кодировки utf-8).

Функция `to_image` принимает изображение в виде массива Numpy и, с использованием библиотеки Pillow (метод `Image.fromarray`), преобразует это массив в изображение цветового пространства RGB, то есть в изображение, с которым работает библиотека Pillow.

Функция `to_data_uri` принимает изображение формата библиотеки Pillow и перекодировывает его в символьную строку стандарта base64. Использование формата base64 можно представить как упаковку изображения в текстовый конверт, который можно как по почте передавать из одного модуля другому. Как только изображение закодировано в текстовый формат, его можно передать и просмотреть в браузере. В Django вынуть изображение из текстового конверта можно непосредственно в шаблоне с использованием следующего тега:

```

```

Но в данном примере аналогичное преобразование выполнено непосредственно в функции `to_data_uri`, то есть функция возвращает URL-адрес изображения, но не с диска, а из памяти компьютера. На выходе из функции мы получаем объект `img_url2` с обработанным изображением, который можно передать в шаблон Django для отображения пользователю.

На завершающем этапе в данном представлении создается словарь `context`, который принимает URL исходного изображения (`img_url`), и URL обработанного изображения (`img_url2`). Этот словарь передается в шаблон с именем `spot_face_image.html`. Данный шаблон имеет следующий код (листинг 1.26).

Листинг 1.26. Шаблон `spot_face_image.html`

Примечание.

В демо-версии книги удалены листинги программ.

В данном шаблоне создан контейнер, в теле которого находятся строки и колонки для вывода текстовой информации и двух изображений в тегах `{{img_url}}` и `{{img_url2}}`.

В интерактивной цифровой книге можно запустить модуль данного примера на выполнение и посмотреть, как выглядит шаблон `spot_face_image.html`.

Примечание.

В интерактивной цифровой книге для запуска данного примера нужно на вкладке «Показ примеров» нажать на кнопку «Примеры главы 1» и активировать ссылку «Распознавание лиц на фото (каскады Хаара)».

Внешний вид шаблона `spot_face_image.html` представлен на рис.1.19.

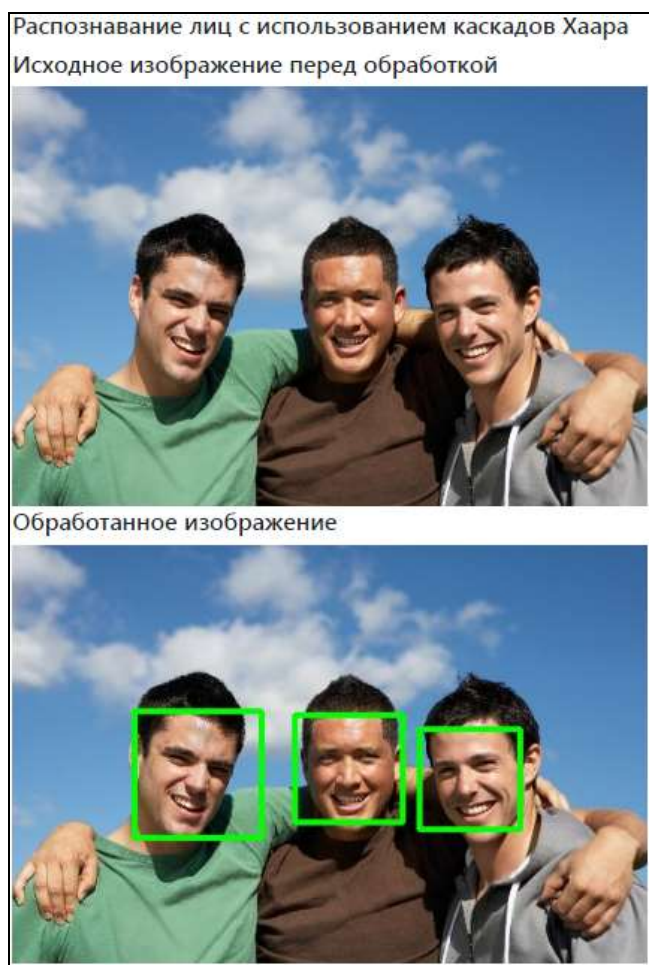


Рис. 1.19. Пример работы web приложения распознавания лиц в изображениях с использованием каскадов Хаара

Как видно из данного рисунка, на изображении распознано три лица. С использованием правой кнопки мыши можно скопировать обработанное изображение в буфер обмена сохранить его на локальном диске.

1.6.2. Распознавание лиц в видео файлах на основе каскадов Хаара

Видео файлы состоят из кадров, при этом каждый кадр представляет собой изображение. Таким образом, чтобы получить выходной видео файл с распознанными лицами, необходимо выполнить обработку каждого кадра по алгоритму, описанному в предыдущем разделе. То есть необходимо последовательно выполнить следующие действия:

- ☐ загрузить видео файл;
- ☐ организовать цикл перебора всех кадров видео файла;
- ☐ распознать лица на текущем кадре и нарисовать обрамляющие рамки;

- ❑ сохранить обработанный кадр в новом видеофайле.

Для распознавания лиц будем использовать XML-файл каскадов Хаара со следующим именем:

```
haarcascade_frontalface_default.xml
```

Данный файл можно скачать из сети интернет и поместить в одну из папок вашего проекта, а можно использовать те XML-файлы, которые включены в библиотеку OpenCV и, после подключения данной библиотеки к проекту Django, находятся в папке: `cv2/data/haarcascades`.

Для запуска модулей распознаваний лиц в видеофайлах в одном из шаблонов или элементе меню необходимо создать ссылку, которая активирует нужный URL маршрут (листинг 1.27).

Листинг 1.27. Ссылка активации маршрута `spot_face_video` для распознавания лиц

Примечание.

В демо-версии книги удалены листинги программ.

По данной ссылке в приложении `api_vision` будет активирован URL маршрут с именем `spot_face_video` (листинг 1.28).

Листинг 1.28. Маршрут URL с именем `spot_face_video`

Примечание.

В демо-версии книги удалены листинги программ.

По этому маршруту будет активировано представление `spot_face_video` (листинг 1.29).

Листинг 1.29. Представление (view) `spot_face_video`

Примечание.

В демо-версии книги удалены листинги программ.

Для работы данного представления необходимо подключить ряд библиотек и сформировать объекты для получения путей к медиа папкам из файла `settings.py`.

В данном представлении создано два объекта:

- ❑ `video_url` - URL адрес исходного видео файла в папке `media`;
- ❑ `video_path` - путь к файлу с исходным видео файлом.

Объект `video_url` нужен для того, чтобы показать исходный видео файл в шаблоне Django, а объект `video_path` нужен для того, чтобы загрузить исходный видео файл в объект `car`. Объект `car` позволяет захватить видеопоток из видео файла для решения следующих задач:

- ❑ получение характеристик видео потока для формирования итогового видео файла;
- ❑ организации цикла перебора кадров видео потока.

На следующем шаге из загруженного видео потока получаем следующие характеристики видео файла:

- ❑ w - ширина кадра;
- ❑ h - высота кадра;
- ❑ fps - частота кадров;
- ❑ total_frames - количество кадров.

Эти характеристики понадобятся для формирования параметров выходного видео файла. После этого создается объект `video_out_path`, в котором задается полный путь к обработанному видео файлу для его сохранения на диске. В данном примере имя исходного видео файла `Dictor.mp4`, а имя выходного (обработанного) файла `Dictor_out.mp4`.

После этого делается проверка, был ли ранее обработан входной файл и сформирован выходной видео файл с именем `Dictor_out.mp4`:

```
if os.path.isfile(video_out_path) is False:
```

Если обработка указанного видео файла уже была выполнена, то фрагмент программного кода, где выполняется распознавание лиц в данном видео, пропускается, а если такой обработки еще не было, то задаются параметры, необходимые для обработки кадров:

- ❑ classifier – классификатор с каскадами Хаара для распознавания лиц из библиотеки `cv2`;
- ❑ codec - кодек для сохранения видео в формате `mp4`;
- ❑ out - параметры для записи на диск выходного видео файла.

Это делается в следующем фрагменте кода:

```
# Создать codec для сохранения видео
# codec = cv2.VideoWriter_fourcc(*'XVID') # для файла avi
# codec = cv2.VideoWriter_fourcc('m', 'p', '4', 'v')
codec = cv2.VideoWriter_fourcc(*'avc1') # для файла mp4
# Задать параметры для выходного видео файла
out = cv2.VideoWriter(video_out_path, codec, fps, (w, h))
```

Примечание.

Здесь важно подобрать нужный кодек, который соответствует параметрами среды разработки или развертывания приложения. Важно, чтобы выбранный кодек обеспечивал корректное отображение видео на HTML страницах. В данном случае были попробованы кодеки «`mp4v`», «`h264`», «`avc1`», из них только последний создал файл, корректно отображаемый на локальном сервере разработки приложения. При развертывании приложения на публичном сервере, возможно, придется подключать другой кодек.

На следующем шаге запускается цикл перебора и обработки всех кадров видеофайла:

```
for i in range(total_frames + 1):
```

В теле данного цикла в объект `img` загружается очередной кадр из видео потока, который передается в детектор (классификатор) для распознавания лиц. Если в кадре были обнаружены лица, то они обрамляются прямоугольными рамками. Обработанный кадр записывается на диск в инструкции `out.write(img)`. По завершению перебора всех кадров происходит освобождение потоков для чтения и записи видео кадров:

- ❑ `out.release()` - освободить поток записи кадров;
- ❑ `cap.release()` - освободить поток чтения кадров.

Затем объект `video_out_url` получает URL-адрес обработанного видео файла.

На завершающем этапе формируется словарь `context` с URL-адресами исходного и обработанного видеофайла, и этот словарь передается в шаблон с именем `spot_face_video.html`. Данный шаблон имеет следующий код (листинг 1.30).

Листинг 1.30. Шаблон `spot_face_video.html`

Примечание.

В демо-версии книги удалены листинги программ.

В данном шаблоне создан контейнер, в теле которого находятся строки и колонки для вывода текстовой информации и двух видео файлов в тегах `{{video_url}}` и `{{ video_out_url }}`.

В интерактивной цифровой книге можно запустить модули данного примера на выполнение и посмотреть, как выглядит шаблон `spot_face_video.html`.

Примечание.

В интерактивной цифровой книге для запуска данного примера нужно на вкладке «Показ примеров» нажать на кнопку «Примеры главы 1» и активировать ссылку «Распознавание лиц на видео (каскады Хаара)».

Внешний вид шаблона `spot_face_video.html` представлен на рис. 1.20.

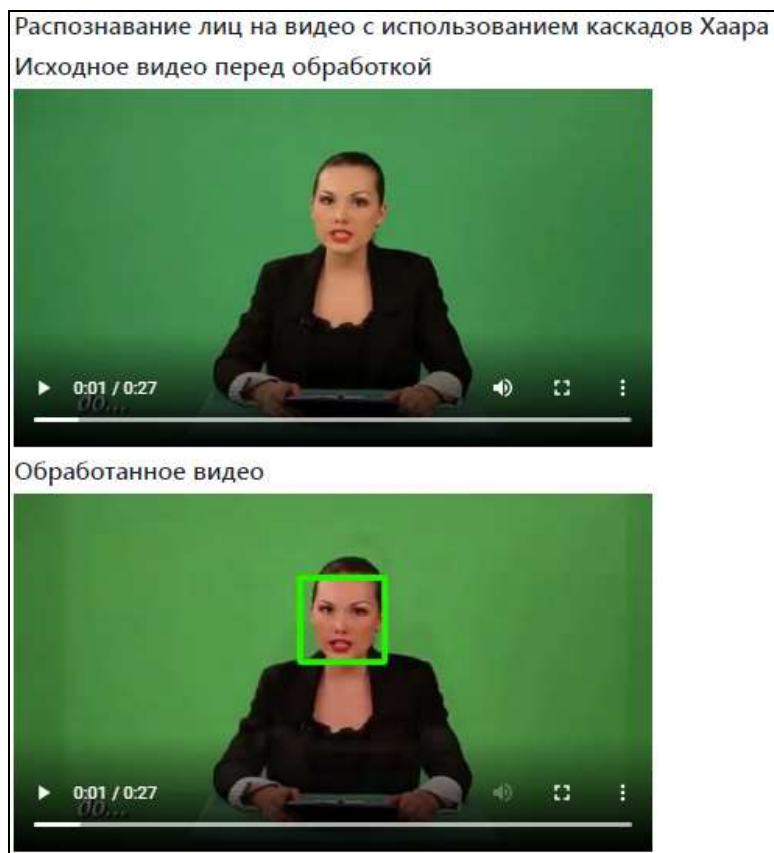


Рис. 1.20. Пример работы web приложения распознавания лиц в видео потоке с использованием каскадов Хаара

Как видно из данного рисунка, на HTML выведено два видео плеера, которые работают автономно. Их можно запустить по очереди, или одновременно. В верхнем видеоплеере можно посмотреть исходный видео файл, в нижнем плеере - видео файл после обработки с распознанным лицом диктора на каждом кадре.

Оба видеоплеера позволяют скачать видеофайлы на локальный компьютер. Для этого необходимо кликнуть левой кнопкой мыши на три точки, расположенные в правом нижнем углу проигрывателя. После этого откроется всплывающее меню, в котором нужно выбрать опцию «Скачать» (рис. 1.21).

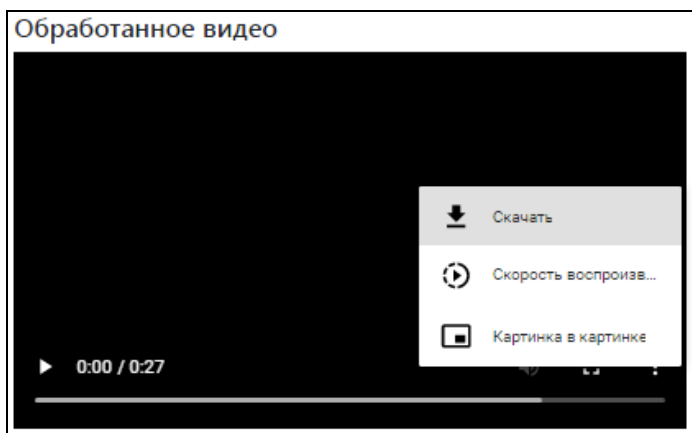


Рис. 1.21. Всплывающее меню для скачивания видео файла на локальный компьютер

1.6.3. Распознавание лиц в потоковом виде с видео камер на основе каскадов Хаара

Потоковое видео с видео камер, как и обычный видео файл, состоит из кадров, при этом каждый кадр представляет собой изображение. Таким образом, чтобы получить выходной видео поток с распознанными лицами, необходимо выполнить обработку каждого кадра видео потока. То есть необходимо последовательно выполнить следующие действия:

- ☐ обеспечить прием видео потока с видеокамеры;
- ☐ организовать цикл перебора всех кадров видео потока;
- ☐ распознать лица на текущем кадре и нарисовать обрамляющие рамки;
- ☐ вывести обработанные кадры видео потока в шаблон Django и показать в окне браузера.

Видео файл можно прочесть с диска и записать на диск, а значить для него достаточно просто получить URL-адрес для вывода кадров в шаблон Django. А вот кадры потокового видео с видео камер находятся в памяти компьютера и их нужно обрабатывать «на лету» без записи на диск. То есть, у них нет фиксированного URL адреса, такой «мнимый» адрес нужно искусственно создавать с использованием дополнительных программных модулей и передавать в шаблон Django.

Для распознавания лиц в потоковом видео будем использовать XML-файл каскадов Хаара со следующим именем:

```
haarcascade_frontalface_default.xml
```

Данный файл можно скачать из сети интернет и поместить в одну из папок вашего проекта, а можно использовать те XML-файлы, которые включены в библиотеку OpenCV и, после подключения данной библиотеки к проекту Django, находятся в папке: cv2/data/haarcascades.

Для запуска модулей распознаваний лиц в потоковом видео в одном из шаблонов или элементе меню необходимо создать ссылку, которая активирует нужный URL маршрут (линтинг 1.31).

Листинг 1.31. Ссылка активации маршрута face_cam распознавания лиц в видео потоке

Примечание.

В демо-версии книги удалены листинги программ.

По данной ссылке в приложении api_vision будет активирован URL маршрут с именем face_cam (листинг 1.32).

Листинг 1.32. Маршрут URL с именем face_cam

Примечание.

В демо-версии книги удалены листинги программ.

По этому маршруту будет активировано представление views.face_cam (листинг 1.33).

Листинг 1.33. Представление (view) face_cam

Примечание.

В демо-версии книги удалены листинги программ.

Это достаточно простой программный код. Данное представление вызывает шаблон с именем video_cam_face.html, который имеет следующий код (листинг 1.34).

Листинг 1.34. Шаблон video_cam_face.html

Примечание.

В демо-версии книги удалены листинги программ.

Это достаточно простой шаблон, который имеет всего один тег, отвечающий за вывод изображения из URL адреса api_vision:livefe_face:

```
<div class="col-4">
  
</div>
```

По сути, из данного шаблона активируется URL-маршрут с именем «livefe_face» (листинг 1.35).

Листинг 1.35. Маршрут URL с именем livefe_face

Примечание.

В демо-версии книги удалены листинги программ.

По данному маршруту будет активировано представление views.livefe_face. Работу это представление сопровождает функция gen() и класс Camera_Face(). Программный код этих модулей приведен в листинге 1.36.

Листинг 1.36. Представление (view) `livefe_face`, функция `gen()` и класс `Camera_Face()`

Примечание.

В демо-версии книги удалены листинги программ.

Представление `livefe_face` представляет собой функцию декоратор, в которой создается объект `cam` на основе класса `Camera_Face()`. Это представление возвращает данные, полученные из функции с именем `gen()`. В свою очередь функция `gen()` обращается в метод `get_frame` класса `Camera_Face()`, откуда получает текущий кадр видеопотока от видеокamеры. В теле данного класса создается объект `self.classifier`, в который загружается классификатор для распознавания лиц на основе каскадов Хаара.

Если обратиться к коду класса `Camera_Face()`, то мы увидим, что там, с использованием библиотеки `OpenCV` принимается поток с видеокamеры в объект `self.video`. При этом текущий кадр из видеопотока принимает объект `self.frame` с использованием метода `self.video.read()`.

Метод `get_frame` получает текущий кадр из видео потока (`image`) и выполняет его обработку. Здесь создается объект `gray` (трансформация цветного изображения в оттенки серого), который передается в классификатор каскадов Хаара для поиска лиц:

```
bboxes = self.classifier.detectMultiScale(gray)
```

Это один из модулей библиотеки `OpenCV`, в котором создаются координаты прямоугольных рамок вокруг каждого распознанного лица. Эти координаты сохраняются в объекте `bboxes`. Затем с использованием инструкции `for` организован цикл, в котором вокруг всех лиц текущего кадра рисуются обрамляющие рамки.

На следующем шаге метод `get_frame` возвращает текущий кадр из видеопотока в виде изображения формата «`jprg`», преобразованного в массив байтов. В данном случае мы не записываем изображение на диск в виде файла, оно находится в памяти компьютера, а класс `Camera_Face()` в конечном итоге позволяет нам получить URL-адрес текущего кадра из памяти компьютера. При этом представление с именем `livefe_face` как раз и формирует URL-адрес последовательности кадров видео потока с видео kamеры после их обработки.

Подводя итог разбора этих программных модулей можно сделать следующие выводы:

- ❑ представление с именем `livefe_face` и сопровождающие его работы программные модули обеспечивают получение URL адреса последовательности кадров из видеопотока;
- ❑ представление с именем `face_cam` и шаблон `video_cam_face.html` обеспечивают вывод кадров видеопотока на HTML страницу.

В интерактивной цифровой книге можно запустить на выполнение набор программных модулей данного раздела и посмотреть, как будут отображаться кадры видеопотока с kamеры в шаблоне `video_cam_face.html`.

Примечание.

В интерактивной цифровой книге для запуска данного примера нужно на вкладке «Показ примеров» нажать на кнопку «Примеры главы 1» и активировать ссылку «Распознавание лиц с kamеры (каскады Хаара)». Эта опция может не работать на стороне сервера из-за отсутствия SSL сертификата.

Внешний вид шаблона `video_cam_face.html` представлен на рис.1.22.

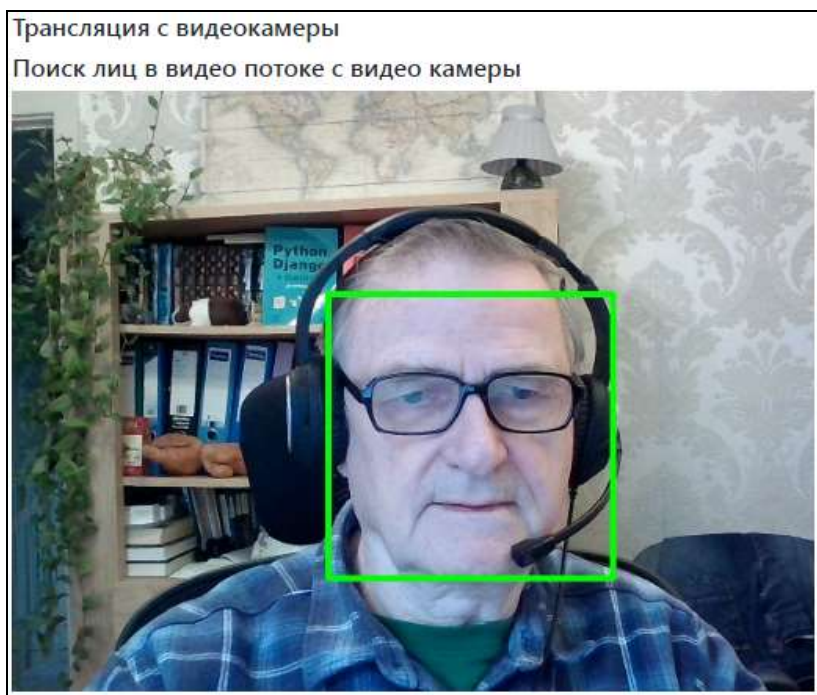


Рис. 1.22. Пример вывода видео потока с камеры в шаблон Django с распознаванием лица

Как видно из данного рисунка, даже наличие наушников и очков не помешали корректному распознаванию лица в потоковом видео с видео камеры.

Метод распознавания объектов на основе каскадов Хаара по-прежнему очень популярен для некоторых объектов, таких как лица, глаза, части тела человека, автомобили и т.д. Это достаточно быстрый метод и он может использоваться для анализа видео потока в реальном времени. Однако данный метод имеет несколько ограничений:

- ☐ более низкая точность по сравнению с современными детекторами объектов на основе нейронных сетей;
- ☐ высокий уровень ложных срабатываний, особенно при наличии пестрого фона;
- ☐ плохо распознает лица при наклоне головы в сторону или при повороте в сторону;
- ☐ требуется ручная настройка параметров для повышения достоверности результатов.

Рассмотрим другие методы распознавания объектов.

1.7. Распознавание лиц на основе детектора SSD (Single-Shot multibox Detection)

В данном разделе мы рассмотрим возможности использования детекторов SSD (Single Shot multibox Detection), которые представляют собой метод обнаружения объектов на изображениях с использованием глубокой нейронной сети.

Примечание.

Следует различать понятия обнаружение объектов и классификацию объектов. Обнаружение объекта — это обнаружение объекта заданного типа и определение места его расположения на изображении, а классификация объектов — это распознавание того, к какому классу принадлежит найденный объект.

С нейронными сетями работает и библиотека OpenCV, она имеет модуль dnn, который позволяет находить лица, используя предварительно обученную модель нейронной сети глубокого обучения. В данном разделе для распознавания лиц мы будем использовать архитектуру нейронной сети ResNet-10 и фреймворк Caffe. Для загрузки модели Caffe будет использован модуль библиотеки OpenCV - cv2.dnn.readNetFromCaffe(). Набор этого инструментария позволяет выполнить обнаружение большого количества лиц на изображении точнее, чем метод обнаружения лиц с использованием каскадов Хаара. В данном примере мы будем использовать следующие файлы нейронной сети ResNet-10:

- ❑ res10_300x300_ssd_iter_140000.caffemodel – размер 10417 Кб;
- ❑ deploy.prototxt – размер 30 Кб.

Данные файлы можно скачать из сети интернет и поместить в одну из папок вашего проекта. В данном проекте эти файлы будут находиться в папке net_model.

1.7.1. Распознавание лиц в изображениях на основе детектора SSD (Single-Shot multibox Detection)

Для запуска модулей распознаваний лиц на изображениях с использованием нейронной сети ResNet-10 в одном из шаблонов или элементе меню необходимо создать ссылку, которая активирует нужный URL маршрут (листинг 1.37).

Листинг 1.37. Ссылка активации маршрута face_foto_resnet для распознавания лиц

Примечание.

В демо-версии книги удалены листинги программ.

По данной ссылке в приложении api_vision будет активирован URL маршрут с именем face_foto_resnet (листинг 1.38).

Листинг 1.38. Маршрут URL с именем face_foto_resnet

Примечание.

В демо-версии книги удалены листинги программ.

По этому маршруту будет активировано представление face_foto_resnet, которое использует дополнительные функции (листинг 1.39).

Листинг 1.39. Представление (view) face_foto_resnet

Примечание.

В демо-версии книги удалены листинги программ.

В данном представлении создано два объекта:

- ❑ `img_url` – URL адрес исходного изображения в папке `media`;
- ❑ `image_path` - путь к файлу с исходным изображением.

Объект `img_url` нужен для того, чтобы показать исходное изображение в шаблоне Django, а объект `image_path` нужен для того, чтобы загрузить исходное изображение из файла и передать его для обработки в нейронную сеть.

С использованием метода `imread` библиотеки OpenCV исходное изображение читается из файла и загружается в объект `image`. По умолчанию изображения в библиотеке OpenCV представляют собой массив NumPy в цветовом пространстве BGR.

На следующем шаге создаем два объекта `prototxt_path` и `model_path`, в которые записываем полный путь к файлам модели нейронной сети и, с использованием библиотеки OpenCV, загружаем эти файлы в объект `model`:

```
# Полный путь к файлам модели
prototxt_path = str(Path('net_model/deploy.prototxt'))
model_path = str(Path('net_model/res10_300x300_ssd_iter_140000.caffemodel'))
# загрузим модель Caffe
model = cv2.dnn.readNetFromCaffe(prototxt_path, model_path)
```

На следующем шаге делается предварительная обработка входного изображения и передача его в объект `blob`. Объект `blob` передается на вход модели нейронной сети, где выполняется обнаружение лиц с записью результатов в объект `output`:

```
# предварительная обработка: изменение размера и вычитание среднего
blob = cv2.dnn.blobFromImage(image, 1.0, (300, 300), (104.0, 177.0, 123.0))
# передаем изображение на вход нейронной сети
model.setInput(blob)
# выполняем логический вывод и получаем результат
output = np.squeeze(model.forward())
```

На следующем шаге необходимо обрамить прямоугольными рамками все обнаруженные на изображении лица. Для этого с использованием инструкции `for` организован цикл, в котором эти рамки вставляются в исходное изображение.

```
# Рисуем прямоугольники вокруг обнаруженных лиц
font_scale = 0.5 # Размер шрифта для надписи достоверности
for i in range(0, output.shape[0]):
    # получить достоверность распознавания
    confidence = output[i, 2]
    # если достоверность выше 50%, то рисуем обрамляющий прямоугольник
    if confidence > 0.5:
        # получить координаты прямоугольника и масштабировать его
        box = output[i, 3:7] * np.array([w, h, w, h])
        # преобразовать координаты в целые числа
        start_x, start_y, end_x, end_y = box.astype(int)
        # рисуем прямоугольник вокруг лица
        cv2.rectangle(image, (start_x, start_y), (end_x, end_y),
                      color=(0, 255, 0), thickness=2)
        # добавляем текст с полученной достоверностью
        cv2.putText(image, f"{confidence * 100:.1f}%", (start_x, start_y - 5),
                   cv2.FONT_HERSHEY_SIMPLEX, font_scale, (0, 255, 0), 1)
```

Для рисования рамок используется метод `rectangle` библиотеки `OpenCV`, в который передается исходное изображение (`image`), координаты левого верхнего и правого нижнего угла рамок, цвет рамок зеленый в формате `RGB (0, 255, 0)`, толщина рамки в пикселах (4).

В этом цикле рамки будут нарисованы только в тех областях изображения, где достоверность распознавания превышает 50%. К каждой рамке будет добавлен текст с полученной достоверностью распознанного лица. По завершению данного цикла обработанное изображение преобразуется в цветовое пространство `RGB` с использованием метода `cvtColor` библиотеки `OpenCV`.

Для того чтобы обработанное изображение показать в шаблоне `Django` нужно сохранить его на диске в виде файла, а затем получить `URL`-адрес этого файла, который передать в шаблон. Однако можно сформировать `URL`-адрес изображения, которое находится в памяти компьютера без записи его на диск. Для этих целей задействованы две функции:

- ❑ `def to_image` – преобразовать массив `Numpy` в изображение цветового пространства `RGB`;
- ❑ `def to_data_uri` – получить изображение в виде строки с перекодировкой двоичных данных в символы `ASCII` (с использованием кодировки `utf-8`).

Функция `to_image` принимает изображение в виде массива `Numpy` и, с использованием библиотеки `Pillow` (метод `Image.fromarray`), преобразует это массив в изображение цветового пространства `RGB`, то есть в изображение, с которым работает библиотека `Pillow`.

Функция `to_data_uri` принимает изображение формата библиотеки `Pillow` и перекодирует его в символьную строку стандарта `base64`. Использование формата `base64` можно представить как упаковку изображения в текстовый конверт, который можно как по почте передавать из одного модуля другому. Как только изображение закодировано в текстовый формат, его можно передать и просмотреть в браузере. В `Django` вынуть изображение из текстового конверта можно непосредственно в шаблоне с использованием следующего тега:

```

```

Но в данном примере аналогичное преобразование выполнено непосредственно в функции `to_data_uri`, то есть функция возвращает `URL`-адрес изображения, но не с диска, а из памяти компьютера. На выходе из функции мы получаем объект `img_url2` с обработанным изображением, который можно передать в шаблон `Django` для отображения пользователю.

На завершающем этапе в данном представлении создается словарь `context`, который принимает `URL` исходного изображения (`img_url`), и `URL` обработанного изображения (`img_url2`). Этот словарь передается в шаблон с именем `face_foto_resnet.html`. Данный шаблон имеет следующий код (листинг 1.40).

Листинг 1.40. Шаблон `face_foto_resnet.html`

Примечание.

В демо-версии книги удалены листинги программ.

В данном шаблоне создан контейнер, в теле которого находятся строки и колонки для вывода текстовой информации и двух изображений в тегах `{{img_url}}` и `{{img_url2}}`.

В интерактивной цифровой книге можно запустить модули данного примера на выполнение и посмотреть, как выглядит шаблон `face_foto_resnet.html`.

Примечание.

В интерактивной цифровой книге для запуска данного примера нужно на вкладке «Показ примеров» нажать на кнопку «Примеры главы 1» и активировать ссылку «Распознавание лиц на фото с ResNet-10».

Внешний вид шаблона face_foto_resnet.html представлен на рис.1.23.

Распознавание лиц на фото с ResNet-10

Исходное изображение



Обработанное изображение

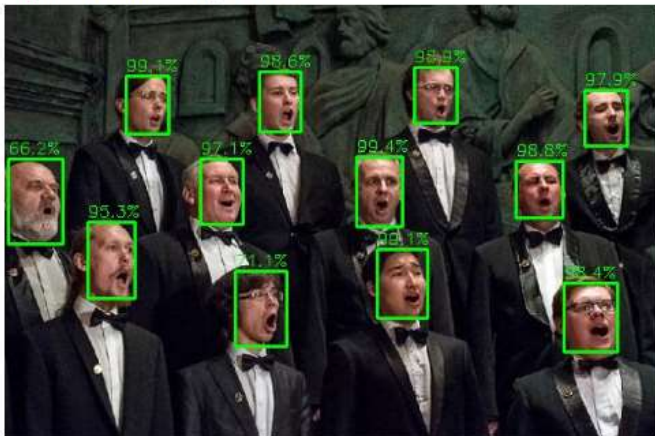


Рис. 1.23. Пример работы web приложения распознавания лиц в изображениях с использованием сети ResNet-10

Как видно из данного рисунка, на изображении распознаны все лица с высокой достоверностью, даже лица в очках и с достаточно большим поворотом головы. С использованием правой кнопки мыши можно скопировать обработанное изображение в буфер обмена и сохранить его на локальном диске

1.7.2. Распознавание лиц в видео файлах на основе детектора SSD (Single-Shot multibox Detection)

В данном разделе мы рассмотрим возможность использования детекторов SSD (Single Shot multibox Detection) для распознавания лиц в видеофайлах. При этом исходный файл будет загружаться с диска, а обработанные кадры видеопотока записываться на диск в виде другого файла. Для распознавания лиц мы будем использовать ту же архитектуру нейронной сети ResNet-10 и фреймворк Caffe.

Для запуска модулей распознаваний лиц в видеофайлах с использованием нейронной сети ResNet-10 в одном из шаблонов или элементе меню необходимо создать ссылку, которая активирует нужный URL маршрут (листинг 1.41).

Листинг 1.41. Ссылка активации маршрута face_video_resnet для распознавания лиц

Примечание.

В демо-версии книги удалены листинги программ.

По данной ссылке в приложении api_vision будет активирован URL маршрут с именем face_video_resnet (листинг 1.42).

Листинг 1.42. Маршрут URL с именем face_video_resnet

Примечание.

В демо-версии книги удалены листинги программ.

По этому маршруту будет активировано представление face_video_resnet (листинг 1.43).

Листинг 1.43. Представление (view) face_video_resnet

Примечание.

В демо-версии книги удалены листинги программ.

Для работы данного представления необходимо подключить ряд библиотек и сформировать объекты для получения путей к медиа папкам из файла settings.py.

В данном представлении создано два объекта:

- ❑ video_url – URL адрес исходного видео файла в папке media;
- ❑ video_path - путь к файлу с исходным видео файлом.

Объект video_url нужен для того, чтобы показать исходный видео файл в шаблоне Django, а объект video_path нужен для того, чтобы загрузить исходный видео файл в объект car. Объект car позволяет захватить видеопоток из видео файла для решения следующих задач:

- ❑ получение характеристик видео потока для формирования итогового видео файла;
- ❑ организации цикла перебора кадров видео потока.

На следующем шаге из загруженного видео потока получаем следующие характеристики видео файла:

- ❑ w - ширина кадра;

- ❑ `h` - высота кадра;
- ❑ `fps` - частота кадров;
- ❑ `total_frames` - количество кадров.

Эти характеристики понадобятся для формирования параметров выходного видео файла. После этого создается объект `video_out_path`, в котором задается полный путь к обработанному видео файлу для его сохранения на диске. В данном примере имя исходного видео файла `Dictor.mp4`, а имя выходного (обработанного) файла `Dictor_res.mp4`.

После этого делается проверка, был ли ранее обработан входной файл и сформирован выходной видео файл с именем `Dictor_res.mp4`:

```
if os.path.isfile(video_out_path) is False:
```

Если обработка видео файла уже была выполнена, то фрагмент программного кода, где выполняется распознавание лиц, пропускается. Если такой обработки еще не было, то создаются два объекта `prototxt_path` и `model_path`, в которые записываем полный путь к файлам модели нейронной сети и, с использованием библиотеки `OpenCV`, загружаем эти файлы в объект `model`:

```
# Полный путь к файлам модели
prototxt_path = str(Path('net_model/deploy.prototxt'))
model_path = str(Path('net_model/res10_300x300_ssd_iter_140000.caffemodel'))
# загрузка модели Caffe
model = cv2.dnn.readNetFromCaffe(prototxt_path, model_path)
```

После этого создаются два важных объекта:

- ❑ `codec` - кодек для сохранения видео в формате `mp4`;
- ❑ `out` - параметры для записи на диск выходного видео файла.

Это делается в следующем фрагменте кода:

```
# Создать codec для сохранения видео
# codec = cv2.VideoWriter_fourcc(*'XVID') # для файла avi
# codec = cv2.VideoWriter_fourcc('m', 'p', '4', 'v')
codec = cv2.VideoWriter_fourcc(*'avc1') # для файла mp4
# Задать параметры для выходного видео файла
out = cv2.VideoWriter(video_out_path, codec, fps, (w, h))
```

Примечание.

Здесь важно подобрать нужный кодек, который соответствует параметрами среды разработки или развертывания приложения. Важно, чтобы выбранный кодек обеспечивал корректное отображение видео на HTML страницах. В данном случае были попробованы кодеки «`mp4v`», «`h264`», «`avc1`» и др., из них только последний создал файл, корректно отображаемый на локальном сервере разработки приложения. При развертывании приложения на публичном сервере, возможно, придется подключать другой кодек.

На следующем шаге запускается цикл перебора и обработки всех кадров видеофайла:

```
for j in range(total_frames + 1):
```

В теле данного цикла в объект `image` загружается очередной кадр из видео потока, который подвергается предварительной обработке. Обработанный кадр сохраняется в объекте `blob`.

Объект blob передается на вход модели нейронной сети, где выполняется обнаружение лиц с записью результатов в объект output:

```
# предварительная обработка: изменение размера и вычитание среднего
blob = cv2.dnn.blobFromImage(image, 1.0, (300, 300), (104.0, 177.0, 123.0))
# передаем изображение на вход нейронной сети
model.setInput(blob)
# выполняем логический вывод и получаем результат
output = np.squeeze(model.forward())
```

На следующем шаге необходимо обrahmen прямоугольными рамками все обнаруженные на изображении лица. Для этого с использованием инструкции for организован цикл, в котором эти рамки вставляются в исходное изображение.

```
# Рисуем прямоугольники вокруг обнаруженных лиц
font_scale = 0.5 # Размер шрифта для надписи достоверности
for i in range(0, output.shape[0]):
    # получить достоверность распознавания
    confidence = output[i, 2]
    # если достоверность выше 50%, то рисуем обрамляющий прямоугольник
    if confidence > 0.3:
        # получить координаты прямоугольника и масштабировать его
        box = output[i, 3:7] * np.array([w, h, w, h])
        # преобразовать координаты в целые числа
        start_x, start_y, end_x, end_y = box.astype(int)
        # рисуем прямоугольник вокруг лица
        cv2.rectangle(image, (start_x, start_y), (end_x, end_y),
                      color=(0, 255, 0), thickness=2)
        # добавляем текст с полученной достоверностью
        cv2.putText(image, f"{confidence * 100:.1f}%", (start_x, start_y - 5),
                   cv2.FONT_HERSHEY_SIMPLEX, font_scale, (0, 255, 0), 1)
```

Для рисования рамок используется метод rectangle библиотеки OpenCV, в который передается исходное изображение (image), координаты левого верхнего и правого нижнего угла рамок, цвет рамок зеленый в формате RGB (0, 255, 0), толщина рамки в пикселах (4).

В этом цикле рамки будут нарисованы только в тех областях изображения, где достоверность распознавания превышает 30%. К каждой рамке будет добавлен текст с полученной достоверностью распознанного лица. Обработанный кадр записывается на диск с использованием инструкции out.write(image).

По завершению перебора всех кадров происходит освобождение потоков для чтения и записи видео кадров:

- ☐ out.release() - освободить поток записи кадров;
- ☐ cap.release() - освободить поток чтения кадров.

Затем объект video_out_url получает URL адрес обработанного видео файла.

На завершающем этапе формируется словарь context с URL адресами исходного и обработанного видеофайла, и этот словарь передается в шаблон с именем face_video_resnet.html. Данный шаблон имеет следующий код (листинг 1.44).

Листинг 1.44. Шаблон face_video_resnet.html

Примечание.

В демо-версии книги удалены листинги программ.

В данном шаблоне создан контейнер, в теле которого находятся строки и колонки для вывода текстовой информации и двух видео файлов в тегах `{{video_url}}` и `{{ video_out_url }}`. В интерактивной цифровой книге можно запустить модули данного примера на выполнение и посмотреть, как выглядит шаблон `face_video_resnet.html`.

Примечание.

В интерактивной цифровой книге для запуска данного примера нужно на вкладке «Показ примеров» нажать на кнопку «Примеры главы 1» и активировать ссылку «Распознавание лиц на видео с ResNet-10».

Внешний вид шаблона `face_video_resnet.html` представлен на рис.1.24.

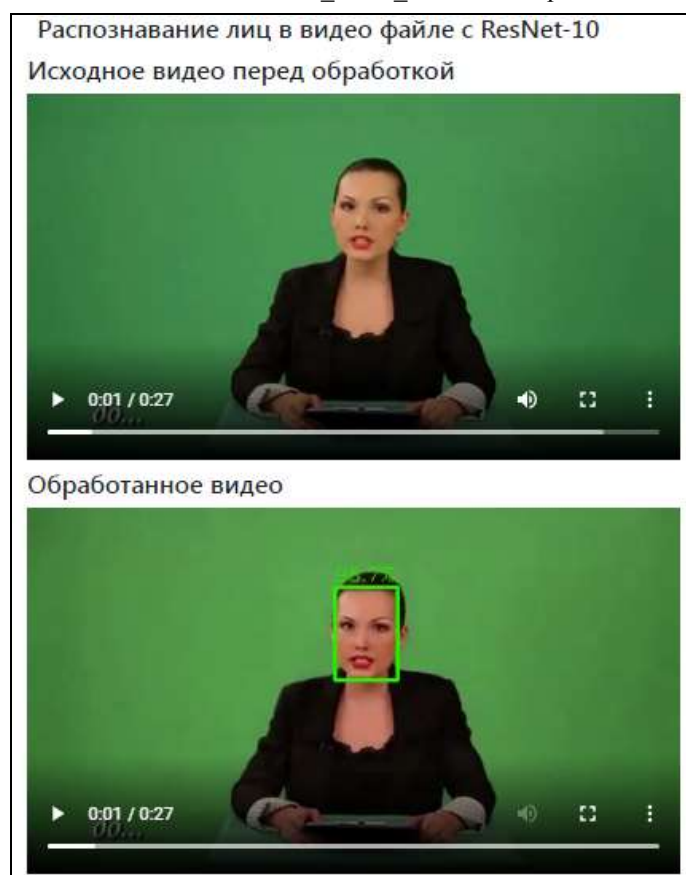


Рис. 1.24. Пример работы web приложения распознавания лиц в видео потоке с использованием сети ResNet-10

Как видно из данного рисунка, на HTML страницу выведено два видео плеера, которые работают автономно. Их можно запустить по очереди, или одновременно. В верхнем видео-

плеере можно посмотреть исходный видео файл, в нижнем плеере - видео файл после обработки с распознанным лицом диктора на каждом кадре.

Оба видеоплеера позволяют скачать видеофайлы на локальный компьютер. Для этого необходимо кликнуть левой кнопкой мыши на три точки, расположенные в правом нижнем углу. После этого откроется всплывающее меню, в котором нужно выбрать опцию «Скачать».

1.7.3. Распознавание лиц в видео потоке с видеокамеры на основе детектора SSD (Single-Shot multibox Detection)

В данном разделе мы рассмотрим возможность использования детекторов SSD (Single Shot multibox Detection) для распознавания лиц в видео потоке с видеокамеры. Для распознавания лиц мы будем использовать архитектуру нейронной сети ResNet-10 и фреймворк Caffe.

Для запуска модулей распознаваний лиц в видео потоке с видеокамеры с использованием нейронной сети ResNet-10 в одном из шаблонов или элементе меню необходимо создать ссылку, которая активирует нужный URL маршрут (листинг 1.45).

Листинг 1.45. Ссылка активации маршрута face_cam_resnet для распознавания лиц

Примечание.

В демо-версии книги удалены листинги программ.

По данной ссылке в приложении api_vision будет активирован URL маршрут с именем face_cam_resnet (листинг 1.46).

Листинг 1.46. Маршрут URL с именем face_cam_resnet

Примечание.

В демо-версии книги удалены листинги программ.

По этому маршруту будет активировано представление face_cam_resnet (листинг 1.47).

Листинг 1.47. Представление (view) face_cam_resnet

Примечание.

В демо-версии книги удалены листинги программ.

Это очень простой программный код. Данное представление вызывает шаблон с именем face_cam_resnet.html, который имеет следующий код (листинг 1.48).

Листинг 1.48. Шаблон face_cam_resnet.html

Примечание.

В демо-версии книги удалены листинги программ.

Это достаточно простой шаблон, который имеет всего один тег, отвечающий за вывод изображения из URL адреса api_vision:livefe_face_resnet:

```
<div class="col-4">
    
</div>
```

По сути, из данного шаблона активируется URL-маршрут с именем «livefe_face_resnet» (листинг 1.49).

Листинг 1.49. Маршрут URL с именем livefe_face_resnet

Примечание.

В демо-версии книги удалены листинги программ.

По данному маршруту будет активировано представление `views.livefe_face_resnet`. Работу это представление сопровождает функция `gen()` и класс `Camera_Face_resnet()`. Программный код этих модулей приведен в листинге 1.50.

Листинг 1.50. Представление (view) livefe_face_resnet, функция gen() и класс Camera_Face_resnet()

Примечание.

В демо-версии книги удалены листинги программ.

Представление `livefe_face_resnet` представляет собой функцию декоратор, в которой создается объект `cam` на основе класса `Camera_Face_resnet()`. Этот объект возвращает данные, полученные из функции с именем `gen()`. В свою очередь функция `gen()` обращается в метод `get_frame` класса `Camera_Face_resnet()`, откуда получает текущий кадр видеопотока от видеокамеры.

Если обратиться к коду класса `Camera_Face_resnet()`, то мы увидим, что там, с использованием библиотеки `OpenCV` принимается поток с видеокамеры в объект `self.cap`. При этом текущий кадр из видеопотока принимает объект `self.frame` с использованием метода `self.cap.read()`.

После этого свои значения получают параметры входного кадра:

- ❑ `self.w` - ширина кадра;
- ❑ `self.h` - высота кадра;
- ❑ `self.fps` - частота кадров.

На следующем шаге создаем три объекта, которые обеспечивают загрузку нейронной сети:

- ❑ `self.prototxt_path` - путь к файлу весов нейронной сети;
- ❑ `self.model_path` - путь к файлу модели нейронной сети;
- ❑ `self.model` – модель.

Метод `get_frame` получает очередной кадр, делает предварительную обработку входного кадра и передает его в объект `blob`. Объект `blob` передается на вход модели нейронной сети, где выполняется обнаружение лиц с записью результатов в объект `output`:

```
# предварительная обработка: изменение размера и вычитание среднего
blob = cv2.dnn.blobFromImage(image, 1.0, (300, 300), (104.0, 177.0, 123.0))
```

```
# передаем изображение на вход нейронной сети
model.setInput(blob)
# выполняем логический вывод и получаем результат
output = np.squeeze(model.forward())
```

На следующем шаге необходимо обрмить прямоугольными рамками все обнаруженные в кадре лица. Для этого с использованием инструкции `for` организован цикл, в котором эти рамки вставляются в текущий кадр (по сути, в изображение).

```
# Рисуем прямоугольники вокруг обнаруженных лиц
font_scale = 0.5 # Размер шрифта для надписи достоверности
for i in range(0, output.shape[0]):
    # получить достоверность распознавания
    confidence = output[i, 2]
    # если достоверность выше 30%, то рисуем обрамляющий прямоугольник
    if confidence > 0.3:
        # получить координаты прямоугольника и масштабировать его
        box = output[i, 3:7] * np.array([w, h, w, h])
        # преобразовать координаты в целые числа
        start_x, start_y, end_x, end_y = box.astype(int)
        # рисуем прямоугольник вокруг лица
        cv2.rectangle(image, (start_x, start_y), (end_x, end_y),
                      color=(0, 255, 0), thickness=2)
        # добавляем текст с полученной достоверностью
        cv2.putText(image, f"{confidence * 100:.1f}%", (start_x, start_y - 5),
                   cv2.FONT_HERSHEY_SIMPLEX, font_scale, (0, 255, 0), 1)
```

Для рисования рамок вокруг найденных лиц используется метод `rectangle` библиотеки OpenCV, в который передается исходное изображение (`image`), координаты левого верхнего и правого нижнего угла рамок, цвет рамок зеленый в формате RGB (0, 255, 0), толщина рамки в пикселах (4).

В этом цикле рамки будут нарисованы только в тех областях изображения, где достоверность распознавания превышает 30%. К каждой рамке будет добавлен текст с полученной достоверностью распознанного лица.

На заключительном этапе метод `get_frame` возвращает текущий кадр из видеопотока в виде изображения формата «jpg», преобразованного в массив байтов. В данном случае мы не сохраняем изображение на диске в виде файла (оно находится в памяти компьютера), а класс `Camera_Face_resnet()` в конечном итоге формирует URL-адрес текущего кадра из памяти компьютера. В конечном итоге представление с именем `livefe_face_resnet` возвращает URL адрес последовательности кадров видео потока с видео камеры после их обработки в шаблон `face_cam_resnet.html`.

В интерактивной цифровой книге можно запустить на выполнение набор программных модулей данного раздела и посмотреть, как будут отображаться кадры видеопотока с камеры в шаблоне `face_cam_resnet.html`.

Примечание.

В интерактивной цифровой книге для запуска данного примера нужно на вкладке «Показ примеров» нажать на кнопку «Примеры главы 1» и активировать ссылку «Распознавание лиц с камеры с ResNet-10». Эта опция может не работать на стороне сервера из-за отсутствия SSL сертификата.

Внешний вид шаблона `face_cam_resnet.html` представлен на рис.1.25.

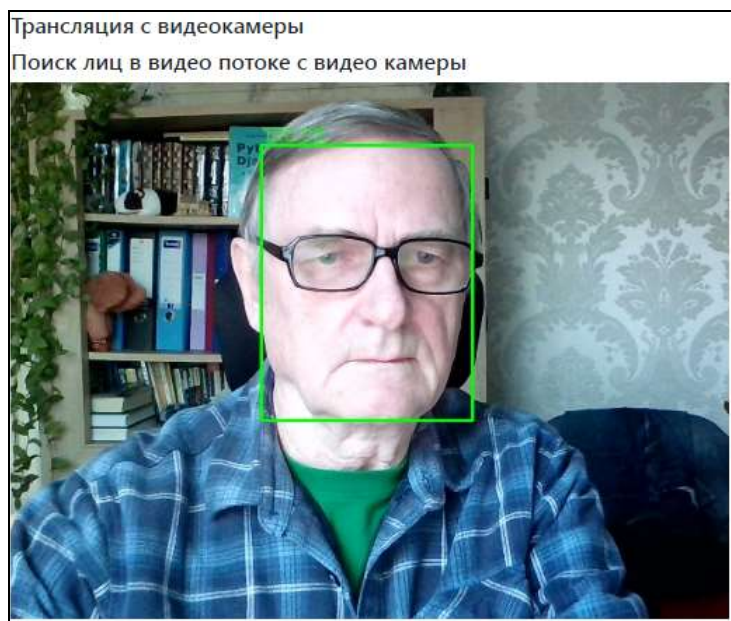


Рис. 1.25. Пример вывода видео потока с камеры в шаблон Django с распознаванием лица с использованием нейронной сети ResNet

Как видно из данного рисунка, достоверность распознавания лица на данном кадре равно 100%. Если выполнять наклоны и повороты головы, то можно убедиться, что в таких ракурсах нейронные сети более уверенно распознают лица, чем каскады Хаара.

1.8. Распознавание лицевых ориентиров с библиотекой Dlib

Реализация способа распознавания лицевых ориентиров (или контрольных точек) на изображении лица не является самоцелью. Это инструмент, открывающий различные возможности в системах искусственного интеллекта. В частности по лицевым ориентирам можно отслеживать мимику и эмоции, направление взгляда, а также производить верификацию и идентификацию личности.

Системы, определяющие эмоциональное состояние, могут значительно повысить уровень безопасности. Данные системы могут определять лица, потенциально настроенных на совершение противоправных действий, и информировать об этом правоохранительные органы до совершения возможных правонарушений. Также они позволят ограничить доступ людей к объектам повышенной опасности, таким как атомная энергетика, склады оружия, авиационная и дорожная техника. Системы определения направления взгляда и степени закрытия глаз у водителей обеспечит повышение безопасности дорожного движения, способствуют снижению аварийности и смертности на дорогах за счет предотвращения засыпания.

Идентификация личности позволит упростить доступ людей к различным устройствам и системам, например, в современные смартфоны встроен сканер лица для открытия доступа к его функциям. В банках используется биометрия лица вместо пин-кода, на транспорте для

автоматизации оплаты проезда (прохода в транспортное средство), системы распознавания лиц широко внедряется в камеры видеонаблюдения в общественных местах и позволяют выявлять лиц, находящихся в розыске.

Для решения задач распознавания лицевых ориентиров необходимо выполнить установку на компьютер специализированной библиотеки Dlib. Эта библиотека содержит алгоритмы и инструменты для обработки изображений лиц, она используется во многих коммерческих проектах, где работает «под капотом».

Для локализации контрольных точек можно воспользоваться готовой обученной моделью (shape_predictor_68_face_landmarks.dat), которую можно скачать с официального сайта разработчика данной библиотеки. Эта модель позволяет выделить на изображении лица 68 контрольных точек. Распределение точек по группам и их номера можно посмотреть в таблице 1.2.

Таблица 1.2
Распределение точек по группам в библиотеке Dlib

Элемент лица	Номера точек	Количество точек
Контур лица	от 1 до 17	17
Бровь правая	от 18 до 22	5
Бровь левая	от 23 до 27	5
Нос	от 28 до 36	9
Глаз правый	от 37 до 42	6
Глаз левый	от 43 до 48	6
Губы	от 49 до 68	20

Индексы и положение 68 точек лицевых ориентиров на изображении лица приведены на рис.1.26.

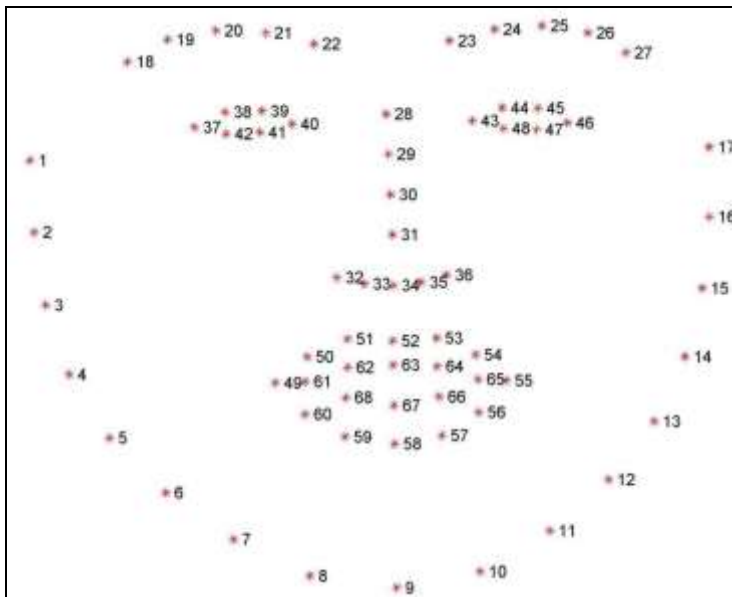


Рис. 1.26. Индексы и положение 68 точек лицевых ориентиров

Для того чтобы реализовать определение на изображениях лицевых ориентиров в Python необходимо установить библиотеку Dlib, а сделать это не совсем просто, особенно под windows. Установка данной библиотеки напоминает «танцы с бубном», которые не всегда приводят к желаемому результату. Дело в том, что данная библиотека написана на C++ и ее необходимо собрать на компьютере пользователя из исходного кода. Для этого требуется установить ряд дополнительных и достаточно громоздких модулей: библиотеки Cython, cmake и пакет MS Visual Studio, а с ними устанавливается еще целый ряд зависимых пакетов. Но и это не гарантирует успеха. В частности мне стоило больших трудов установить dlib на Python под Windows. Если использовать данную библиотеку с Django на публичном сервере, то необходимо устанавливать библиотеку dlib в linux, так как именно эта операционная система используется на публичных серверах. К счастью установка dlib в linux выполняется значительно проще.

В linux нужно открыть терминал и обновить установщик с использованием следующей команды:

```
sudo apt-get update
```

После этого необходимо установить пакет cmake:

```
sudo apt-get install cmake
```

На следующем этапе необходимо установить модуль boost python

```
sudo apt-get install libboost-all-dev
```

После этого можно, либо в терминале активировать нужное виртуальное окружение, либо в PyCharm открыть свой проект и в его виртуальное окружение установить dlib:

```
pip install dlib
```

Здесь нужно запастись терпением, так как установка и сборка пакета идет достаточно долго, порядка 10 мин. Итак, нужные библиотеки установлены, и можно приступить к написанию программного кода приложения.

При установке данной библиотеки на публичный сервер могут возникнуть некоторые проблемы, связанные с нехваткой памяти. В частности на арендованном сервере компании timeweb данная библиотека отказывалась устанавливаться на VDS сервер с объемом оперативной памяти 1 Гб. В этом случае нужно либо арендовать сервер с большим объемом оперативной памяти, либо создавать на диске раздел Swap (виртуальную память).

Примечание.

Swap в Linux — это файл подкачки, или специальный раздел на жёстком диске, расширяющий объем ОЗУ в случае, если его не хватает. Это один из механизмов виртуальной памяти, при котором отдельные фрагменты памяти (обычно неактивные) перемещаются из ОЗУ во вторичное хранилище (отдельный раздел или файл), освобождая ОЗУ для загрузки других активных фрагментов памяти. Обычно swap подключается, если ОЗУ переполнена: часть неиспользуемой информации переносится в файл подкачки, а в ОЗУ появляется место для новых данных.

Проверить наличие или создать на диске раздел Swap можно с использованием в терминале следующих команд (из документации службы поддержки timeweb). Перед осуществлением каких-либо действий со swap выполните следующую команду, чтобы узнать его текущее расположение:

```
swapon -s
```

Если раздел swap не создан или имеет не ту конфигурацию, то в первую очередь нужно отключить swap:

```
swapoff -a
```

Далее измените размер swap-файла. Если файл отсутствует в системе, эта же команда позволит его создать:

```
sudo dd if=/dev/zero of=/swap bs=1M count=1024
```

Укажите в параметре count необходимый размер swap, например, 512, 1024 или 4096.

Подготовьте swap-файл:

```
chmod 600 /swap && mkswap /swap
```

Включите swap:

```
swapon /swap
```

Проверьте результат командой `swapon -s` (должен быть получен примерно следующий результат):

Filename	Type	Size	Used	Priority
/swap	file	1048572	0	-1

Чтобы внесенные изменения были применены при следующей загрузке сервера, нужно добавить информацию о swap в файл `/etc/fstab`. Для этого выполните команду:

```
echo "/swap swap swap defaults 0 0"| sudo tee -a /etc/fstab
```

При публикации материалов данной интерактивной книги на VDS сервер только после выполнения этих команд удалось выполнить установку библиотеки `dlib`.

1.8.1. Распознавание лицевых ориентиров в изображениях с библиотекой Dlib

В данном разделе мы рассмотрим возможность использования библиотек OpenCV и dlib для формирования ключевых лицевых точек на изображениях. Для этого в одном из шаблонов или элементе меню необходимо создать ссылку, которая активирует нужный URL маршрут (листинг 1.51).

Листинг 1.51. Ссылка активации маршрута face_point_img

Примечание.

В демо-версии книги удалены листинги программ.

По данной ссылке в приложении api_vision будет активирован URL маршрут с именем face_point_img (листинг 1.52).

Листинг 1.52. Маршрут URL с именем face_point_img

Примечание.

В демо-версии книги удалены листинги программ.

По этому маршруту будет активировано представление face_point_img (листинг 1.53).

Листинг 1.53. Представление (view) face_point_img

Примечание.

В демо-версии книги удалены листинги программ.

Это достаточно простой программный код. На первом шаге из папки проекта modul.py импортируется модуль с именем face_point.py. Затем было создано три объекта, содержащих пути к изображениям, и к модели библиотеки dlib:

- ❑ img_path_in – путь и имя файла исходного изображения;
- ❑ img_path_out1 - путь и имя файла обработанного изображения;
- ❑ predict_path - путь и имя модели библиотеки dlib .

После этого выполняется обращение к функции fp.start_point программного модуля обработки изображений face_point.py, в которую передаются созданные на предыдущем шаге пути к файлам с изображениями, и к модели библиотеки dlib. В программном модуле face_point.py происходит чтение с диска исходного изображения, его обработка и запись обработанных изображений на диск. Затем URL адреса исходного и обработанного изображений загружаются в объекты img_url_1, img_url_2, которые через контекстную переменную context передаются в шаблон face_point_img.html.

Шаблон с именем face_point_img.html имеет следующий код (листинг 1.54).

Листинг 1.54. Шаблон face_point_img.html

Примечание.

В демо-версии книги удалены листинги программ.

Это достаточно простой шаблон, в котором имеется два тега, отвечающих за вывод изображений из их URL адресов.

Теперь рассмотрим код программного модуля `face_point.py`, в котором собственно и происходит процесс обработки исходного изображения (листинг 1.55).

Листинг 1.55. Программный модуль `face_point.py`

Примечание.

В демо-версии книги удалены листинги программ.

В этом модуле на первом шаге делается импорт необходимых библиотек. После этого следует код стартовой функции `start_point`, которая принимает пути к файлам с изображениями (`img_path_in`, `img_path_out`) и путь к модели нейронной сети библиотеки `dlib` (`pr_path`). В теле данной функции создается два объекта:

- ❑ `detector` - детектор лица `dlib`;
- ❑ `predictor` - предсказатель ориентиров на лице.

Затем в объект `image` загружается исходно изображение, которое преобразуется в оттенки серого (объект `gray`). С использованием детектора на изображении выполняется обнаружения лица (или множества лиц), а результаты работы детектора сохраняются в объекте `rects` (прямоугольники). На данном этапе, по сути, с использованием библиотеки `dlib`, выполняется поиск лиц на изображении и выделение прямоугольных областей для каждого лица.

После этого с использованием инструкции `for` организован цикл для каждого обнаруженного лица, в теле которого выполняются следующие действия:

- ❑ определяются лицевые ориентиры на лице (объект `shape`);
- ❑ вокруг каждого найденного лица рисуется обрамляющая рамка и указывается номер обнаруженного лица;
- ❑ во вложенном цикле на каждое лицо наносятся точки лицевых ориентиров;
- ❑ координаты найденных точек записываются в файл с именем `«points.txt»`;
- ❑ выполняется закрытие текстового файла с координатами лицевых ориентиров.

В интерактивной цифровой книге можно запустить на выполнение набор программных модулей данного раздела и посмотреть, как в шаблоне `face_point_img.html` будут выглядеть точки лицевых ориентиров.

Примечание.

В интерактивной цифровой книге для запуска данного примера нужно на вкладке «Показ примеров» нажать на кнопку «Примеры главы 1» и активировать ссылку «Лицевые точки на изображении».

Внешний вид шаблона `face_point_img.html` представлен на рис.1.27.

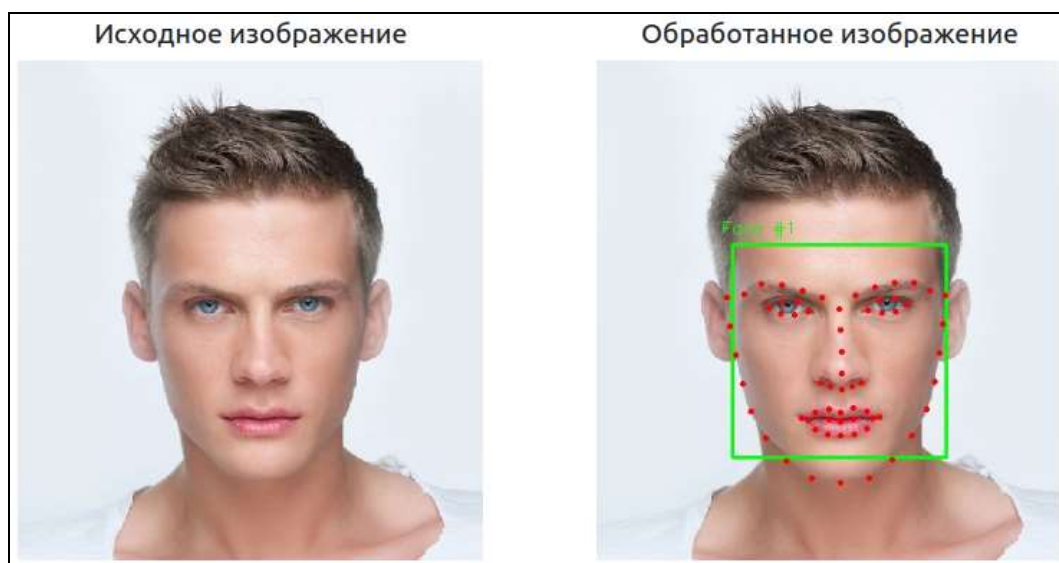


Рис. 1.27. Пример работы программы формирования лицевых ориентиров на изображении

Как видно из данного рисунка, программа корректно обнаружила лицо, выделила его прямоугольником, и обозначила все ключевые точки.

Как было сказано вначале данного раздела, формирование ключевых точек не является самоцелью, но на их основе системы компьютерного зрения и искусственного интеллекта могут решать различные практические задачи. В частности лицевые ориентиры служат основой построения диаграмм Вороного и триангуляции Делоне, которые используются в различных системах, связанных с распознаванием конкретных личностей в системах безопасности, в транспортных, банковских и мобильных приложениях.

1.9. Диаграмма Вороного и триангуляция Делоне

Диаграмму Вороного разработал русский математик Георгий Вороной задолго до революции, и она была названа его именем. Согласно его теореме конечное множество точек S на плоскости разбивает плоскость на многоугольники, каждая точка которых более близка к одному из элементов множества S , чем к любому другому элементу (рис. 1.28).



Рис. 1.28. Пример диаграммы Вороного

Диаграмма Вороного используется в совершенно разных областях. Это кристаллография, информатика, физика, электроника, астрономия, астрофизика, химия, микробиология, офтальмологии. Кроме того диаграмма Вороного используется при решении задач компьютерного зрения и искусственного интеллекта, то есть везде, где возникает потребность в расслоении пространства оптимальным образом на отдельные области.

Триангуляция - это метод разбиения плоскости (или 3D пространства) на треугольники с особыми свойствами. В отличие от диаграммы Вороного при триангуляции плоскость разбивается не на многоугольники, а на треугольники. При этом для любого треугольника все точки из массива S (за исключением точек, являющихся его вершинами), лежат вне окружности, описанной вокруг треугольника. Впервые алгоритмы триангуляции были разработаны в 1934 г. советским математиком Борисом Делоне и названы его именем (рис.1.29).

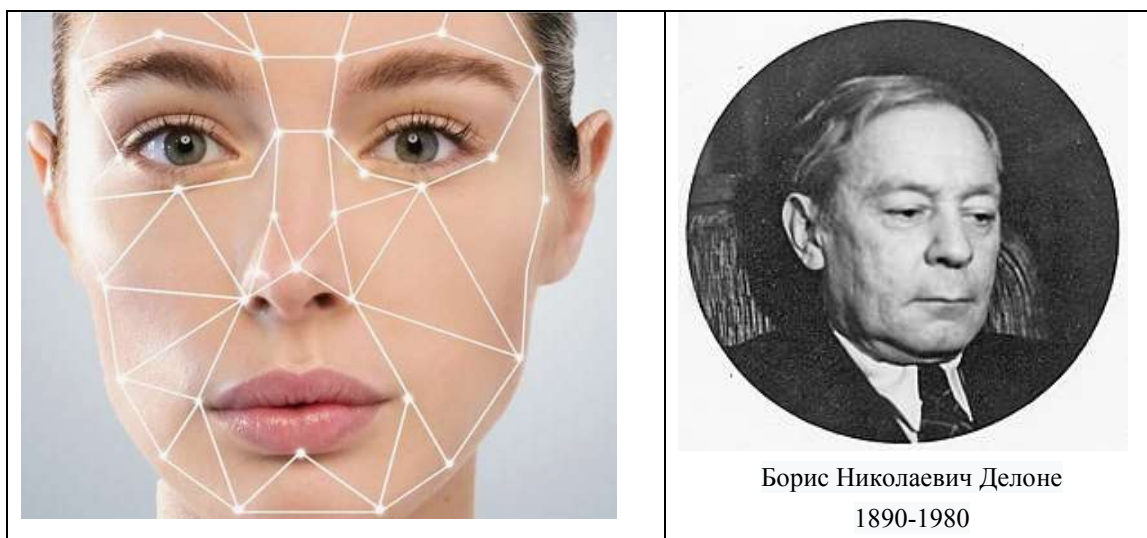


Рис. 1.29. Пример триангуляции Делоне

В компьютерной графике триангуляция Делоне используется для представления сложных объектов в виде сетки треугольников. Это особенно важно для моделирования поверхностей объектов (например, в играх на основе трехмерной графики). В географических информационных системах триангуляция Делоне используется для построения цифровых моделей местности или при создании карт. В некоторых алгоритмах машинного обучения триангуляция помогает создавать более компактные и информативные представления данных. В инженерии, например, при проектировании электрических сетей, водоснабжения или коммуникационных систем, триангуляция Делоне используется для оптимизации распределения ресурсов.

1.9.1. Создание диаграммы Вороного и триангуляции Делоне для лица

В данном разделе мы рассмотрим возможность использования библиотеки OpenCV для создания диаграммы Вороного и триангуляции Делоне по ключевым точкам лица на изображении. Для этого в одном из шаблонов или элементе меню необходимо создать ссылку, которая активирует нужный URL маршрут (листинг 1.56).

Листинг 1.56. Ссылка активации маршрута `triang_delone_img`

Примечание.

В демо-версии книги удалены листинги программ.

По данной ссылке в приложении `api_vision` будет активирован URL маршрут с именем `triang_img` (листинг 1.57).

Листинг 1.57. Маршрут URL с именем `triang_img`

Примечание.

В демо-версии книги удалены листинги программ.

По этому маршруту будет активировано представление `triang_img` (листинг 1.58).

Листинг 1.58. Представление (view) `triang_img`

Примечание.

В демо-версии книги удалены листинги программ.

Это достаточно простой программный код. На первом шаге из папки проекта `modul_py` импортируется модуль с именем `Triangulator.py`. Затем было создано три объекта, содержащих пути к изображениям:

- ❑ `img_path_in` – путь и имя файла исходного изображения;
- ❑ `img_path_out1` - путь и имя файла обработанного изображения (триангуляция Делоне);
- ❑ `img_path_out2` - путь и имя файла обработанного изображения (диаграмма Вороного).

После этого выполняется обращение к функции `start_trian` модуля обработки изображений `Triangulator.py`, в который передаются созданные на предыдущем шаге пути к файлам с изображениями. В программном модуле `Triangulator.py` происходит чтение с диска исходного изображения, его обработка и запись обработанных изображений на диск. Затем формируются URL адреса трех изображений, которые загружаются в объекты `img_url_1`, `img_url_2`, `img_url_3`. Эти URL адреса через контекстную переменную `context` передаются в шаблон `triang_img.html`.

Шаблон с именем `triang_img.html` имеет следующий код (листинг 1.59).

Листинг 1.59. Шаблон `triang_delone_img.html`

Примечание.

В демо-версии книги удалены листинги программ.

Это достаточно простой шаблон, в котором имеется три тега, отвечающих за вывод изображения из их URL адресов.

Теперь рассмотрим код программного модуля `Triangulator.py`, в котором собственно и происходит процесс обработки исходного изображения (листинг 1.60).

Листинг 1.60. Программный модуль `Triangulator.py`

Примечание.

В демо-версии книги удалены листинги программ.

В этом программном модуле стартовой является функция с именем `start_trian`. Данная функция принимает пути к файлам с изображениями (`img_path_in`, `img_path_out1`, `img_path_out2`). В теле функции в объект `img` загружается исходное изображение, определяется его размер (объект `size`) и формируется прямоугольная область такого же размера, куда будут вставлены ключевые лицевые точки (объект `rect`).

После этого формируется пустое двумерное пространство Делоне (объект `subdiv`), куда можно будет добавлять ключевые лицевые точки с помощью функции `insert()`.

Затем создается пустой массив для таких точек (объект – `points`) и они загружаются из файла `points.txt` (данный файл был создан программным модулем предыдущего раздела). После этого выполняется обращение к функции рисования треугольников Делоне (`draw_delaunay`), в которую передается исходное изображение (`img`) и пространство Делоне (`subdiv`). Обработанное данной функцией изображение с треугольниками Делоне записывается в файл. Функция `draw_delaunay` для рисования треугольников использует ряд дополнительных функций.

После этого запускается процесс формирования диаграммы Вороного. Организован цикл с использованием инструкции `for`, в теле которого лицевые точки наносятся на изображение (`img`) с использованием функции `draw_point()`. Затем формируется пустое двумерное пространство для диаграммы Вороного (объект `img_voronoi`). На следующем шаге делается обращение к функции рисования диаграммы Вороного – `draw_voronoi()`, в которую передаются объекты `img_voronoi` и `subdiv`. На заключительном этапе диаграмма Вороного записывается в файл.

В интерактивной цифровой книге можно запустить на выполнение набор программных модулей данного раздела и посмотреть, как будет выглядеть триангуляция Делоне и диаграмма Вороного в шаблоне `triang_img.html`.

Примечание.

В интерактивной цифровой книге для запуска данного примера нужно на вкладке «Показ примеров» нажать на кнопку «Примеры главы 1» и активировать ссылку «Триангулятор для изображения».

Внешний вид шаблона `triang_img.html` представлен на рис.1.30.

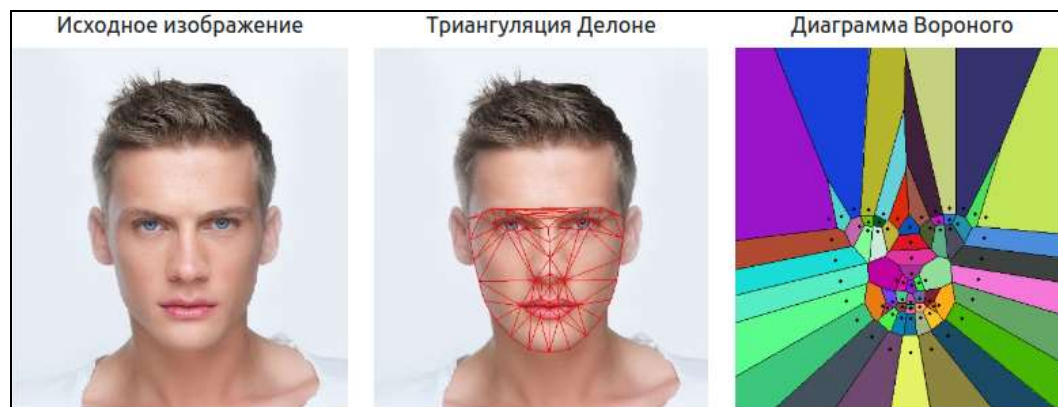


Рис. 1.30. Пример создания триангуляции Делоне и диаграммы Вороного на изображении

1.10. Краткие итоги

Итак, мы познакомились с некоторыми базовыми основами компьютерного зрения, библиотеками и алгоритмами, используемыми для обработки изображений, видеофайлов и потокового видео в веб приложениях на Django.

Теперь можно приступить к изучению более сложных алгоритмов и программных модулей, которые обеспечивают обработку изображений с использованием нейронных сетей. Из материалов следующей главы вы узнаете, как можно стилизовать изображения с использованием нейронных сетей с выводом результатов в шаблоны Django.

ГЛАВА 2



Стилизация изображений с использованием нейронных сетей

Передача нейронного стиля NST (neural style transfer) используется в программных продуктах, которые обрабатывают цифровые изображения (включая видео). При этом за основу берется содержимое исходного изображения, на которое накладывается стиль (внешнее оформление) другого изображения. Алгоритмы NST для преобразования изображений предполагает использование нейронных сетей. Чаще всего технология NST используется для переноса стиля известных картин на фотографии, предоставленные пользователями. Этот метод использовался художниками и дизайнерами для создания рекламного контента или новых картин на основе существующих стилей. В этой главе мы не будем разбирать теоретические основы и методику стилизации изображений, а узнаем, как можно реализовать эту технологию на языке программирования Python в локальных и web приложениях на Django. Из материалов данной главы вы узнаете:

- ☐ как можно стилизовать изображения с использованием предварительно обученных нейронных сетей;
- ☐ как создать базовый модуль стилизации изображений на Python и отладить его работу на локальном компьютере;
- ☐ как подключить базовый модуль стилизации изображений к веб приложению;
- ☐ как можно выполнить стилизацию кадров потокового видео;
- ☐ как можно создать видеофайл с плавным изменением стиля изображения и реализаций этого подходы в веб приложениях;
- ☐ как можно использовать библиотеку PyTorch и модель нейронной сети 21styles.model для переноса любого стиля на любое изображение с использованием методов машинного обучения.

2.1. Стилизация изображений с библиотекой OpenCV и предварительно обученных нейронных сетей

В данном разделе будет показано, как можно применять алгоритм нейронной передачи стиля с использованием библиотеки OpenCV и Python. Метод, который здесь будет использован, способен работать очень быстро (практически в реальном времени) даже на центральном процессоре, и обеспечивает сверхбыструю производительность на графическом процессоре.

Высокая скорость стилизации этого метода обеспечивается тем, что в нем используется предварительно обученные нейронные сети. После того, как сеть обучена формировать

определенный стиль, ее можете использовать для стилизации любого изображения. Начнем с реализации простого примера реализации нейронной передачи стиля с использованием библиотеки OpenCV.

Для того чтобы реализовать возможность стилизации изображений необходимо к своему проекту добавить набор необходимых библиотек с использованием команды `pip`:

```
pip install imutils
pip install opencv-python
```

Кроме того необходимо скачать предварительно обученные модели нейронных сетей. Для данного примера будет использовано 9 моделей, для которых мы создадим отдельный каталог с именем `net_model`:

- ❑ `the_wave.t7`;
- ❑ `udnie.t7`;
- ❑ `the_scream.t7`;
- ❑ `starry_night.t7`;
- ❑ `mosaic.t7`;
- ❑ `la_muse.t7`;
- ❑ `feathers.t7`;
- ❑ `composition_vii.t7`;
- ❑ `models/candy.t7`.

2.1.1. Базовый модуль Python быстрого переноса нейронного стиля для настольных приложений

Начнем с создания базовой функции, которая будет выполнять стилизацию изображений (листинг 2.1).

Листинг 2.1. Модуль `Style_1.py` для стилизации изображений

Примечание.

В демо-версии книги удалены листинги программ.

В первых строках данной функции выполняется импорт библиотек `imutils` и `cv2`. Первая служит для изменения размера изображения, вторая для чтения исходного изображения и записи на диск обработанного изображения, всю основную работу по стилизации будет выполнять нейронная сеть.

Программные строки за пределом тела функции служат для отладки работа программного модуля. Здесь инициализированы следующие переменные:

- ❑ `inp_img = '../media/images/Lexa.jpg'` - путь к исходному изображению;
- ❑ `out_img = '../media/images/style_img_fast.jpg'` - путь к обработанному изображению;
- ❑ `model = '../net_model/mosaic.t7'` - путь к модели стиля.

Примечание.

Пути к файлам указаны в соответствии с требованиями ОС Windows. Если вы работаете в Linux, то необходимо будет изменить синтаксис в соответствии с требованиями Linux.

После этого делается обращение к функции `start_style()`, в которую передаются значения созданных переменных.

В теле функции объект `net` принимает предварительно обученную нейронную сеть, а в объект `image` загружается исходное изображение. Затем следует группа команд, выполняющих стилизацию, а обработанный файл сохраняется на диск.

Данный программный модуль можно запустить на выполнение в среде Python и проверить корректность его работы на локальном компьютере. Но наша конечная цель интегрировать его в среду Django и обеспечить его работу в рамках web приложения. Как это можно сделать на уровне программного кода, показано в следующем разделе, а принципиальная схема интеграции модулей Python с веб приложением на Django приведена на рис.2.1.

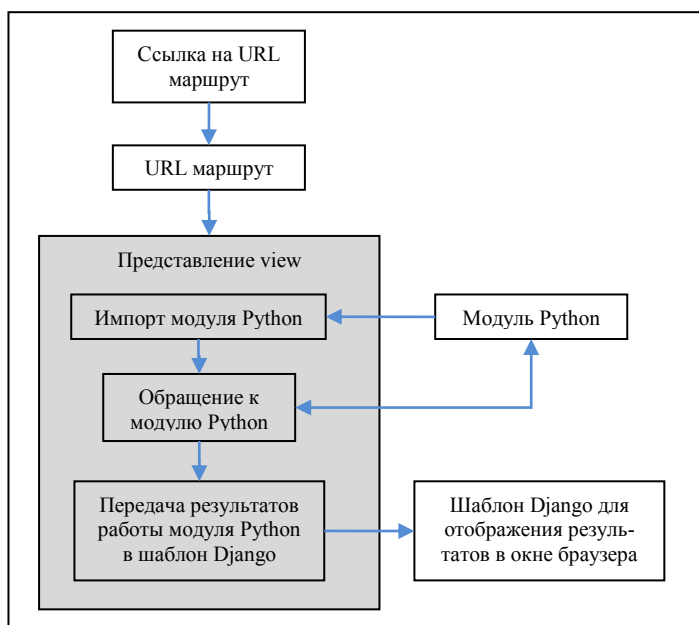


Рис. 2.1. Принципиальная схема интеграции модулей Python с веб приложением на Django

При такой структуре приложения модули Python отделены от кода модулей Django, их работу можно отлаживать отдельно, располагать в отдельных папках и просто подключать к различным представлениям (view) с помощью инструкций импорта (`import <имя_модуля>`)

2.1.2. Использование функции быстрого переноса нейронного стиля в web приложениях

А теперь покажем, как можно встроить программные модули, о которых говорилось в предыдущем разделе, в web приложение и вывести стилизованное изображение в шаблон Django. Для этого в одном из шаблонов или элементе меню необходимо создать ссылку, которая активирует нужный URL маршрут (листинг 2.2).

Листинг 2.2. Ссылка активации маршрута fast_style

Примечание.

В демо-версии книги удалены листинги программ.

По данной ссылке в приложении `api_vision` будет активирован URL маршрут с именем `fast_style` (листинг 2.3).

Листинг 2.3. Маршрут URL с именем fast_style

Примечание.

В демо-версии книги удалены листинги программ.

По этому маршруту будет активировано представление `fast_style` (листинг 2.4).

Листинг 2.4. Представление (view) fast_style

Примечание.

В демо-версии книги удалены листинги программ.

В первой строке этого программного кода выполняется импорт модуля `Style_1` (см. предыдущий раздел). Затем были созданы переменные, в которые загружены пути к файлам с изображениями и к модели нейронной сети:

- ❑ `img_path_in = os.path.join(media_dir, 'images/Lexa.jpg')` - путь к исходному изображению;
- ❑ `img_path_out = os.path.join(media_dir, 'images/style_img_fast.jpg')` - путь к обработанному изображению;
- ❑ `style_model = os.path.join(base_dir, input_model)` - путь к модели нейронной сети.

Обратите внимание, что в проекте изображения находятся в папке `media/images`, модель нейронной сети в папке `net_model`, которая является дочерней по отношению к базовой директории проекта.

На следующем шаге выполняется обращение к классу `start_style`, где и выполняется стилизация исходного изображения. После этого формируются URL адреса входного и обработанного изображений, они загружаются в контекстную переменную и передаются в шаблон с именем `style_img.html`. Этот шаблон имеет следующий код (листинг 2.5).

Листинг 2.5. Шаблон style_img.html

Примечание.

В демо-версии книги удалены листинги программ.

Это достаточно простой шаблон, в котором имеется два контейнера: первый контейнер отвечает за вывод исходного изображения, второй – за вывод стилизованного изображения. Теперь можно проверить работу данных модулей.

Примечание.

В интерактивной цифровой книге для запуска данного примера нужно на вкладке «Показ примеров» нажать на кнопку «Примеры главы 2» и активировать ссылку «Перенос нейронного стиля с OpenCV».

Если выполнить набор вышеприведенных моделей, то получим следующий результат (рис.2.2).



Рис. 2.2. Фотография, стилизованная предварительно обученной нейронной сетью

Стилизация изображений предварительно обученной нейронной сетью выполняется очень быстро, и она может использоваться для стилизации кадров видео фалов, и кадров прямого медиа потока с видеокамеры. В интерфейсе веб приложения можно реализовать выбор стиля и загрузку изображения, обеспечив, таким образом, стилизацию любого изображения любым выбранным стилем.

2.1.3.Модуль стилизация видеопотока с web камеры в настольных приложениях

Теперь проверим, действительно ли предварительно обученная нейронная сеть сможет на ходу стилизовать кадры видеопотока с web камеры ноутбука или настольного компьютера. Для этого напишем небольшой программный код (листинг 2.6).

Листинг 2.6. Модуль Styl_camera.py для стилизации видео потока

Примечание.

В демо-версии книги удалены листинги программ.

Это достаточно простой программный модуль, в работе которого задействована библиотека OpenCV и девять предварительно обученных моделей нейронных сетей. В начале программы выполнен импорт библиотеки OpenCV. Затем, после кода функции styling, формируются следующие элементы:

- ❑ `input_model` - объект, содержащий путь к модели нейронной сети;
- ❑ `net` – объект с загруженной нейронной сетью;
- ❑ `cap` – объект, иницирующий видео поток с видео камеры.

Примечание.

В этом варианте объект `cap` будет успешно работать под ОС Windows. Если вы работаете под Linux, то эта строка должна выглядеть следующим образом - `cap = cv2.VideoCapture(0)`

После этого с использованием инструкции `while` создан бесконечный цикл, в теле которого объект `frame` захватывает текущий кадр с видеокамеры. После этого вызывается функция `styling`, в которую передается захваченный кадр и модель нейронной сети. Функция `styling` возвращает стилизованный кадр, который отображается на экране с использованием метода `cv2.imshow`. Для прерывания видео потока пользователь должен прервать бесконечный цикл, нажав на клавишу `Esc`. После этого трансляция видео потока прекратится и окно, в которое он выводился, будет закрыто.

Примечание.

Представленный в данном разделе программный код может работать только на локальном компьютере.

Если запустить программный код модуля `Styl_camera.py` на выполнение, то будет получен следующий результат (рис.2.3).



Рис. 2.3. Стилизованный кадр видео потока с web камеры предварительно обученной нейронной сетью

На данном примере приведена стилизация изображения с использованием модели `mosaic.t7`. Обработка одного кадра без использования ускорения на GPU занимает примерно 0.67 секунды. Поскольку видеопоток транслируется со скоростью от 24 до 30 кадров в секунду, то будет наблюдаться некоторая задержка вывода видеопотока. Если компьютер будет иметь видеокарту `nvidia` с графическим ускорителем и памятью порядка 4-8 Гб, то такой задержки наблюдаться не будет.

2.1.4. Модуль создания видео файла с плавным изменением стиля изображения в настольных приложениях

С использованием библиотеки `OpenCV` можно создать видео ролик, который будет демонстрировать плавное изменение стиля изображения. Реализуем алгоритм плавного перетекания стиля в модуле с именем `Make_Styl_Video.py` (листинг 2.7).

Листинг 2.7. Модуль `Make_Styl_Video.py` для плавной стилизации изображения

Примечание.

В демо-версии книги удалены листинги программ.

Это достаточно простой программный модуль, в работе которого задействована библиотека `OpenCV` и девять предварительно обученных моделей нейронных сетей. В начале программы выполнен импорт библиотеки `OpenCV`. Затем, после кода функции `style_video`, формируются следующие элементы:

- ❑ `input_img_path` - объект, содержащий путь к исходному изображению;
- ❑ `out_img_path` - объект, содержащий путь к стилизованному изображению;
- ❑ `video_out_path` - объект, содержащий путь к итоговому видео файлу;
- ❑ `model_path` - объект, содержащий путь к модели нейронной сети.

Примечание.

Пути к файлам указаны в соответствии с требованиями ОС `Windows`. Если вы работаете в `Linux`, то необходимо будет изменить синтаксис в соответствии с требованиями `Linux`.

После этого вызывается функция `style_video`, в которую передаются упомянутые выше объекты. Функция `style_video` сначала стилизует полученное изображение и сохраняет его на диске. Затем создается объект `out`, который будет принимать последовательность кадров и записывать их в видео файл. Для формирования видеокадров создано два цикла `for`. В теле этих циклов происходит смешивание исходного и стилизованного изображения с использованием метода `cv2.addWeighted` библиотеки `OpenCV`. Последовательность этих кадров записывается в видео файл с использованием инструкции `out.write(img)`. Сохраненный на диске видео файл можно просмотреть с использованием видео проигрывателя.

Данный программный модуль можно запустить на выполнение в среде `Python` и проверить корректность его работы на локальном компьютере. Но наша конечная цель интегрировать его в среду `Django` и обеспечить работы в рамках `web` приложения. Как это можно сделать показано в следующем разделе.

2.1.5. Использование модуля плавного переноса нейронного стиля в web приложениях

А теперь покажем, как можно встроить программный модуль, о котором говорилось в предыдущем разделе, в web приложение и вывести видео плавной стилизации изображения в шаблон Django. Для этого в одном из шаблонов или элементе меню необходимо создать ссылку, которая активирует нужный URL маршрут (листинг 2.8).

Листинг 2.8. Ссылка активации маршрута show_style_video

Примечание.

В демо-версии книги удалены листинги программ.

По данной ссылке в приложении api_vision будет активирован URL маршрут с именем show_style_video (листинг 2.9).

Листинг 2.9. Маршрут URL с именем show_style_video

Примечание.

В демо-версии книги удалены листинги программ.

По этому маршруту будет активировано представление show_style_video (листинг 2.10).

Листинг 2.10. Представление (view) show_style_video

Примечание.

В демо-версии книги удалены листинги программ.

В первой строке этого программного кода выполняется импорт модуля Make_Styl_Video (см. предыдущий раздел). Затем были созданы переменные, в которые загружены пути к файлам с изображениями и к модели нейронной сети:

- ❑ input_model = 'net_model/the_wave.t7' - путь к модели нейронной сети с заданным стилем;
- ❑ img_path_in = os.path.join(media_dir, 'images/Toma.jpg') - полный путь к исходному изображению;
- ❑ img_path_out = os.path.join(media_dir, 'images/style_img_vid.jpg') - полный путь к обработанному (стилизованному) изображению;
- ❑ video_out_path = os.path.join(media_dir, 'video/style_img_vid.jpg') - полный путь к итоговому видео файлу;
- ❑ model_path = os.path.join(base_dir, input_model) – полный путь к модели нейронной сети с заданным стилем.

Обратите внимание, что в проекте изображения находятся в папке `media/images`, модель нейронной сети в папке `net_model`, которая является дочерней по отношению к базовой директории проекта.

На следующем шаге выполняется обращение к функции `style_video`, где выполняется стилизация исходного изображения и формирование видео файла. После этого формируются URL адрес итогового видео файла, который загружается в контекстную переменную и передается в шаблон с именем `show_style_video.html`. Этот шаблон имеет следующий код (листинг 2.11).

Листинг 2.11. Шаблон `show_style_video.html`

Примечание.

В демо-версии книги удалены листинги программ.

Это достаточно простой шаблон, в котором имеется два контейнера: первый отвечает за вывод текстовой строки, второй – за вывод видео файла с плавной стилизацией изображения. Теперь можно проверить работу данных модулей.

Примечание.

В интерактивной цифровой книге для запуска данного примера нужно на вкладке «Показ примеров» нажать на кнопку «Примеры главы 2» и активировать ссылку «Плавная стилизация изображения».

Если выполнить набор вышеприведенных моделей, то получим следующий результат (рис.2.4).



Рис. 2.4. Видео ролик с плавной сменой стиля изображения

В этом примере исходное изображение было стилизовано с использованием модели `the_wave.t7`. Если проиграть данный видео ролик, то исходное изображение плавно перетекает в стилизованное, а потом также плавно вернется в исходное состояние. Можно последовательно использовать разные стили для стилизации исходного изображения, то есть создавать привлекательные рекламные видео ролики. В интерфейсе веб приложения можно реализовать выбор стиля и загрузку изображения, обеспечив, таким образом, стилизацию любого изображения любым выбранным стилем.

2.2. Стилизация изображений с использованием библиотеки PyTorch и нейронных сетей

В этом разделе будет рассмотрено практическое применение методов машинного обучения в области компьютерного зрения, называемых передачей нейронного стиля. В программных модулях данного раздела будут использоваться методы глубокого обучения и алгоритмы, которые позволяют смешивать содержание одного изображения с художественным стилем другого. Мы не будем вдаваться в теорию и в математическое описание сути данных методов, а сосредоточимся на их практическом использовании в настольных и веб приложениях.

Обработка данных при работе с моделями машинного обучения может потребовать больших затрат памяти, поэтому для выполнения некоторых модулей данного раздела на вашем компьютере должно быть не менее 8 ГБ памяти.

2.2.1. Базовый модуль Python для переноса нейронного стиля

Для того чтобы реализовать возможность стилизации изображений с использованием библиотеки PyTorch необходимо к своему проекту добавить набор необходимых библиотек с использованием команды `pip`:

```
pip install torch
pip install numpy
pip install pillow
```

Кроме того в проект нужно будет загрузить базовую модель нейронной сети. В данном случае мы используем модель нейронной сети с именем `21styles.model`, которая имеет размер порядка 10 Мб.

Примечание.

Модель `21styles.model`, используется совместно с PyTorch-Multi-Style-Transfer для стилизации изображений с помощью предварительно обученной модели MSG-Net. Данная модель позволяет применять сложные алгоритмы и методы для эффективного анализа и обработки данных. PyTorch — это фреймворк для языка программирования Python, предназначенный для машинного обучения. Он включает в себя набор инструментов для работы с моделями, используется в обработке естественного языка, в системах компьютерного зрения и других похожих направлениях.

Нужно учесть, что для разных операционных систем (Windows, Linux), и разных версий Python нужно будет подобрать свой набор совместимых версий вышеприведенных библиотек.

Начнем с создания базового класса, который будет инициировать стилизацию изображений (листинг 2.12).

Листинг 2.12. Модуль `M_S_21.py` для стилизации изображений

Примечание.

В демо-версии книги удалены листинги программ.

Мы не будем делать подробный анализ этого программного кода, остановимся только на его ключевых моментах. В самом начале программы выполняется импорт необходимых

библиотек, и импорт программного модуля `net`. Модуль `net` содержит набор функций и классов, которые отвечают за конфигурацию и работу нейронной сети (листинг 2.13).

Листинг 2.13. Модуль `net.py`, используемый при стилизации изображений

Примечание.

Это достаточно сложный и длинный программный код, имеющий около 300 строк, поэтому здесь мы не будем останавливаться на его структуре, но приведем его полный код (см. листинг в Приложении 1).

В нижней части этого листинга находятся строки, которые использовались для отладки работы данного модуля. В этих строках задано несколько переменных:

- ❑ `in_image = './media/images/Men1.jpg'` – путь к исходному изображению;
- ❑ `out_image = './media/images/style_img.jpg'` – путь к обработанному изображению;
- ❑ `in_styl = './media/styles/pencil.jpg'` – путь к изображению с базовым стилем;
- ❑ `styl_model = './net_model/21styles.model'` – путь к модели нейронной сети;
- ❑ `size = 1024` - ширина обработанного изображения.

Примечание.

Пути к файлам указаны в соответствии с требованиями ОС Windows. Если вы работаете в Linux, то необходимо будет изменить синтаксис в соответствии с требованиями Linux.

После этого делается обращение к классу `styl_21()`, в который передаются значения созданных переменных.

В теле класса `styl_21()` иницируются внутренние переменные (`self`), которые принимают переданные в класс параметры. Затем создаются объекты для стилизации изображения:

- ❑ `content_image` - принимает исходное изображение;
- ❑ `style` - принимает изображение с базовым стилем;
- ❑ `style_model` – формирует конфигурацию нейронной сети;
- ❑ `model_dict` - принимает модель нейронной сети.

Далее включается в работу магия нейросетевых технологий, в результате чего на выходе будет получено стилизованное изображение.

Данный программный модуль можно запустить на выполнение в среде Python и проверить корректность его работы на локальном компьютере. Но наша конечная цель интегрировать его в среду Django и обеспечить работы в рамках web приложения. Как это можно сделать показано в следующем разделе.

2.2.2. Использование базового модуля переноса нейронного стиля в web приложениях

А теперь покажем, как можно встроить программные модули, о которых говорилось в предыдущем разделе, в web приложение и вывести стилизованное изображение в шаблон Django. Для этого в одном из шаблонов или элементе меню необходимо создать ссылку, которая активирует нужный URL маршрут (листинг 2.14).

Листинг 2.14. Ссылка активации маршрута style_img

Примечание.

В демо-версии книги удалены листинги программ.

По данной ссылке в приложении `api_vision` будет активирован URL маршрут с именем `style_img` (листинг 2.15).

Листинг 2.15. Маршрут URL с именем style_img

Примечание.

В демо-версии книги удалены листинги программ.

По этому маршруту будет активировано представление `style_img` (листинг 2.16).

Листинг 2.16. Представление (view) style_img

Примечание.

В демо-версии книги удалены листинги программ.

В первой строке этого программного кода выполняется импорт модуля `M_S_21` (см. предыдущий раздел). В данном случае модуль `net` был встроен в программный модуль `M_S_21.py` для того, чтобы все необходимые элементы были импортированы одной командой. Затем были созданы переменные, в которые загружены пути к файлам с изображениями и к модели нейронной сети:

- ❑ `img_path_in = os.path.join(media_dir, 'images/Men1.jpg')` - путь к исходному изображению;
- ❑ `img_path_out = os.path.join(media_dir, 'images/style_img.jpg')` - путь к обработанному изображению;
- ❑ `img_path_style = os.path.join(media_dir, 'styles/pencil.jpg')` - путь к изображению с базовым стилем;
- ❑ `style_model = os.path.join(base_dir, 'net_model/21styles.model')` - путь к модели нейронной сети;
- ❑ `size = 1024` - размер выходного файла.

Обратите внимание, что в проекте изображения находятся в папке `media/images`, изображения с базовым стилем в папке `media/styles`, модель нейронной сети в папке `net_model`, которая является дочерней по отношению к базовой директории проекта.

На следующем шаге выполняется обращение к классу `styl_21`, где и выполняется стилизация исходного изображения. После этого формируются URL адреса входного и обработанного изображений, они загружаются в контекстную переменную и передаются в шаблон с именем `style_img.html`. Этот шаблон имеет следующий код (листинг 2.17).

Листинг 2.17. Шаблон style_img.html

Примечание.

В демо-версии книги удалены листинги программ.

Это достаточно простой шаблон, в котором имеется два контейнера: первый отвечает за вывод исходного изображения, второй – за вывод стилизованного изображения. Теперь можно проверить работу этих программных модулей

Примечание.

В интерактивной цифровой книге для запуска данного примера нужно на вкладке «Показ примеров» нажать на кнопку «Примеры главы 2» и активировать ссылку «Рисунок в карандаше с PyTorch».

Внешний вид шаблона style_img.html представлен на рис.2.5.



Рис. 2.5. Фотография, стилизованная нейронной сетью под карандашный рисунок

В данном примере подключено изображение с базовым стилем pencil.jpg (карандашный рисунок). В директории styles находятся файлы со следующими базовыми стилями изображений (рис.2.6):



 frida_kahlo.jpg	 la_muse.jpg	 mosaic.jpg	 mosaic_ducks_massimo.jpg
 noise.jpg	 pencil.jpg	 picasso_selfport1907.jpg	 rain_princess.jpg
 Portrait.jpg	 seated-nude.jpg	 shipwreck.jpg	 starry_night.jpg
 stars2.jpg	 strip.jpg	 the_scream.jpg	 udnie.jpg
 wave.jpg	 woman-with-hat-matisse.jpg		

Рис. 2.6. Изображения с базовыми стилями

В интерфейсе пользователя нужно предусмотреть выбор базового стиля и загрузку исходного изображения. Но пока, для упрощения, в данном программном модуле зафиксировано стилизуемое изображение - Men1.jpg, изображение с базовым стилем - pencil.jpg. В интерфейсе веб приложения можно реализовать выбор стиля и загрузку изображения, обеспечив, таким образом, стилизацию любого изображения любым выбранным стилем.

2.3. Краткие итоги

Итак, мы познакомились с некоторыми основами использования нейронных сетей для стилизации изображений в настольных и веб приложениях. Теперь можно перейти к изучению возможности использования нейронных сетей в телеграмм ботах. Рассмотрению этих вопросов и посвящена следующая глава.

ГЛАВА 3

Интеграция нейронных сетей с Telegram ботами на Python



Большинству из вас приходилось сталкиваться с ботами в мобильном приложении Telegram. Боты помогают автоматизировать решение различных задач для обычных пользователей приложения Telegram. Чаще боты отвечают на простые (или типовые) вопросы, которые задают пользователи, но могут выполнить и достаточно сложную работу, например, бот может обратиться к внешнему интернет ресурсу и вернуть актуальный на данный момент курс валют, выслать рекламные материалы компании и т.п. В этой главе опишем процесс создания Telegram ботов на Python и их интеграции с нейронными сетями. В качестве примера в данной главе будут созданы варианты Telegram ботов, которые будут принимать изображения (фотографии) от пользователей, стилизовать их с использованием нейронных сетей и возвращать пользователю обработанные изображения.

Из материалов данной главы вы узнаете:

- ☐ Принципиальные основы создания и работы телеграмм ботов;
- ☐ как можно создать простейший телеграмм бот на Python;
- ☐ как можно интегрировать телеграмм бот с нейронными сетями;
- ☐ как в телеграмм ботах можно создать и использовать кнопки для взаимодействия с пользователями;
- ☐ как создать бота художника для стилизации изображений с использованием синхронного режима работы программных модулей;
- ☐ как создать бота художника для стилизации изображений с использованием асинхронного режима работы программных модулей.

Примечание.

Программные модули, приведенные в данном разделе, достаточно требовательны к ресурсам компьютера, особенно к потребностям в памяти. Это связано с тем, что телеграмм боты в фоновом режиме используют библиотеку машинного обучения PyTorch и модель нейронной сети 21styles.model. Примеры приведенных программ достаточно быстро работают на ПК с объемом оперативной памяти 12-16 Гб. Соответственно аналогичные ресурсы нужно иметь на публичных серверах при развертывании на них телеграмм ботов. Так как публичный сервер с интерактивными цифровыми книгами имеет оперативную память всего 1 Гб, то приведенные в данной главе телеграмм боты не развернуты на публичном сервере, однако вы можете опробовать их работу на своих локальных компьютерах, если они имеют достаточный размер оперативной памяти.

3.1. Создание Telegram бота

Сначала давайте разберемся, как работает телеграмм бот. Схематично принцип работы телеграмм бота представлен на рис. 3.1.

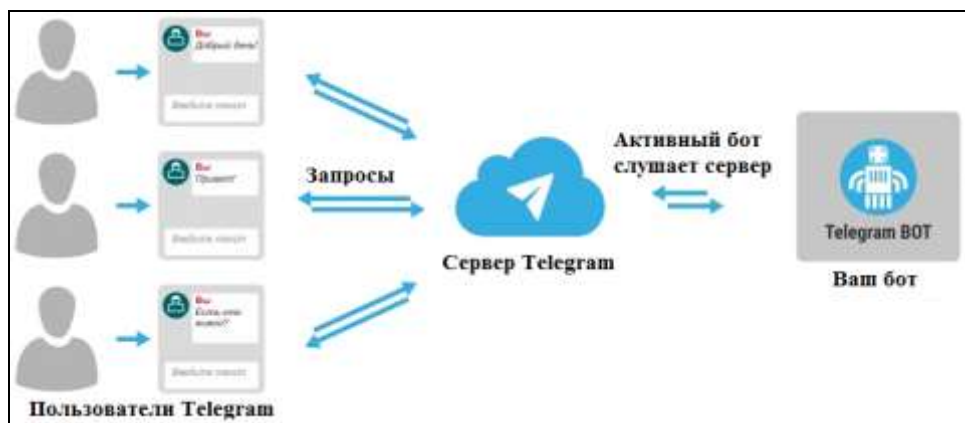


Рис. 3.1. Принцип работы телеграмм бота

Главным элементом в схеме работы ботов является сервер Telegram. Когда вы создали и активировали своего телеграмм бота, он через сеть интернет соединяется с сервером Telegram и слушает его сообщения. Если активных сообщений нет, то бот просто находится на связи с сервером. Пользователи Telegram имеют на своих смартфонах мобильное приложение, или имеют аналогичное приложение на своем компьютере или ноутбуке. Это приложение также находится в постоянной связи с сервером Telegram.

Пользователи находят вашего бота, и пишет ему сообщение. Все сообщения пользователей сначала попадают на сервер Telegram. Если ваша программа-бот запущен, то он будет получать эти сообщения и отвечать на них.

Первый шаг, который необходимо сделать, это создать (зарегистрировать) своего бота на сервере Telegram, для этого существует отец все ботов (BotFather). На смартфоне необходимо открыть приложение Telegram и через поиск найти бот с именем «BotFather» (рис. 3.2).



Рис. 3.2. Активация бота с именем BotFather

Вам будет найдено много ботов с таким названием, но необходимо будет выбрать тот, который отмечен синей галочкой. После этого BotFather покажет все свои команды, которыми впоследствии можно будет управлять своим ботом. Для создания нового бота, нужно набрать команду /newbot. Вас попросят дать название своему боту (рис. 3.3).



Рис. 3.3. Приглашение бота BotFather к вводу имени вашего бота

Здесь нужно ввести любое имя для вашего бота. Это то имя, которое будут видеть другие пользователи Telegram. После ввода имени нужно придумать и ввести уникальный идентификатор для бота (username). Этот идентификатор должен быть на английском языке и всегда заканчиваться на слово Bot (в этом слове не должно быть пробелов). Нужно иметь в виду, что наиболее популярные идентификаторы для ботов уже зарегистрированы другими пользователями. Сервер Telegram проверит уникальность введенного идентификатора. Если бот с заданным вами идентификатором уже существует, то BotFather предложит изменить

его, если бота с таким идентификатором нет, то процесс регистрации вашего бота будет завершен. Предположим, что боту был задан идентификатор `it_is_my_telegr_Bot`. После ввода этого имени от сервера будет получен следующий ответ (рис. 3.4).

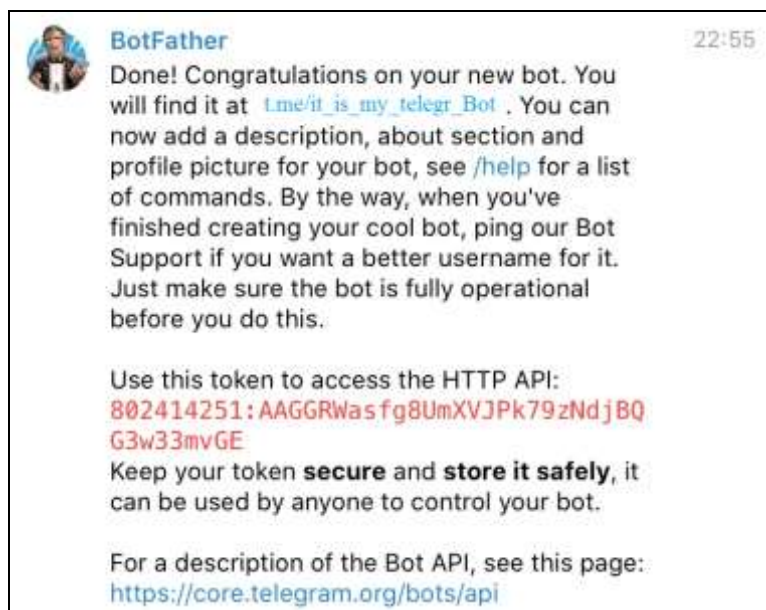


Рис. 3.4. Завершение процесса регистрации вашего бота

В самом начале данного сообщения синим цветом выделена ссылка на ваш бот, начинающаяся на «t.me/». Нажав на эту ссылку, вы перейдете к вашему боту. Эту ссылку можно отправлять другим пользователям для того, чтобы они могли воспользоваться услугами вашего бота. На данном рисунке красным цветом показан токен (token) вашего бота.

Примечание.

Токен бота в Telegram — это уникальный идентификатор, по которому бот идентифицируется в мессенджере.

Этот токен является, как бы паролем, который обеспечивает обращение к вашему боту. Его нужно сохранить и никому не сообщать, он будет использован при написании программного кода вашего бота.

Токен представляет собой набор чисел и литералов латинского алфавита в верхнем и нижнем регистре. Он выполняет следующие функции:

- ☐ позволяет серверу идентифицировать запросы пользователей;
- ☐ шифрует доступ к каналу боту, чтобы мошенники не получили к нему доступ.

Токен нужно хранить в защищенном месте и никому не сообщать, иначе к боту могут подключиться злоумышленники и использовать его в своих целях.

Итак, вы зарегистрировали официальный бот, который пока ничего не умеет. Теперь нужно научить его принимать запросы пользователей, формировать и отправлять им соответствующие ответы. Сначала мы напишем простейший бот на Python, потом научим его стилизовать изображения с использованием нейронных сетей.

3.2. Простейший Telegram бот «Попугай»

В данном разделе мы создадим простейший бот «Попугай», который будет просто повторять все запросы пользователя. Для начала нам нужно в приложение установить API для взаимодействия с сервером Telegram:

```
pip install pyTelegramBotAPI
```

Напишем код программного модуля `simple_bot`, в котором будет реализован очень простой телеграмм бот (листинг 3.1).

Листинг 3.1. Модуль `simple_bot.py` – простой телеграмм бот

Примечание.

В демо-версии книги удалены листинги программ.

В данном примере сначала выполняется импорт пакета `telebot` (это по сути API для взаимодействия вашего бота с сервером Telegram). Затем создана текстовая переменная `TOKEN`, которой присвоено значение токена вашего бота, и создается ваш бот (объект `bot`).

Примечание.

В этом примере значение токена представляет набор случайных символов, вам здесь необходимо использовать то значение токена, которое было получено от бота BotFather.

Далее определены две функции пользователя, которые «обернуты» в функцию - декоратор `@bot.message_handler()`.

Примечание.

В Python символ "@" используется для обозначения декораторов. Декораторы — это функции, которые модифицируют или оборачивают другие функции или методы. Они позволяют добавлять функциональность к готовым функциям или методам без изменения их исходного кода. В данном примере `@bot.message_handler(commands=['help', 'start'])` — это декоратор, который регистрирует функцию `url` в качестве обработчика сообщений, связанных с командами `'help'` и `'start'`. Декоратор `@bot.message_handler` принимает аргументы (в данном случае `commands=['help', 'start']`) и применяет их к функции `url`, которая имеется в загруженном пакете `pyTelegramBotAPI`.

Первая функция является обработчиком запросов пользователя `'/start'` или `'/help'`.

```
# Обработать запрос '/start' или '/help'
@bot.message_handler(commands=['help', 'start'])
def send_welcome(message):
    text_answer = 'Привет, я бот-попугай! \nНапишите что-нибудь, а я повторю!'
    bot.reply_to(message, text_answer)
```

Если пользователь в мобильном приложении Telegram наберет команду «`/start`» или «`/help`», то бот отправит ему в ответ содержимое текстовой переменной «`text_answer`». А если пользователь наберет любой другой текст, то бот «Попугай» вернет этот текст пользователю. Теперь можно запустить этот программный код на вашем компьютере, а на смартфоне проверить его работу, результаты представлены на рис. 3.5.

Примечание.

Ваш компьютер, на котором запущена программа с телеграмм ботом, должен быть подключен к сети интернет.

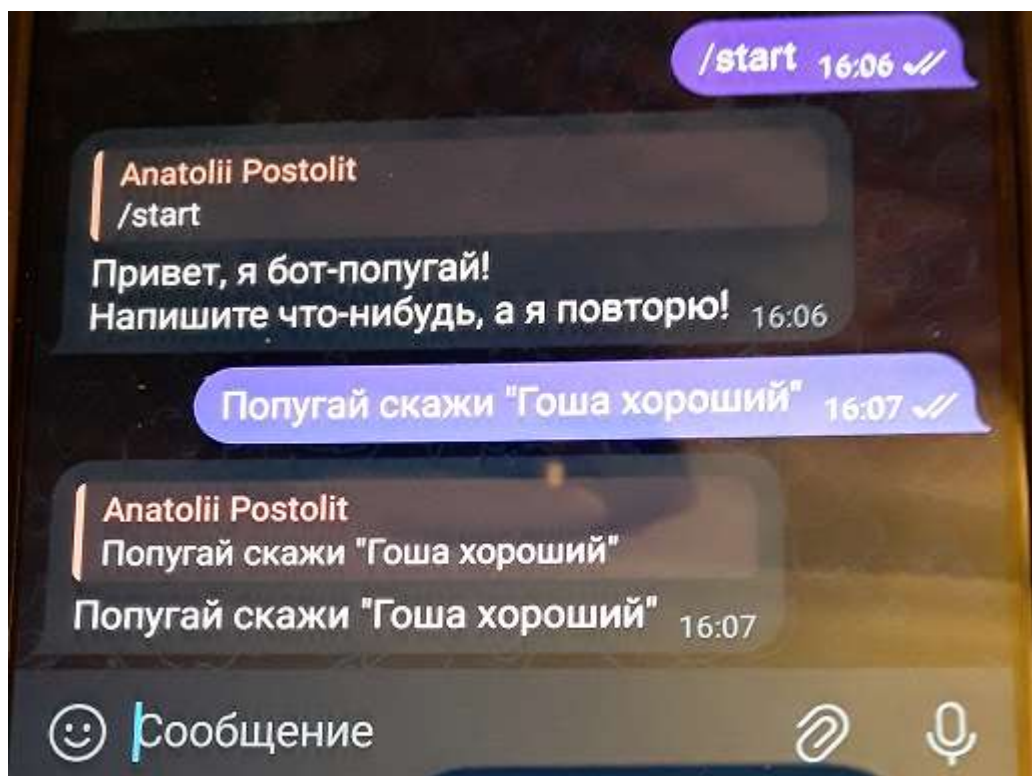


Рис. 3.5. Пример диалога пользователя с ботом «Попугай».

Как видно из данного рисунка, на команду `start` бот прислал ответ: «Привет, я бот-попугай! Напишите что-нибудь, а я повторю». А при отправке другого текста, он был возвращен пользователю. Пока наш бот работает на локальном компьютере разработчика, который имеет выход в сеть интернет. Пока приложение `simple_bot` активно, бот «Попугай» будет отвечать на запросы пользователя. Как только вы остановите это приложение или выключите свой компьютер, связь с сервером Telegram оборвется и бот перестанет принимать запросы. Для того чтобы телеграмм бот был постоянно активен, его нужно развернуть и запустить на публичном сервере.

Итак, в качестве примера мы создали простейший бот, но с точки зрения пользователя от него нет никакой пользы. А теперь давайте создадим бота художника, который, используя нейронную сеть, будет стилизовать любые фотографии, присланные пользователем.

3.3. Интеграция телеграмм бота с нейронными сетями

В данном разделе реализуем бот, который по запросу пользователя выполняет стилизацию фотографии под карандашный рисунок. Для этого будем использовать нейронную сеть (`21styles.model`), изображение с базовым стилем (`pencil.jpg`), подключать модуль `M_S_21.py`,

который объединен с модулем net, которые описаны в главе 2. Код M_S_21.py с включенным в него модулем net приведен в листинге 3.2.

Листинг 3.2. . Код модуля M_S_21.py с включенным в него модулем net

Примечание.

Так как код модуля M_S_21.py, используемого совместно с телеграмм ботами достаточно длинный, то его листинг вынесен в приложение 2.

Программный код модуля Bot_styl_foto.py, в котором реализован очень простой телеграмм бот – художник, приведен в листинге 3.3.

Листинг 3.3. Модуль Bot_styl_foto.py – телеграмм бот - художник

Примечание.

В демо-версии книги удалены листинги программ.

В начале этого программного модуля выполняется импорт необходимых библиотек (telebot, os), программного модуля (M_S_21). После этого создана текстовая переменная token, которой присвоено значение токена вашего бота, и создается ваш бот (объект bot).

Примечание.

В этом примере значение токена представляет набор случайных символов, вам здесь необходимо использовать то значение токена, которое было получено от бота BotFather.

На следующем шаге создана функция, которая принимает запросы пользователя (get_tetx_messages):

```
# Обработка запросов пользователя
@bot.message_handler(content_types=['text'])
def get_text_messages(message):
```

Пользовательская функция get_tetx_messages(), обернута в функцию – декоратор подключенного API. Она в параметре message принимает потупивший от пользователя запрос, и в зависимости от содержания данного запроса, с использованием метода bot.send_message отправляет ему один из ответов:

- ❑ «Привет, отправте фото» – если получен запрос «Привет»;
- ❑ «Напиши Привет» - если получен запрос «/help»;
- ❑ «Вопрос не понят. Напиши /help» - если от пользователя получен любой другой запрос.

Если пользователь написал «Привет», то бот отправит ему ответ - «Привет, отправте фото». Далее пользователь должен отправить боту ответное сообщение, к которому нужно «пристегнуть» любое фото из своего смартфона.

Полученное от пользователя фото будет обработано функцией def photo(message):

```
# Получить фото от пользователя и обработать
@bot.message_handler(content_types=['photo'])
def photo(message):
```

Пользовательская функция `def photo()`, обернута в функцию – декоратор подключенного API. Она в параметре `message` принимает потупившую от пользователя фотографию, и приступает к ее обработке. В данной функции сначала фото, принятое от пользователя, сохраняется на диске:

```
fileID = message.photo[-1].file_id
# print('fileID =', fileID)
file_info = bot.get_file(fileID)
# print('file.file_path =', file_info.file_path)
# Загрузка файла с фото
downloaded_file = bot.download_file(file_info.file_path)
# Сгенерированное имя полученного файла
file_name = file_info.file_path

# Сохранить файл с именем file_name
with open(file_name, 'wb') as new_file:
    new_file.write(downloaded_file)
```

Чтобы посмотреть, где бот сохраняет принятые фото, можно снять комментарий со строк с инструкцией `print`. Бот создаст директорию с именем `photos`, где будет сохранять принятые от пользователя фотографии, добавляя к имени файла очередной номер (например, `photos/file_7.jpg`). Если фото успешно принято и сохранено, то пользователю будет отправлено сообщение:

```
# Отправить ответ пользователю
bot.reply_to(message, "Фото получено, запущен процесс обработки...")
```

На следующем шаге формируются аргументы, которые будут переданы в модуль стилизации изображения:

```
# Обращение к классу стилизации изображения
in_image = file_name # исходное изображение
in_styl = './images/21styles/pencil.jpg' # стиль
styl_model = './model_21/21styles.model' # модель
size = 1024 # размер
# Запуск процесса стилизации фото
M_S_21.styl_21(in_image, in_styl, styl_model, size)
```

В этом программном коде определяются следующие переменные:

- ❑ `in_image` – путь к файлу с полученным изображением (фото присланное пользователем);
- ❑ `in_styl` – путь к имени файла, в котором находится изображение базового стиля;
- ❑ `styl_model` – путь к модели нейронной сети;
- ❑ `size` – размер стилизованного изображения.

Примечание.

Обратите внимание на пути к тем или иным файлам. В вашем проекте эти файлы должны располагаться именно в указанных директориях, а синтаксис инструкций должен соответствовать используемой операционной системе (Windows или Linux).

После того, как значение аргументов заданы, делается обращение к классу `M_S_21.styl_21`, где и выполняется процесс стилизации полученной фотографии. Если заглянуть код программного модуля `M_S_21.py`, то можно увидеть, что обработанное нейронной сетью изображение сохраняется на диске под именем `'./images/styl21_out/Out_Img.jpg'`.

После завершения процесса стилизации изображения пользователю будет отправлено сообщение:

```
# Отправка ответа пользователю
bot.reply_to(message, "Фото обработано, получите ответ")
```

Затем производится чтение с диска стилизованного изображения и отправка его пользователю:

```
# Формирование имени стилизованного фото
out_file = './images/styl21_out/Out_Img.jpg'
# Чтение стилизованного фото
with open(out_file, "rb") as file:
    styl_foto = file.read()
# Отправка стилизованного фото пользователю
bot.send_photo(message.from_user.id, photo=styl_foto)
```

Для того чтобы не «засорят» диск лишними данными, полученную от пользователя фотографию можно удалить:

```
# Удалит с диска принятое фото
os.remove(file_name)
```

Работу этого модуля завершает инструкция:

```
# Инициализация работы бота
bot.polling(none_stop=True, interval=0)
```

Эта инструкция обязательно должна быть последней. Она, по сути, устанавливает связь с сервером, и запускает бесконечный цикл прослушивания поступающих запросов. Любая команда после данной инструкции просто никогда не будет выполнена.

Теперь можно запустить данный программный модуль на выполнение и протестировать работу модуля со своего смартфона. После отправки боту запроса «Привет», пользователь получит сообщение:

```
«Привет, отправте фото».
```

Далее пользователь должен выбрать на смартфоне одну из фотографий и опрavitь ее боту. После этого начнется обработка полученного изображения, которая длится не более 5-10 секунд. По завершению этого процесса пользователю вернется стилизованная фотография (рис. 3.6).

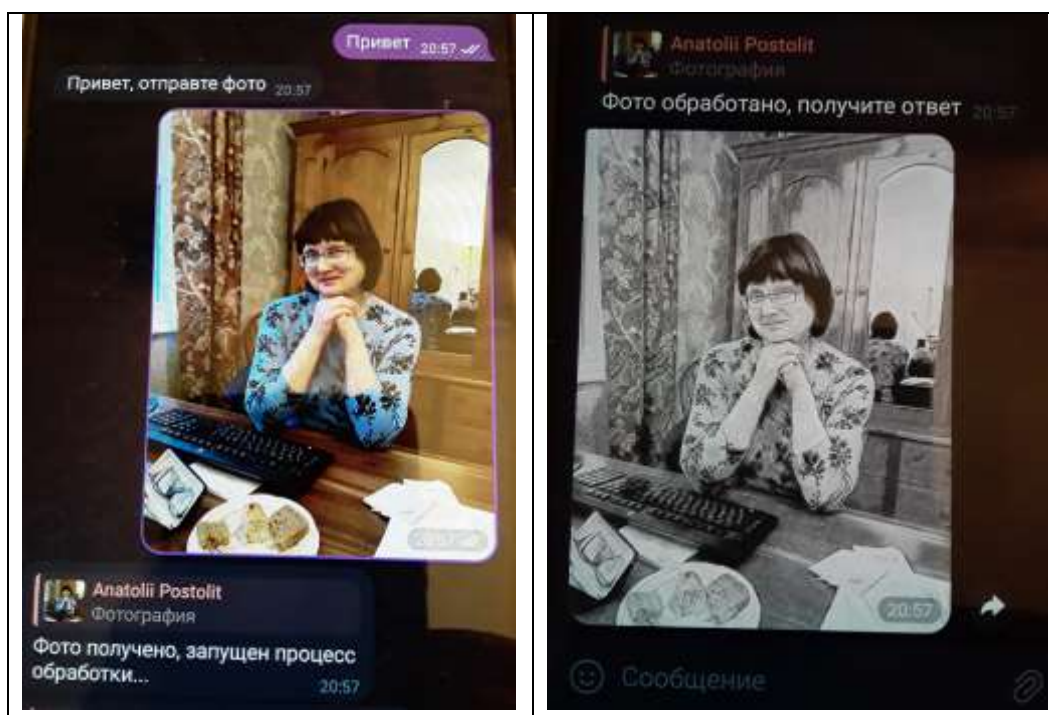


Рис. 3.6. Диалог пользователя с ботом – художником на смартфоне (исходное и обработанное фото)

В данном примере присланная фотография обработана стилем «Карандашный рисунок».

Общаться с ботом можно не только с помощью смартфона, это можно сделать с использованием ноутбука или стационарного компьютера.

Чтобы открыть на компьютере веб версию Telegram нужно выполнить следующие действия:

- ☐ перейдите по ссылке <https://web.telegram.org/k/>;
- ☐ в приложении смартфона Telegram активируйте меню (нажмите на три полоски в левой верхней части экрана);
- ☐ в меню выберите пункты «настройки» > «устройства» > «сканировать QR-код» или «Подключить устройство»;
- ☐ с использованием смартфона отсканируйте QR-код на экране компьютера, после чего web версия приложения Telegram откроется.

Теперь и на компьютере можно вести диалог с ботом – художником (рис. 3.7.)

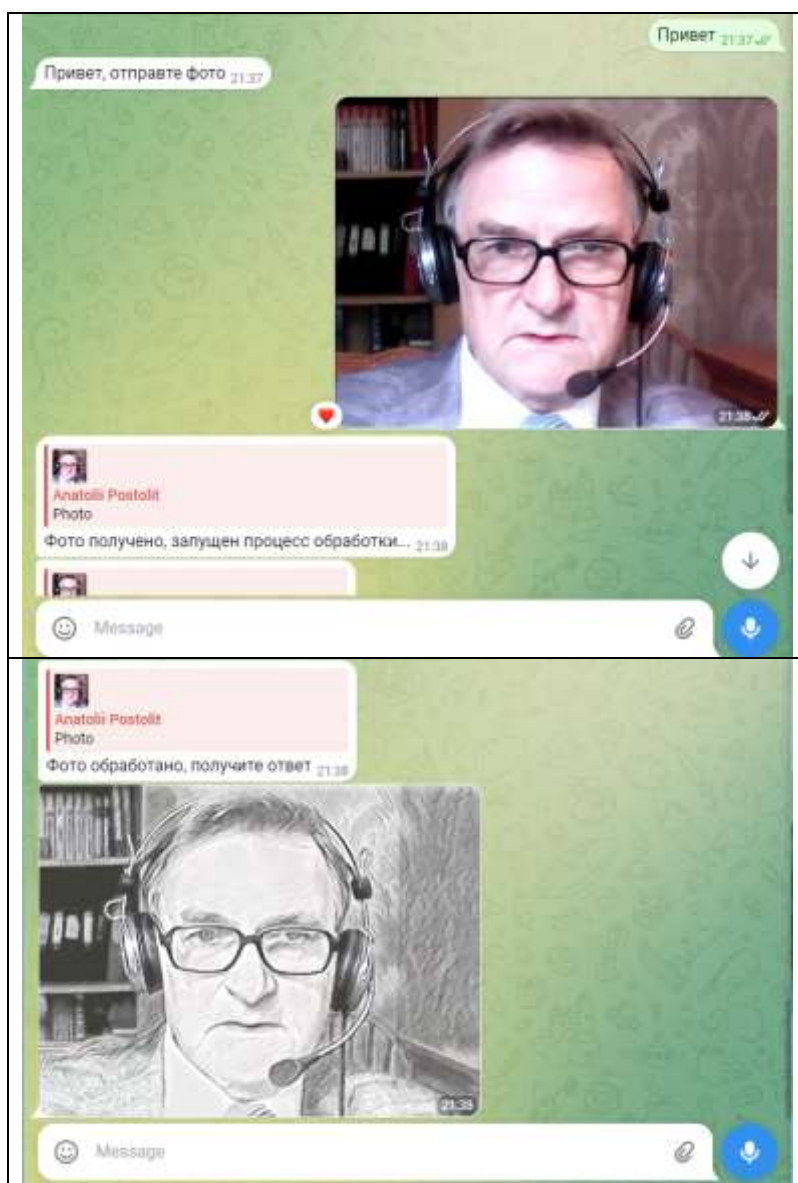


Рис. 3.7. Диалог пользователя с ботом – художником на компьютере (исходное и обработанное фото)

Как видно из данного рисунка, присланная боту фотография с использованием нейронной сети была стилизована под карандашный рисунок, и обработанный рисунок был возвращен пользователю.

3.4. Использование кнопок в Telegram ботах

Пользователям Telegram не совсем удобно набирать команды на клавиатуре, тем более что их текст может быть достаточно длинным, или не все команды известны их список нужно

запрашивать по команде /help. Процесс выбора команд можно значительно упростить, если связать их с кнопками. Давайте рассмотрим простейший пример создания кнопок в ботах. Код этого примера (модуль bot_buton.py) приведен в листинге 3.4.

Листинг 3.4. Модуль bot_buton.py – телеграмм бот - художник

Примечание.

В демо-версии книги удалены листинги программ.

В начале этого программного модуля выполняется импорт необходимых библиотек (telebot, types). После этого создана текстовая переменная token, которой присвоено значение токена вашего бота, и создается ваш бот (объект bot).

Примечание.

В этом примере значение токена представляет набор случайных символов, вам здесь необходимо использовать то значение токена, которое было получено от бота BotFather.

На следующем шаге создана функция, которая принимает запросы пользователя (start):

```
@bot.message_handler(commands=['start'])
def start(message):
    # Создать кнопки главного меню
    markup = types.ReplyKeyboardMarkup(resize_keyboard=True)
    btn1 = types.KeyboardButton("Задать вопрос")
    btn2 = types.KeyboardButton("Выбрать язык")
    markup.add(btn1, btn2)
    # Отправить пользователю сообщение
    ms_text = 'Привет, я бот! Выбери действия на кнопках'
    bot.send_message(message.chat.id, text=ms_text, reply_markup=markup)
```

Пользовательская функция start(), обернута в функцию – декоратор подключенного API. Она в параметре message принимает потупившую от пользователя команду, и создает две кнопки со следующим текстом:

- ❑ btn1 - Задать вопрос;
- ❑ btn2 - Выбрать язык.

После этого отправляет пользователю сообщение следующего содержания – «Привет, я бот! Выбери действия на кнопках».

Далее следует функция ling(), которая принимает и обрабатывает запросы пользователя. В данной функции с использованием инструкции if реализовано несколько разветвлений.

Если пользователь в главном меню нажал на кнопку «Задать вопрос», то бот сгенерирует текстовое сообщение от пользователя «Задать вопрос», которое будет обработано в инструкции:

```
if message.text == "Задать вопрос":
    ms_text = "Жду ваш вопрос!"
    bot.send_message(message.chat.id, ms_text)
```

В этом случае пользователь получит от бота ответ следующего содержания «Жду ваш вопрос!».

Если пользователь в главном меню нажал на кнопку «Выбрать язык», то бот сгенерирует текстовое сообщение от пользователя «Выбрать язык», которое будет обработано в инструкции:

```
elif message.text == "Выбрать язык":
    markup = types.ReplyKeyboardMarkup(resize_keyboard=True)
    btn1 = types.KeyboardButton("RU Русский")
    btn2 = types.KeyboardButton('GB English')
    back = types.KeyboardButton("Возврат в меню")
    markup.add(btn1, btn2, back)
    ms_text = "RU Выберите язык / GB Choose your language"
    bot.send_message(message.from_user.id, ms_text, reply_markup=markup)
```

В этом случае для выбора языка будут созданы три новые кнопки с надписями: «Русский», «English», «Возврат в меню», а пользователю будет отправлено сообщение «Выберите язык / Choose your language».

Примечание.

Обратите внимание на символы RU и GB – это эмодзи, которые обозначают флаги стран. Вместо этих символов на экране смартфона будет показан российский и английский флаг. Более подробные сведения об эмодзи в телеграмм ботах вы можете узнать в сети интернет или в специальной документации.

Что в этом случае произойдет с экраном смартфона: две кнопки главного меню исчезнут, а вместо них появятся три кнопки дочернего меню.

Если пользователь в дочернем меню нажал на кнопку «Русский», то бот сгенерирует текстовое сообщение от пользователя «Русский», которое будет обработано в инструкции:

```
elif message.text == "RU Русский":
    ms_text = "Выбран язык RU Русский"
    bot.send_message(message.chat.id, text=ms_text)
```

В этом случае пользователю будет отправлено сообщение «Выбран язык Русский»

Если пользователь в дочернем меню нажал на кнопку «English», то бот сгенерирует текстовое сообщение от пользователя «English», которое будет обработано в инструкции:

```
elif message.text == "GB English":
    ms_text = "Выбран язык GB English"
    bot.send_message(message.chat.id, text=ms_text)
```

В этом случае пользователю будет отправлено сообщение «Выбран язык English».

Если пользователь в дочернем меню нажал на кнопку «Возврат в меню», то бот сгенерирует текстовое сообщение от пользователя «Возврат в меню», которое будет обработано в инструкции:

```
elif message.text == "Возврат в меню":
    # Создать кнопки главного меню
    markup = types.ReplyKeyboardMarkup(resize_keyboard=True)
    button1 = types.KeyboardButton("Задать вопрос")
    button2 = types.KeyboardButton("Выбрать язык")
```

```
markup.add(button1, button2)
ms_text = "Вы вернулись в главное меню"
bot.send_message(message.chat.id, text=ms_text, reply_markup=markup)
```

В этом случае три кнопки дочернего меню исчезнут, а вместо них будут восстановлены две кнопки главного меню: с надписями: «Задать вопрос», «Выбрать язык», а пользователю будет отправлено сообщение «Вы вернулись в главное меню».

Наконец, завершающая ветка разветвления if обрабатывает любые другие команды пользователя, которые относятся к неизвестным командам:

```
else:
    ms_text = "Неизвестная команда? Набери /start"
    bot.send_message(message.chat.id, text=ms_text)
```

В этом случае пользователю будет отправлено сообщение «Неизвестная команда? Набери /start». Конструкция if среди прочих разветвлений должна обязательно иметь такую ветку. Она обработает любой произвольный (даже заведомо не верный) запрос пользователя и подскажет ему, как нужно действовать, чтобы активировать работу бота.

Итак, мы сделали полный разбор фрагментов этого программного кода для того, чтобы было понятно, как в Telegram ботах можно создавать кнопки и обрабатывать события их нажатия. Теперь можно проверить работу данного бота, для чего запустим этот программный модуль на выполнение.

Примечание.

Программный модуль bot_buton.py можно запустить на локальном компьютере, который имеет выход в интернет. При этом ваш бот будет доступен удаленным пользователем до тех пор, пока запущенное приложение находится в работе, и станет не активным после завершения работы модуля или после выключения компьютера. Для того, чтоб бот всегда был в состоянии активности его нужно разместить на публичном сервере.

Результаты работы программного модуля представлены на рис. 3.8.

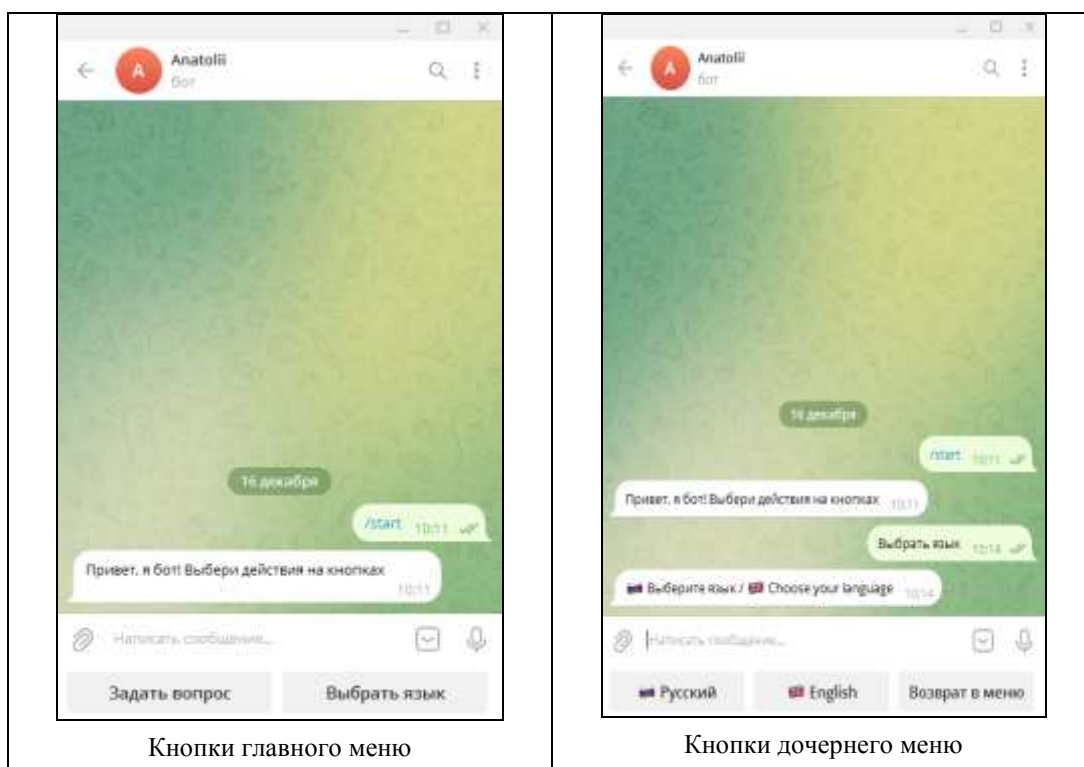


Рис. 3.8. Кнопки главного и дочернего меню приложения bot_buton.py

Как видно из данного рисунка, после запроса пользователя /start появились кнопки главного меню. После нажатия на кнопку «Выбрать язык» появились кнопки дочернего меню. После нажатия на кнопки «Русский» и «English» пользователь получит сообщение о том, какой язык он выбрал (рис. 3.9).

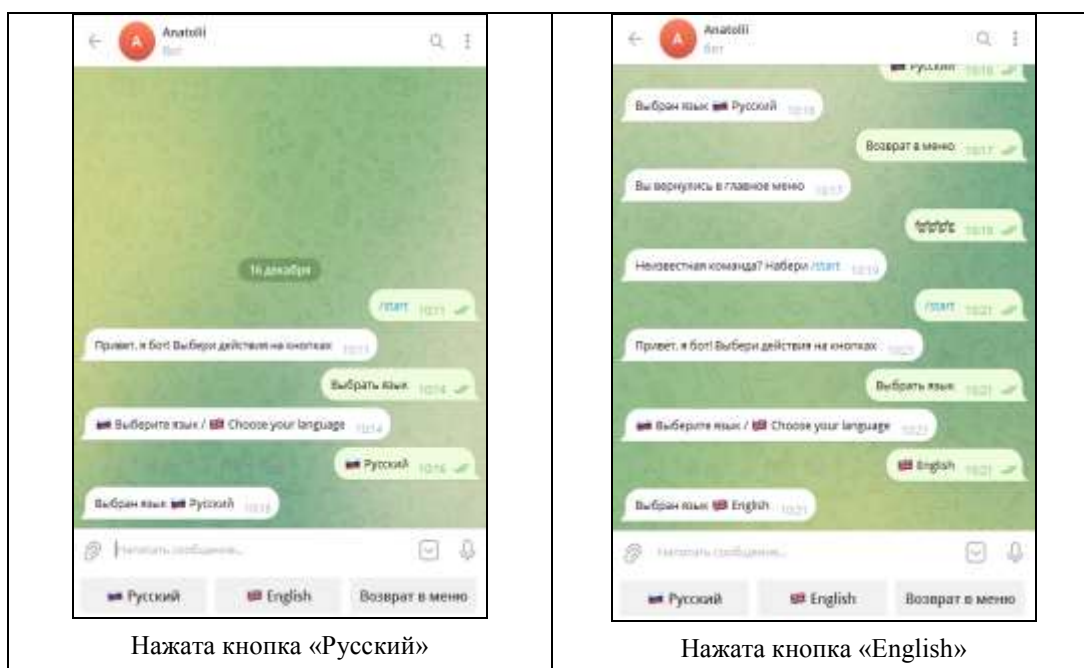


Рис. 3.9. Результаты нажатия на кнопки с выбором языка

Результаты нажатия на кнопку «Возврат в меню» и набора неизвестной команды представлены на рис. 3.10.

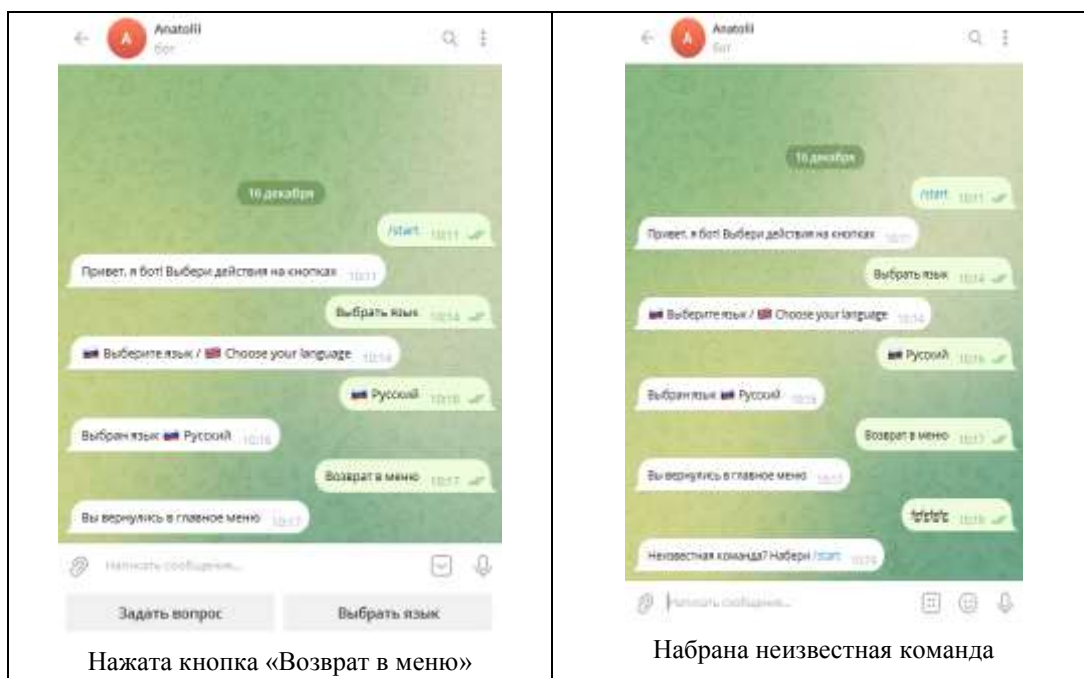


Рис. 3.10. Результаты нажатия на кнопку «Возврат в меню» и набора неизвестной команды

Итак, мы познакомились с тем, как можно в Telegram ботах реализовать диалог с пользователем с использованием кнопок. Теперь модифицируем программу «Бот – художник», где с использованием кнопок пользователь может выбрать предлагаемый стиль для трансформации фото в стилизованное изображение.

3.5. Бот – художник для стилизации фотографий разными стилями (синхронный режим)

Теперь мы знаем, как создаются кнопки для ботов, то есть теперь можно создать бот художника, который по запросу пользователя выполнит стилизацию фотографии под рисунок выбранного стиля. Для этого будем использовать нейронную сеть (21styles.model), изображение с базовым стилем (pencil.jpg), подключать модули M_S_21.py и net (см. листинг в приложении 2).

Создадим программный код (модуль bot_buton_1.py), в котором будет реализован телеграмм бот – художник, способный использовать 22 стиля для трансформации фотографий в стилизованные изображения (листинг 3.5).

Листинг 3.5. Модуль bot_buton_1.py – телеграмм бот - художник

Примечание.

Так как код модуля bot_buton_1.py достаточно длинный, то его листинг вынесен в приложение 3.

В папке, где находится данный модуль, создайте директорию с именем photo, она по умолчанию будет использоваться ботом для хранения промежуточных файлов с фотографиями.

Это достаточно длинный программный код (более 200 строк), но он имеет очень простую структуру и состоит из следующих блоков:

- ☐ формирование кнопок главного меню;
- ☐ формирования кнопок выбора стиля;
- ☐ задание параметров стиля для стилизации фотографий;
- ☐ получение, обработка и отправка фотографии пользователю.

Для стилизации фотографий будет подключена нейронная сеть и образцы изображений с базовыми стилями.

В начале этого программного модуля выполняется импорт необходимых библиотек (telebot, types, os), программного модуля (M_S_21). После этого создана текстовая переменная token, которой присвоено значение токена вашего бота, и создается ваш бот (объект bot).

Примечание.

В этом примере значение токена представляет набор случайных символов, вам здесь необходимо использовать то значение токена, которое было получено от бота BotFather.

На следующем шаге задан стиль по умолчанию (in_style) и создана функция, которая принимает команду пользователя (start):

```
# Стиль по умолчанию
in_style = './images/21styles/pencil.jpg'
text_style = 'Карандаш'
```

```
@bot.message_handler(commands=['start'])
def start(message):
    # Создать кнопки
    markup = types.ReplyKeyboardMarkup(resize_keyboard=True)
    btn1 = types.KeyboardButton("Загрузить фото")
    btn2 = types.KeyboardButton("Сменить стиль")
    markup.add(btn1, btn2)
    # Отправить пользователю сообщение
    ms_text = 'Привет! Я бот художник. Выбери действия на кнопках. \nЗадан
    стиль-' + text_style
    bot.send_message(message.chat.id, text=ms_text, reply_markup=markup)
```

Пользовательская функция start(), обернута в функцию – декоратор подключенного API. Она в параметре message принимает потупившую от пользователя команду, создает кнопки главного меню, и, с использованием метода bot.send_message, отправляет ему сообщение «Привет! Я бот художник. Выбери действия на кнопках» с указанием стиля по умолчанию. Пользователь имеет возможность либо сразу загрузить фото и обрабатывать его стилем по умолчанию, либо выбрать другой стиль.

Если пользователь нажимает на кнопку «Сменить стиль», то начинает работать функция func(), в частности ветка разветвления if – «Сменить стиль»:

```
@bot.message_handler(content_types=['text'])
def func(message):
    ...
    # Набор кнопок для смены стиля
    elif message.text == "Сменить стиль":
        markup = types.ReplyKeyboardMarkup(resize_keyboard=True)
        btn_candi = types.KeyboardButton("Кандинский 1")
        btn_compos = types.KeyboardButton("Кандинский 2")
        btn_escher = types.KeyboardButton("Мауриц Корнелис Эшер")
        btn_feathers = types.KeyboardButton("Перья")
        btn_frida_kahlo = types.KeyboardButton("Фрида Карло")
        btn_la_muse = types.KeyboardButton("Пикассо муза")
        btn_mos1 = types.KeyboardButton("Мозаика 1")
        btn_mos2 = types.KeyboardButton("Мозаика 2")
        btn_noise = types.KeyboardButton("Звездный шум")
        btn_pencil = types.KeyboardButton("Карандаш")
        btn_pic_auto = types.KeyboardButton("Пикассо автопортрет")
        btn_rain_princess = types.KeyboardButton("Принцесса дождя")
        btn_del_port = types.KeyboardButton("Делоне портрет")
        btn_pic_kub = types.KeyboardButton("Пикассо кубизм")
        btn_ship = types.KeyboardButton("Айвазовский")
        btn_van_gog = types.KeyboardButton("Ван Гог Звездная ночь")
        btn_stars = types.KeyboardButton("Звезды")
        btn_strip = types.KeyboardButton("Штрихи")
        btn_scream = types.KeyboardButton("Эдвард Мунк Крик")
        btn_udnie = types.KeyboardButton("Франсис Пикабия")
        btn_wave = types.KeyboardButton("Хокусай Волны")
        btn_matisse = types.KeyboardButton("Матисс")
        back = types.KeyboardButton("Возврат в главное меню")

        markup.add(btn_candi, btn_compos, btn_escher, btn_feathers,
                    btn_frida_kahlo, btn_la_muse, btn_mos1, btn_mos2,
```

```

        btn_noise, btn_pencil, btn_pic_auto, btn_rain_princess,
        btn_del_port, btn_pic_kub, btn_ship, btn_van_gog, btn_stars,
        btn_strip, btn_scream, btn_udnie, btn_wave, btn_matisse, back)
    bot.send_message(message.chat.id, text="Выберите другой стиль", re-
ply_markup=markup)
...

```

В результате работы этого фрагмента кода кнопки главного меню будут заменены кнопками для выбора стиля.

После задания нового стиля пользователь может передать боту фотографию, и она будет обработана функцией pfoto().

```

# Получить фото от пользователя и обработать
@bot.message_handler(content_types=['photo'])
def photo(message):
...

```

Примечание.

Данная функция и модуль M_S_21 описаны в главе 2.

Если пользователь нажимает кнопку «Возврат в главное меню», то включается в работу следующий фрагмент кода функции func():

```

elif message.text == "Возврат в главное меню":
    markup = types.ReplyKeyboardMarkup(resize_keyboard=True)
    button1 = types.KeyboardButton("Загрузить фото")
    button2 = types.KeyboardButton("Сменить стиль")
    markup.add(button1, button2)
    ms_text = "Вы вернулись в главное меню"
    bot.send_message(message.chat.id, text=ms_text, reply_markup=markup)

```

После этого кнопки выбора стиля заменяются кнопками главного меню.

Если пользователь нажимает на кнопку «Загрузить фото», то включается в работу следующий фрагмент кода функции func():

```

if message.text == "Загрузить фото":
    # Ответ на запрос загрузить фото
    bot.send_message(message.chat.id, "Выберите и отправте фото")

```

В этом случае пользователю отправляется напоминание, что он может передать боту любую фотографию для обработки.

Если от пользователя поступает неизвестное боту сообщение, то включается в работу следующий фрагмент кода функции func():

```

else:
    ms_text = "Неизвестная команда? Набери /start"
    bot.send_message(message.chat.id, text=ms_text)

```

То есть пользователю передается сообщение, что для активации бота и его главного меню необходимо набрать команду /start.

Необходимо сделать важное замечание, в функции func() необходимо обязательно объявить глобальную переменную in_style:

```

def func(message):
    global in_style

```

...

Если этого не сделать, то пользователь не сможет переопределить стиль, который задан по умолчанию, и любая фотография будет стилизована под карандашный рисунок.

Теперь можно активировать нашего бота – художника и проверить его работу.

Примечание.

Программный модуль bot_buton_1.py можно запустить на локальном компьютере, который имеет выход в интернет. При этом ваш бот будет доступен удаленным пользователем до тех пор, пока запущенное приложение находится в работе, и станет не активным после завершения работы модуля или после выключения компьютера. Для того, чтоб бот всегда был в состоянии активности его нужно разместить на публичном сервере.

Результаты работы программного модуля представлены на рис. 3.11.

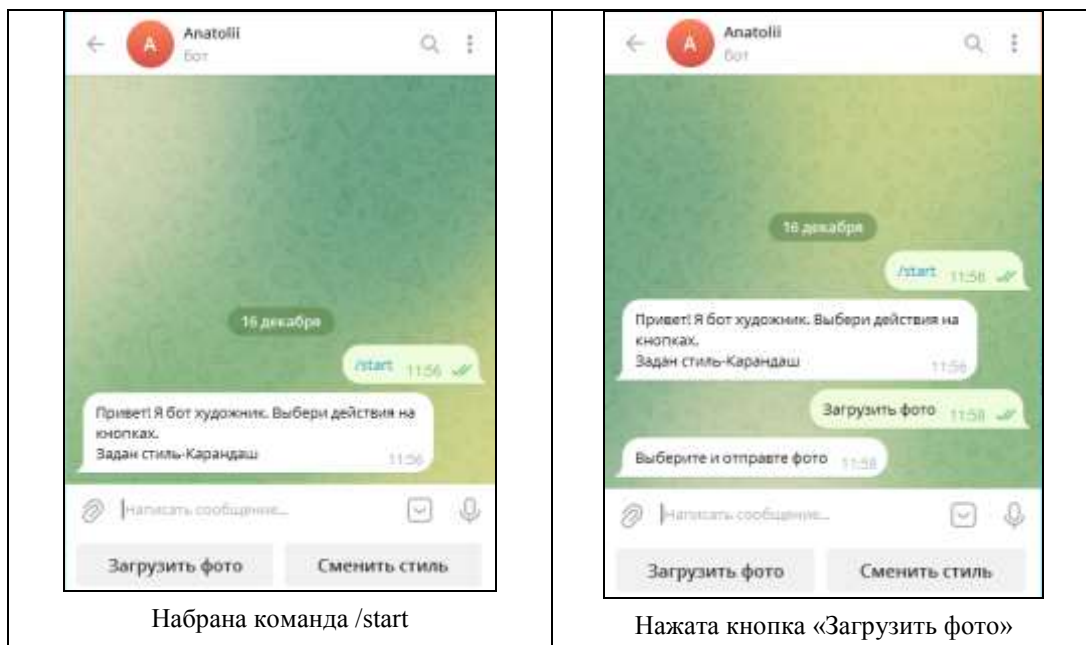


Рис. 3.11. Начальная стадия работы бота –художника

Как видно из данного рисунка, после набора команды /start появились две кнопки главного меню. После нажатия на кнопку «Загрузите фото» бот прислал ответ «Выберите и отправьте фото». После нажатия на изображение со скрепкой пользователь может выбрать любую фотографию на смартфоне или компьютере. Это фото появится в интерфейсе. Для отправки фото нужно нажать на кнопку «Отправить». После нажатия на кнопку «Отправить» фотография будет получена ботом, который приступит к ее обработке, а пользователю отправит сообщение «Фото получено, запущен процесс обработки» (рис. 3.12).

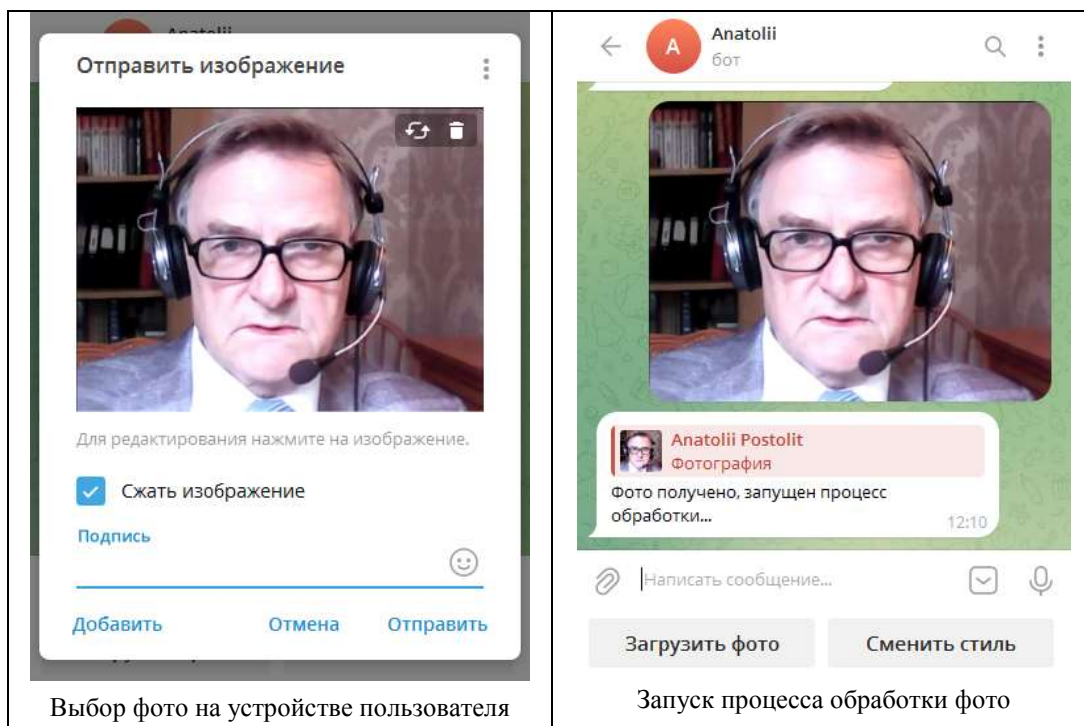


Рис. 3.12. Выбор фотографии и отправка ее боту для обработки

Когда процесс стилизации будет завершен, бот вернет пользователю стилизованное изображение с сообщением «Фото обработано, получите ответ» (рис. 3.13).

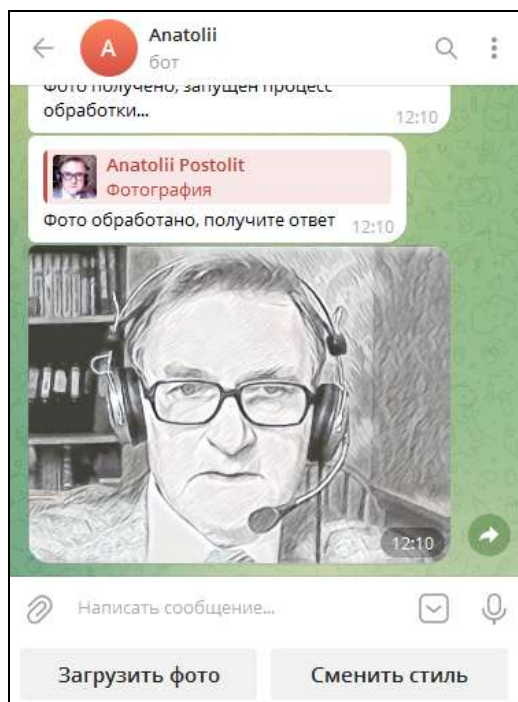


Рис. 3.13. Завершение процесса стилизации фотографии под карандашный рисунок

Как видно из данного рисунка, бот обработал фотографию стилем «карандашный рисунок». Если пользователь нажмет на кнопку сменить стиль, то появится набор кнопок для выбора другого стиля, а пользователю придет сообщение «Выберите другой стиль». Нажав на кнопку с интересующим стилем, пользователь сменит стиль, который был задан по умолчанию и получит сообщение с указанием нового стиля, например, «Выбран стиль Фрида Карло» (рис. 3.14).



Рис. 3.14. Выбор нового стиля для стилизации фотографии

Теперь пользователь может выбрать и передать боту любую фотографию для стилизации, она будет обработана с использованием нового стиля и возвращена пользователю (рис. 3.15).

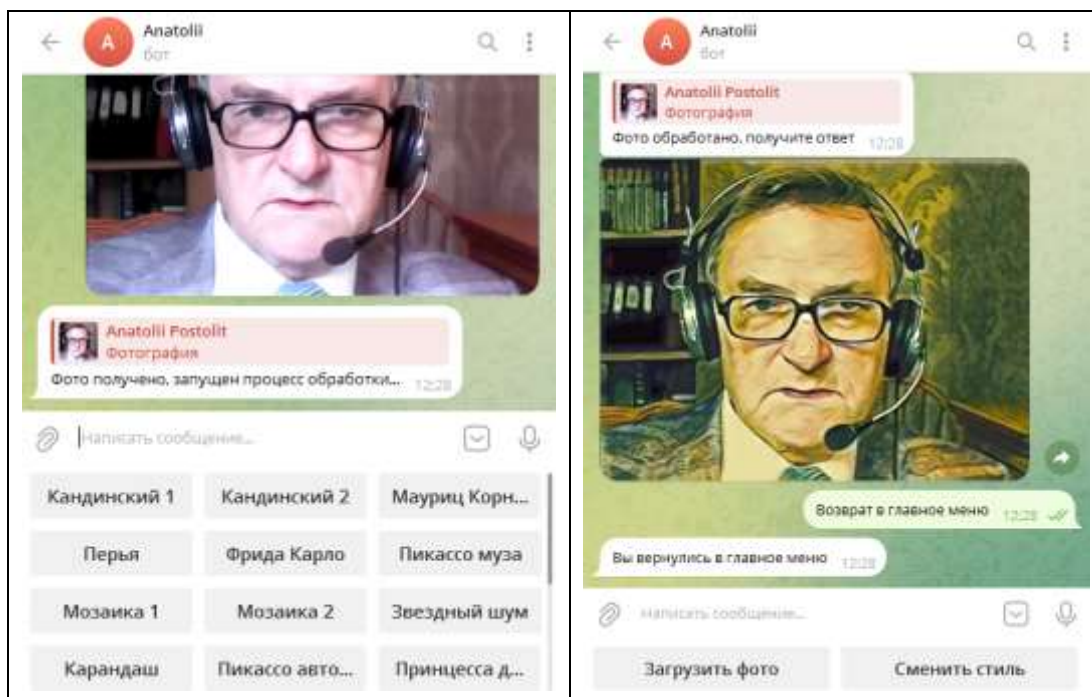


Рис. 3.15. Стилизации фотографии стилем Фрида Карло

Как видно из данного рисунка, пользователю вернулась фотография, обработанная стилем «Фрида Карло». На данном рисунке стилизованное фото показано после того, как в дочернем меню была нажата кнопка «Возврат в главное меню». После этого кнопки выбора стиля были заменены кнопками главного меню, а пользователь получил сообщение «Вы вернулись в главное меню».

Итак, мы разобрались с вопросом, как можно реализовать бота на Python с подключением к его работе нейронной сети. Пока наш бот обрабатывал изображение на локальном компьютере программными модулями, работающими в синхронном режиме. Теперь рассмотрим возможность обработки данных программными модулями, работающими в асинхронном режиме.

3.6. Бот – художник для стилизации фотографий (асинхронный режим)

В предыдущем разделе был создан бот, который работает в синхронном режиме. При небольшом количестве запросов пользователей он вполне справится со своей задачей. Но с увеличением количества пользователей будут создаваться задержки, что вызовет некоторые неудобства и даже зависание сервера. В этом случае на помощь придут методы асинхронного программирования.

Асинхронное программирование стало важным инструментом в арсенале разработчиков Python, особенно в контексте создания высокопроизводительных и масштабируемых приложений. Оно позволяет приложению выполнять другие задачи, пока одна из задач ожидает

завершения ввода-вывода или другого медленного процесса. Это особенно полезно для сетевых операций, взаимодействия с базами данных и других задач, которые могут занимать значительное время. В этом примере мы будем использовать библиотеку **asyncio** - это стандартная библиотека Python для написания асинхронного кода с использованием синтаксиса `async` и `await`. Она предоставляет базовые примитивы для работы с асинхронным программированием, такие как циклы событий, корутины, задачи и потоки.

Примечание.

Корутины (сопрограммы) — это блоки кода, которые работают асинхронно, то есть по очереди. В нужный момент исполнение такого блока приостанавливается с сохранением всех его свойств, чтобы запустился другой код. Когда управление возвращается к первому блоку, он продолжает работу. В результате программа выполняет несколько функций одновременно.

Кроме асинхронности в данном примере мы реализуем бот, который сможет общаться с пользователем на разных языках (на русском и на английском). Для работы бота в асинхронном режиме необходимо будет установить следующие библиотеки:

```
pip install pyTelegramBotAPI
pip install aiohttp

pip install pillow
pip install torch==1.8.1
pip install numpy
```

Первые две библиотеки необходимы для организации асинхронной работы, следующие три – для стилизации изображений с использованием нейронных сетей. Для стилизации изображений будем использовать нейронную сеть (`21styles.model`) и подключаемый модуль `M_S_21.py` (Приложение 2).

Это достаточно большой программный модуль (более 400 строк), поэтому мы не будем делать подробный обзор только некоторых его ключевых моментов. Этот модуль состоит из двух основных блоков: блок конфигурирования генеративной сети (`MSG-Net`), и класса стилизации изображения (`styl_21`).

В первом блоке реализован алгоритм работы нейронной сети с особой стратегией свертки (с повышающей дискретизацией), который позволяет избежать эффектов «шахматной доски», и повышает качество генерируемого изображения. Этот алгоритм использует возможности современных библиотек машинного обучения `Torch` и `PyTorch`.

Второй блок представляет собой класс (`styl_21`), который принимает исходное изображение, передает его в нейронную сеть для обработки, и возвращает стилизованное изображение. Здесь мы рассмотрим только фрагменты кода, связанные передачей этому классу необходимых параметров. Обратите внимание, что для задания путей к файлам используется библиотека `pathlib`, которая обеспечивает корректное формирование путей к папкам и файлам для различных операционных систем. Поэтому, если вы вели разработку бота на компьютере с ОС `Windows`, вы без проблем перенесете его на публичный сервер под ОС `Linux`. Итак, данному классу нужно передать следующие данные:

- ❑ `in_image` – путь к исходному изображению;
- ❑ `out_image` – путь к обработанному (стилизованному) изображению;
- ❑ `in_styl` – путь к изображению с базовым стилем;
- ❑ `styl_model` – модель нейронной сети;
- ❑ `size` – размер итогового изображения.

Теперь создадим программный код (модуль `asin_bot.py`), в котором будет реализован телеграмм бот – художник, способный стилизовать фотографии в асинхронном режиме с интерфейсом пользователя на двух языках (листинг 3.6).

Листинг 3.6. Модуль `asin_bot.py` – телеграмм бот – художник (асинхронный режим)

Примечание.

Так как код модуля `asin_bot.py` достаточно длинный, то его листинг вынесен в приложение 4.

В папке, где находится данный модуль, создайте директорию с именем `photo`, она по умолчанию будет использоваться ботом для хранения промежуточных файлов с фотографиями.

Это достаточно длинный программный код (более 500 строк), но он имеет достаточно простую структуру и состоит из следующих блоков:

- ❑ формирование кнопки `start` при любом (неизвестном) запросе пользователя;
- ❑ формирования базового меню на двух языках при нажатии на кнопку `start`;
- ❑ формирования меню на русском языке;
- ❑ формирования меню на английском языке;
- ❑ формирования меню для выбора языка;
- ❑ формирования меню выбора стиля;
- ❑ задание параметров стиля для стилизации фотографий на русском языке;
- ❑ задание параметров стиля для стилизации фотографий на английском языке;
- ❑ получение, обработка и отправка фотографии пользователю.

Для стилизации фотографий будет подключена нейронная сеть и образцы изображений с базовыми стилями.

В начале этого программного модуля выполняется импорт необходимых библиотек:

```
import asyncio – для асинхронного вызова функций;
from telebot.async_telebot import AsyncTeleBot, types – асинхронный бое;
import config – файл, содержащий токен;
import M_S_21 – модуль для стилизации изображений;
import os – модуль для формирования путей к файлам;
from pathlib import Path – модуль для формирования путей к файлам.
```

После этого создана текстовая переменная `token`, которой присвоено значение токена вашего бота, и создается ваш бот (объект `bot`).

Примечание.

В этом примере значение токена представляет набор случайных символов, вам здесь необходимо использовать то значение токена, которое было получено от бота `BotFather`.

На следующем шаге заданы параметры по умолчанию:

```
# Язык общения
language = 'ru'

# Путь к базовым изображениям стиля
path_styles = os.path.join(my_app, 'styles')
```

```

# Путь к папке со стилем по умолчанию
in_style = os.path.join(my_app, 'styles', 'pencil.jpg')
text_style = 'Карандаш'
text_style_ru = 'Карандаш'
text_style_en = 'Pencil'

# Путь к папке с моделью нейронной сети
style_model = os.path.join(my_app, 'net_model', '21styles.model')

# Путь к папке записи стилизованного фото
path_style_img = os.path.join(my_app, 'images', 'Out_Img.jpg')

# Размер стилизованного фото
size = 1024

```

Далее будем разбирать фрагменты программного кода и связанный с этим кодом пользовательский интерфейс.

Если пользователь оправит боту команду из любого набора символов, то это команда будет обработана следующим фрагментом программного кода:

```

else:
    # На любой вопрос - создать кнопку start
    markup = types.ReplyKeyboardMarkup(resize_keyboard=True)
    btn_start = types.KeyboardButton("/start")
    markup.add(btn_start)
    ms_text = "Для запуска бота нажми кнопку /start"
    await bot.send_message(message.chat.id, ms_text, reply_markup=markup)

```

В результате обработки неизвестной команды будет создана кнопка с надписью «/start», а пользователю отправлено сообщение «Для запуска бота нажми кнопку /start» (рис. 3.16).



Рис. 3.16. Ответ бота на неизвестную команду

Если пользователь нажмет на кнопку «/start» или отправит боту команду «/start», то начнет работу фрагмент программного кода, обрабатывающего данную команду:

```
# Обработка команд '/start' и '/help'
@bot.message_handler(commands=['help', 'start'])
async def send_welcome(message):
    # global in_style, text_style_ru, text_style_en
    # Создать кнопки
    markup = types.ReplyKeyboardMarkup(resize_keyboard=True)
    btn1 = types.KeyboardButton("Сменить стиль / Change the style")
    btn2 = types.KeyboardButton("Сменить язык / Change the language")
    markup.add(btn1, btn2)

    # Отправить сообщение
    ms1 = 'Привет, я бот художник!'
    ms2 = '\nПо умолчанию заданы настройки:'
    ms3 = '\n    Стиль - ' + text_style
    ms4 = '\n    Язык - ' + language
    ms5 = '\nЗагрузите фото, или смените настройки'
    ms6 = '\nUpload a photo, or change the settings'
    ms_text = ms1 + ms2 + ms3 + ms4 + ms5 + ms6
    await bot.send_message(message.chat.id, text=ms_text, reply_markup=markup)
```

В этом случае пользователю будет показаны настройки по умолчанию и меню на двух языках с предложением загрузить фото или сменить настройки (рис. 3.17).

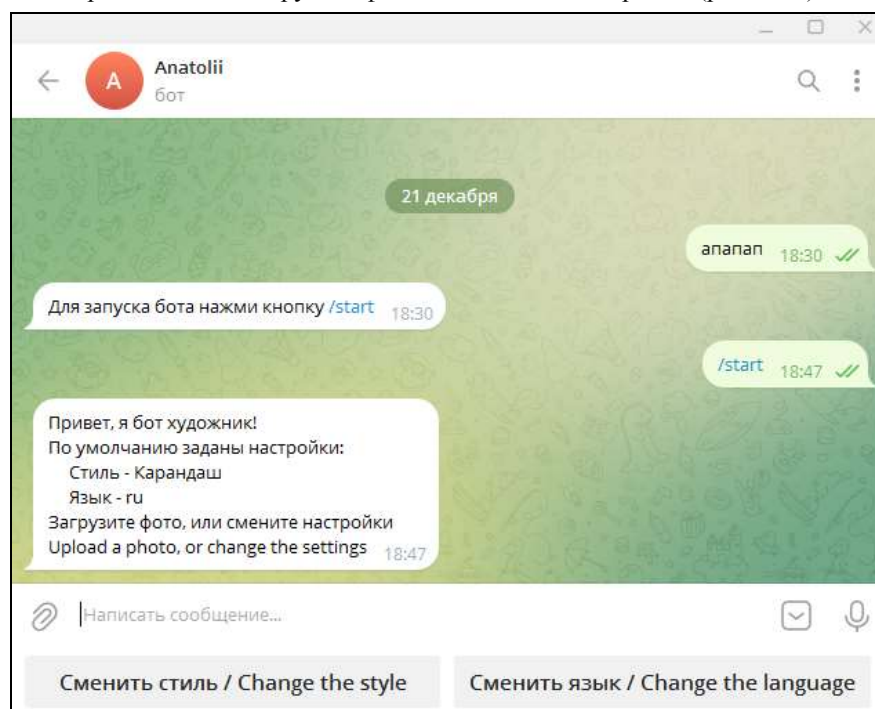


Рис. 3.17. Ответ бота на команду /start

Для большего удобства работы сначала сделаем выбор языка интерфейса. После нажатия на кнопку «Сменить язык» будет выполнен следующий фрагмент программного кода:

```
# Меню для смены языка
elif message.text == "Сменить язык / Change the language":
    markup = types.ReplyKeyboardMarkup(resize_keyboard=True)
    btn1 = types.KeyboardButton("Меню RU Русский")
    btn2 = types.KeyboardButton('Menu GB English')
    markup.add(btn1, btn2)
    ms_text = "Смените язык \nChange the language"
    await bot.send_message(message.chat.id, ms_text, reply_markup=markup)
```

Пользователю будет предложено сменить язык, и показаны две кнопки (рис. 3.18).

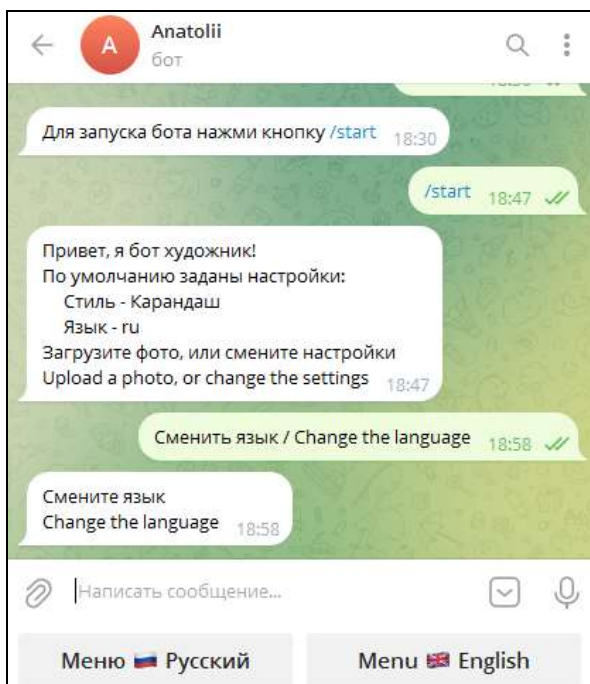


Рис. 3.18. Кнопки для выбора языка

Если будет нажата кнопка «Меню Русский», то вступит в работу следующий фрагмент программного кода:

```
# После смены языка на русском
elif message.text == "Меню RU Русский":
    # Создать кнопки
    markup = types.ReplyKeyboardMarkup(resize_keyboard=True)
    btn_st = types.KeyboardButton("Сменить стиль / Change the style")
    btn_lg = types.KeyboardButton("Сменить язык / Change the language")
    markup.add(btn_st, btn_lg)

    # Отправить сообщение
    language = 'ru'
    text_style_ru = 'Карандаш'
    ms2 = '\nЗаданы новые настройки:'
```

```

ms3 = '\n      - ' + text_style
ms4 = '\n      - Язык: ' + language
ms5 = '\nЗагрузите фото, или смените настройки'
ms_text = ms2 + ms3 + ms4 + ms5
await bot.send_message(message.chat.id, ms_text, reply_markup=markup)

```

После этого загрузится главное меню, и все сообщения будут отображаться на русском языке (рис. 3.19).

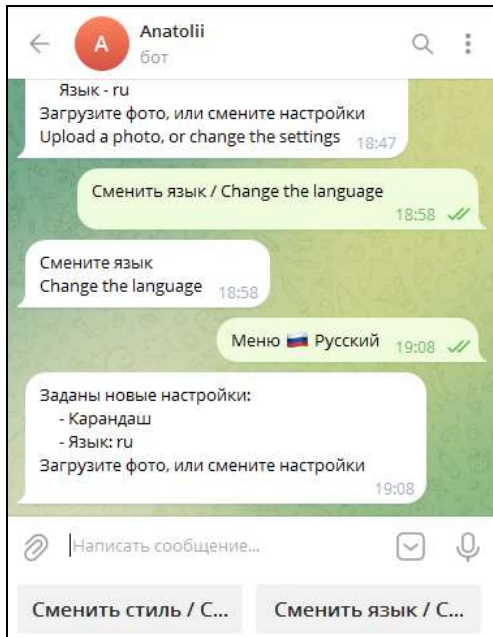


Рис. 3.19. Интерфейс пользователя после смены языка

Теперь пользователь может выбрать и отправить боту любое фото. Отправленное фото будет обработано следующим фрагментом программного кода:

```

# Получить фото от пользователя и обработать
@bot.message_handler(content_types=['photo'])
async def photo(message: types.Message):
    if message.photo:
        # Идентификатор полученного фото
        image_id = message.photo[len(message.photo) - 1].file_id

        # Путь к папке с полученным фото
        file_path = (await bot.get_file(image_id)).file_path
        # print('file_path=', file_path)

        # Загрузка полученного фото
        downloaded_file = bot.download_file(file_path)

        # Сгенерированное имя полученного файла
        file_name = file_path

        # Сохранить файл с именем file_name
        with open(file_name, 'wb') as new_file:

```

```

        new_file.write(await downloaded_file)

        # Отправить ответ пользователю
        if language == 'ru':
            await bot.reply_to(message, "Фото получено, запущен процесс обра-
ботки...")
        else:
            await bot.reply_to(message, "The photo has been re-
ceived,\nprocessing has begun...")

        # Обращение к классу стилизации изображения
        in_image = file_name # исходное изображение
        img_style = in_style # изображение стиля

        # Запуск процесса стилизации фото
        M_S_21.styl_21(in_image, path_style_img, img_style, style_model, size)

        # Отправка ответа пользователю
        if language == 'ru':
            await bot.reply_to(message, "Фото обработано, получите ответ")
        else:
            await bot.reply_to(message, "The photo has been processed,\nyou
will receive a response")

        # Чтение стилизованного фото
        with open(path_style_img, "rb") as file:
            style_foto = file.read()
        # Отправка стилизованного фото пользователю
        await bot.send_photo(message.from_user.id, photo=style_foto)

        # Удалит с диска принятое фото
        os.remove(file_name)
    else:
        await bot.reply_to(message, "Вы прислали не фотографию")

```

Здесь бот примет фото от пользователя, передаст его нейронной сети для обработки и со-хранит на диске стилизованное изображение. Затем вернет обработанное изображение пользователю (рис. 3.20).

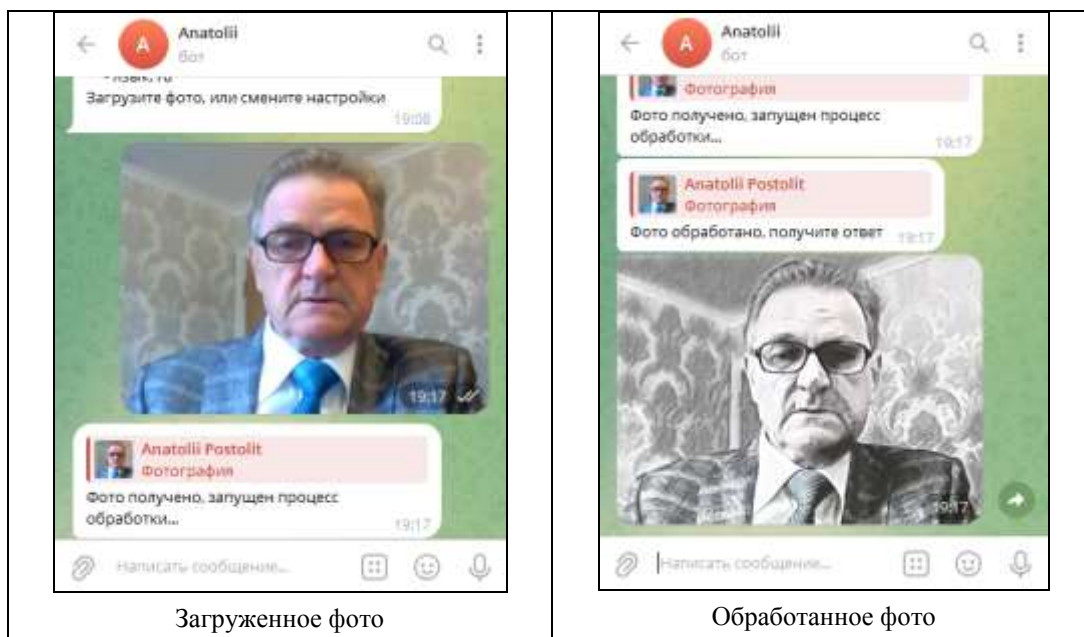


Рис. 3.20. Загруженное и обработанное фото в стиле «Карандаш»

Как видно из данного рисунка, в этом случае фото обработано стилем «Карандашный рисунок».

Если пользователь нажмет кнопку «Сменить стиль», то отработает следующий фрагмент программного кода:

```
# Набор кнопок для смены стиля на русском
elif message.text == "Сменить стиль / Change the style":
    markup = types.ReplyKeyboardMarkup(resize_keyboard=True)
    # На русском
    if language == 'ru':
        btn_candi = types.KeyboardButton("Кандинский 1")
        btn_compos = types.KeyboardButton("Кандинский 2")
        btn_escher = types.KeyboardButton("Мауриц Корнелис Эшер")
        btn_feathers = types.KeyboardButton("Перья")
        btn_frida_kahlo = types.KeyboardButton("Фрида Карло")
        btn_la_muse = types.KeyboardButton("Пикассо муза")
        btn_mos1 = types.KeyboardButton("Мозаика 1")
        btn_mos2 = types.KeyboardButton("Мозаика 2")
        btn_noise = types.KeyboardButton("Звездный шум")
        btn_pencil = types.KeyboardButton("Карандаш")
        btn_pic_auto = types.KeyboardButton("Пикассо автопортрет")
        btn_rain_princess = types.KeyboardButton("Принцесса дождя")
        btn_del_port = types.KeyboardButton("Делоне портрет")
        btn_pic_kub = types.KeyboardButton("Пикассо кубизм")
        btn_ship = types.KeyboardButton("Айвазовский")
        btn_van_gog = types.KeyboardButton("Ван Гог Звездная ночь")
        btn_stars = types.KeyboardButton("Звезды")
        btn_strip = types.KeyboardButton("Штрихи")
        btn_scream = types.KeyboardButton("Эдвард Мунк Крик")
```

```

        btn_udnie = types.KeyboardButton("Франсис Пикабия")
        btn_wave = types.KeyboardButton("Хокусай Волны")
        btn_matisse = types.KeyboardButton("Матисс")
        back = types.KeyboardButton("Меню RU Русский")
# На английском
else:
    btn_candi = types.KeyboardButton("Kandinsky 1")
    btn_compos = types.KeyboardButton("Kandinsky 2")
    btn_escher = types.KeyboardButton("Mauritz Cornelis Escher")
    btn_feathers = types.KeyboardButton("Feathers")
    btn_frida_kahlo = types.KeyboardButton("Frida Carlo")
    btn_la_muse = types.KeyboardButton("Picasso the Muse")
    btn_mos1 = types.KeyboardButton("Mosaic 1")
    btn_mos2 = types.KeyboardButton("Mosaic 2")
    btn_noise = types.KeyboardButton("Starry Noise")
    btn_pencil = types.KeyboardButton("Pencil")
    btn_pic_auto = types.KeyboardButton("Picasso's self-portrait")
    btn_rain_princess = types.KeyboardButton("Rain Princess")
    btn_del_port = types.KeyboardButton("Delaunay portrait")
    btn_pic_kub = types.KeyboardButton("Picasso Cubism")
    btn_ship = types.KeyboardButton("Aivazovsky")
    btn_van_gog = types.KeyboardButton("Van Gogh Starry Night")
    btn_stars = types.KeyboardButton("Stars")
    btn_strip = types.KeyboardButton("Strokes")
    btn_scream = types.KeyboardButton("Edvard Munch Creek")
    btn_udnie = types.KeyboardButton("Francis Picabia")
    btn_wave = types.KeyboardButton("Hokusai Waves")
    btn_matisse = types.KeyboardButton("Matisse")
    back = types.KeyboardButton("Menu GB English")

markup.add(btn_candi, btn_compos, btn_escher, btn_feathers,
            btn_frida_kahlo, btn_la_muse, btn_mos1, btn_mos2,
            btn_noise, btn_pencil, btn_pic_auto, btn_rain_princess,
            btn_del_port, btn_pic_kub, btn_ship, btn_van_gog, btn_stars,
            btn_strip, btn_scream, btn_udnie, btn_wave, btn_matisse, back)
await bot.send_message(message.chat.id, text="Смените стиль \nChange the
style", reply_markup=markup)

```

В этом фрагменте кода формируется набор кнопок, при нажатии на которые будет выбран новый стиль (рис. 3.21).

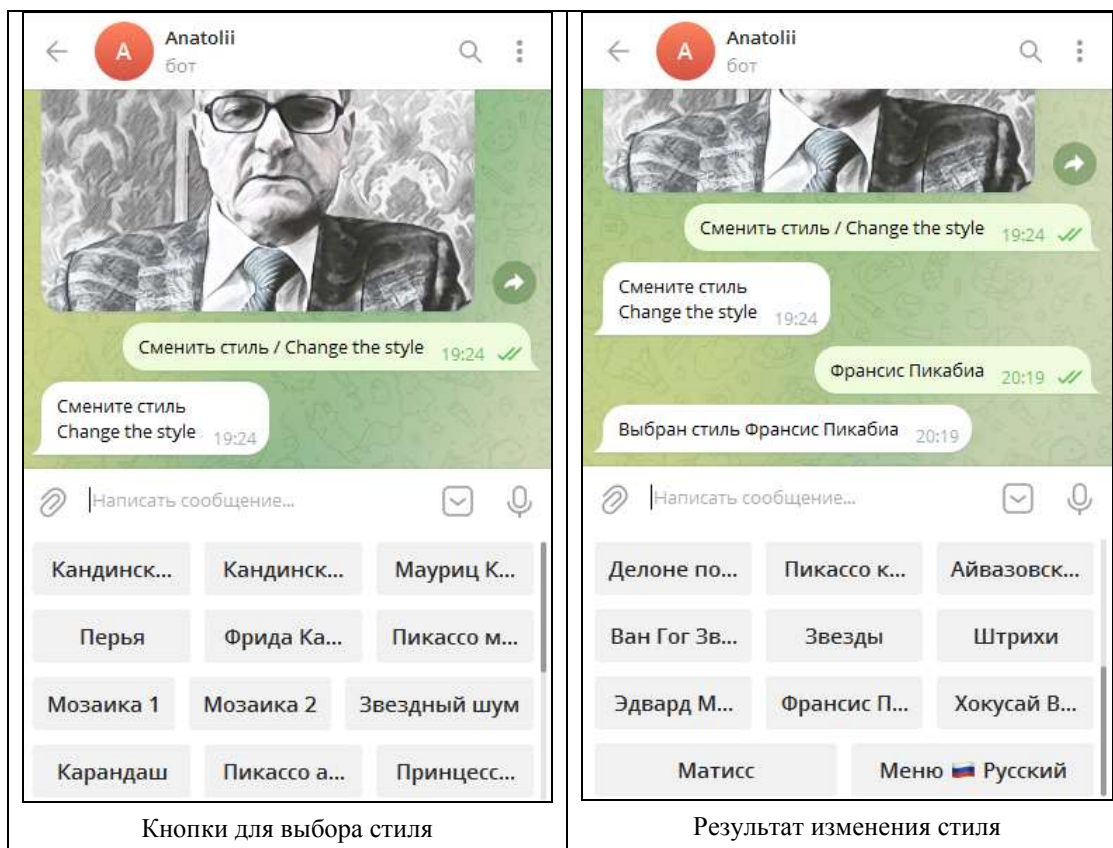


Рис. 3.21. Интерфейс пользователя для смены стиля

Так как все кнопки не помещаются в экран, то с правой стороны появляется полоса прокрутки с бегуном. Пользователь может путем скроллинга перемещать кнопки и, нажатием на кнопку, сменить стиль. Информация о выбранном стиле будет возвращена пользователю. После этого можно снова загрузить фотографию и она будет обработана другим стилем (рис. 3.22).

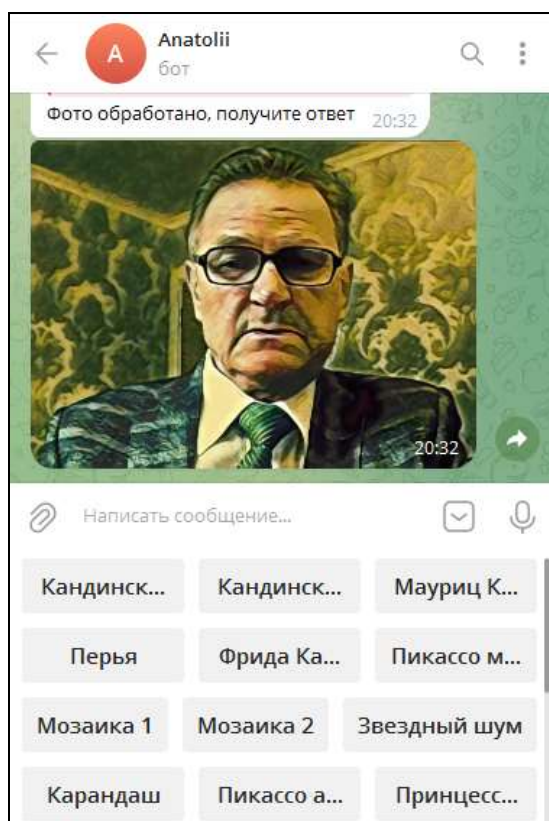


Рис. 3.22. Фото, обработанное стилем Фрида Карло

Как видно из данного рисунка, фотография пользователя обработана стилем «Фрида Карло».

В программном коде обратите внимание на префиксы `async`, которые стоят перед именами функций, и `await`, которые стоят в строках с отправкой сообщений. Именно эти префиксы обеспечивают асинхронную обработку запросов пользователей.

Примечание.

Программный модуль `bot_buton_1.py` можно запустить на локальном компьютере, который имеет выход в интернет. При этом ваш бот будет доступен удаленным пользователем до тех пор, пока запущенное приложение находится в работе, и станет не активным после завершения работы модуля или после выключения компьютера. Для того, чтоб бот всегда был в состоянии активности его нужно разместить на публичном сервере.

3.7. Краткие итоги

Итак, мы познакомились с некоторыми базовыми основами создания телеграмм ботов на Python и технологии взаимодействия телеграмм ботов с нейронными сетями. Научились создавать кнопки для упрощения взаимодействия пользователей с ботами и использовать методы асинхронного программирования.

Теперь перейдем к изучению возможностей обработки изображений с использованием нейронных сетей. Рассмотрению этих вопросов и посвящена следующая глава.

ГЛАВА 4

Обработка изображений с использованием нейронных сетей



Нейронные сети стали, пожалуй, наиболее мощным и удобным инструментом в области обработки изображений. Используемые алгоритмы машинного обучения позволяют компьютерам анализировать, понимать и извлекать информацию из графических данных, что привело к революционным результатам в различных отраслях, начиная от медицины и завершая промышленностью, транспортом и развлечениями.

Нейронные сети используются для анализа медицинских изображений, таких как рентгеновские снимки, МРТ, УЗИ для выявления заболеваний на ранней стадии. На транспорте нейронные сети применяются для обнаружения объектов на дороге, распознавания дорожных знаков, оценки окружающей обстановки, управления беспилотными автомобилями. На производстве с использованием нейронных сетей выполняется обнаружения дефектов в различных изделиях.

Нейронные сети играют решающую роль в обработке изображений, превращая большие объемы визуальной информации в ценные данные. С каждым днем нейронные сети в обработке изображений становятся все более эффективными и точными, что делает их ключевым элементом в современном мире технологий.

В данной главе будут представлены только некоторые базовые принципы использования нейронных сетей для обработки изображений и интеграции нейронных сетей в веб приложения. Из материалов данной главы вы узнаете:

- ☐ что такое бесшовное клонирование изображений, обеспечивающее объединение нескольких изображений в одно целое;
- ☐ как можно черно-белое изображение (например, старое фото) превратить в цветное;
- ☐ как можно калоризовать (раскрасить) черно-белый видео файл или старый черно белый фильм;
- ☐ как с использованием дизеринга можно цветное изображение трансформировать в черно-белое (нарисованное точками);
- ☐ как можно сгенерировать изображение на основе его текстового описания.

4.1. Бесшовное клонирование с использованием OpenCV

Одна из функций, представленных в OpenCV, называется Seamless Cloning (бесшовное клонирование). С помощью этой функции можно взять объект из одного изображения и вставить его в другое изображение. Другими словами передний план одного изображения нало-

жить на другой фон. Если выполнить такой наложение без дополнительной обработки изображений, результат бы имел довольно неприглядный вид. Предположим, что мы имеем фоновое изображение с такими элементами, как поле, лес и небо, и накладываем на него изображение самолета (рис.4.1).



Рис. 4.1. Наложение изображения самолета на фоновую картинку в графическом редакторе

Конечно, перед наложением можно обработать изображение самолета в графическом редакторе, но на это уйдет достаточно много времени, нужно иметь опыт отделения переднего плана от фона и создания 4-канальных изображений. К счастью всю эту черновую работу может взять на себя библиотека OpenCV, причем достаточно быстро и качественно.

Для реализации данного метода на компьютер необходимо установить следующие библиотеки:

```
pip install python-opencv
```

```
pip install numpy
```

Создадим программный модуль, который будет обеспечивать бесшовное клонирование изображения, то есть выделение объект в одном изображении и перенос его на другое изображение (листинг 4.1).

Листинг 4.1. Модуль Clone.py

Примечание.

В демо-версии книги удалены листинги программ.

Это достаточно простой программный код. Здесь в теле функции clone создается два объекта:

- ❑ fon – изображение фона;
- ❑ obj – изображение объекта переднего плана.

После этого для объекта переднего плана создается маска (mask), и определяются координаты на изображении фона, куда будет помещен центр объекта. После этого с использованием метода `seamlessClone()` библиотеки `OpenCV` выполняется смешивание изображений, или наложение объекта (передний план) на фон. Итоговое изображение будет сохранено на диске. Если запустить данный модуль на выполнение в среде `Python`, то будет получен следующий результат (рис.4.2).



Рис. 4.2. Наложение изображения самолета на фоновое изображение с библиотекой `OpenCV`

Как видно из данного рисунка, мы получили практически бесшовное объединение двух изображений. Теперь встроим этот модуль в приложение `Django`. Для этого в одном из шаблонов или элементе меню необходимо создать ссылку, которая активирует нужный URL маршрут (листинг 4.2).

Листинг 4.2. Ссылка активации маршрута `clone_1`

Примечание.

В демо-версии книги удалены листинги программ.

По данной ссылке в приложении `api_vision` будет активирован URL маршрут с именем `clone_1` (листинг 4.3).

Листинг 4.3. Маршрут URL с именем `clone_1`

Примечание.

В демо-версии книги удалены листинги программ.

По этому маршруту будет активировано представление `clone_1` (листинг 4.4).

Листинг 4.4. Представление (view) clone_1

Примечание.

В демо-версии книги удалены листинги программ.

Это достаточно простой программный код, в котором формируются пути к файлам с изображениями:

- ❑ `img_fon` - путь к изображению фона;
- ❑ `img_obj` - путь к изображению объекта;
- ❑ `img_rez` - путь к итоговому файлу.

Затем формируются URL адреса изображений:

- ❑ `img_obj_url` - URL адрес изображения объекта;
- ❑ `img_fon_url` - URL адрес изображения фона;
- ❑ `img_rez_url` - URL адрес итогового изображения.

На следующем шаге делается обращение к функции `clone` модуля `clon`, где выполняется совмещение двух изображений.

После этого создается контекстная переменная `context`, которая передается в шаблон с именем `clone_1`. Этот шаблон имеет следующий код (листинг 4.5).

Листинг 4.5. Шаблон clone_1.html

Примечание.

В демо-версии книги удалены листинги программ.

Это достаточно простой шаблон, в котором имеется три контейнера: первый отвечает за вывод изображения объекта переднего плана, второй – за вывод изображения фона, третий – за вывод совмещенного изображения.

Примечание.

В интерактивной цифровой книге для запуска данного примера нужно на вкладке «Показ примеров» нажать на кнопку «Примеры главы 4» и активировать ссылку «Бесшовное клонирование».

Внешний вид шаблона `clone_1.html` представлен на рис.4.3.



Рис.4.3. Исходные изображения и обработанное изображение в шаблон Django

4.2. Трансформация черно-белых изображений в цветные с использованием нейронных сетей

С использованием нейронных сетей можно раскрашивать старые черно-белые фотографии, и даже фильмы. В данном разделе будет рассмотрена реализация этого процесса на Python. Трансформацию черно-белых изображений в цветные можно реализовать как на локальном компьютере, так и в веб приложении.

В данном разделе показан пример раскрашивания черно-белых изображений с использованием предварительно обученной нейронной сети. Это нейронная сеть была обучена на более чем 1.3 млн. изображений, а файлы сохраненной модели занимают на диске около 300 Мб. Эта нейронная сеть, не смотря на внушительные размеры, работает очень быстро и формирует цветные изображения очень высокого качества. Принцип работы данной сети схематично представлен на рис. 4.4.

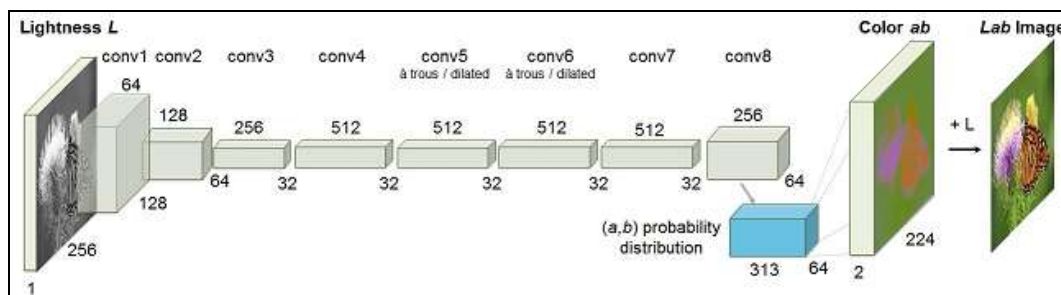


Рис.4.4. Принцип работы нейронной сети для формирования цветных изображений

4.2.1. Класс для трансформации черно-белых изображений в цветные

Мы не будем рассматривать теоретические принципы и структуру данной нейронной сети, а реализуем набор программных модулей для ее практического использования.

На первом шаге нужно найти в сети интернет и скачать следующие файлы модели:

- ☐ colorization_deploy_v2.prototxt;
- ☐ colorization_release_v2.caffemodel;
- ☐ pts_in_hull.npy.

Данные файлы нужно поместить в папку с моделями нейронных сетей (в данном примере это папка с именем «net_model»). Сначала создадим программный модуль Grey_Color_Img.py, в котором реализуем класс для генерации цветных изображений из черно-белых (листинг 4.6).

Листинг 4.6. Программный код модуля Grey_Color_Img.py

Примечание.

В демо-версии книги удалены листинги программ.

Для работы данного класса нам понадобилось выполнить импорт всего двух библиотек – OpenCV и numpy. Для отладки работы этого модуля в среде Python на локальном компьютере были сформированы следующие пути к папкам и файлам:

- ☐ path_image = '../media/images/grey_img.jpg' - путь к черно-белому изображению;
- ☐ proto_model = '../net_model/colorization_deploy_v2.prototxt' – путь к файлу нейронной сети;
- ☐ caf_model = '../net_model/colorization_release_v2.caffemodel' – путь к файлу нейронной сети;
- ☐ pts = '../net_model/pts_in_hull.npy' – путь к файлу нейронной сети.

Примечание.

Так как отладка данного программного модуля выполнялась в среде Windows, то пути к папкам и файлам указаны в соответствии с требованиями ОС Windows. Если вы работаете под Linux, то пути к файлам и папкам нужно указывать в соответствии с требованиями ОС Linux.

В этом приложении с использованием библиотеки OpenCV в объект `image` было загружено черно-белое изображение, а затем данное изображение вместе с путями к файлам нейронной сети переданы классу `Made_Color`. Делать подробный разбор программного кода данного класса делать не будем, так как сведения об основных процедурах указаны в соответствующих комментариях программного кода. Данный класс возвращает цветное изображение в объект с именем `new_img` и записывается на диск под именем `color_img.jpg`. Исходное и обработанное изображение будут показаны на экране с использованием библиотеки OpenCV.

Примечание.

Данный модуль не включен в состав исполняемых примеров интерактивной цифровой книги, так как он предназначен для работы на локальном компьютере. Но вы имеете возможность скопировать листинг данной программы, запустить данный программный модуль на выполнение на своем локальном компьютере и использовать в своих приложениях.

Если данный программный код запустить на выполнение на локальном компьютере, то будет получен следующий результат (рис. 4.5).



Рис.4.5. Результаты работы программного модуля `Grey_Color_Img.py`

Как видно из данного рисунка, нейронная сеть справилась со своей задачей. На выходе было получено качественное цветное изображение, а время работы программного модуля составило менее одной секунды.

4.2.2. Трансформации черно-белых изображений в цветные в веб приложениях

Теперь посмотрим, как можно реализовать подход, описанный в предыдущем разделе в веб приложении на Django.

Для активации модулей преобразования черно-белых изображений в цветные нужно создать следующую ссылку (листинг 4.7):

Листинг 4.7. Ссылка для активации модулей черно-белых изображений в цветные

Примечание.

В демо-версии книги удалены листинги программ.

По этой ссылке будет активирован следующий маршрут URL (листинг 4.8):

Листинг 4.8. Маршрут `made_color`

Примечание.

В демо-версии книги удалены листинги программ.

По этому маршруту будет вызвано представление `views.made_color` (листинг 4.9):

Листинг 4.9. Представление `views.made_color`

Примечание.

В демо-версии книги удалены листинги программ.

В данном представлении сначала выполнен импорт модуля `Grey_Color_Img`, это модуль Python с классом `Made_Color`, который был описан в предыдущем разделе. Затем были сформированы пути к папкам и файлам проекта:

- ❑ `my_app = base_dir` - путь к приложению;
- ❑ `path_media = os.path.join(my_app, 'media')` - путь к папке `media`;
- ❑ `path_grey_image = os.path.join(path_media, 'images/grey_img.jpg')` - путь к черно-белому изображению;
- ❑ `path_color_image = os.path.join(path_media, 'images/color_img.jpg')` - путь к цветному изображению;
- ❑ `path_model = os.path.join(my_app, 'net_model')` - путь к папке с моделями;
- ❑ `proto_model = os.path.join(path_model, 'colorization_deploy_v2.prototxt')` - путь к файлу модели;
- ❑ `caf_model = os.path.join(path_model, 'colorization_release_v2.caffemodel')` - путь к файлу модели;
- ❑ `pts = os.path.join(path_model, 'pts_in_hull.npy')` - путь к файлу модели.

После этого исходное черно-белое изображение загружено в объект `image`, на основе класса `Made_Color` создан объект `may_color`, которому передано исходное изображение и пути к файлам модели нейронной сети. В классе `Made_Color` было сформировано цветное изображение, которое записано на диск с именем, указанным в объекте `path_color_image`.

На следующем шаге сформировано два URL адреса: URL адрес черно-белого изображения (`grey_url`), URL адрес цветного изображения (`color_url`).

Затем URL адреса черно-белого и цветного изображения через контекстную переменную `context` передаются в шаблон `made_color.html`, который имеет следующий код (листинг 4.10).

Листинг 4.10. Шаблон `made_color.html`

Примечание.

В демо-версии книги удалены листинги программ.

В этом шаблоне всего два тега – для вывода черно белого и цветного изображения.

Примечание.

В интерактивной цифровой книге для запуска данного примера нужно на вкладке «Показ примеров» нажать на кнопку «Примеры главы 4» и активировать ссылку «Калоризация изображения».

Если выполнить данный программный модуль, то в окне браузера отобразится следующий результат (рис. 4.6).



Рис.4.6. Результаты обработки черно-белого изображения в шаблоне made_color.html

4.3. Трансформация черно-белых видео файлов в цветные с использованием нейронных сетей

4.3.1. Класс для трансформации черно-белых видео файлов в цветные

Нейронные сети сильно упрощают процесс обработки видео, в частности цветовую коррекцию кадров. Теперь достаточно просто превратить черно-белый фильм в цветной, не прибегая к «ручному» раскрашиванию кадров. Нейронные сети способны анализировать цвета в видео материалах и автоматически подбирать оптимальный баланс различных цветов, контрастность, яркость и насыщенность.

Если вы когда-нибудь открывали коробку со старыми домашними видео и обнаруживали, что они в черно-белом цвете, то не нужно расстраиваться. Даже на обычном домашнем компьютере вы сможете превратить их в яркое цветное кино. Сегодня получила широкое распространение и развитие технология под названием «колоризация видео», которая может вдохнуть в черно-белые видеоролики совершенно новую жизнь!

В предыдущих разделах были рассмотрены возможности колоризации изображений, эти же принципы и нейронные сети можно использовать и для обработки видео. Ведь видео файлы это набор из последовательности изображений, значить, в программах нужно организовать цикл перебора кадров видео файлов с обработкой каждого кадра.

Для начала создадим класс, который обеспечит циклическую обработку видео кадров с использованием той же модели нейронной сети, что и в предыдущих разделах:

- ❑ `colorization_deploy_v2.prototxt;`
- ❑ `colorization_release_v2.caffemodel;`
- ❑ `pts_in_hull.npy.`

Создадим программный модуль `Grey_Color_Video.py`, в котором реализуем класс для формирования цветных видео кадров из черно-белых (листинг 4.11).

Листинг 4.11. Программный код модуля `Grey_Color_Video.py`

Примечание.

В демо-версии книги удалены листинги программ.

В данном программном модуле реализован тот же подход, что и в модуле `Grey_Color_Image.py` предыдущего раздела. Для работы созданного класса понадобилось выполнить импорт всего двух библиотек – `OpenCV` и `numpy`. Для отладки работы этого модуля в среде `Python` на локальном компьютере были сформированы следующие пути к папкам и файлам:

- ❑ `path_video = '../media/video/grey_video.mp4'` - путь к исходному видео;
- ❑ `path_out = '../media/video/video_out.mp4'` - путь к обработанному видео;
- ❑ `proto_model = '../net_model/colorization_deploy_v2.prototxt` – путь к файлу нейронной сети;
- ❑ `caf_model = '../net_model/colorization_release_v2.caffemodel` – путь к файлу нейронной сети;
- ❑ `pts = '../net_model/pts_in_hull.npy'` – путь к файлу нейронной сети.

Примечание.

Так как отладка данного программного модуля выполнялась в среде `Windows`, то пути к папкам и файлам указаны в соответствие с требованиями ОС `Windows`. Если вы работаете под `Linux`, то пути к файлам и папкам нужно указывать в соответствие с требованиями ОС `Linux`.

В этом приложении объекты с путями к видео и файлам нейронной сети переданы классу `Made_Color_Video`. Подробный разбор программного кода данного класса делать не будем, так как сведения об основных процедурах указаны в соответствующих комментариях в программном коде. В программном коде данного класса организован цикл захвата и перебора кадров видео файла. В теле цикла каждый кадр передается нейронной сети для обработки, а обработанные цветные кадры записываются в новый видео файл.

Примечание.

Данный модуль не включен в состав исполняемых примеров интерактивной цифровой книги, так как он предназначен для работы на локальном компьютере. Но вы имеете возможность скопировать листинг данной программы, запустить данный программный модуль на выполнение на своем локальном компьютере и использовать в своих приложениях.

Если данный программный код запустить на выполнение на локальном компьютере, то в паке `media/video` будет создан новый видео файл с именем `video_out.mp4`. Данный файл вы можете просмотреть на своем локальном компьютере.

Следует отметить, что на обработку каждого кадра уходит порядка одной секунды. Видео файл данного примера содержит 4684 кадра и его обработка на ПК с объемом памяти 12 Гб заняла порядка одного часа. Поэтому при калоризации черно-белых видео нужно запастись терпением, а еще, так как процессор будет задействован почти на 100%, вентиляторы вашего компьютера будут сильно шуметь, работая на полную мощность.

Рисунок с результатами работы данного программного модуля будет показан в следующем разделе, где мы реализуем калоризацию видео файлов в веб приложении на Django.

4.3.2. Трансформации черно-белых видео файлов в цветные в веб приложениях

Теперь посмотрим, как можно реализовать подход калоризации видео файлов, описанный в предыдущем разделе в веб приложении на Django.

Для активации модулей преобразования черно-белых видео файлов в цветные нужно создать следующую ссылку (листинг 4.12):

Листинг 4.12. Ссылка для активации модулей черно-белых видео в цветные

Примечание.

В демо-версии книги удалены листинги программ.

По этой ссылке будет активирован следующий маршрут URL (листинг 4.13):

Листинг 4.13. Маршрут `color_video`

Примечание.

В демо-версии книги удалены листинги программ.

По этому маршруту будет вызвано представление `views.color_video` (листинг 4.14):

Листинг 4.14. Представление `views.color_video`

Примечание.

В демо-версии книги удалены листинги программ.

В данном представлении сначала выполнен импорт модуля `Grey_Color_Video`, это модуль Python с классом `Made_Color_Video`, который был описан в предыдущем разделе. Затем были сформированы пути к папкам и файлам проекта:

- ☐ `my_app = base_dir` - путь к приложению;
- ☐ `path_media = os.path.join(my_app, 'media')` - путь к папке `media`;
- ☐ `path_grey_video = os.path.join(path_media, 'video/video_grey.mp4')` - путь к черно-белому видео;

- ❑ `path_color_video = os.path.join(path_media, 'video/video_out.mp4')` - путь к цветному видео;
- ❑ `path_model = os.path.join(my_app, 'net_model')` - путь к папке с моделями;
- ❑ `proto_model = os.path.join(path_model, 'colorization_deploy_v2.prototxt')` - путь к файлу модели;
- ❑ `caf_model = os.path.join(path_model, 'colorization_release_v2.caffemodel')` - путь к файлу модели;
- ❑ `pts = os.path.join(path_model, 'pts_in_hull.npy')` - путь к файлу модели.

После этого на основе класса `Made_Color_Video` создан объект `may_color`, которому передано пути к видео файлам и файлам модели нейронной сети. В классе `Made_Color_Video` был сформирован цветной видео файл, который записан на диск с именем, указанном в объекте `path_color_video`.

На следующем шаге сформировано два URL адреса: URL адрес черно-белого видео файла (`grey_url`), URL адрес цветного видео файла (`color_url`).

Затем URL адреса черно-белого и цветного видеофайлов через контекстную переменную `context` передаются в шаблон `color_video.html`, который имеет следующий код (листинг 4.15).

Листинг 4.15. Шаблон `color_video.html`

Примечание.

В демо-версии книги удалены листинги программ.

В этом шаблоне всего два тега – для вывода видео проигрывателя для черно-белого и цветного видео.

Примечание.

В интерактивной цифровой книге для запуска данного примера нужно на вкладке «Показ примеров» нажать на кнопку «Примеры главы 4» и активировать ссылку «Калоризация видео файлов».

Если выполнить данный программный модуль, то в окне браузера отобразится следующий результат (рис. 4.7).



Рис.4.7. Результаты обработки черно-белого видео файла в шаблоне color_video.html

Как видно из данного рисунка, в шаблоне имеется два видео проигрывателя, которые позволяют смотреть черно-белое и цветное видео, не зависимо друг от друга.

Примечание.

Так как формирование цветного видеофайла данного примера длится порядка одного часа, то на публичном сервере строка обращения к функции калоризации видео файла закомментирована (для сокращения времени формирования результатов). В данном шаблоне показан заранее созданный данным приложением итоговый видео файл.

4.4. Применение методов дизеринга для обработки изображений

Дизеринг — это метод, используемый для сглаживания цветовых градиентов и уменьшения полос в цифровых изображениях. В Python можно применять методы дизеринга с помощью библиотеки Pillow, которая является ответвлением библиотеки Python Imaging Library (PIL). Pillow предоставляет несколько алгоритмов дизеринга, включая дизеринг Флойда-Стейнберга, именно этот алгоритм будет использован в данном разделе для обработки цветных изображений. С помощью данного метода можно преобразовать цветное фото в черно-белое изображение (белые точки на черном фоне). Такие изображения обычно используются ритуальными службами для нанесения изображений на камне с помощью ударно-гравировочных станков.

4.4.1. Функция для дизеринга изображений

Для начала создадим программный модуль, который обеспечит дизеринг изображений. Для работы данного модуля необходимо на компьютер установить библиотеку Pillow:

```
pip install pillow
```

Программный код модуля на Python, реализующий данный метод, представлен в листинге 4.16.

Листинг 4.16. Программный код модуля point_img.py

Примечание.

В демо-версии книги удалены листинги программ.

Примечание.

Так как отладка данного программного модуля выполнялась в среде Windows, то пути к папкам и файлам указаны в соответствии с требованиями ОС Windows. Если вы работаете под Linux, то пути к файлам и папкам нужно указывать в соответствии с требованиями ОС Linux.

Это достаточно простой программный код, в котором выполнен импорт модуля Image из библиотеки Pillow. Далее следует функция, которая принимает путь к исходному и к обработанному изображению. Затем исходное изображение загружается в объект «im», обрабатывается методами библиотеки Pillow и обработанное изображение сохраняется на диске.

Примечание.

Данный модуль не включен в состав исполняемых примеров интерактивной цифровой книги, так как он предназначен для работы на локальном компьютере. Но вы имеете возможность скопировать листинг данной программы, запустить данный программный модуль на выполнение на своем локальном компьютере и использовать в своих приложениях.

Если данный программный код запустить на выполнение на локальном компьютере, то будет получен следующий результат (рис. 4.8 исходное изображение, рис 4.9 — обработанное изображение).



Рис.4.8. Исходное цветное изображение



Рис.4.9. Изображение, обработанное методом дизеринга

Как видно из данного рисунка, из цветного изображение убраны все цвета, кроме черного и белого. Фон изображения черный, а тени, и рельеф элементов объекта созданы путем нанесения белых точек на разном расстоянии друг от друга.

4.4.2. Применение методов дизеринга для обработки изображений в веб приложениях

Теперь рассмотрим, как этот метод обработки изображений можно реализовать в веб приложениях на Django.

Для активации модулей дизеринга изображений нужно создать следующую ссылку (листинг 4.17):

Листинг 4.17. Ссылка для активации модулей дизеринга изображений

Примечание.

В демо-версии книги удалены листинги программ.

По этой ссылке будет активирован следующий маршрут URL (листинг 4.18):

Листинг 4.18. Маршрут point_image

Примечание.

В демо-версии книги удалены листинги программ.

По этому маршруту будет вызвано представление `views.point_image` (листинг 4.19):

Листинг 4.19. Представление views.point_image

Примечание.

В демо-версии книги удалены листинги программ.

В данном представлении сначала выполнен импорт модуля `point_img`, это модуль Python с функцией `image_point`, которая была описана в предыдущем разделе. Затем были сформированы пути к папкам и файлам проекта:

- ❑ `my_app = base_dir` - путь к приложению;
- ❑ `path_media = os.path.join(my_app, 'media')` - путь к папке `media`;
- ❑ `path_inp_img = os.path.join(path_media, 'images/Atlet.jpg')` - путь к исходному изображению;
- ❑ `path_out_img = os.path.join(path_media, 'images/point_Atlet.jpg')` - путь к обработанному изображению.

После этого сделано обращение к функции `image_point`, в которую переданы пути к исходному и обработанному изображению. В теле данной функции выполняется дизеринг исходного изображения с записью обработанного изображения на диск.

На следующем шаге сформировано два URL адреса: URL адрес исходного изображения (`inp_url`), URL адрес обработанного изображения (`out_url`).

Затем URL адреса исходного и обработанного изображений через контекстную переменную `context` передаются в шаблон `point_image.html`, который имеет следующий код (листинг 4.20).

Листинг 4.20. Шаблон point_image.html

Примечание.

В демо-версии книги удалены листинги программ.

В этом шаблоне всего два тега – для вывода исходного и обработанного изображений.

Примечание.

В интерактивной цифровой книге для запуска данного примера нужно на вкладке «Показ примеров» нажать на кнопку «Примеры главы 4» и активировать ссылку «Дизеринг изображений».

Если выполнить данный программный модуль, то в окне браузера отобразится следующий результат (рис. 4.10).



Рис.4.10. Исходное изображение и изображение, обработанное методом дизеринга

4.5. Генерация изображений на KerasCV с помощью Stable Diffusion

В настоящее время появилась возможность генерировать новые изображения по текстовому описанию при помощи нейронных сетей и методов машинного обучения. Stable Diffusion — это нейронная сеть от группы студии Stability.ai с открытым исходным кодом, которая позволяет генерировать изображения на основе текстового запроса, а также дорисовывать наброски и редактировать исходные изображения. В основе Stable Diffusion лежит диффузионная модель, которая является аналогом популярных нейронных сетей Midjourney и DALL-E, но, в отличие от них, Stable Diffusion можно использовать бесплатно.

Данная нейронная сеть имеет открытый исходный код, что позволяет скачать ее на Github и использовать на своем компьютере. Но есть и нюанс — у пользователя должен быть мощный компьютер и видеокарта Nvidia с объемом графической памяти порядка 4-8 Гбайт. Программные модули будут работать и на компьютере с другими видеокартами (объем оперативной памяти ПК 8-16 Гбайт) с обработкой данных на центральном процессоре, но при этом время генерации изображения будет лежать в пределах 30-60 минут.

4.5.1. Функция для генерации изображений с помощью Stable Diffusion

Для начала нужно выполнить установку пакетов с использованием следующей инструкции:

```
pip install --upgrade keras-cv
pip install matplotlib
```

Из библиотеки `keras-cv` будет использована модель `keras_cv.models.StableDiffusion()`.

Итак, напомним программный модуль для генерации изображения по его текстовому описанию (листинг 4.21).

Листинг 4.21. Программный код модуля `Stable_Diff.py`

Примечание.

В демо-версии книги удалены листинги программ.

В начале программы выполнен импорт библиотек `keras_cv` и `matplotlib`. Первая нужна для генерации изображения, вторую будет использована для отображения итогового изображения и записи его на диск.

Затем создана функция `made_images`, которая принимает следующие параметры:

- ❑ `text` – текстовая строка, описывающая создаваемое изображение;
- ❑ `out_img` – путь, по которому сформированное изображение будет записано на диск;
- ❑ `n` – количество вариантов изображений, которые сгенерирует нейронная сеть.

В теле функции создан объект `model` (на основе класса `StableDiffusion`), которому переданы размеры создаваемого изображения. Затем для созданного объекта `model` вызван метод преобразования текста в изображение (`text_to_image`), которому передано текстовое описание изображения (`text`) и количество вариантов создаваемого изображений (`n`). Сгенерированные изображения будут сохранены в списке `image`.

После этого на основе инструкции `for` организован цикл, в теле которого все сгенерированные изображения сохраняются на диске и выводятся на экран (в данном примере будет сгенерировано всего один вариант изображения).

После завершения кода функции `made_images` содержится программный код, обеспечивающий обращение к данной функции. Здесь задана текстовая переменная с описанием изображения (`txt`), задан путь и имя выводного файла (`path_out_img`), и количество вариантов генерируемых изображений (`n=1`). Текстовое описание изображения должно быть задано на английском языке. В данном примере задан следующий текст – «polar bear in the Arctic» (полярный медведь в Арктике).

Примечание.

Данный модуль не включен в состав исполняемых примеров интерактивной цифровой книги, так как он предназначен для работы на локальном компьютере. Но вы имеете возможность скопировать листинг данной программы, запустить данный программный модуль на выполнение на своем локальном компьютере и использовать в своих приложениях.

Если данный программный код запустить на выполнение на локальном компьютере, то будет получен следующий результат (рис. 4.11).



Рис.4.11. Изображение, сгенерированное нейронной сетью по текстовому описанию

Как видно из данного рисунка, нейронная сеть выдала достаточно качественное изображение.

4.5.2. Генерация изображений с помощью Stable Diffusion в веб приложениях

Теперь рассмотрим, как этот метод генерации изображений по текстовому описанию можно реализовать в веб приложениях на Django.

Для активации модулей генерации изображений нужно создать следующую ссылку (листинг 4.22):

Листинг 4.22. Ссылка для активации модулей дизеринга изображений

Примечание.

В демо-версии книги удалены листинги программ.

По этой ссылке будет активирован следующий маршрут URL (листинг 4.23):

Листинг 4.23. Маршрут `gen_image`

Примечание.

В демо-версии книги удалены листинги программ.

По этому маршруту будет вызвано представление `views.gen_image` (листинг 4.24):

Листинг 4.24. Представление `views.gen_image`

Примечание.

В демо-версии книги удалены листинги программ.

В данном представлении сначала выполнен импорт модуля `Stable_Diff`, это модуль Python с функцией `made_images`, которая была описана в предыдущем разделе. Затем были сформированы пути к папкам и файлам проекта:

- ❑ `my_app = base_dir` - путь к приложению;
- ❑ `path_media = os.path.join(my_app, 'media')` - путь к папке `media`;
- ❑ `path_out_img = os.path.join(path_media, 'images/Rabbit.png')` - путь к обработанному изображению.

Затем сделано текстовое описание генерируемого изображения:

```
txt = "Cute Rabbit, Ultra HD, realistic, futuristic, sharp, octane render, " \
      "photoshopped, photorealistic, soft, pastel, Aesthetic, Magical background"
```

Примечание.

Текстовое описание изображения должно быть задано на английском языке. В данном примере задан следующий текст – «Милый кролик, Ультра HD, реалистичный, футуристический, четкий, рендеринг с октановым числом, фотошоп, фотореалистичный, мягкий, пастельный, Эстетичный, волшебный фон».

При использовании данной технологии можно задать генерирование нескольких альтернативных изображений, но в данном примере задана генерация одного изображения (`n=1`).

После этого сделано обращение к функции `made_image` модуля `Stable_Diff`, в которую передана текстовая переменная и путь к обработанному изображению. В теле данной функции выполняется генерация изображения с записью итогового изображения на диск.

На следующем шаге сформирован URL адрес обработанного изображения (`out_url`).

Затем URL адрес обработанного изображения через контекстную переменную `context` передаются в шаблон `gen_image.html`, который имеет следующий код (листинг 4.25).

Листинг 4.25. Шаблон `gen_image.html`

Примечание.

В демо-версии книги удалены листинги программ.

В этом шаблоне всего один тег – для вывода обработанного изображения.

Примечание.

В интерактивной цифровой книге для запуска данного примера нужно на вкладке «Показ примеров» нажать на кнопку «Примеры главы 4» и активировать ссылку «Генерация изображений с KerasCV».

Если выполнить данный программный модуль, то в окне браузера отобразится следующий результат (рис. 4.12).

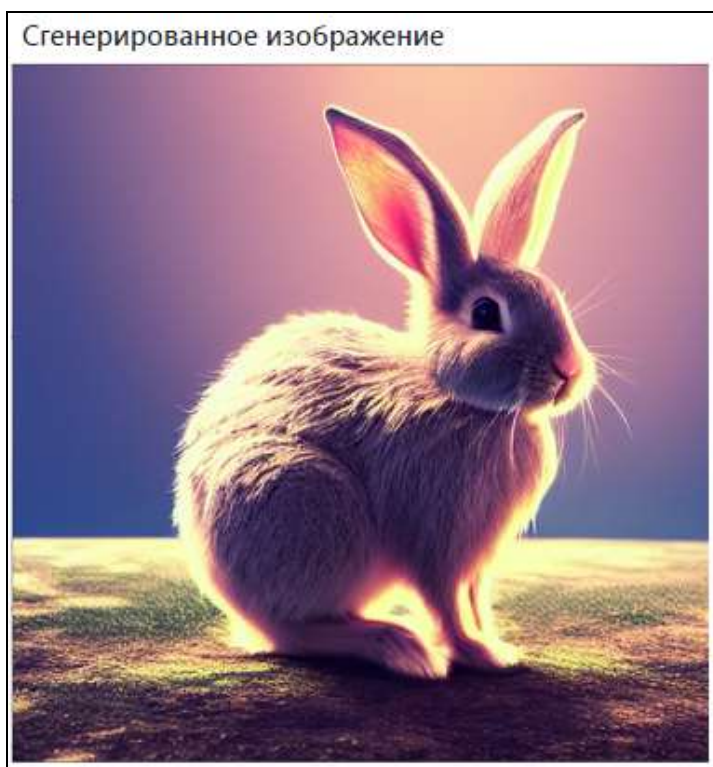


Рис.4.12. Изображение в окне браузера, сгенерированное нейронной сетью по текстовому описанию

Как видно из данного рисунка, нейронная сеть сгенерировало изображение достаточно высокого качества.

Примечание.

Так как генерирование изображения данного примера длится порядка 30 минут, то на публичном сервере строка обращения к функции генерирования изображения закомментирована (для сокращения времени формирования результатов). В данном шаблоне показано заранее созданное данным приложением итоговое изображение по заданному текстовому описанию.

Вам не получится по одному и тому же текстовому запросу получить одинаковые изображения. При каждом обращении к нейронной сети она будет создавать новые изображения.

4.6. Краткие итоги

Итак, мы познакомились с некоторыми базовыми принципами и примерами обработки изображений с использованием нейронных сетей. Теперь познакомимся с методами обработки PDF документов (отображение в шаблонах Django, извлечению текста, рисунков, формирования новых документов и их защиты шифрованием). Рассмотрению этих вопросов и посвящена следующая глава.

ГЛАВА 5

Обработка PDF документов в программах на Python



По мере развития нейронных сетей и элементов искусственного интеллекта, связанных с переводом и озвучиванием текстов, растет и важность технологий извлечения и обработки текстовых данных. В мире существует множество типов документов, содержащих подобные виды неструктурированной информации, от веб-статей и постов в блогах до цифровых книг, инструкций и различной технической документации. Существенная часть этих данных хранится и передается в формате PDF. По данным PDF Association ежегодно в Outlook открывают более двух миллиардов документов в формате PDF, а в Google Drive и электронной почте ежедневно сохраняют 73 миллионов новых PDF файлов.

Исходя из этого, особую актуальность приобретает программное обеспечение для извлечения из подобных документов текстовой информации с последующей ее обработкой, в частности перевод и озвучивание текста на различных языках. Разумеется, главным инструментом для решения подобных задач будет язык программирования Python, набор специализированных библиотек и нейронные сети. Из материалов данной главы вы узнаете:

- ☐ какие библиотеки Python используются для обработки PDF файлов;
- ☐ об основах обработки PDF файлов модулями библиотеки reportlab;
- ☐ как можно выполнить загрузку PDF файлов из представлений в шаблоны Django;
- ☐ как можно создавать PDF файлы в приложениях на Django с библиотекой reportlab;
- ☐ об основах обработки PDF файлов модулями библиотеки PyPDF2;
- ☐ как можно создавать PDF файлы с использованием списков на Python и Django;
- ☐ как можно создавать PDF файлы с использованием класса Canvas;
- ☐ как можно создавать PDF файлы с текстом и рисунками в приложениях на Python;
- ☐ как можно создавать PDF файлы с текстом и рисунками в приложениях на Django;
- ☐ как можно защищать PDF файлы паролем в программах на Python и Django;
- ☐ как можно в приложениях на Django выполнять дешифровку защищенных файлов.

После того, как будут освоены основные приемы обработки PDF файлов, можно будет перейти к технологиям перевода текстов PDF документов и их озвучиванием на различных языках с использованием нейронных сетей. Итак, приступаем к изучению базовых приемов обработки и формирования PDF документов с использованием Python и Django.

5.1. Библиотеки для работы с PDF файлами

Перед началом разработки того или иного программного модуля нужно установить необходимые библиотеки. Будем предполагать, что у вас установлен Python версии 3.10 или более

новая версия. Из множества библиотек, предназначенных для обработки PDF файлов, мы рассмотрим только две - PyPDF2 и reportlab.

PyPDF2 — это бесплатная и открытая библиотека для обработки PDF файлов на Python, способная разделять, объединять, обрезать и преобразовывать страницы PDF-файлов. Она также может добавлять пользовательские данные, параметры просмотра и пароли в PDF-файлы. PyPDF2 также может извлекать текст и метаданные из PDF-файлов. Установить библиотеку PyPDF2 можно с использованием следующей команды:

```
pip install PyPDF2
```

Библиотека PyPDF2 может использоваться для обработки существующих файлов PDF, но она не предназначена для создания новых PDF-документов.

А вот библиотека reportlab позволяет создавать документы в формате Adobe Portable Document Format (PDF) с использованием языка программирования Python. Она также обеспечивает создание таблиц, диаграмм и графиков в различных растровых и векторных форматах, а также в формате PDF. Библиотека reportlab напрямую создает PDF файл на основе набора графических команд. Нет никаких промежуточных шагов. Разработанные приложения могут генерировать отчеты чрезвычайно быстро, иногда на порядки быстрее, чем традиционные инструменты для написания отчетов.

Библиотека reportlab будет полезна как минимум для решения следующих задач:

- динамическая генерация PDF-файлов в локальных и веб приложениях;
- формирование корпоративной отчетности с возможностью взаимодействия с базами данных;
- реализация сборки сложных документов с рисунками, таблицами и текстом (статистические отчеты и научные статьи).

Установить библиотеку reportlab можно с использованием следующей команды:

```
pip install reportlab
```

Для того чтобы определять размеры изображений, которые, встроены в PDF файлы, будет использована еще одна библиотека (imagesize), которую можно установить с использованием следующей команды:

```
import imagesize
```

Теперь можно приступить к реализации модулей, позволяющих обрабатывать и создавать PDF файлы.

5.2. Модуль Python для создания PDF файлов с библиотекой reportlab

Для начала создадим простейший программный модуль, который позволяет создать PDF файл с загрузкой в него одной строки текста и изображения (листинг 5.1).

Листинг 5.1. Модуль pdf_create.py

Примечание.

В демо-версии книги удалены листинги программ.

Для работы данного модуля выполняется импорт следующих пакетов:

```
# Модули библиотеки reportlab
from reportlab.lib.pagesizes import letter
from reportlab.pdfgen import canvas
# Модули для подключения шрифтов с кириллицей
from reportlab.pdfbase import pdfmetrics
from reportlab.pdfbase.ttfonts import TTFont
# Модули для задания путей к файлам
from pathlib import Path
import os
```

На следующем шаге создается ряд объектов, которые содержат пути к файлам приложения:

- ❑ `my_app = Path.cwd()` - путь к папке с приложением;
- ❑ `path_pdf = os.path.join(my_app, 'pdf')` - путь к папке с файлами pdf;
- ❑ `path_img = os.path.join(my_app, 'images')` - путь к папке с изображениями;
- ❑ `out_pdf = os.path.join(path_pdf, 'canvas.pdf')` - путь к создаваемому файлу pdf;
- ❑ `file_img = os.path.join(path_img, 'Men1.jpg')` - путь к файлу с изображением;

Здесь для формирования путей к файлам использованы пакеты `Path` и `os`. Это обеспечит корректное формирование путей к файлам не зависимо от операционной системы.

После этого регистрируем в приложении шрифт (`Arial`), который используется для вывода текста на большинстве европейских языков, включая русский:

```
pdfmetrics.registerFont(TTFont('Arial', 'arial.ttf'))
```

Если этого не сделать, то символы русского языка будут выводиться в виде черных квадратов.

На следующем шаге создается объект «с» (холст, страница) на основе базового класса `Canvas`, которому передаем имя выходного PDF файла (`out_pdf`) и размер страницы (`pagesiz`). Размеры страницы будут соответствовать базовым параметрам страницы `letter` (аналог европейского формата листа бумаги A4):

```
c = canvas.Canvas(out_pdf, pagesize=letter)
width, height = letter
```

В объекты (`width, height`) загружаем размеры страницы (из `letter`). Эти параметры нам пригодятся для обрамления страницы рамкой.

После этого задаются параметры шрифта, которым будет написан текст, формируется текстовая строка, после чего текст и изображения помещаются на холст, то есть на страницу PDF документа:

```
# Тип и размер шрифта
c.setFont("Arial", 12)
# Формируем текстовую строку
text = "Это строка текста в документе формата PDF"
```

```
# Добавление текста в PDF файл
c.drawString(x=100, y=700, text=text)
# Добавление изображения в PDF файл
c.drawImage(file_img, x=100, y=200, width=400, height=400)
```

Здесь для строки с текстовыми данными задан шрифт «Arial» размером 12, сформирована текстовая строка (text), которая помещена на страницу документа с использованием метода drawString. Этому методу переданы координаты начала строки (x, y) и текст выводимой строки (text).

Следует обратить внимание, что в библиотеке reportlab начало координат на холсте отличается от начала координат традиционных графических объектов Canvas языка Python. Если для традиционных графических объектов началом координат является верхний левый угол холста, то у объектов Canvas библиотеки reportlab началом координат является левый нижний угол холста. То есть, по сути, и текстовые строки, и изображения начинают размещаться в координатной сетке страницы PDF документа, начиная с левого нижнего угла (рис. 5.1).

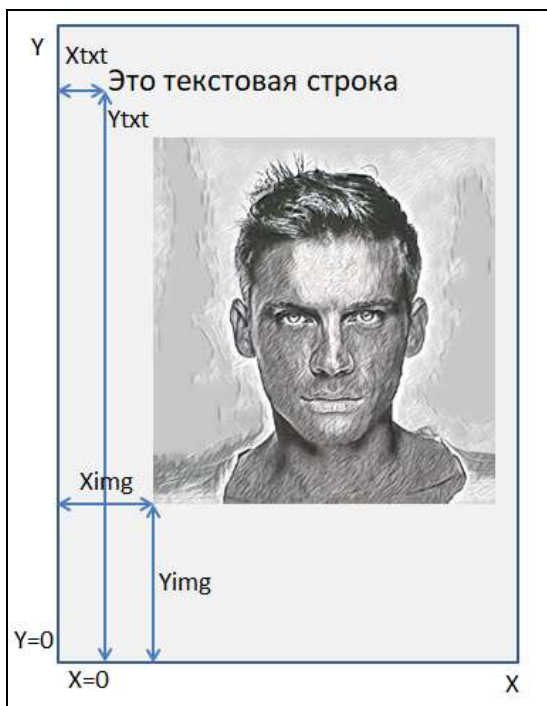


Рис.5.1. Координаты текстовой строки (Xtxt, Ytxt) и изображения (Ximg, Yimg) на холсте

Таким образом, при выводе каждой новой строки нужно уменьшать координату «Y», а не увеличивать ее, а при выводе изображения нужно учитывать размер изображения по высоте.

В приведенном выше фрагменте программного кода текстовая строка выводится в координатах $x=100$, $Y=700$, а изображение в координатах $x=100$, $y=200$, при этом указана высота и ширина рамки, в которую будет вписано изображение.

Затем страница обрамляется рамкой, это делать не обязательно, но в данном примере рамка нарисована для того, чтобы явно показать истинный размер страницы.

```
# Добавление рамки к странице  
c.setStrokeColor("black")  
c.rect(50, 50, width - 100, height - 100)
```

В завершающих строках программы на страницу вставляется признак завершения страницы, и созданный PDF файл записывается на диск.

```
# Задать признак конца страницы  
c.showPage()  
# Сохранить PDF файл  
c.save()
```

Примечание.

Данный модуль не включен в состав исполняемых примеров интерактивной цифровой книги, так как он предназначен для работы на локальном компьютере. Но вы имеете возможность скопировать листинг данной программы, запустить данный программный модуль на выполнение на своем локальном компьютере и использовать в своих приложениях.

Если данный программный код запустить на выполнение на локальном компьютере, то будет получен следующий результат (рис. 5.2).

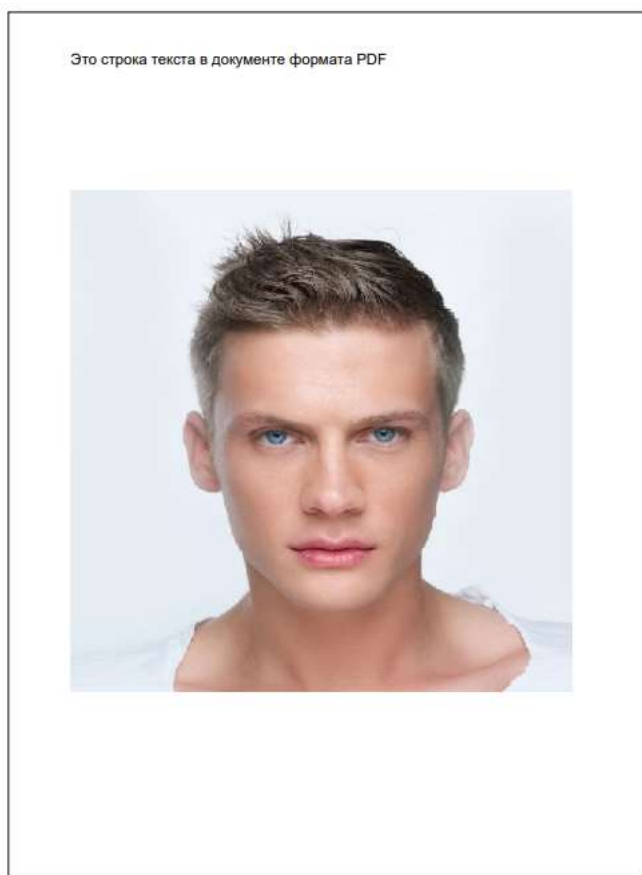


Рис.5.2. Результаты работы программного модуля create_pdf.py

5.3. Загрузка PDF файлов из представлений в шаблоны Django

Достаточно часто возникает необходимость показать содержимое PDF файлов в шаблонах Django вместе с другими элементами. Для этого потребуется преобразовать считанный с диска pdf файл из двоичного формата в строковый объект. В данном разделе будет создан набор программных модулей, которые выполнят такое преобразование

Для начала создадим ссылку, которая будет активировать работу модулей отображения содержимого PDF файлов в шаблонах Django (листинг 5.2).

Листинг 5.2. Ссылка активации модулей вывода содержимого файлов PDF в шаблоны Django

Примечание.

В демо-версии книги удалены листинги программ.

По данной ссылке будет вызван URL маршрут с именем `pdf_template` (листинг 5.3).

Листинг 5.3. URL маршрут с именем `pdf_template`

Примечание.

В демо-версии книги удалены листинги программ.

По данному маршруту будет активировано представление `views.pdf_template` (листинг 5.4).

Листинг 5.4. Представление `pdf_template`

Примечание.

В демо-версии книги удалены листинги программ.

В данном представлении сначала определяется путь к PDF файлу (объект `path_pdf_open`). Затем выполняется чтение и загрузка содержимого PDF файла в объект `pdf`. После этого выполняется преобразование содержимого pdf файла из двоичного формата в строковый — создается объект `pdf_content`, который через контекстный словарь `context` передается в шаблон `pdf_template.html` (листинг 5.5).

Листинг 5.5. Шаблон `pdf_template.html`

Примечание.

В демо-версии книги удалены листинги программ.

В данном шаблоне может присутствовать несколько блоков с выводом различной информации, один из которых будет предназначен для отображения содержимого PDF файла. В данном примере для этой цели используется тег `<embed>`, в котором задана переменная с содержимым PDF файла - `{{pdf_content}}`, а также ширина (`width`) и высота (`height`) области, в которой будет отображаться содержимое PDF файла.

Примечание.

В интерактивной цифровой книге для запуска данного примера нужно на вкладке «Показ примеров» нажать на кнопку «Примеры главы 5» и активировать ссылку «Показать PDF в шаблоне».

Если выполнить приведенные выше программные модули, то будет получен следующий результат (рис. 5.3).

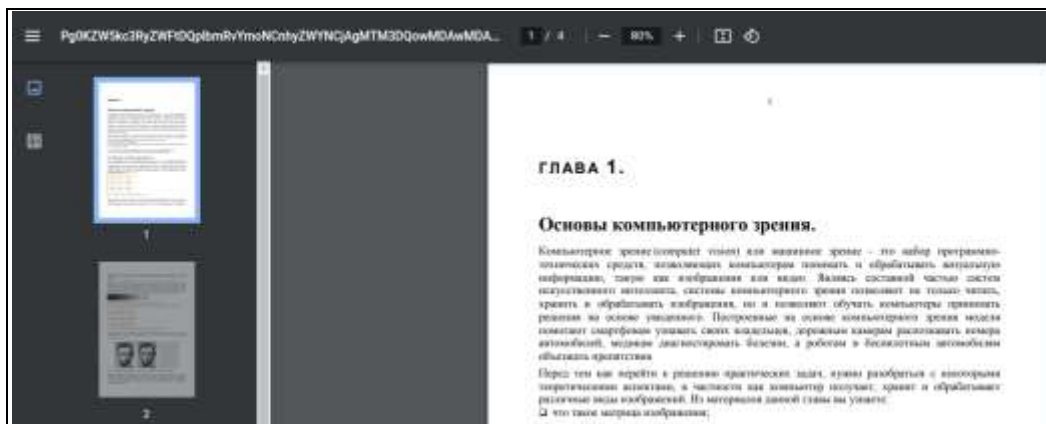


Рис.5.3. Отображение содержимого PDF файла в шаблоне Django

Как видно из данного рисунка, окно для отображения PDF документа состоит из нескольких блоков:

- ☐ верхняя строка с элементами меню;
- ☐ левый блок с макетами страниц;
- ☐ правый блок с содержимым страницы PDF документа.

5.4. Создание PDF файлов в приложениях на Django с библиотекой reportlab

В веб приложениях, написанных на Django, имеется возможность создавать новые PDF файлы с использованием библиотеки reportlab. Данная библиотека позволяет напрямую (в программных модулях) создавать документы в формате PDF с использованием языка программирования Python. Такие документы кроме текста могут содержать изображения, таблицы и графики. Созданные документы можно архивировать, отправлять по электронной почте, размещать в сети Интернет и получать бумажные копии на принтерах. Следует отметить, что формат файлов PDF — это сложный индексированный двоичный формат. При работе с PDF документом невозможно напрямую набирать текст с клавиатуры, как это делается для обычного текстового документа. Описание формата PDF содержит более 600 страниц, и даже один лишний пробел в теле PDF документа может сделать файл не читаемым. К счастью, библиотека reportlab позволяет автоматизировать процесс формирования PDF файлов.

В данном разделе будет создан набор программных модулей, которые выполняют формирование простейшего документа, включающего текст и рисунок.

Для начала создадим ссылку, которая будет активировать работу модулей создания и отображения содержимого PDF файлов в шаблонах Django (листинг 5.6).

Листинг 5.6. Ссылка активации модулей создания файлов PDF

Примечание.

В демо-версии книги удалены листинги программ.

По данной ссылке будет вызван URL маршрут с именем `pdf_new` (листинг 5.7).

Листинг 5.7. URL маршрут с именем `pdf_new`

Примечание.

В демо-версии книги удалены листинги программ.

По данному маршруту будет активировано представление `views.pdf_new` (листинг 5.8).

Листинг 5.8. Представление `pdf_new`

Примечание.

В демо-версии книги удалены листинги программ.

В данном представлении сначала выполнен импорт необходимых библиотек, включая отдельный программный модуль, в котором создается PDF файл (`pdf_create`). Здесь сознательно код формирования PDF файла выделен в отдельный модуль, который можно изменять для формирования документов с различным содержанием, не зависимо от кода представления.

На следующем шаге создано два объекта: `out_file` — путь и имя создаваемого PDF файла, `in_image` - путь и имя изображения, которое будет добавлено в итоговый PDF файл. После этого делается обращение к функции `fn_pdf_create`, в которую передаются объекты `out_file` и `in_image`.

Программный код модуля `pdf_create.py` представлен в листинге 5.9.

Листинг 5.9. Код модуля `pdf_create.py`

Примечание.

В демо-версии книги удалены листинги программ.

В данном программном модуле выполняется импорт необходимых библиотек и подключение шрифтов для формирования текстовых строк. Затем в теле функции создается объект `canvas` «с», задаются параметры страницы, определяются ее размеры ширина (`width`) и высота (`height`), а также задается тип и размер шрифта (`c.setFont`). После этого выполняется формирование содержимого документа. В документ добавляется две строки текста, рисунок, рисуется рамка вокруг страницы, сформированный документ сохраняется на диске.

В представлении `pdf_new` после возврата из функции формирования PDF файла выполняется открытие сформированного PDF документа и его загрузка в объект `pdf`. После этого содержимое PDF файла преобразуется из двоичного формата в текстовый (объект `pdf_content`), и уже в этом виде через контекстную переменную `context` передается в шаблон `pdf_new.html` (листинг 5.10).

Листинг 5.10. Шаблон `pdf_new.html`

Примечание.

В демо-версии книги удалены листинги программ.

В данном шаблоне для отображения содержимого PDF файла используется тег `<embed>`, в котором находится переменная с содержимым созданного документа - `{{pdf_content}}`, а также задана ширина (width) и высота (height) области, в которой будет отображаться содержимое PDF файла.

Примечание.

В интерактивной цифровой книге для запуска данного примера нужно на вкладке «Показ примеров» нажать на кнопку «Примеры главы 5» и активировать ссылку «Создать новый PDF файл».

Если запустить приведенные выше программные модули на выполнение, то будет получен следующий результат (рис. 5.4).

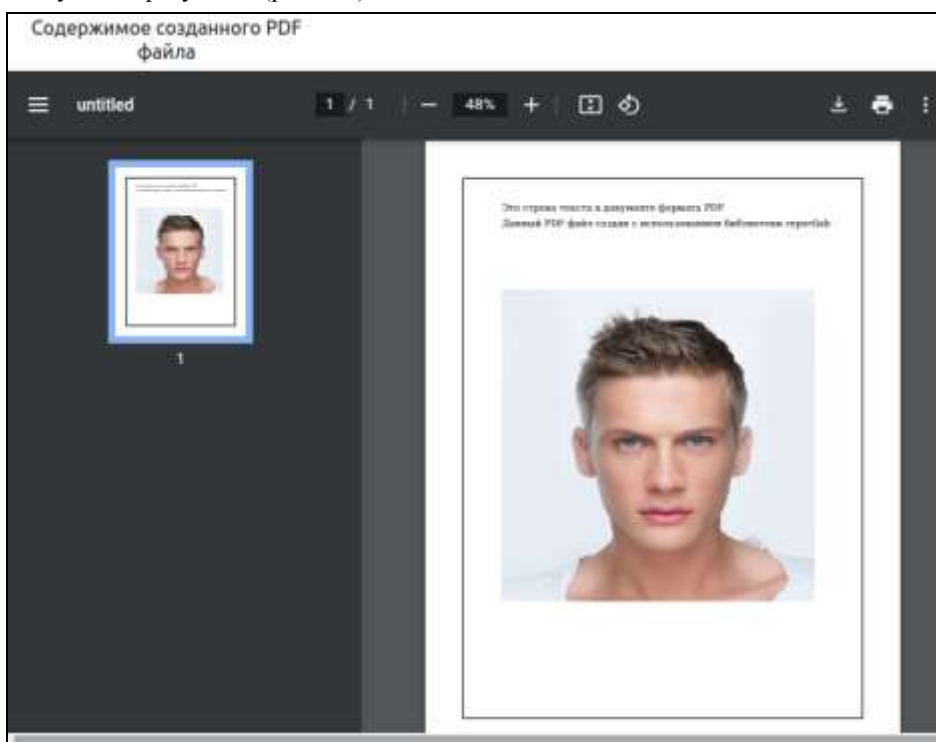


Рис.5.4. Содержимое созданного PDF файла в окне браузера

Как видно из данного рисунка, модули библиотеки `reportlab` успешно создали PDF файл, он был сохранен на диске и отображен в окне браузера.

5.5. Модуль Python для чтения PDF файлов с библиотекой PyPDF2

Рассмотрим некоторые особенности работы с данной библиотекой. В библиотеке `PyPDF2` для чтения содержимого PDF файлов используется метод `PdfReader`. Известно, что в Python все открытые файлы должны быть закрыты до завершения работы кода программы. Объект, созданный на основе `PyPDF2.PdfReader`, сделает это самостоятельно (по умолчанию), поэтому в дальнейшем программист может не беспокоиться об открытии или закрытии PDF

файла. У каждого объекта типа PdfReader есть атрибуты pages, который вы можете использовать для перебора всех страниц в PDF документе по порядку, а количество страниц в документе можно определить на основе функции len(pages).

Примечание.

В данном примере мы используем PyPDF2 версии 3.0.1. Если вы используете данную библиотеку более ранних версий, то в них набор методов и их параметры могут отличаться от тех, которые приведены в данном разделе.

Для начала создадим простейший программный модуль, который позволяет открыть PDF файл и извлечь из него текст и изображения (листинг 5.11).

Листинг 5.11. Модуль pdf_read.py

Примечание.

В демо-версии книги удалены листинги программ.

Для работы данного модуля выполняется импорт следующих пакетов:

```
from PyPDF2 import PdfReader
# Для задания путей к файлам
from pathlib import Path
import os
```

На следующем шаге создается ряд объектов, которые содержат пути к файлам приложения:

- ❑ my_app = Path.cwd() - путь к папке с приложением;
- ❑ path_pdf = os.path.join(my_app, 'pdf') - путь к папке с файлами pdf;
- ❑ path_img = os.path.join(my_app, 'images') - путь к папке с изображениями;
- ❑ input_pdf = os.path.join(path_pdf, 'test.pdf') - путь к исходному файлу pdf.

Здесь для формирования путей к файлам использованы пакеты Path и os. Это обеспечит корректное формирование путей к файлам не зависимо от операционной системы.

После этого происходит открытие PDF файла (объект pdf_file) и его чтение в объект pdf:

```
pdf_file = open(input_pdf, "rb")
pdf = PdfReader(pdf_file)
```

Теперь можно получить общую информацию о PDF файле с использованием метода - pdf.metadata, и определить количество страниц в pdf файле - num_pages = len(pdf.pages).

На следующем шаге с использованием инструкции for организован цикл, в теле которого выполняется постраничный разбор pdf файла: объект page_txt получает текстовое содержание страницы, а объект img_data – данные об изображениях, имеющихся на странице. Найденные на странице изображения сохраняются на диске в текущей директории под именами image_<номер страницы>_<номер изображения на странице>. Например:

- image_1_0.jpg – это изображение на первой странице под номером 1;
- image_2_0.jpg – это изображение на второй странице под номером 1;
- image_2_1.jpg – это изображение на второй странице под номером 2.

Нужно иметь в виду, что нумерация изображений начинается с индекса – 0. Для поиска изображений на каждой странице используется следующая инструкция:

```
for i, image in enumerate(page.images):
```

Здесь с использованием инструкции `for` организован цикл, в теле которого происходит запись найденных изображений на диск. Наличие изображений на странице pdf файла определяется с помощью метода - `page.images`, который передается функции `enumerate`.

Примечание.

Функция `enumerate()` — это встроенная функция в Python, которая преобразует итерируемый объект в перечисляемый одноразовый объект. Когда итерируемый объект обрабатывается функцией `enumerate()`, то каждому его элементу присваивается порядковый номер (индекс). Это упрощает доступ к любому нужному элементу объекта.

То есть, здесь с использованием функция `enumerate()` на каждой странице определяется количество рисунков, каждый рисунок получает индекс (начиная с нуля), а инструкция `for` организует цикл перебора всех рисунков.

Примечание.

Данный модуль не включен в состав исполняемых примеров интерактивной цифровой книги, так как он предназначен для работы на локальном компьютере. Но вы имеете возможность скопировать листинг данной программы, запустить данный программный модуль на выполнение на своем локальном компьютере и использовать в своих приложениях.

Если данный программный код запустить на выполнение на локальном компьютере, то будет получена следующая информация о PDF файле:

```
{'/Author': 'Anatoly', '/Creator': 'Microsoft® Word 2010', '/CreationDate':  
"D:20250129103102+00'00'", '/ModDate': "D:20250129103102+00'00'", '/Producer': 'Microsoft®  
Word 2010'}  
-----  
num_pages= 2
```

Здесь содержатся данные об авторе, который создал PDF файл, дате его создании, инструментарии, с помощью которого он создан, и количестве страниц (две).

Затем будет выдано текстовое содержание файла в разрезе каждой страницы:

```
----- Текущая страница- 0 -----
```

Воробей

Если присмотреться к воробьям, то можно заметить, что это непоседы. Для них трудно устроить фотосессию, так как они постоянно находятся в движении, и фотографироваться не хотят совершенно. Воробьи живут рядом с нами, а много ли мы о них знаем? Думаю, вам будет интересно узнать о них некоторые любопытные факты.

Воробьи не могут долго находиться в воздухе. Птички быстро устают, и им необходимо приземлиться, чтобы передохнуть. Это не самая выносливая птичка. Рядом с человеком воробьи живут уже более 10000 лет! Цифра впечатляет, не правда ли? Природное любопытство птички известно многим. А ещё воробьи достаточно легко приручаемы, они хорошо идут на контакт. Но это относится к выкормленным человеком птицам, взрослые особи в неволе, как правило, чахнут — наверно от тоски по небу и свободе.

Как выдумаете, как воробушки принимают ванну, и чистят свои перышки? Летом птички чистятся, купаясь в песке, а зимой - в снегу. Впрочем, и просто в воде воробьи тоже любят поплескаться. Мойдодыр был бы ими доволен.

Воробей – это птичка -невеличка. Вес взрослой особи редко достигает больше тридцати граммов, а некоторых видов и вовсе - 10-12 граммов (совсем крошечки). Без пропитания воробьи не протянут больше двух суток, так как в полете они тратят много энергии.

Рис.1. Воробей

Большинство животных и птиц видят мир в черно-белых тонах (немного грустно, не правда ли?), но воробьи в этом плане уникальны. Эти птицы видят мир в позитивном розовом цвете!

Воробей очень горяч! Абсолютно нормальная, средняя температура тела птички - 44 градуса. Человек с такой температурой – просто не жилец. В зимнее время, чтобы не замерзнуть, воробьи сбиваются в кучки, это позволяет им сохранять тепло. Наверно с такой-то температурой тела у них это неплохо получается.

Некоторые птички ходят по земле. А вот воробей ходить не может - лапки слишком короткие. Поэтому передвигается он исключительно прыжками!

Никогда не пугайте воробьев! Эта пугливая и хрупкая птичка может умереть от разрыва сердца, если сильно испугается.

----- Текущая страница- 1 -----

Любовь совсем не чужда воробьям! Как и многие другие виды птиц, они часто формируют пару на всю жизнь. Яйца птички откладывают 2-3 раза в год. В среднем в кладке 5-7 яиц, но бывает и больше. Мама-воробиха высиживает их 11-13 дней. А уже через 10 дней, после того, как маленькие воробушки вылупятся, они уже вылетают из гнезда!

Организм воробья рассчитан на достаточно долгую жизнь - около 10 лет. Но на практике в суровых условиях воробьи редко переживают и одну зиму ... Это очень грустно!

В России обитает всего два вида воробьев. Наиболее нам знакомый - домовый, и ещё есть полевой. А всего в мире насчитывается 27 видов. В мире живет около миллиарда воробьев. Немало!

Вспомним про поучительный факт из истории, как жестокость по отношению к воробьям чуть не привела к голоду. В 1958 году в Китае по инициативе небезызвестного Мао Цзэдуна было принято решение массово истреблять воробьев. Китайцы пугали птиц, а так как те не могли долго находиться в воздухе, то они просто падали на землю уже бездыханными... Таким варварским методом были убиты миллиарды воробьев! В этот же год китайцы получили ожидаемый результат в виде большего урожая. Но... Его немилосердно пожрала саранча с гусеницами, которыми раньше питались те самые воробьи, уничтожаемые китайцами. В тот год в страну пришел настоящий голод! А после туда намеренно завозили воробьев из СССР, чтобы всё исправить.

В Бостоне имеется памятник воробью, за уничтожение вредителей! Это самый первый памятник этой птичке. Также памятник воробью есть в городе Ульм, Германия (по легенде, воробей помог при строительстве города, указав своей смекалкой, на примете тростинки, как заносить бревна). Также воробушков можно найти в Токио и в Сингапуре. Ещё памятник воробью располагается в городе Барановичи в Беларуси. Также памятник имеется и в Минске. В России самый знаменитый памятник воробью -

это Чижик -Пыжик на Фонтанке в Санкт -Петербурге. А ещё воробей поселился в Кемерово!

Рис.2. Воробей на поляне

В сильные холода воробей пользуется веществом из своей копчиковой железы, которая вырабатывает жир . Этот жир птичка клювом распределяет по перьям, и так утепляется.

Обратите внимание, что в тексте имеются ошибки (внутри некоторых слов имеются лишние пробелы), но в целом данная библиотека показала неплохой результат.

Примечание.

Несколько возможных причин, по которым при разборе текста с помощью PyPDF2 некоторые слова могут иметь лишний пробел. Возможно, что имеется неправильное представление некоторых слов в PDF-файле. В этом формате не всегда слова представлены как отдельные элементы, чаще это отдельные символы с указанием абсолютного положения каждого символа в документе. В строках этого формата могут находиться спецсимволы (это, по сути, пробелы, но нулевой видимой длины) которые при разборе строк могут быть восприняты как реальные пробелы. Чтобы решить проблему, можно попробовать использовать альтернативную библиотеку, например Textract, которая обеспечивает более точное извлечение текста, в том числе с правильной обработкой пробелов.

Если теперь заглянуть в папку с приложением, то там появятся два рисунка с именами image_1_0.jpg и image_2_0.jpg (рис. 5.5).



Рис.5.5. Изображения, извлеченные из файла test.pdf

5.6. Модуль Python для формирования многостраничного файла PDF с использованием списков

При переводе текста PDF файла с одного языка на другой может возникнуть ситуация, когда количество страниц после перевода будет совершенно иным. Например, при переводе

текста с английского или русского языка на китайский язык количество страниц может сократиться более чем в два раза. В этом случае, если сохранить количество страниц в исходном и переведенном тексте, то каждая страница будет заполнена только наполовину. В этом случае целесообразно загрузить весь текст исходного файла в объект, а потом из этого объекта сформировать выходной файл с новым количеством страниц. При этом для формирования строк выходного файла могут быть использованы стили, отличные от стилей исходного файла.

Создадим программный модуль, который позволяет извлечь текстовое содержимое исходного pdf файла, а потом построчно выгрузить это содержимое в другой PDF файл (листинг 5.12).

Листинг 5.12. Модуль pdf_many_pages.py

Примечание.

В демо-версии книги удалены листинги программ.

Для работы данного модуля выполняется импорт необходимых модулей из библиотек reportlab и PyPDF2. Затем с использованием модулей библиотек Path и os прописываются пути к исходному (input_pdf) и выходному (out_pdf) PDF файлу.

После этого с использованием класса PdfReader в объект «pdf» загружается содержимое исходного PDF файла и определяется количество страниц (num_pages). Затем с использованием инструкции for организован цикл, в теле которого создается объект page (страница), считывается содержимое каждой страницы и заносится в объект long_text. Таким образом, по завершению цикла объект long_text будет содержать всю текстовую информацию исходного PDF файла. После этого с полученной текстовой информацией можно выполнять различные манипуляции (например, озвучить, перевести на другой язык, сохранить в текстовом файле и т.п.).

В данном программном модуле полученный текст будет сохранен в другом файле PDF с заменой параметров строк. Для этого далее создан объект doc с использованием класса SimpleDocTemplate. Этому классу переданы параметры: путь и имя выходного файла (out_path), размер страницы документа (letter – американский аналог формата бумаги A4). Затем создается объект style, в который загружается стиль для всех строк текста выходного PDF файл.

На следующем шаге с использованием инструкции for организован цикл, в теле которого построчно формируется содержимое выходного файла. Здесь перед входом в цикл создается пустой список (объект elements), который в теле цикла заполняется строками из объекта paragraph с использованием стиля (style) для каждой строки. Количество строк в тексте определяется на основе метода split объекта long_text. Признаком конца строки являются служебные символы «\n». Наконец в последней строке этого модуля выходной файл сохраняется на диске с использованием инструкции doc.build(elements).

Фрагмент текста исходного PDF файла представлен на рис. 5.6.

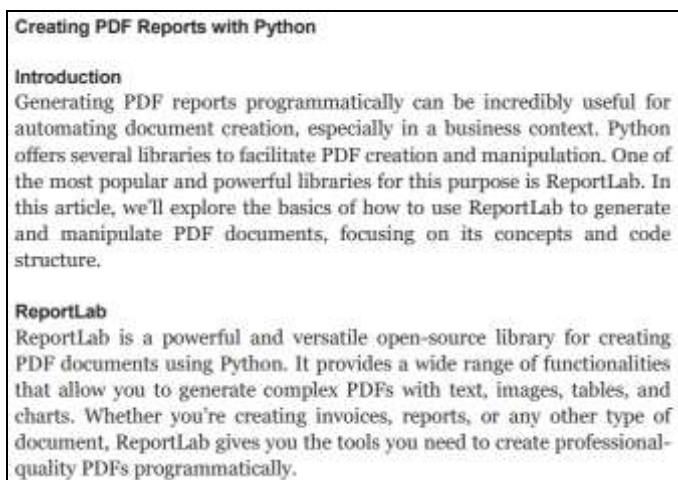


Рис.5.6. Фрагмент текста исходного файла txt_en.pdf

Как видно из данного рисунка, этот файл содержит текст на английском языке. В качестве примера здесь взят файл с англоязычным текстом, так как в примерах следующих разделах он будет использован для перевода текста на другие языки, а также для озвучивания текстовых файлов на разных языках.

Примечание.

Данный модуль не включен в состав исполняемых примеров интерактивной цифровой книги, так как он предназначен для работы на локальном компьютере. Но вы имеете возможность скопировать листинг данной программы, запустить данный программный модуль на выполнение на своем локальном компьютере и использовать в своих приложениях.

Если данный программный код запустить на выполнение на локальном компьютере, то мы получим новый PDF файл, в котором будет присутствовать тот же текст, но с другими параметрами строк (рис. 5.7).



Рис.5.7. Фрагмент текста выходного файла many_pages.pdf

Как видно из приведенных выше рисунков, исходный и выходной файла имеют строки с одинаковым содержанием, но с разными размерами шрифтов и разными интервалами между строк. Для сервисов, которые будут обеспечивать перевод и озвучивание текста, стиль написания строк не имеет принципиального значения. Важно то, что используемые библиотеки успешно справились с поставленной задачей.

5.7. Формирование многостраничного файла PDF с использованием списков в приложениях на Django

При переводе содержимого PDF файла с одного языка на другой язык требуется, и считать содержимое исходного файла, и создавать новый файл с результатами перевода. Использование API для перевода текстов на основе искусственного интеллекта будет рассмотрено в следующей главе, а в данном разделе создадим группу программных модулей веб приложения на Django, которые позволяют извлечь текстовое содержимое исходного pdf файла, а потом построчно выгрузить это содержимое в другой PDF файл (листинг 5.13).

Листинг 5.13. Ссылка на модули формирования многостраничных PDF файлов

Примечание.

В демо-версии книги удалены листинги программ.

По этой ссылке будет активирован URL маршрут с именем `pdf_new_many_pg` (листинг 5.14).

Листинг 5.14. Маршрут с именем `pdf_many_pg`

Примечание.

В демо-версии книги удалены листинги программ.

По этому маршруту будет активировано представление `views.pdf_new_many_pg` (листинг 5.15).

Листинг 5.15. Представление `pdf_new_many_pg`

Примечание.

В демо-версии книги удалены листинги программ.

В данном представлении сначала выполнен импорт необходимых библиотек, включая отдельный программный модуль, в котором создается PDF файл (`pdf_many_pages`). Здесь сознательно код формирования PDF файла выделен в отдельный модуль, который можно изменять для формирования документов с различным содержанием, не зависимо от кода представления.

На следующем шаге создано два объекта: `inp_file` — путь и имя исходного PDF файла, `out_file` — путь и имя создаваемого PDF файла. После этого делается обращение к функции `fn_pdf_pages` (модуля `pdf_many_pages.py`), в которую передаются объекты `inp_file` и `out_file`. Программный код модуля `pdf_many_pages.py` представлен в листинге 5.16.

Листинг 5.16. Код модуля `pdf_many_pages.py`

Примечание.

В демо-версии книги удалены листинги программ.

В данном программном модуле выполняется импорт необходимых библиотек и подключение шрифтов для формирования текстовых строк. Затем в теле функции открывается исходный файл, его содержимое загружается в объект `pdf` и определяется количество страниц в документе (`num_pages`). После этого с использованием инструкции `for` организован цикл, в теле которого в объект `long_text` происходит постраничная загрузка текста из исходного документа.

На следующем шаге создается объект `doc` (новый документ) и задается стиль для текста в данном документе (`style`). После этого выполняется формирование содержимого нового документа путем построчного дополнения строк в список (объект `elements`) в теле инструкции `for`. По завершению цикла выполняется запись сформированного документа на диск.

В представлении `pdf_new_many_pg` после возврата из функции формирования PDF файла выполняется открытие сформированного PDF документа и его загрузка в объект `pdf`. После этого содержимое PDF файла преобразуется из двоичного формата в текстовый (объект `pdf_content`), и уже в этом виде через контекстную переменную `context` передается в шаблон `many_pages.html` (листинг 5.17).

Листинг 5.17. Шаблон many_pages.html

Примечание.

В демо-версии книги удалены листинги программ.

В данном шаблоне для отображения содержимого PDF файла используется тег `<embed>`, в котором находится переменная с содержимым созданного документа - `{{pdf_content}}`, а также задана ширина (width) и высота (height) области, в которой будет отображаться содержимое PDF файла.

Примечание.

В интерактивной цифровой книге для запуска данного примера нужно на вкладке «Показ примеров» нажать на кнопку «Примеры главы 5» и активировать ссылку «Создать многостраничный PDF файл».

Если запустить приведенные выше программные модули на выполнение, то будет получен следующий результат (рис. 5.8).

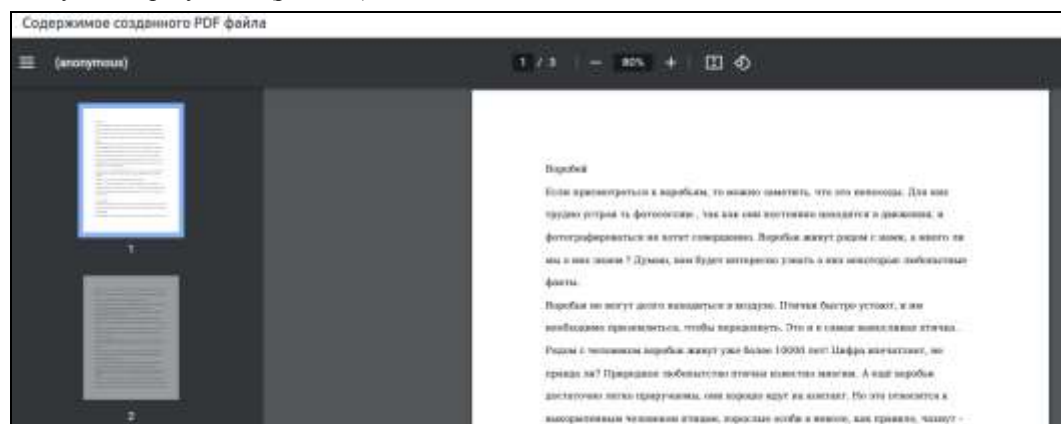


Рис.5.8. Фрагмент содержимое созданного PDF файла в окне браузера

Как видно из данного рисунка, модули библиотеки `reportlab` успешно создали многостраничный PDF файл, он был сохранен на диске и отображен в окне браузера.

Если теперь загрузить англоязычный текст, то мы получим новый документ с тестом на английском языке (рис. 5.9.).

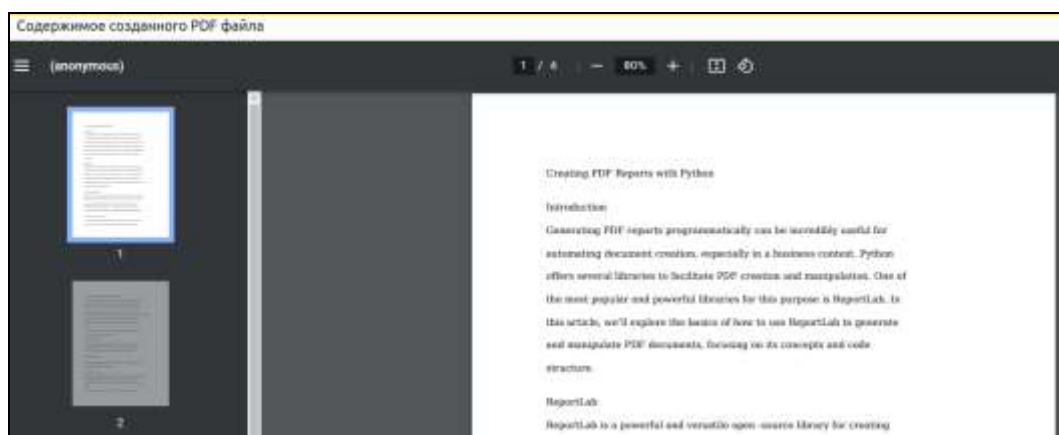


Рис.5.9. Фрагмент содержимое созданного PDF файла в окне браузера (английский язык).

Как видно из этих рисунков, используемый в данном приложении шрифт DejaVuSerif поддерживает вывод текста на латинице и на кириллице.

5.8. Модуль Python для формирования многостраничного файла PDF с использованием класса Canvas

Есть еще один способ формирования выходных PDF файлов с использованием класса Canvas (холст) пакета pdfgen. По своей сути объект, созданный на основе класса Canvas, это страница заданного размера, на которой можно размещать строки текста, изображения, таблицы и другие элементы документа. При создании экземпляра класса Canvas можно изменить размер страницы с помощью необязательного параметра `pagesize`. Этот параметр принимает кортеж из двух чисел типа `float`, представляющих ширину и высоту страницы в точках.

Например, чтобы установить размер страницы 8,5 дюймов в ширину и 11 дюймов в высоту, нужно создать следующий объект `canvas`:

```
canvas = Canvas("file.pdf", pagesize=(612.0, 792.0))
```

Здесь ширина `w=612` точек (или `72*8,5` дюйма), а высота `h=792` (или `72*11` дюймов). Здесь число `72` означает количество точек в одном дюйме. Если оперировать сантиметрами, то нужно использовать другое число – количество точек в одном сантиметре (`28.346456692913385`). Работать с такими длинными числами не совсем удобно, поэтому в библиотеке `reportlab` есть соответствующие константы `inch` – дюймы и `cm` – сантиметры. Эти константы можно импортировать с использованием следующей инструкции:

```
from reportlab.lib.units import inch, cm
```

С использованием этих констант можно установить размер страницы 8,5 дюймов в ширину и 11 дюймов в высоту с использованием следующей инструкции:

```
canvas = Canvas("file.pdf", pagesize=(8.5*inch, 11*inch))
```

С использованием встроенных констант пользователь может задать любой размер страницы. Однако если нужно использовать традиционный размер страницы, в пакете reportlab есть несколько стандартных встроенных размеров, с которыми проще работать. Для этого нужно импортировать следующий модуль:

```
from reportlab.lib.pagesizes import letter, A4
```

Здесь объект letter – это стандартный американский размер листа бумаги, A4 - стандартный европейский размер листа бумаги. С использованием этих объектов можно установить стандартный размер страницы с использованием следующих инструкций:

```
canvas = Canvas("file.pdf", pagesize= letter)  
canvas = Canvas("file.pdf", pagesize=A4)
```

Создадим программный модуль, который позволяет извлечь текстовое содержимое исходного pdf файла, а потом построчно выгрузить это содержимое в другой PDF файл с использованием класса Canvas (листинг 5.18).

Листинг 5.18. Модуль pdf_many_pg_can.py

Примечание.

В демо-версии книги удалены листинги программ.

Для работы данного модуля выполняется импорт необходимых пакетов из библиотек reportlab и PyPDF2. Затем с использованием модулей библиотек Path и os прописываются пути к исходному (input_pdf) и выходному (out_pdf) PDF файлу.

После этого с использованием класса PdfReader в объект «pdf» загружается содержимое исходного PDF файла и определяется количество страниц (num_pages). Затем с использованием инструкции for организован цикл, в теле которого создается объект page (страница), считывается содержимое каждой страницы и заносится в объект long_text. Таким образом, по завершению цикла объект long_text будет содержать всю текстовую информацию исходного PDF файла. После этого с полученной текстовой информацией можно выполнять различные манипуляции (например, озвучить, перевести на другой язык, сохранить в текстовом файле и т.п.).

В данном программном модуле полученный текст будет сохранен в другом файле PDF с заменой параметров строк. Для этого далее создан объект styles, в который загружены стандартные параметры строк класса SimpleDocTemplate.

На следующем шаге формируются параметры страницы создаваемого PDF файла (размеры страницы, отступы от края, полезное поле для вывода строк), и определяется начально положение первой строки текста (координаты x и y). После этого текст, полученный из исходного PDF файла, разбивается на строки (объект paragraphs), и создается холст (объект «c» на основе класса Canvas).

После этого на основе инструкции for создан цикл по строкам текста, в теле которого очередная страница заполняется строками текста. Здесь каждая строка загружается в объект paragraph, вписывается в ширину страницы и путем вызова метода drawOn помещается на холст с использованием следующей инструкции:

```
paragraph.drawOn(c, x, y - height)
```

Здесь в метод drawOn переданы следующие параметры:

- с - холст (canvas) или страница;
- x – координата «х» положения начала строки;
- у - координата «у» положения начала строки;
- height – полезная высота страницы в точках.

После вывода очередной строки происходит уменьшение координаты вывода следующей строки (координата «у» уменьшается на 12 точек).

Перед выводом каждой новой строки с использованием инструкции if проверяется условие – помещается ли текущая строка на страницу. Если строка помещается, то выполняется ее вывод, если не помещается, то создается новая страница - инструкция c.showPage(), и задается начальное положение строки по оси «у» на новой странице. По завершению цикла выполняется запись созданного PDF файла на диск - инструкция c.save().

Примечание.

Данный модуль не включен в состав исполняемых примеров интерактивной цифровой книги, так как он предназначен для работы на локальном компьютере. Но вы имеете возможность скопировать листинг данной программы, запустить данный программный модуль на выполнение на своем локальном компьютере и использовать в своих приложениях.

Если запустить данный модуль на выполнение на локальном компьютере, то мы получим выходной PDF файл, у которого все страницы будут полностью заполнены текстом, правда, в нем будут отсутствовать рисунки, но при этом сохранены некоторые фрагменты стиля. На рис. 5.10 приведен фрагмент страницы исходного файла, на которой имеется маркированный текст.

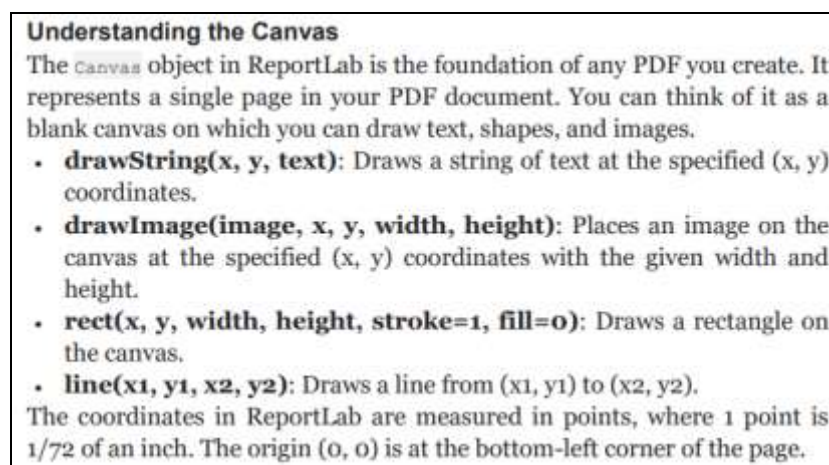


Рис.5.10. Маркированный фрагмент текста в исходном PDF файле

Маркировка этого фрагмента текста сохранилась и в выходном PDF файле (рис. 5.11).

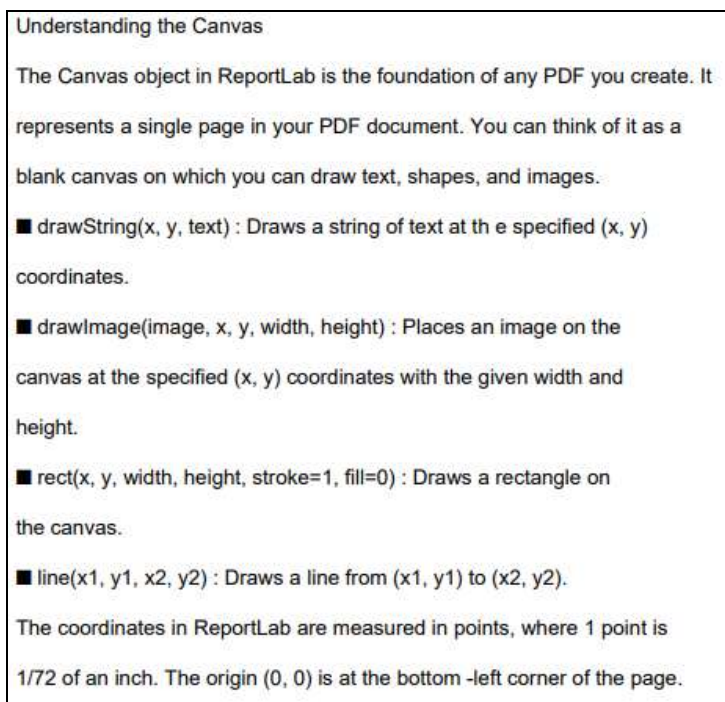


Рис.5.11. Маркированный фрагмент текста в выходном PDF файле

Из вышеприведенных рисунков видно, что параметры шрифта и расстояния между строк в исходном и выходном файле отличаются. Однако поскольку мы в дальнейшем собираемся переводить и озвучивать содержимое PDF файлов, то нам не важны параметры написания текста, важно получение текстового содержания исходного файла и вывод переведенного текста в выходной файл.

В приведенном выше модуле в выходном файле были потеряны рисунки, которые содержались в исходном файле. Теперь рассмотрим возможность переноса из одного файла в другой и текста, и рисунков.

5.9. Модуль Python для формирования многостраничного файла PDF с текстом и рисунками

Достаточно часто требуется, чтобы исходный файл и выходной файл имели одинаковое количество страниц и рисунков, не зависимо от языка, на который текст был переведен. Это важно для веб сервисов, где, например, при озвучивании текстового файла пользователю нужно одновременно показывать текст на языке оригинала и переведенный текст.

Создадим программный модуль, который позволяет извлечь текстовое содержимое исходного pdf файла и рисунки, а потом построчно выгрузить это содержимое в другой PDF файл с сохранением количества страниц (листинг 5.19).

Листинг 5.19. Модуль pdf_many_pg_img.py

Примечание.

В демо-версии книги удалены листинги программ.

Для работы данного модуля выполняется импорт необходимых пакетов и библиотек: `reportlab`, `PyPDF2`, `imagesize` и др. Затем с использованием модулей библиотек `Path` и `os` прописываются пути к исходному (`input_pdf`) и выходному (`out_pdf`) PDF файлу.

После этого с использованием класса `PdfReader` в объект «pdf» загружается содержимое исходного PDF файла и определяется количество страниц (`num_pages`).

В данном программном модуле полученный текст будет сохранен в другом файле PDF с заменой параметров строк. Для этого далее создан объект `styles`, в который загружены стандартные параметры строк класса `SimpleDocTemplate`.

На следующем шаге формируются параметры страницы создаваемого PDF файла (размеры страницы, отступы от края, полезное поле для вывода строк), и определяется начально положение первой строки текста (координаты `x` и `y`). После этого создается холст (объект «с» на основе класса `Canvas`).

После этого на основе инструкции `for` создан цикл по страницам текста исходного файла, в теле которого очередная страница заполняется строками текста. Здесь каждая страница загружается в объект `page`, из которого все строки текста страницы извлекаются в объект `text`. После этого текст текущей страницы разбивается на строки (объект `paragraphs`).

Затем с использованием инструкции `for` создан вложенный цикл по строкам страницы. Каждая новая строка заносится в объект `paragraph`, строка вписывается в ширину страницы и с использованием метода `drawOn` помещается на холст с использованием следующей инструкции:

```
paragraph.drawOn(c, x, y - height)
```

Здесь в метод `drawOn` переданы следующие параметры:

- `c` - холст (`canvas`) или страница;
- `x` – координата «х» положения начала строки;
- `y` - координата «у» положения начала строки;
- `height` – полезная высота страницы в точках.

После вывода очередной строки происходит уменьшение координаты вывода следующей строки (координата «у» уменьшается на величину `height + 3`, где 3 - расстояние между строками).

После вывода всех строк с использованием инструкции `if` делается проверка – есть ли изображения на данной странице. Если изображения есть (а их может быть несколько), то задаются координаты для вывода первого изображения на страницу (`x_fig`, `y_fig`) и, с использованием инструкции `for`, создается цикл обработки всех изображений. Извлеченное из страницы изображение загружается в объект `img_data` и сохраняется на диске под именем:

```
image_<номер страницы>_<номер изображения на странице>/jpg
```

После этого в объектах `w_img`, `h_img` запоминаются ширина и высота изображения. Затем изображение вставляется в текущую страницу с использованием следующей инструкции:

```
c.drawImage(f'image_{page_num + 1}_{i}.jpg', x_fig, y_fig, w_img, h_img)
```

Здесь методу `drawImage` переданы следующие параметры:

- `f'image_{page_num + 1}_{i}.jpg'` – имя файла с изображением;
- `x_fig` – координата «х» нижнего левого угла изображения;
- `y_fig` – координата «у» нижнего левого угла изображения;
- `w_img` – ширина изображения;
- `h_img` – высота изображения.

После того, как изображение вставлено на страницу, координата «у» уменьшается на высоту текущего изображения (она будет использоваться для вывода следующего изображения), и текущее изображение удаляется с диска.

После завершения обработки всех строк и изображений текущей страницы в выходном документе создается новая страница – инструкция `s.showPage()`, а после обработки всех страниц выходной PDF файл записывается на диск – инструкция `s.save()`.

Примечание.

Данный модуль не включен в состав исполняемых примеров интерактивной цифровой книги, так как он предназначен для работы на локальном компьютере. Но вы имеете возможность скопировать листинг данной программы, запустить данный программный модуль на выполнение на своем локальном компьютере и использовать в своих приложениях.

Если запустить данный модуль на выполнение на локальном компьютере, то мы получим выходной PDF файл, с тем же количеством страниц, как и у исходного файла, при этом будет перенесен и текст и рисунки. На рис. 5.12. представлена страница исходного файла с текстом и изображением.

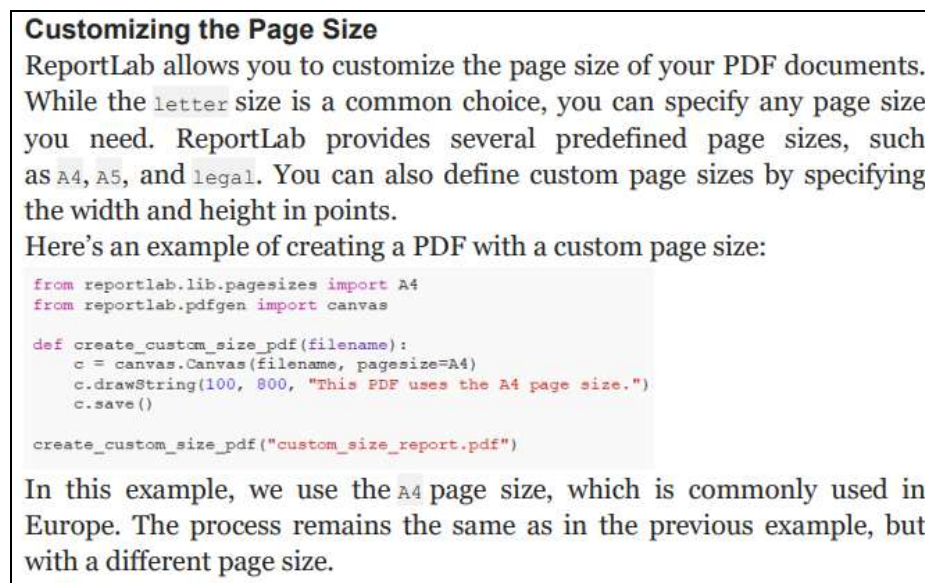


Рис.5.12. Фрагмент текста с изображением в исходном PDF файле

На данном рисунке изображением является листинг программного кода. В документации, которую нужно переводить на другие языки и озвучивать, листинги программ необходимо вставлять в документы в виде изображений, так как их не нужно переводить с английского, и их нет необходимости озвучивать.

На рис. 5.13 представлен фрагмент той же страницы в выходном документе.



Рис.5.13. Фрагмент текста с изображением в выходном PDF файле

Как видно из данного рисунка, изображение на странице имеет некоторое смещение относительно последней строки текста. Но для решения поставленной задачи, сделать перевод документа на другой язык с сохранением количества страниц это смещение не играет решающей роли.

5.10. Формирование многостраничного файла PDF с текстом и рисунками в веб приложениях на Django

Документы формата PDF могут содержать не только текст, но и рисунки. При переводе таких текстов на другие языки требуется, чтобы исходный файл и выходной файл имели одинаковое количество страниц и рисунков, не зависимо от языка, на который текст был переведен. В предыдущем разделе был создан модуль, работающий на Python, который переносит текст и рисунки из одного документа в другой. В этом разделе мы создадим группу программных модулей, которые решают аналогичную задачу в рамках веб приложения на Django.

Программный код ссылки, которая иницирует работу модулей переноса текста и рисунков из одного PDF документа в другой, представлен в листинге (листинг 5.20).

Листинг 5.20. Ссылка на активацию модулей переноса из PDF файлов текста и рисунков

Примечание.

В демо-версии книги удалены листинги программ.

По данной ссылке будет активирован URL маршрут с именем pdf_new_txt_img (листинг 5.21).

Листинг 5.21. URL маршрут PDF pdf_new_txt_img

Примечание.

В демо-версии книги удалены листинги программ.

По данному маршруту будет вызвано представление views.pdf_new_txt_img (листинг 5.22)

Листинг 5.22. Представление pdf_new_txt_img

Примечание.

В демо-версии книги удалены листинги программ.

В данном представлении сначала выполнен импорт необходимых библиотек, включая отдельный программный модуль, в котором создается PDF файл (pdf_many_pg_img). Здесь сознательно код формирования PDF файла выделен в отдельный модуль, который можно изменять для формирования документов с различным содержанием, не зависимо от кода представления.

На следующем шаге создано два объекта: inp_file — путь и имя исходного PDF файла, out_file — путь и имя создаваемого PDF файла. Кроме того, создан объект lng, в котором указан исходный язык PDF документа (lng=«en» - английский язык, lng=«ru» - русский язык). Указание языка необходимо для того, чтобы загрузить нужные шрифты. Если текст документа создан на латинице, то никаких действий предпринимать не нужно (библиотека reportlab поддерживает фонт латиницы по умолчанию). Если же в тексте присутствует кириллица, то необходимо загружать фонт DejaVuSerif, который поддерживает и кириллицу и латиницу.

После этого делается обращение к функции fn_pdf_txt_img (модуля pdf_many_pg_img.py), в которую передаются объекты inp_file, out_file и lng. Программный код модуля pdf_many_pg_img.py представлен в листинге 5.23.

Листинг 5.23. Код модуля pdf_many_pg_img.py

Примечание.

В демо-версии книги удалены листинги программ.

В данном программном модуле выполняется импорт необходимых библиотек и подключение шрифтов для формирования текстовых строк. Затем в теле функции открывается исходный файл, его содержимое загружается в объект pdf, определяется количество страниц в документе (num_pages) и задается стиль документа (style) в зависимости от языка, на котором написан документ. На следующем шаге формируются параметры страницы создаваемого документа, определяются начальные параметры для вывода текста (x, y) и создается холст (или страница) - объект «с».

После этого с использованием инструкции `for` организован цикл по страницам исходного файла, в теле которого извлекается текст очередной страницы (объект `text`), делается разбиение текста страницы на строки (объект `paragraphs`), и вставка текущей строки в новый документ (на холст).

После обработки всех строк страницы делается проверка — есть ли рисунки на текущей странице. Если рисунки присутствуют (а их может быть несколько), то они в цикле (инструкция `if`) добавляются к странице после текста.

После обработки текущей страницы исходного файла формируется новый холст для следующей страницы - инструкция `c.showPage()` и определяется начальное положение строки на новой странице (координата `y`). По завершению обработки всех страниц делается запись сформированного документа на диск - инструкция `c.save()`.

В представлении `pdf_new_txt_img` после возврата из функции формирования PDF файла выполняется открытие сформированного PDF документа и его загрузка в объект `pdf`. После этого содержимое PDF файла преобразуется из двоичного формата в текстовый (объект `pdf_content`), и уже в этом виде через контекстную переменную `context` передается в шаблон `many_pages.html` (листинг 5.24).

Листинг 5.24. Шаблон `many_pages.html`

Примечание.

В демо-версии книги удалены листинги программ.

В данном шаблоне для отображения содержимого PDF файла используется тег `<embed>`, в котором находится переменная с содержимым созданного документа - `{{pdf_content}}`, а также задана ширина (`width`) и высота (`height`) области, в которой будет отображаться содержимое PDF файла.

Примечание.

В интерактивной цифровой книге для запуска данного примера нужно на вкладке «Показ примеров» нажать на кнопку «Примеры главы 5» и активировать ссылку «Перенос текста и рисунков в PDF».

Если запустить приведенные выше программные модули на выполнение, то будет получен следующий результат (рис. 5.14).

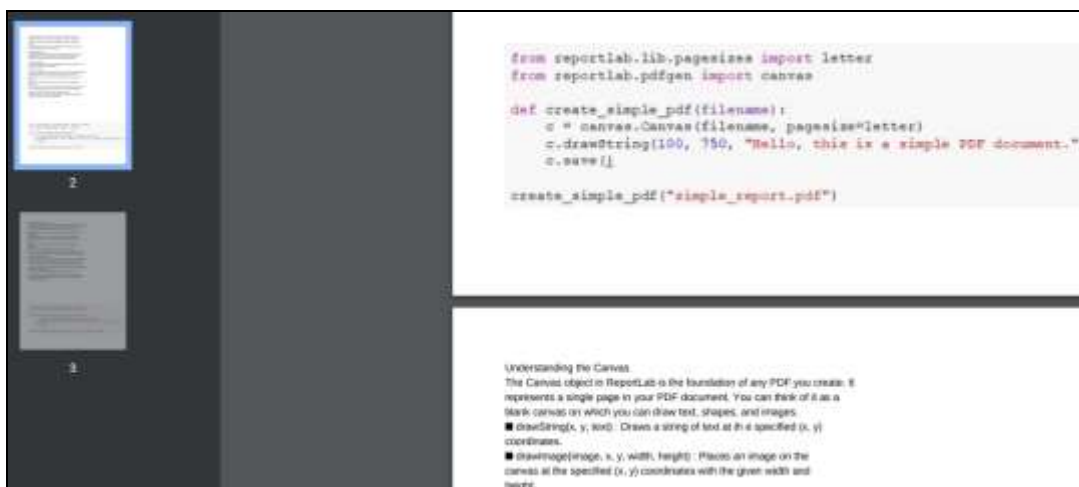


Рис.5.14. Фрагмент содержимое созданного PDF файла в окне браузера (английский язык)

Как видно из данного рисунка, модули библиотеки reportlab успешно создали многостраничный PDF файл с текстом на английском языке и рисунками (здесь в виде рисунка представлен листинг программы).

Если теперь загрузить русскоязычный текст, то мы получим новый документ с текстом на русском языке с корректным отображением символов на кириллице (рис. 5.15.).

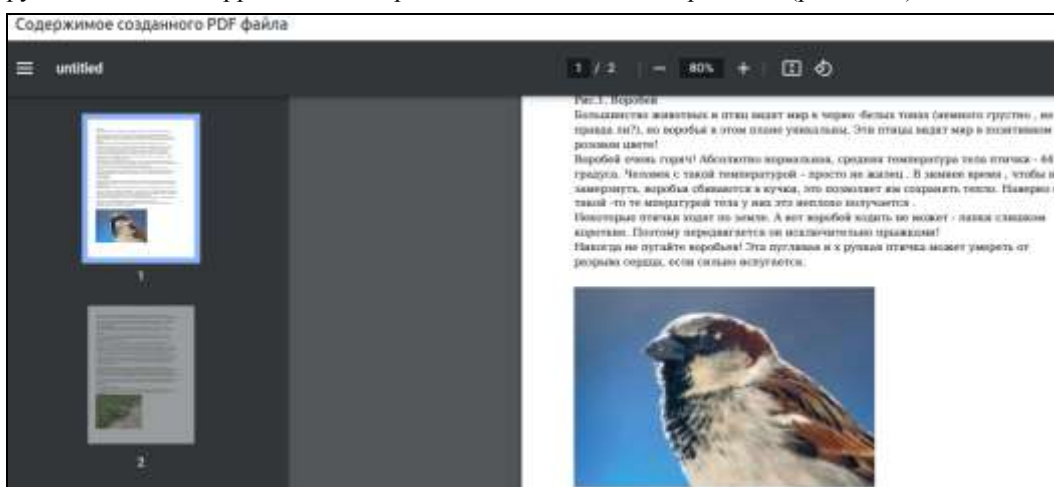


Рис.5.15. Фрагмент содержимое созданного PDF файла в окне браузера (русский язык).

5.11. Модуль Python для защиты файла PDF паролем с библиотекой PyPDF2

Защита файлов PDF необходима для обеспечения доступа к конфиденциальной информации, содержащейся в файле. Доступ к зашифрованным данным в защищенных файлах будут иметь только пользователи, знающие пароль, или программные средства, которые могут дешифровать данные по доступному паролю. Защита PDF файлов паролем добавляет уро-

вень безопасности, особенно для файлов, которые выкладываются в сеть интернет или хранятся в общих сетях. Благодаря обширным библиотекам Python защита файлов PDF с помощью паролей становится простой. В этом разделе рассматривается возможность шифрования PDF файлов с использованием библиотеки PyPDF2.

Создадим программный модуль, который позволяет сначала зашифровать содержимое pdf файла, а потом открыть PDF файл, дешифровать его и извлечь из него текстовую информацию (листинг 5.25).

Листинг 5.25. Модуль pdf_protect.py

Примечание.

В демо-версии книги удалены листинги программ.

Для работы данного модуля выполняется импорт следующих пакетов:

```
from PyPDF2 import PdfReader, PdfWriter
# Для задания путей к файлам
from pathlib import Path
import os
```

На следующем шаге создается ряд объектов, которые содержат пути к файлам приложения:

- ❑ `my_app = Path.cwd()` - путь к папке с приложением;
- ❑ `path_pdf = os.path.join(my_app, 'pdf')` - путь к папке с файлами pdf;
- ❑ `input_pdf = os.path.join(path_pdf, 'test.pdf')` - путь к исходному файлу pdf
- ❑ `input_pdf = os.path.join(path_pdf, 'protect.pdf')` - путь к защищенному файлу pdf.

Здесь для формирования путей к файлам использованы пакеты Path и os. Это обеспечит корректное формирование путей к файлам не зависимо от операционной системы.

После этого происходит открытие PDF файла (объект `pdf_file`) и его чтение в объект `pdf_reader`:

```
pdf_file = open(input_pdf, "rb")
pdf_reader = PdfReader(pdf_file)
```

Затем создается объект для записи PDF файлов и с его помощью выполняется копирование исходного файла в выходной (защищенный):

```
# Создание объекта PdfWriter для записи нового PDF файла
pdf_writer = PdfWriter(out_pdf)
# Скопировать страницы исходного файла в выходной
for page in pdf_reader.pages:
    pdf_writer.add_page(page)
```

На следующем шаге создается объект пароль (`password`), выполняется шифрование (защита) выходного файла и его запись на диск:

```
# Создать пароль
password = '12345'
# Зашифровать PDF файл паролем
pdf_writer.encrypt(user_password=password)
```

```
# Сохранить выходной файл с паролем
with open(out_pdf, mode="wb") as output_file:
    pdf_writer.write(output_file)
```

В нашем случае зашифрованный файл будет сохранен на диске под именем protect.pdf и пользователи смогут открыть его только после ввода пароля. Однако мы сможем открыть его и программным способом, если дешифруем его в данном программном модуле, что и сделано в следующих строках:

```
# Создание объекта PdfReader для чтения PDF файла с паролем
pdf_file = open(out_pdf, "rb")
pdf_reader = PdfReader(pdf_file)
# Расшифровать файл с паролем
pdf_reader.decrypt(password=password)
```

Теперь в объект pdf_reader находится дешифрованный файл и из него можно извлечь текстовую информацию в объект page_txt.

Примечание.

Данный модуль не включен в состав исполняемых примеров интерактивной цифровой книги, так как он предназначен для работы на локальном компьютере. Но вы имеете возможность скопировать листинг данной программы, запустить данный программный модуль на выполнение на своем локальном компьютере и использовать в своих приложениях.

Если запустить данный модуль на выполнение на локальном компьютере, то будет создан зашифрованный файл с именем protect.pdf, после этого он будет загружен с диска, декодирован и на печать будет выведено его текстовое содержимое. При попытке открыть такой файл сторонним пользователем от него потребуются ввод пароля (рис.5.16).

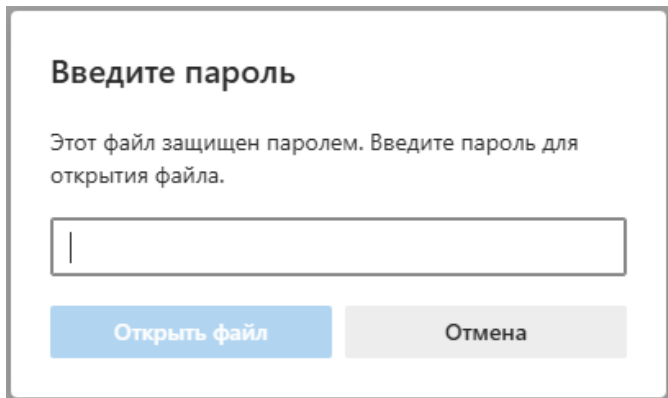


Рис.5.16. Запрос на ввод пароля для защищенного файла test.pdf

Если пользователь введет правильный пароль, то содержимое файла будет открыто (рис. 5.17).

Воробей

Если присмотреться к воробьям, то можно заметить, что это непоседы. Для них трудно устроить фотосессию, так как они постоянно находятся в движении, и фотографироваться не хотят совершенно. Воробьи живут рядом с нами, а много ли мы о них знаем? Думаю, вам будет интересно узнать о них некоторые любопытные факты.

Воробьи не могут долго находиться в воздухе. Птички быстро устают, и им необходимо приземлиться, чтобы передохнуть. Это не самая выносливая птичка.

Рядом с человеком воробьи живут уже более 10000 лет! Цифра впечатляет, не правда ли? Природное любопытство птички известно многим. А ещё воробьи достаточно легко приручаемы, они хорошо идут на контакт. Но это относится к выкормленным человеком птицам, взрослые особи в неволе, как правило, чахнут – наверно от тоски по небу и свободе.

Как выдумаете, как воробушки принимают ванну, и чистят свои перышки? Летом птички чистятся, купаясь в песке, а зимой - в снегу. Впрочем, и просто в воде воробьи тоже любят поплескаться. Мойдодыр был бы ими доволен.

Воробей – это птичка-невеличка. Вес взрослой особи редко достигает больше тридцати граммов, а некоторых видов и вовсе - 10-12 граммов (совсем крошки). Без пропитания воробьи не протянут больше двух суток, так как в полете они тратят много энергии.



Рис.1. Воробей

Большинство животных и птиц видят мир в черно-белых тонах (немного грустно, не правда ли?), но воробьи в этом плане уникальны. Эти птицы видят мир в позитивном розовом цвете!

Рис.5.17. Содержимое файла test.pdf после ввода пароля

Если попытаться открыть данный файл программным способом без указания пароля, то будет выдана следующая ошибка:

```
PyPDF2.errors.FileNotDecryptedError: File has not been decrypted
```

5.12. Защита PDF файлов паролем в приложениях на Django

Шифрование PDF файлов паролем позволяет обеспечить защиту конфиденциальной информации. Зашифрованный документ могут просмотреть только те пользователи, у которых есть пароль для открытия файла. Библиотека PyPDF2 предоставляет возможность, как установить защиту паролем для PDF документов, так и снять защиту программным способом. Фактически программные модули создают новый (зашифрованный) файл на основе исходного файла, либо новый (дешифрованный) файл на основе зашифрованного.

В данном разделе будет создан набор программных модулей, которые выполняют следующие операции:

- ☐ чтение с диска исходного PDF файла;
- ☐ шифрование содержимого загруженного файла с использованием заданного пароля;
- ☐ запись зашифрованного файла на диск под новым именем;

- ☐ чтение с диска зашифрованного PDF файла;
- ☐ декодирование зашифрованного PDF файла с использованием заданного пароля;
- ☐ получение текстовой информации из декодированного файла;
- ☐ вывод в шаблон Django текстового содержимого обработанного PDF файла.

Для начала создадим ссылку, которая будет активировать работу модулей защиты PDF файлов (листинг 5.26).

Листинг 5.26. Ссылка активации модулей защиты файлов PDF

Примечание.

В демо-версии книги удалены листинги программ.

По данной ссылке будет вызван URL маршрут с именем `prot_pdf` (листинг 5.27).

Листинг 5.27. URL маршрут с именем `prot_pdf`

Примечание.

В демо-версии книги удалены листинги программ.

По данному маршруту будет активировано представление `views.prot_pdf`, которое работает в паре с функцией `protect_pdf` (листинг 5.28).

Листинг 5.28. Представление `prot_pdf` и функция `protect_pdf`

Примечание.

В демо-версии книги удалены листинги программ.

В этом представлении сначала задается путь к исходному PDF файлу (`inp_pdf_file`), путь к защищенному PDF файлу (`out_pdf_file`), а также пароль (`password`). После этого делается обращение к функции защиты файла (`protect_pdf`), в которую передаются сформированные аргументы.

В функции `protect_pdf` выполнен импорт модулей библиотеки `PyPDF2`, необходимых для обработки PDF файлов. После этого содержимое исходного PDF файла загружается в объект `pdf_reader`. После этого создается объект `pdf_writer` для формирования нового (защищенного) PDF файла, в который, с использованием инструкции `for`, перегружается содержимое исходного PDF файла в разрезе каждой страницы. После этого с использованием метода `encrypt` объекта `pdf_writer` выполняется шифрование выходного файла и запись его на диск.

В представлении `prot_pdf` после возвращению из функции шифрования данных выполняется открытие и чтение зашифрованного файла в объект `pdf_reader`. Затем, с использованием метода `decrypt` объекта `pdf_reader` происходит снятие пароля и дешифровка защищенного файла.

На следующем шаге с использованием инструкции `for` выполняется чтение текстовой информации из дешифрованного файла с сохранением ее в текстовой переменной `txt_ru`. Эта тек-

стовая переменная через контекстный словарь context передается в шаблон pdf_protect.html, код которого представлен в листинге 5.29.

Листинг 5.29. Шаблон pdf_protect.html

Примечание.

В демо-версии книги удалены листинги программ.

В этом в теге `{{txt_ru}}` вывод текста, который был получен из зашифрованного PDF файла.

Примечание.

В интерактивной цифровой книге для запуска данного примера нужно на вкладке «Показ примеров» нажать на кнопку «Примеры главы 5» и активировать ссылку «Защита PDF файла паролем».

Если запустить приведенные выше программные модули на выполнение, то будет получен следующий результат (рис. 5.18).

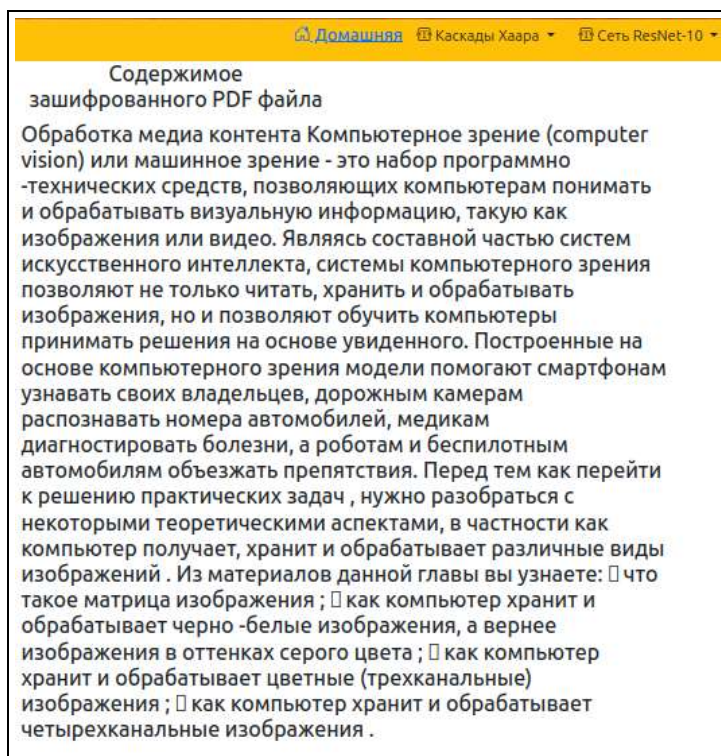


Рис.5.18. Результаты работы программы шифрования и дешифровки PDF файла

5.13. Программная дешифровка PDF файлов в приложениях на Django

В приложении Django можно дешифровать документ программным способом, не требуя от пользователя ввода пароля. В этом случае программные модули создают новый (дешифрованный) файл на основе зашифрованного, и содержимое дешифрованного файла будет выведено на HTML страницу. Про этом не обязательно создавать шаблон Django.

В данном разделе будет создан набор программных модулей, которые выполняют следующие операции:

- ☐ чтение с диска зашифрованного PDF файла;
- ☐ декодирование зашифрованного PDF файла с использованием заданного пароля;
- ☐ сохранение декодированного файла на диске;
- ☐ вывод содержимого декодированного PDF файла пользователю без использования шаблона Django.

Для начала создадим ссылку, которая будет активировать работу модулей дешифровки PDF файлов (листинг 5.30).

Листинг 5.30. Ссылка активации модулей дешифровки файлов PDF

Примечание.

В демо-версии книги удалены листинги программ.

По данной ссылке будет вызван URL маршрут с именем `decode_pdf` (листинг 5.31).

Листинг 5.31. URL маршрут с именем `decode_pdf`

Примечание.

В демо-версии книги удалены листинги программ.

По данному маршруту будет активировано представление `views.decode_pdf`, которое работает в паре с функцией `decrypt_pdf` (листинг 5.32).

Листинг 5.32. Представление `decode_pdf` и функция `decrypt_pdf`

Примечание.

В демо-версии книги удалены листинги программ.

В этом представлении сначала выполняется импорт необходимых пакетов, а также задается путь к защищенному PDF файлу (`path_pdf_clos`), путь к декодированному PDF файлу (`path_pdf_open`), а также пароль (`password`). После этого делается обращение к функции декодирования файла (`decrypt_pdf`), в которую передаются сформированные аргументы.

В функции `decrypt_pdf` выполняется открытие зашифрованного и создаваемого файла. После этого содержимое зашифрованного PDF файла загружается в объект `pdf_reader` и выполняется его декодирование. После этого создается объект `pdf_writer` для формирования нового (декодированного) PDF файла, в который, с использованием инструкции `for`, пере-

гружается содержимое исходного PDF файла в разрезе каждой страницы. Декодированный файл сохраняется на диске.

В представлении `decode_pdf` после возвращению из функции декодирования данных выполняется открытие и чтение декодированного файла в объект `pdf_data`, который передается в функцию `HttpResponse` для отображения на HTML странице.

Примечание.

В интерактивной цифровой книге для запуска данного примера нужно на вкладке «Показ примеров» нажать на кнопку «Примеры главы 5» и активировать ссылку «Программная дешифровка PDF файла».

Если запустить приведенные выше программные модули, то будет получен следующий результат (рис. 5.19).

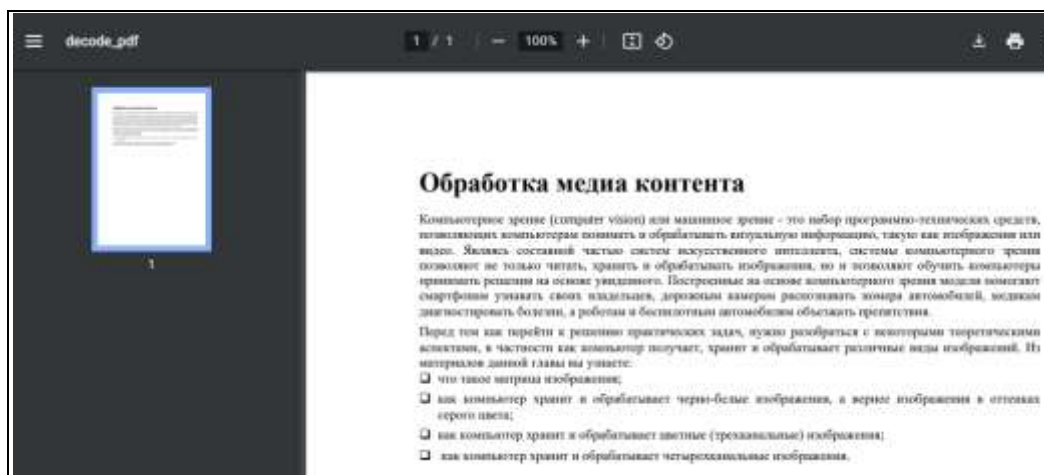


Рис.5.19. Отображение декодированного PDF файла на HTML странице

5.14. Дешифровка PDF файлов путем ввода пароля в шаблоне Django

В приложении Django зашифрованные файлы можно открыть только после того, как пользователь ввел соответствующий пароль. В этом случае на HTML страницу загружается зашифрованный PDF файл с запросом на ввод пароля. До тех пор пока пользователь не введет правильный пароль, содержимое файла не будет выведено на HTML страницу.

Для начала создадим ссылку, которая будет активировать работу модулей отображения зашифрованного PDF файла (листинг 5.33).

Листинг 5.33. Ссылка активации модулей вывода содержимого зашифрованного файла PDF

Примечание.

В демо-версии книги удалены листинги программ.

По данной ссылке будет вызван URL маршрут с именем `open_user_pdf` (листинг 5.34).

Листинг 5.34. URL маршрут с именем `open_user_pdf`

Примечание.

В демо-версии книги удалены листинги программ.

По данному маршруту будет активировано представление `views.open_user_pdf` (листинг 5.35).

Листинг 5.35. Представление `open_user_pdf`

Примечание.

В демо-версии книги удалены листинги программ.

В данном представлении сначала выполняется импорт необходимых пакетов и определяется путь к зашифрованному PDF файлу (объект `path_pdf_clos`). Затем выполняется открытие и считывание содержимого PDF файла в объект `pdf_data`, который передается в функцию `HttpResponse`.

Примечание.

В интерактивной цифровой книге для запуска данного примера нужно на вкладке «Показ примеров» нажать на кнопку «Примеры главы 5» и активировать ссылку «Открыть PDF файла паролем».

Если запустить приведенные выше программные модули, то будет получен следующий результат (рис. 5.20).

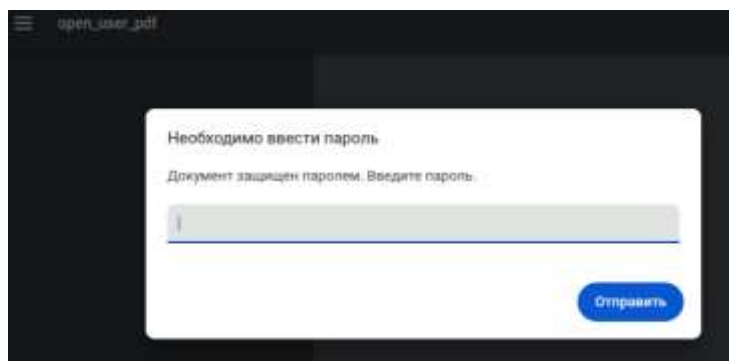


Рис.5.20. Окно ввода пароля для дешифровки PDF файла

Примечание.

Для дешифровки данного файла нужно ввести пароль – 12345.

Как видно из данного рисунка, на HTML странице появилось окно ввода пароля для дешифровки файла. После того, как пользователь введет корректный пароль, содержимое PDF файла будет декодировано и показано на HTML странице (рис. 5.21).

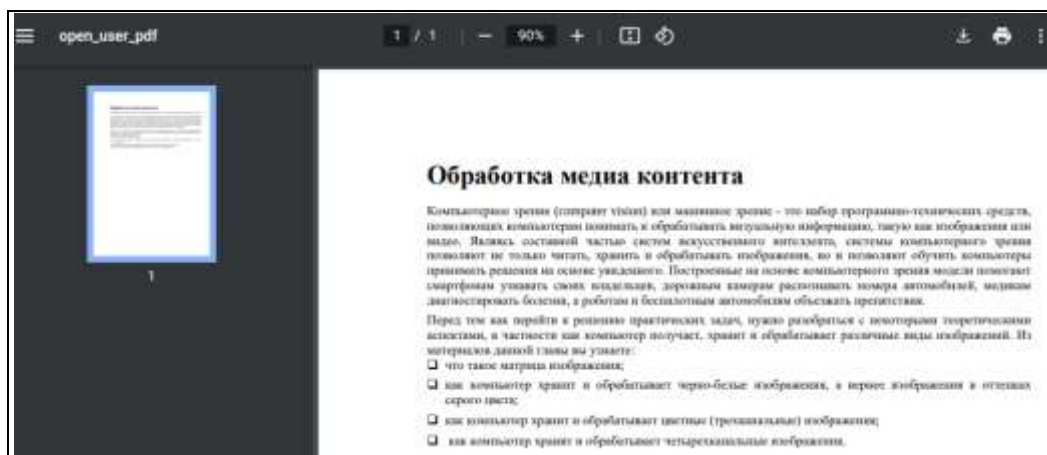


Рис.5.21. Отображение содержимого PDF файла на HTML странице после ввода пароля и дешифровки

5.15. Краткие итоги

Итак, мы познакомились с некоторыми базовыми принципами обработки PDF файлов в модулях Python и веб приложениях на Django. Научились переносить содержимое одного PDF документа в другой, включая рисунки, с сохранением количеством страниц. Разобрались с возможностями защиты PDF файлов паролями, и дешифровкой закодированных PDF файлов программным способом.

Далее нам предстоит узнать, как можно озвучивать содержимое PDF документов и переводить их на различные языки с использованием нейронных сетей. Рассмотрению этих вопросов и посвящена следующая глава.

ГЛАВА 6



Модули Python для распознавания речи и перевода текстов

Программное обеспечение синтеза речи обеспечивает преобразование текст в аудио дорожку или в файл. Современные системы синтеза речи используют глубокое обучение и нейронные сети для создания естественного звучания голоса на различных языках. Примеры таких систем являются Google Text-to-Speech, Amazon Polly и Microsoft Azure TTS. Эти системы не только преобразуют текст в речь, но и позволяют настраивать голос, интонацию и скорость озвучивания. Синтез речи нашел широкое применение в различных областях. Например, он используется в навигационных системах, где голосовые подсказки помогают водителям ориентироваться на дороге. Также эти технологии применяются в системах помощи людям с ограниченными возможностями, позволяя им взаимодействовать с компьютерами и мобильными устройствами.

Технология Speech-to-Text (STT) или другими словами транскрипция – это процесс преобразования устной речи в текст. Основой этой технологии являются алгоритмы машинного обучения и нейронные сети, которые анализируют оцифрованные звуковые волны и преобразуют их в текст. Эта технология востребована для голосового управления компьютером и другими устройствами. Транскрипционные сервисы позволяют автоматизировать преобразование аудиозаписи в текстовые документы, что значительно упрощает обработку интервью, выступлений на конференциях и форумах.

В настоящее время очень востребованы программы для машинного перевода текстов с одного языка на другой с использованием искусственного интеллекта. В совокупности технологии синтеза речи, распознавания речи и перевода позволяют реализовать различные полезные сервисы, которые могут быть встроены в веб приложения или мобильные устройства. В данной главе приводятся сведения об основных библиотеках и алгоритмах, которые позволяют выполнить программную реализацию этих сервисов на Python. Из материалов данной главы вы узнаете:

- ☐ какие библиотеки Python можно использовать для распознавания речи;
- ☐ как можно реализовать распознавание речи в аудио файлах и в аудио потоке с микрофона;
- ☐ как можно реализовать преобразование текста в речь с пакетом gTTS API и библиотекой pyttsx3;
- ☐ как выполняется распознавание речи с микрофона с записью в текстовый файл;
- ☐ как можно выполнить перевод текстов с использованием пакета API Google Translate и библиотекой deep-translator;
- ☐ как можно реализовать перевод содержимого многостраничных файлов формата PDF;
- ☐ как можно озвучить текст PDF файлов;

- ❑ как можно выполнить перевода и озвучивание текста PDF файлов с использованием различных библиотек;
- ❑ как можно реализовать голосовое управление компьютером.

6.1. Распознавание речи и обработка аудио файлов на Python

6.1.1. Библиотеки Python для распознавания речи

Включение элементов распознавания речи в приложение Python позволяет обеспечить интерактивное взаимодействие пользователя с приложением, и даже управление работой приложения с помощью голосовых команд.

Современные системы распознавания речи имеют огромный словарный запас и могут работать с разными языками. При этом речь преобразуется из физического звука в электрический сигнал с помощью микрофона, а затем аналоговый сигнал с микрофона преобразуется в цифровые данные, которые можно использовать для трансформации аудио данных в текстовой формат.

В репозитории PyPI, который предоставляет централизованное место для хранения открытых библиотек на Python, можно найти самые авторитетные и актуальные версии библиотек для распознавания речи:

- `apiai`;
- `assemblyai`;
- `google-cloud-speech`;
- `pocketsphinx`;
- `SpeechRecognition`;
- `watson-developer-cloud`;
- `wit`.

Некоторые из этих пакетов, такие как `wit` и `apiai`, предлагают встроенные функции, такие как, обработка на естественном языке для определения намерений говорящего, которые выходят за рамки базового распознавания речи. Другие пакета, такие как `google-cloud-speech`, сосредоточены исключительно на преобразовании речи в текст.

Пакет `SpeechRecognition` позволяет создавать сценарии для доступа к микрофонам и обработки аудиофайлов в текст с нуля. Библиотека `SpeechRecognition` действует как оболочка для нескольких популярных речевых API и, таким образом, является чрезвычайно гибкой, и это является отличным выбором для любого проекта Python. Тем не менее, поддержка каждой функции каждого API, который он включает, не может гарантировать успешность применения данного пакета для решения ваших задач. Поэтому вам нужно будет потратить некоторое время на изучение доступных опций, чтобы выяснить, будет ли `SpeechRecognition` работать в вашем конкретном случае.

6.1.2. Установка библиотек для распознавания речи и обработки аудио файлов

Установить пакет распознавания речи SpeechRecognition можно из терминала с помощью следующей команды pip:

```
pip install SpeechRecognition
```

Пакета SpeechRecognition будет вполне достаточно для обработки аудио-файлов. Однако если потребуется захват аудио сигнала с микрофона, то необходимо установить пакет PyAudio с использованием следующей команды:

```
pip install PyAudio
```

Если необходимо озвучить аудиофайл, то понадобится пакет для воспроизведения аудио файлов playsound, который можно установить с использованием следующей команды:

```
pip install playsound==1.2.2
```

Для того чтобы подключить к проекту модули пакета SpeechRecognition необходимо выполнить его импорт с использованием следующей команды:

```
import speech_recognition as sr
```

На следующем шаге необходимо создать экземпляр Recognizer, основной целью которого является распознавание речи. Создать экземпляр Recognizer можно с использованием следующей инструкции:

```
r = sr.Recognizer()
```

Каждый экземпляр Recognizer имеет семь методов для распознавания речи из аудио-источника с использованием различных API:

- recognize_bing(): Microsoft Bing Speech;
- recognize_google(): Google Web Speech API;
- recognize_google_cloud(): Google Cloud Speech - требует установки пакета google-cloud-speech package;
- recognize_houndify(): Houndify by SoundHound;
- recognize_ibm(): IBM Speech to Text;
- recognize_sphinx(): CMU Sphinx - требует установки PocketSphinx;
- recognize_wit(): Wit.ai.

Из семи методов только recognize_sphinx() работает в автономном режиме с движком CMU Sphinx. Остальные шесть требуют подключения к интернету.

Все семь методов распознавания класса Recognizer требуют наличия аргумента audio_data. В каждом случае audio_data должен быть экземпляром класса AudioData SpeechRecognition. Существует два способа создания экземпляра AudioData: из аудио-файла или аудио потока с микрофона.

6.1.3. Модуль Python для распознавания речи в аудио файлах

Для начала рассмотрим программный модуль, который позволяет выполнить распознавание речи в аудио файлах (листинг 6.1).

Листинг 6.1. Модуль `txt_forom_audio.py`

Примечание.

В демо-версии книги удалены листинги программ.

Для работы данного модуля выполняется импорт следующих пакетов:

- ❑ `import speech_recognition as sr` – распознавание речи в аудио контенте;
- ❑ `from playsound import playsound` – воспроизведение аудио контента;
- ❑ `from pathlib import Path` – формирование пути к приложению;
- ❑ `import os` – формирование пути к файлам приложения.

На следующем шаге создается ряд объектов, которые содержат пути к файлам приложения:

- ❑ `my_app = Path.cwd()` - путь к папке приложения;
- ❑ `path_audio = os.path.join(my_app, 'audio')` - путь к папке audio (с аудиофайлами);
- ❑ `path_audio_file = os.path.join(path_audio, 'test_1.wav')` - путь к аудио файлу.

После это воспроизводим (озвучиваем) исходный аудио файл:

```
playsound(path_audio_file)
```

Эта инструкция обеспечить проигрывание исходно аудиофайла на компьютере пользователя.

Затем создаем объект «г», который обеспечивает распознавание речи в аудио файле:

```
r = sr.Recognizer()
```

На следующем шаге в объект `audio_file` загружаем исходный аудиофайл:

```
audio_file = sr.AudioFile(path_audio_file)
```

Следующие инструкции обеспечивают распознавание текста в аудио файле:

```
with audio_file as source:  
    audio_data = r.record(source)  
    text = r.recognize_google(audio_data, language="ru-RU")
```

Здесь в объект `audio_data` загружается исходный аудио файл. Затем для объекта «г» вызывается метод `recognize_google`, которому передается исходный аудио файл и указывается язык, на котором звучит речь в загруженном файле (в данном примере речь звучит на русском языке - "ru-RU"). Распознанный текст будет сохранен в объекте «text».

Распознанный текст можно либо сохранить в текстовом файле, либо вывести на печать. В данном примере текст будет выведен на печать:

```
print(text)
```

Примечание.

Данный модуль не включен в состав исполняемых примеров интерактивной цифровой книги, так как он предназначен для работы на локальном компьютере. Но вы имеете возможность

скопировать листинг данной программы, запустить данный программный модуль на выполнение на своем локальном компьютере и использовать в своих приложениях.

Если данный программный код запустить на выполнение на локальном компьютере, то будет получен следующий результат (рис. 6.1).

```
Вас приветствует фабрика информационных технологий дождитесь ответа секретаря
Process finished with exit code 0
```

Рис. 6.1. Результаты распознавания текста в аудио файле

Сначала пользователь услышит содержимое аудиофайла, которое озвучивается через динамик компьютера, а потом на печать будет выведен текст, содержащийся в данном аудио файле.

6.1.4. Модуль Python для распознавания речи с микрофона

В этом разделе рассмотрим программный модуль, который позволяет выполнить распознавание речи из аудио потока микрофона (листинг 6.2).

Листинг 6.2. Модуль txt_forom_mic.py

Примечание.

В демо-версии книги удалены листинги программ.

Для работы данного модуля выполняется импорт пакета распознавания речи в аудио контенте:

```
import speech_recognition as sr
```

На следующем шаге создается два объекта: «г» - для распознавания речи, «mic» - источник аудио данных (микрофон).

Следующие инструкции обеспечивают прием и запись звука из источника данных (в данном случае с микрофона):

```
with mic as source:
    # Устанавливаем паузу в 1 секунды
    r.pause_threshold = 1
    # Удаление посторонних шумов из аудио потока
    r.adjust_for_ambient_noise(source, duration=1)

    print('Говорите')
    # Записываем аудио поток в переменную audio
    audio = r.record(source, duration=10)
    print('Конец записи')
```

Обратите внимание, что для активации микрофона перед приемом данных создается пауза в 1 секунду. Для устранения лишних шумов используется метод `adjust_for_ambient_noise`. С использованием метода `record` объекта «г» выполняется прием аудио потока данных с микрофона и его запись в объект `audio`. Продолжительность записи указана в параметре `duration=10` (10 секунд). Когда поток данных с микрофона прерывается (после истечении 10

секунд), запускается процесс распознавание речи в полученных аудио данных. Так как аудио данные могут содержать сильный посторонний шум или набор звуков, не связанных с текстовой информацией, то инструкции распознавания речи необходимо заключить в блоки «try - except»:

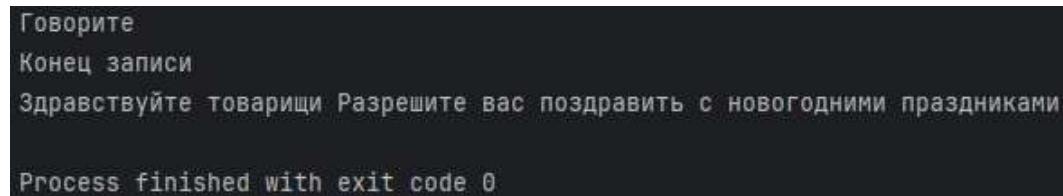
```
try:
    # Распознавание текста из аудио потока
    text = r.recognize_google(audio, language="ru-RU")
    print(text)
except sr.UnknownValueError:
    print('Текст не распознан')
```

Здесь для объекта «г» вызывается метод `recognize_google`, которому передается принятый аудио поток и указывается язык, на котором звучит речь в аудио потоке (в данном примере речь звучит на русском языке - "ru-RU"). Распознанный текст будет сохранен в объекте «text». Если проблем с распознаванием речи не возникнет, то распознанный текст будет выведен на печать. Если в аудио потоке будут преобладать шумы или присутствовать не распознаваемые звуки, то на печать будет выдано сообщение «Текст не распознан».

Примечание.

Данный модуль не включен в состав исполняемых примеров интерактивной цифровой книги, так как он предназначен для работы на локальном компьютере. Но вы имеете возможность скопировать листинг данной программы, запустить данный программный модуль на выполнение на своем локальном компьютере и использовать в своих приложениях.

Если данный программный код запустить на выполнение на локальном компьютере, то будет получен следующий результат (рис. 6.2).



```
Говорите
Конец записи
Здравствуйтесь товарищи Разрешите вас поздравить с новогодними праздниками
Process finished with exit code 0
```

Рис. 6.2. Результаты распознавания текста в аудио потоке с микрофона

Как видно из данного рисунка, после активации микрофона на экране появится сообщение «Говорите». С этого момента в микрофон можно наговаривать текст в течение 10 секунд, он будет принят и сохранен в памяти компьютера. По истечении этого времени на экране появится сообщение «Конец записи», после чего будет запущен процесс распознавания текста, и результаты распознавания выйдут на печать в виде текстовой строки.

При таком подходе получится распознавать только короткие фрагменты речи. А теперь создадим программный модуль, который позволит распознать достаточно продолжительную речь и сохранить ее на диске в виде текстового файла.

6.2. Преобразование текста в речь с пакетом gTTS API

6.2.1. Библиотеки для преобразования текста в речь

Преобразование текста в речь используется для создания голосового вывода на основе текста. Это может быть полезно, когда пользователю необходимо выдать голосовое сообщение. Кроме того, преобразование текста в речь можно использовать для создания голосового сопровождения видео и рекламного контента. В этом разделе будет рассмотрена возможность преобразования текста в речь на Python с использованием пакета gTTS API (Google Text-to-Speech API).

Этот инструмент очень прост в использовании и предоставляет множество встроенных функций, которые выполняют преобразования текстовых переменных или текстового файла в формат mp3. Программисту не нужно создавать собственную нейронную сеть и заниматься ее обучением, можно просто использовать API для решения своих задач.

API gTTS предоставляет возможность конвертировать текстовые файлы с разных языков, таких как английский, немецкий, французский, русский, хинди и многие другие. Также можно управлять темпом речи и воспроизводить ее в быстром или медленном режиме.

Для установки пакета API gTTS можно использовать следующую команду:

```
pip install gTTS
```

Для воспроизведения аудио файлов нам понадобится библиотека playsound, которую можно установить с помощью следующей команды:

```
pip install playsound
```

Чтобы использовать API gTTS для преобразования текста в речь необходимо выполнить следующие шаги:

- ☐ импортировать пакет gTTS в свой программный модуль;
- ☐ определить фрагмент текста для преобразования его в аудио (текстовая переменная или текстовый файл);
- ☐ передать текст в gTTS, указав язык и скорость воспроизведения;
- ☐ сохранить результат преобразования текста в файл формата mp3;
- ☐ открыть медиа файл и прослушать записанную речь.

6.2.2. Модуль Python для преобразования текста в речь

Программный модуль, который выполняет преобразование текста в голосовой аудио файл, представлен в листинге 6.3.

Листинг 6.3. Модуль txt_to_speech.py

Примечание.

В демо-версии книги удалены листинги программ.

В первых строках программы выполняется импорт необходимых пакетов, которые используются для следующих целей:

- ❑ `from gtts import gTTS` – для конвертирования текста в аудио файл;
- ❑ `from playsound import playsound` – для озвучивания аудио файлов;
- ❑ `from pathlib import Path` - для формирования путей к файлам;
- ❑ `import os` – для формирования путей к файлам.

Для формирования путей к файлам используются библиотеки `Path` и `os`, так как они обеспечивают корректную трактовку путей к файлам вне зависимости от используемой операционной системы.

На следующем шаге сформированы пути к файлам приложения:

- ❑ `my_app = Path.cwd()` - путь к папке с приложением;
- ❑ `path_audio = os.path.join(my_app, 'audio')` - путь к папке audio файлами;
- ❑ `path_audio_file = os.path.join(path_audio, 'from_gtts.mp3')` - путь к аудио файлу.

После этого создана текстовая переменная (`text_val`), в которую загружен исходный текст и указан язык (русский):

```
# Исходный текст
text_val = 'Программа конвертации текста в аудио файл '
# Язык - русский
language = 'ru'
```

Затем выполняется конвертация текста в аудио данные их запись в медиа файл формата mp3:

```
# Конвертация текста в аудио
obj = gTTS(text=text_val, lang=language, slow=False)
# Запись аудио файла в формате mp3
obj.save(path_audio_file)
```

Теперь можно озвучить созданный аудио файл:

```
# Озвучивание аудио файла
playsound(path_audio_file)
# удалить голосовой файл
os.remove(path_audio_file)
```

После того, как воспроизведение файла завершиться, он будет удален с диска. На этом этапе завершен процесс преобразование текста из текстовой переменной в аудио дорожку.

В следующем фрагменте программного кода реализован процесс преобразования текста в аудио дорожку из текстового файла. Сначала определены пути к папке с текстовыми файлами и исходному текстовому файлу

- ❑ `path_txt = os.path.join(my_app, 'texts')` - путь к папке с текстовыми файлами;
- ❑ `path_my_txt = os.path.join(path_txt, 'my_text.txt')` - путь к итоговому текстовому файлу.

Затем содержимое текстового файла считывается в переменную `text_val` и для контроля выводится на печать:

```
# Загрузить текстовый файл
with open(path_txt_file, "r", encoding="utf-8") as f:
    text_val = f.read()
print(text_val)
```

А затем повторяется процесс преобразования текста в аудио дорожку и проигрывание полученного медиа файла:

```
# Конвертация текста в аудио
obj = gTTS(text=text_val, lang=language, slow=False)
# Запись аудио файла в формате mp3
obj.save(path_audio_file)
# Озвучивание аудио файла
playsound(path_audio_file)
# удалить голосовой файл
os.remove(path_audio_file)
```

После завершения проигрывания медиа файла он удаляется с диска, чтобы освободить дисковое пространство.

Примечание.

Данный модуль не включен в состав исполняемых примеров интерактивной цифровой книги, так как он предназначен для работы на локальном компьютере. Но вы имеете возможность скопировать листинг данной программы, запустить данный программный модуль на выполнение на своем локальном компьютере и использовать в своих приложениях.

Если данный программный код запустить на выполнение на локальном компьютере, то программа озвучит содержимое текстовой переменной и текстового файла. В данном примере будет озвучено содержание следующего текстового файла:

```
книга посвящена вопросам разработки веб-приложений с использованием языка программирования python,
фреймворков django, bootstrap и интерактивной среды разработки paysharm.
рассмотрены основные технологии и рабочие инструменты создания веб-приложений,
описаны фреймворки джанго, bootstrap и структура создаваемых веб-приложений.
на простых примерах показана обработка и маршрутизация запросов пользователей,
формирование ответных веб-страниц.
рассмотрено создание шаблонов веб-страниц и форм для пользователей,
показано взаимодействие пользователей с различными типами баз данных через модели.
описана работа с базами данных через встроенные в джанго классы без использования sql-запросов,
приведён пошаговый пример создания сайта от его проектирования до формирования программных модулей
и развёртывания в сети интернет с базами данных sql lite 3 и my sql.
электронный архив на сайте издательства содержит коды всех примеров данной книги.
```

Примечание.

Данный программный модуль будет работать только при подключении компьютера к сети интернет, так как элементы искусственного интеллекта, обеспечивающие преобразование текста в аудио дорожку, находятся на публичных серверах компании Google.

Теперь рассмотрим голосового помощника, который работает в автономном режиме.

6.2.3. Преобразование текста в речь с библиотекой pyttsx3

Если требуется выполнить преобразование текста в речь в автономном режиме, то можно воспользоваться библиотекой pyttsx3, которая использует движки TTS (преобразователи текста в речь), которые имеются в операционной системе компьютера. Это более простой способ, который справляется с поставленной задачей, правда генерируемая аудио дорожка не такая качественная, как создаваемая нейронными сетями.

Для начала необходимо установить библиотеку pyttsx3 с использованием следующей команды:

```
pip install pyttsx3
```

Программный модуль, который выполняет преобразование текста в голосовой аудио файл с использованием данной библиотеки, представлен в листинге 6.4.

Листинг 6.4. Модуль use_pyttsx.py

Примечание.

В демо-версии книги удалены листинги программ.

В первых строках программы выполняется импорт необходимых пакетов, которые используются для следующих целей:

- ❑ import pyttsx3 – для конвертирования текста в аудио файл;
- ❑ from pathlib import Path - для формирования путей к файлам;
- ❑ import os – для формирования путей к файлам.

На следующем шаге выполняется инициализация движка преобразования текста в речь:

```
engine = pyttsx3.init()
```

После этого формируется текстовая переменная (text), которая передается в движок преобразования текста в речь, и запускается процесс озвучивания аудио дорожки:

```
# Создание текстовой переменной
text = "автономный конвертор текста в голосовую дорожку"
# Преобразование текста в речь
engine.say(text)
# Воспроизведение речи
engine.runAndWait()
```

На следующем шаге формируются путь к текстовому файлу:

```
# Путь к папке с приложением
my_app = Path.cwd()
# Путь к папке с текстами
path_txt = os.path.join(my_app, 'texts')
```

```
# Путь к текстовому файлу
path_txt_file = os.path.join(path_txt, 'new_txt.txt')
```

Содержимое текстового файла загружается в переменную text:

```
# Загрузить текстовый файл
with open(path_txt_file, "r", encoding="utf-8") as f:
    text = f.read()
```

И, наконец, задаем скорость воспроизведения звуковой дорожки, передаем содержимое текстового файла методу преобразования текста в речь и озвучиваем результат:

```
# Задать скорость речи (по умолчанию 200)
engine.setProperty("rate", 150)
# Преобразование текста в речь
engine.say(text)
# Воспроизведение речи
engine.runAndWait()
```

Примечание.

Данный модуль не включен в состав исполняемых примеров интерактивной цифровой книги, так как он предназначен для работы на локальном компьютере. Но вы имеете возможность скопировать листинг данной программы, запустить данный программный модуль на выполнение на своем локальном компьютере и использовать в своих приложениях.

Если данный программный код запустить на выполнение на локальном компьютере, то программа озвучит содержимое текстовой переменной и текстового файла.

6.3. Распознавание речи с микрофона с записью в текстовый файл

Для записи в текстовый файл продолжительной речи будем использовать следующий алгоритм:

- ❑ организуем бесконечный цикл, в теле которого будем распознавать аудио поток с микрофона, состоящего из непродолжительных фраз (предложений);
- ❑ каждая фраза будет сохранена в памяти компьютера в виде аудио данных, которые будут переданы в блок распознавания речи (в нейронную сеть);
- ❑ если распознавание речи будет выполнено успешно, то распознанный текст будет новой строкой дописан в текстовый файл;
- ❑ если аудио фрагмент речи не будет распознан, то пользователю будет предложено повторить его более разборчиво;
- ❑ для завершения бесконечного цикла пользователь должен сказать в микрофон ключевую фразу, например, «конец записи».

Программный модуль, который позволяет выполнить распознавание речи из аудио потока микрофона и сохранение ее в виде текстового файла, приведен в листинге 6.5.

Листинг 6.5. Модуль save_to_file.py

Примечание.

В демо-версии книги удалены листинги программ.

В первых строках программы выполняется импорт необходимых пакетов, которые используются для следующих целей:

- ❑ `import speech_recognition as sr` – для распознавания речи;
- ❑ `import sys` – для завершения программы;
- ❑ `from gtts import gTTS` – для конвертирования текста в аудио файл;
- ❑ `import os` – для формирования путей к файлам;
- ❑ `from playsound import playsound` – для озвучивания аудио файлов;
- ❑ `from pathlib import Path` - для формирования путей к файлам.

Для формирования путей к файлам используются библиотеки `Path` и `os`, так как они обеспечивают корректную трактовку путей к файлам вне зависимости от используемой операционной системы.

На следующем шаге сформированы пути к файлам приложения:

- ❑ `my_app = Path.cwd()` - путь к папке с приложением;
- ❑ `path_txt = os.path.join(my_app, 'texts')` - путь к папке с текстовыми файлами;
- ❑ `path_my_txt = os.path.join(path_txt, 'my_text.txt')` - путь к итоговому текстовому файлу.

В следующих строках создана функция `def text_to_speech(text, lang='ru')`, которая обеспечивает озвучивание текстовых строк. Эта функция в качестве параметров принимает текстовую строку (`text`), и язык, на котором написан текст (`lang='ru'`), в данном случае это русский язык. В теле функции с использованием пакета `gTTS API` выполняется преобразование текстовой строки в аудио данные, то есть в речь – объект `tts`. Затем полученные аудио данные записываются на диск в виде голосового файла формата `mp3` - `tts.save('out.mp3')`. После этого выполняется озвучивание голосового файла - `playsound('out.mp3')`. После озвучивания голосовой файл удаляется - `os.remove('out.mp3')`.

В следующих строках выполняется обращение к функции `text_to_speech` и пользователю озвучиваются следующие фразы:

```
text_to_speech('старт программы распознавания голоса')
text_to_speech('для завершения программы скажите, конец записи')
```

После этого создается объект «г», который обеспечивает распознавание голосовых данных:

```
g = sr.Recognizer()
```

После этого созданы две функции:

- ❑ функция `def command()` служит для приема аудио данных с микрофона и формирования команд;
- ❑ функция `def makeSomething(zadanie)` - служит для обработки команд.

К этим функциям вернемся немного позже, а сейчас обратим внимание на последние две строчки программы, которые запускают бесконечный цикл приема данных с микрофона и их обработки:

```
# Бесконечный цикл
while True:
    # Вызов функции приема данных с микрофона
```

```
makeSomething(command())
```

Здесь с использованием инструкции `while True` создается бесконечный цикл, в теле которого выполняется обращение к функции `makeSomething()`, а в качестве аргумента указана другая функция - `command()`. В теле функции `command()` инициируется прослушивание и прием аудио данных с микрофона:

```
# Начинаем прослушивать микрофон и записываем данные
with sr.Microphone() as source:
    # Устанавливаем паузу 1 сек.
    r.pause_threshold = 1
    # Удаление посторонних шумов из аудио дорожки
    r.adjust_for_ambient_noise(source, duration=1)
    # Сообщение о том, что можно говорить
    print("Говорите")
    # Слушаем и пишем данные в audio как mp3
    audio = r.listen(source)
```

После вывода на печать текста «Говорите» пользователь может начать произносить в микрофон не очень длинную фразу (например, предложение). Эта фраза будет сохранена в объекте `audio` в виде аудио данных. После завершения фразы (пауза), аудио данные будут переданы в объект «г» для распознавания текста. Распознанный текст будет помещен в объект `zadanie`. Поскольку фраза может быть не распознана из-за посторонних шумов или не четкого произношения, то процедура распознавание текста помещена в инструкцию обработки исключений «try – except».

Если ошибок при распознавании голоса не произошло, то пользователю выводится на печать распознанный текст, и он же записывается в виде отдельной строки в текстовый файл.

Если при распознавании голоса возникла ошибка, то пользователю выдается голосовое сообщение:

```
text_to_speech('я вас не поняла, повторите фразу')
```

После этого делается возврат в функцию `command()` для повторения не распознанной фразы.

Функция `def makeSomething(zadanie)` – принимает и обрабатывает сказанные пользователем фразы и обеспечивает завершение бесконечного цикла (выход из программы). Данная функция не будет предпринимать никаких действий до тех пор, пока пользователь не произнесет фразу «конец записи». После этой фразы будут выполнены следующие инструкции:

```
if 'конец записи' in zadanie:
    text_to_speech('я вас поняла, завершаю программу')
    # Выходим из программы
    sys.exit()
```

Пользователю будет озвучена фраза «я вас поняла, завершаю программу», после чего работа программы завершится.

Примечание.

Данный модуль не включен в состав исполняемых примеров интерактивной цифровой книги, так как он предназначен для работы на локальном компьютере. Но вы имеете возможность скопировать листинг данной программы, запустить данный программный модуль на выполнение на своем локальном компьютере и использовать в своих приложениях.

Если данный программный код запустить на выполнение на локальном компьютере и поговорить в микрофон несколько предложений, то в текстовом файле будет сохранен, например, следующий результат:

```
книга посвящена вопросам разработки веб-приложений с использованием языка python
framework django bootstrap и интерактивной среды разработки ray
sharm
рассмотрены основные технологии и рабочие инструменты создания
веб-приложений
описаны фреймворки джанго bootstrap и структура создаваемых веб-
приложений
на простых примерах показана обработка и маршрутизация запросов
пользователей
формирование ответных веб-страниц
рассмотрено создание шаблонов веб-страниц и форм для пользователей
показано взаимодействие пользователей с различными типами баз дан-
ных через модели
описана работа с базами данных через встроенные в джанго классы
без использования sql запросов
приведён пошаговый пример создания сайта от
его проектирование до формирования программных модулей и развёрты-
вания в интернет с базами данных
sqlite и mysql
электронный архив на сайте издательства содержит коды всех приме-
ров
конец записи
```

Как видно из данного текста, программа вполне успешно справилась со своей задачей. Даже некоторые английские термины написаны на латинице (python, django, mysql). Правда в тексте отсутствуют знаки препинания (точки и запятые), предложения начинаются с маленькой буквы. Однако, даже в таком виде можно, не трогая клавиатуру, быстро получить заготовку текста, который затем можно отредактировать. Такие программы могут быть полезны для слабовидящих пользователей, так как они голосом могут набирать текст, а голосовой помощник позволит ориентироваться в ходе выполнения программы.

6.4. Перевод текстов на Python

6.4.1. Модуль Python перевода текстов с использованием пакета API Google Translate

На сегодняшний день Google Translate – это, пожалуй, самый эффективный инструмент, который позволяет сделать перевод текста с одного языка на другой с помощью нейронных сетей и элементов искусственного интеллекта. Разработчики со всего мира могут использовать заложенный в этом инструментарии функционал в приложениях благодаря облачному API перевода от компании Google. С помощью этого API можно переводить тексты из тысяч языковых пар. Возможности машинного перевода Translation API позволяют получать результаты, которые достаточно близки к переводам профессиональных переводчиков.

Для разработки программ языкового перевода на Python будем использовать библиотеку googletrans. Это бесплатный и свободный к использованию пакет Python, который реализует функции Google Translate API. Для установки библиотеки googletrans можно использовать следующую команду:

```
pip install googletrans
```

На текущий момент доступна версия 4.0.2. Следует оговорить, что начиная с версии 4.0, данная библиотека поддерживает асинхронный режим работы программных модулей, соответственно в паре с данной библиотекой нужно использовать функции пакета asyncio. Если вы создаете программы, работающие в синхронном режиме, то нужно использовать более ранние версии библиотеки googletrans. В данном разделе будут рассмотрены примеры на основе googletrans версии 4.0.2.

Программный модуль, который позволяет выполнить перевод текста с английского языка на русский язык, и с русского языка на китайский язык, приведен в листинге 6.6.

Листинг 6.6. Модуль perevod.py

Примечание.

В демо-версии книги удалены листинги программ.

В этом программном модуле сначала выполнен импорт необходимых библиотек:

- ❑ import asyncio – для обеспечения асинхронной работы программы;
- ❑ from googletrans import Translator – для перевода текстов.

На следующем шаге реализована асинхронная функция переводчик - translate_text(). Данная функция принимает следующие параметры:

- ❑ txt – исходный текст, который необходимо перевести на другой язык;
- ❑ src – язык исходного текста;
- ❑ dest – язык, на который нужно перевести исходный текст

В теле функции на основе базового класса Translator() создается объект translator. Далее объект translator вызывает метод translate (переводить), которому передаются параметры: исходный текст (txt), язык исходного текста (src), язык, на который нужно перевести исходный текст (dest). Результаты перевода будут сохранены в объекте result, который содержит следующие элементы:

- ❑ src – язык исходного текста;
- ❑ dest – язык, на который переведен исходный текст;
- ❑ text – переведенный текст;
- ❑ pronunciation – произношение переведенного текста.

Для нас важен параметр text, который и будет возвращен из данной функции в следующей инструкции:

```
return result.text
```

Следующие фрагменты программного кода выполняют переводы:

- ❑ с русского языка на английский;

- ☐ с английского языка на русский;
- ☐ с русского языка на китайский.

Примечание.

Данный модуль не включен в состав исполняемых примеров интерактивной цифровой книги, так как он предназначен для работы на локальном компьютере. Но вы имеете возможность скопировать листинг данной программы, запустить данный программный модуль на выполнение на своем локальном компьютере и использовать в своих приложениях.

Если данный программный код запустить на выполнение на локальном компьютере, то будет получен следующий результат:

```
Translated(src=ru, dest=en, text=Good afternoon, pronunciation=None, extra_data="{\"translat...\"})
Good afternoon
Translated(src=en, dest=ru, text=Добрый день, pronunciation=Dobryy den', extra_data="{\"translat...\"})
Добрый день
Translated(src=ru, dest=zh-cn, text=你好, pronunciation=Nǐ hǎo, extra_data="{\"translat...\"})
你好
```

Как видно из данных результатов, программа успешно справилась с поставленной задачей, текст был переведен с русского языка на английский, с английского на русский, и с русского на китайский язык.

6.4.2. Модуль Python для перевода текстов с библиотекой deep-translator

Библиотека deep-translator – это еще один инструментарий, который позволяет сделать перевод текста с одного языка на другой с помощью нейронных сетей и элементов искусственного интеллекта. Это бесплатный и свободный к использованию пакет Python. По сути, в этом пакете интегрировано нескольких известных переводчиков:

- ☐ Google Translate;
- ☐ Microsoft;
- ☐ Pons;
- ☐ Linguee;
- ☐ Mymemory;
- ☐ Яндекс;
- ☐ QcriTranslator;
- ☐ DeepLTranslator;
- ☐ Papago;
- ☐ Libre;
- ☐ ChatGpt.

Некоторые из приведенных выше переводчиков требуют получение платных или бесплатных ключей (ApiKey), который можно получить на сайте разработчика. В этом разделе мы будем подключать переводчик Google Translate, который по умолчанию использует бес-

платный ApiKey. Для установки библиотеки deep-translator можно использовать следующую команду:

```
pip install deep-translator
```

На текущий момент доступна версия 1.11.4. Также к базовому пакету можно установить различные дополнения, например, перевод файлов docx и PDF.

Программный модуль, который позволяет выполнить перевод текста с английского языка на русский язык, и с русского языка на китайский язык, приведен в листинге 6.7.

Листинг 6.7. Модуль perevod_dp.py

Примечание.

В демо-версии книги удалены листинги программ.

В этом программном модуле сначала выполнен импорт необходимых библиотек:

- ❑ `import asyncio` – для обеспечения асинхронной работы программы;
- ❑ `from deep_translator import GoogleTranslator` – для перевода текстов.

На следующем шаге реализована асинхронная функция переводчик - `translate_text()`. Данная функция принимает следующие параметры:

- ❑ `txt` – исходный текст, который необходимо перевести на другой язык;
- ❑ `src_lang` – язык исходного текста;
- ❑ `dst_lang` – язык, на который нужно перевести исходный текст.

В теле функции базовому классу `GoogleTranslator()` передаются параметры: исходный текст (`txt`), язык исходного текста (`src_lang`), язык, на который нужно перевести исходный текст (`dst_lang`). Результаты перевода будут сохранены в объекте `result` и возвращены в точку вызова функции (`return result`).

Следующие фрагменты программного кода выполняют переводы:

- ❑ с русского языка на английский;
- ❑ с английского языка на русский;
- ❑ с русского языка на китайский.

Примечание.

Данный модуль не включен в состав исполняемых примеров интерактивной цифровой книги, так как он предназначен для работы на локальном компьютере. Но вы имеете возможность скопировать листинг данной программы, запустить данный программный модуль на выполнение на своем локальном компьютере и использовать в своих приложениях.

Если данный программный код запустить на выполнение на локальном компьютере, то будет получен следующий результат:

Good afternoon

Добрый день

你好

Как видно из данных результатов, программа успешно справилась с поставленной задачей, текст был переведен с русского языка на английский, с английского на русский, и с русского на китайский язык.

6.4.3. Модуль Python для перевода многостраничного файла PDF с библиотекой `deep_translator`

Достаточно часто требуется выполнить перевод PDF документа на другой язык с сохранением количества страниц, и с переносом изображений в переведенный документ. Это важно, например, для технической документации, где ссылки на рисунки и сами рисунки должны находиться на одной странице.

Создадим программный модуль, который позволяет извлечь текстовое содержимое исходного pdf файла и рисунки, потом выполнить перевод текстового содержимого на другой язык и выгрузить переведенный текст с рисунками в другой PDF файл с сохранением количества страниц (листинг 6.8).

Листинг 6.8. Модуль `pdf_deep_trans.py`

Примечание.

В демо-версии книги удалены листинги программ.

Для работы данного модуля выполняется импорт необходимых пакетов и библиотек, `deep_translator`, `reportlab`, `PyPDF2` и `imagesize` и др.

Затем определяются языки для исходного текста и для текста перевода. Язык исходного текста может быть задан явно (здесь задан английский – `en`), а может определяться автоматически (параметр `auto`). Для определения языка перевода здесь создано несколько строк, часть из которых закомментировано. Поочередно снимая комментарии с одной из строк, можно переводить исходный текст на разные языки.

На следующем шаге с использованием модулей библиотек `Path` и `os` прописываются пути к исходному (`input_pdf`), выходному (`outpup_pdf`) PDF файлу, и создается холст (объект «с» на основе класса `Canvas`).

После этого с использованием класса `PdfReader` в объект «pdf» загружается содержимое исходного PDF файла и определяется количество страниц (`num_pages`).

В данном программном модуле полученный текст будет сохранен в другом файле PDF с заменой параметров строк. Для этого далее создан объект `styles`, в который загружены стандартные параметры строк класса `SimpleDocTemplate`. В данном примере текст будет переводиться на разные языки, а они имеют совершенно разный набор шрифтов (например, китайский, арабский и т.п.). Поэтому далее, в зависимости от выбранного языка, устанавливается нужный фонт, пользовательский стиль и размер шрифта для выходного PDF файла.

На следующем шаге формируются параметры страницы создаваемого PDF файла (размеры страницы, отступы от края, полезное поле для вывода строк), и определяется начально положение первой строки текста (координаты `x` и `y`).

После этого на основе инструкции `for` создан цикл по страницам текста исходного файла, в теле которого очередная страница заполняется строками текста. Здесь каждая страница загружается в объект `page`, из которого все строки текста страницы извлекаются в объект `text`.

После этого текст текущей страницы переводится на выбранный язык и разбивается на строки (объект paragraphs).

Затем с использованием инструкции for создан вложенный цикл по строкам страницы. Каждая новая строка заносится в объект paragraph, строка вписывается в ширину страницы и с использованием метода drawOn помещается на холст с использованием следующей инструкции:

```
paragraph.drawOn(c, x, y - height)
```

Здесь в метод drawOn переданы следующие параметры:

- ☐ c - холст (canvas) или страница;
- ☐ x – координата «x» положения начала строки;
- ☐ y - координата «y» положения начала строки;
- ☐ height – полезная высота страницы в точках.

После вывода очередной строки происходит уменьшение координаты вывода следующей строки (координата «y» уменьшается на величину height + 3, где 3 - расстояние между строками).

После вывода всех строк с использованием инструкции if делается проверка – есть ли изображения на данной странице. Если изображение есть (а их может быть несколько), то задаются координаты для вывода первого изображения на страницу (x_fig, y_fig) и, с использованием инструкции for, создается цикл обработки всех изображений. Извлеченное из страницы изображение загружается в объект img_data и сохраняется на диске под именем:

```
image_<номер страницы>_<номер изображения на странице>/.jpg
```

После этого в объектах w_img, h_img запоминаются ширина и высота изображения. Затем изображение вставляется в текущую страницу с использованием следующей инструкции:

```
c.drawImage(f'image_{page_num + 1}_{i}.jpg', x_fig, y_fig, w_img, h_img)
```

Здесь методу drawImage переданы следующие параметры:

- ☐ f'image_{page_num + 1}_{i}.jpg' – имя файла с изображением;
- ☐ x_fig – координата «x» нижнего левого угла изображения;
- ☐ y_fig - координата «y» нижнего левого угла изображения;
- ☐ w_img – ширина изображения;
- ☐ h_img – высота изображения.

После того, как изображение вставлено на страницу, координата «y» уменьшается на высоту текущего изображения (она будет использоваться для вывода следующего изображения), и текущее изображение удаляется с диска.

После завершения обработки всех строк и изображений текущей страницы в выходном документе создается новая страница - инструкция c.showPage(), а после обработки всех страниц выходной PDF файл записывается на диск – инструкция c.save().

Примечание.

Данный модуль не включен в состав исполняемых примеров интерактивной цифровой книги, так как он предназначен для работы на локальном компьютере. Но вы имеете возможность скопировать листинг данной программы, запустить данный программный модуль на выполнение на своем локальном компьютере и использовать в своих приложениях.

Если данный программный код запустить на выполнение на локальном компьютере, то мы получим переведенный выходной PDF файл, с тем же количеством страниц, как и у исходного файла, при этом будет перенесен и текст и рисунки. На рис. 6.1 представлен фрагмент страницы исходного файла на английском языке с текстом и изображением.

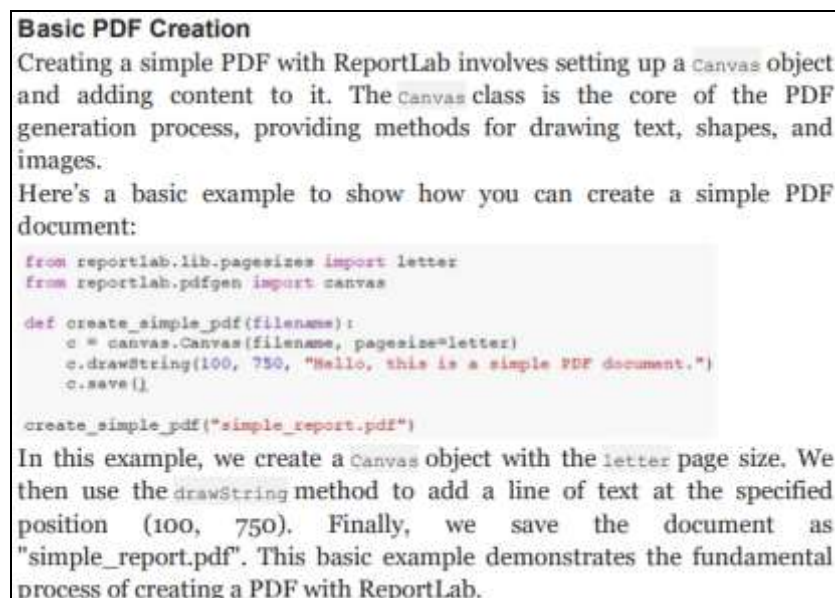


Рис.6.1. Фрагмент текста с изображением в исходном PDF файле на английском языке

На данном рисунке изображением является листинг программного кода. В документации, которую нужно переводить на другие языки и озвучивать, листинги программ необходимо вставлять в документы в виде изображений, так как изображения не нужно переводить и озвучивать.

На рис. 6.2 представлен фрагмент страницы итогового файла, переведенного на русский язык.

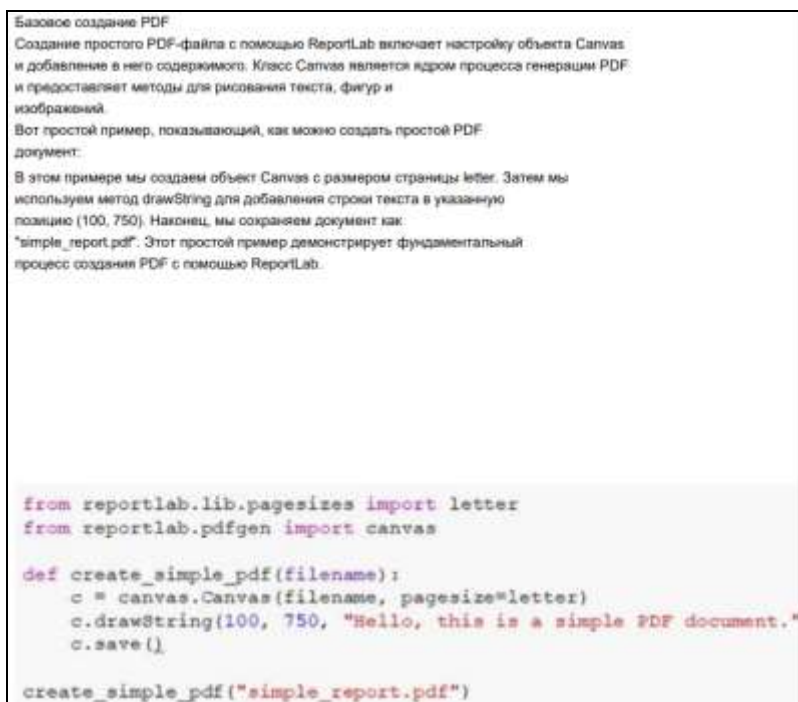


Рис.6.2. Фрагмент текста с изображением в выходном PDF файле на русском языке

Как видно из данного рисунка, переводчик корректно перевел текст на русский язык, при этом технические термины, связанные с программированием, оставлены на английском языке. В нижнюю часть страницы было перенесено изображение.

Теперь прокомментируем строку с языком перевода «русский», и снимем комментарий со строки язык перевода «китайский». Результаты работы программы представлены на рис. 6.3.

基本 PDF 创建

使用 ReportLab 创建简单的 PDF 涉及设置 Canvas 对象并向其添加内容。Canvas 类是 PDF 生成过程的核心，提供绘制文本、形状和图像的方法。

以下是一个基本示例，用于说明如何创建简单的 PDF 文档：

在此示例中，我们创建一个具有信函页面大小的 Canvas 对象。然后，我们使用 drawString 方法在指定位置 (100, 750) 添加一行文本。最后，我们将文档保存为 "simple_report.pdf"。此基本示例演示了使用 ReportLab 创建 PDF 的基本过程。

```
from reportlab.lib.pagesizes import letter
from reportlab.pdfgen import canvas

def create_simple_pdf(filename):
    c = canvas.Canvas(filename, pagesize=letter)
    c.drawString(100, 750, "Hello, this is a simple PDF document.")
    c.save()

create_simple_pdf("simple_report.pdf")
```

Рис.6.3. Фрагмент текста с изображением в выходном PDF файле на китайском языке

Как видно из данного рисунка, программа корректно перевела текст на китайский язык, при этом технические термины, связанные с программированием, оставлены на английском языке. В нижнюю часть страницы было перенесено изображение.

Теперь прокомментируем строку с языком перевода «китайский», и снимем комментарий со строки язык перевода «арабский». Результаты работы программы представлены на рис. 6.4.

قح فصل ال م ج ح ص ص ص خ ت
 ك ب ة ص ا خ ال PDF ت ا د ن ت س م ل ق ح فصل ال م ج ح ص ص ص خ ت ReportLab ك ل ح ي ت ي
 ق ح فصل م ج ح ي ا د ي د ح ت ك ن ك م ي ، ا ع ئ ا ش ا ز ا ي خ ف ر ح ال م ج ح د ع ي ا م ن ي ب و
 ل ث م ، ا ق ب س م ة د ح م ل ت ا خ فصل ال م ا ج ح ا ن م د ي د ع ال ReportLab ر ف و ي . ة ج ا ت ح ت
 د ي د ح ت ل ا ل خ ن م ق ص ص خ م ل ت ا خ فصل ال م ا ج ح ا د ي د ح ت ا ض ي ا ك ن ك م ي . A4 و A5 legal
 ط ا ق ن ا ل ا ب ع ا ف ت ر ا ل ا و ص ر ع ل
 ص ص خ م ق ح فصل م ج ح ب PDF ف ل م ا ش ن ا ل ل ا ث م ي ل ي ا م ي ف
 ي ف ة د ا ع م د خ ت س ر ي ي ذ ل و ، A4 ق ح فصل ال م ج ح م د خ ت س ن ، ل ا ث م ل ا ذ ه ي ف
 ن ك ل و ، ق ب ا س ل ل ا ث م ل ا ي ف ي ه ا م ك ة ي ل م ع ل ل ط ت . ا ب و ر و
 ف ل ت خ م ق ح فصل م ج ح ب

```

from reportlab.lib.pagesizes import A4
from reportlab.pdfgen import canvas

def create_custom_size_pdf(filename):
    c = canvas.Canvas(filename, pagesize=A4)
    c.drawString(100, 800, "This PDF uses the A4 page size.")
    c.save()

create_custom_size_pdf("custom_size_report.pdf")
  
```

Рис.6.4. Фрагмент текста с изображением в выходном PDF файле на арабском языке

Как видно из данного рисунка, программа корректно перевела текст на арабский язык, при этом технические термины, связанные с программированием, оставлены на английском языке. В нижнюю часть страницы было перенесено изображение.

6.5. Модуль Python озвучивания текста PDF файлов с библиотекой PyPDF2

В этом разделе мы узнаем, как с использование Python можно читать PDF-файлы, распознавать в них текст и озвучивать его на языке оригинала. Первое, что необходимо сделать, — это извлечь текст из PDF файла. Для извлечения текста из PDF файла будем использовать библиотеку PyPDF2, для озвучивания текста — библиотеку gTTS, а для воспроизведения аудио файлов — библиотеку playsound. Установить эти пакеты можно с использованием следующих команд:

```

pip install PyPDF2
pip install gTTS
pip install playsound
  
```

Программный модуль, который позволяет выполнить выборку текста из PDF файла и его озвучивание его на языке оригинала, приведен в листинге 6.9.

Листинг 6.9. Модуль pdf_audio_book.py

Примечание.

В демо-версии книги удалены листинги программ.

В этом программном модуле сначала выполнен импорт необходимых библиотек:

- ❑ `import PyPDF2` – для извлечения текста из PDF файла;
- ❑ `import gtts` – для преобразования текста в аудио файл;
- ❑ `from playsound import playsound` – для проигрывания аудио файла
- ❑ `from pathlib import Path` – для указания путей к файлам;
- ❑ `import os` – для указания путей к файлам;

На следующем шаге заданы пути к файлам и язык, на котором написан текст в PDF файле (объект `lang`):

- ❑ `my_app = Path.cwd()` - путь к папке с приложением;
- ❑ `path_txt = os.path.join(my_app, 'texts')` - путь к папке с текстами;
- ❑ `path_pdf_file = os.path.join(path_txt, 'test_ru.pdf')` - путь к текстовому файлу на русском языке;
- ❑ `lang = "ru";`
- ❑ `# path_pdf_file = os.path.join(path_txt, 'test_en.pdf')` - путь к текстовому файлу на английском языке;
- ❑ `# lang = "en";`
- ❑ `path_audio = os.path.join(my_app, 'audio')` - путь к папке audio;
- ❑ `path_audio_file = os.path.join(path_audio, 'from_gtts.mp3')` - путь к аудио файлу.

В данном примере будет озвучено два PDF файла: на русском языке – `test_ru.pdf`, и на английском языке - `test_en.pdf` (закомментировано).

После этого выполняется открытие PDF файла, чтение из него текстовой информации и определения количества страниц текста:

```
# Открытие и чтение PDF файла
pdf_file = open(path_pdf_file, "rb")
pdf_reader = PyPDF2.PdfReader(pdf_file)
# Определение количества страниц в PDF файле
num_pages = len(pdf_reader.pages)
print('num_pages=', num_pages)
```

После этого выполняется постраничная обработка текста, для чего организован цикл с использованием инструкции `for`:

```
# Постраничное чтение и обработка текста
for page in range(0, num_pages):
    print('Номер страницы-', page)
    # Загрузить страницу
    page = pdf_reader.pages[page]
    # Извлечь текст из страницы
    my_txt = page.extract_text()
    print('----- Текст на языке оригинала -----')
    print(my_txt)
    # Преобразовать текст в аудио дорожку с gTTS
```

```

text_audio = gtts.gTTS(my_txt, lang=lang)
# Сохранить аудио дорожку на диске как файл mp3
text_audio.save(path_audio_file)
# Озвучить аудио дорожку
playsound(path_audio_file)
# удалить голосовой файл
os.remove(path_audio_file)

```

В теле цикла выполняется загрузка одной страницы текста в объект page и извлечение текста из данной страницы в объект my_txt. После этого текст загруженной страницы и язык, на котором написан текст, передаются в метод gTTS пакета gtts для преобразования текста в аудио дорожку (объект text_audio). Сформированная аудио дорожка записывается на диск в виде файла формата mp3. После этого аудио файл озвучивается с использованием библиотеки playsound. После завершения проигрывания аудио файл удаляется с диска и начинает обрабатываться следующая страница PDF файла. После завершения данного цикла выполняется закрытие обрабатываемого PDF файла.

Примечание.

Данный модуль не включен в состав исполняемых примеров интерактивной цифровой книги, так как он предназначен для работы на локальном компьютере. Но вы имеете возможность скопировать листинг данной программы, запустить данный программный модуль на выполнение на своем локальном компьютере и использовать в своих приложениях.

Если данный программный код запустить на выполнение на локальном компьютере, то будет получен следующий результат:

```

num_pages= 1
Номер страницы- 0
----- Текст на языке оригинала -----
Обработка медиа контента
Компьютерное зрение (computer vision) или машинное зрение - это набор программно -
технических средств,
позволяющих компьютерам понимать и обрабатывать визуальную информацию, такую
как изображения или
видео. Являясь составной частью систем искусственного интеллекта, системы компью-
терного зрения
позволяют не только читать, хранить и обрабатывать изображения, но и позволяют обу-
чить компьютеры
принимать решения на основе увиденного. Построенные на основе компьютерного зре-
ния модели помогают
смартфонам узнавать своих владельцев, дорожным камерам распознавать номера авто-
мобилей, медикам
диагностировать болезни, а роботам и беспилотным автомобилям объезжать препят-
ствия.
Перед тем как перейти к решению практических задач , нужно разобраться с некоторы-
ми теоретическими
асpekтами, в частности как компьютер получает, хранит и обрабатывает различные ви-
ды изображений . Из
материалов данной главы вы узнаете:
□ что такое матрица изображения ;

```

- ☐ как компьютер хранит и обрабатывает черно-белые изображения, а вернее изображения в оттенках серого цвета ;
- ☐ как компьютер хранит и обрабатывает цветные (трехканальные) изображения ;
- ☐ как компьютер хранит и обрабатывает четырехканальные изображения .

Это тот текст, который был выбран из PDF файла и озвучен медиа проигрывателем компьютера.

Теперь прокомментируем строки, определяющие путь к исходному PDF файлу на русском языке, и снимем комментарии со строк, определяющие путь к исходному PDF файлу на английском языке. Если после этого запустить данный программный код, то будет получен следующий результат:

```
num_pages= 1
Номер страницы- 0
----- Текст на языке оригинала -----
My Daily Routine
Dear Marcos, Hi, my name is Rebecca and I am your new pen pal. I'm ten years
old, and I live in Barcelona, Spain. I go to the 4th grade in Saint Michael
Elementary School. On weekdays, I get up at 6 o'clock. The school starts at 7:30 in
the morning. I have my lunch at the school cafeteria and I usually leave around
3:30 in the afternoon. I am good at math, I also like history and geography but I
my favorite subject is science. I like playing basketball and soccer. I also play
tennis, and I like swimming. On the weekends, I go to the movie theater with my
friends. We like action movies. I usually go to church with my parents on Sundays.
Please write and tell me about you and your life. What's your family like? What
do you like doing in your spare time? I hope to hear from you soon. Best wishes,
Rebecca .
```

Это тот текст, который был выбран из PDF файла и озвучен медиа проигрывателем компьютера, но уже на английском языке. Таким образом, с использованием данного программного модуля можно озвучивать текст, написанным на любом языке.

6.6. Модуль Python перевода и озвучивания текста PDF файлов с библиотекой Googletrans

В этом разделе мы узнаем, как с использованием Python можно читать PDF-файлы, а также переводить их текстовое содержимое на различные языки с использованием библиотеки googletrans и озвучивать переведенный текст.

В данном разделе для перевода текста будет использована библиотека googletrans версии 4.0.2, которая поддерживает асинхронный режим, и для ее работы нужно подключения пакета asyncio. Программный модуль, который позволяет выполнить выборку текста из PDF файла, его перевод на другой язык и озвучивание переведенного текста, приведен в листинге 6.10.

Листинг 6.10. Модуль pdf_google_trans_s.py

Примечание.

В демо-версии книги удалены листинги программ.

В этом программном модуле сначала выполнен импорт необходимых библиотек:

- ❑ `from PyPDF2 import PdfReader` – для извлечения текста из PDF файла;
- ❑ `import gtts` – для преобразования текста в аудио файл;
- ❑ `from playsound import playsound` – для проигрывания аудио файла;
- ❑ `import asyncio` – для обеспечения асинхронной работы программных модулей;
- ❑ `from googletrans import Translator` – для перевода текстов на другие языки;
- ❑ `from pathlib import Path` – для указания путей к файлам;
- ❑ `import os` – для указания путей к файлам.

На следующем шаге реализована асинхронная функция переводчик - `translate_text()`. Данная функция принимает следующие параметры:

- ❑ `txt` – исходный текст, который необходимо перевести на другой язык;
- ❑ `src` – язык исходного текста;
- ❑ `dest` – язык, на который нужно перевести исходный текст

В теле функции на основе базового класса `Translator()` создается объект `translator`. Далее объект `translator` вызывает метод `translate` (переводить), которому передаются параметры: исходный текст, язык исходного текста, язык, на который нужно перевести исходный текст. Результаты перевода будут сохранены в объекте `result`, который содержит следующие элементы:

- ❑ `src` – язык исходного текста;
- ❑ `dest` – язык, на который переведен исходный текст;
- ❑ `text` – переведенный текст;
- ❑ `pronunciation` – произношение переведенного текста.

Для нас важен параметр `text`, который и будет возвращен из данной функции в следующей инструкции:

```
return result.text
```

На следующем шаге заданы пути к файлам:

- ❑ `my_app = Path.cwd()` - путь к папке с приложением;
- ❑ `path_txt = os.path.join(my_app, 'texts')` - путь к папке с текстами;
- ❑ `path_pdf_file = os.path.join(path_txt, 'test_ru.pdf')` - путь к текстовому файлу;
- ❑ `path_audio = os.path.join(my_app, 'audio')` - путь к папке audio;
- ❑ `path_audio_file = os.path.join(path_audio, 'from_gtts.mp3')` - путь к аудио файлу.

После этого выполняется открытие PDF файла, чтение из него текстовой информации и определения количества страниц текста.

В данном примере загружается PDF файл на русском языке (`test_ru.pdf`), а потом он будет переведен на английский язык с озвучиванием на английском языке, и на китайский язык с озвучиванием на китайском языке (параметры китайского языка закомментированы). Здесь параметр исходного языка загружен в объект «`my_src`», а параметр языка перевода – в объект «`my_dest`».

После этого выполняется постраничная обработка текста, для чего организован цикл с использованием инструкции `for`. В теле цикла выполняется загрузка одной страницы текста в объект `page` и извлечение текста из данной страницы в объект `my_txt`. После этого текст загруженной страницы выводится на печать.

После этого выполняется перевод текста на другой язык. Здесь в асинхронном режиме вызывается функция `translate_text`, в которую передаются: исходный текст (`my_txt`), язык исходного текста (`my_src`), язык, на который нужно перевести текст (`my_dest`). Переведенный текст будет сохранен в объекте «`out_txt`». После этого переведенный текст будет выведен на печать.

На следующем этапе выполняется преобразование переведенного текста в аудио дорожку. Здесь переведенный текст, передается в метод `gTTS` пакета `gtts` для преобразования текста в аудио дорожку (объект `text_audio`). Сформированная аудио дорожка записывается на диск в виде файла формата `mp3`. После этого аудио файл озвучивается с использованием библиотеки `playsound`. После завершения проигрывания аудио файл удаляется с диска и начинается обрабатываться следующая страница PDF файла. После завершения данного цикла выполняется закрытие обрабатываемого PDF файла.

Примечание.

Данный модуль не включен в состав исполняемых примеров интерактивной цифровой книги, так как он предназначен для работы на локальном компьютере. Но вы имеете возможность скопировать листинг данной программы, запустить данный программный модуль на выполнение на своем локальном компьютере и использовать в своих приложениях.

Если данный программный код запустить на выполнение на локальном компьютере, то будет получен следующий результат:

```
num_pages= 1
Текущая страница- 0
-----Оригинальный текст-----
Обработка медиа контента
Компьютерное зрение (computer vision) или машинное зрение - это набор программно -
технических средств,
позволяющих компьютерам понимать и обрабатывать визуальную информацию, такую
как изображения или
видео. Являясь составной частью систем искусственного интеллекта, системы компью-
терного зрения
позволяют не только читать, хранить и обрабатывать изображения, но и позволяют обу-
чить компьютеры
принимать решения на основе увиденного. Построенные на основе компьютерного зре-
ния модели помогают
смартфонам узнавать своих владельцев, дорожным камерам распознавать номера авто-
мобилей, медикам
диагностировать болезни, а роботам и беспилотным автомобилям объезжать препят-
ствия.
Перед тем как перейти к решению практических задач , нужно разобраться с некоторы-
ми теоретическими
асpekтами, в частности как компьютер получает, хранит и обрабатывает различные ви-
ды изображений . Из
материалов данной главы вы узнаете:
□ что такое матрица изображения ;
```

☐ как компьютер хранит и обрабатывает черно-белые изображения, а вернее изображения в оттенках

серого цвета ;

☐ как компьютер хранит и обрабатывает цветные (трехканальные) изображения ;

☐ как компьютер хранит и обрабатывает четырехканальные изображения .

-----Переведенный текст-----

Media content processing

Computer vision or machine vision is a set of software and hardware tools allowing computers to understand and process visual information such as images or video. Being an integral part of artificial intelligence systems, computer vision systems allow not only to read, store and process images, but also allow you to train computers make decisions based on what you see. Models built on the basis of computer vision help smartphones to recognize their owners, traffic cameras to recognize car license plates, doctors diagnose diseases, and allow robots and self-driving cars to avoid obstacles.

Before moving on to solving practical problems, you need to understand some theoretical aspects, in particular how a computer receives, stores and processes different types of images. From

materials in this chapter you will learn:

☐ what is an image matrix;

☐ how a computer stores and processes black and white images, or rather images in shades gray;

☐ how a computer stores and processes color (three-channel) images;

☐ how a computer stores and processes four-channel images.

=====КОНЕЦ СТРАНИЦЫ=====

Это тот текст, который был выбран из PDF файла, переведен и озвучен медиа проигрывателем компьютера на английском языке.

Теперь прокомментируем строку с параметром перевода на английский язык, и снимем комментарий со строки с параметром перевода на китайский язык. Если теперь снова запустить данный программный код, то будет получен следующий результат:

num_pages= 1

Текущая страница- 0

-----Оригинальный текст-----

Обработка медиа контента

Компьютерное зрение (computer vision) или машинное зрение - это набор программно-технических средств,

позволяющих компьютерам понимать и обрабатывать визуальную информацию, такую как изображения или

видео. Являясь составной частью систем искусственного интеллекта, системы компьютерного зрения

позволяют не только читать, хранить и обрабатывать изображения, но и позволяют обучить компьютеры

принимать решения на основе увиденного. Построенные на основе компьютерного зрения модели помогают

смартфонам узнавать своих владельцев, дорожным камерам распознавать номера автомобилей, медикам

диагностировать болезни, а роботам и беспилотным автомобилям объезжать препятствия.

Перед тем как перейти к решению практических задач, нужно разобраться с некоторыми теоретическими

асpekтами, в частности как компьютер получает, хранит и обрабатывает различные виды изображений. Из

материалов данной главы вы узнаете:

- ☐ что такое матрица изображения;
- ☐ как компьютер хранит и обрабатывает черно-белые изображения, а вернее изображения в оттенках серого цвета;
- ☐ как компьютер хранит и обрабатывает цветные (трехканальные) изображения;
- ☐ как компьютер хранит и обрабатывает четырехканальные изображения.

-----Переведенный текст-----

媒体内容处理

计算机视觉或机器视觉是一组软件和硬件工具

允许计算机理解和处理视觉信息，例如图像或

视频。作为人工智能系统不可或缺的一部分，计算机视觉系统

不仅可以读取、存储和处理图像，还可以训练计算机

根据您所看到的做出决定。基于计算机视觉构建的模型有帮助

智能手机识别车主、交通摄像头识别汽车牌照、医生

诊断疾病，并允许机器人和自动驾驶汽车避开障碍物。

在继续解决实际问题之前，您需要了解一些理论

方面，特别是计算机如何接收、存储和处理不同类型的图像。从

在本章中您将学到的材料：

- ☐ 什么是图像矩阵；
- ☐ 计算机如何存储和处理黑白图像，或者更确切地说是阴影图像灰色的；
- ☐ 计算机如何存储和处理彩色（三通道）图像；
- ☐ 计算机如何存储和处理四通道图像。

=====КОНЕЦ СТРАНИЦЫ=====

Это тот текст, который был выбран из PDF файла, переведен и озвучен медиа проигрывателем компьютера на китайском языке.

6.7. Модуль Python перевода и озвучивания текста PDF файлов с библиотекой deep-translator

В этом разделе мы узнаем, как с использованием Python можно читать PDF-файлы, а также переводить их текстовое содержимое на различные языки с использованием библиотеки deep-translator и озвучивать переведенный текст.

В данном разделе для перевода текста будет использована библиотека deep-translator версии 1.11.4. Программный модуль, который позволяет выполнить выборку текста из PDF файла, его перевод на другой язык и озвучивание переведенного текста, приведен в листинге 6.11.

Листинг 6.11. Модуль pdf_deep_trans_s.py

Примечание.

В демо-версии книги удалены листинги программ.

Программный этого модуля практически полностью повторяет код предыдущего раздела. Отличие заключается лишь в том, что здесь выполняется импорт модуля GoogleTranslator из библиотеки deep_translator:

```
from deep_translator import GoogleTranslator
```

А так же изменена функция переводчик - translate_text(). В теле функции делается обращение к переводчику GoogleTranslator, а результаты перевода сохраняются в объекте result.

Примечание.

Данный модуль не включен в состав исполняемых примеров интерактивной цифровой книги, так как он предназначен для работы на локальном компьютере. Но вы имеете возможность скопировать листинг данной программы, запустить данный программный модуль на выполнение на своем локальном компьютере и использовать в своих приложениях.

Если данный программный код запустить на выполнение на локальном компьютере, то будет получен следующий результат:

```
num_pages= 1
Текущая страница- 0
-----Оригинальный текст-----
Обработка медиа контента
Компьютерное зрение (computer vision) или машинное зрение - это набор программно -
технических средств,
позволяющих компьютерам понимать и обрабатывать визуальную информацию, такую
как изображения или
видео. Являясь составной частью систем искусственного интеллекта, системы компью-
терного зрения
позволяют не только читать, хранить и обрабатывать изображения, но и позволяют обу-
чить компьютеры
принимать решения на основе увиденного. Построенные на основе компьютерного зре-
ния модели помогают
смартфонам узнавать своих владельцев, дорожным камерам распознавать номера авто-
мобилей, медикам
диагностировать болезни, а роботам и беспилотным автомобилям объезжать препят-
ствия.
Перед тем как перейти к решению практических задач , нужно разобраться с некоторы-
ми теоретическими
```

асpekтами, в частности как компьютер получает, хранит и обрабатывает различные виды изображений . Из

материалов данной главы вы узнаете:

- ☐ что такое матрица изображения ;
- ☐ как компьютер хранит и обрабатывает черно -белые изображения, а вернее изображения в оттенках серого цвета ;
- ☐ как компьютер хранит и обрабатывает цветные (трехканальные) изображения ;
- ☐ как компьютер хранит и обрабатывает четырехканальные изображения .

-----Переведенный текст-----

Processing Media Content

Computer vision or machine vision is a set of software and hardware tools that allow computers to understand and process visual information, such as images or

video. As an integral part of artificial intelligence systems, computer vision systems

not only allow you to read, store and process images, but also allow you to train computers

to make decisions based on what they see. Models built on the basis of computer vision help

smartphones to recognize their owners, traffic cameras to recognize car license plates, doctors

to diagnose diseases, and robots and unmanned vehicles to avoid obstacles.

Before moving on to solving practical problems, you need to understand some theoretical

aspects, in particular, how a computer receives, stores and processes different types of images. From

the materials of this chapter you will learn:

- ☐ what is an image matrix; ☐ how a computer stores and processes black and white images, or rather images in shades of gray;
- ☐ how a computer stores and processes color (three-channel) images;
- ☐ how a computer stores and processes four-channel images.

=====КОНЕЦ СТРАНИЦЫ=====

После завершения перевода текста он будет озвучен на английском языке.

6.8. Голосовое управление компьютером

Давайте представим ситуацию, что вы сидите за компьютером и управляете им не с помощью клавиатуры, а с использованием голосовых команд. Вместо того чтобы открывать браузер и набирать в нем веб адрес сайта вы просто говорите: «Открыть сайт Яндекс», и через секунду открывается браузер с запрошенным сайтом. Или, например, команда «Завершить программу» приведет к закрытию приложения.

Реализация голосового управления компьютером включает в себя несколько шагов:

- ☐ захват звука с микрофона и формирование звуковой дорожки;

- ❑ обработка звуковой дорожки и преобразование аудио сигнала в текст (с указанием языка);
- ❑ анализ полученного текста и выполнение заданной команды.

Создадим программный модуль, который будет выполнять описанные выше шаги, и по команде пользователя, открывать тот или иной сайт 6.12.

Листинг 6.12. Модуль voice_commands.py

Примечание.

В демо-версии книги удалены листинги программ.

Для работы данного программного модуля необходимо выполнить импорт ряда библиотек:

- ❑ `import speech_recognition as sr` - для преобразования речи в текст;
- ❑ `from gtts import gTTS` - для преобразования текста в речь;
- ❑ `from playsound import playsound` - для воспроизведения аудио дорожки;
- ❑ `import webbrowser` - для открытия веб сайтов;
- ❑ `import os` - для работы с файлами;
- ❑ `import sys` для работы с файлами.

После этого создана функция преобразования текста в речь (`def text_to_speech`). Эта функция принимает два параметра: текст - `text`, и язык данного текста - `lang` (в этом примере для текста указан русский язык). Для преобразования текста в речь используется модуль `gTTS`. Звуковая дорожка записывается в объект `tts`, на основе которого на диск записывается звуковой файл формата `mp3`. Здесь же звуковой файл проигрывается с использованием модуля `playsound`.

Работа программы начинается с голосового сообщения - 'запущена программа голосового управления компьютером', после чего запускается бесконечный цикл, в теле которого выполняется обращение к функции `def makeSomething()`, аргументом которой является другая функция – `command()`.

В функции `def command()` создается объект для распознавания речи (`r`) и активируется микрофон для приема входящих голосовых команд. Голосовая команда из аналогового сигнала микрофона преобразуется в оцифрованную звуковую дорожку формата `mp3`, которая принимается объектом `audio`. После этого звуковая дорожка трансформируется в текстовый объект (`zadanie`), который передается в функцию `makeSomething()`. Если команда была озвучена неразборчиво, то пользователю предлагается повторить команду.

В функции `makeSomething()` с использованием инструкций `if` анализируются полученные текстовые команды. Если в одной из инструкций `if` присутствует полученная команда, то она будет выполнено. В частности, в данном примере предлагается либо открыть один из сайтов – с использованием инструкции `webbrowser.open(url)`, либо завершить работу программы – инструкция `sys.exit()`.

Примечание.

Данный модуль не включен в состав исполняемых примеров интерактивной цифровой книги, так как он предназначен для работы на локальном компьютере. Но вы имеете возможность скопировать листинг данной программы, запустить данный программный модуль на выполнение на своем локальном компьютере и использовать в своих приложениях.

Если данный программный код запустить на выполнение на локальном компьютере и задать несколько голосовых команд, то на печать будут выведены следующий строки:

```
я жду вашей команды, говорите
Вы сказали: открыть сайт умник
я жду вашей команды, говорите
Вы сказали: завершить программу
Process finished with exit code 0
```

После начала работы программы компьютер напечатал текст и озвучил голосовое сообщение:

```
я жду вашей команды, говорите
```

После этого пользователь произнес в микрофон фразу: «Открыть сайт умник». Эта фраза была корректно распознана, о чем компьютер сообщил в строке:

```
Вы сказали: открыть сайт умник
```

Команда «открыть сайт умник» присутствует среди команд функции makeSomething():

```
if 'открыть сайт умник' in zadanie:
```

Следовательно, далее сайт с URL адресом url = 'https://umnik.ru' был открыт в браузере компьютера (рис. 6.5).

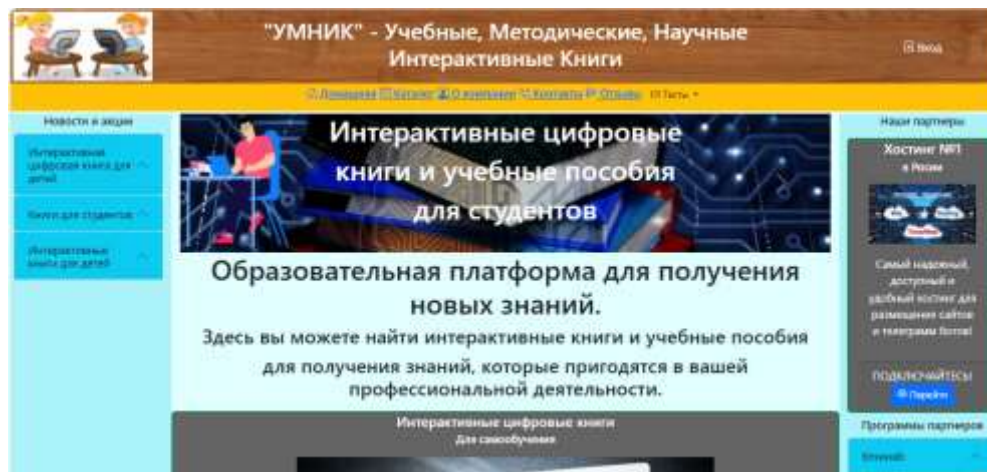


Рис.6.5. Сайт, открытый в окне браузера по голосовой команде

Таким же образом по другим голосовым командам можно открыть сайт академия питон и яндекс.

6.9. Краткие итоги

Итак, мы познакомились с некоторыми методами и программными модулями, которые позволяют выполнять базовые операции обработки PDF файлов на Python. Теперь можно перейти к изучению вопросов реализации этих же методов и подходов в веб приложениях. . В частности нам предстоит узнать, как можно извлекать из PDF документов текстовую информацию и рисунки, выполнять перевод документов на другие языки с выводом результатов в шаблоны Django и озвучивать переведенный текст. Рассмотрению этих вопросов и посвящена следующая глава.

ГЛАВА 7

Перевод и озвучивание PDF файлов в приложениях Django



Нейронные сети позволяют обрабатывать аудио контент (преобразовывать текстовые данные в речь и генерировать текст из аудио данных), а также выполнять перевод текстовой информации на различные языки с возможностью озвучивания. В предыдущей главе такие преобразования выполнялись на локальном компьютере в рамках интерпретатора Python. Здесь же будет рассмотрена возможность выполнения аналогичных операций в веб приложениях, написанных на Django.

Из материалов данной главы вы узнаете:

- ☐ как преобразовать текстовую переменную в речь с пакетом Django-Gtts;
- ☐ как преобразовать текстовую переменную в речь с библиотекой gTTS;
- ☐ как озвучить текст PDF файлов с пакетом Django-gTTS;
- ☐ как озвучить текст PDF файлов с библиотекой gTTS;
- ☐ как выполнить перевод и озвучивание текста PDF файлов с библиотекой gogletrans и пакетом Django-Gtts;
- ☐ как выполнить перевод и озвучивание текста PDF файлов с библиотеками gogletrans и gtts;
- ☐ как выполнить перевод и озвучивание многостраничного текста PDF файлов с пакетом Django-gTTS;
- ☐ как выполнить перевод и озвучивание многостраничного текста PDF файлов с библиотекой gtts;
- ☐ как выполнить перевод и озвучивание текста PDF файлов с переносом изображений с библиотекой gtts.

7.1. Преобразование текста в речь с библиотекой Django-Gtts

Преобразование текста в речь используется для создания голосового вывода на основе текста. Это может быть полезно, когда необходимо не читать текст, а слушать его, например, для пользователей с ограниченным зрением. Кроме того, некоторые инструменты преобразования текста в речь можно использовать для создания реалистичного голоса для видео или рекламы.

В Django для преобразования текста в речь и кэширования сгенерированных файлов в формате mp3 используется пакет Django-Gtts. Для установки данного пакета можно использовать следующую команду:

```
pip install Django-Gtts
```

Это пакет нужно добавить в модуль settings.py

```
INSTALLED_APPS = [  
    ...  
    'gTTS',  
]
```

После этого нужно будет выполнить миграцию:

```
python manage.py makemigrations gTTS  
python manage.py migrate gTTS
```

Примечание.

При использовании пакета Django-Gtts в вашем приложении должна быть создана база данных.

Для активации модулей преобразования текста из текстовой переменной в речь нужно создать следующую ссылку (листинг 7.1):

Листинг 7.1. Ссылка для активации модулей преобразования текста из текстовой переменной в речь

Примечание.

В демо-версии книги удалены листинги программ.

По этой ссылке будет активирован следующий маршрут URL (листинг 7.2):

Листинг 7.2. Маршрут txt_to_speech

Примечание.

В демо-версии книги удалены листинги программ.

По этому маршруту будет вызвано представление views.txt_to_speech (листинг 7.3):

Листинг 7.3. Представление views.txt_to_speech

Примечание.

В демо-версии книги удалены листинги программ.

В данном представлении используются два объекта:

- ❑ txt — текстовая строка, содержащая озвучиваемый текст;
- ❑ ling- язык, на котором необходимо озвучить текстовую строку.

В данном примере фактически задано три текстовой строки и определено три языка для их озвучивания: русский, английский и китайский:

- ❑ txt_ru - текстовая переменная на русском языке;
- ❑ ling_ru - язык озвучивания текста русский;

- ❑ txt_en - текстовая переменная на английском языке;
- ❑ ling_en - язык озвучивания текста английский;
- ❑ txt_cn - текстовая переменная на китайском языке;
- ❑ ling_cn - язык озвучивания текста китайский.

Затем все эти объекты через контекстную переменную context передаются в шаблон txt_to_speech.html, который имеет следующий код (листинг 7.4).

Листинг 7.4. Шаблон txt_to_speech.html

Примечание.

В демо-версии книги удалены листинги программ.

В этом шаблоне присутствует тег для активации API gTTS - {% load gTTS %}. После этого создано три блока, которые содержат аудио проигрыватель:

```
<audio
  src="{% say ling_ru txt_ru %}"
  controls></audio>
<audio
  src="{% say ling_en txt_en %}"
  controls></audio>
<audio
  src="{% say ling_cn txt_cn %}"
  controls></audio>
```

В этом блоке заданы команды озвучить текст (say) на русском, английском и китайском языках (объекты lang_ru, lang_en, lang_cn), и объекты, в которых содержится озвучиваемый текст на данных языках (txt_ru, txt_en, txt_cn).

Примечание.

В интерактивной цифровой книге для запуска данного примера нужно на вкладке «Показ примеров» нажать на кнопку «Примеры главы 7» и активировать ссылку «Текст в речь с Django-Gtts».

Если выполнить данный программный модуль, то в окне браузера отобразится следующий результат (рис. 7.1).

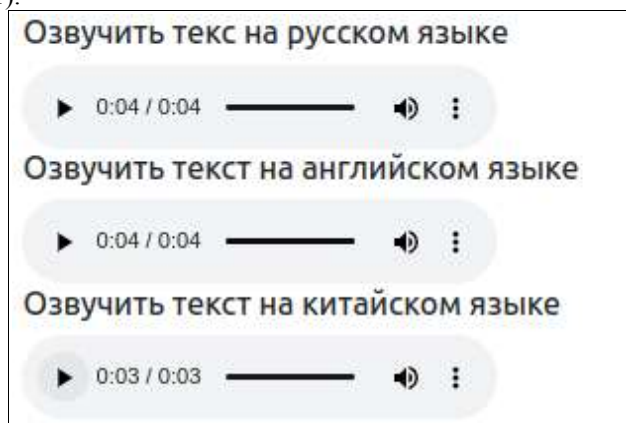


Рис. 7.1. Шаблон `txt_to_speech.html` с проигрывателями для озвучивания текста

Если нажать на кнопку старта проигрывания аудио дорожки одного из проигрывателей, то мы услышим озвученную текстовую фразу на одном из языков (русском, английском или китайском).

Как видно из данного раздела, озвучивать текстовую информацию с пакетом Django-Gtts достаточно просто и без явного создания mp3 файлов. Однако при развертывании приложения на публичном сервере с настройками пакета Django-Gtts могут возникнуть проблемы, решить которые не всегда удастся, особенно если приложение развернуто не на VDS серверах, а на хостинге. Но этих проблем можно избежать, если использовать библиотеку gtts, однако в этом случае придется явно создавать на диске файлы формата mp3.

7.2. Преобразование текста в речь с библиотекой gTTS

В Django для преобразования текста в речь с явным формированием файлов в формат mp3 используется библиотека gTTS. Для установки данной библиотеки можно использовать следующую команду:

```
pip install gTTS
```

Примечание.

При использовании библиотеки gTTS в вашем приложении не обязательно создавать базу данных, но для озвучивания текстовой информации необходимо явно формировать файлы формата mp3.

Для активации модулей преобразования текста из текстовой переменной в речь нужно создать следующую ссылку (листинг 7.5):

Листинг 7.5. Ссылка для активации модулей преобразования текста из текстовой переменной в речь

Примечание.

В демо-версии книги удалены листинги программ.

По этой ссылке будет активирован следующий маршрут URL (листинг 7.6):

Листинг 7.6. Маршрут `txt_to_speech_gtts`

Примечание.

В демо-версии книги удалены листинги программ.

По этому маршруту будет вызвано представление `views.txt_to_speech_gtts` (листинг 7.7):

Листинг 7.7. Представление `views.txt_to_speech_gtts`

Примечание.

В демо-версии книги удалены листинги программ.

В данном представлении выполняется импорт модуля gTTS и формируются пути к папкам с аудио файлами.

Для обращения к модулю gTTS задействовано два объекта:

- ❑ txt — текстовая строка, содержащая озвучиваемый текст;
- ❑ ling - язык, на котором необходимо озвучить текстовую строку.

В данном примере фактически задано три текстовой строки и определено три языка для их озвучивания: русский, английский и китайский:

- ❑ txt_ru - текстовая переменная на русском языке;
- ❑ ling_ru - язык озвучивания текста русский;
- ❑ txt_en - текстовая переменная на английском языке;
- ❑ ling_en - язык озвучивания текста английский;
- ❑ txt_cn - текстовая переменная на китайском языке;
- ❑ ling_cn - язык озвучивания текста китайский.

Затем для каждого языка создается объект sound, с помощью которого создаются три файла формата mp3 (audio_ru, audio_en, audio_cn), которые сохраняются на диске. Для каждого из этих файлов формируется свой URL адрес (url_mp3_ru, url_mp3_en, url_mp3_cn). После этого все три URL адреса через контекстную переменную context передаются в шаблон txt_to_speech_gtts.html, который имеет следующий код (листинг 7.8).

Листинг 7.8. Шаблон txt_to_speech_gtts.html

Примечание.

В демо-версии книги удалены листинги программ.

В этом шаблоне присутствует три блока, которые содержат аудио проигрыватель:

```
<audio controls>
  <source src="{{ url_mp3_ru }}" type="audio/mpeg">
</audio>

<audio controls>
  <source src="{{ url_mp3_en }}" type="audio/mpeg">
</audio>

<audio controls>
  <source src="{{ url_mp3_cn }}" type="audio/mpeg">
</audio>
```

В этом блоке заданы команды показать проигрыватели для озвучивания текст на русском, английском и китайском языках.

Примечание.

В интерактивной цифровой книге для запуска данного примера нужно на вкладке «Показ примеров» нажать на кнопку «Примеры главы 7» и активировать ссылку «Текст в речь с gTTS».

Если выполнить данный программный модуль, то в окне браузера отобразится следующий результат (рис. 7.2).

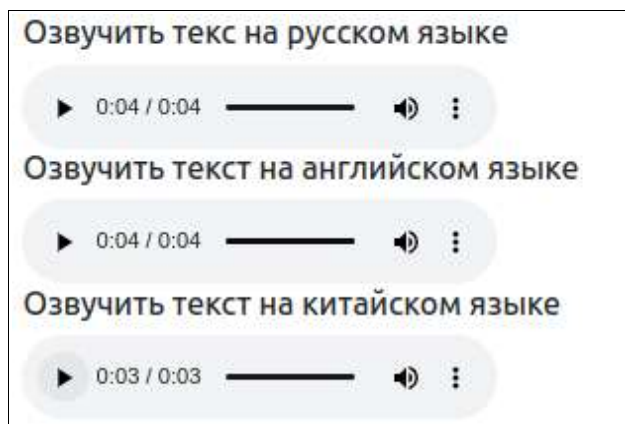


Рис.7.2. Шаблон `txt_to_speech.html` с проигрывателями для озвучивания текста

Если нажать на кнопку старта проигрывания аудио дорожки одного из проигрывателей, то мы услышим озвученную текстовую фразу на одном из языков (русском, английском или китайском).

7.3. Озвучивание PDF файлов с пакетом Django-gTTS

Для того чтобы озвучить содержимое PDF файла, сначала нужно найти в файле фрагменты текста и выгрузить их в текстовую переменную, после чего можно приступить к озвучиванию текстовой информации. Для выборки текстовой информации из PDF файлов можно использовать библиотеку `PyPDF2`, а для преобразования текста в речь - пакет `Django-Gtts`. Для установки библиотеки `PyPDF2` можно использовать следующую команду:

```
pip install PyPDF2
```

Для активации модулей преобразования текста из файла PDF в речь нужно создать следующую ссылку (листинг 7.9).

Листинг 7.9. Ссылка для активации модулей преобразования текста из PDF файла в речь

Примечание.

В демо-версии книги удалены листинги программ.

По этой ссылке будет активирован следующий маршрут URL с именем `pdf_to_speech` (листинг 7.10).

Листинг 7.10. Маршрут `pdf_to_speech`

Примечание.

В демо-версии книги удалены листинги программ.

По этому маршруту будет вызвано представление `views.pdf_to_speech` (листинг 7.11).

Листинг 7.11. Представление `views.pdf_to_speech`

Примечание.

В демо-версии книги удалены листинги программ.

В данном представлении выполнен импорт библиотеки PyPDF2, задается путь к PDF файлу (path_pdf_file) и язык, на котором необходимо озвучить текст, содержащийся в файле (в данном случае это будет русский язык lang_ru = "ru").

После этого выполняется чтение PDF файла (загрузка в объект pdf_reader) и определение количества страниц в файле (num_pages). Затем с использованием инструкции for организован цикл, в теле которого текстовое содержимое PDF файла загружается в объект txt_ru. Затем из текстового объекта txt_ru удаляются символы перехода к следующей строке «\n». Это нужно для того, чтобы удалить ненужные паузы при озвучивании текста (символы перехода к следующей строке иницируют паузу, которая отрицательно влияет на качество озвучивания текста).

На следующем этапе создается контекстный словарь, который принимает текст PDF файла и язык, на котором его необходимо озвучить, и этот словарь передается в шаблон Django с именем pdf_to_speech.html. Этот шаблон имеет следующий код (листинг 7.12).

Листинг 7.12. Шаблон pdf_to_speech.html

Примечание.

В демо-версии книги удалены листинги программ.

В этом шаблоне присутствует тег для активации API gTTS - {% load gTTS %}. После этого создан блок, который содержит аудио проигрыватель:

```
<audio
  src="{% say lang_ru txt_ru %}"
controls></audio>
```

В этом блоке задана команда озвучить текст (say), на русском языке (объект lang_ru), и объект, в котором содержится озвучиваемый текст (txt_ru). Ниже находятся еще один блок для вывода озвучиваемого текста (на русском языке).

Примечание.

В интерактивной цифровой книге для запуска данного примера нужно на вкладке «Показ примеров» нажать на кнопку «Примеры главы 7» и активировать ссылку «Озвучить PDF с Django-Gtts».

Если выполнить данный программный модуль, то в окне браузера отобразится следующий результат (рис. 7.3).

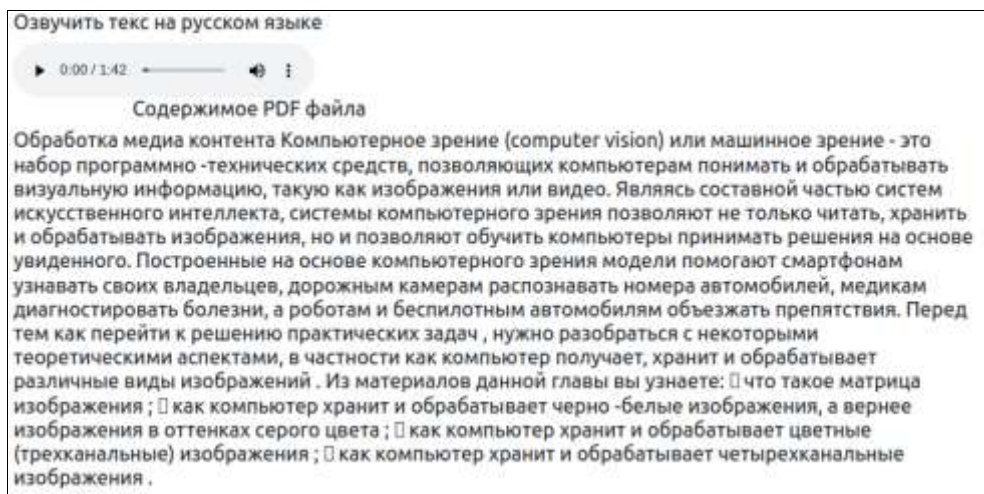


Рис.7.3. Шаблон pdf_to_speech.html с проигрывателем для озвучивания текста

Если нажать на кнопку старта проигрывателя аудио дорожки, то мы услышим озвученное содержимое текста, загруженного из PDF файла на русском языке.

7.4. Озвучивание текста PDF файлов с библиотекой gTTS

Для того чтобы озвучить содержимое PDF файла с библиотекой gTTS, сначала нужно найти в файле фрагменты текста и выгрузить их в текстовую переменную, после чего можно приступить к созданию файла формата mp3 для озвучивания текстовой информации. Для выборки текстовой информации из PDF файлов можно использовать библиотеку PyPDF2, а для преобразования текста в речь - библиотеку gTTS.

Для установки необходимых библиотек можно использовать следующие команды:

```
pip install PyPDF2
pip install gTTS
```

Для активации модулей преобразования текста из файла PDF в речь нужно создать следующую ссылку (листинг 7.13).

Листинг 7.13. Ссылка для активации модулей преобразования текста из PDF файла в речь

Примечание.

В демо-версии книги удалены листинги программ.

По этой ссылке будет активирован маршрут URL с именем pdf_to_speech_gtts (листинг 7.14).

Листинг 7.14. Маршрут pdf_to_speech_gtts

Примечание.

В демо-версии книги удалены листинги программ.

По этому маршруту будет вызвано представление `views.pdf_to_speech_gtts` (листинг 7.15).

Листинг 7.15. Представление `views.pdf_to_speech_gtts`

Примечание.

В демо-версии книги удалены листинги программ.

В данном представлении выполняется импорт модулей PdfReader, gTTS и формируются пути к папкам с аудио файлами и PDF файлами.

Для обращения к модулю gTTS задействовано два объекта:

- `txt_ru` — текстовая строка, содержащая озвучиваемый текст;
- `ling_ru` - язык, на котором необходимо озвучить текстовую строку.

После этого выполняется открытие PDF файла, чтение из него текстовой информации и загрузка текстовых данных в объект `txt_ru`. Из данной текстовой строки удаляются признаки конца строки («\n»). Это делается для того, чтобы при озвучивании текста они не вызывали лишнюю паузу.

Затем создается объект `sound`, с помощью которого формируется файл формата mp3 (`pdf_audio_ru`) и он сохраняется на диске. Для данного файла формируется URL адрес (`url_mp3_ru`). После этого URL адреса и текстовая строка через контекстную переменную `context` передаются в шаблон `pdf_to_speech_gtts.html`, который имеет следующий код (листинг 7.16).

Листинг 7.16. Шаблон `pdf_to_speech_gtts.html`

Примечание.

В демо-версии книги удалены листинги программ.

В этом шаблоне присутствует тег, который содержит аудио проигрыватель:

```
<audio controls>
  <source src="{{ url_mp3_ru }}" type="audio/mpeg">
</audio>
```

А также имеется тег для вывода текстового содержимого PDF файла:

```
<div class="col-6 mx-2 text-start" >
  <h5>{{txt_ru}}</h5>
</div>
```

Примечание.

В интерактивной цифровой книге для запуска данного примера нужно на вкладке «Показ примеров» нажать на кнопку «Примеры главы 7» и активировать ссылку «Озвучить PDF с gTTS».

В меню на сервере – Глава 7/Озвучить PDF с gTTS

Страница на сервере - http://kalinkashop.tw1.ru/api_vision/pdf_to_speech_gtts

Если выполнить данный программный модуль, то в окне браузера отобразится следующий результат (рис. 7.4).

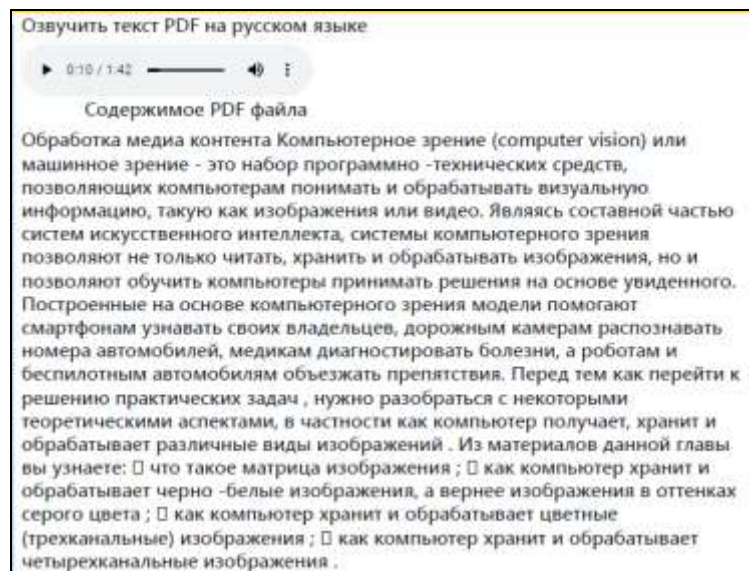


Рис.7.4. Шаблон pdf_to_speech.html с проигрывателем для озвучивания текста

Если нажать на кнопку старта проигрывателя аудио дорожки, то мы услышим озвученное содержимое текста, загруженного из PDF файла на русском языке.

7.5. Перевод и озвучивание PDF файлов с gogletrans и Django-Gtts

Нверняка многие пользовались переводчиком Google Translate в рамках интернет браузера. Это один из лучших переводчиков, который помогает переводить текст на более чем 100 различных языков. Такой перевод возможен благодаря API Google Translate, разработанного компанией Google. Googletrans - это бесплатная библиотека Python, реализующая API Google Translate. Она делает неофициальные вызовы AJAX к API Google Translate для перевода текста с одного языка на другие иностранные языки. Она хорошо работает с Python версиями 3.6 и выше. Начиная с версии 4.0, библиотека googletrans поддерживает асинхронный режим работы программных модулей для Python с версиями 3.8 и выше.

Для разработки программы языкового перевода в Django будем использовать библиотеку googletrans, для получения текста из PDF файла библиотеку PyPDF2, а для озвучивания переведенного текста API gTTS. Библиотека googletrans это бесплатный и свободно используемый пакет, который задействует функции перевода Google Translate API. Для установки библиотеки googletrans можно использовать следующую команду:

```
pip install googletrans
```

Для активации модулей преобразования текста из файла PDF в речь с переводом на другой язык с озвучиванием переведенного текста нужно создать следующую ссылку (листинг 7.17).

Листинг 7.17. Ссылка для активации модулей перевода и озвучивания текста из PDF файла

Примечание.

В демо-версии книги удалены листинги программ.

По этой ссылке будет активирован маршрут URL с именем pdf_translate (листинг 7.18).

Листинг 7.18. Маршрут pdf_translate

Примечание.

В демо-версии книги удалены листинги программ.

По этому маршруту будет вызвано представление views.pdf_translate и функция - переводчик (листинг 7.19).

Листинг 7.19. Представление views.pdf_translate и функция переводчик

Примечание.

В демо-версии книги удалены листинги программ.

Для работы этого программного модуля необходимо выполнить импорт нескольких библиотек:

- ❑ from pathlib import Path - для задания путей к файлам;
- ❑ import os - для задания путей к файлам;
- ❑ from PyPDF2 import PdfReader— для чтения PDF файлов;
- ❑ import asyncio — для работы модулей в асинхронном режиме;
- ❑ from googletrans import Translator — для перевода текста на другие языки.

Для перевода текста с одного языка на другой в асинхронном режиме создана функция (translate_text). Данная функция принимает следующие параметры:

- ❑ txt — исходный текст, который нужно перевести на другой язык;
- ❑ scr — язык исходного текста;
- ❑ dest- язык, на который нужно перевести исходный текст.

В теле функции на основе базового класса Translator создан объект translator. Далее объект translator вызывает метод translate (переводить), которому передаются следующие параметры:

- ❑ исходный текст (txt),
- ❑ язык исходного текста (scr),
- ❑ язык, на который нужно перевести исходный текст (dest).

Результаты перевода будут сохранены в объекте result, который содержит несколько параметров. Для нас важен возвращаемый параметр text, который и будет возвращен из функции в следующей инструкции:

```
return result.text
```

Обратите внимание, что данная функция работает в асинхронном режиме.

В данном представлении задается путь к PDF файлу (`path_pdf_file`) и язык исходного текста, (`lang_ru = "ru"`), в данном случае это будет русский язык.

После этого выполняется чтение PDF файла (загрузка в объект `pdf_reader`) и определение количества страниц в файле (`num_pages`). Затем с использованием инструкции `for` организован цикл, в теле которого текстовое содержимое PDF файла загружается в объект `txt_ru`. Затем из текстового объекта `txt_ru` удаляются символы перехода к следующей строке «\n». Это нужно для того, чтобы удалить ненужные паузы при озвучивании текста (символы перехода к следующей строке инициируют паузу, которая отрицательно влияет на качество озвучивания текста).

После этого задается язык, на который нужно перевести текст и делается обращение к функции перевода текста (`translate_text`)

```
# Итоговый язык перевода английский
my_dest = 'en'
# Итоговый язык перевода китайский
# my_dest = 'zh-cn'
# Обращение к функции перевода
out_txt = asyncio.run(translate_text(txt_ru, lang_ru, my_dest))
```

В данном модуле предполагается перевод на один из двух языков — английский и китайский (параметр перевода на китайский язык закомментирован). В функцию `translate_text` переданы следующие аргументы:

- ❑ `txt_ru` — текст исходного языка;
- ❑ `lang_ru` — язык, на котором написан исходный текст;
- ❑ `my_dest` - язык, на который нужно перевести исходный текст.

Переведенный текст будет передан объекту `out_txt`.

На следующем этапе создается контекстный словарь, который принимает переведенный текст (`out_txt`) и язык, на котором его необходимо озвучить (`my_dest`), и этот словарь передается в шаблон Django с именем `pdf_translate.html` (листинг 7.20).

Листинг 7.20. Шаблон `pdf_translate.html`

Примечание.

В демо-версии книги удалены листинги программ.

В этом шаблоне присутствует тег для активации API gTTS - `{% load gTTS %}`. После этого создан блок, который содержит аудио проигрыватель:

```
<audio
  src="{% say lang_ru txt_ru %}"
controls></audio>
```

В этом блоке задана команда озвучить текст (`say`) на заданном языке (объект `my_dest`), и объект, в котором содержится озвучиваемый текст (`out_txt`). Ниже находятся еще один блок для вывода переведенного текста.

Примечание.

В интерактивной цифровой книге для запуска данного примера нужно на вкладке «Показ примеров» нажать на кнопку «Примеры главы 7» и активировать ссылку «Перевести и озвучить PDF с Django-Gtts».

Если выполнить данный программный модуль, то в окне браузера отобразится следующий результат (рис. 7.5).

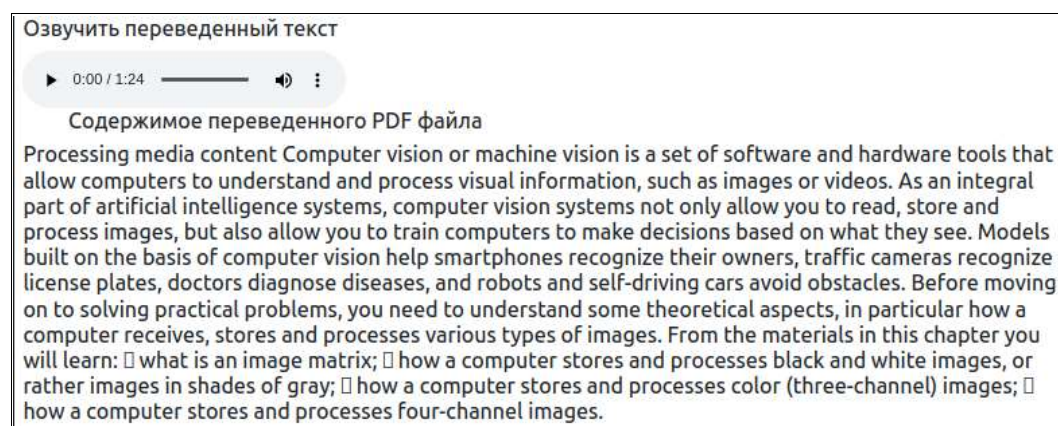


Рис.7.5. Шаблон pdf_translate.html с проигрывателем для озвучивания текста на английском языке

Как видно из данного рисунка, программный модуль перевел текст PDF файла с русского языка на английский. Если нажать на кнопку старта проигрывателя аудио дорожки, то мы услышим озвученное содержимое текста, загруженного из PDF файла на английском языке.

Теперь поставим комментарии на строки активации английского языка, и снимем комментарии со строк активации китайского языка:

```
# Итоговый язык перевода английский
# my_dest = 'en'
# Итоговый язык перевода китайский
my_dest = 'zh-cn'
```

Если после этих изменений выполнить данный программный модуль, то в окне браузера отобразится следующий результат (рис. 7.6):



Рис.7.6. Шаблон pdf_translate.html с проигрывателем для озвучивания текста на китайском языке

Как видно из данного рисунка, программный модуль перевел текст PDF файла с русского языка на китайский язык. Если нажать на кнопку старта проигрывателя аудио дорожки, то мы услышим озвученное содержимое текста, загруженного из PDF файла на китайском языке.

Примечание.

Для арабского языка необходимо скачать набор шрифтов Scheherazade, распаковать полученный файл, а затем файл с именем Scheherazade-Regular.ttf перенести в папку проекта – `venv/lib/site-packages/reportlab/fonts/`.

7.6. Перевод и озвучивание PDF файлов с googletrans и gtts

Для разработки программы языкового перевода в Django будем использовать библиотеку googletrans, для получения текста из PDF файла библиотеку PyPDF2, а для озвучивания переведенного текста библиотеку gtts. Библиотека googletrans это бесплатный и свободно используемый пакет, который задействует функции перевода Google Translate API. Для установки библиотеки googletrans можно использовать следующую команду:

```
pip install googletrans
```

Для активации модулей преобразования текста из файла PDF в речь с переводом на другой язык и озвучивание переведенного текста нужно создать следующую ссылку (листинг 7.21).

Листинг 7.21. Ссылка для активации модулей перевода и озвучивания текста из PDF файла

Примечание.

В демо-версии книги удалены листинги программ.

По этой ссылке будет активирован маршрут URL с именем `pdf_translate_gtts` (листинг 7.22).

Листинг 7.22. Маршрут pdf_translate_gtts

Примечание.

В демо-версии книги удалены листинги программ.

По этому маршруту будет вызвано представление `views.pdf_translate_gtts` и функция - переводчик (листинг 7.23).

Листинг 7.23. Представление views.pdf_translate_gtts и функция переводчик

Примечание.

В демо-версии книги удалены листинги программ.

Для работы этого программного модуля необходимо выполнить импорт нескольких библиотек:

- ❑ `from pathlib import Path` - для задания путей к файлам;
- ❑ `import os` - для задания путей к файлам;

- ❑ `from PyPDF2 import PdfReader` — для чтения PDF файлов
- ❑ `import asyncio` — для работы модулей в асинхронном режиме;
- ❑ `from googletrans import Translator` — для перевода текста на другие языки.

Для перевода текста с одного языка на другой в асинхронном режиме создана функция (`translate_text`). Данная функция принимает следующие параметры:

- ❑ `txt` — исходный текст, который нужно перевести на другой язык;
- ❑ `scr` — язык исходного текста;
- ❑ `dest` — язык, на который нужно перевести исходный текст.

В теле функции на основе базового класса `Translator` создан объект `translator`. Далее объект `translator` вызывает метод `translate` (переводить), которому передаются следующие параметры:

- ❑ исходный текст (`txt`),
- ❑ язык исходного текста (`scr`),
- ❑ язык, на который нужно перевести исходный текст (`dest`).

Результаты перевода будут сохранены в объекте `result`, который содержит несколько параметров. Для нас важен возвращаемый параметр `text`, который и будет возвращен из функции в следующей инструкции:

```
return result.text
```

Обратите внимание, что данная функция работает в асинхронном режиме.

В данном представлении задается путь к PDF файлу (`path_pdf_file`) и язык исходного текста, (`lang_ru = "ru"`), в данном случае это будет русский язык.

После этого выполняется чтение PDF файла (загрузка в объект `pdf_reader`) и определение количества страниц в файле (`num_pages`). Затем с использованием инструкции `for` организован цикл, в теле которого текстовое содержимое PDF файла загружается в объект `txt_ru`.

Затем из текстового объекта `txt_ru` удаляются символы перехода к следующей строке «\n». Это нужно для того, чтобы удалить ненужные паузы при озвучивании текста (символы перехода к следующей строке инициируют паузу, которая отрицательно влияет на качество озвучивания текста).

После этого задается язык, на который нужно перевести текст и делается обращение к функции перевода текста (`translate_text`)

```
# Итоговый язык перевода английский
my_dest = 'en'
# Итоговый язык перевода китайский
# my_dest = 'zh-cn'
# Обращение к функции перевода
out_txt = asyncio.run(translate_text(txt_ru, lang_ru, my_dest))
```

В данном модуле предполагается перевод на один из двух языков — английский и китайский (параметр перевода на китайский язык закомментирован). В функцию `translate_text` переданы следующие аргументы:

- ❑ `txt_ru` — текст исходного языка;
- ❑ `lang_ru` — язык, на котором написан исходный текст;
- ❑ `my_dest` — язык, на который нужно перевести исходный текст.

Переведенный текст будет передан объекту out_txt.

Затем создается объект sound, с помощью которого формируется файл формата mp3 (pdf_audio_trans) и он сохраняется на диске. Для данного файла формируется URL адрес (url_mp3). После этого URL адреса и текстовая строка через контекстную переменную context передаются в шаблон pdf_to_speech_gtts.html, который имеет следующий код (листинг 7.24).

Листинг 7.24. Шаблон pdf_translate_gtts.html

Примечание.

В демо-версии книги удалены листинги программ.

В этом шаблоне присутствует тег, который содержит аудио проигрыватель:

```
<audio controls>
  <source src="{{ url_mp3 }}" type="audio/mpeg">
</audio>
```

А также имеется тег для вывода переведенного содержимого PDF файла:

```
<div class="col-6 mx-2 text-start" >
  <h5>{{out_ru}}</h5>
</div>
```

Примечание.

В интерактивной цифровой книге для запуска данного примера нужно на вкладке «Показ примеров» нажать на кнопку «Примеры главы 7» и активировать ссылку «Перевести и озвучить с gTTS».

Если выполнить данный программный модуль, то в окне браузера отобразится следующий результат (рис. 7.7).

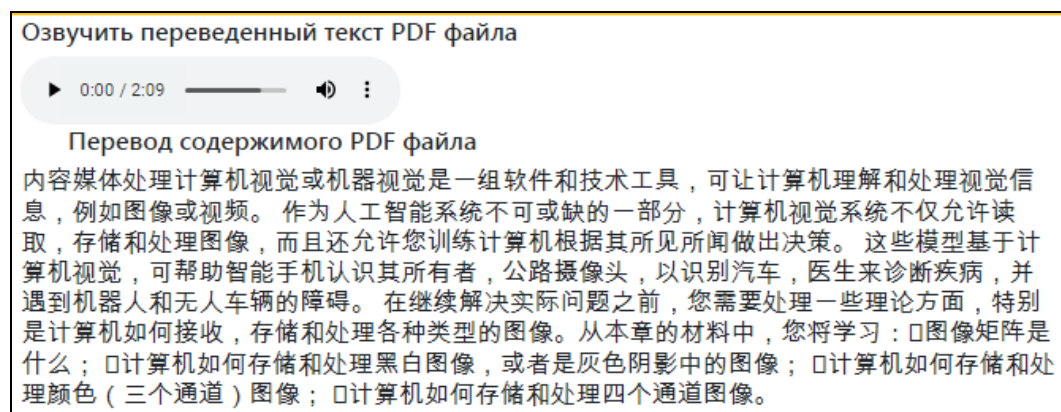


Рис.7.7. Шаблон pdf_translate.html с проигрывателем для озвучивания текста на китайском языке

Как видно из данного рисунка, программный модуль перевел текст PDF файла с русского языка на китайский язык. Если нажать на кнопку старта проигрывателя аудио дорожки, то

мы услышим озвученное содержимое текста, загруженного из PDF файла на китайском языке.

Следует отметить, что перевод текста с одного языка на другой и формирование аудио дорожки занимает некоторое время, которое может быть продолжительным при большом количестве страниц в PDF файле. Поэтому желательно дать возможность пользователю выбрать одну из страниц загруженного файла, и выбрать язык, на который нужно перевести текст выбранной страницы. Также следует учитывать, что у библиотеки `googlettrans` есть ограничение по количеству символов в переводимом тексте, поэтому иногда страницы с большим количеством строк приходится разбивать на части, переводить каждую часть, а затем «склеивать» переведенные фрагменты текста. Как реализовать такую возможность будет показано в следующем разделе.

7.7. Перевод и озвучивание многостраничных PDF файлов с пакетом Django-gTTS

В предыдущем разделе для перевода и озвучивания текста был загружен файл PDF с одной страницей, время обработки такого файла занимает 10-15 сек. А если файл имеет порядка 500 страниц текста, то его обработка может затянуться на несколько часов. В этом случае пользователь может обрабатывать такой файл в постраничном режиме.

Для того чтобы реализовать обработку многостраничных PDF файлов с выбором страницы и сменой языка перевода необходимо создать форму. Форма должна содержать поля для смены номера страницы, выбора языка перевода и кнопки активации перевода с озвучиванием. Номера страниц должны быть представлены целыми числами, а языки перевода текста - в виде выпадающего списка.

Для начала необходимо создать форму для предоставления пользователю возможности выбора страницы текста и указания языка перевода (листинг 7.25).

Листинг 7.25. Форма `LangForm`

Примечание.

В демо-версии книги удалены листинги программ.

В этой форме создан список с перечнем языков, на которые можно переводить текст, и задано два поля:

- ☐ `lang` — язык, на который нужно перевести текст;
- ☐ `num_str` — номер страницы.

Для активации модулей преобразования текста из многостраничного файла PDF в речь с переводом на другой язык нужно создать следующую ссылку (листинг 7.26).

Листинг 7.26. Ссылка для активации модулей перевода и озвучивания текста из многостраничного PDF файла

Примечание.

В демо-версии книги удалены листинги программ.

По этой ссылке будет активирован маршрут URL с именем pdf_long (листинг 7.27):

Листинг 7.27. Маршрут pdf_long

Примечание.

В демо-версии книги удалены листинги программ.

По этому маршруту будет вызвано представление views.pdf_long (листинг 7.28).

Листинг 7.28. Представление pdf_long

Примечание.

В демо-версии книги удалены листинги программ.

В этом представлении сначала создан словарь (dic_sound), который содержит информацию о тех языках, на которых может быть озвучен текст с использованием API gTTS. Это словарь необходим для того, чтобы в ходе выполнения программы можно было убедиться в том, что переведенный текст будет озвучен (библиотека перевода текста поддерживает более широкий список языков, чем библиотека озвучивания текста).

После этого создаются следующие объекты:

- ❑ path_pdf_file = os.path.join(media_dir, 'txt_pdf/long_txt.pdf') - путь к текстовому файлу (на русском языке);
- ❑ lang_ru = "ru" - язык исходного текста (русский);
- ❑ url_pdf = '/media/txt_pdf/long_txt.pdf' — URL адрес исходного файла PDF (в папке media).

Затем выполняется открытие и чтение содержимого файла PDF (в объект pdf_reader) и определение количества страниц текста (объект num_pages). Пропускаем строки программы, которые находятся в теле инструкции if, так как они выполняются только после того, как пользователь выгрузил данные из формы (POST-запрос).

Следуя логике программы, в следующих строках выполняется чтение первой страницы из файла PDF в объект txt_ru:

```
tek_page = 0
page = pdf_reader.pages[tek_page]
txt_ru = page.extract_text()
```

Обратите внимание, что первая страница файла PDF будет иметь нулевой индекс. Поскольку пользователю необходимо показать не индекс первой страницы, а ее истинный номер, то в объекте tek_page номер страницы увеличиваем на единицу. Кроме того, чтобы при озвучивании текста не возникали лишние паузы, мы удаляем из текста символы переноса строк («\n»)

```
tek_page = tek_page + 1
# Удалить символы переноса строки - \n
txt_ru = txt_ru.replace("\n", "")
```

После этого создается объект my_dest, он содержит язык, который по умолчанию будет использоваться для перевода текста (английский), и выполняется обращение к функции перевода текста:

```
# Язык перевода на английский
my_dest = 'en'
# Обращение к функции перевода
out_txt = asyncio.run(translate_text(txt_ru, lang_ru, my_dest))
```

В функцию перевода текста передаются следующие аргументы:

- ❑ `txt_ru` — исходный текст (на русском языке);
- ❑ `lang_ru` — язык исходного текста;
- ❑ `my_dest` — язык, на который нужно перевести текст (в данном случае английский).

Переведенный текст будет сохранен в объекте `out_txt`.

После этого создан объект форма (`form`) для изменения параметров обработки текста. В данном модуле задан язык перевода английский, а мы видим, что ключ для английского языка (`en`) присутствует в словаре допустимых языком для озвучивания текстов, поэтому признаку гарантированного озвучивания текста присваиваем истинное значение (`sound=True`).

Далее создается контекстный словарь (`context`), включающий все необходимые сформированные данные, который передается в шаблон `pdf_long.html` (листинг 7.29).

Листинг 7.29. Шаблон `pdf_long.html`

Примечание.

В демо-версии книги удалены листинги программ.

В начале данного шаблона созданы два важных тега:

- ❑ `{% load gTTS %}` - подключение API gTTS (для озвучивания голосовой дорожки);
- ❑ `{% load static %}` - для загрузки файла PDF из папки `static`.

В данном примере файл PDF не закачен пользователем, а загружен в приложении на этапе проектирования, поэтому он передается в шаблон не из папки `media` по его URL адресу, а из папки `static`.

Затем в шаблоне созданы две вкладки: первая для постраничного перевода и озвучивания текста, вторая для вывода оригинального содержимого файла PDF.

На первую вкладку выводится форма для изменения параметров перевода и озвучивания текста, а также оригинальный и переведенный текст. Вторая вкладка предназначена для показа текста страницы оригинального PDF файла.

Перед озвучиванием переведенного текста в теге `if` делается проверка, есть ли для выбранного пользователем языка соответствующий аудио движок в API gTTS. Если такой движок есть, то в шаблоне будет отображен аудио проигрыватель.

Примечание.

В интерактивной цифровой книге для запуска данного примера нужно на вкладке «Показ примеров» нажать на кнопку «Примеры главы 7» и активировать ссылку «Перевести и озвучить большой PDF с Django-Gtts».

Если выполнить данный программный модуль, то в окне браузера отобразится следующий результат (рис. 7.8).

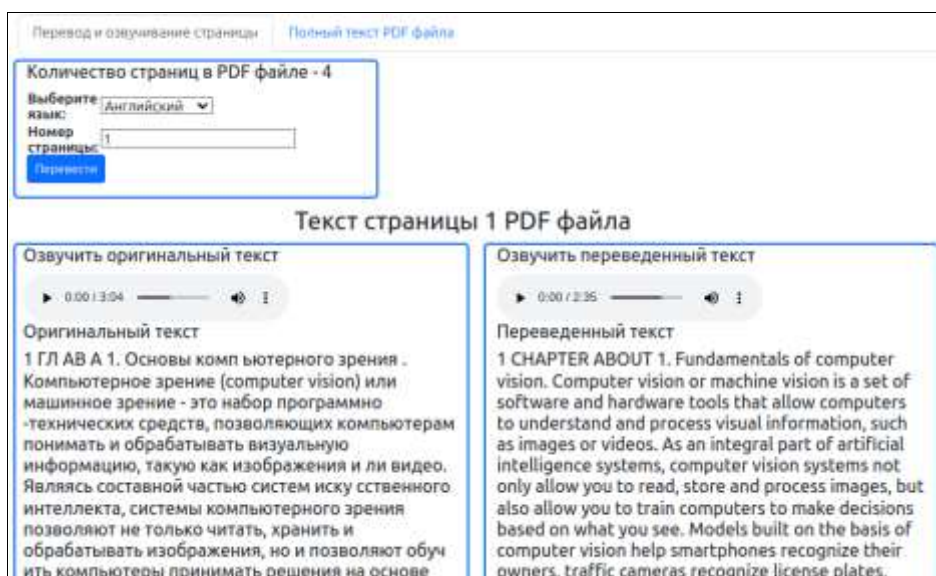


Рис.7.8. Шаблон pdf_long.html с проигрывателями для озвучивания текста (вкладка «Перевод и озвучивание страницы», язык - английский)

Как видно из данного рисунка, шаблон имеет две вкладки: «Перевод и озвучивание страницы», «Полный текст PDF файла». На вкладке «Перевод и озвучивание страницы» имеется форма для выбора языка перевода и смены номера страницы, блок с текстом текущей страницы на русском языке и проигрывателем, блок с переведенным текстом текущей страницы на английский язык и проигрывателем.

Если перейти на вторую вкладку - «Полный текст PDF файла», то на ней будет показано полное содержание файла PDF (рис. 7.9).

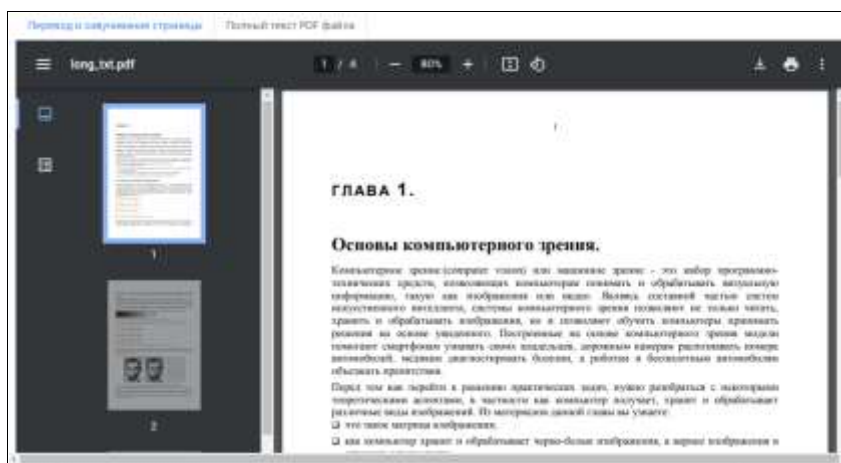


Рис.7.9. Шаблон pdf_long.html с полным текстом PDF файла

Теперь выберем в форме другой язык (китайский) и другую страницу (4-ю). После нажатия на кнопку «Перевести» будет получен следующий результат (рис. 7.10).

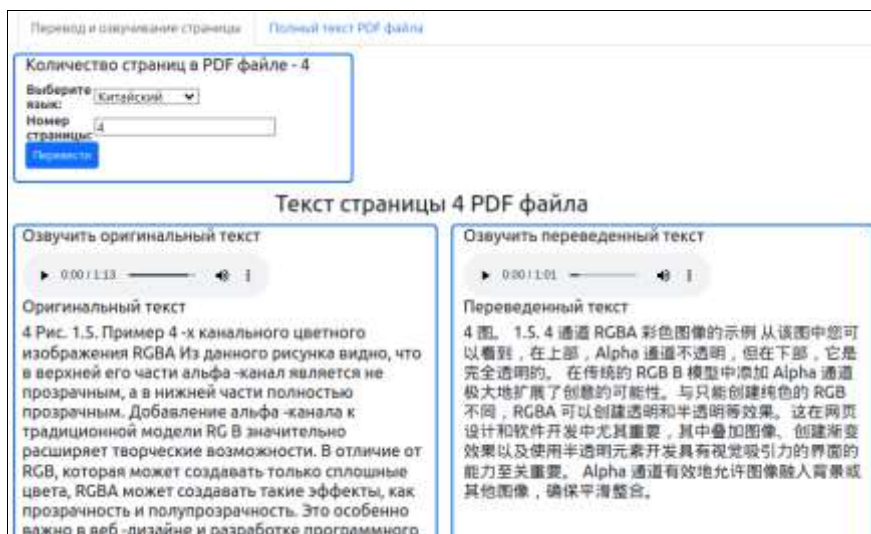


Рис.7.10. Шаблон pdf_long.html с проигрывателями для озвучивания текста (вкладка «Перевод и озвучивание страницы», язык - китайский)

Если теперь нажать кнопку «старт» проигрывателя в левом блоке, то будет озвучен текст на русском языке. Проигрыватель в правом блоке озвучит текст на китайском языке. Во время проигрывания аудио дорожки пользователь может перейти на вторую вкладку и просматривать оригинальный текст и рисунки.

При нажатии на кнопку «Перевести» программа возвращается к представлению pdf_long, где начинает работать код POST - запроса (блок if request.method == 'POST'). Здесь сначала из формы передаются параметры обрабатываемого текста, которые сохраняются в следующих объектах:

- ☐ my_dest — язык, на который нужно выполнить перевод текста;
- ☐ tek_page — номер страницы, которую нужно перевести и озвучить.

После этого (если пользователь задал не существующий номер мтраницы) делается коррективровка указанного пользователем номера страницы:

- ☐ если пользователь указал отрицательный номер страницы, то принудительно задается номер первой страницы;
- ☐ если пользователь указал номер страницы больший, чем имеется в тексте, то принудительно задается номер последней страницы.

Затем выполняется загрузка текста выбранной страницы в объект txt_ru, удаление из текста признаков конца строки и обращение к функции перевода текста на выбранный пользователем язык. Результаты перевода текста сохраняются в объекте out_txt.

На следующем шаге в инструкции if проверяется возможность озвучивания текста на выбранном языке:

```
if my_dest in dic_sound:
    sound = True
else:
    sound = False
```

После этого формируется контекстный словарь `context` и делается возврат в шаблон `pdf_long.html`.

Если пользователь указал язык, который не поддерживается в API gTTS (например, казахский), то в представлении признак `sound` получит значение `False` и в шаблоне `pdf_long.html` не будет активирован тег с проигрывателем на данном языке (рис. 7.11).

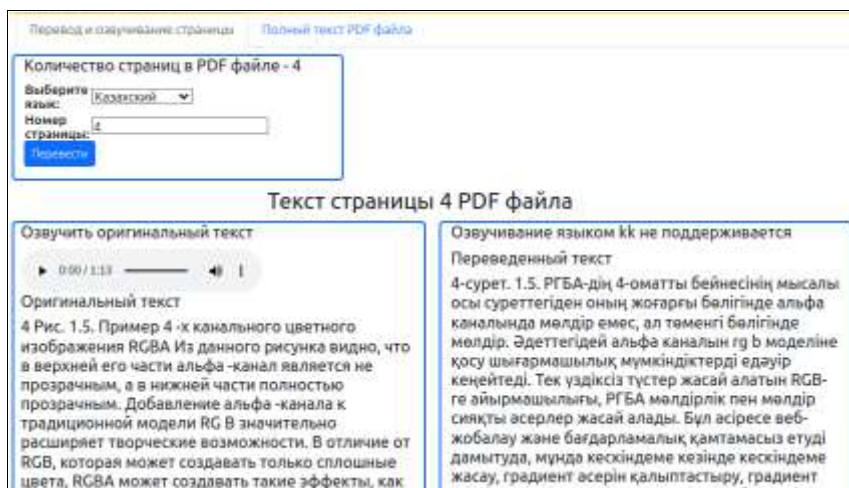


Рис.7.11. Шаблон `pdf_long.html` без проигрывателя для озвучивания текста (вкладка «Перевод и озвучивание страницы», язык - казахский)

Как видно из данного рисунка, программой выполнен только перевод выбранной страницы на казахский язык, а аудио проигрыватель не показан в блоке с текстом на казахском языке.

7.8. Перевод и озвучивание многостраничного текста PDF файлов с библиотекой `gtts`

В программных модулях данного раздела будет использована форма `LangForm`, которая описана в предыдущем разделе (листинг 9.13).

Для активации модулей преобразования текста из многостраничного файла PDF в речь с переводом на другой язык нужно создать следующую ссылку (листинг 7.30).

Листинг 7.30. Ссылка для активации модулей перевода и озвучивания текста из многостраничного PDF файла

Примечание.

В демо-версии книги удалены листинги программ.

По этой ссылке будет активирован маршрут URL с именем `pdf_long_gtts` (листинг 7.31).

Листинг 7.31. Маршрут `pdf_long_gtts`

Примечание.

В демо-версии книги удалены листинги программ.

По этому маршруту будет вызвано представление `views.pdf_long_gtts` (листинг 7.32).

Листинг 7.32. Представление `pdf_long_gtts`

Примечание.

В демо-версии книги удалены листинги программ.

В этом представлении выполнен импорт необходимых библиотек, заданы пути к папкам и файлам, а так же создан словарь (`dic_sound`), который содержит информацию о тех языках, на которых может быть озвучен текст с использованием библиотеки `gtts`. Это словарь необходим для того, чтобы в ходе выполнения программы можно было убедиться в том, что переведенный текст будет озвучен (библиотека перевода текста поддерживает более широкий список языков, чем библиотека озвучивания текста).

После этого создаются следующие объекты:

- ❑ `path_pdf_file = os.path.join(media_dir, 'txt_pdf/long_txt.pdf')` - путь к текстовому файлу (на русском языке);
- ❑ `lang_ru = "ru"` - язык исходного текста (русский);
- ❑ `url_pdf = '/media/txt_pdf/long_txt.pdf'` — URL адрес исходного файла PDF (в папке `media`).

Содержимое исходного PDF файла загружается в объект `pdf`, который преобразуется из двоичного формата в строковый (объект `pdf_content`).

Затем выполняется открытие и чтение содержимого файла PDF (в объект `pdf_reader`) и определение количества страниц текста (объект `num_pages`). Пропускаем строки программы, которые находятся в теле инструкции `if`, так как они выполняются только после того, как пользователь выгрузил данные из формы.

Следуя логике программы, в следующих строках выполняется чтение первой страницы из файла PDF в объект `txt_ru`:

```
tek_page = 0
page = pdf_reader.pages[tek_page]
txt_ru = page.extract_text()
```

Обратите внимание, что первая страница файла PDF будет иметь нулевой индекс. Поскольку пользователю необходимо показать не индекс первой страницы, а ее истинный номер, то в объекте `tek_page` номер страницы увеличиваем на единицу. Кроме того, чтобы при озвучивании текста не возникали лишние паузы, мы удаляем из текста символы переноса строк («\n»)

```
tek_page = tek_page + 1
# Удалить символы переноса строки - \n
txt_ru = txt_ru.replace("\n", "")
```

После этого создается файл формата `mp3` (`long_audio_ru.mp3`), он записывается на диск и формируется URL адрес этого файла (`url_mp3_ru_long`).

Затем создается объект `my_dest`, он содержит язык, который по умолчанию будет использоваться для перевода текста (английский), и выполняется обращение к функции перевода текста:

```
# Язык перевода на английский
my_dest = 'en'
# Обращение к функции перевода
out_txt = asyncio.run(translate_text(txt_ru, lang_ru, my_dest))
```

В функцию перевода текста передаются следующие аргументы:

- ❑ `txt_ru` — исходный текст (на русском языке);
- ❑ `lang_ru` — язык исходного текста;
- ❑ `my_dest` — язык, на который нужно перевести текст (в данном случае английский).

Переведенный текст будет сохранен в объекте `out_txt`.

Затем создается файл формата mp3 (`long_audio_trans.mp3`), он записывается на диск и формируется URL адрес этого файла (`url_mp3_trans_long`). Этот файл представляет собой звуковую дорожку переведенного текста.

После этого создан объект форма (`form`) для изменения параметров обработки текста. В данном модуле задан язык перевода английский, а мы знаем, что ключ для английского языка (`en`) присутствует в словаре допустимых языком для озвучивания текстов, поэтому признаку гарантированного озвучивания текста присваиваем истинное значение (`sound=True`).

Далее создается контекстный словарь (`context`), включающий все необходимые сформированные данные, который передается в шаблон `pdf_long_gtts.html` (листинг 7.33).

Листинг 7.33. Шаблон `pdf_long_gtts.html`

Примечание.

В демо-версии книги удалены листинги программ.

В данном шаблоне созданы две вкладки: первая для постраничного перевода и озвучивания текста, вторая для вывода оригинального содержимого файла PDF.

На первую вкладку выводится форма для изменения параметров перевода и озвучивания текста, а также оригинальный и переведенный текст. Перед озвучиванием переведенного текста в теге `if` делается проверка, есть ли для выбранного пользователем языка соответствующий аудио движок в API gTTS. Если такой движок есть, то в шаблоне будет отображен аудио проигрыватель. Вторая вкладка предназначена для показа текста страницы оригинального PDF файла.

Кнопка с надписью «Перевести и озвучить» имеет скрытый спин или вращающееся кольцо, для анимации которого в конце данной страницы имеется `java` — скрипт.

Примечание.

В интерактивной цифровой книге для запуска данного примера нужно на вкладке «Показ примеров» нажать на кнопку «Примеры главы 7» и активировать ссылку «Перевести и озвучить большой PDF с gTTS».

Если выполнить данный программный модуль, то в окне браузера отобразится следующий результат (рис. 7.12).

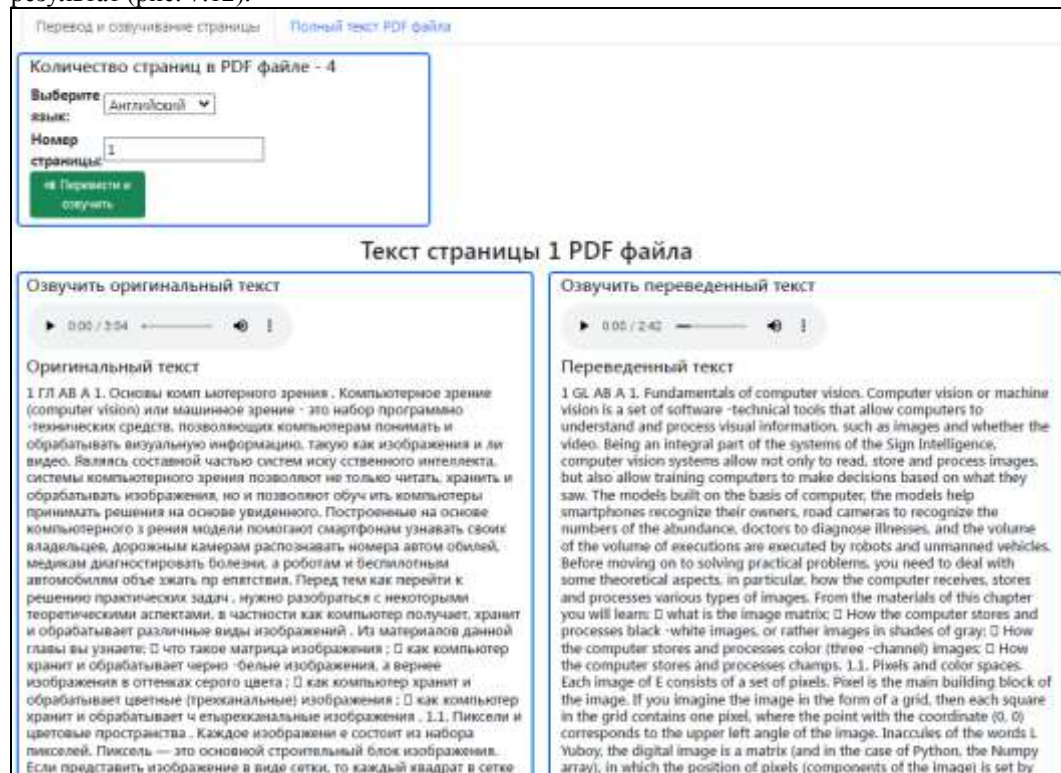


Рис.7.12. Шаблон pdf_long.html с проигрывателями для озвучивания текста (вкладка «Перевод и озвучивание страницы», язык - английский)

Как видно из данного рисунка, шаблон имеет две вкладки: «Перевод и озвучивание страницы», «Полный текст PDF файла». На вкладке «Перевод и озвучивание страницы» имеется форма для выбора языка перевода и смены номера страницы, блок с текстом текущей страницы на русском языке и проигрывателем, блок с переведенным текстом текущей страницы на английский язык и проигрывателем.

Если перейти на вторую вкладку - «Полный текст PDF файла», то на ней будет показано полное содержание файла PDF (рис. 7.13).

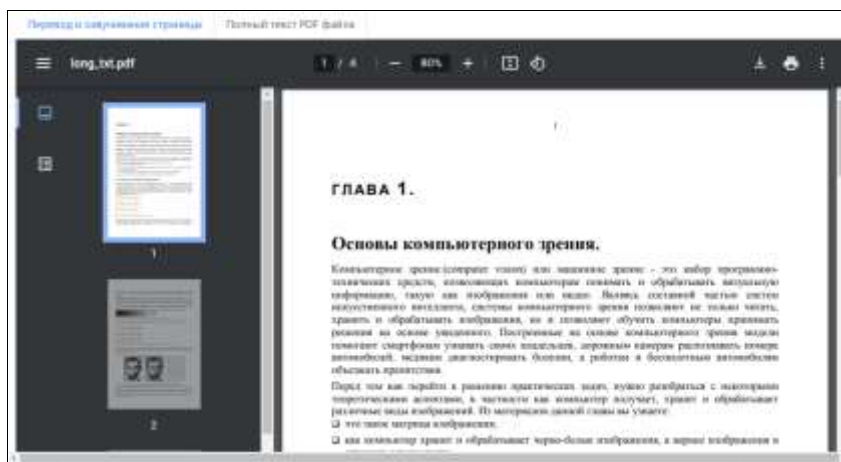


Рис.7.13. Шаблон pdf_long.html с полным текстом PDF файла

Теперь выберем в форме другой язык (китайский) и другую страницу (4-ю). После нажатия на кнопку «Перевести» будет получен следующий результат (рис. 7.14).

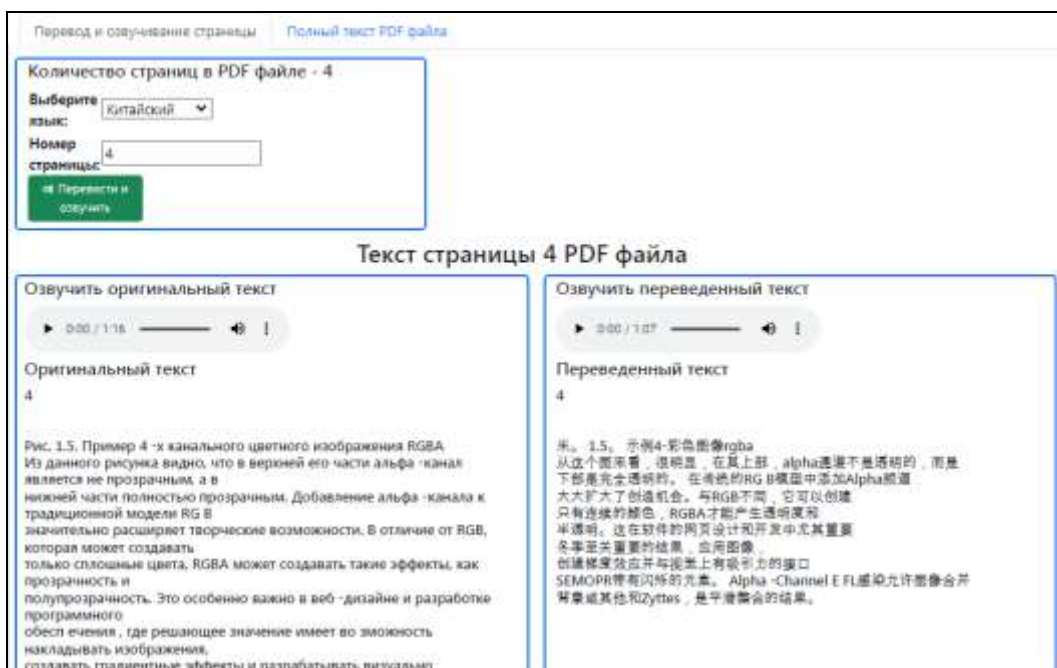


Рис.7.14. Шаблон pdf_long.html с проигрывателями для озвучивания текста (вкладка «Перевод и озвучивание страницы», язык - китайский)

Если теперь нажать кнопку «старт» проигрывателя в левом блоке, то будет озвучен текст на русском языке. Проигрыватель в правом блоке озвучит текст на китайском языке. Во время проигрывания аудио дорожки пользователь может перейти на вторую вкладку и просматривать оригинальный текст и рисунки.

При нажатии на кнопку «Перевести» программа возвращается к представлению `pdf_long_gtts`, где начинает работать код POST - запроса (блок `if request.method == 'POST'`). Здесь сначала из формы передаются параметры обрабатываемого текста, которые сохраняются в следующих объектах:

- ☐ `my_dest` — язык, на который нужно выполнить перевод текста;
- ☐ `tek_page` — номер страницы, которую нужно перевести и озвучить.

После этого (если пользователь задал не существующий номер страницы) делается корректировка выбранной пользователем номера страницы:

- ☐ если пользователь указал отрицательный номер страницы, то принудительно задается номер первой страницы;
- ☐ если пользователь указал номер страницы больший, чем имеется в тексте, то принудительно задается номер последней страницы.

Затем выполняется загрузка текста выбранной страницы в объект `txt_ru`, из текста удаляются признаки конца строки, и формируется файл формата `mp3` для озвучивания текста страницы на русском языке. Здесь также создается URL адрес созданной аудио дорожки.

После этого выполняется обращение к функции перевода текста на выбранный пользователем язык. Результаты перевода текста сохраняются в объекте `out_txt`.

На следующем шаге в инструкции `if` проверяется возможность озвучивания текста на выбранном языке:

```
if my_dest in dic_sound:
    sound = True
else:
    sound = False
```

Если приложением поддерживается озвучивание текста на выбранном языке, формируется файл формата `mp3` для озвучивания текста переведенной страницы. Здесь также создается URL адрес созданной аудио дорожки.

После этого формируется контекстный словарь `context` и делается возврат в шаблон `pdf_long_gtts.html`.

Если пользователь указал язык, который не поддерживается в API `gTTS` (например, казахский), то в представлении признак `sound` получит значение `False` и в шаблоне `pdf_long_gtts.html` не будет активирован тег с проигрывателем на данном языке (рис. 7.15).

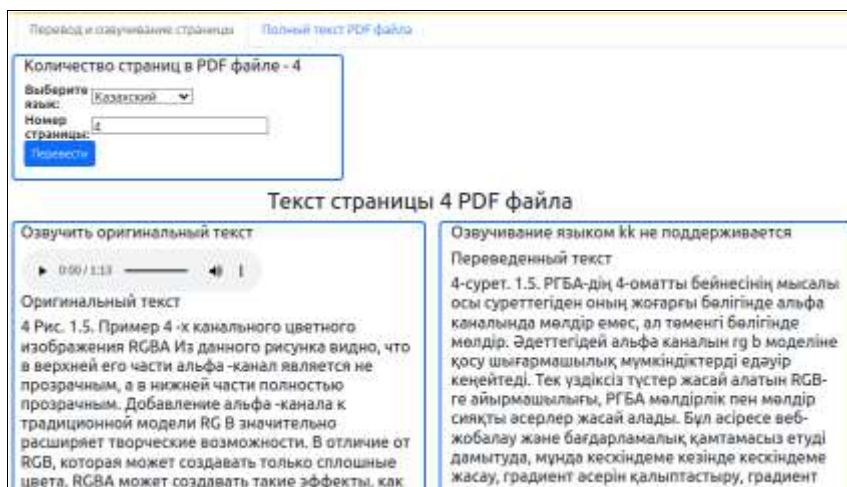


Рис.7.15. Шаблон pdf_long.html без проигрывателя для озвучивания текста (вкладка «Перевод и озвучивание страницы», язык - казахский)

Как видно из данного рисунка, программой выполнен только перевод выбранной страницы на казахский язык и аудио проигрыватель не показан в блоке с текстом на казахском языке.

7.9. Перевод и озвучивание текста PDF файлов с переносом изображений с библиотекой gtts

В предыдущем разделе после перевода текста на другой язык пользователю был показан только текст переведенной страницы без рисунков, что не совсем удобно, так как для просмотра рисунков нужно переключаться на вкладку с исходным PDF файлом. А теперь реализуем возможность постраничного перевода исходного PDF файла с формированием выходной страницы в формате PDF с переносом не только текста, но и рисунков. В этом случае пользователь может просматривать и озвучивать текст любой страницы документа (оригинальный и переведенный) без лишних переключений между вкладками.

Для обработки исходного PDF файла нужно будет выполнить следующие шаги. С диска загрузить содержимое исходного PDF файла, но не показывать его в шаблоне и не выполнять обработку страниц данного файла. В шаблоне необходимо отобразить только форму с полем для выбора страницы PDF документа и выпадающим списком для выбора языка, на который нужно перевести выбранную страницу. К форме необходимо добавить три кнопки:

- ☐ Озвучить оригинал;
- ☐ Выполнить перевод;
- ☐ Перевести и озвучить.

Нажатие этих кнопок нужно будет обработать в представлении по POST-запросу.

1. При нажатии на кнопку «Озвучить оригинал» необходимо показать оригинал страницы в формате PDF и на основе текста страницы сформировать звуковую дорожку на языке оригинала.

2. При нажатии на кнопку «Выполнить перевод» необходимо показать переведенный текст страницы без формирования звуковой дорожки.
3. При нажатии на кнопку «Перевести и озвучить» необходимо показать переведенный текст страницы и сформировать звуковую дорожку на языке, указанном для перевода.

В программных модулях данного раздела будет использована форма LangForm, которая описана в предыдущем разделе.

Для активации модулей преобразования текста из многостраничного файла PDF в речь с переводом на другой язык и с переносом изображений нужно создать следующую ссылку (листинг 7.34).

Листинг 7.34. Ссылка для активации модулей перевода и озвучивания текста из многостраничного PDF файла с переносом изображений

Примечание.

В демо-версии книги удалены листинги программ.

По этой ссылке будет активирован маршрут URL с именем `page_select` (листинг 7.35).

Листинг 7.35. Маршрут `page_select`

Примечание.

В демо-версии книги удалены листинги программ.

По этому маршруту будет вызвано представление `views.page_select` (листинг 7.36).

Листинг 7.36. Представление `page_select`

Примечание.

В демо-версии книги удалены листинги программ.

В этом представлении выполнен импорт необходимых библиотек, создан объект `ling_gtts`, который содержит информацию о тех языках, на которых может быть озвучен текст с использованием библиотеки `gtts`. Это словарь необходим для того, чтобы в ходе выполнения программы можно было убедиться в том, что переведенный текст будет озвучен (библиотека перевода текста поддерживает более широкий список языков, чем библиотека озвучивания текста). Здесь также создаются объекты, в которых заданы пути к папкам и файлам приложения.

Затем выполняется открытие и чтение содержимого файла PDF (в объект `pdf_reader`) и определение количества страниц текста (объект `num_pages`). Пропускаем строки программы, которые находятся в теле инструкции `if`, так как они выполняются только после того, как пользователь выгрузил данные из формы.

После этого создан объект форма (`form`) для изменения параметров обработки текста. Далее создается контекстный словарь (`context`), включающий все необходимые сформированные данные, который передается в шаблон `page_select.html` (листинг 7.37).

Листинг 7.37. Шаблон page_select.html

Примечание.

В демо-версии книги удалены листинги программ.

В данном шаблоне отображается номер текущей страницы (по умолчанию - 1), язык перевода (по умолчанию - английский) и имеется три кнопки:

- ☐ Озвучить оригинал;
- ☐ Выполнить перевод;
- ☐ Перевести и озвучить.

После нажатия на данные кнопки делается возврат к представлению **page_select** в виде POST-запроса с передачей имени (name) нажатой кнопки. В блоке обработки POST запроса для всех кнопок создаются следующие объекты:

- ☐ `show_pdf = False` – показать содержимое страницы с оригинальным текстом из файла PDF;
- ☐ `show_trs = False` - показать содержимое переведенной страницы;
- ☐ `show_img = False` - показать рисунки на переведенной странице;
- ☐ `list_img = []` – создать пустой список изображений на переведенной странице.

После этого создается объект форма (form), в который загружаются данные, выбранные пользователем. Если содержимое формы валидно, то объект `my_dest` получает выбранный пользователем язык перевода, а объект `tek_page` - номер текущей страницы. Так как в файле PDF нумерация страниц начинается не с единицы, а с нуля, то здесь текущий номер страницы уменьшается на единицу.

Поскольку пользователь при выборе текущей страницы может выйти за рамки допустимых страниц PDF файла, то далее присутствует блок, который предотвращает подобную ошибку. Если пользователь указал отрицательную страницу, то она приводится к нулевой странице PDF файла. Если пользователь указал номер страницы больший, чем имеется в файле, то она приводится к последней странице файла.

На следующем этапе проверяется наличие не актуальных аудио файлов, файлов изображений и их удаление. Здесь с использованием инструкции `for` организован цикл, в теле которого проверяется наличие в папке `path_audio` аудио файлов с маской `"page_*.mp3"`. Если такие файлы найдены, то они будут удалены с использованием инструкции `os.remove(f)`.

После этого проверяется наличие не актуальных файлов изображений и их удаление. Здесь с использованием инструкции `for` организован цикл, в теле которого проверяется наличие в папке `path_images` файлов изображений с маской `"image_*.jpg"`. Если такие файлы найдены, то они будут удалены с использованием инструкции `os.remove(f)`.

Далее идут фрагменты программного кода, которые обрабатывают одну из нажатых кнопок.

Фрагмент кода для озвучивания оригинала текста выбранной страницы

Для озвучивания текста на языке оригинала нужно выполнить следующие действия:

- ☐ получить текущую страницу, выбранную пользователем;
- ☐ извлечь текст текущей страницы;
- ☐ сформировать временный PDF файл с текущей страницей;

- ☐ получить URL адрес PDF файл с текущей страницей;
- ☐ создать аудио дорожку из текста страницы;
- ☐ записать аудио дорожку страницы на диск;
- ☐ получить URL адрес аудио дорожки.

Нажатие кнопки «Озвучить оригинал» в шаблоне обрабатывается в следующем фрагменте кода:

```
<!-- Кнопка с крутящимся спином Озвучить-->
<div class="col-3 mx-1 my-1">
  <button type="submit" class="btn btn-success btn-sm toggle-text" name="sound">
    <span><i class="bi bi-volume-up"></i> Озвучить оригинал</span>
    <span class="hidden spinner-border spinner-border-sm" role="status" aria-
hidden="true"></span>
  </button>
</div>
```

Фрагмент программного кода, который обеспечивает озвучивание оригинального текста текущей страницы, находится в теле инструкции:

```
if 'sound' in request.POST:
```

Здесь объект `page` получает текущую страницу файла, а извлеченный из страницы текст сохраняется в объекте `txt_ru`. Если на странице нет текста (только рисунки), то объект `txt_ru` получает значение «На данной странице нет текста для озвучивания». Затем объект `writer` получает копию текущей страницы, сохраняет ее на диске под именем `temp.pdf`. На следующем шаге PDF файл считывается в объект `pdf` (в двоичном формате), и двоичный формат преобразуется в строковый (объект `pdf_content`).

После этого с использованием API `gTTS` из текста объекта `txt_ru` создается аудио дорожка (`sound`), которая записывается на диск в виде аудио файла под именем:

```
file_name_mp3 = 'page_' + str(tek_page + 1) + '.mp3'
```

Для того чтобы в шаблоне подключить сформированный аудио файл к проигрывателю, формируем его URL адрес (объект `url_mp3`). Для того что бы в шаблоне пользователю показать истинный номер страницы создается объект `show_page`.

После этого все сформированные объекты загружаются в контекстный словарь `context`, который передается в шаблон `api_vision/page_select.html`.

Примечание.

В интерактивной цифровой книге для запуска данного примера нужно на вкладке «Показ примеров» нажать на кнопку «Примеры главы 7» и активировать ссылку «Обработка PDF с рисунками с `gTTS`».

Пример озвучивания оригинального текста страницы 4 PDF файла приведен на рис. 7.16.

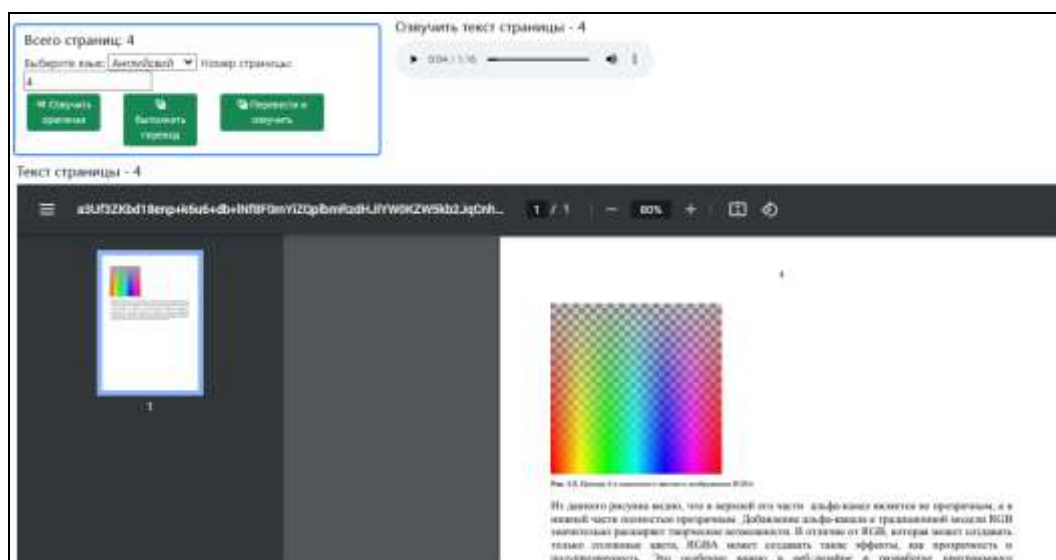


Рис.7.16. Пример озвучивания оригинального текста страницы 4 PDF файла

Как видно из данного рисунка, в верхней части страницы появился аудио проигрыватель, а ниже показан полный текст текущей страницы в виде PDF файла. При нажатии на кнопку проигрывателя «Старт» будет озвучен текст текущей страницы PDF файла.

Фрагмент кода для перевода текста выбранной страницы

Для перевода текста выбранной страницы нужно выполнить следующие действия:

- ☐ получить текущую страницу;
- ☐ извлечь текст текущей страницы;
- ☐ выполнить перевод текста всей страницы (или по частям, если он больше 40 строк) на выбранный язык;
- ☐ извлечь рисунки из текущей страницы (если они есть).

Нажатие кнопки «Выполнить перевод» в шаблоне обрабатывается в следующем фрагменте кода:

```
<!-- Кнопка с крутящимся спином Перевести-->
<div class="col-3 mx-1 my-1">
  <button type="submit" class="btn btn-success btn-sm toggle-text" name="trans">
    <span><i class="bi bi-translate"></i> Выполнить перевод</span>
    <span class="hidden spinner-border spinner-border-sm" role="status" aria-
hidden="true"></span>
  </button>
</div>
```

Фрагмент программного кода, который обеспечивает перевод текста текущей страницы, находится в теле инструкции:

```
if 'trans' in request.POST:
```

Здесь объект page получает текущую страницу файла, а извлеченный из страницы текст сохраняется в объекте txt. Затем проверяется условие, есть ли на текущей странице текст.

Если текста нет (только рисунки), то объект `txt` получает значение «На данной странице нет текста для перевода», а объект `kol_str` (количество строк на текущей странице) получает значение `kol_str=1`. Если на текущей странице есть текст, то объект `kol_str` получает количество строк в текущей странице.

Если количество строк меньше или равно 40, то запускается процесс перевода всей страницы. Для перевода текста в асинхронном режиме вызывается функция `translate_text`, в которую передаются следующие аргументы:

- ❑ `txt` – текст страницы на исходном языке (русский);
- ❑ `'ru'` – ключ словаря языка оригинального текста (русский);
- ❑ `my_dest` – ключ словаря языка перевода текста (выбирается пользователем).

Переведенный текст сохраняется в объекте `out_txt`.

Если количество строк на странице превышает 40, то страница разбивается на две части, и перевод выполняется поочередно (для первой и второй части страницы). Нейронная сеть, которая обеспечивает перевод текста, не принимает к переводу длинные строковые переменные, поэтому здесь выполняется такая разбивка. В итоге получаем два итоговых объекта с переведенными частями страницы (`out_txt1` и `out_txt2`), которые затем объединяются в единый объект `out_txt`.

На выбранной пользователем странице могут содержаться рисунки, поэтому далее проверяется наличие на странице рисунков с использованием инструкции:

```
if '/XObject' in page['/Resources']:
```

На странице может быть несколько рисунков, поэтому с использованием инструкции `for` организован цикл поиска всех рисунков:

```
for i, image in enumerate(page.images):
```

Для каждого найденного рисунка формируется имя по следующему шаблону:

```
image_{номер_страницы}_{номер_рисунка_на_странице}.jpg
```

Для того что бы в шаблоне пользователю показать истинный номер страницы создается объект `show_page`.

После этого все сформированные объекты загружаются в контекстный словарь `context`, который передается в шаблон `api_vision/page_select.html`.

Примечание.

В интерактивной цифровой книге для запуска данного примера нужно на вкладке «Показ примеров» нажать на кнопку «Примеры главы 7» и активировать ссылку «Обработка PDF с рисунками с gTTS».

В меню на сервере – Обработка аудио/ Обработка PDF с рисунками с gTTS

Страница на сервере - http://kalinkashop.tw1.ru/api_vision/page_select

Пример перевода оригинального текста страницы 4 на китайский язык приведен на рис. 7.17.

Всего страниц: 4

Выберите язык: Китайский ▼ Номер страницы: 4

Озвучить оригинал
Выполнить перевод
Перевести и озвучить

Переведенный текст страницы - 4

4

米。1.5。示例4-彩色图像rgba
 从这个图来看，很明显，在其上部，alpha通道不是透明的，而是下部是完全透明的。在传统的RGB模型中添加Alpha频道大大扩大了创造机会。与RGB不同，它可以创建只有连续的颜色，RGBA才能产生透明度和半透明。这在软件的网页设计和开发中尤其重要冬季至关重要的结果，应用图像，创建梯度效应并与视觉上有吸引力的接口SEMOPR带有闪烁的元素。Alpha-Channel E FL感染允许图像合并背景或其他和Zyttes，是平滑整合的结果。

Изображения текущей страницы - 4

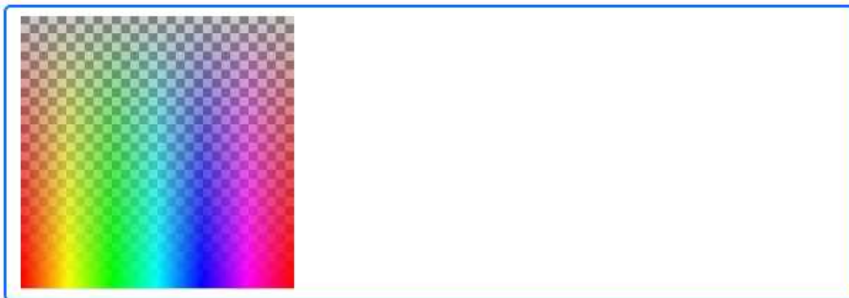


Рис.7.17. Пример фрагмента перевода текста страницы 4 на китайский язык

Как видно из данного рисунка, в шаблоне показан переведенный текст текущей страницы в виде текстовых строк, после которых следуют все рисунки текущей страницы.

Фрагмент кода для перевода и озвучивания текста выбранной страницы

Для перевода и озвучивания переведенного текста нужно выполнить следующие действия:

- ☐ получить текущую страницу;
- ☐ извлечь текст текущей страницы;
- ☐ выполнить перевод текста все страницы (или по частям, если он больше 40 строк) на выбранный язык;
- ☐ создать аудио дорожку из переведенного текста;
- ☐ сохранить аудио дорожку на диске в виде файла формата mp3;
- ☐ получить URL аудио дорожки страницы;
- ☐ извлечь рисунки из текущей страницы (если они есть).

Нажатие кнопки «Перевести и озвучить» в шаблоне обрабатывается в следующем фрагменте кода:

```
<!-- Кнопка с крутящимся спином перевести и озвучить-->
<div class="col-4 mx-1 my-1">
  <button type="submit" class="btn btn-success btn-sm toggle-text" name="trans_sound">
    <span><i class="bi bi-translate"></i> Перевести и озвучить</span>
    <span class="hidden spinner-border spinner-border-sm" role="status" aria-
hidden="true"></span>
  </button>
</div>
```

Фрагмент программного кода, который обеспечивает перевод и озвучивание текста текущей страницы, находится в теле инструкции:

```
if 'trans_sound' in request.POST:
```

Здесь объект `page` получает текущую страницу, а извлеченный из страницы текст сохраняется в объекте `txt`.

Затем проверяется условие, есть ли на текущей странице текст. Если текста нет (только рисунки), то объект `txt` получает значение «На данной странице нет текста для озвучивания», а объект `kol_str` (количество строк на текущей странице) получает значение `kol_str=1`. Если на текущей странице есть текст, то объект `kol_str` получает количество строк в текущей странице.

После этого запускается процесс перевода текста страницы на выбранный язык. Если количество строк меньше или равно 40, то запускается процесс перевода всей страницы. Для перевода текста в асинхронном режиме вызывается функция `translate_text`, в которую передаются следующие аргументы:

- ☐ `txt` – текст страницы на исходном языке (русский);
- ☐ `'ru'` – ключ словаря языка оригинального текста (русский);
- ☐ `my_dest` - ключ словаря языка перевода текста (выбирается пользователем).

Переведенный текст сохраняется в объекте `out_txt`.

Если количество строк на странице превышает 40, то страница разбивается на две части, и перевод выполняется поочередно (для первой и второй части страницы). Нейронная сеть, которая обеспечивает перевод текста, не принимает к переводу длинные строковые переменные, поэтому здесь выполняется такая разбивка. В итоге получает два итоговых объекта с переведенными частями страницы (`out_txt1` и `out_txt2`), которые затем объединяются в единый объект `out_txt`.

После этого делается проверка, можно ли для выбранного языка перевода создать аудио дорожку:

```
if my_dest in ling_gtts:
```

Если аудио дорожка может быть создана, то формируется имя аудио файла:

```
file_name_mp3 = 'page_' + str(tek_page + 1) + '.mp3'
```

После этого делается проверка – есть на диске файл с таким именем. Если он есть, то будет удален.

Если API gTTS позволяет создать аудио дорожку для выбранного языка, то из текста объекта `out_txt` создается аудио дорожка (`sound`), которая записывается на диск. Для того чтобы в шаблоне подключить сформированный аудио файл к проигрывателю, формируем его URL адрес (объект `url_mp3`).

Если API gTTS не позволяет создать аудио дорожку для выбранного языка, то формирование аудио дорожки не выполняется и объект `play` получает значение `False` (не показывать проигрыватель в шаблоне).

На выбранной пользователем странице могут содержаться рисунки, поэтому далее проверяется наличие на странице рисунков с использованием инструкции:

```
if '/XObject' in page['/Resources']:
```

На странице может быть несколько рисунков, поэтому с использованием инструкции `for` организован цикл поиска всех рисунков:

```
for i, image in enumerate(page.images):
```

Для каждого найденного рисунка формируется имя по следующему шаблону:

```
image_{номер_страницы}_{номер_рисунка_на_странице}.jpg
```

После этого все сформированные объекты загружаем в контекстный словарь `context`, который возвращаем в шаблон `api_vision/page_select.html`.

Примечание.

В интерактивной цифровой книге для запуска данного примера нужно на вкладке «Показ примеров» нажать на кнопку «Примеры главы 7» и активировать ссылку «Обработка PDF с рисунками с gTTS».

Пример перевода и проигрывания переведенного текста страницы 4 приведен на рис. 7.18.



Рис.7.18. Пример перевода и озвучивания фрагмента текста страницы на китайском языке

Как видно из данного рисунка, в шаблоне показан переведенный текст текущей страницы в виде текстовых строк, а рядом с формой отображен проигрыватель аудио файлов. Пользователь имеет возможность, как просматривать перевод, так и прослушивать аудио версию текста текущей страницы.

Примечание.

Изображения в PDF файлах рисуются в координатах левого нижнего угла страницы. Для того чтобы определить место вставки изображения сразу за текстом, нужно знать координату строки, на которой закончился текст, а затем от этой координаты отнять высоту изображения.

7.10. Краткие итоги

Итак, мы познакомились с некоторыми базовыми элементами фреймворка Bootstrap, которые нам пригодятся при формировании шаблонов HTML-страниц. Но прежде чем приступить к их использованию в веб-приложениях, нужно познакомиться с некоторыми важными параметрами элементов Bootstrap, и способами их подключения к шаблонам Django. В частности нам предстоит узнать:

- ❑ как можно менять цветовое оформление элементов Bootstrap;

- ❑ как можно встраивать одни шаблоны Django в другие;
- ❑ механизм подключения иконок к элементам Bootstrap ;
- ❑ как можно сделать разбивку HTML страницы на горизонтальные и вертикальные вкладки и организовать навигацию по ним.
- ❑ какие есть варианты использования атрибута href в шаблонах Django для создания ссылок на URL адреса.
- ❑ как создать и использовать всплывающие модальные окна;
- ❑ как отобразить в шаблонах Django статические компоненты Bootstrap (на примере аккордеона, карточек и карусели).

Рассмотрению этих вопросов и посвящена следующая глава.

ГЛАВА 8



Транскрибация медиа контента в приложениях на Python и Django

Транскрибация медиа контента — это преобразование речи, записанной в виде аудио файла или видео файла, в письменную форму. Во время транскрибации прослушивают исходный материал, расшифровывают слова, ставят правильную пунктуацию и форматируют итоговый текст. Основная цель транскрибации — сохранить медиа контент в письменной форме, то есть запротоколировать устную речь. Это необходимо журналистами для того, чтобы оперативно напечатать интервью, расшифровав аудио запись. Кроме того, транскрибация позволяет сделать содержание медиа файлов доступным для людей с нарушениями слуха.

Транскрибацию используют в разных сферах:

- ☐ в судебной практике (формирование протоколов судебных заседаний);
- ☐ в журналистике (подготовка публикаций на основе расшифровки интервью и материалов пресс-конференций);
- ☐ в образовании (создание конспектов на основе записанных аудио лекции);
- ☐ в бизнес-процессах и маркетинге (конвертация видео и аудио материалов в отчеты).

В примерах этой главы будет показана возможность транскрибации медиа контента в программах на Python и Django. Из материалов данной главы вы узнаете:

- ☐ какие библиотеки Python можно использовать для транскрибации медиа контента;
- ☐ как можно на Python реализовать универсальный базовый модуль для транскрибации аудио контента;
- ☐ как можно подключить базовый модуль для транскрибации аудио контента к веб приложению и вывести результаты его работы в шаблон Django;
- ☐ как можно на Python реализовать универсальный базовый модуль для транскрибации видео контента;
- ☐ как можно подключить базовый модуль для транскрибации видео контента к веб приложению и вывести результаты его работы в шаблон Django;

8.1. Библиотеки Python для транскрибации медиа контента

В первую очередь для транскрибации медиа контента нужна библиотека для распознавания речи. В основе систем распознавания речи лежит Марковская модель, суть которой заключается в том, что при анализе сигнала в промежутке небольшой длительности (от пяти до 10

миллисекунд), возможна его аппроксимация, как при стационарном процессе. Основой систем распознавания речи являются обученные нейронные сети.

В данной главе мы будем использовать opensource решения, которые, согласно условиям лицензии, можно использовать без оплаты. Существуют следующие библиотеки для распознавания речи с открытым исходным кодом: DeepSpeech, Kaldi, Vosk, SpeechRecognition и др.

Здесь мы будем использовать библиотеку SpeechRecognition. Отличие этой библиотеки от других заключается в том, что для распознавания речи в ней задействовано несколько разных движков и сервисов. Для транскрипции данная система применяет Google Web Speech API, Sphinx, Bing Speech API, Google Cloud Speech API и некоторые другие. Саму библиотеку легко использовать в скриптах на Python, что делает решение подходящим для прикладных задач. У неё есть удобный интерфейс, с помощью которого можно интегрировать систему распознавания речи в собственное приложение. Данная библиотека имеет следующие положительные качества.

1. Библиотеку можно запускать на устройствах под управлением Windows, macOS, и Linux. Разработчик вызывает функции библиотеки в скрипте Python, который автоматически интерпретируется для конкретной операционной системы. Это позволяет создавать кроссплатформенные проекты.
2. Транскрипция работает как с облачными сервисами, так и с офлайн-движками. Так как библиотека поддерживает несколько платформ, разработчик может создать проект, который не нуждается в подключении к интернету для перевода аудиофайла в текст.
3. Библиотека распознает речь на разных языках в зависимости от используемого программного интерфейса или движка.

Однако у бесплатной версии данной библиотеки есть ограничение, она уверенно работает только с короткими фрагментами медиа данных. Но эту проблему можно решить, если дополнительно использовать другую библиотеку - PyDub.

Решение задачи транскрипции большого аудио файла можно выполнить в три этапа.

1. Большой аудио файл «нарезать» на более мелкие фрагменты.
2. Выполнить транскрипцию каждого фрагмента с получением группы мелких текстовых данных.
3. Группу мелких текстовых данных объединить в один файл.

С использованием библиотеки PyDub можно успешно реализовать первый этап данного процесса – разбивку длинного аудио файла на фрагменты. PyDub — это библиотека Python, которая упрощает процесс работы с аудио файлами. Она предоставляет простой и интуитивно понятный интерфейс для различных задач по редактированию аудио. Это мощный инструмент для работы с аудио файлами, будь то извлечение аудио из видео файлов, преобразование форматов, применение эффектов или выполнение других задач. Основным классом в библиотеке Pydub является класс AudioSegment, который представляет собой контейнер для загрузки, обработки и сохранения аудио.

Для транскрипции видео контента нам понадобится еще одна библиотека – MoviePy. Используя MoviePy, можно достаточно просто извлекать аудио дорожку из видео файла. Эта возможность особенно полезна в сценариях, где нужно выполнить обработку аудио файла не затрагивая основного видео контента.

Для использования функций данных библиотек их необходимо установить в вашу рабочую среду с использованием следующих инструкций:

```
pip install SpeechRecognition
pip install pydub
pip install movepy
```

8.2. Модуль Python для транскрибации аудио контента

Для начала рассмотрим программный модуль Python, который позволяет выполнить транскрибацию речи в аудио файлах (листинг 8.1).

Листинг 8.1. Модуль `txt_long_audio.py`

Примечание.

В демо-версии книги удалены листинги программ.

Для работы данного модуля выполняется импорт следующих пакетов:

- ❑ `import speech_recognition as sr` – для распознавание речи в аудио контенте;
- ❑ `import os` – для формирования путей к файлам приложения;
- ❑ `from pydub import AudioSegment` – для разбивки аудио файла на фрагменты;
- ❑ `from pydub.silence import split_on_silence` - для разбивки аудио файла на фрагменты;
- ❑ `import glob` – для удаления временных файлов.

После этого находится программный код функции «`def transcribe_large_audio`», которая принимает два параметра: `path` – путь к исходному аудио файлу, `out_txt` – путь к файлу с распознанным в аудио файле текстом. Затем на основе класса `Recognizer()` библиотеки `speech_recognition` создан объект «`r`» для распознавания речи, а в объект `sound` загружен исходный аудио файл. После этого выполняется разбивка загруженного аудио файла на фрагменты с записью их в итерируемый объект `chunks`. Разбивка длинного аудио файла на мелкие фрагменты выполняется автоматически с использованием признака того, что пауза между словами не будет превышать 700 мили секунд.

На следующем шаге создается папка с именем «`audio-chunks`» для хранения временных фрагментов аудио файлов. После того создается пустая текстовая переменная `whole_text`, в которую будет записываться распознанный текст, и организован цикл с использованием следующей инструкции:

```
for i, audio_chunk in enumerate(chunks, start=1):
```

В этом цикле делается перебор итерируемого объекта `chunks` (фрагменты длинного аудио файла), для каждого фрагмента временного аудио файла формируется уникальное имя (`chunk_filename`), и эти фрагменты записываются на диск в папку «`audio-chunks`». После этого каждый фрагмент аудио файла считывается в объект `audio_listened`, который передается объекту «`r`» для распознавания текста, а распознанный текст сохраняется в объекте `text`:

```
text = r.recognize_google(audio_listened, language="ru-RU")
```

Поскольку большинство фраз аудио фрагмент будут являться предложениями, то первый символ каждого предложения делается заглавным, а после точки в конце предложения добавляется пробел:

```
text = f"{text.capitalize()}". "
```

После этого все фразы распознанного текста собираются в итоговую текстовую переменную:

```
whole_text += text
```

По завершению цикла `for` объект `whole_text` сохраняется на диске в формате текстового документа, а все временные фрагменты аудио файлов удаляются с диска.

Так как существует вероятность того, что не все фрагменты аудио файла будут распознаны, то строки кода распознавания текста включены в инструкцию обработки исключений (`try - except`) . В этом случае на место нераспознанной фразы будет вставлен текст следующего содержания:

```
'ОШИБКА В РАСПОЗНАВАНИИ. '
```

По завершению процесса распознавания текста пользователь может загрузить итоговый файл в текстовый редактор и выполнить его редактирование (расставить знаки препинания и исправить возможные ошибки).

Примечание.

Данный модуль не включен в состав исполняемых примеров интерактивной цифровой книги, так как он предназначен для работы на локальном компьютере. Но вы имеете возможность скопировать листинг данной программы, запустить данный программный модуль на выполнение на своем локальном компьютере и использовать в своих приложениях.

Если данный программный код запустить на выполнение на локальном компьютере и загрузить итоговый файл в текстовый редактор, то будет получен следующий результат (рис. 8.1).

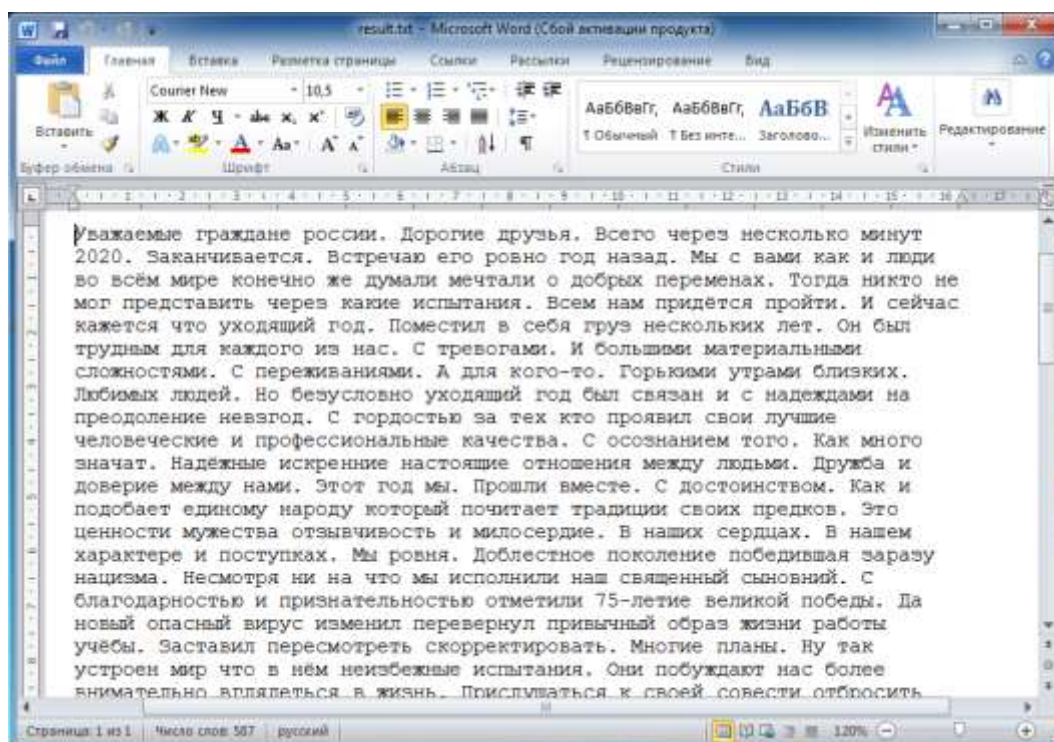


Рис. 8.1. Результаты распознавания текста в аудио файле

8.3. Транскрибация аудио контента в приложениях на Django

Для активации модулей транскрибации аудио файлов в приложениях на Django нужно создать следующую ссылку (листинг 8.2):

Листинг 8.2. Ссылка для активации модулей транскрибации аудио файлов

Примечание.

В демо-версии книги удалены листинги программ.

По этой ссылке будет активирован следующий маршрут URL (листинг 8.3):

Листинг 8.3. Маршрут trans_audio

Примечание.

В демо-версии книги удалены листинги программ.

По этому маршруту будет вызвано представление `views.trans_audio` (листинг 8.4):

Листинг 8.4. Представление `views.trans_audio`

Примечание.

В демо-версии книги удалены листинги программ.

В данном представлении сначала выполнен импорт модуля `txt_long_audio`, это модуль Python с функцией `transcribe_large_audio`, которая была описана в предыдущем разделе. Затем были сформированы пути к папкам и файлам проекта:

- ❑ `my_app = base_dir` - путь к приложению;
- ❑ `path_media = os.path.join(my_app, 'media')` - путь к папке `media`;
- ❑ `path_inp_wav = os.path.join(path_media, 'audio/putin.wav')` - путь к исходному аудио файлу;
- ❑ `path_out_txt = os.path.join(path_media, 'txt_pdf/result.txt')` - путь к полученному текстовому файлу.

После этого сделано обращение к функции `transcribe_large_audio` модуля `txt_long_audio`, в которую передан путь к аудио файлу (`path_media`) путь к итоговому текстовому файлу (`result.txt`). В теле данной функции выполняется транскрибация аудио файла с записью итогового текстового файла на диск.

На следующем шаге содержимое полученного текстового файла загружается в объект `content` и формируется URL адрес исходного аудио файла (`inp_url`).

Затем URL адрес исходного аудио файла и содержимое текстового файла через контекстную переменную `context` передаются в шаблон `trans_audio.html`, который имеет следующий код (листинг 8.5).

Листинг 8.5. Шаблон `trans_audio.html`

Примечание.

В демо-версии книги удалены листинги программ.

В этом шаблоне присутствует тег с аудио проигрывателем и тег для вывода текста, распознанного в аудио файле.

Примечание.

В интерактивной цифровой книге для запуска данного примера нужно на вкладке «Показ примеров» нажать на кнопку «Примеры главы 8» и активировать ссылку «Транскрибация аудио файлов».

Если выполнить данный программный модуль, то в окне браузера отобразится следующий результат (рис. 8.2).

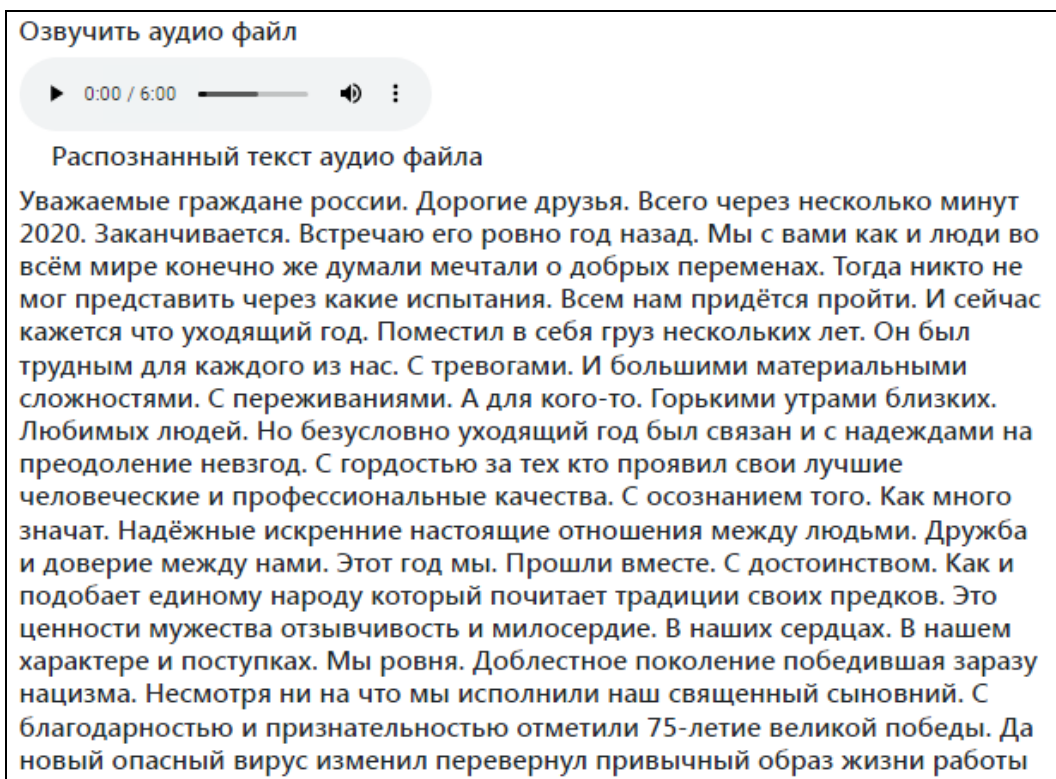


Рис.8.2. Шаблон `trans_audio.html` с аудио проигрывателем и фрагментом текста, извлеченного из аудио записи

Если нажать на кнопку старта проигрывания аудио дорожки, то мы услышим исходную аудио запись.

8.4. Модуль Python для транскрибации видео контента

Теперь рассмотрим программный модуль Python, который позволяет выполнить транскрибацию речи в видео файлах (листинг 8.6).

Листинг 8.6. Модуль `txt_long_video.py`

Примечание.

В демо-версии книги удалены листинги программ.

Для работы данного модуля выполняется импорт следующих пакетов:

- ❑ `import speech_recognition as sr` – для распознавание речи в аудио контенте;
- ❑ `import os` – для формирование путей к файлам приложения;
- ❑ `from pydub import AudioSegment` – для разбивки аудио файла на фрагменты;

- ❑ `from pydub.silence import split_on_silence` - для разбивки аудио файла на фрагменты;
- ❑ `import glob` – для удаления временных файлов;
- ❑ `from moviepy.editor import *` - для загрузки аудио дорожки из видео файла.

После этого находится программный код функции «def transcribe_video», которая принимает три параметра: `path` – путь к исходному видео файлу, `path_a` – путь к извлеченной из видео файла аудио дорожке, `out_txt` – путь к файлу с распознанным в аудио файле текстом. В этой функции на первом шаге делается извлечение аудио дорожки из видео файла и ее запись на диск, для чего используются модули библиотеки `moviepy`.

Затем на основе класса `Recognizer()` библиотеки `speech_recognition` создан объект «r» для распознавания речи, а в объект `sound` загружен исходный аудио файл. После этого выполняется разбивка загруженного аудио файла на фрагменты с записью их в итерируемый объект `chunks`. Разбивка длинного аудио файла на мелкие фрагменты выполняется автоматически с использованием признака того, что пауза между словами не будет превышать 700 миллисекунд.

На следующем шаге создается папка с именем «audio-chunks» для хранения временных фрагментов аудио файлов. После того создается пустая текстовая переменная `whole_text`, в которую будет записываться распознанный текст, и организован цикл с использованием следующей инструкции:

```
for i, audio_chunk in enumerate(chunks, start=1):
```

В этом цикле делается перебор итерируемого объекта `chunks` (фрагменты длинного аудио файла), для каждого фрагмента аудио файла формируется уникальное имя (`chunk_filename`), и эти фрагменты записываются на диск в папку «audio-chunks». После этого каждый фрагмент аудио файла считывается в объект `audio_listened`, который передается объекту «r» для распознавания текста, а распознанный текст сохраняется в объекте `text`:

```
text = r.recognize_google(audio_listened, language="ru-RU")
```

Поскольку большинство фраз аудио фрагмент будут являться предложениями, то первый символ каждого предложения делается заглавным, а после точки в конце предложения добавляется пробел:

```
text = f"{text.capitalize()}. "
```

После этого все распознанные фразы распознанного текста собираются в итоговую текстовую переменную:

```
whole_text += text
```

По завершению цикла `for` объект `whole_text` сохраняется на диске в формате текстового документа, все временные фрагменты аудио файлов удаляются с диска, а также удаляется извлеченная из видео файла аудио дорожка

Так как существует вероятность того, что не все фрагменты аудио файла будут распознаны, то строки кода распознавания текста включены в инструкцию обработки исключений (`try - except`). В этом случае на место нераспознанной фразы будет вставлен текст следующего содержания:

```
'ОШИБКА В РАСПОЗНАВАНИИ. '
```

По завершению процесса распознавания текста пользователь может загрузить итоговый файл в текстовый редактор и выполнить его редактирование (расставить знаки препинания и исправить возможные ошибки).

Примечание.

Данный модуль не включен в состав исполняемых примеров интерактивной цифровой книги, так как он предназначен для работы на локальном компьютере. Но вы имеете возможность скопировать листинг данной программы, запустить данный программный модуль на выполнение на своем локальном компьютере и использовать в своих приложениях.

Если данный программный код запустить на выполнение на локальном компьютере и загрузить итоговый файл в текстовый редактор, то будет получен следующий результат (рис. 8.3).

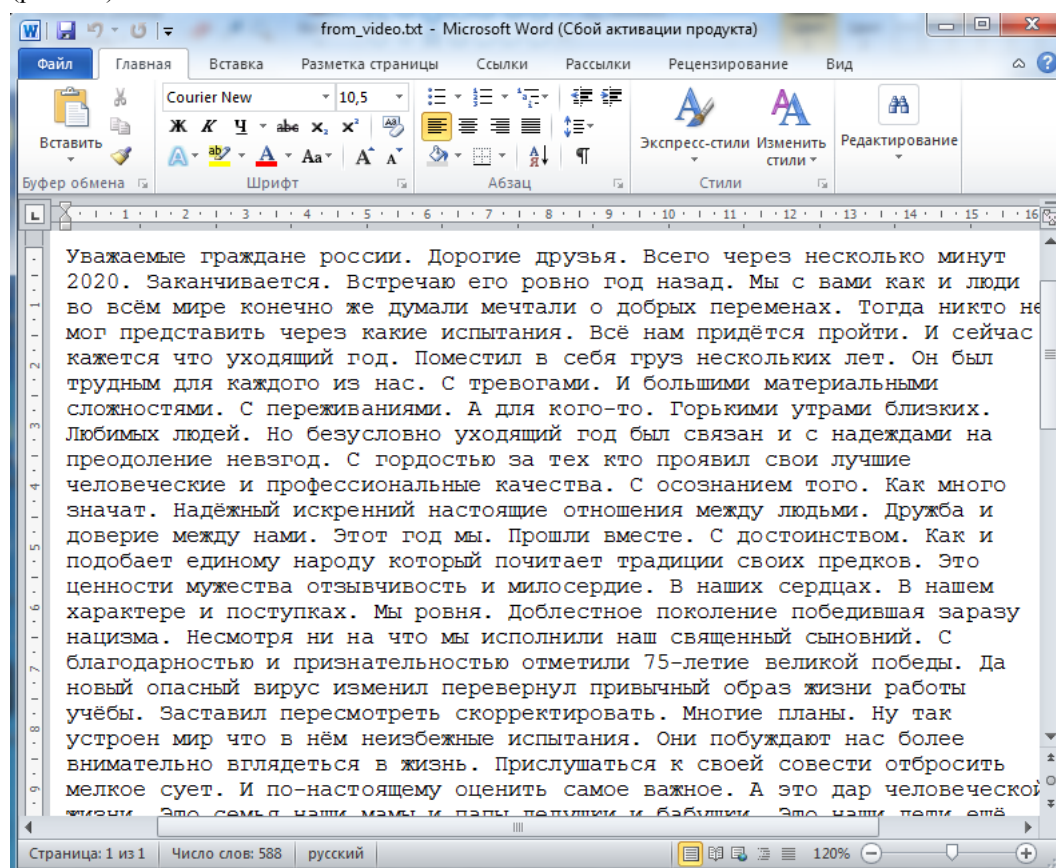


Рис. 8.3. Результаты распознавания текста в видео файле

8.5. Транскрибация видео контента в приложениях на Django

Для активации модулей транскрибации видео файлов в приложениях на Django нужно создать следующую ссылку (листинг 8.7):

Листинг 8.7. Ссылка для активации модулей транскрибации видео файлов

Примечание.

В демо-версии книги удалены листинги программ.

По этой ссылке будет активирован следующий маршрут URL (листинг 8.8):

Листинг 8.8. Маршрут trans_video

Примечание.

В демо-версии книги удалены листинги программ.

По этому маршруту будет вызвано представление views.trans_audio (листинг 8.9):

Листинг 8.9. Представление views.trans_video

Примечание.

В демо-версии книги удалены листинги программ.

В данном представлении сначала выполнен импорт модуля `txt_long_video`, это модуль Python с функцией `transcribe_video`, которая была описана в предыдущем разделе.

Затем были сформированы пути к папкам и файлам проекта:

- ☐ `my_app = base_dir` - путь к приложению;
- ☐ `path_media = os.path.join(my_app, 'media')` - путь к папке media;
- ☐ `path_video = os.path.join(path_media, 'video/Put.mp4')` - путь к исходному видео;
- ☐ `path_audio = os.path.join(path_media, 'audio/put.wav')` - путь к создаваемому аудио файлу;
- ☐ `path_out_txt = os.path.join(path_media, 'txt_pdf/from_video.txt')` - путь к полученному текстовому файлу.

После этого сделано обращение к функции `transcribe_video` модуля `txt_long_video`, в которую переданы следующие аргументы: путь к исходному видео файлу (`path_video`), путь к извлеченной аудио дорожке (`path_audio`), путь к итоговому текстовому файлу (`path_out_txt`). В теле данной функции выполняется транскрибация аудио дорожки с записью итогового текстового файла на диск.

На следующем шаге содержимое полученного текстового файла загружается в объект `content` и формируется URL адрес исходного видео файла (`video_url`).

Затем URL адрес исходного видео файла и содержимое текстового файла через контекстную переменную `context` передаются в шаблон `trans_video.html`, который имеет следующий код (листинг 8.10).

Листинг 8.10. Шаблон trans_video.html

Примечание.

В демо-версии книги удалены листинги программ.

В этом шаблоне присутствует тег с видео проигрывателем и тег для вывода текста, распознанного в аудио дорожке видео файла.

Примечание.

В интерактивной цифровой книге для запуска данного примера нужно на вкладке «Показ примеров» нажать на кнопку «Примеры главы 8» и активировать ссылку «Транскрибация видео файлов».

Если выполнить данный программный модуль, то в окне браузера отобразится следующий результат (рис. 8.4).

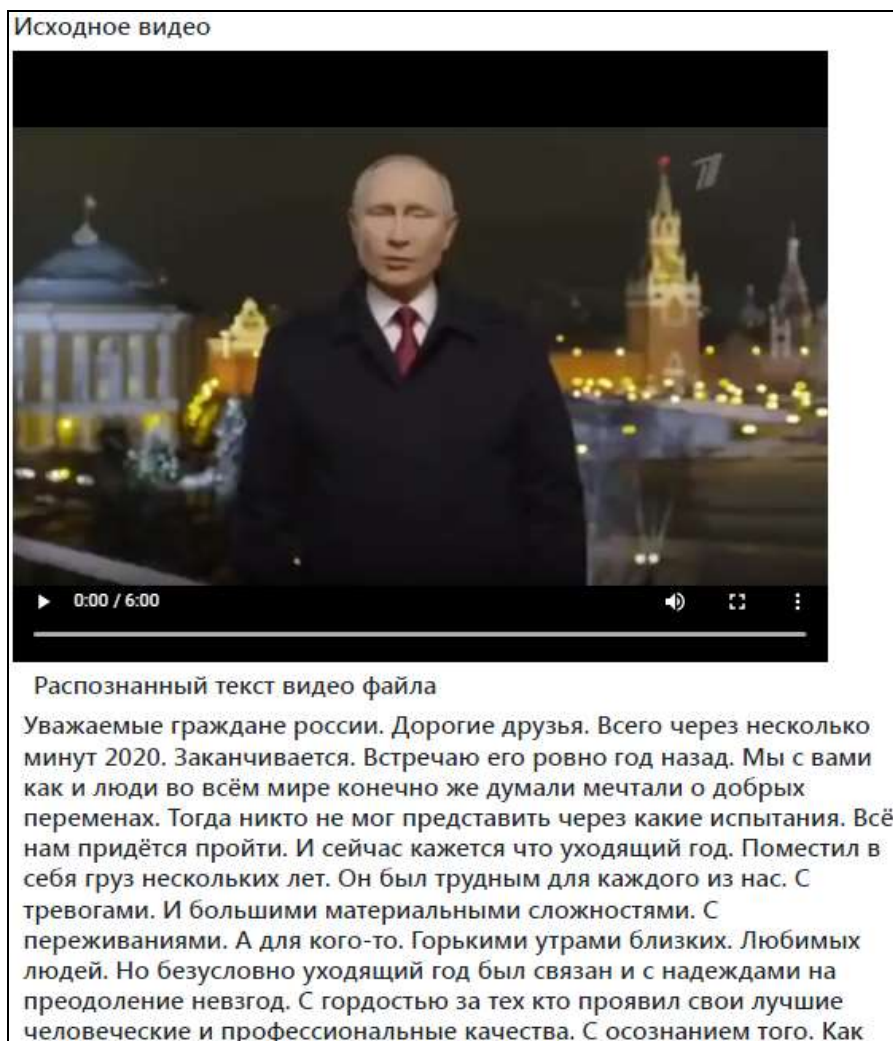


Рис.8.4. Шаблон `trans_video.html` с видео проигрывателем и фрагментом текста, извлеченного из видео файла

Если нажать на кнопку старта проигрывания видео файла, то мы воспроизведем видео и услышим исходную аудио дорожку.

8.6. Краткие итоги

Итак, мы познакомились с некоторыми базовыми принципами транскрибации медиа контента и разобрали используемые для этих целей коды программных модулей. Теперь вы можете выполнять извлечение текста из аудио и видео файлов. В приведенных программных модулях для упрощения примеров исходные файлы были заранее положены в медиа папки проекта. Теперь, с использованием модулей этих примеров, вы сможете самостоятельно разработать веб сервисы, которые позволят удаленным пользователям загружать в приложение любой медиа контент, выполнять его обработку, выводить результаты на html

страницы и скачивать итоговые файлы с публичного сервера на свой компьютер. Но для этого нужно будет дополнительно задействовать базу данных, модели таблиц и формы для загрузки пользовательских данных.

Послесловие

В данной цифровой интерактивной книге были рассмотрены только некоторые базовые приемы использования компьютерного зрения и нейронных сетей в приложениях на Python и Django. По возможности были максимально упрощены примеры и шаблоны html страниц. Основная цель книги – познакомить читателей с некоторыми тонкостями интегрирования нейронных сетей с приложениями на Python и Django. Если после знакомства с приведенными материалами, вы почувствовали, что Python и Django прекрасно взаимодействуют с библиотеками компьютерного зрения и нейронными сетями, то основная цель данного издания достигнута.

В данном издании охвачены далеко не все тонкости использования элементов искусственного интеллекта с web приложений. Достаточно актуальными являются вопросы удаленной загрузки и хранения изображений и медиа контента в таблицах баз данных, выборки из БД загруженной информации с последующей обработкой нейронными сетями и выдачи результатов в интернет браузер. Создание аудио и видео чатов между пользователями сети интернет с использованием голосовых команд, голосовое взаимодействие с телеграмм ботами, создание обучающих курсов с использованием нейронных сетей для формирования видео контента и т.п. Если данное издание окажется востребованным, то следующим шагом автора, будет подготовка интерактивной цифровой книги по освещению этих технологий.

А в завершении остается пожелать удачного применения в своей практической деятельности тех навыков и знаний, которые вы приобрели при изучении данного материала.

д.т.н., профессор Анатолий Постолинт

Список источников и литературы

1. Встроенные кнопки в Telegram Bot API — pyTelegramBotAPI.
<https://habr.com/ru/articles/335886/>
2. Голосовой помощник на Python. <https://pythonist.ru/golosovoj-pomoshchnik-na-python/>
3. Документация на pyTelegramBotAPI. <https://pytba.readthedocs.io/ru/latest/index.html>
4. Загружаем telegram-бота на облачный сервер. <https://www.ixbt.com/live/sw/zagruzhaem-telegram-bota-na-virtualnyy-server.html?ysclid=m4rfaljv8j182778152>
5. Извлечение текста из файлов PDF при помощи Python.
<https://habr.com/ru/companies/ruvds/articles/765246/>
6. Как простой Python-скрипт с использованием ИИ может оптимизировать ваш рабочий процесс. <https://habr.com/ru/companies/bothub/articles/838260/>
7. Навигация и вкладки. <https://bootstrap-5.ru/docs/5.0/components/navs-tabs/?ysclid=lp84mg3zwc988494364>.
8. Постоит А. Python, Django и Bootstrap для начинающих. Санкт-Петербург. «БХВ-Петербург», 2023. с. 623.
9. Размещаем бота для Telegram: от выбора хостинга до запуска.
<https://timeweb.com/ru/community/articles/razmeshchaem-bota-dlya-telegram-ot-vybora-hostinga-do-zapuska>
10. Решаем задачу перевода русской речи в текст с помощью Python и библиотеки Vosk.
<https://proglab.io/p/reshaem-zadachu-perevoda-russkoy-rechi-v-tekst-s-pomoshchyu-python-i-biblioteki-vosk-2022-06-30>
11. Руководство по веб-фреймворку Django. <https://metanit.com/python/django/>.
12. Django работа с переменными в шаблоне. <https://indigotlt.ru/django-rabota-s-peremennymi-v-shablone/>.
13. WEB учебник. <https://www.schoolsw3.com/>.
14. Самый полный стартовый гайд по ботам Telegram (python).
<https://habr.com/ru/articles/697052/>
15. Создание аудиокниг с помощью Python. <https://pythonist.ru/sozдание-audioknig-s-pomoshhyu-python/>
16. Создаем и изменяем PDF файлы в Python.
<https://awesomeandrew.ru/2020/11/16/%D1%81%D0%BE%D0%B7%D0%B4%D0%B0%D0%B5%D0%BC-%D0%B8-%D0%B8%D0%B7%D0%BC%D0%B5%D0%BD%D1%8F%D0%B5%D0%BC-pdf-%D1%84%D0%B0%D0%B9%D0%BB%D1%8B-%D0%B2-python/>
17. Создание кнопок для телеграмм бота с использованием библиотеки pyTelegramBotAPI.
<https://habr.com/ru/sandbox/163347/>
18. Создание телеграмм бота на Django + pyTelegramBotApi.
<https://habr.com/ru/sandbox/186542/>

19. Django 5 с ботом на python-telegram-bot 21.3. <https://yourtodo.life/ru/posts/django-5-s-botom-na-python-telegram-bot-213/>
20. Yandex Translate API создания переводчика на Python.
<https://ramziv.com/article/4?ysclid=lsup81h25o622799937>
21. Neural Style Transfer with OpenCV. <https://pyimagesearch.com/2018/08/27/neural-style-transfer-with-opencv/>
22. How To Perform Neural Style Transfer with Python 3 and PyTorch.
<https://www.digitalocean.com/community/tutorials/how-to-perform-neural-style-transfer-with-python-3-and-pytorch>
23. PocketSphinx 5.0.4 is released! <https://cmusphinx.github.io/>

Приложение 1. Листинг 2.13 программного модуля net.py (вызывается из модуля M_S_21.py)

Примечание.

В демо-версии книги удалены листинги программ.

Приложение 2. Листинг 3.2 (код модуля M_S_21.py с включенным в него кодом модуля net)

Примечание.

В демо-версии книги удалены листинги программ.

Приложение 3. Листинг 3.5 (модуль bot_buton_1.py)

Примечание.

В демо-версии книги удалены листинги программ.

Приложение 4. Листинг 3.6 (модуль `asin_bot.py`)

Примечание.

В демо-версии книги удалены листинги программ.