СОЗДАНИЕ ГОЛОСОВОГО АССИСТЕНТА ДЛЯ ОПЕРАЦИОННОЙ СИСТЕМЫ WINDOWS

К. Е. Филей

Белорусский государственный университет, пр. Независимости, 4, 220030, г. Минск, Беларусь, rct.filey@bsu.by
Научный руководитель — И. Э. Хейдоров, кандидат физико-математических наук, доцент

В статье рассматривается создание и работа голосового ассистента для операционной системы Windows, разработанного на базе среды разработки PyCharm и основных библиотек Python: speech recognition, fuzzywuzzy, pyttsx3.

Ключевые слова: голосовой ассистент; Windows; speech recognition; fuzzywuzzy; pyttsx3; распознавание речи; синтез речи; Python.

ВВЕДЕНИЕ

В последние годы голосовые ассистенты активно внедряются в мобильные устройства и настольные операционные системы. Технологии, такие как Amazon Alexa, Google Assistant и Apple Siri, завоевали популярность благодаря своей способности облегчить взаимодействие с устройствами, выполняя команды, предоставляя информацию и автоматизируя задачи. Однако на рынке голосовых ассистентов для Windows существует дефицит решений с аналогичным уровнем функциональности и интеграции.

Цель создания голосового ассистента для Windows - повысить удобство и эффективность взаимодействия с компьютером. Он станет неотъемлемой частью системы, обеспечивая простой и естественный способ работы с компьютером с помощью голоса.

Для достижения этой цели необходимо решить следующие задачи: провести обзор существующих технологий в области голосовых ассистентов, разработать архитектуру системы, включающую модули распознавания речи, обработки естественного языка и генерации речи, реализовать прототип ассистента с использованием выбранных технологий, а также провести тестирование для оценки функциональности и производительности.

АРХИТЕКТУРА ГОЛОСОВОГО АССИСТЕНТА

Работа голосового ассистента для операционной системы Windows включает несколько ключевых этапов: преобразование аудиосигнала в

текст, анализ и понимание пользовательских команд, выполнение функций, а также формирование и синтез речевого отклика.

Первым и важным этапом в работе голосового ассистента является преобразование аудиосигнала в текст. Этот процесс начинается со сбора аудиосигнала через микрофон. Для этого используется модуль PyAudio, который позволяет захватывать и передавать аудиоданные в реальном времени. Затем аудиосигнал проходит предобработку, включая фильтрацию шумов и нормализацию уровня громкости.

После предобработки аудиосигнал передается в библиотеку speech_recognition, где используется модуль Google Voice Recognition для распознавания речи. Эта библиотека выбрана из-за её точности и скорости, особенно в условиях русского языка. Единственным минусом является необходимость подключения к Интернету.[1]

После преобразования аудиосигнала в текст начинается второй важный этап — анализ и понимание команд пользователя. На этом этапе текст проходит через несколько шагов обработки. Происходит разбиение текста на отдельные слова или токены, приведение слов к их начальной форме для упрощения анализа, исключение незначительных слов например, предлогов для улучшения качества анализа, определение типа команды например, запуск приложения, поиск информации, идентификация основных действий и параметров команды.

Для обеспечения точного распознавания команд, даже если пользователь говорит нечетко, используется библиотека fuzzywuzzy. Она реализует алгоритмы нечеткого сравнения, такие как расстояние Левенштейна, что позволяет сопоставлять и интерпретировать команды с высокой точностью.[2]

Выполнение и отладка функций — это третий этап, на котором голосовой ассистент интерпретирует пользовательскую команду и предпринимает соответствующие действия. Этот процесс включает несколько важных шагов: определение последовательности действий, выполнение команд, обработка ошибок, отладка.

Формирование и синтез речевого отклика — это заключительный этап в работе голосового ассистента, на котором система генерирует и воспроизводит ответ пользователю. Этот процесс включает несколько шагов: генерация текстового отклика, синтез речи, вывод речевого отклика.

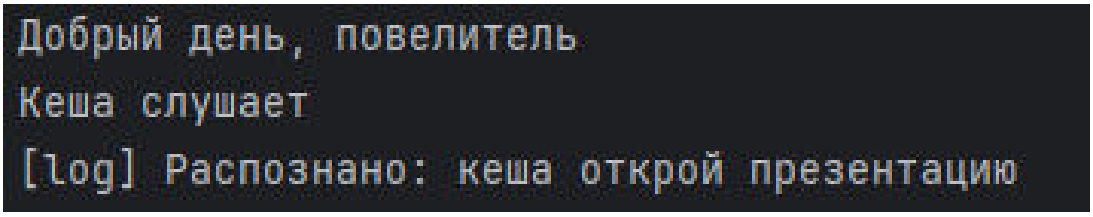
Для преобразования текста в речь нам понадобится pyttsx3. Интуитивно понятный API библиотеки pyttsx3 значительно упрощает интеграцию синтеза речи в приложения. Разработчики могут без особых усилий настраивать голос, скорость, громкость и другие параметры.[3]

Благодаря модульной архитектуре, pyttsx3 легко расширяется новыми возможностями. Разработчики могут интегрировать дополнительные возможности, такие как поддержка нескольких голосов, распознавание речи или интеграция с облачными сервисами.[3]

В процессе разработки голосового ассистента использовались несколько инструментов. Первым инструментом является язык программирования Python, который выбран за его популярность, простоту использования и наличие обширных библиотек для обработки и получения данных. Вторым инструментом явились библиотеки: pyttsx3, speech_recognition, os, fuzzywuzzy, datetime, time, webbrowser, re, openai, requests, bs4, pyautogui, tkinter.

ПРОГРАММНАЯ РЕАЛИЗАЦИЯ ГОЛОСОВОГО АССИСТЕНТА

Процесс выполнения команд происходит следующим образом. Пользователь голосом формулирует команду, например, "открой презентацию". Ассистент захватывает аудиосигнал через микрофон и передаёт его в модуль распознавания речи. Аудиосигнал обрабатывается, фильтруется от шумов и нормализуется по уровню громкости. Затем с помощью библиотеки "speech_recognition" и модуля "Google Voice Recognition" аудиосигнал преобразуется в текстовый запрос. Текстовый запрос проходит через несколько этапов обработки: токенизация, лемматизация, удаление стоп-слов, классификация команд и идентификация ключевых слов и параметров. На этом этапе система определяет, что запрос пользователя относится к запуску презентации. Ассистент интерпретирует текстовый запрос и определяет последовательность действий для выполнения команды. В данном примере это будет открытие программы PowerPoint. Ассистент использует соответствующий системный вызов или библиотеку, такую как "os", для запуска PowerPoint. В процессе выполнения команды ассистент выводит промежуточные или финальные результаты в консоль для целей отладки и мониторинга. В данном случае текстовый запрос "открой презентацию" будет выведен в консоль (см.рис.), подтверждая успешную интерпретацию и выполнение команды. После выполнения команды ассистент формирует текстовый отклик для пользователя, подтверждая выполнение задачи. Этот текстовый отклик преобразуется в аудиоформат с помощью библиотеки pyttsx3 и воспроизводится пользователю.



Результат распознавания голосовых команд

ЗАКЛЮЧЕНИЕ

В итоге был разработан голосовой ассистент для операционной системы Windows, позволяющий с помощью голосового обращения пользователя находить любую информацию, например, открывать статью на заданную тему, включать музыку, загружать презентации, взаимодействовать с клавиатурой компьютера, рассказывать анекдоты, а также говорить точное время по МСК на основе представленных голосовых данных. При этом он проявляет устойчивость к шуму и избегает ошибок, не делая неоправданных выводов без достаточной уверенности. Использование специфических библиотек и архитектуры позволило эффективно анализировать входные данные и использовать их для ответа на поставленный запрос.

В процессе тестового использования ассистент успешно выполнял запросы на основе голосовых данных, демонстрируя устойчивость к нерелевантным словам.

Библиографические ссылки

1. Документация библиотеки SpeechRecognition // Python Package Index [Электронный ресурс]. URL: https://pypi.org/project/SpeechRecognition/ (дата обращения 20.02.2024).

2. Документация библиотеки FuzzyWuzzy // Python Package Index [Электронный ресурс]. URL: https://pypi.org/project/fuzzywuzzy/ (дата обращения 20.02.2024).

3. Документация библиотеки Pyttsx3 // Python Package Index [Электронный ресурс]. URL: https://pypi.org/project/pyttsx3/ (дата обращения 20.02.2024).