Муниципальное бюджетное общеобразовательное учреждение

"Средняя общеобразовательная школа № 76

имени Идоленко И.Н. города Белово"

**ПРИЛОЖЕНИЕ “ГОЛОСОВОЙ ПОМОЩНИК”**

Подготовил:

Овчинников Егор Дмитриевич,

ученик 11 класса

МБОУ «СОШ № 76 города Белово»

Руководитель:

Стешенкова Екатерина Викторовна, преподаватель кафедры экономических наук и информационных технологий БИФ КемГУ

Белово 2023

**Оглавление**

Введение 2

**Глава 1.** Теоретическая часть 3

**Глава 2.** Практическая часть 4

Заключение 6

Интернет ресурсы 7

Приложение 1 8

**Введение**

В начале XXI века бурное развитие получили технологии машинного обучения и искусственного интеллекта. Многие компании стали вкладывать крупные финансовые и материальные ресурсы в исследования в этой области. Одним из наиболее перспективных направлений была выделена разработка технологий распознавания человеческой речи и создания на их базе умных голосовых помощников. Такое внимание к этой теме связано с тем, что в будущем основным средством общением между человеком и машиной(компьютером) может стать именно голос, вытеснив обычные для нас сегодня клавиатуру, компьютерную мышь и сенсорный экран. На сегодняшний день голосовые помощники уже научились управлять телевизорами, чайниками, лампами, умеют совершать поиск информации в интернете или включать интересующую вас музыку всего лишь по одной вашей команде. Таким образом, **актуальность нашей работы** заключается в том, что в современном мире большое внимание уделяется развитию цифровых голосовых помощник, как будущим средствам взаимодействия человека и машины.

Узнав эту информацию нам стало интересно, а возможно ли самому у себя дома создать собственного разговаривающего помощника. Исходя из этого мы поставили перед собой **цель** – разработать приложение умного цифрового помощника, с голосовым управлением, с использованием современных технологий в области машинного обучения и распознавания речи.

Для выполнения цели решались следующие **задачи**:

* Изучить инструменты для распознавания и обработки речи;
* Изучить алгоритм работы голосовых помощников;
* Выбрать инструменты для создания программы;
* Анализ и обобщение полученной информации.

Объект исследования – технологии распознавания речи.

Предмет исследования – приложение голосового помощника.

**Практическая значимость** нашей работы заключается в том, что разработанную программу и полученную информацию можно использовать на уроках информатики и ИКТ в старших классах школ.

**Глава 1. Теоретическая часть**

Голосовой помощник – это приложение на основе машинного обучения, распознающее человеческую речь и способное выполнять определенные действия в ответ на голосовой запрос. Такие приложения умеют управлять многими бытовыми приборами, от лампочки до робота пылесоса, а также совершать интернет запросы, звонки, устанавливать таймеры, вести диалог и др. Сегодня подобные системы можно встретить в смартфонах, персональных компьютерах, телевизорах, умных колонках, они используются людьми из-за простоты, быстроты и удобства их использования. Наиболее распространенными и передовыми приложениями данного типа сегодняшний день являются Apple Siri, Google Assistant, Yandex Алиса и Amazon Alexa.

Поняв, что представляет из голосовой помощник давайте разберемся с тем, как он работает. Алгоритм работы(приложение 1, схема 1) голосового помощника можно разделить на несколько основных этапов. Первый этап – это распознавание произнесенной фразы. Так если сказать боту “Кто такой Александр Сергеевич Пушкин ?” , то он услышит звуковой сигнал, в котором одни звуки переходят в другие, а не пять раздельных слов. Из этого следует, что основная задача системы распознавания речи заключается восстановлении фразы человека по полученному звуковому сигналу. Иногда этот процесс может осложнятся тем, что одна и тажа фраза, произнесеная разными людьми, будет распознана как разные сигналы, из-за чего программа может неправильно понять смысл сказанной команды. Для того, чтобы точно определять слова в заданном вопросе, полученная аудиодорожка делится на фонемы и множество более мелких частей – фреймов, длиной в несколько сотых долей секунды. После этого фреймы подвергаются математическим преобразованиям, в результате которых выделяются коэффициенты, описывающие частотные характеристики каждого фрейма в отдельности. На основании полученных данных помощник может определить к какой фонеме соответствует фрейм, и какое было сказано слово. В случаях когда голосовому помощнику не удается хорошо распознать сказанную фразу, он начинает достраивать ее исходя из контекста и статистических данных. Для этого программа имеет таблицу вероятностей переходов, где указано, что например после буквы “б” с одной вероятностью идет “а”, а с другой “и”. Это позволяет определять варианты последовательности фонем и затем варианты слов, которые могли быть сказаны пользователем.

Следующим этапом, после распознавания и перевода речи в текст, идет определение намерений пользователя. Голосовой помощник разделяет полученную фразу на слова и знаки препинания, после чего производит классификацию запроса по ключевым словам. Для этого используется словарь, в котором хранятся сценарии намерений пользователя, ключевые слова для их запуска и возможные быстрые ответы. Например если, пользователь скажет: “Какая сегодня будет погода?”, то ключевым словом будут слова “сегодня” и “погода”, после чего голосовой помощник запустит сценарий с поиском прогноза погоды на сегодня. Часто бывает, что человек говорит не всю нужную для ответа информацию, из-за чего программе приходится самой ее достраивать. Она это делает либо попросив пользователя повторить свой запрос, либо задав уточняющий вопрос. Эти методы позволяют программе дать подходящий ответ.

Завершающим этапом работы голосового помощника является вывод ответа пользователю. Для этого программа использует синтезатор текста в речь, проигрывая заранее заготовленные ответы или используя генератор фраз.

Таким образом, разобравшись с алгоритмом работы голосового ассистента и с тем как происходит распознавание речи мы можем приступить к созданию нашего собственного помощника.

**Глава 2: Практическая часть**

Перед тем как начать программировать приложение, нам предстояло

выбрать операционную систему, для которой будет сделана программа, язык программирования и IDE(интегрированная среда разработки), программа в которой мы будем заниматься созданием нашего приложения, а также определить функционал помощника. Платформой, для которой будет создан наш помощник, мы выбрали Microsoft Windows, версий 8.1, 10 и 11. Наш выбор обусловлен тем, что данная система является одной из наиболее распространенных и позволяет без особых сложностей запускать любое программное обеспечение. В качестве языка программирования для нашей программы мы выбрали Python 3.7, так как это высокоуровневый язык программирования, имеющий большое количество библиотек для распознавания речи и машинного обучения, а также большое активное сообщество программистов со всего мира. При работе мы активно использовали IDE Microsoft Visual Studio 2019, позволяющую писать код, следить за изменениями и ошибками в коде, запускать тестовые версии программы и создавать исполнительные файлы программ. Функционал приложения нами был определен следующим образом: возможность поиска в интернете, установка таймера и функция приветствия и прощания. Выбрав платформу, средства для разработки и определившись с первоначальным функционалом, мы приступили к написанию нашего голосового помощника.

Сначала мы занялись написанием модуля распознавания и обработки речи, так как это является основой на которой будут основываться остальные части приложения. Для этого мы воспользовались открытыми библиотеками PyAudio и Speech Recognition. PyAudio подключается с помощью команды import pyaudio и позволяет нашему приложению подключаться к микрофону устройства: телефона, планшета, компьютера. Speech Recognition, подключается командой import speech\_recognition as sr, является оболочкой для разных речевых API, в нашем проекте мы используем – Google Web Speech API, и отвечает за распознавание и обработку команд произнесенных пользователем. Код модуля представлен ниже: def zapis\_and\_analiz():

data = ""

with mic:

rec.adjust\_for\_ambient\_noise(mic, duration=0.5)

try:

audio = rec.listen(mic)

except sr.WaitTimeoutError:

return

try:

data = rec.recognize\_google(audio, language="ru")

except sr.UnknownValueError:

pass

return data

После этого мы приступили к написанию модуля отвечающего за синтез речи при проигрывании ответов пользователю. Для этого мы использовали библиотеку PyTTSx3. Она позволяет озвучивать фразы и текст на различных языках при помощи встроенных драйверов платформы, в нашем случае SAPI5 для Windows. Код этого модуля состоит из двух частей. Первая часть отвечает за настройку параметров синтезатора, таких как язык, пол, и выглядит следующим образом:

def setup\_voice():

voices = tts.getProperty('voices')

if VA.sp\_language == "ru":

VA.re\_language = "ru-RU"

if VA.gender == "femail":

tts.setProperty('voices', voices[0].id)

else:

VA.re\_language = "en-US"

tts.setProperty('voices', voices[1].id)

Вторая часть отвечает за проигрывание фраз и выглядит вот так:

def speech\_play(text\_speech):

tts.say(str(text\_speech))

tts.runAndWait()

Закончив с основой нашего ассистента мы занялись обработкой входящих команд. Для этого мы использовали JSON-подобный объект, в котором хранятся названия функций для выполнения и ключи при которых эти функции начнут исполнятся. По итогу у нас получился следующий код позволяющий определять и запускать нужные пользователю функции:

commands = {

("Привет", "день", "вечер", "Hi", "Здравствуй"): start\_speech,

("стоп", "закончили", "конец"): finish\_speech,

("найди", "скажи", "когда", "Почему"): internet\_zapros,

("Поставь", "Напомни"): timer,}

Графический интерфейс мы сделали на основе популярной библиотеки Kivy. Эта библиотека позволяет создавать интерфейсы для мобильных и компьютерных приложений и создавать исполнительные файлы для Windows, MacOS, Linux и Android. Интерфейс нашего помощника(приложение, рис.) состоит из кнопки, находящейся по центру экрана, на которой отображается команда произнесенная пользователем и ответ найденный ассистентом.

Создать исполнительный файл для Windows мы смогли с помощью встроенных средств IDE.

Таким образом, у нас получилось приложение умного голосового помощника, с функциями интернет запроса и таймера, способное работать на мобильной операционной системе Android.

**Заключение**

Подводя итоги нашего проекта можно сделать вывод, что поставленной цели мы достигли: нам удалось создать цифрового голосового помощника способного воспринимать голосовые команды и давать ответ с помощью человеческой речи.

В процессе работы мы:

* изучили способ распознавания и обработки речи;
* изучили алгоритм работы голосовых ассистентов;
* изучили современные технологии для распознавания и синтеза речи;

В дальнейшем мы хотим дорабатывать наше приложение, добавив поддержку других платформ, помимо Windows, а также функции для управления умным домом.

Созданное приложение и собранный материал по данной теме, который в дальнейшем можно будет использовать как учебное пособие на уроках информатики и ИКТ в старших классах школ, можно скачать и посмотреть отсканировав QR-код(приложение 1, фото 1) и перейдя на страницу проекта в GitHub.

**Интернет-ресурсы**

1. <https://pyttsx3.readthedocs.io/en/latest/engine.html> Документация по библиотеке pyttsx3(дата обращения 01.12.2022г.)
2. <https://pyneng.readthedocs.io/ru/latest/book/17_serialization/json.html> Работа с файлами в формате JSON(дата обращения 15.12.2022г.)
3. <https://github.com/Uberi/speech_recognition/blob/master/reference/library-reference.rst> Справочник по библиотеке распознавания речи speech\_recognition (дата обращения 25.01.2023г.)
4. <https://kivy.org/doc/stable/guide/basic.html> Руководство по программированию » Основы Kivy(дата обращения 20.02.2023г.)
5. <https://docs.github.com/ru> Документация по GitHub(дата обращения 10.02.2023г.)

**Приложение 1**

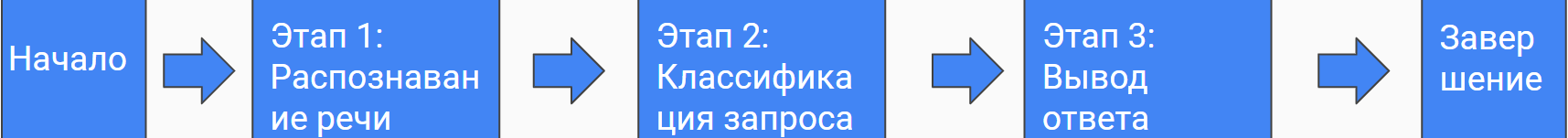


Схема 1

Определение сказанной фразы

Запись фразы в аудиофайл

Деление файла на фонемы

Деление фонем на фреймы

Выделение коэффициентов

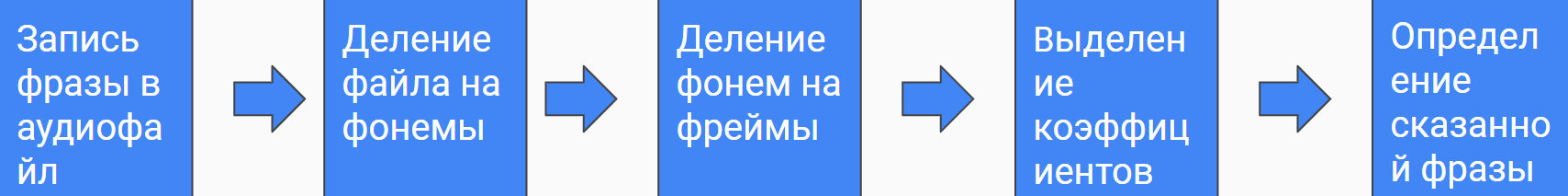


Схема 2



Фото 1