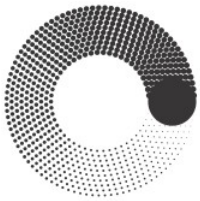
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ



МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Факультет информационных технологий
Кафедра Информатики и информационных технологий

направление подготовки
09.04.02 «Информационные системы и технологии»,

ПРАКТИЧЕСКАЯ РАБОТА № 4

Дисциплина: Искусственный интеллект в мобильных системах

Тема: Разработка программы распознавания и генерации речи

Выполнил(а): студент(ка) группы 224-371 Лейн Ф. Е.
(Фамилия И.О.)

Проверил(а): Попов Д.И.
(Фамилия И.О.)

Дата, подпись _____ _____
(Дата) (Подпись)

Москва
2024

ОГЛАВЛЕНИЕ

ЗАДАНИЕ

Используя материалы лекции "Распознавание и генерация речи в мобильных системах" написать программу на Python, которая бы воспринимала 7-8 голосовых команд и реагировала на голосовую команду stop - окончить работу.

ХОД РАБОТЫ

Для реализации преобразования текста в речь была использована библиотека pyttsx3. С её помощью вы можете создавать программы, которые могут озвучивать текстовые сообщения или предупреждения. Эта библиотека обеспечивает простой интерфейс для работы с различными системами синтеза речи, включая SAPI5 на Windows, NSSpeechSynthesizer на macOS и espeak на Linux.

С помощью pyttsx3 можно управлять параметрами речи, такими как скорость, высота голоса, громкость и другие. Она поддерживает как синхронный, так и асинхронный способы озвучивания текста. Процесс инициализации голосового движка показан в листинге 1.

Листинг 1 — инициализация библиотеки pyttsx3

```
# Initialize TTS
logging.info("Initializing pyttsx3 voice engine")
voice_engine = pyttsx3.init()
voices = voice_engine.getProperty("voices")
logging.info(f"Available: {len(voices)} voices")
for i, voice in enumerate(voices):
print(i, voice.id, voice)

# Selecting voices
voice_engine.setProperty("voice", voices[VOICE_ID].id)
voice_engine.setProperty("rate", VOICE_RATE)
voice_engine.setProperty("volume", VOICE_VOLUME)
```

Для распознавания речи используется пакет от Google из библиотеки speech_recognition. Процесс генерации речи и распознавания речи в текст (команды) показан в листинге 2.

Листинг 2 — преобразование текста в речь и распознавание речи

```
def talk(voice_engine: pyttsx3.engine.Engine, words: str) -> None:
"""Converts text to speech

Args:
voice_engine (pyttsx3.engine.Engine): initialized pyttsx3 voice engine
words (str): words to say
"""
logging.info(f"Saying: {words}")
```

```
voice_engine.say(words)
voice_engine.runAndWait()


def recognize_command(voice_engine: pyttsx3.engine.Engine) -> str:
"""Tries to recognize command in a recursion

Args:
voice_engine (pyttsx3.engine.Engine): initialized pyttsx3 voice engine

Returns:
str: recognized command
"""
logging.info("Initializing recognizer")
recognizer = sr.Recognizer()
with sr.Microphone() as source:
talk(voice_engine, "Speak now")
recognizer.pause_threshold = 1
# recognizer.adjust_for_ambient_noise(source, duration=1)
logging.info("Listening...")
audio = recognizer.listen(source)

# Try to recognize
try:
command = recognizer.recognize_google(audio).lower()
logging.info(f"Recognized command: {command}")
talk(voice_engine, f"Recognized command: {command}")
recognizer.pause_thresholds = 1

# Recognition error -> recognize again in recursion
except sr.UnknownValueError:
talk(voice_engine, "Command is not recognized. Try again")
command = recognize_command(voice_engine)
return command
```

Ниже представлен список распознаваемых команд

1. Команда "play music": Проигрывает музыку и открывает Spotify.
2. Команда "stop": Останавливает выполнение программы.
3. Команда "tell me a joke": Программа рассказывает шутку про атомы.
4. Команда "search video": Спрашивает пользователя, какое видео искать, затем открывает YouTube с результатами поиска.
5. Команда "find recipe": Спрашивает пользователя, какой рецепт искать, затем открывает Google с результатами поиска.

6. Команда "read book": Спрашивает пользователя, название книги, затем открывает сайт Gutenberg.org с результатами поиска книг.
7. Команда "news": Открывает новостной сайт BBC News.

На рисунках 1 и 2 показан процесс выполнения команды search video. После получения данной команды, задаётся вопрос — What video should I search for? Ответ на данный вопрос будет подставлен в поиск сервиса YouTube

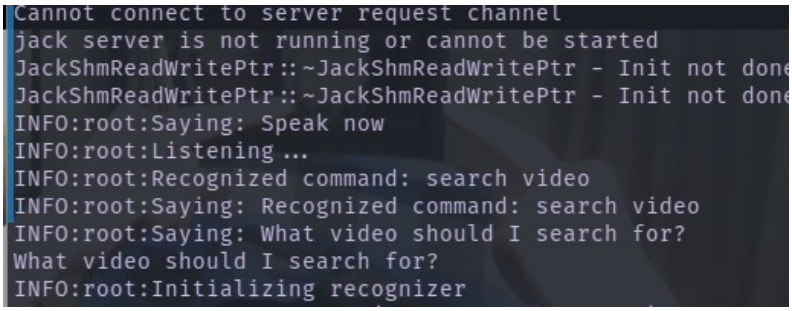


Рисунок 1 — Распознавание команды search video

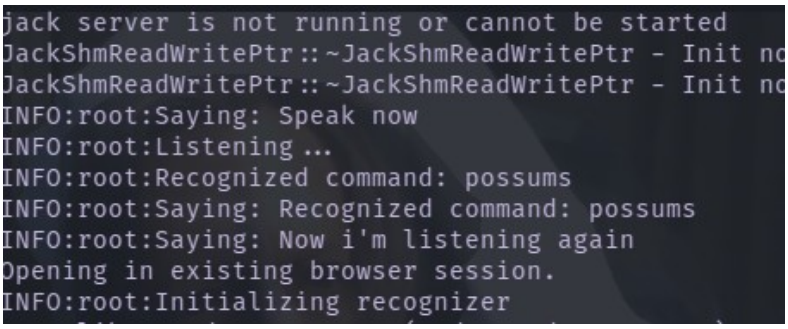


Рисунок 2 — Контекст запроса possums

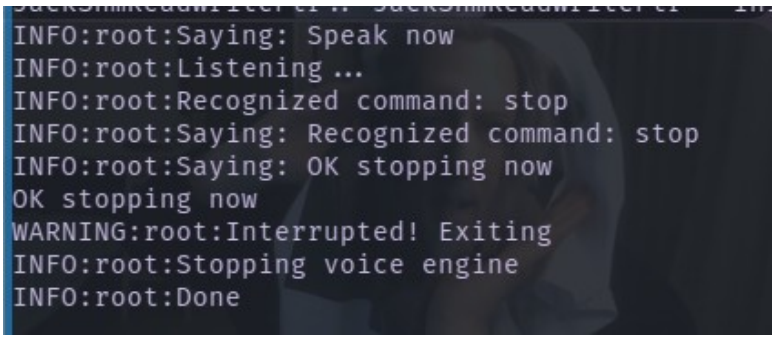


Рисунок 3 — Команда stop

ВЫВОД

В результате выполнения данной практической работы была разработана программа на Python, способная распознавать 7-8 голосовых команд и реагировать на команду "stop" для окончания работы. Для распознавания голосовых команд была использована библиотека pyttsx3, которая предоставляет удобный интерфейс для синтеза речи. Каждая голосовая команда имеет соответствующий ответ программы, открывая определённые веб-ресурсы или предоставляя информацию. Команда "stop" приводит к завершению работы программы. Таким образом, разработанная программа демонстрирует использование распознавания и синтеза речи для создания простого голосового интерфейса на языке Python.

Приложение А — исходный код практической работы

```
import sys
import webbrowser
import logging

import pyttsx3
import speech_recognition as sr


# Selected voice properties
VOICE_ID = 10
VOICE_RATE = 180
VOICE_VOLUME = 1.0


def talk(voice_engine: pyttsx3.engine.Engine, words: str) -> None:
"""Converts text to speech

Args:
voice_engine (pyttsx3.engine.Engine): initialized pyttsx3 voice engine
words (str): words to say
"""
logging.info(f"Saying: {words}")
voice_engine.say(words)
voice_engine.runAndWait()


def recognize_command(voice_engine: pyttsx3.engine.Engine) -> str:
"""Tries to recognize command in a recursion

Args:
voice_engine (pyttsx3.engine.Engine): initialized pyttsx3 voice engine

Returns:
str: recognized command
"""
logging.info("Initializing recognizer")
recognizer = sr.Recognizer()
with sr.Microphone() as source:
talk(voice_engine, "Speak now")
recognizer.pause_threshold = 1
# recognizer.adjust_for_ambient_noise(source, duration=1)
logging.info("Listening...")
audio = recognizer.listen(source)

# Try to recognize
try:
command = recognizer.recognize_google(audio).lower()
logging.info(f"Recognized command: {command}")
talk(voice_engine, f"Recognized command: {command}")
recognizer.pause_thresholds = 1
```

```
# Recognition error -> recognize again in recursion
except sr.UnknownValueError:
talk(voice_engine, "Command is not recognized. Try again")
command = recognize_command(voice_engine)
return command


def execute_command(voice_engine: pyttsx3.engine.Engine, command: str) ->
None:
"""Parses and executes command

Args:
voice_engine (pyttsx3.engine.Engine): initialized pyttsx3 voice engine
command (str): command to execute
"""
if "play music" in command:
talk(voice_engine, "Playing music now")
print("Playing music now")
URL = "https://www.spotify.com"
webbrowser.open(URL)

elif "stop" in command:
talk(voice_engine, "OK stopping now")
print("OK stopping now")
sys.exit()

elif "tell me a joke" in command:
talk(voice_engine, "Why don't scientists trust atoms? Because they make up
everything!")
print("Why don't scientists trust atoms? Because they make up everything!")

elif "search video" in command:
talk(voice_engine, "What video should I search for?")
print("What video should I search for?")
search = recognize_command(voice_engine)
url = "https://www.youtube.com/results?search_query=" + search
webbrowser.open(url)

elif "find recipe" in command:
talk(voice_engine, "What is the recipe?")
print("What is the recipe?")
recipe = recognize_command(voice_engine)
url = "https://www.google.com/search?q=" + recipe + "+recipe"
webbrowser.open(url)

elif "read book" in command:
talk(voice_engine, "What is the name of the book?")
print("What is the name of the book?")
name = recognize_command(voice_engine)
url = "https://www.gutenberg.org/ebooks/search/?query=" + name
webbrowser.open(url)

elif "news" in command:
```

```
talk(voice_engine, "Opening news")
print("Opening news")
url = "https://www.bbc.com/news"
webbrowser.open(url)


def main() -> None:
"""Main entry"""
# Initialize logging
logging.basicConfig(level=logging.INFO)

# Initialize TTS
logging.info("Initializing pyttsx3 voice engine")
voice_engine = pyttsx3.init()
voices = voice_engine.getProperty("voices")
logging.info(f"Available: {len(voices)} voices")
for i, voice in enumerate(voices):
print(i, voice.id, voice)

# Selecting voices
voice_engine.setProperty("voice", voices[VOICE_ID].id)
voice_engine.setProperty("rate", VOICE_RATE)
voice_engine.setProperty("volume", VOICE_VOLUME)

# Main TTS loop
while True:
try:
execute_command(voice_engine, recognize_command(voice_engine))
talk(voice_engine, "Now i'm listening again")

except (SystemExit, KeyboardInterrupt):
logging.warning("Interrupted! Exiting")
break

# Exit gracefully
logging.info("Stopping voice engine")
voice_engine.stop()
logging.info("Done")


if __name__ == "__main__":
main()
```