ГБОУ МО «Одинцовский «Десятый лицей»

**ИНДИВИДУАЛЬНЫЙ ПРОЕКТ**

Цифровой помощник

Работу выполнил:

Ученик 11 «Б» класса,

Севастьянов Матвей Иванович

Руководитель:

Деткова Людмила Анатольевна, учитель информатики

Одинцово

2023

**Оглавление**

Паспорт проекта…………………ю……………………………………………...3

Введение…………………………………………………………………………...4

Проблема……………………………………………………………………4

Актуальность……………………………………………………………….4

Цели…………………………………………………………………………4

Задачи……………………………………………………………………….4

Методы……………………………………………………………………...4

Основная часть…………………………………………………………………….5

Реализация задачи 1……..………………………………………………...5

Реализация задачи 2………………..……………………………………...7

Реализация задачи 3…………………..…………………………………...9

Функционал ассистента……………………………………………..9

Базовый функционал ассистента……………………...…………..10

Навыки ассистента…………………………………………………12

Заключение……………………………………………………………………….17

Источники информации…………………………………………………………18

Приложение………………………………………………………………………19

**Паспорт проекта**

|  |  |
| --- | --- |
| **Название** | Цифровой помощник |
| **Автор** | Севастьянов Матвей, ученик 10 «Б» класса |
| **Научный руководитель** | Деткова Людмила Анатольевна, учитель информатики |
| **Цели** | 1. Изучение языка программирования Python. 2. Создание голосового ассистента. |
| **Задачи** | 1. Ознакомиться с основными концепциями создания голосовых ассистентов, его возможными функциями, программными решениями. 2. Установить на свой компьютер необходимые программы и получить базовые знания для предстоящей работы. 3. Написать базовую программу для ассистента и его навыки. |
| **Даты** | 01.09.2022 ─ 28.03.2023 |
| **Материально-техническое обеспечение** | Компьютер, некоторые программы. |

**Введение**

**Проблема**

Перед всеми старшеклассниками стоит вопрос о профориентации. Своё будущее я хочу связать с программированием. Я решил попробовать создать своего голосового ассистента, чему и посвящён данный проект.

**Актуальность**

Актуальность ассистентов состоит в том, что интеграция голосовых помощников активно внедряется в разные виды жизнедеятельности человека или бизнеса и постепенно становиться неизменной составляющей высокоэффективных и новейших интерактивно-маркетинговых коммуникаций.

**Цели**

1. Изучение языка программирования Python.
2. Создание голосового ассистента.

**Задачи**

1. Ознакомиться с основными концепциями создания голосовых ассистентов, его возможными функциями, программными решениями.
2. Установить на свой компьютер необходимые программы и получить базовые знания для предстоящей работы.
3. Написать базовую программу для ассистента и его навыки.

**Методы**

Для написания программы голосового ассистента я буду использовать дополнительные библиотеки языка программирования Python, такие как pyaudio, SpeechRecognition и т. д. Эти библиотеки полностью свободны, т. е. с открытым исходным кодом.

Мой проект также будет свободным. Весь код будет доступен в репозиториях GitHub (ссылка находится в приложении), его может скачать и собрать каждый. Это один из главных принципов создания, в отличие от коммерческих ассистентов (Яндекс Алиса, Google Assistant, Apple Siri).

Ещё одним плюсом является кроссплатформенность, т. е. возможность выполнения кода на разных операционных системах и платформах (Linux, Windows, MacOS, Android, iOS).

Также ассистент может уметь говорить разными голосами и на разных языках.

**Основная часть**

**Реализация задачи 1**

Принцип работы голосового ассистента на примере «Алисы» от компании «Яндекс»

Рассмотрим, как принципиально устроены голосовые ассистенты на примере «Алисы» от компании «Яндекс».

Так как «Алиса» является интеллектуальным помощником для смартфонов и персональных компьютеров, то она позволяет решать общие задачи пользователей, такие как поиск информации в интернете, поиск мест на карте, прокладывание маршрутов, сообщение прогноза погоды, может поддерживать разговор, развлекать пользователя и т. д. Для этого «Алиса» использует облачные средства компании «Яндекс», к которым она обращается через API посредствам сети Интернет. Схема работы представлена на рис. 1 в приложении.

На первом этапе происходит активация, например, произношением ключевой фразы. Ассистент постоянно прослушивает окружающие звуки, анализирует наличие ключевой фразы и, если она будет распознана, переходит в активный режим.

Дальше пользователь произносит текст, который может объяснить помощнику, что пользователь хочет сделать. Система распознавания (Automatic Speech Recognition) превращает текст в N-лучших гипотез того, что сказал пользователь. Затем система распознавания естественного языка (Natural Language Understanding) превращает текст в N-лучших вариантов понимания фразы пользователя, далее диалоговый движок интерпретирует и классифицирует эти фразы и определяет, что необходимо сделать на основе полученной информации. Например, обратиться в различные сервисы для получения информации.

После получения необходимых данных система производит процесс возвращения информации пользователю, т. е. система генерации естественного языка (Natural Language Generation) генерирует текст для ответа пользователю, далее система генерации голоса (Text-To-Speech) на основе обученных моделей генерирует звуковую информацию, которая и объявляется пользователю в качестве ответной реакции. Кроме ответа, может также происходить любое действие на мобильном телефоне или компьютере, например запуск приложения или поиска информации в поисковой системе.

Недостатки существующих голосовых ассистентов и актуальность развития сферы

Несмотря на то что такие ассистенты существуют уже достаточно долгое время, они так и не получили широкого распространения из-за существования ряда ограничений и сфер, где они не могут быть применены. К основным недостаткам можно отнести: ориентированность на решения общих задач, зависимость от интернета и облачных сервисов, сложность или в большинстве случаев невозможность интеграции со сторонними сервисами, незащищённость персональных данных.

Google Speech-To-Text API

**Что такое API?**

API ─ описание способов взаимодействия одной компьютерной программы с другими. Обычно входит в описание какого-либо интернет-протокола, программного каркаса или стандарта вызовов функций операционной системы. Часто реализуется отдельной программной библиотекой или сервисом операционной системы. Используется программистами при написании всевозможных приложений.

Проще говоря, это набор компонентов, с помощью которых компьютерная программа (бот или же сайт) может использовать другую программу. Google позволяет всем заимствовать или арендовать свои API в обмен на цену, а иногда и бесплатно. Если мы хотим использовать сервис, мы можем просто зайти на их сайт и протестировать его бесплатно.

У Google есть отличный API для распознавания речи. Этот API преобразует произносимый текст (микрофон) в письменный текст (строки Python), вкратце речь в текст. Мы можем просто говорить в микрофон, и Google API переведёт это в письменный текст. API отлично работает с русским и английским языками.

API распознавания речи разгружает логику, так что мы можем просто отправить веб-запрос в API, который затем возвращает распознанный текст. Это делается непосредственно из кода Python, скрипту потребуется доступ в Интернет.

**Реализация задачи 2**

Установка необходимого программного обеспечения

**Python 3.11 —** Высокоуровневый язык программирования общего назначения с динамической строгой типизацией и автоматическим управлением памятью, ориентированный на повышение производительности разработчика, читаемости кода и его качества, а также на обеспечение переносимости написанных на нём программ. Язык является полностью объектно-ориентированным в том плане, что всё является объектами. Необычной особенностью языка является выделение блоков кода пробельными отступами. Синтаксис ядра языка минималистичен, за счёт чего на практике редко возникает необходимость обращаться к документации. Сам же язык известен как интерпретируемый и используется в том числе для написания скриптов.

**Visual Studio Code** — Текстовый редактор, разработанный Microsoft для Windows, Linux и macOS. Позиционируется как «лёгкий» редактор кода для кроссплатформенной разработки веб- и облачных приложений. Включает в себя отладчик, инструменты для работы с Git, подсветку синтаксиса, IntelliSense и средства для рефакторинга. Имеет широкие возможности для кастомизации: пользовательские темы, сочетания клавиш и файлы конфигурации. Распространяется бесплатно, разрабатывается как программное обеспечение с открытым исходным кодом.

**Git** — распределённая система управления версиями. Проект был создан Линусом Торвальдсом для управления разработкой ядра Linux, первая версия выпущена 7 апреля 2005 года.

**ОС Windows —** Группа семейств коммерческих проприетарных операционных систем корпорации Microsoft, ориентированных на управление с помощью графического интерфейса. MS-DOS - является прародителем Windows.

**RHVoice —** бесплатный синтезатор речи с открытым исходным кодом.

**Для работы над проектом я использовал сторонние библиотеки для языка Python**

**PyAudio** **—** PyAudio предоставляет привязки Python для PortAudio v19, кроссплатформенной библиотеки аудио ввода-вывода. С помощью PyAudio вы можете легко использовать Python для воспроизведения и записи аудио на различных платформах, таких как GNU / Linux, Microsoft Windows и Apple macOS.

**Wikipedia API —** Wikipedia-Api - это простая в использовании Python-оболочка для API Википедии. Он поддерживает извлечение текстов, разделов, ссылок, категорий, переводов и т. д. из Википедии. Документация содержит фрагменты кода для наиболее распространённых случаев использования.

**google —** Привязки Python к поисковой системе Google.

**pyttsx3 —** это библиотека преобразования текста в речь на Python. В отличие от альтернативных библиотек, он работает в автономном режиме и совместим как с Python 2, так и с 3.

**SpeechRecognition —** Библиотека для выполнения распознавания речи, с поддержкой нескольких движков и API, онлайн и оффлайн.

**deep-translator —** Гибкий бесплатный и неограниченный инструмент для простого перевода с разных языков с использованием нескольких переводчиков.

**Реализация задачи 3**

**Функционал ассистента**

Базовый функционал

* Распознавание речи пользователя.
* Синтез речи ассистента.

Навыки

* Приветствие и рассказ о себе.
* Вывод доступных команд на экран.
* «Подбросить монетку».
* Поиск информации в Google.
* Поиск видео на YouTube.
* Поиск информации в Википедии.
* Перевод с английского языка на русский и наоборот,
* Открытие базовых программ.

Прежде чем писать и тестировать код, убедимся, что операционная система готова к синтезу речи, в том числе на русском языке.

Чтобы компьютер заговорил, нужны:

* голосовой движок (синтезатор речи) с поддержкой нужных нам языков,
* голоса дикторов для этого движка.

В Windows есть штатный речевой интерфейс Microsoft Speech API (SAPI). Голоса к нему выпускают, помимо Microsoft, сторонние производители: Nuance Communications, Loquendo, Acapela Group, IVONA Software.

Есть и свободные кроссплатформенные голосовые движки:

* eSpeak и его ответвление — eSpeak NG — c поддержкой более 100 языков и диалектов, включая даже латынь. NG означает New Generation — «новое поколение». Эта версия разрабатывается сообществом с тех пор, как автор оригинальной eSpeak перестал выходить на связь. Система озвучит ваш текст в Windows, Android, Linux, Mac, BSD. При этом старый eSpeak стабилен в ОС Windows 7 и XP, а eSpeak NG совместим с Windows 8 и 10.
* RHVoice от Ольги Яковлевой — имеет четыре голоса для русского языка (один мужской и три женских), а также поддерживает татарский, украинский, грузинский, киргизский, эсперанто и английский. Работает в Windows, GNU/Linux и Android.

Я буду использовать RHVoice, т. к. мой ассистент будет кроссплатформенным.

Перейдём к написанию кода проекта

**Базовый функционал**

Для начала нужно импортировать установленные библиотеки Python:

**import** speech\_recognition # распознавание пользовательской речи

**import** pyttsx3 # Синтез речи

**import** webbrowser # Работа с браузером

**import** traceback # вывод traceback без остановки работы

**import** wikipediaapi # найти в википедии

**import** random # модуль рандомайзера

**from** deep\_translator **import** GoogleTranslator #модуль переводчика

**import** subprocess # модуль запуска сторонних программ

Затем необходимо инициализировать синтез речи и найти на компьютере голосовые модули ассистента. Для этого используется библиотеку pyttsx3:

"""Инициализация синтеза речи"""

tts = pyttsx3.init()

voices = tts.getProperty("voices")

# Задать голос по умолчанию

tts.setProperty("voice", "ru")

# Попробовать установить предпочтительный голос

**for** voice **in** voices:

ru = voice.id.find("RHVoice\Anna") # Найти Анну от RHVoice

**if** ru > -1: # если нашли, выбираем этот голос

tts.setProperty("voice", voice.id)

Для преобразования текста в речь напишем функцию:

**def** play\_voice\_assistant\_speech(text\_to\_speech):

"""

:param text\_to\_speech: текст, который нужно преобразовать в

речь

"""

tts.say(str(text\_to\_speech))

tts.runAndWait()

Далее, для преобразования речи пользователя в текст, добавим функцию:

**def** record\_and\_recognize\_audio(\*args: tuple):

"""

Запись и распознавание аудио

"""

**with** microphone:

recognized\_data = ""

# регулирование уровня окружающего шума

recognizer.adjust\_for\_ambient\_noise(microphone, duration=2)

**try**:

**print**("Listening...")

audio = recognizer.listen(microphone, 10, 15)

**except** speech\_recognition.WaitTimeoutError:

**print**("Can you check if your microphone is on, please?")

**return**

# использование online-распознавания через Google

**try**:

**print**("Started recognition...")

recognized\_data = recognizer.recognize\_google(

audio, language="ru"

).lower()

**except** speech\_recognition.UnknownValueError:

**pass**

"""

в случае проблем с доступом в Интернет происходит выброс

ошибки

"""

**except** speech\_recognition.RequestError:

**print**("Check your Internet Connection, please")

play\_voice\_assistant\_speech(

"Пожалуйста, проверьте соединение с Интернетом!"

)

**return** recognized\_data

Для обработки исключений здесь используется конструкция with ─ try ─ except.

Для запуска базовой программы напишем main-функции:

**if** \_\_name\_\_ == "\_\_main\_\_":

# инициализация инструментов распознавания и ввода речи

recognizer = speech\_recognition.Recognizer()

microphone = speech\_recognition.Microphone()

**while** True:

# старт записи речи с последующим выводом распознанной речи

# и удалением записанного в микрофон аудио

voice\_input = record\_and\_recognize\_audio()

**print**('Пользователь:', voice\_input)

**Навыки ассистента**

Для начала создадим словарь с кодовыми фразами и названиями функций:

commands = {

("подбрось", "heads"): flip\_a\_coin,

("hello", "hi", "morning", "привет", "здорова", "хэй"): play\_greetings,

("bye", "goodbye", "quit", "exit", "stop", "пока", "хватит", "стоп"): play\_farewell\_and\_quit,

("victoria", "help", "вика", "виктория", "помощь"): name\_trigger,

("search", "google", "find", "найди", "погода", "прогноз", "гугл", "интернет", "интернете"): search\_for\_term\_on\_google,

("video", "youtube", "watch", "видео", "ютуб"): search\_for\_video\_on\_youtube,

("wikipedia", "definition", "about", "определение", "википедия", "википедии"): search\_for\_definition\_on\_wikipedia,

("translate", "interpretation", "translation", "перевод", "перевести", "переведи", "переводчик"): get\_translation,

}

И их вывод для пользователя:

**print**(

    "Доступные команды: ",

    "Приветствие:                Привет",

    "Помощь (выводит это меню):  Помощь; Виктория",

    "Закончить разговор:         Пока; Хватит; Стоп",

    "Подбросить монетку:         Подбрось монетку; Heads or tails",

    "Запустить переводчик:       Перевод; Перевести; Переведи;",

    "Искать в Google:            Найди; гугл; <запрос>",

    "Искать в Википедии:         Найди в википедии; <запрос>",

    "Искать в Ютуб:              ютуб; youtube <запрос>",

sep="\n"

)

Для выполнения этих команд добавим функцию с дополнительными аргументами:

**def** execute\_command\_with\_name(command\_name: str, \*args: list):

"""

Выполнение заданной пользователем команды с дополнительными аргументами

:param command\_name: название команды

:param args: аргументы, которые будут переданы в функцию

:return:

"""

**for** key **in** commands.keys():

**if** command\_name **in** key:

commands[key](\*args)

**else**:

**pass**

Затем допишем main-функцию. Теперь она включает код для выполнения навыков ассистента:

**if** \_\_name\_\_ == "\_\_main\_\_":

# инициализация инструментов распознавания и ввода речи

recognizer = speech\_recognition.Recognizer()

microphone = speech\_recognition.Microphone()

**while** True:

# старт записи речи с последующим выводом распознанной речи

# и удалением записанного в микрофон аудио

voice\_input = record\_and\_recognize\_audio()

**print**('Пользователь:', voice\_input)

voice\_input = voice\_input.split(" ")

command = voice\_input[0]

command\_options = [

str(input\_part) **for** input\_part **in** voice\_input[1:len(voice\_input)]

]

execute\_command\_with\_name(command, command\_options)

**Далее идёт описание всех функций навыков.**

Поиск видео на YouTube:

**def** search\_for\_video\_on\_youtube(\*args: tuple):

"""

Поиск видео на YouTube с автоматическим открытием ссылки на список результатов

:param args: фраза поискового запроса

"""

**if** **not** args[0]: **return**

search\_term = " ".join(args[0])

url = "https://www.youtube.com/results?search\_query=" + search\_term

webbrowser.get().open(url)

# для мультиязычных голосовых ассистентов лучше создать

# отдельный класс, который будет брать перевод из JSON-файла

**print**("Вот что я нашла по запросу " + search\_term + " в ютуб.")

play\_voice\_assistant\_speech("Вот что я нашла по запросу " + search\_term + " в ютуб.")

Поиск в Google:

**def** search\_for\_term\_on\_google(\*args: tuple):

"""

Поиск в Google с автоматическим открытием ссылок (на список результатов и на сами результаты, если возможно)

:param args: фраза поискового запроса

"""

**if** **not** args[0]: **return**

search\_term = " ".join(args[0])

# открытие ссылки в браузере

url = "https://google.com/search?q=" + search\_term

webbrowser.get().open(url)

**print**("Вот что я нашла по запросу {}.".format(search\_term))

play\_voice\_assistant\_speech("Вот что я нашла по запросу {}.".format(search\_term))

Поиск в Википедии:

**def** search\_for\_definition\_on\_wikipedia(\*args: tuple):

"""

Поиск в Wikipedia определения с последующим озвучиванием результатов и открытием ссылок

:param args: фраза поискового запроса

"""

**if** **not** args[0]: **return**

search\_term = " ".join(args[0])

# установка языка (в данном случае используется язык, на котором говорит ассистент)

wiki = wikipediaapi.Wikipedia("ru")

# поиск страницы по запросу, чтение summary, открытие ссылки на страницу для получения подробной информации

wiki\_page = wiki.page(search\_term)

**try**:

**if** wiki\_page.exists():

**print**("Вот что я нашла по запросу {} в Википедии".format(search\_term))

play\_voice\_assistant\_speech("Вот что я нашла по запросу {} в википедии".format(search\_term))

webbrowser.get().open(wiki\_page.fullurl)

# чтение ассистентом первых двух предложений summary со страницы Wikipedia

# (могут быть проблемы с мультиязычностью)

play\_voice\_assistant\_speech(wiki\_page.summary.split(".")[:2])

**else**:

# открытие ссылки в браузере в случае, если на Wikipedia не удалось найти ничего по запросу

**print**("Такой информации нет в Википедии, вот результаты в google.")

play\_voice\_assistant\_speech("Такой информации нет в Википедии, вот результаты в google.")

url = "https://google.com/search?q=" + search\_term

webbrowser.get().open(url)

# поскольку все ошибки предсказать сложно, то будет произведен отлов с последующим выводом без остановки программы

**except**:

**print**("Seems like we have a trouble. See logs for more information")

play\_voice\_assistant\_speech("Seems like we have a trouble. See logs for more information")

traceback.print\_exc()

**return**

Подбросить монетку:

**def** flip\_a\_coin(\*args: tuple):

l = random.randint(0, 2)

**if** l == 0:

**print**("Орёл!")

play\_voice\_assistant\_speech("Орёл!")

**else**:

**print**("Решка!")

play\_voice\_assistant\_speech("Решка!")

**return**

Помощь:

**def** name\_trigger(\*args: tuple):

**print**("Чем я могу помочь?")

play\_voice\_assistant\_speech("Чем я могу пом+очь?")

**print**(

    "Доступные команды: ",

    "Приветствие:                Привет",

    "Помощь (выводит это меню):  Помощь; Виктория",

    "Закончить разговор:         Пока; Хватит; Стоп",

    "Подбросить монетку:         Подбрось монетку; Heads or tails",

    "Запустить переводчик:       Перевод; Перевести; Переведи;",

    "Искать в Google:            Найди; гугл; <запрос>",

    "Искать в Википедии:         Найди в википедии; <запрос>",

    "Искать в Ютуб:              ютуб; youtube <запрос>",

sep="\n"

)

Переводчик:

**def** get\_translation(\*args: tuple):

"""

Переводчик

:param args:

:return:

"""

**print**("Запускаю навык 'Перевод'!")

play\_voice\_assistant\_speech("Запускаю навык 'Перевод'!")

play\_voice\_assistant\_speech("Говорите целевой язык.")

lang = record\_and\_recognize\_audio()

**if** lang == "русский":

target = "ru"

**else**:

target = "en"

play\_voice\_assistant\_speech("Говорите фразу для перевода.")

to\_translate = record\_and\_recognize\_audio()

translated = GoogleTranslator(source='auto', target=target).translate(to\_translate)

**print**(translated)

play\_voice\_assistant\_speech(translated)

**return**

Поздороваться с пользователем:

**def** play\_greetings(\*args: tuple):

"""

Приветствие пользователя

"""

**print**("Привет, пользователь! Я - голосовой помощник Виктория. Чем я могу помочь вам?")

play\_voice\_assistant\_speech("Привет, пользователь! Я - голосовой помощник Виктория. Чем я могу пом+очь вам?")

Завершение программы:

**def** play\_farewell\_and\_quit(\*args: tuple):

"""

Проигрывание прощальной речи и выход

"""

**print**("До свидания! Хорошего дня!")

play\_voice\_assistant\_speech("До свидания! Хорошего дня!")

tts.stop()

quit()

Открытие базовых программ:

**def** open\_app(\*args: tuple):

"""

Отрыть одну из стандартных программ Windows

"""

apps = {

("блокнот", "notepad"): "notepad",

("калькулятор", "calculator"): "calc",

("браузер", "browser"): "browser",

("wordpad", "вордпад"): "write",

("пэйнт", "paint"): "mspaint",

("проводник", "explorer"): "explorer"

}

args = ''.join(args[0])

error = 0

**for** key **in** apps.keys():

**if** args **in** key:

**if** apps[key] != "browser":

subprocess.Popen(apps[key])

**print**(f'Запускаю {args}...')

play\_voice\_assistant\_speech(f'Запускаю {args}...')

**break**

**else**:

webbrowser.open('https://google.com')

**print**(f'Запускаю {args}...')

play\_voice\_assistant\_speech(f'Запускаю {args}...')

**break**

**else**: error += 1

**if** error >= len(apps):

**print**('Такой программы не найдено...')

play\_voice\_assistant\_speech('Такой программы не найдено.')

**Заключение**

В ходе выполнения проектных задач я получил навыки, необходимые для написания программы голосового ассистента.

Все поставленные задачи выполнены.

**Источники информации**

<https://moluch.ru/archive/360/80416/>

<https://samag.ru/archive/article/3570>

<https://habr.com/ru/post/529590/>

<https://gb.ru/posts/tts_python>

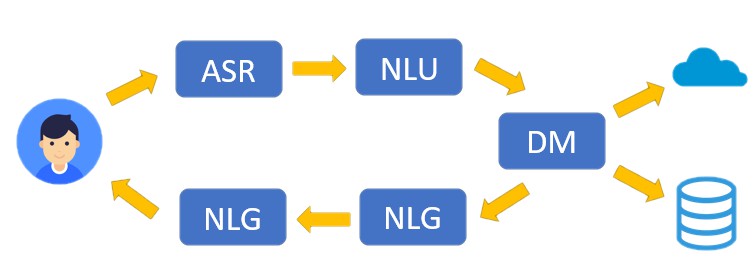
<https://progler.ru/blog/raspoznavanie-rechi-v-python-s-ispolzovaniem-google-speech-api>

<https://pypi.org>

**Приложение**

**GitHub проекта —** <https://github.com/matvey100/victoria-assistant>

Рисунок 1. Принцип работы голосового ассистента «Алиса».



**Программное обеспечение**

Visual Studio Code — <https://code.visualstudio.com/>

Python 3.11 — <https://www.python.org/>

Git — <https://git-scm.com/>

Windows — <https://support.microsoft.com/ru-ru/windows>

RHVoice — <https://github.com/RHVoice/RHVoice>

**Библиотеки Python**

PyAudio — <https://github.com/jleb/pyaudio>

Wikipedia API — <https://github.com/martin-majlis/Wikipedia-API>

Google — <https://pypi.org/project/google/>

pyttsx3 — <https://github.com/nateshmbhat/pyttsx3>

SpeechRecognition — <https://github.com/Uberi/speech_recognition>

deep-translator — <https://github.com/nidhaloff/deep-translator>