

**Компьютерное приложение “QazVoice”.**

**Голосовой ассистент на казахском языке.**

**Автор проекта:**

Ученики 10С класса НИШ ФМН г.Талдыкорган

Дуйсенбиев Есет, Қалияс Алихан

**Руководитель работы:**

Учитель информатики НИШ ФМН г.Талыдкорган

Искакова Айжан Тоқтарбаевна

**Направление:**

“Математическое моделирование экономических и социальных процессов”

**Секция:**

Экономика, Информатика

г.Талдыкорган 2023 год

Оглавление

[Аннотация](#аннотация)...................................................................................2

[Введение](#введение)......................................................................................3

[Исследовательская часть](#исследовательская)............................................................5

[Функциональные возможности ассистента QazVoice](#функциональныевозможности)….........9

[Этапы создания ассистента QazVoice(дневник](#дневник))……………..10

[Проблемы с которыми мы столкнулись](#проблемы)....................................11

[Коды и объяснение их роли в програмном коде](#коды)…………......14

[Заключение](#заключение)..................................................................................31

[Планы на будущее](#планы).......................................................................33

[Отзыв руководителя на работу](#отзыв)...................................................34

[Список использованных источников и ресурсов](#Список_литературы).....................35

**Аннотация**

В мире технологий и 21-го века голосовые ассистенты стали неотъемлемой частью повседневной жизни. Они значительно упрощают взаимодействие человека с компьютерами и устройствами, позволяя выполнять разнообразные задачи, используя голосовые команды. Однако, в большинстве случаев, такие ассистенты ориентированы на англоязычную и на русскоязычную аудиторию в нашей стране, что ограничивает доступность данной технологии для людей, говорящих на казахском языке.

Проект "Голосовой ассистент на казахском языке QazVoice" стремится заполнить этот пробел и создать инновационное решение, которое позволит казахоязычным пользователям в полной мере воспользоваться преимуществами голосового управления. Наш проект будет ориентирован на адаптацию и локализацию существующих технологических разработок для казахской аудитории.

С помощью "QazVoice" пользователи смогут выполнять различные задачи, используя только свой голос на родном языке. Этот проект не только сделает технологии доступными для казахскоязычной аудитории, но также способствует развитию и совершенствованию голосовых технологий в Казахстане.

“QazVoice” это голосовой ассистент, который понимает казахский язык и отвечает пользователю на казахском языке. Бот работает когда слышит триггер. Триггером для бота является ключевые строки ‘али’, 'али' , 'алибек' , 'әли' , 'ғали' В основе бота лежит локальная база данных команд. При распознавании триггера модель обрабатывает другие услышанные слова и в случае распознавания других ключевых команд с базы данных data\_set бот отвечает соответсвуещим ответом и выполняет команду.

Например, вы хотите узнать погоду в вашем городе и вызываете ассистента по имени и даете команду узнать погоду в городе. “әли, ауа райын білгім келіп тұр”. Распознав 2 ключевых словосочетания(әли, ауа-райы) он дает ответ: “қазір далада (weather)”. Так же с другими командами, бот сперва обрабатывает слова и решает отвечать или нет, если есть триггер, но если команды с ключевых слов не найдено он отвечает, что команда не найдена.

**Введение**

В данном введении мы рассмотрим основные цели и задачи проекта "Голосовой ассистент на казахском QazVoice", а также выявим его значимость для облегчения повседневной жизни казахскоязычных пользователей и развития технологической инфраструктуры страны.

**Актуальность проекта:** На сегодняшний день мы можем встретить множество русскоговорящих голосовых ассистентов. Таких как: Алиса, Siri, Google Assistant и так далее. Но на просторах казахстанского интернета практически нет казахоговорящих голосовых ассистентов. Данный проект решает эту проблему.

**Цель:** Подцелью проекта является создание компьютерного приложения в сфере где имеется лингвистический баррьер между технологиями и казахским языком. Главная цель проекта это развитие казахского языка в цифровом мире.

**Этапы проекта:**

1. **Подготовительный этап:**
   * Формирование команды: создать команду для проекта и определиться с дальнейшими действиями
   * Исследовательские работы: поиски и изучение нужных материалов для создания проекта
   * Изучение новых языков, таких как Java и Python
2. **Разработка кода и базовой функциональности приложения:**
   * Голосовое распознавание: Внедрение библиотек для обучения модели распознавания казахской речи.
   * Базовые функции: Реализация основных функций, таких как поиск сайтов в сети, поиск информации, доступ к открыванию приложении
3. **Тестирование и отладка:**
   * Тестирование функционала: Выполнение тестов, обнаружение и устранение затруднении.
   * Определение производительности: Оценка скорости и эффективности работы ассистента. Оптимизация кода и улучшение скорости.

**Гипотеза**: команда QazVoice расчитывает на то, что наше приложение поможет решить технологическую проблему свяазанную с ограничением языка и поспособствует развитию казахского языка. А так же будет полезен в использовании, в улучшении кахаского языка пользователя и другие поставленные задачи.

**Исследовательская часть**

По ходу создания проекта, мы решили расспросить преподавателя казахского языка и куратора в школе НИШ ФМН г. Талдыкорган про актуальность данной проблемы и как на их взгляд можно решить эту проблему. Так же они оценят наше решение данной проблемы.

Мы обратились к ним с просьбой ответить на наши вопросы. В первую очередь мы их ознакомили с главной идеей и целью нашего проекта, а потом с результатом который мы получили(с приложением). Они ответили на все наши вопросы и дали обратную связь на наш проект.

Алихан берет интервью и демонстрирует приложение преподавателю казахского языка Қалықбекове Бұлбұл Тілеуғазиевне:



Есет берет интервью и проводит демонстрацию проекта

у куратора 6, 9, 10 классов Маржан Тілеуқабылқызы:





**Сұрақ№1:** Сіздің ойыңызша, қазіргі таңда қазақ тілі технологиялық даму жағынан артыда қалып қалды деген тұжырым қаншалықты дұрыс?

**Қазақ тілі пән мұғалімі**: Негізі бұл мәселеге екі жақты қарауға болады. Кейбір қазақ тіліндегі интернет ресурстар жеткілікті ақырындап дамып келе жатыр. Қазақша контент пен түрлі жобалар дамып келе жатыр. Дегенмен, жетілдіретін тұстарымыз көп.

**Куратор**: мен бұл тұжырыммен толықтай келісемін. Қазақ тілі біздің мемлекеттік тіл болса да, қазіргі таңда 100 болмаса да, тіліміз 50-60 пайызға басқа тілдерден дамуы төмен. Себебі бәрі орыс тілін қолданып, орыс тілінде ойлайды.

**Сұрақ№2:** Сіздің ойыңызша, бұл мәселені қалай шешуге болады?

**Қазақ тілі мұғалімі**: Қазір соңғы кезде IT технологиялар кеңінен дамып жатыр. Әсіресе, IT саласына қызығушылық жастар арасында көп. Сондықтан, осы технологиялар мүмкіндіктерін пайдаланып, жасанды интеллект мен подкасттар арқылы жас аудиторияға көп көмектесуге болады Және де қазіргі таңда нәтижелі стартап болса, мемлекет тарапынан қаржыландырады. Болмаса демеушілер тауып алуға болады.

**Куратор**: бұл мәселені шешу үшін әр адам бірінші кезекте өзімен жұмыс жасауы қажет. Мысалы, қарапайым цифрлік банкоматтың алдына барғанның өзінде орыс тілін таңдайтын адамдар саны көп. Біз өзіміздің таңдауымыздың бәрін қазақша жасаса, мүмкін қазақ тілінің мәртебесі көтеріледі.

**Сұрақ№3:** Біздің топпен жасалған QazVoice атты жүктеме арқылы қазақ тілін қаншалықты дамытуға болады деп ойлайсыз?

**Қазақ тілі мұғалімі**: Сіздердің жүктемелерінізбен сіздердің айтуларын бойынша біраз таңыстым. Көптеген функциялар бар екен, десекте жетілуді қажет етеді. Ең бірінші мені қуантатыны - бұрың сонды болмаған осындай жобалардың серпім алып жатқаны. Өйткені біздің әріп білмейтін жас балалар батысқа еліктеп, Сири мен Алиса арқылы сұрақ қойып жатады. Енді қазақ тілінде сондай сұрақтар қойып, жауап алып жатса Алиса сияқты QazVoice тағы жігіттерініз біраз көмек береді деп ойлаймын.

**Куратор**: Сіздердің жобаларыңыз көмегімен біз қазақ тілді жақсы деңгейге дейін дамыта аламыз.

**Сұрақ№4:** Алдымызға қойылған мақсатқа біз қаншалықты жақындадық деп ойлайсыз?

**Қазақ тілі мұғалімі**: Мақсаттарыныз айқын. Ең бірінші бір қуанталығы - сіздер бұл жобаны қолға алдыңыздар. Яғни қазақта “сен қозғал” дегендей жұмыс басталды. Басталған затты ары қарай жетілдіруге болады, мақсатқа жетуге болады. Өздерініз ізденіп тиімді жақтарын көріп қызықты қылып жасасаныздар ол мақсатқа әрі қарай қол жеткізуге болады

**Куратор**: Сіздердің жобаларыңыз күн көргеннен кейін алдыларыңызға қойылған мақсаттарыңызға 10 қадам болса да жақындадыңыздар. Енді тек қана керекті адамдарға, халыққа бұл жобаны жариялап, әрі қарай дамыту қажет.

**Сұрақ№5:** Бізге осы мақсатқа қол жеткізіп, әрі қарай даму үшін қандай ұсыныстар бере аласыз?

**Қазақ тілі мұғалімі**: Әрине уақыт керек, біраз қоғаммен жұмыс жасап, қоғамға таныстыру керек, қызығушылықтарын ояту керек. Сондықтан мақсатқа жетуге болады.

**Куратор**: Біріншіден өздеріңізді жарнамалап, халық алдына жобаларыңызды жақсылап түсіндіріп, жарияласаңыздар ел арасында бұл жүктеме таралуы әбден мүмкін.

**Функциональные возможности ассистента QazVoice**

На данном этапе развития проекта наш голосовой ассистент может выполнять базовые функции. Такие как:

1. Браузер: Выполнение задач связанных с открытием и использованием браузера.
2. Кино: Открытие сайтов связанных с просмотром фильмов. Такие как киногоу
3. Игры: Выполняет запросы, связанные с открытием конкретных приложении. Таких как: Steam, Telegram, WhatsApp и так далее.
4. Отключение бота: Бот выключает самого себя при исполнении этой программы
5. Пассивные ответы: Примеры пассивных ответов ассистента на приветствия и вопросы о его состоянии.
6. Погода: Может рассказать какая погода сейчас на улице.

Приложение запускается около 10-12 секунд и выдает фразу “сәлем”, “қалың қалай?”. Таким образом бот дает знать пользователю, что он готов к работе.

Скорость обработки запроса и выдачи ответа в среднем котируется между 0.9-1.0 секунд. Мы засекали данный показатель благодаря библиотеке time внтури самого языка. А точнее, мы использовали “секундомер” который включался в начале обработки слов и выключался после завершения обработки и вывода ответа. При выводе мы получали нужный ответ и под ними количество секунд, за сколько работает обработка команды.

Самой медленной функцией оказалась погода, ведь для использования данной функции мы получали команду, преобразовавали ее в двоичный код, получали ответ с сайта 'https://api.openweathermap.org/data/2.5/weather'

на английском языке и переводили его на казахский с помощью Google Translator и преобразовали числа в текстовый формат. А дальше уже обработанную информацию выдавали как ответ. Поэтому скорость выполнения именно данной функции имеет медленную скорость.

**Этапы создания QazVoice и дневник**

|  |  |
| --- | --- |
| Май 2023 год | Объединились в команду и сразу же решили что приступим изучению языков Java и Python. Цель проекта не определена, все еще обсуждается в команде. |
| Июнь 2023 год | Обучение идет активно на платформе Stepik.Определились с целью проекта. |
| Июль 2023 год | Решили на каком языке будем писать програмный код. Обсуждаем видение нашего проекта, гепотезу. Ищем материал и ресурсы которые можно использовать в будущем |
| Август 2023 год | Приступили к написанию самого кода и использовали материал который нашли ранее. Сталкиваемся со множеством проблем и просим помощи на таких платформах как Habr Q&A, берем ресурсы с GitHub-а и т.д. |
| Сентябрь 2023 год | Почти закончили проект. Столкнулись с проблемой оптимизации. Программа запускалась долго и выполняла задачи медленнее ожидаемых показателей. Решили эту проблему и провели тестовые вхождения и удоволетворились результатом |
| Октябрь 2023 год | Приступили к документации, создании презентации и подготовке выступления перед жюри. Распечатали все что нужно и готовы к выступлению. |

**Проблемы с которыми столкнулись по ходу создания проекта:**

**1.**

* **Не поддерживаемость казахского языка**. Многие инструменты, библиотеки не поддерживают казахский язык, что ограничивало возможности нашего проекта. Эта проблема была одна из самых весомых, ведь почти все библиотеки подготовлены для англоязычного пользователя и в меньшей степени для русскоязычного пользователя.
* **Решение:** с помощью инструмента translate переводили информацию с русского языка на казахский и использовали эту информацию
* **Например:** Для получения данные о погоде мы использовали сайт openweathermap API .Он вернул ответ на английском языке. Поэтому пришлось перевести текст на казахский и вывести его.

**2.**

* **Выбор устройства для синтеза речи между Silero и Issai:**

Обе библиотеки имеют свои достоинства и свои минусы. Синтез речи Issai сделан учениками нашего соотечественного универистета имени Нурсултана Назарбаева(Nazarbayev University). А Silero это исскуственный интеллект который поддерживает систему tts(text-to-specch) stt(speech-to-text).

Наш выбор все таки остановился на Silero, ведь его внедренее в программу не составило труда, а разобраться как работает Issai мы так и не смогли

**3.**

* **Не поддерживаемость чисел**: устройство Silero не синтезирует числа и любые цифры.(‘123’ , ‘25’ и другие)
* **Решение:** с помощью устройства num2words мы преобразовали числа в текстовый формат.
* **Например:** В случаи вывода такого текста как– «Далада 28 градус» : выводит «Далада градус». Поэтому этот текст превращается в : «Далада жиырма сегіз градус».

**4.**

* **Низкая точность распознавания:** бот имел слабую способность распозновать речь.
* **Решение:** Мы использовали машинное обучение чтобы выбрать наиболее подходящую команду на распознанный текст для улучшения качество распознавания.

**5.**

* **Низкая скорость распозанавания:** бот работал очень медленно. Около 10с на обработку лишь 1-го запроса. Эти показатели нас не устроили
* **Решение:** мы ввели в начале программы модуль силеро. Благодаря тому, что он выводит в начале 2 начальные фразы ‘салем’, ‘қалың қалай?’ бот полностью включается и уже работает намного быстрее.
* **Например:** начальный вывод в начале ‘салем’, ‘қалың қалай?’ работает 10-12 секунд и после этого все последующие команды работают за 0.9-1.0 секунд.

6.

* **Включение случайных функции:** когда бот обрабатывал полученные данные и не получал нужных ключевых слов, он включал случайные функции и выполнял задачу, которую перед ним не поставили
* **Решение:** Мы указываем порог совпадения и сравниваем наиболее подходящую функцию которая не превышает порог совпадения
* **Например:** Бот получает информацию на обработку и если эта информация не совпадает с ключевыми словами, то бот выдает, что команда не найдена.

7.

* **Неправильное распознавание триггеров**: при обращении к голосовому ассистенту мы столкнулись с проблемой не распознования триггера. То есть, когда мы обращались к боту по имени али он распознавал это как алибек, әли или ғали.
* **Решение**: мы решили добавить эти же слова в словарь, чтобы когда бот не правильно распознавал триггер это не мешало ему работать.

8.

* **Отсутствие качество интерфейса:** изначально мы хотели сделать приложение с приятным интерфейсом. Мы пришли к этапу подготовки интефейса слишком поздно и не успели сделать достойный вариант. Главной проблемой было то, что у нас не хватило знании и опыта для создания удобного и приятного интерфейса.
* **Решение:** Мы сделали не идеальный вариант графического интерфейса и в будущем будем его улучшать.

**Примеры вывода до добавления интерфейса:**

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

**Пример вывода после добавления графического интерфейса:**

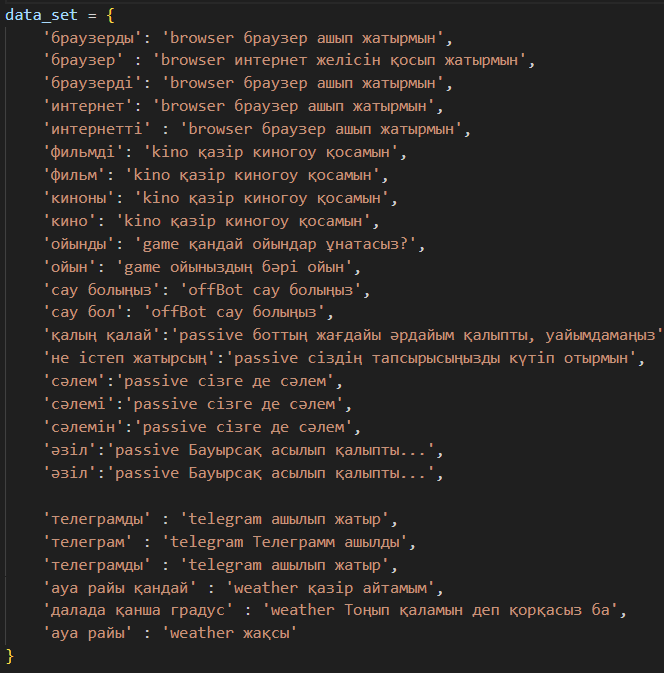
Изображение выглядит как текст, программное обеспечение, снимок экрана, Мультимедийное программное обеспечение

Автоматически созданное описание

**Коды и объяснение их роли в програмном коде:**

**Words.py –**файл с данными. Модель определяет в этих данных команду, наиболее похожую на ключеовое словочетание, произнесенную пользователем и выполняет соответствующую команду.

Слова для обращения к ассистенту**,** триггеры. Бот начинает работу когда слышит эти триггеры, пока триггер не сказан бот работать не будет.****

****

Переменная **data\_set** представляет собой словарь, где ключами являются различные фразы или запросы, которые пользователь может отправить голосовому ассистенту, а значениями являются ответы или действия, которые ассистент должен выполнить в ответ на эти запросы. Этот словарь служит внутренней базой данных для ассистента и используется для определения того, какой ответ предоставить пользователю в зависимости от входящего запроса.

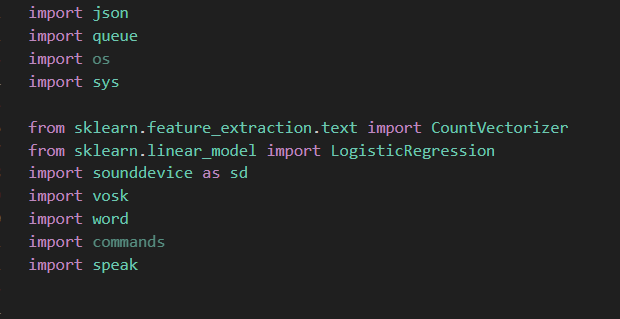
В данном случае, **data\_set** содержит примеры запросов и соответствующие ответы для нескольких категорий запросов:

1. Браузер: Примеры запросов, связанных с открытием и использованием браузера.
2. Кино: Примеры запросов, связанных с просмотром фильмов.
3. Ойын: Примеры запросов, связанных с играми.
4. Отключение бота: Примеры запросов, связанных с завершением работы с ассистентом.
5. Пассивные ответы: Примеры пассивных ответов ассистента на приветствия и вопросы о его состоянии.
6. Телеграм: Примеры запросов, связанных с использованием мессенджера Telegram.
7. Ауа-райы: Примеры запросов, связанных с получением информации о погоде.

Когда пользователь отправляет запрос голосовому ассистенту, программа проверяет этот запрос на совпадение с ключами в **data\_set**. Если запрос соответствует ключу, ассистент выбирает соответствующее значение (ответ или действие) из словаря **data\_set** и выполняет соответствующее действие или возвращает соответствующий ответ пользователю. Это позволяет ассистенту реагировать на различные запросы и обеспечивать интерактивное взаимодействие с пользователем.

**2: main.py –** это программа, отвечающая за обнаружение произнесенного слова или команды и ее анализ.

Импортирование нужных библиотек в программу.



Построение моделей Model, Queue, device , samplerate .

q = queue.Queue()  #Создаем очередь

model = vosk.Model('vosk-model-kz')      #Создание модели

try:

    default\_device = sd.default.device  #Определяем устройство ввода звука по умолчанию

    samplerate = int(sd.query\_devices(default\_device[0], 'input')['default\_samplerate'])  #получаем частоту микрофона

except:

    speak.speaker('Микрофонды қос')  #при ошибке завершаем программу

    sys.exit(1)

Приведенный код выполняет следующие действия:

1. В функции **main()** выводится сообщение "Слушаем", что сигнализирует о начале работы ассистента.
2. Далее идет обучение модели машинного обучения. Обучение происходит на основе данных, представленных в **data\_set**, где ключи - это фразы или запросы, а значения - это соответствующие ответы или действия ассистента.
   * **CountVectorizer()** используется для создания матрицы слов на основе ключей (фраз) из **data\_set**. Этот процесс преобразует текстовые данные в числовой формат, который может быть использован для обучения машинной модели.
   * **LogisticRegression()** - это модель машинного обучения, используемая для классификации текстовых данных. Модель обучается на матрице, созданной с помощью **CountVectorizer**, и соответствующими значениями из **data\_set**. Это позволяет ассистенту классифицировать входящие запросы и выбирать соответствующий ответ.
3. Далее следует блок кода, который обеспечивает прослушивание микрофона и распознавание речи с использованием библиотеки Vosk.
   * **sd.RawInputStream** настраивает прослушивание микрофона с заданными параметрами, такими как частота дискретизации (**samplerate**), размер блока, устройство микрофона и другие параметры.
   * **rec = vosk.KaldiRecognizer(model, samplerate)** создает объект распознавания речи на основе модели (**model**) и частоты дискретизации (**samplerate**), используя библиотеку Vosk.
4. В бесконечном цикле **while True**, программа ожидает входящие аудиоданные из микрофона (**data = q.get()**), а затем передает эти данные в распознаватель Vosk.
5. Если распознавание успешно (**rec.AcceptWaveform(data)**), программа получает текстовый результат распознавания (**rec.Result()**) и извлекает текст из этого результата (**json.loads(rec.Result())['text']**).
6. Полученный текст передается функции **recognize(data, vectorizer, clf)**. Вероятно, эта функция обрабатывает распознанный текст и возвращает соответствующий ответ с помощью обученной модели и данных из **data\_set**.
7. Наконец, текст распознавания выводится на экран (**print(data)**), что позволяет отслеживать процесс распознавания и обработки запросов пользователей.

Этот код представляет собой основную структуру голосового ассистента, который прослушивает аудиоданные с микрофона, распознает речь, классифицирует запросы и предоставляет соответствующие ответы на эти запросы

def main():

    print('Слушаем')

    # Обучение матрицы на data\_set модели

    vectorizer = CountVectorizer()

    vectors = vectorizer.fit\_transform(list(word.data\_set.keys()))

    clf = LogisticRegression()

    clf.fit(vectors, list(word.data\_set.values()))

    # постоянная прослушка микрофона

    with sd.RawInputStream(samplerate=samplerate, blocksize = 16000, device=default\_device[0], dtype='int16',

                            channels=1, callback=callback):

        rec = vosk.KaldiRecognizer(model, samplerate)

        while True:

            data = q.get()

            if rec.AcceptWaveform(data):

                data = json.loads(rec.Result())['text']

                recognize(data, vectorizer, clf)

                print(data)

Код **recognize(data, vectorizer, clf)** выполняет следующие действия:

1. Проверяет длину входных данных (распознанного текста). Если длина меньше 7 символов, функция завершает выполнение, так как текст слишком короткий для обработки.
2. Проверяет наличие фразы обращения к ассистенту в тексте. Если в распознанном тексте отсутствуют фразы-триггеры (определенные в **word.TRIGGERS**), то функция также завершает выполнение.
3. Удаляет из распознанного текста фразу с именем ассистента. Это делается путем разделения текста на слова и удаления слов, содержащихся в фразах-триггерах.
4. Преобразует очищенный текст в числовой вектор с использованием **vectorizer.transform()**, чтобы подготовить его для классификации.
5. Предсказывает вероятности принадлежности команды к каждому классу (возможным ответам) с помощью обученной модели (**clf**).
6. Устанавливает порог вероятности (**probability**). Если максимальная вероятность предсказания превышает этот порог, то функция выбирает наиболее подходящий ответ и продолжает выполнение. В противном случае, функция сообщает, что команда не была распознана.
7. Получает имя функции из ответа в **data\_set**, разделяя ответ на слова и выбирая первое слово (которое представляет собой имя функции).
8. Воспроизводит озвученный ответ на основе ответа из **data\_set**.
9. Запускает функцию из модуля **commands** с именем, полученным из ответа. Это предполагает, что в модуле **commands** есть соответствующая функция для выполнения команды.

Таким образом, функция **recognize(data, vectorizer, clf)** выполняет обработку распознанного текста, классифицирует команду, выбирает наиболее подходящий ответ и выполняет соответствующую функцию из **commands**.

def recognize(data, vectorizer, clf):

    if len(data) < 7:   #пропускаем слова если его длина меньше 7

        return

    #если нет фразы обращения к ассистенту, то завершаем код

    trigger = 0

    for w in data.split():

        for t in word.TRIGGERS:

            if t in w or w in t:

                trigger = 1

                break

        if trigger == 1:

            break

    if trigger == 0:

        return

    data = data.split()

    clear\_data = [w for w in data for t in word.TRIGGERS if t not in w or w not in t]   #удаляем из команды имя асистента

    data = ' '.join(clear\_data)

    #получаем вектор полученного текста

    #сравниваем с вариантами, получая наиболее подходящий ответ

    # Преобразование команды пользователя в числовой вектор

    command\_vector = vectorizer.transform([data])

    # Предсказание вероятностей принадлежности к каждому классу

    predicted\_probabilities = clf.predict\_proba(command\_vector)

    # Задание порога совпадения

    probability = 0.2

    # Поиск наибольшей вероятности и выбор ответа, если он превышает порог

    max\_probability = max(predicted\_probabilities[0])

    print(max\_probability)

    if max\_probability >= probability:

        answer = clf.classes\_[predicted\_probabilities[0].argmax()]

    else:

        speak.speaker("Команда табылмады")

        return

    #получение имени функции из ответа из data\_set

    func\_name = answer.split()[0]

    #озвучка ответа из модели data\_set

    speak.speaker(answer.replace(func\_name, ''))

    #запуск функции из commands

    exec("commands." + func\_name + '()')

**3: speak.py – ақпаратты синтездеу** программасы**. Команда орындалуда айтылатын ақпаратты алып, синтездейді.**

1. Этот код позволяет использовать библиотеку Silero TTS для синтеза речи на основе заданного текста и параметров. Мы узнали про этот и другие модели синтеза речи на казахском языке отсюда [[**https://qna.habr.com/q/1305954**](https://qna.habr.com/q/1305954)] Давайте рассмотрим его по шагам:
2. **lang** и **model\_id** определяют язык и идентификатор модели для синтеза речи. В данном случае, установлен язык "cyrillic" (кириллица) и идентификатор модели "v4\_cyrillic".
3. **device** задает устройство для выполнения операций с моделью. В данном случае, выбрано CPU.
4. Загрузка модели Silero TTS с помощью **torch.hub.load()**. Модель загружается из репозитория **snakers4/silero-models** и устанавливается для заданного языка и спикера.
5. **def speaker(text)** – это функция, которая
   * будет принимать текст и синтезировать речь на основе этого текста с помощью загруженной модели.
6. Внутри функции происходит следующее:
   * Задается частота дискретизации (**sample\_rate**) для аудио, равная 48,000 Гц.
   * Выбирается спикер с именем "kz\_M2".
   * Устанавливаются настройки речи, такие как добавление акцента и обработка буквы "ё".
   * Вызывается метод **apply\_tts** модели для синтеза аудио на основе переданного текста и параметров.
   * Аудио воспроизводится с использованием библиотеки **sounddevice**, а затем функция ждет, пока аудио завершит воспроизведение, используя **time.sleep**.

Таким образом, эта функция **speaker(text)** может быть использована для синтеза и воспроизведения речи на основе текста, переданного ей как аргумент.

import torch

import sounddevice as sd

import time

lang = "cyrillic"      #определение языка и модели

model\_id = 'v4\_cyrillic'

device = torch.device('cpu')  #использование ram памяти

model , \_ = torch.hub.load(repo\_or\_dir='snakers4/silero-models',   #получение модели с помощью библиотеки torch

                           model = 'silero\_tts',

                           language=lang,

                           speaker=model\_id)

model.to(device)

def speaker(text):

    sample\_rate = 48000   #определение дискретизации

    speaker = "kz\_M2"  #Выбор спикера

    put\_accent = True  #Настройка речи

    put\_yo = True

    audio = model.apply\_tts(text=text,         #Создание аудио

                            speaker=speaker,

                            sample\_rate=sample\_rate,

                            put\_accent=put\_accent,

                            put\_yo=put\_yo)

    print(text)

    sd.play(audio , sample\_rate)  #проигрыватель аудио

    time.sleep(len(audio) / sample\_rate)

    sd.stop()

**4: command.py –** командалар жинағы. Data\_set – тан алынған ақпарат арқылы сәйкес функция қосылады.

Импорт нужных библиотек, инструментов.

import os

import webbrowser

import sys

import subprocess

import speak

import requests

from ctypes import cast, POINTER

from comtypes import CLSCTX\_ALL

from pycaw.pycaw import AudioUtilities, IAudioEndpointVolume

import translator

**browser()**: Эта функция открывает браузер и переходит на веб-сайт Google, используя модуль **webbrowser**. Это позволяет открыть браузер и перейти на указанный веб-сайт.

#Функция для открытие браузера

def browser():

    webbrowser.open('https://www.google.com', new=2)

**kino()**: Эта функция открывает браузер и переходит на веб-сайт kinogo.biz, опять же, используя модуль **webbrowser**. Функция предназначена для открытия веб-сайта с фильмами.

#Функция для открытие кино сайта

def kino():

    webbrowser.open('https://kinogo.biz/' , new=2)

**weather()**: Эта функция предназначена для получения данных о погоде с использованием API OpenWeatherMap. Функция отправляет запрос к API OpenWeatherMap, указывая город (Taldykorgan) и несколько параметров (единицы измерения, язык, и API ключ). Ответ API анализируется, и затем озвучивается информация о текущей погоде (описание и температура) с помощью функции **speak.speaker()**.

#функция для получения данных о погоде с API openweathermap

def weather():

    '''Для работы этого кода нужно зарегистрироваться на сайте

    https://openweathermap.org или переделать на ваше усмотрение под что-то другое'''

    try:

        params = {'q': 'Taldykorgan', 'units': 'metric',

                  'lang': 'en', 'appid': "f6f1874a6149418109550839c5ea245e"}

        response = requests.get(

            f'https://api.openweathermap.org/data/2.5/weather', params=params)

        if not response:

            raise

        w = response.json()

        speak.speaker(

            translator.translate(f"Weather is {w['weather'][0]['description']} {round(w['main']['temp'])} degrees"))

    except:

        speak.speaker(

            "Қате!")

**game()**: Эта функция предназначена для запуска игры. Она использует модуль **subprocess** для выполнения указанного файла, который, вероятно, является исполняемым файлом игры на компьютере пользователя. В случае возникновения ошибки (например, неверно указан путь к файлу), функция озвучивает сообщение об ошибке.

#Функция для подключение игры

def game():

    try:

        subprocess.Popen(r"C:\Program Files (x86)\Steam\steam.exe")

    except:

        #print('Файл жолы табылмады, оның дұрыстығын тексеріңіз')

        speak.speaker('Файл жолы табылмады, оның дұрыстығын тексеріңіз')

**telegram()**: Эта функция предназначена для запуска приложения Telegram на компьютере пользователя. Она использует **os.system** для выполнения команды, которая запускает Telegram. Путь к исполняемому файлу Telegram указан в команде.

#Функция для подключение телеграмма

def telegram():

     os.system('"C:\Program Files\WindowsApps\TelegramMessengerLLP.TelegramDesktop\_4.9.7.0\_x64\_\_t4vj0pshhgkwm\Telegram.exe"')

**offBot()**: Эта функция просто завершает выполнение программы с помощью **sys.exit()**. Таким образом, она используется для отключения голосового ассистента.

def offBot():

    '''Отключает бота'''

    sys.exit()

**passive()**: Эта функция представляет собой пустую заглушку, которая ничего не делает. Она используется в тех случаях, когда не требуется выполнение конкретных действий, и служит для простого ответа при диалоге с ассистентом, например, на приветствия пользователя.

def passive():

    '''Функция заглушка при простом диалоге с ботом'''

    pass

**5: num2words.py**

Бұл код мәтіндегі барлық сандарды олардың мәтіндік көрінісіне ауыстыратын replace\_numbers\_with\_words (text) функциясын анықтайды.

1. **def replace\_numbers\_with\_words(text)**: Это объявление функции, которая принимает один аргумент **text**, который представляет собой текст, в котором нужно выполнить замену чисел.
2. **def replace(match)**: Внутри функции определена внутренняя функция **replace(match)**, которая используется для замены каждого числа в тексте на его текстовое представление. Эта функция принимает объект **match**, который представляет собой совпавшее числовое значение, найденное регулярным выражением.
3. **result = re.sub(r'\d+', replace, text)**: Здесь используется функция **re.sub()**, которая выполняет замену совпавших подстрок в тексте на результат выполнения функции **replace(match)**. Регулярное выражение **r'\d+'** используется для поиска всех последовательных цифр в тексте.
   * **match.group(0)** извлекает совпавшее числовое значение из объекта **match**.
   * **num2words(num, lang='en')** преобразует целое число **num** в его текстовое представление на английском языке с помощью библиотеки **num2words**. Результат заменяет найденное число в тексте.
4. **return result**: Функция возвращает результат замены, то есть текст, в котором числа заменены на их текстовые представления.

import re

from num2words import num2words

def replace\_numbers\_with\_words(text):

    def replace(match):

        num = int(match.group(0))

        return num2words(num, lang='en')

    # Ищем все числа в строке и заменяем их на числительные

    result = re.sub(r'\d+', replace, text)

    return result

**6. translate.py**

Этот код использует библиотеку Google Translate (**googletrans**) и функцию **replace\_numbers\_with\_words()** для перевода текста на казахский язык с преобразованием чисел в их текстовое представление. Давайте рассмотрим функцию **translate(text)**:

from googletrans import Translator

import num2word

translator = Translator()

#перевод текста на казахский

def translate(text):

    return translator.translate(num2word.replace\_numbers\_with\_words(text)  ,'kk').text

Ниже приведена тестовая таблица нашей программы. Мы проверили каждую возможную команду и ниже показаны полученные данные.

**7. gui.py**

Это файл с кодом, который создает графический интерфейс для нашего консольного приложения. Мы использовали готовый шаблон и поменяли фрагменты шаблона на нужные нам фрагменты. Репазиторий на этот код:

[https://github.com/TomSchimansky/CustomTkinter/tree/master]

import customtkinter

import os

from PIL import Image

import webbrowser

class App(customtkinter.CTk):

    str\_text = ""

    def \_\_init\_\_(self):

        super().\_\_init\_\_()

        self.title("QazVoice")

        self.geometry("700x410")

        # set grid layout 1x2

        self.grid\_rowconfigure(0, weight=1)

        self.grid\_columnconfigure(1, weight=1)

        # load images with light and dark mode image

        image\_path = os.path.join(os.path.dirname(os.path.realpath(\_\_file\_\_)), "test\_images")

        self.logo\_image = customtkinter.CTkImage(Image.open(os.path.join(image\_path, "logo.png")), size=(50, 50))

        self.large\_test\_image = customtkinter.CTkImage(Image.open(os.path.join(image\_path, "large.png")), size=(700, 180))

        self.micro\_image = customtkinter.CTkImage(Image.open(os.path.join(image\_path, "micro\_light.png")), size=(70, 70))

        self.home\_image = customtkinter.CTkImage(Image.open(os.path.join(image\_path, "home\_light.png")), size=(20, 20))

        self.info\_image = customtkinter.CTkImage(Image.open(os.path.join(image\_path, "info.png")), size=(70, 70))

        self.setting\_image = customtkinter.CTkImage(Image.open(os.path.join(image\_path, "settings.png")), size=(70, 70))

        # create navigation frame

        self.navigation\_frame = customtkinter.CTkFrame(self, corner\_radius=0)

        self.navigation\_frame.grid(row=0, column=0, sticky="nsew")

        self.navigation\_frame.grid\_rowconfigure(4, weight=1)

        self.navigation\_frame\_label = customtkinter.CTkLabel(self.navigation\_frame, text="  QazVoice", image=self.logo\_image,

                                                            compound="left", font=customtkinter.CTkFont(size=15, weight="bold"))

        self.navigation\_frame\_label.grid(row=0, column=0, padx=60, pady=60)

        self.home\_button = customtkinter.CTkButton(self.navigation\_frame, corner\_radius=0, height=40, border\_spacing=10, text="Home",

                                                    fg\_color="transparent", text\_color=("gray10", "gray90"), hover\_color=("gray70", "gray30"),

                                                    image=self.home\_image, anchor="w", command=self.home\_button\_event)

        self.home\_button.grid(row=1, column=0, sticky="ew")

        # create home frame

        self.home\_frame = customtkinter.CTkFrame(self, corner\_radius=0, fg\_color="transparent")

        self.home\_frame.grid\_columnconfigure(0, weight=1)

        self.home\_frame\_large\_image\_label = customtkinter.CTkLabel(self.home\_frame, text="QazVoice Assistant", font=("Montserrat" , 20), image=self.large\_test\_image)

        self.home\_frame\_large\_image\_label.grid(row=0, column=0, padx=20, pady=10)

        self.home\_button\_2 = customtkinter.CTkButton(self.home\_frame,text="" , image=self.info\_image, command=self.info\_event)

        self.home\_button\_2.grid(row=1, column=0, padx=20, pady=10)

        self.home\_button\_3 = customtkinter.CTkButton(self.home\_frame, text="", image=self.setting\_image , command=self.settings\_event)

        self.home\_button\_3.grid(row=2, column=0, padx=20, pady=10)

        self.select\_frame\_by\_name("home")

    def select\_frame\_by\_name(self, name):

        # set button color for selected button

        self.home\_button.configure(fg\_color=("gray75", "gray25") if name == "home" else "transparent")

        if name == "home":

            self.home\_frame.grid(row=0, column=1, sticky="nsew")

        else:

            self.home\_frame.grid\_forget()

    def home\_button\_event(self):

        self.select\_frame\_by\_name("home")

    def info\_event(self):

        webbrowser.open("https://www.canva.com/design/DAFwQ0b2BCc/j6HHe1nLrOIDpcTp7Cdw7A/edit" , new=2)

    def settings\_event(self):

        os.system("config.py")

    def change\_appearance\_mode\_event(self, new\_appearance\_mode):

        customtkinter.set\_appearance\_mode(new\_appearance\_mode)

|  |  |  |
| --- | --- | --- |
| Входные данные | Ожидаемый вывод | Получаемый вывод |
| Фильм қарағым келіп тұр | \*kinogo.com сайты қосылады\* Қазір киногоу қосамын | \*kinogo.com сайты қосылады\* Қазір киногоу қосамын |
| Кино көремін | \*kinogo.com сайты қосылады\* Қазір киногоу қосамын | \*kinogo.com сайты қосылады\* Қазір киногоу қосамын |
| Ойынды қосшы | \*ойындар қосылады\*  Қандай ойындар ұнатасыз? | \*ойындар қосылады\*  Қандай ойындар ұнатасыз? |
| Ойын ойнағым келіп тұр | \*ойындар қосылады\*  Ойыныздың бәрі ойын | \*ойындар қосылады\*  Ойыныздың бәрі ойын |
| Сау бол | \*Бот өшеді\*  Сау болыңыз | \*Бот өшеді\*  Сау болыңыз |
| Қалың қалай? | Боттың жағдайы әрдайым қалыпты, уайымдамаңыз | Боттың жағдайы әрдайым қалыпты, уайымдамаңыз |
| Не істеп жатырсың? | Сіздің тапсырыңызды күтіп отырмын | Сіздің тапсырыңызды күтіп отырмын |
| Сәлем | Сізге де сәлем | Сізге де сәлем |
| Әзіл айтып берші | Бауырсақ асылып қалыпты | Бауырсақ асылып қалыпты |
| Телеграмды аш | \*Telegram ашылады\*  Ашылып жатыр | \*Telegram ашылады\*  Ашылып жатыр |
| Телеграм | \*Telegram ашылады\*  Телеграм ашылды | \*Telegram ашылады\*  Телеграм ашылды |
| Ауа райы қандай? | \*Ауа райы айтылады\*  Қазір айтамын | \*Ауа райы айтылады\*  Қазір айтамын |
| Далада қанша градус? | \*Ауа райы айтылады\*  Тоңып қаламын деп қорқасыз ба? | \*Ауа райы айтылады\*  Тоңып қаламын деп қорқасыз ба? |
| Ауа райы | \*Ауа райы айтылады\*  Жақсы | \*Ауа райы айтылады\*  Жақсы |
| Браузерді қос | \*Браузер қосылады\* Браузер ашып жатырмын | \*Браузер қосылады\* Браузер ашып жатырмын |
| Браузер | \*Браузер қосылады\* Интернет желісін қосып жатырмын | \*Браузер қосылады\* Интернет желісін қосып жатырмын |
| Интернет желісін қос | \*Браузер қосылады\* Браузер ашып жатырмын | \*Браузер қосылады\* Браузер ашып жатырмын |

**Заключение**

Голосовой ассистент на казахском языке, разработанный в рамках данного проекта, представляет собой важный момент в области голосовых технологий и искусственного интеллекта на просторах каз.интернета. Он обеспечивает пользователей удобством и функциональностью взаимодействия с компьютерными системами с помощью голосовых команд и запросов.

Основная цель этого проекта заключалась в создании голосового ассистента, способного выполнять разнообразные задачи и функции на **казахском языке**. В ходе разработки были рассмотрены и реализованы множество функциональных возможностей, включая поиск информации в интернете, получение данных о погоде, навигацию и многое другое.

Важным компонентом проекта стало использование технологий машинного обучения и обработки речи. Голосовой ассистент спроектирован для распознавания голосовых команд и генерации голосовых ответов, что делает его более доступным и удобным для пользователей.

Интеграция с внешними сервисами и API, такими как OpenWeatherMap, позволяет ассистенту предоставлять актуальную информацию о погоде. А использование Silero TTS (Text-to-Speech) для синтеза голосовых ответов добавляет натуральности и гармоничности взаимодействия с пользователем.

Этот голосовой ассистент обладает огромным потенциалом в различных сферах, таких как информационные поиски, навигация, развлечения, здоровье, образование, электронная коммерция и другие. Он может стать незаменимым помощником для пользователей, облегчая их повседневные задачи и предоставляя важную информацию.

Важно отметить, что разработанный голосовой ассистент не ограничивается имеющимися функциями, и его потенциал будет дополнен и расширен в будущем. С учетом развития технологий и запросов пользователей этот проект способен динамично адаптироваться и стать более мощным инструментом.

**Возможный перевод Алисы на казахский язык:**

Когда мы уже поставили перед собой цели, задачи и дошли до ¼ финиша проекта, стало известно, что Яндекс ассистент Алиса будет говорить на казахском языке. Эта новость вышла в сеть 6 го сентября и примерные сроки объявляются как конец года. Это был непредвиденный форс-мажор, но мы не стали менять идею проекта. Мы лишь попытались сделать лучший вариант казахского ассистента за данные нам сроки и наши ожидания превратились в реальность. Самое главное, что мы попытались сделать самый лучший проект на который мы способны. И у нас получилось.

**Планы на будущее**

* 1. **Добавить новые функции по расширению голосового ассистента**:
* Добавить функции, которые будут связаны с устройствами самого компьютера. Например, включение микрофона, выключение, убавление звука, прибавление звука и так далее.
* Добавить больше разных функции по включению сайтов и приложении.
* Добавить
  1. **Оптимизировать программу:**
* Улучшить читабельность кода
* Добавить больше комментарии по описанию кода
* Улучшить скоросость запуска приложения.
  1. **Расширение библиотеки Vosk:**
* Расширить словарь библиотеки и добавить новые слова, словосочетания для улучшения работы приложения.
* Мы нашли статьи которые помогут нам в этом

[https://habr.com/ru/amp/publications/735480/]

[https://ask.bc-pf.org/t/biblioteka-vosk/8849]

* 1. **Добавить в код программы изменение по шумоизоляции**
* Добавить функции современного krisp.ai для шумоподавления лишних звуков
  1. **Создать свою ИИ на казахском:**
* Создать
  1. **Перевести голосового ассистента в ООП:**
* Улучшить удобный и приятный интерфейс для использования приложения
  1. **Создать возможный аналог “Яндекс Станции”**

**Отзыв**



**Список использованных источников и ресурсов**

**Курсы по python и java:**

1. [**https://stepik.org/course/58852**](https://stepik.org/course/58852)
2. **https://stepik.org/course/68343**
3. **https://stepik.org/course/82867**

**Вопросы:**

1. [**https://qna.habr.com/q/1305954**](https://qna.habr.com/q/1305954)
2. [**https://ask.bc-pf.org/t/biblioteka-vosk/8849**](https://ask.bc-pf.org/t/biblioteka-vosk/8849)
3. **https://www.youtube.com/watch?v=MXdsPKZyZ48&pp=ygUQ0LrRgNC10L3QtNC10LvRjA%3D%3D**
4. [**https://youtu.be/fodf4Pttve4?feature=shared**](https://youtu.be/fodf4Pttve4?feature=shared)
5. **https://youtu.be/xWUIGUVfOTg?feature=shared**