ОЛИМПИАДА ШКОЛЬНИКОВ «ШАГ В БУДУЩЕЕ»

НАУЧНО-ОБРАЗОВАТЕЛЬНОЕ СОРЕВНОВАНИЕ «ШАГ В БУДУЩЕЕ, МОСКВА»

ШМ0145

регистрационный номер

Информатика и системы управления

название факультета

Компьютерные системы (ИУ-6)

название кафедры

Разработка кроссплатформенного фреймворка для создания ботов с голосовым двусторонним интерфейсом

название работы

Автор: *Недов Андрей Владимирович*

фамилия, имя, отчество

МБОУ Одинцовский лицей №10, 11«Г»

наименование учебного заведения, класс

Научный руководитель: *Фирсов Денис Владимирович*

фамилия, имя, отчество

ООО "Диджитал Репаблик"

место работы

Заместитель генерального директора

звание, должность

_____

подпись научного руководителя

Москва - 2017

Оглавление

Введение

Компьютер – величайшее изобретение, универсальный инструмент, созданный человеком. Окунувшись в историю взаимодействия человека с компьютером, можно наблюдать разные методы и подходы к подобному взаимодействию. Можно вспомнить консольные интерфейсы, которые работали на пару с клавиатурой и требовали от пользователя определенного образования, а также подготовки. Так, например, для работы с новой машиной необходимо было прочесть «краткое» руководство на несколько десятков страниц. Но прогресс не стоял на месте, и вскоре положение дел значительно поправили графические интерфейсы. Основанные на принципах когнитивной графики и графических метафор эта разновидность интерфейсов значительно расширила аудиторию пользователей компьютеров.

Но, несмотря на это, мы изо дня в день сотни раз кликаем мышью, нажимаем клавиши на клавиатуре и тыкаем в экран нашего смартфона, направляя на него свой взгляд. И многим хотелось бы ускорить и упростить процесс взаимодействия с современными устройствами. Решение этой проблемы появилось относительно недавно – голосовой интерфейс. Понимая перспективность данной технологии крупные корпорации уже ведут разработку и интеграцию подобных интерфейсов в свои многомиллионные проекты. Но речь идет о крупных корпорациях с многомиллиардным оборотом и огромными средствами на исследовательскую и научную деятельность. А теперь представьте частного разработчика, который ввиду своих интеллектуальных способностей считает, что может написать голосового ассистента более совершенного, чем любой из ныне существующих.

Проблема:

Ввиду своей сложности голосовой интерфейс не особо популярен среди частных разработчиков ПО. Вследствие чего, такое перспективное направление как голосовой ассистент, который, например, значительно облегчит управление умным домом, вашим смартфоном или даже автомобилем, развивается очень медленно. А вместе с этим мы отсрочиваем появление программ, способных вести диалог с человеком на человекоподобном уровне.

Цель:

Создать кроссплатформенное ПО, которое позволит сторонним разработчикам, не имеющим особых знаний в области распознавания и синтеза речи, встраивать в свои проекты ботов, распознающих и синтезирующих речь.

Задачи проекта:

1. Изучить проблему.
2. Провести анализ вариантов решения проблемы.
3. Создать работающую версию ПО на базе современных ресурсов.
4. Написать подробные инструкции по использованию созданного ПО.
5. В качестве примера, на основе данного ПО написать самообучающегося чат-бота.

Актуальность проекта:

На сегодняшний день голосовой интерфейс стремительно встраивается в жизнь современного человека, предоставляя массу возможностей. Но речь идет не просто о удобстве и простоте использования подобных систем, но и о экономии личного времен, что является актуальной проблемой на протяжении всего существования человечества.

Раздел 1. Теоретическая часть

1.1 Распознавание речи

1.1.1 История технологии

Моментом зарождения систем распознавания речи по праву можно считать 1952 год, когда корпорация Bell Labs представила систему, названную Audrey. Хотя в реальности, Audrey даже не обладала словарным запасом в привычном для нас понимании, она оперировала только цифрами. При этом система обладала целым набором ограничений, невыполнение которых снижало точность Audrey до 60-70%. Так, например, диктующий человек должен был быть мужчиной, так же он должен был уже ранее работать с системой, а пауза между словами должна была составлять примерно 350 миллисекунд. Если все было сделано по требованиям, точность составляла порядка 90%.

Следующий толчок в развитии данной технологии произошел в 1962 году, когда на Всемирной выставке в Сиэтле компания IBM представила компьютер Shoebox, который был способен распознавать уже целых шестнадцать слов. Таким образом для перехода от цифры к букве потребовалось целых 10 лет.

Далее, в 70-ых годах данной технологией всерьез заинтересовалось министерство обороны США. Все инвестиции оказались полностью оправданы, ведь в период с 1971 по 1976 год была создана система Harpy. Harpy понимала 1011 слов, что является средним словарным запасом трехлетнего ребенка.

Несмотря на огромное количество средств, сил и времени, вложенных в разработку подобных систем, первый поистине массовый продукт с их использованием появился лишь в 2001, когда корпорация Microsoft выпустила Office XP – новейшую версию пакета офисных программ с интегрированной системой распознавания речи.

Уже 2002 году компания Google запускает, правда в тестовом режиме, Voice Search, предназначенного для голосового поиска в сети интернет. Но данную разработку пришлось сразу свернуть. Дело в том, что, чтобы выполнять данный поиск, требовалось звонить на специальный номер, что было весьма неудобно. Но Google не опустил рук, и продолжал разработки в этом направлении. В 2011 году Google учел ошибки прошлых лет, результатом чего явилась функция распознавания в голосовом поиске смартфонов под управлением ОС Android и в браузере Chrome. Были устранены ненужные звонки и прочие неудобства. На сегодняшний день в базе насчитывается около 230 миллиардов слов на многих языках мира.

1.1.2 Принцип работы

Для понимания принципа работы систем распознавания речи, необходимо понять, что же такое речь с физической точки зрения. Наша речь – это последовательность определенных звуков. Звук в свою очередь – это набор звуковых колебаний, звуковых волн, которые являются ничем иным как колебаниями частиц в атмосфере. Для анализа этих колебаний компьютером, необходимо предоставить ему их в цифровом виде, что достигается посредством микрофона.

Тут стоит отметить, что подходов к интерпретации полученных данных существует много и они могут существенно отличаться, в зависимости от требований к системе.

Системы распознавания речи классифицируются:

1. по размеру словаря (ограниченный набор слов, словарь большого размера);
2. по зависимости от диктора (дикторозависимые и дикторонезависимые системы);
3. по типу речи (слитная или раздельная речь);
4. по назначению (системы диктовки, командные системы);
5. по используемому алгоритму (нейронные сети, скрытые Марковские модели, динамическое программирование);
6. по типу структурной единицы (фразы, слова, фонемы, дифоны, аллофоны);
7. по принципу выделения структурных единиц (распознавание по шаблону, выделение лексических элементов).

Так корпорация Google для получения наиболее качественного результата использует сервера с параллельно работающими нейронными сетями, что, несмотря на необходимость подключения к интернету, дает заметное преимущество над ныне существующими off-line системами.

Так же стоит отметить одну из главных преград на пути к качественному распознаванию. Речь идет о шумах и помехах. Для систем автоматического распознавания речи, помехозащищённость обеспечивается, прежде всего, использованием двух механизмов:

1. Использование нескольких, параллельно работающих, способов выделения одних и тех же элементов речевого сигнала на базе анализа акустического сигнала;
2. Параллельное независимое использование сегментного (фонемного) и целостного восприятия слов в потоке речи.

1.2 Синтез речи

1.2.1 История технологии

С наступлением XX века началась эра электрических машин, и учёные получили возможность использовать генераторы звуковых волн и на их базе строить алгоритмические модели.

В 1930-х годах работник Bell Labs Хомер Дадли, работая над проблемой поиска путей для снижения пропускной способности, необходимой в телефонии, чтобы увеличить её передающую способность, разрабатывает — управляемый с помощью клавиатуры электронный анализатор и синтезатор речи. Идея Дадли заключалась в том, чтобы проанализировать голосовой сигнал, разобрать его на части и пересинтезировать в менее требовательный к пропускной способности линии. Усовершенствованный вариант вокодера Дадли, VODER, был представлен на Нью-Йоркской Всемирной выставке 1939 года.

Первые синтезаторы речи звучали довольно неестественно, и часто едва можно было разобрать производимые ими фразы. Однако качество синтезированной речи постоянно улучшалось, и речь, генерируемую современными системами синтеза речи, порой не отличить от реальной человеческой речи. Но несмотря на успехи электронных синтезаторов речи, исследования в области создания механических синтезаторов речи по-прежнему ведутся, например, для использования в роботах-гуманоидах.

Первые системы синтеза речи на базе вычислительной техники стали появляться в конце 1950-х годов, а первый синтезатор «текст-в-речь» был создан в 1968 году.

1.2.2 Принцип работы

Стоит отметить, что главным синтезатором человеческой речи является сам человек. С подобным синтезом мы справляемся вполне успешно в течение уже многих тысяч лет. А вот искусственный синтез речи – задача на порядок труднее.

Задача преобразования текста в речь – очень зависит от требований к системе и имеет множество подходов к реализации. Так, например, для объявления времени отправления поездов на железнодорожном вокзале необходимо всего лишь записать набор отдельных слов, произнесенных диктором, и далее выстраивать из этих слов необходимые предложения. Это самый просто и древний метод синтеза речи, который используется и по сей день. Однако, что если пользователю необходимо управлять интонацией, тембром, скоростью произносимой речи? Или просто, нужно озвучить книгу,

в которой находится несколько тысяч разных слов. Как и в случае с распознаванием речи, подходов к реализации системы, удовлетворяющей данным требованиям, существует множество.

Все способы синтеза речи можно подразделить на группы:

1. параметрический синтез;
2. конкатенативный, или компиляционный синтез;
3. синтез по правилам;
4. предметно-ориентированный синтез.

В качестве примера рассмотрим технологию синтеза речи компании Яндекс, описанную на их официальном сайте.

«Специальный алгоритм подготавливает текст, чтобы роботу было удобно его читать: записывает все числа словами, разворачивает сокращения. Затем текст делится на фразы, то есть на словосочетания с непрерывной интонацией — для этого компьютер ориентируется на знаки препинания и устойчивые конструкции. Для всех слов составляется фонетическая транскрипция. Чтобы понять, как читать слово и где поставить в нём ударение, робот сначала обращается к классическим, составленным вручную словарям, которые встроены в систему. Если нужного слова в словаре нет, компьютер строит транскрипцию самостоятельно — опираясь на правила, заимствованные из академических справочников.

Когда транскрипция готова, компьютер рассчитывает, как долго будет звучать каждая фонема, то есть сколько в ней фреймов — так называют фрагменты длиной 25 миллисекунд. Затем каждый фрейм описывается по множеству параметров: частью какой фонемы он является и какое место в ней занимает; в какой слог входит эта фонема; если это гласная, то ударная ли она; какое место она занимает в слоге; слог — в слове; слово — в фразе; какие знаки препинания есть до и после этой фразы; какое место фраза занимает в предложении; наконец, какой знак стоит в конце предложения и какова его главная интонация.

Другими словами, для синтеза каждых 25 миллисекунд речи используется множество данных. Информация о ближайшем окружении обеспечивает плавный переход от фрейма к фрейму и от слога к слогу, а данные о фразе и предложении в целом нужны для создания правильной интонации синтезированной речи.»

1.3 Кроссплатформенность

Кроссплатформенность – способность ПО работать более чем на одной аппаратной платформе и/или операционной системе. Это критически важная способность для ПО в век, когда даже часы имеют свою операционную систему. Кроссплатформенное ПО способно не только значительно упростить процесс программирования, но и сократить количество специалистов, необходимых для реализации продукта, ведь код пишется один раз и не зависит от платформы.

Кроссплатформенный голосовой бот будет иметь массу возможностей. Например, разработчик сможет встроить подобного бота в систему «умный дом», а взаимодействовать с ним через смартфон, персональный компьютер и даже через одноплатный компьютер Raspberry Pi. И все это при условии того, что код пишется разработчиком один раз.

Раздел 2. Анализ вариантов решения проблемы

2.1. Решение проблемы кроссплатформенности

Чтобы сделать ПО кроссплатформенным, необходимо прежде всего выявить то, что объединяет платформы. Решением проблемы стал WEB.

WEB – это интернет пространство. Все это пространство размечено по одним законам и правилам. И для каждой платформы разработчики строят свои «мосты» к данному пространству – браузеры. Благодаря браузерам пользователи способны просматривать содержимое Web, а также взаимодействовать с ним. Код разметки Web-страницы одинаков для любой платформы, а соответственно программное решение, фундаментом которой является Web, имеет все шансы на кроссплатформенность.

Ввиду того, что наше ПО будет базироваться на Web, распознавать и синтезировать речь мы будем тоже с помощью Web-технологий. Среди современных инструментов можно выделить JavaScript библиотеки, встроенные в браузер объекты. За основу данного проекта были выбраны именно эти библиотеки.

Запускаться проект будет в браузере, а нашими инструментами будут JavaScript, HTML и CSS (приложение 1).

2.2 Анализ и выбор средств для распознавания речи

Было рассмотрено несколько технологий для распознавателя речи:

SpeechKit

SpeechKit – это набор онлайн JavaScript библиотек созданная компанией Яндекс, созданный для создания голосовых интерфейсов.

Плюсы:

1. Качество распознавания. Разработчиком заявлена точность распознавания в 94%.
2. Поддержка. На сайте производителя есть отдельный раздел «обратная связь», в который можно обратиться, если пользователь не нашел ответ на свой вопрос в документации.
3. Качественное API. Продуманная разработчиками система не препятствует пониманию кода, и программировать голосовой интерфейс, используя SpeechKit, действительно удобно.

Минусы:

1. Регистрация. Для использования данной библиотеки необходима регистрация на сайте разработчика. Если русскоязычных пользователей эта процедура не затруднит, то вот иностранным разработчикам придется провести определенное количество времени в компании переводчика.
2. Ключ с каждым запросом. С каждым запросом на распознавание требуется вставлять в код свой уникальный ключ, который предоставляется разработчику только после регистрации.
3. Платная основа. После пробного периода в один месяц, нужно заключить договор. Стоимость лицензии зависит от количества запросов. В среднем она составляет 400 рублей за 1000 запросов.

PocketSphinx

PocketSphinx – это JavaScript библиотека для распознавания речи.

Плюсы:

1. Off-line распознавание. Для работы данной библиотеки подключение к интернету не требуется.
2. Бесплатно. Библиотека находится в свободном доступе.

Минусы:

1. Низкая точность распознавания. Ресурсы одного компьютера весьма ограничены, ввиду чего данная библиотека хорошо справляется лишь с распознаванием заранее заданных команд. Распознает речь PocketSphinx плохо.
2. Необходимость дополнительных ресурсов. Для распознавания речи вам понадобятся библиотеки: pocketsphinx.js, recognizer.js, audioRecorder.js и callbackManager.js. Которые в сумме усложняют код.

Web Speech API Specification

Web Speech API Specification – это объект встроенный во все браузеры Google Chrome для распознавания и синтеза речи.

Плюсы:

1. Высокая точность распознавания.

2. Качественное API. Программировать голосовой интерфейс, используя Web Speech API Specification, очень удобно. Даже подключать ничего не требуется, ведь объект уже встроен в браузер.
3. Бесплатно. С 2012 года технология находится в свободном доступе.

Минусы:

1. On-line распознавание. Для работы данной технологии необходимо подключение к интернету, ввиду того, что распознавание происходит на серверах.

Решение:

Виду платности SpeechKit и низкой точности распознавания речи PocketSphinx данные технологии распознавания речи были исключены из рассмотрения. В качестве оптимальной технологии был выбрана технология Web Speech API Specification. В во втором десятилетии 21-го века подключение к интернету не является трудновыполнимым условием. Ввиду вышеописанных плюсов и некритичности минуса, для интеграции была выбрана данная технология.

2.3 Анализ и выбор средств для синтеза речи

Были найдены и проанализированы следующие решения для синтеза речи:

Web Speech API Specification

Web Speech API Specification – как было уже сказано, данный объект может не только распознавать, но и синтезировать речь.

Плюсы:

1. Правдоподобное звучание.
2. Качественное API.
3. Бесплатно.
4. Возможность настройки голоса. Есть возможность настройки голоса по высоте, скорости, и громкости.

Минусы:

1. On-line синтез.
2. Не поддерживает ОС Android. Данный объект не встроен ни в мобильный браузер Google Chrome, ни в Android WebView.

ResponsiveVoice

ResponsiveVoice – онлайн библиотека для синтеза речи.

Плюсы:

1. Правдоподобное звучание.
2. Качественное API.
3. Бесплатно.
4. Возможность настройки голоса. Так же имеется возможность настройки голоса по высоте, скорости, и громкости.

Минусы:

1. On-line синтез.

Решение:

Ввиду отсутствия поддержки ОС Android объект Web Speech API Specification в качестве синтезатора речи далее не рассматривается. Поскольку мы уже решили, что будем работать с интернетом, то выбираем ResponsiveVoice.

Раздел 3. Разработка проектной технологии

Сборка средств распознавания и синтеза речи в единую систему

3.1 Код распознавания речи

Давайте рассмотрим код для запуска распознавания речи с использованием Web Speech API Specification:

```
var recognition = new webkitSpeechRecognition()
recognition.onresult = function(event) {
if (event.results.length > 0) {
console.log("Final sentence is : " + identificated )
        q.value = event.results[0][0].transcript
}       }
recognition.start()
```

Данный достаточно неинформативный код позволяет лишь начать прослушивание аудиосигнала. Если же добавить пару строк кода и воспользоваться функцией recognition.stop(), то результат распознавания выведется в консоль браузера. Но записать результат в переменную и взаимодействовать с ним не получится.

Для взаимодействия с результатами распознавания необходим более длинный и соответственно более неинформативный код:

```
var recognition = new webkitSpeechRecognition()
var result
recognition.start()
recognition.onresult = function(event) {
        for (var i = event.resultIndex; i < event.results.length; ++i) {
identificated = event.results[i][0].transcript;
if (event.results[i].isFinal) {
        console.log("Final sentence is : " + identificated)
        setTimeout(function(){
                recognition.stop()
                result = identificated
        },1500)       }       }       }       ...
```

Программистам, не работавшим с данной технологией, знакомство с ней, будет стоить большого количества времени, которое могло было быть потрачено на разработку проекта. Да и в итоге код проекта будет труден для понимания, ведь он будет достаточно громоздким.

Для упрощения работы с создаваемым ПО весь выше представленный код был заменен на две строчки:

```
listen()                    // начать прослушивание
```

```
function onListenEnd(inn){…}
                    // получаем результат и работаем с ним
```

Для выбора языка распознавания необходимо указать необходимый параметр строкой:

```
language()                  // язык распознавания
```

В случае, если данный параметр не указывается, язык устанавливается на язык, установленный в браузере.

Для остановки прослушивания и возвращения результатов распознавания была создана функция:

```
listenStop()                // остановка прослушивания
```

Если не указать в коде данную функцию, то прослушивание завершится на интонационной паузе, означающей конец предложения.

Получается, что для того, чтобы распознать текст фразы, сказанной на русском языке и вывести результат с помощью alert(), необходимо написать вот этот небольшой код:

```
language("ru")
listen()
function onListenEnd(inn){ alert(inn) }
```

3.2 Код синтеза речи

Код для синтеза речи с использованием ResponsiveVoice.JS значительно проще и понятнее, чем код для распознавания.

На практике, чтобы синтезировать «привет» на русском языке с помощь данной библиотеки необходимо написать следующий код:

```
<script src='https://code.responsivevoice.org/responsivevoice.js'></script>

    responsiveVoice.speak(
        "Привет",
        "Russian Female",
        {
            onstart:null,
            onend:null,
            pitch:1,
            rate:1,
            volume:1
        }
    )
```

Но и этот код было решено упростить:

```
say("Привет","Russian Female",null,null,1,1,1)
                    //говорим «привет» на русском!
```

Таким образом использовать данное ПО сможет даже программист-новичок, не обладающий глубокими познаниями ни в синтезе, ни в распознавании речи, ни в даже JavaScript.

3.3 Запуск фреймворка на разных операционных системах

Поскольку Web Speech API Specification – объект, встроенный исключительно в Google Chrome, запускать проекты, созданные на основе нашего ПО можно будет только в данном браузере.

Но относительно недавно компания Google выпустила обновление для Chrome, которое блокирует любые попытки доступа к микрофону и камере устройства из локального файла.

Для решения этой проблемы были созданы исполняемые файлы со скриптами для разных операционных систем, которые будут запускать Google Chrome в специальном режиме, который, в свою очередь, позволит взаимодействовать с аппаратным обеспечением компьютера с помощью локально сохраненного кода.

3.3.1 Создание файла запуска на ОС Windows

Для ОС Windows было решено создать пакетный файл с расширением .bat. Сам код файла выглядит так:

```
"C:\ProgramFiles(x86)\Google\Chrome\Application\
chrome.exe"--allow-file-access-from-files
C:\Users\GoodInI\Desktop\Casper_3.0\index.html
```

3.3.2 Создание файла запуска на ОС Linux и MacOS

Для ОС Linux и ОС MacOS был создан bash-скрипт, код которого выглядит так:

```
import os
os.system("open /Applications/Google\
Chrome.app --args --allow-file-access-from-files")
```

3.3.3 Запуск на ОС Android

С ОС Android все немного иначе. Дело в том, что для данной ОС не существует исполняемых файлов кроме APK, да и взаимодействовать с программой через браузер на смартфоне – очень некомфортное и не эстетичное занятие.

Поддержка данной платформы была достигнута благодаря использованию фреймворка Apache Cordova (приложение 2).

Данный фреймворк позволяет создать APK файл, с развернутым на всю диагональ экрана WebView, в котором, в свою очередь, исполняется локальный HTML или JS файл. Подключив к нашему Cordova-проекту два плагина

1. cordova-plugin-media,
2. cordova-plugin-crosswalk-webview

мы можем запускать проект на ОС Android в виде обычного приложения.

3.4 Разработка архитектуры

Стоит отметить, что наше ПО имеет полное право называться фреймворком, ввиду того, что оно является программным обеспечением, облегчающим разработку и объединение разных компонентов большого программного проекта.

Назвать данный фреймворк было решено UGhost (от your ghost, твой призрак)

Была выбрана следующая архитектура:

- **папка «me»**
  - файл «subconscious.js»
- **папка «start»**
  - файл «Windows.bat»
  - файл «Linux_MacOS.bash»
- **папка «write_me»**
  - файл «consciousness.js»
- файл «index.html»

subconscious.js – файл, содержащий все функции, описанные в пунктах 1 и 2 третьего раздела;

Windows.bat и Linux_MacOS.bash – файлы описанные в третьем пункте третьего раздела;

consciousness.js – файл, предназначенный для js-кода разработчика;

index.html – файл предназначенный для html и css-кода разработчика.

Раздел 4. Создание бота, на основе разработанного ПО

Для демонстрации работы созданного фреймворка было решено создать бота, распознающего и синтезирующего речь с использованием элемента самообучения.

Принцип работы бота:

- У бота есть база данных в виде двумерной таблицы, в которой в которой в первый столбец записываются запросы, а во второй ответы. Так, каждой ячейке с запросом соответствует только одна ячейка с ответом.
- После нажатия кнопки START компьютер начинает прослушивание запроса. Пользователь произносит запрос.
- Алгоритм сверяет запрос с базой данных. Если данного запроса в базе нет, то бот вносит его в свободную ячейку запросов и синтезирует фразу «Что мне нужно ответить?», тем самым уведомляя пользователя, что необходимо внести в базу ответ.
- Далее начинается прослушивание ответа. Когда ответ был произнесен и распознан он вносится в соответствующую ячейку ответов.
- Бот уведомляет пользователя о том, что ответ был записан, синтезируя слово «Запомнила», и, возвращаясь к началу алгоритма, начинает прослушивание запроса.
- Если же запрос в базе данных уже есть, то синтезируется ответ из соответствующей ему ячейки ответов.

Алгоритм работы созданного бота представлен на блок-схеме в приложении 3.

Заключение

В результате реализации проекта был создан уникальный кроссплатформенный фреймворк, а также руководство по его использованию (приложение 4). Это ПО позволяет сторонним разработчикам, не имеющим особых знаний в области распознавания и синтеза речи, создавать ботов, распознающих и синтезирующих речь. Тем самым с помощью данной разработки появилась возможность реализовывать голосовой интерфейс в практически любом программном проекте. Достоинствами данной разработки также являются простота в ее использовании, бесплатность и открытый программный код.

Список литературы

1. http://antonkozlov.ru/istoriya/istoriya-sistem-raspoznavaniya-rechi.html
2. https://habrahabr.ru/post/226143/
3. https://ru.wikipedia.org
4. https://yandex.ru/blog/company/kak-eto-rabotaet-sintez-rechi
5. http://responsivevoice.org/
6. https://dvcs.w3.org/hg/speech-api/raw-file/tip/speechapi.html
7. https://wsd.events/2014/10/26/pres/speech/
8. http://stackoverflow.com
9. https://tech.yandex.ru/speechkit/
10. https://syl22-00.github.io/pocketsphinx.js/
11. https://cordova.apache.org/
12. https://crosswalk-project.org/
13. https://github.com

Приложения

Приложение 1.

HTML - это язык разметки веб-страниц, широко используемый в создании сайтов. Он представляет из себя набор графических объектов и их контейнеров, которые могут содержать атрибуты, а также работать с CSS-стилями и JavaScript-обработчиками событий.

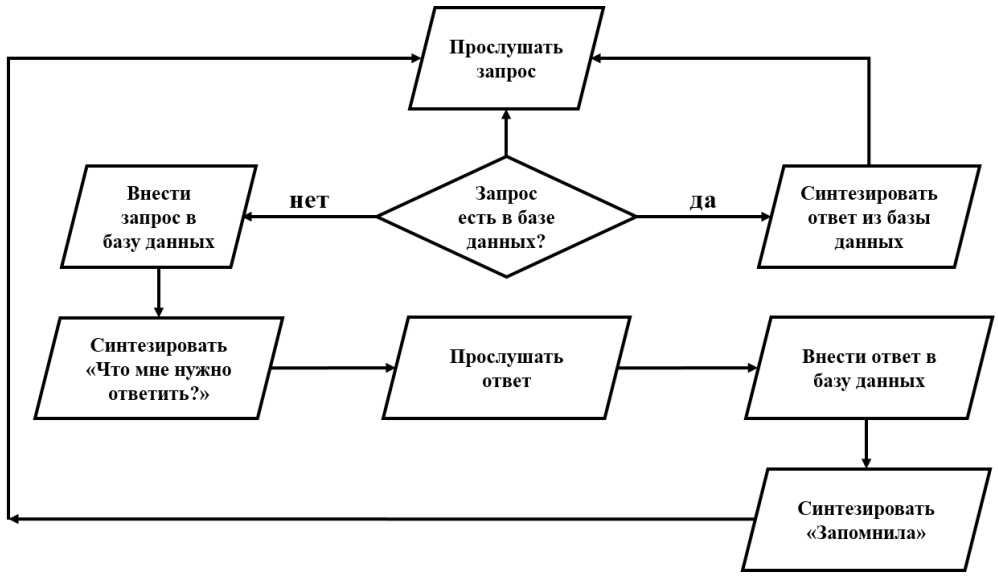
CSS представляет из себя набор классов и их свойств. С помощью CSS были заданы шаблоны кнопок, панелей меню и шрифты.

JavaScript – это прототипно-ориентированный сценарный язык программирования. Наиболее широкое применение находит в браузерах как язык сценариев для придания интерактивности веб-страницам.

Приложение 2.

Apache Cordova — это платформа разработки мобильных приложений с открытым исходным кодом. Она позволяет использовать стандартные веб-технологии, такие как HTML5, CSS3 и JavaScript для кросс платформенной разработки, избегая родного языка разработки для каждой из мобильных платформ. Приложения выполняются внутри обертки нацеленной на каждую платформу и полагаются на стандартные API для доступа к датчикам устройства, данным и состоянию сети.

Приложение 3.



Приложение 4

UGhost

Кроссплатформенный фреймворк для создания ботов с голосовым двусторонним интерфейсом

Содержание

Оглавление

UGhost API

Создатель:

Недов Андрей

О фреймворке

UGhost – кроссплатформенный фреймворк, позволяющего встраивать в свои проекты ботов, распознающих и синтезирующих речь.

Платформы

- ✓ Windows (7 и выше)
- ✓ Linux (+GCC v4.6 и +GTK v2.24)
- ✓ iOS (7.0 и выше)
- ✓ Android (4.4 и выше)

Установка

1. Скачайте и установите Google Chrome
2. Скачайте архив по ссылке:
3. Разархивируйте архив в любую директорию
4. В папке start ->
    - для Windows:
        1. файл *Windows.bat* -> правая кнопка мыши -> изменить -> после *--allow-file-access-from-files* укажите путь до *index.html*

        2. вместо *chrome.exe* укажите путь до *chrome.exe*

        пример:
        "C:\Program Files (x86)\Google\Chrome\Application\chrome.exe" --allow-file-access-from-files C:\Users\MY_PC\Desktop\Casper_2.0\index.html

- для Linux и Mac OS:

5. файл *Linux_MacOS* -> изменить расширение на .txt -> вместо - && укажите путь до *chrome.app* -> изменить расширение обратно на .bash
6. после запуска файла в адресную строку введите путь до index.html

пример:

import os os.system("open /Applications/Google\ Chrome.app --args --allow-file-access-from-files")

ВНИМАНИЕ! Все пути должны быть прописанны исключительно на английском! Символы других алфавитов приведут к ошибкам!

Запуск на Windows, Linux и Mac OS

1. Завершите все процессы Google Chrome
2. Запустите
    - для Windows - *Windows.bat*
    - для Linux и Mac OS - *Linux_macOS.bash*

Запуск на Android

1. Создайте проект cordova
2. Подключите плагин *cordova-plugin-media* (**$cordova plugin add cordova-plugin-media**)
3. Подключите плагин *cordova-plugin-crosswalk-webview* (**$cordova plugin add cordova-plugin-crosswalk-webview**)
4. Очистите директорию *www* Вашего cordova проекта
5. Скопируйте всё содержимое папки фреймворка *casper,* за исключением папки *start,* в папку *www* Вашего cordova проекта
6. Измените шапку (head) вашей index.html страницы для работы с cordova
7. Соберите Ваш cordova проект

Рабочее пространство

- файл *index.html* для разметки HTML страницы Вашего проекта
- папка *write_me* предназначена для дополнительных файлов Вашей HTML страницы
- файл *consciousness.js* предназначен для логических скриптов Вашей HTML страницы

Распознавание речи

language(lang) // выбор языка распознавания

listen() // начать распознавание

listenStop() // прервать распознавание (с получением результатов)

function onListenEnd(inn){…} // распознавание окончено (с получением результатов)

Пример 1

```
language("ru")
listen()
function onListenEnd(inn)
{
      alert(inn)
}
```

В данном примере сигналом для окончания распознавания является интонационная пауза

Синтез речи

say(текст, язык, event start, event stop, высота, скорость, громкость)

пределы регулировки:

- высота 0 – 2
- скорость 0 – 1.5
- громкость 0 – 1

Пример 2

```
language("ru")
listen()
function onListenEnd(inn)
{
	say(inn, "Russian Female", null, null,1,1,1)
}
```

Языки

Синтез:

- ✓ UK English Female
- ✓ UK English Male
- ✓ US English Female
- ✓ Spanish Female
- ✓ French Female
- ✓ Deutsch Female
- ✓ Italian Female
- ✓ Greek Female
- ✓ Hungarian Female
- ✓ Turkish Female
- ✓ Russian Female
- ✓ Dutch Female
- ✓ Swedish Female
- ✓ Norwegian Female
- ✓ Japanese Female
- ✓ Korean Female
- ✓ Chinese Female
- ✓ Hindi Female
- ✓ Serbian Male
- ✓ Croatian Male
- ✓ Bosnian Male
- ✓ Romanian Male
- ✓ Catalan Male
- ✓ Australian Female
- ✓ Finnish Female
- ✓ Afrikaans Male
- ✓ Albanian Male
- ✓ Arabic Male
- ✓ Armenian Male
- ✓ Czech Female
- ✓ Danish Female
- ✓ Esperanto Male
- ✓ Hatian Creole Female
- ✓ Icelandic Male
- ✓ Indonesian Female
- ✓ Latin Female
- ✓ Latvian Male
- ✓ Macedonian Male
- ✓ Moldavian Male
- ✓ Montenegrin Male
- ✓ Polish Female
- ✓ Brazilian Portuguese Female

- ✓ Portuguese Female
- ✓ Serbo-Croatian Male
- ✓ Slovak Female
- ✓ Spanish Latin American Female
- ✓ Swahili Male
- ✓ Tamil Male
- ✓ Thai Female
- ✓ Vietnamese Male
- ✓ Welsh Male

Распознавание:

- ✓ Afrikaans [af-ZA]
- ✓ Bahasa Indonesia [id-ID]
- ✓ Bahasa Melayu [ms-MY]
- ✓ Català [ca-ES]
- ✓ Čeština [cs-CZ]
- ✓ Dansk [da-DK]
- ✓ Deutsch [de-DE]
- ✓ English [en]
  - [en-AU] Australia
  - [en-CA] Canada
  - [en-IN] India
  - [en-NZ] New Zealand
  - [en-ZA] South Africa
  - [en-GB] United Kingdom
  - [en-US] United States
- ✓ Español [es]
  - [es-AR] Argentina
  - [es-BO] Bolivia
  - [es-CL] Chile
  - [es-CO] Colombia
  - [es-CR] Costa Rica
  - [es-EC] Ecuador
  - [es-SV] El Salvador
  - [es-ES] España
  - [es-US] Estados Unidos
  - [es-GT] Guatemala
  - [es-HN] Honduras
  - [es-MX] México
  - [es-NI] Nicaragua
  - [es-PA] Panamá
  - [es-PY] Paraguay
  - [es-PE] Perú
  - [es-PR] Puerto Rico
  - [es-DO] República Dominicana
  - [es-UY] Uruguay

	•	[es-VE]	Venezuela
✓ Euskara		[eu-ES]	
✓ Filipino		[fil-PH]	
✓ Français		[fr-FR]	
✓ Galego		[gl-ES]	
✓ Hrvatski		[hr_HR]	
✓ IsiZulu		[zu-ZA]	
✓ Íslenska		[is-IS]	
✓ Italiano		[it-IT]	Italia
	•	[it-CH]	Svizzera
	•	[lt-LT]	Lietuvių
✓ Magyar		[hu-HU]	
✓ Nederlands		[nl-NL]	
✓ Norsk bokmål		[nb-NO]	
✓ Polski		[pl-PL]	
✓ Português		[pt-BR]	Brasil
	•	[pt-PT]	Portugal
✓ Română		[ro-RO]	
✓ Slovenščina		[sl-SI]	
✓ Slovenčina		[sk-SK]	
✓ Suomi		[fi-FI]	
✓ Svenska		[sv-SE]	
✓ Tiếng Việt		[vi-VN]	
✓ Türkçe		[tr-TR]	
✓ Ελληνικά		[el-GR]	
✓ Български		[bg-BG]	
✓ Русский		[ru-RU]	
✓ Српски		[sr-RS]	
✓ Українська		[uk-UA]	
✓ 한국어		[ko-KR]	
✓ 中文		[cmn-Hans-CN]	普通话 (中国大陆)
	•	[cmn-Hans-HK]	普通话 (香港)
	•	[cmn-Hant-TW]	中文 (台灣)
	•	[yue-Hant-HK]	粵語 (香港)
✓ 日本語		[ja-JP]	
✓ हिन्दी		['hi-IN]	
✓ ภาษาไทย		[th-TH]	

Информация о лицензиях

Распознавание речи: https://dvcs.w3.org/hg/speech-api/raw-file/tip/speechapi.html

Синтез речи: http://responsivevoice.org/