СЕВАСТОПОЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

UI

JS

ASR

Плагин

API

РЕФЕРАТ

Объем дипломного проекта составляет 100 страниц, в нем содержится 5 таблиц, 6 иллюстраций, 15 приложений и 20 источников литературы.

Ключевые слова: программное обеспечение, веб-ориентированная система, голосовое управление, распознавание речи, обучаемый помощник.

Целью работы является разработка и программная реализация модульной обучаемой веб-ориентированной системы голосового управления, предоставляющей интерфейс для написания пользователями собственных голосовых функций.

Были изучены существующие аналоги, рассмотрены современные технологии распознавания речи и методы классификации текстовых документов. Приведено обоснование выбора и характеристика комплекса технических средств для реализации программного продукта и оценка его информационных характеристик.

Результатом дипломного проектирования стала система голосового управления «Джарвис», разработанная в двух версиях: в виде desktop-приложения под все современные операционные системы и в виде расширения для браузеров, основанных на движке webkit (Google Chrome, Yandex Browser).

Областью применения спроектированной системы является

СОДЕРЖАНИЕ

1. Введение
2. Основная часть, содержащая концепции и решения, сведения о методах, алгоритмах, программах (в целом и по подзадачам); используемые технологии, средства и инструменты разработки; результаты по подзадачам, результаты тестирования разработанных подсистем.

2. Теоретическа часть

2.1 История развития систем распознавания речи

2.2 Классификация систем распознавания речи

2.3 Методы и алгоритмы систем распознавания речи

2.4 Методы и алгоритмы систем классификации текстовых документов

2.5 Анализ существующих систем распознавания речи

2.6 Актуальность и выводы

3. Практическая часть

3.1 Обоснование выбора и характеристика языка программирования

3.2 Обоснование выбора и характеристика комплекса программных и аппаратных средств

3.3 Описание алгоритма функционирования системы

3.4 Характеристика системы

1. -описание математических методов, обоснование вводимых ограничений и допущений, используемых при решении задачи;
2. -описание алгоритмов и возможного взаимодействия программ с другими программами;
3. -описание и обоснование выбора метода организации входных и выходных данных;
4. -обоснование выбора языка программирования;
5. -описание программ и их применения;
6. -руководство для программиста, программиста и оператора;

ВВЕДЕНИЕ

Десятилетиями ученые и инженеры искали способы, которые позволили бы людям общаться с компьютером так же, как они общаются между собой, а не заставлять пользователей приспосабливаться к языку, который понятен машине. Речь - это самое прямое и естественное средство общения между людьми, и как следствие, это должно быть самым естественным способом взаимодействия с машинами.

В настоящее время пользователями вычислительных машин и средств, оснащенных вычислительными машинами, становятся люди, не являющиеся специалистами в области программирования. Проблема речевого управления возникла, кроме того, в связи с тем, что в некоторых областях применения речь стала единственно возможным средством общения с техникой (в условиях перегрузок, темноты или резкого изменения освещенности, при занятости рук, чрезвычайной сосредоточенности внимания на объекте, который не позволяет отвлечься ни на секунду, и т.д.).

Повсеместное внедрение средств вычислительной техники в различные сферы человеческой деятельности делает разработку речевого интерфейса для взаимодействия с ЭВМ очень актуальной.

Современность темы дипломной работы обусловлена тем, что рынок систем распознавания голоса рос и продолжает стремительно развиваться. По данным исследований BBC, размер рынка в 2011 году составлял 47 миллиардов долларов, 53 миллиарда в 2012 году и ожидается рост до 113 миллиардов к 2017 году, что составляет в общей сложности 16.2% прироста. Это обусловлено тем, что такие технологические гиганты как Google, Facebook, Apple и Microsoft продолжают инвестировать в развитие данных технологий. При этом большую часть рынка занимает потребительский сектор, что отражает увеличивающийся интерес у конечных пользователей современных компьютеров и мобильных устройств. Следует отметить, что увеличивается спрос на системы распознавания голоса и в сфере здравоохранения. Росту также способствуют значительное увеличение вычислительной мощности и доступные объемы памяти среднего настольного персонального компьютера. Теперь нет никаких проблем с техническим обеспечением, необходимым для работы систем распознавания голоса. А увеличение количества поставщиков этой технологии и конкуренция снизили цены разработок до вполне приемлемого уровня.

Практической ценностью работы является разработка кросс-платформенной расширяемой системы с голосовым интерфейсом.

Объектом исследования являются средства распознавания речи и методы классификации текстовых документов.

Предметом исследования является анализ возможности применения средств распознавания речи в задаче разработки ПО с голосовым интерфейсом под десктопные и браузерные платформы.

Цель работы - разработка и программной реализации модульной обучаемой веб-ориентированной системы голосового управления, предоставляющей интерфейс для написания пользователями собственных голосовых функций.

Для достижения цели были поставлены следующие задачи:

* Произвести анализ предметной области, определить состояние современного технического обеспечения, изучить существующие аналоги, установить возможные проблемы и ограничения.
* Разработать схему функционирования системы с голосовым управлением.
* Определить комплекс программных и аппаратных средств для реализации модульной системы с голосовым управлением.
* Разработать модульную систему с голосовым управлением.
* Реализовать систему плагинов для разработанной системы

Дипломная работа состоит из введения, основной части, экономического раздела, раздела охраны труда и безопасности жизнедеятельности, заключения. В введении указывается цель и задачи для ее достижения, обосновывается современность и актуальность решаемой задачи. Основная часть представлена теоретическим и практическим разделами. В теоретическом разделе производится анализ предметной области: рассматривается история развития, классификация и алгоритмы функционирования систем распознавания речи, а так же приводятся существующие разрабатываемой системы. В практическом разделе осуществляется выбор способа реализации поставленной перед автором задачи, описываются алгоритмы, методы и вводимые ограничения. Производится обоснование выбора и характеристика языка программирования, а так же комплекса программных и аппаратных средств для реализации описанной системы. Приводится схема функционирования разработанной системы, руководство для программиста и оператора. Описываются результаты разработки, в частности, характеристики разработанной системы. В экономическом разделе приводятся результаты обоснования принятых решений с точки зрения их экономической целесообразности. Приводятся расчеты себестоимости разработанного продукта и сроков окупаемости, капитальных и эксплуатационных затрат, его конкурентоспособности. В разделе охраны труда и безопасности жизнедеятельности производится анализ условий труда на рабочем месте инженера-программиста и разрабатываются меры по их улучшению. В заключении производятся общие выводы о проделанной работе, указываются основные и смежные отрасли возможного использования спроектированного объекта. Делаются выводы о направлениях и перспективах его дальнейшего развития.

1 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

Речевой интерфейс обладает рядом бесспорных и очевидных преимуществ:

* Преодоление физических недостатков. Для пользователей с нарушением зрения или нарушениями моторных функций зачастую трудно работать с компьютерами посредством клавиатуры и мыши. Приложения с качественным речевым интерфейсом предоставляют людям с данными заболеваниями более комфортный способ взаимодействия c машинами.
* Простота. Не требуются навыки работы с компьютером для взаимодействия с системой.
* Возможность одновременного использования стандартных и голосовых средств ввода информации.
* Свобода рук и глаз. Обеспечение мобильности. Речевой интерфейс позволяет человеку взаимодействовать с компьютером с помощью беспроводной гарнитуры, таким образом обеспечивается не зависимое от расположения человека взаимодействие.
* Удобство использования даже при неблагоприятных условиях. Акустические системы имеют большое преимущество когда затрудняется восприятие визуальной информации, например, темнота, слишком большое количество света, блики, вибрации.
* Речевые интерфейсы позволяют выполнять сразу две задачи одновременно, например работать с измерительным устройством и вводить отображаемые значения в систему с помощью голоса. Это преимущество не только в терминах времени, но и в терминах удобства.
* Эффективность. Средний работник офиса печатает со скоростью 50-70 слов в минуту. Однако системы распознавания речи позволяют обрабатывать 120 слов в минуту с высокой точностью. Таким образом, приложения с голосовым интерфейсом позволяют значительно увеличить производительность при написании заметок, отчетов и прочих документов.

Речь является основным средством человеческого общения. Проектирование машины, которая имитирует человеческое поведение, в частности, способность говорить на естественном языке и реагировать должным образом на человеческую речь интриговала инженеров и ученых на протяжении 5 десятилетий. Чтобы понять возможности современных систем распознавания и как они развились до их текущей формы, исторический обзор может оказаться полезным.

Эволюцию систем распознавания речи за последние 5 десятилетий можно разбирать на этапы следующим образом:

* Первое поколение: первые попытки в 1950-1960 годах
* Второе поколение: технологии, основанные на шаблонах в конце 1960-1970 годах
* Третье поколение: технологии, основанные на статистических моделях в 1980 годах
* Позднее третье поколение: продвижения в 1990-х и 2000-х годах

Таким образом, история развития систем распознавания речи берет свое начало еще с 1952 года, когда исследователи американской компании Bell Labs разработали систему для изолированного распознавания чисел, сказанных одним голосом. Другие ранние системы распознавания речи 1950-х годов так же былы спикерозависимыми и ограничены словарным запасом в 10 слов.

В 1960-х годах несколько японских компаний продемонстрировали их возможости в создании специализированных аппаратных средств для выполнения задач распознавания речи. Наиболее примечательными были фонетические распознаватели, позволяющие распознавать 4 гласных звуков и 9 согласных [10]. Путем добавление статистической информации о допустимой последовательности фонем в английском языке они увеличили точность распознавания фонем для слов, состоящих из двух и более фонем. Эта работа отмечена как первое использование статистической информации о синтаксисе на уровне фонем в задачах распознавания речи.

Второе поколение рассматриваемых технологий ознаменовали два значительных открытия: изобретение метода динамического программирования (Dynamic Time Warping - DTW)[25] и скрытых марковский моделей. В течении того периода была основана первая коммерческая компания распознавания речи Threshold Technology, которая разработала продукт VIP-100, способный распознавать до 200 слов. Однако достигнуть спикеронезависимости исследователи того периода так и не смогли.

Одними из известнейших поставщиков ПО в этой области являются Google (Google Voice Search) и Apple (Siri). На отечественном рынке речевыми технологиями в основном занимается Центр Речевых Технологий.

Обзор систем с голосовым управлением. В ходе анализа было выяснено, что в мире существует достаточное количество таких систем. Наиболее известными и продвинутыми голосовыми помощниками на текущий момент являются iPhone's Siri, Android's Google Now и Windows 10's Cortana.  
Чем же отличается разрабатываемая система от остальных?

* Кроссплатформенность. Прежде всего, она не привязана к конкретным аппаратным устройствам или операционным системам, а может функционировать как на мобильных устройствах в виде веб-приложения, так и на устройствах под управлением Windows, Linux или OS X в виде desktop-приложения.
* Офлайн. Благодаря стеку используемых технологий система не требует доступа к интернету для выполнения ваших команд.
* Самообучаемость. То есть ассистент буквально умеет обучаться новым командам от своего владельца. Если система не поняла чего-то, то она всегда спросит вас «Что с этим делать?», а вы сможете направить ее, сказав какое из действий, приведенных в подсказке, выполнить.
* Система плагинов. Система предоставляет простое, удобное и вместе с тем, позволяющее использовать все возможности и преимущества фреймворка AngularJS, API для создания собственных голосовых функций (плагинов). Реализована система управления плагинами: система предоставляет интерфейс для доступа к списку всех доступных плагинов и автоматизирует процесс установки / обновления / удаления их по клику без каких-либо дополнительных действий.

Вы когда-нибудь задумывались о том, как работают те же Siri и Cortana? Ваш голос превращается в текст, потом происходит какая-то магия, и программа выполняет ваш запрос. Или не выполняет, если не поняла, что вы имели в виду. Система предоставляет возможность заглянуть внутрь этой «магии» и создать свои собственные голосовые функции. Причём их возможности становятся действительно безграничными.

* Русский язык. На сегодняшний день все рассмотренные системы в первую очередь ориентированы на англоязычное население, а некоторые и вовсе поддерживают только его. В связи с этим новая функциональность появляется с существенной задержкой для русскоговорящих пользователей. Таким образом лучшей альтернативой может стать стороннее решение, а именно то, что изначально создавалось для русского языка.
* Мультиязычность. Несмотря на изначальную направленность разработанной системы на русскоговорящих пользователей, алгоритм функционирования системы реализован таким образом, что система будет так же хорошо выполнять собственные голосовые функции иноязычных пользователей.
  1. Анализ методов реализации веб-ориентированных систем с голосовым управлением.

В общем случае система с голосовым управлением может быть представлена в виде совокупности следующих подсистем:

* система распознавания речи
* система классификации текста
* система извлечения ключевых слов из текста
* база знаний
* система принятия решений
* агент
  + 1. Анализ существующих систем с голосовым интерфейсом

Разрабатываемая система будет работать на десктопных операционных системах и в браузерах. Выбор таких платформ обусловлен тем, что на сегодняшний день мобильный рынок голосовых помощников переполнен. Даже не смотря на то, что в каждой из трех лидирующих мобильных ОС уже встроен свой интеллектуальный голосовой помощник от ведущих мировых производителей, существует большое количество их аналогов. Таким образом в данном разделе будут рассматриваться только десктопные и браузерные голосовые ассистенты.

Среди веб-приложений наиболее популярными являются:

1. SpeechTexter, Speechnotes
2. ПРАКТИЧЕСКАЯ ЧАСТЬ

2.1 Разработка алгоритма функционирования веб-ориентированной системы с голосовым управлением

После анализа текущего состояния предметной области была разработана схема функционирования веб-ориентированной система с голосовым интерфейсом.

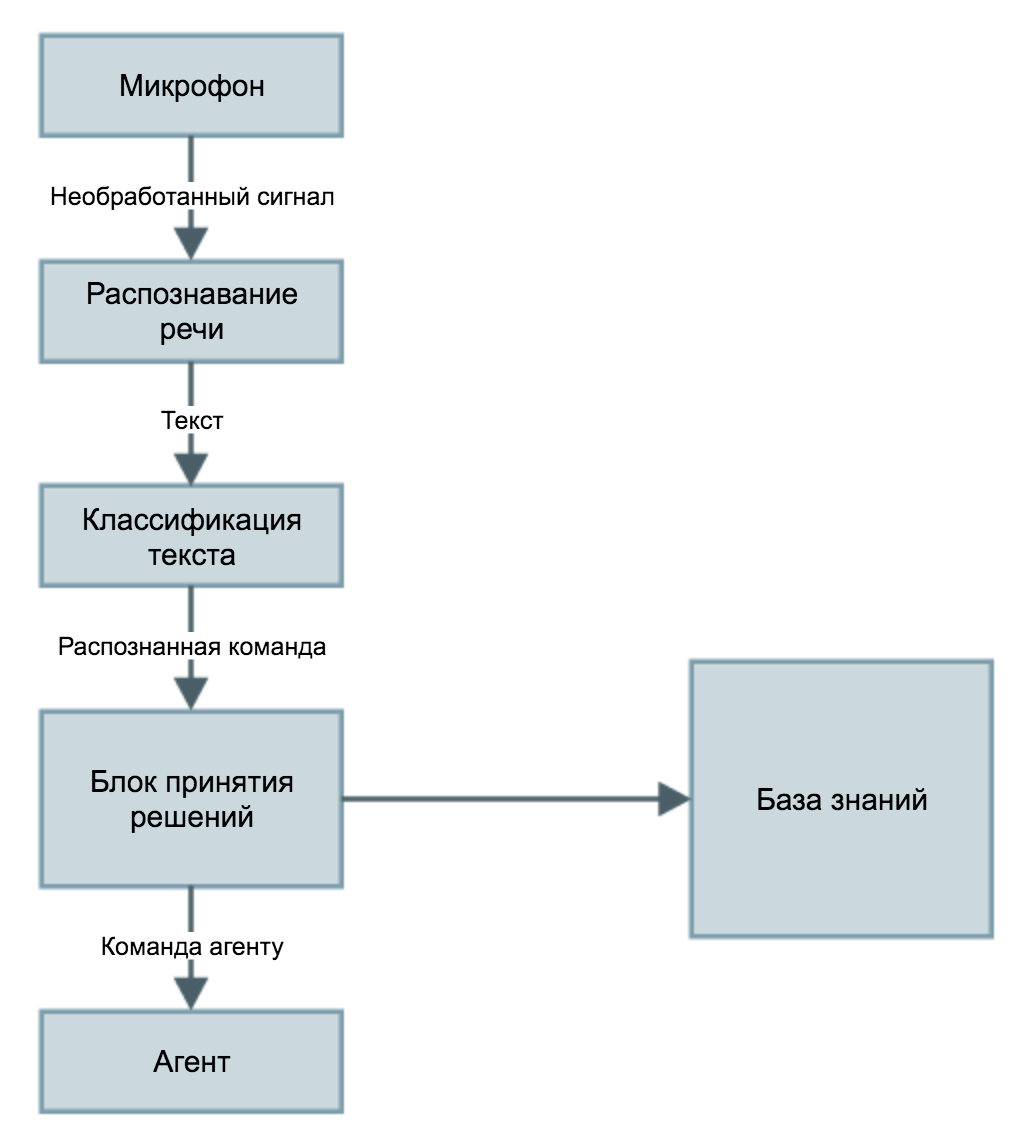


Рисунок 1 – Схема функционирования системы с голосовым интерфейсом

2.1.1 Подсистема распознавания речи

Дипломная работа автора не предполагает реализацию собственной системы распознавания речи, так как это является очень сложной, трудоемкой и ресурсозатратной задачей, которую сложно выполнить в рамках данной работы. В связи с этим анализ представляет собой обзор уже существующих систем, с целью выбрать наиболее оптимальную для разрабатываемого продукта, с возможностью использовать ее на основе лицензионного соглашения.

Были рассмотрены наиболее известные речевые системы преобразования голоса в текст, которые имеют открытое API для интеграции с разрабатываемой системой.

Прежде всего, исходя из требований к разрабатываемой системе, следует отметить основные характеристики, по которым производился выбор оптимальной аудио-системы распознавания речи:

* Поддержка русского языка
* Мультиязычность
* Доступность на всех ОС
* Ориентация на короткие фразы (3-7 слов)
* Скорость и точность обработки
* Высокие нагрузки
* Надежность и отказоустойчивость
* Ограничения на количество обращений
* Сложность интеграции

На сегодняшний день существует достаточно большой выбор среди систем распознавания речи, но лишь немногие соответствуют требованиям, представленным выше. Рассмотрим их:

1. Yandex SpeechKit

Прежде всего, взаимодействие с Yandex SpeechKit ведется через HTTP API. Из этого следует, что технология является доступной для всех платформ и операционных систем, а затраты на развертывание и поддержку собственной инфраструктуры – минимальны. Доля верно распознаваемых русских слов колеблется в диапазоне от 82 до 95%. Распознавание может происходить в реальном времени одновременно с передачей звуковых данных. Задержка от окончания отправки данных до получения результата распознавания не превышает 1 секунды. Инфраструктура SpeechKit спроектирована с учетом высоких нагрузок, что обеспечивает доступность и безотказную работу системы даже при большом количестве одновременных обращений. Кроме того, технология от Yandex обеспечивает простоту интерфейсов, высокую доступность и надежность. Так же, стоит отметить, что данная технология является одной из лучших для распознавания русского и турецкого языка, однако остальные языки она не поддерживает.

1. Google Cloud Speech

Google Cloud Speech, так же как и Yandex SpeechKit, предоставляет разработчикам удобный для использования web API для конвертации аудио в текст с помощью мощных моделей нейронных сетей. Однако, в отличии от технологии Яндекса, Google Cloud Speech распознает более 80 языков. Данная технология обладает так же рядом других особенностей: обработка шума и потоковая передача данных, т.е. возвращение промежуточных результатов распознавания как только они становятся доступны еще во время разговора. Минусом данной технологии является ограничение на количество запросов: до 10 000 в сутки.

1. HTML5 Speech Recognition

HTML5 Speech Recognition является одной из малоизвестных составляющих спецификации HTML5, которую сейчас поддерживают только браузеры, основанные на Webkit. Таким образом, будучи технологией, встроенной в браузеры, она не накладывает каких-либо ограничений на количество запросов в сутки и даже не требует подключения к сети Интернет. Данная технология способна распознавать как непрерывную речь, так и одиночные фразы, а так же работает с большим количеством языков, включая русский. Соответственно ее минусом является то, что не все современные браузеры ее поддерживают. Стоит отметить, что в HTML5 спецификации есть еще одна малоизвестная технология - Speech Synthesis, которая позволяет выполнить обратную операцию: синтезировать речь, используя большое разнообразие голосов.

Таким образом, рассмотрев наиболее подходящие, для поставленной задачи, технологии распознавания речи, можно сделать вывод, что по своей библиотеке данных наиболее точным следует считать продукт Google Cloud Speech. Он более всего подходит для задач распознавания на базе разрабатываемой системы. Однако необходимо заметить, что данный инструментарий имеет ограничение на максимальное количество запросов в сутки, что является критическим недостатком. В то же время у технологии от Яндекса таким недостатком является поддержка слишком малого количества языком. Следовательно, наиболее оптимальным выбором будет являтся HTML5 Speech Recognition, которая изначально интегрирована в браузер, имеет высокую точность и скорость распознавания, поддерживает множество языком, не накладывает каких-либо ограничений по количеству запросов, а так же может работать с непрерывным речевым потоком.

2.1.2 Подсистема классификации текста

После того как речь была распознана, необходимо произвести интеллектуальный анализ текст, а именно – классификацию. Задача классификации - отнесение объекта к одной из заранее определенных категорий на основании его формализованных признаков.

Перед тем, как приступать к любой задаче классификации, одна из наиболее фундаментальных процедур, которую необходимо выполнить, это выбрать представление документа и отобрать признаки. Хоть отбор признаков и применяется довольно часто в других задачах классификации, он особенно важен в задаче классификации текстов ввиду высокой размерности (большого количества признаков) и наличия нерелевантных (шумовых) признаков. В большинстве случаев, представление текста происходит одним из двух способов. Первый – документ как набор слов, в котором документу сопоставляются слова и частота их встречаемости в нем. Такое представление независимо от порядка слов, в котором они встречаются в тексте. Второй метод состоит в представлении текста собственно как набор строк, в котором документ является последовательностью слов. Большинство алгоритмов классификации текстов используют первое представление ввиду его простоты и удобства для задач классификации. Наиболее популярными способами отбора признаков являются удаление стоп-слов и стемминг. При удалении стоп-слов, мы определяем общие слова документов, которые не являются специфичными или разделяющими для разных классов. При стемминге, разные формы одного слова приводятся к одному виду. Например, так объединяются слова разного рода/формы/времени/падежа и пр. Подобная обработка — уже устоявшийся способ уменьшения размерности задачи и улучшения производительности.

Джарвису необходимо знать вероятности того что классифицируемый документ относится к некоторому классу и в случае если она ниже определенного значения предлагать пользователю самому соотнести документ с некоторым классом. Так же необходимо было, чтобы текст внутри классификатора обрабатывался правильно, с учетом особенностей каждого языка, в частности русского.

Были рассмотрены реализации различных алгоритмов классификации текста: kNN, Naive Bayes, логическая регрессия, n-грамм, SVM, дельта TF-IDF для SVM

Машина опорных векторов (Support Vectors Machine, SVM) была предложена для числовых данных. Но ее можно применять и для классификации документов. Главным принципом SVM является определение разделителя в искомом пространстве, который разделяет классы наилучшим образом. Т.е. SVM умеет определять принадлежность документов только к одному из двух классов. Следовательно, чтобы классифицировать документ на принадлежность к одному из N классов, потребуется N классификаторов для каждого класса. Для распознавания текста машина опорных векторов предполагает создание словаря – списка всех слов в обучающей выборки и представление документов в виде вектора, где i-элемент вектора это мера вхождения i-го слова словаря в документ. По мере обучения классификатора, получаются вектора большой размерности, большинство элементов которых – нули. Таким образом данный метод достаточно медлителен, особенно для данных высокой размерности, таких, как текстовые документы.

Наивный байесовский классификатор

Модель наивного Баера показала достаточную эффективность

Если классов всего два («спам / не спам», «давать кредит / не давать кредит», «красное / черное»), то задача называется бинарной классификацией. Если классов несколько — многоклассовая (мультиклассовая) классификация. Также могут иметься образцы каждого класса — объекты, про которые заранее известно к какому классу они принадлежат. Такие задачи называют обучением с учителем, а известные данные называются обучающей выборкой.

В конечном счете, после апробирования более 15 моделей, реализованных под платформу Node.js, путем кросс-проверки на небольшой выборке данных, лучший результат показала BrainJS - имплементация на языке JavasScript простой нейронной сети типа многослойный перцептрон, которая продемонстрировала порядка 91% точности. Для этого было решено несколько проблем. Первая проблема заключалась в том, что BrainJS позволяет обучать нейронную сеть только массивами или хэшами чисел от 0 до 1, следовательно слова текста необходимо преобразовать в цифры, а предложения – в векторы из этих чисел. Следующей проблемой была подача на вход нейронной сети цельного предложения. Дело в том, что обычные нейронные сети прямого распространения должны иметь входные данные фиксированной длины.

А так как предложения содержат разное количество слов, необходимо было привести их к единой длине. Самый простой способ - сложить все вектора, получив таким образом результирующий вектор предложения. Но так как важной особенностью разрабатываемой системы является предоставление возможности пользователям динамически обучать ее, то необходимо расширить этот результирующий вектор на некоторую константу. Таким образом на стадии обучения нейронной сети, после представления предложений обучаемой выборки в виде векторов чисел, они складывались, и к результирующему вектору добавлялось 1000 пустых элементов, что, по мнению автора более чем достаточно для пользователя в рамках одной сессии работы с разработанной системой. Все описанные решения были реализованы с помощью библиотеки Natural, реализованной под платформу Node.js.

Рассмотрим пример предварительной обработки текста. Пусть в обучаемой выборке содержится два предложения: «Ну и какой был в этом смысл?» и «Какая же самая высокая гора в мире?». Тогда перед тем как попасть на вход нейронной сети они пройдут следующие этапы обработки:

1. Токенизация – разбиение текста на слова

tokenize('Ну и какой был в этом смысл?')

// -> [ 'Ну', 'и', 'какой', 'был', 'в', 'этом', 'смысл']

tokenize('Какая же самая высокая гора в мире?')

// -> [ 'Какая', 'же', 'самая', 'высокая', 'гора', 'в', 'мире']

1. Стемминг и удаление стоп слов

stem([ 'Ну', 'и', 'какой', 'был', 'в', 'этом', 'смысл' ])

// -> [ 'Как', 'эт', 'смысл' ]

stem([ 'Какая', 'же', 'самая', 'высокая', 'гора', 'в', 'мире' ])

// -> [ 'Как', 'самый', 'высок', 'гор', 'мир' ]

1. Объединение без пересечения

join([ 'Как', 'эт', 'смысл' ], [ 'Как', 'самый', 'высок', 'гор', 'мир' ])

// -> [ 'Как', 'эт', 'смысл', 'самый', 'высок', 'гор', 'мир' ] + 1000 пустых элементов

1. Преобразование предложений в вектор цифр

// [ 'Как', 'эт', 'смысл', 'самый', 'высок', 'гор', 'мир' ]

features([ 'Как', 'эт', 'смысл' ])

// -> [ 1, 1, 1, 0, 0, 0, 0 ] + 1000 нулевых элементов

features([ 'самый', 'высок', 'гор', 'мир' ])

// -> [ 0, 0, 0, 1, 1, 1, 1 ] + 1000 нулевых элементов

1. Обучение

var net = new brain.NeuralNetwork();

// [ 'Как', 'эт', 'смысл', 'самый', 'высок', 'гор', 'мир' ]

net.train([

// [ 'Как', 'эт', 'смысл' ]

{ input: [ 1, 1, 1, 0, 0, 0, 0 ], output: { 'сорок два': 1 } },

// [ 'Как', 'самый', 'высок', 'гор', 'мир' ]

{ input: [ 1, 0, 0, 1, 1, 1, 1 ], output: { 'Эверест': 1 } }

]);

Пусть пользователь сказал «Можешь сказать какая самая высокая гора в мире?». Тогда классификация текста будет выглядеть следующим образом:

1. Токенизация и стемминг

tokenizeAndStem('Можешь сказать какая самая высокая гора в мире')

// -> [ 'можеш', 'сказ', 'как', 'самый', 'высок', 'гор', 'мир' ]

1. Преобразование предложений в вектор цифр

features([ 'можеш', 'сказ', 'как', 'самый', 'высок', 'гор', 'мир' ])

// -> [ 1, 1, 1, 0, 0, 0, 0 ] + 1000 нулевых элементов

1. Классификация

net.run([ 1, 1, 1, 0, 0, 0, 0 ]); + 1000 нулевых элементов

// -> { Эверест: 0.932615111520198, "сорок два": 0.0695108432441562 }

По-умолчанию, классификатор игнорирует русские слова, хотя поддержка русского языка в проекте есть. Чтобы заставить классификатор понимать русский язык, нужно проинициализировать классификатор, передав в него стиммер для русского языка, заменив таким образом используемый по-умолчанию английский стиммер.

Обучение нейронной сети является ресурсозатратным процессом, однако данная библиотека позволяет сохранять результаты обучения и соответственно загружать, что избавляет от необходимости при каждом запуске повторно тратить время на ее обучение.

2.1.3 Подсистема извлечения ключевых параметров текста

Сравнительно с системой классификации текста, механизм извлечения ключевых параметров довольно примитивен. Исходя из предположения, что классифицируемое предложение, с определенной долей вероятности, является схожим с уже изученным предложением, то механизм представляет собой сравнение слов рядом с меткой. Например, если классифицируемое предложение «Не знаешь, холодно ли в Лондоне» с меткой «Лондон» будет сопоставлено с предложением из обучаемой выборки «Скажи пожалуйста, холодно ли в Севастополе?», то механизм пометит слова после «холодно ли» как местоположение. Разумеется, если предложение будет сопоставлено с несколькими предложениями из обучаемой выборки, то алгоритм пометит то слово, возле которого будет найдено больше всего совпадений. Хоть это и не самый идеальный алгоритм, он работает достаточно хорошо благодаря точности классификации естественного языка.

2.1.4 База знаний

Как уже отмечалось ранее, основное отличие Джарвиса от остальных подобных систем заключается в том, что он предоставляет пользователям возможность самостоятельно обучать ее. Для этого Джарвису необходимо где-то хранить тренировочные данные. Так как формат данных для обучения нейронной сети максимально прост: это текст и соответствующие тексту класс и метка, и никаких дополнительных данных и связей между ними нет и не предвидится, то самым разумным решением, по мнению автора, будет использовать легковесную базу данных DiskDB, основанную на файловой системе и позволяющую хранить данные в json-формате.

2.1.5 Подсистема принятия решений

После того как подсистема классификации текста распознала команду, блок принятия решений передает управление модулю, отвечающему за выполнение этой команды.

2.1.6 Агент

Агент – это отдельный модуль, который выполняет некоторые действия по голосовой команде.

2.2 Обоснование выбора и характеристика языка программирования

Для разработки программного обеспечения в качестве языка программирования был выбран Javascript - высокоуровневый, кросс-платформенный, объектно-ориентированный и интерпретируемый язык программирования. Наряду с HTML и CSS, это одна из трех основных веб-технологий, поддерживаемых всеми современными браузерами без расширений. Он широко используется для реализации как клиентской, так и серверной части веб-приложений. Однако Javascript не ограничивается только веб-ориентированными средами выполнения, а так же может использоваться на платформе node.js и базе данных Apache CouchDB. Характерными особенностями языка являются динамическая типизация, автоматическое управление памятью, прототипное наследование, функции как объекты первого класса, поддержка императивного и функционального стиля программирования. Он имеет API для работы с текстом, массивами, датами и регулярными выражениями, но не предоставляет средств для ввода-вывода, полагаясь в этом на среды выполнения.

Выбор данного языка обусловлен следующими факторами:

* Javascript является самым популярным языком программирования в мире, а следовательно имеет огромное сообщество и большое количество информационных и справочных источников
* Javascript можно использовать как на клиентской так и на серверной стороне, что увеличивает скорость разработки, так как можно сосредоточиться на изучении особенностей только одного языка и увеличивается количество переиспользуемого кода
* Javascript поддерживает объектно-ориентированное программирование, что упрощает модульную разработку.
* Он очень быстрый. Платформа node.js – это среда выполнения javascript, которая использует движок V8, разработанный компанией Google для использования в браузере Google Chrome. V8 компилирует JS в нативный машинный код, что делает его быстрее чем когда-либо
* npm - пакетный менеджер node.js. – позволяет быстро и эффективно управлять модулями и зависимостями и содержит огромное количество готовых модулей
* существует множество инструментов, облегчающих разработку и отладку javascript-программ

2.3 Обоснование выбора и характеристика комплекса технических средств для реализации ИС

Для разработки веб-ориентированной версии системы была выбрана платформа Node.js - среда выполнения JavaScript, основанная на движке V8 и предназначенная для создания масштабируемых распределённых сетевых приложений, таких как веб-сервер. Так же был использован Express.js – минималистичный и гибкий Node.js фреймворк, который предоставляет широкий набор функций, избавляет от необходимо дублировать код и позволяет огранизовать веб-приложение в виде View-Model-Controller (MVC) архитектуры.

Для реализации desktop-версии приложения была использована технология Electron.

Разработка десктопных приложений является достаточно сложной задачей, так как требуется заботится о таких вещах как упаковка и установка приложения, управление обновлениями, а так же о дийзане приложения и портировании его под различные операционные системы. Поддержка различных платформ обычно предполагает использование целого набора специализированных инструментов.

Electron – это фреймворк от компании GitHub, который позволяет создавать нативные кроссплатформенные приложения с использованием исключительно веб-технологий, таких как: JavaScript, HTML и CSS. Он заботится о всех сложных нюансах разработки десктопных приложений, предоставляя унифицированный интерфейс для взаимодействия с различными операционными системами и позволяя сфокусироваться на сути самого приложения. В добавок к своему собственному набору API, Electron включает в себя весь набор Chromium API, все встроенные модули Node.js и полностью поддерживает модули третьих лиц. Electron использует веб-страницы для создания пользовательского интерфейса, что снимает присущие десктопным приложениям ограничения в дизайне. В данном случае разработчики не ограничены системным набором элементов интерфейса конкретной платформы и могут создавать UI с использованием огромных возможностей HTML5 и CSS3. А учитывая, что данный фреймворк использует один из самых продвинутых браузеров, мы можем использовать самые новые веб-технологии. Кроме того, выбор данной технологии обусловлен так же тем, что дальнейшее портирование десктопного приложения на веб-сервер Node.js займет минимум затрат по причине того, что Electon сам использует Node.js для работы с десктопной средой.

Для реализации клиентской части как десктопного так и серверного приложения был выбран фреймворк AngularJS. AngularJS - это производительный и гибкий open source фреймворк для создания веб-приложений. AngularJS предназначен для построения single-page веб приложений и является набором JavaScript функций для организации кода на стороне клиента. В основе Angular JS лежит шаблон проектирования Model View Controller, что дает ряд преимуществ при разработке и тестировании приложения. Фреймворк адаптирует и расширяет традиционный HTML, чтобы обеспечить двустороннюю привязку данных для динамического контента, что позволяет автоматически синхронизировать модель и представление. В результате AngularJS уменьшает роль DOM-манипуляций и улучшает тестируемость.

В качестве среды разработки был выбран Atom – открытый, кроссплатформенный текстовый редактор от компании GitHub, расширяемый плагинами, написанными на Node.js. Редактор предоставляет средства кросс-платформенного редактирования кода, включает встроенный пакетный менеджер и интерфейс навигации по файловой системе, предоставляет средства для одновременной совместной работы с кодом, обладает интеллектуальной системой автодополнения ввода, предоставляет режимы совместимости с Vim и Emacs. Несколько файлов могут быть открыты в разных вкладках и одновременно отображены с использованием вертикального или горизонтального разбиения панелей. Интерфейс может настраиваться через темы оформления, поддерживаются вкладки, закладки, умный контекстный поиск кода, схлопывание блоков кода, одновременное использование нескольких курсоров и областей выделения, наглядная пометка изменений, автодополнение и проверка кода для разных языков (Ruby, Python, SQL, PHP, Perl, Objective-C, C/C++, JavaScript, Java, Go и т.п.). Для формирования статей и документации может быть использована разметка Markdown. Функциональность редактора формируется за счёт предоставления набора пакетов-дополнений, для установки которых предлагается встроенный пакетный менеджер «apm», похожий на «npm» от проекта Node.js. Формат пакетов аналогичен npm и отличается предоставлением некоторых дополнительных блоков для определения меню, стилей, клавиатурных комбинаций, задания логики активации. Разработка дополнения мало чем отличается от создания приложения для Node.js, в том числе доступны все модули Node.js, а также популярные JavaScript-библиотеки, такие как jQuery, Underscore и SpacePen. Через дополнения реализованы все функции, выходящие за рамки базового редактирования кода, в том числе панели, подсветка синтаксиса, оформление интерфейса, формы работы с файлами и т.п. Кроме базовых дополнений предоставляется каталог сторонних пакетов, в котором уже присутствует более двух тысяч дополнений и почти семьсот тем оформления. Стоит отметить, что Atom так же был разработан на технологии Electron, а Electron ранее носил название Atom-Shell.

Для автоматизации сборки проекта был использован Gulp. Gulp – это быстрый потоковый Javascript task-менеджер, написанный на Node.js. Он используется для автоматизации скучных и рутинных, но, от того, не менее важных, задач, которые приходится постоянно выполнять в процессе разработки проекта. Такие задачи включают в себя, к примеру, отслеживание изменений в файлах проекта и повторный запуск связанных с этими файлами задач, таких как запуск модульных тестов, линтеров, конкатенацию файлов, минификацию, препроцессинг CSS. Просто создав task-файл, можно проинструктировать task-менеджер, каким образом следует выполнить ту или иную задачу. Довольно простая идея, которая позволяет сэкономить очень много времени, и помогает сохранять фокус на задачах, связанных непосредственно с разработкой проекта.

Все компоненты выбранного комплекса программных средств распространяются под свободными лицензиями, таким образом, разработанная система «Джарвис» была реализована без использования дорогостоящего проприетарного программного обеспечения.

ВЫВОДЫ

Когда голосовой помощник в полной мере станет понимать естественный язык?

Для этого, конечно, потребуется время. Развитие голосовых помощников, в этом плане, станет постоянным процессом, так как любой язык, являясь живой материей, приносит все новые слова и парадигмы, которым нужно время для интеграции. Необходимо также, чтобы технологии распознавания акцентов работали как надо.

Я задумался над тем, чтобы сделать для себя RSS-клиент с обучаемым фильтром статей на NodeJS. В принципе, под ноду есть готовые RSS ридеры, есть готовые нейронные сети с классификаторами, так что написать какой-то прототип мне показалось не особенно сложной задачей.

В конечном счете, после апробирования реализаций более 15 моделей под Node.js, включая многослойные нейронные сети на примере BrainJS и Node-fann, в результате кросс-проверки на небольшой выборке данных, лучший результат показала библиотека Natural, которая использует обычный наивный байесовский классификатор с корнями слов и POS-тегами, дав порядка 85%.

<http://speech-soft.ru/info/raspoznavanie-golosa> (продолжение)

Нерешенные задачи и взгляд в будущее.  
Три основных барьера стоят на пути развития систем распознавание речи:

1. большие объемы словарей  
2. шаблоны непрерывной речи  
3. различные акценты и произношения

Хотя в этой области и достигнуты существенные успехи, тем не менее, системы распознавания еще весьма далеки по своим возможностям от человеческих.

#### Бум приложений по распознаванию речи указывает на то, что время распознавания речи пришло, и мы можем ожидать огромного количества их и в будущем. Эти приложения не только позволят контролировать компьютер с помощью голоса или конвертировать голос в текст - они будут также способны различать разные языки, позволят выбирать голос помощника из различных вариантов. Вполне вероятно, что технология распознавания речи перейдет и на другие типы девайсов. Нетрудно представить, как в будущем мы будем управлять кофеварками, разговаривать с принтерами и говорить освещению, чтобы оно выключалось.

СПИСОК ЛИТЕРАТУРЫ

[1] K. H. Davis, R. Biddulph and S. Balashek, “Automatic recognition of spoken digits”, J. Acoust. Soc. Am., 24, pp. 637-642 (1952).

[2] J. Suzuki and K. Nakata, “Recognition of Japanese vowels – preliminary to the recognition of speech”, J. Radio Res. Lab, 37, pp. 193-212 (1961).