Московский государственный университет им. М. В. Ломоносова
Факультет вычислительной математики и кибернетики
Кафедра алгоритмических языков

Курсовая работа

Применение метода опорных векторов для анализа речевых команд в приложении-помощнике для мобильных устройств

Выполнил: студент 425 группы
Николай Иванов
Научный руководитель: к.ф.-м.н., доцент
Бордаченкова Е.А.

Москва, 2013

Содержание

1. Введение и постановка задачи

Интеллектуальный персональный помощник (англ. Intelligent Personal Assistant) [1] – это приложение для мобильных устройств или персональных компьютеров, основной задачей которого является помощь человеку в использовании различных функций данного мобильного устройства или ПК.

Далее рассматриваются интеллектуальные персональные помощники только для мобильных устройств и они будут коротко называться “интеллектуальные помощники” или “приложения-помощники”.

К типичным действиям, которые выполняют интеллектуальные помощники, относятся:

- телефонный вызов,
- отправка sms и электронных писем,
- поиск информации в интернете,
- добавление напоминаний о предстоящих событиях.

Отличительной особенностью интеллектуальных помощников является выполнение голосовых команд пользователя, что является альтернативой традиционному использованию кнопок и сенсорного экрана мобильного устройства. Во многих ситуациях голосовые команды оказываются существенно более удобным и быстрым способом управления мобильным устройством.

Самым известным интеллектуальным помощником является приложение Siri для смартфонов iPhone [2]. Обзор Siri и других интеллектуальных помощников, а также описание общих принципов их работы можно найти в [3].

В настоящее время почти все интеллектуальные помощники ориентированы на работу с английским языком, и только несколько приложений поддерживают русскоязычные голосовые команды [7].

Чтобы найти новые методы анализа пользовательских команд на русском языке, была поставлена задача создать собственный русскоязычный интеллектуальный помощник. К настоящему моменту разработан прототип, в котором поддерживается работа с семью различными типами голосовых команд.

В рамках курсовой работы подробно рассматривается реализация одного из модулей данного приложения, в задачи которого входит извлечение нужной информации из пользовательской команды. Входными данными для этого модуля является текстовая форма команды пользователя, которая получена в результате распознавания голосовой команды специальным сервисом распознавания речи.

Понятие «извлечение информации» поясним примером.

Из текста пользовательской команды

«отправь смс васе с текстом перезвони мне пожалуйста»

необходимо определить:

- тип команды – отправка смс-сообщения
- параметры команды:
  - адресат – Вася
  - текст смс – «перезвони мне пожалуйста»

При анализе пользовательских команды возникает ряд сложностей, одной из которых является большое разнообразие различных формулировок команд.

Так для команды отправки сообщения из приведенного выше примера возможна следующая формулировка:

«давай теперь напишем смс сообщение василию ивановичу со словами перезвони мне пожалуйста» .

Для практической применимости интеллектуального помощника используемые в нем методы анализа должны быть пригодны для достаточно большого числа различных формулировок пользовательских команд.

Анализ пользовательских команд усложняется из-за наличия различного рода ошибок при распознавании речи.

К примеру, результатом распознавания голосовой команды

«Позвони Васе»

может быть строка

«звонить в асе» .

Небольшие ошибки в распознавании речи (как, например, неправильный падеж слова

при сохранении общего смысла слова) не должны мешать интеллектуальному помощнику выполнять запрашиваемые действия.

Кроме того, отдельной нетривиальной задачей является выделение из текста пользовательской команды последовательности слов, которая должна быть интерпретирована как единая сущность.

В примерах ниже такие последовательности выделены жирным шрифтом:

- «отправь смс васе с текстом **перезвони мне пожалуйста**»
- «позвони на номер **плюс 7 917 123 45 ноль один**»
- «покажи статью про **теорию относительности** в википедии»

В данной работе при анализе пользовательских команд предлагается использовать метод опорных векторов (далее SVM) [4], поскольку с его помощью удается довольно эффективно организовать извлечение нужной информации из текстов пользовательских команд, а также в целом упрощается решение перечисленных выше проблем.

Таким образом, в рамках данной курсовой работы были поставлены следующие задачи:

1. реализовать модуль анализа пользовательских команд, применив SVM для извлечения нужной информации;
2. оценить качество работы данного модуля на имеющихся данных.

В разделе «Проект интеллектуального помощника» описана текущая версия приложения. Далее в разделе «Анализ пользовательских команд при помощи SVM» подробно рассматривается применение SVM для определения типа команды и ее параметров. Работа завершается оценкой качества работы модуля-анализатора.

2. Проект мобильного помощника

Прежде чем переходить к деталям реализации модуля-анализатора, рассмотрим текущую версию разрабатываемого интеллектуального помощника.

2.1 Функциональность приложения

Интеллектуальный помощник запускается на мобильном устройстве путем нажатия на иконку данного приложения. Типичный сценарий обращения к некоторой функции телефона с помощью интеллектуального помощника следующий:

1. пользователь инициирует процесс распознавания речи, нажимая в приложении на специально предназначенную для этого кнопку, и произносит команду;
2. записанная речевая команда пересылается через интернет-соединение сервису распознавания речи, который приводит данную речевую команду к текстовому виду, и полученный результат высылается обратно;
3. специальный модуль интеллектуального помощника анализирует текстовое представление команды, определяя ее тип и параметры;
4. интеллектуальный помощник запускает функцию операционной системы телефона, соответствующую данному типу команды и параметрам, и выводит на экран результаты ее работы.

Ниже в Таблице 1 перечислены поддерживаемые функции текущей версии приложения, а также соответствующие им примеры пользовательских команд:

тип	**пользовательская команда**
позвонить	«позвони васе»
отправить sms	«отправь смс васе с текстом привет»
проверить баланс	«покажи баланс
написать email	«напиши письмо васе с текстом как жизнь»
открыть сайт	«открой википедию»
искать в интернете	«найди в яндексе что значит дзен»
поставить будильник	«поставь на завтра будильник на 7 30»

Таблица 1. Типы пользовательских команд с примерами

2.2 Требования к приложению

На этапе проектирования к приложению были предъявлены следующие требования:

1. Устойчивость.
   Небольшие неточности распознавания речи не должны сильно влиять на качество работы приложения.

   Например, результатом распознавания фразы
   «Найди мне в википедии значение слова «интерференция» »
   может быть
   «найди мне википедия значение слова интерференция» ,
   однако такие несущественные ошибки распознавания как неправильный падеж или отсутствие малоинформативных слов, не должно помешать приложению выполнить запрашиваемое действие.

2. Практическая применимость.
   Метод анализа пользовательских команд, используемый в приложении интеллектуального помощника, должен быть применимым к достаточно большому набору различных формулировок пользовательских команд.

   Так, например, интеллектуальный помощник должен корректно работать с каждой из следующих команд телефонного вызова:

   - позвони васе
   - набери васю пожалуйста
   - немедленно соедини меня с василием ивановичем

3. Языковая расширяемость.
   Добавление в приложение поддержки новых формулировок команд не должно требовать значительных изменений в существующих структурах приложения.

   Если, к примеру, для команды телефонного вызова потребуется добавить формулировку «вызываю васю», которая до этого не поддерживалась, то для этого нужно будет всего лишь добавить слово «вызываю» в соответствующий список ключевых слов.

4. Функциональная расширяемость.
   Добавление в приложение новых типов пользовательских команд не должно требовать значительного изменения его архитектуры.

   Например, если потребуется добавить команду «показать фотографию», то, возможно, это потребует создания нового модуля, который будет отвечать за взаимодействие с базой данных фотографий, но больших изменений в уже существующих модулях приложения не потребуется.

5. Обучаемость.
   В процессе работы приложение должно запоминать и накапливать полезную информацию из пользовательских команд (к примеру, частые ключевые слова, имена и команды), чтобы впоследствии эти данные могли быть проанализированы с целью улучшения качества работы приложения.

2.3 Интерфейс приложения

Ниже на Рис. 1 показан общий вид графического интерфейса текущей версии интеллектуального помощника с описанием его основных элементов:

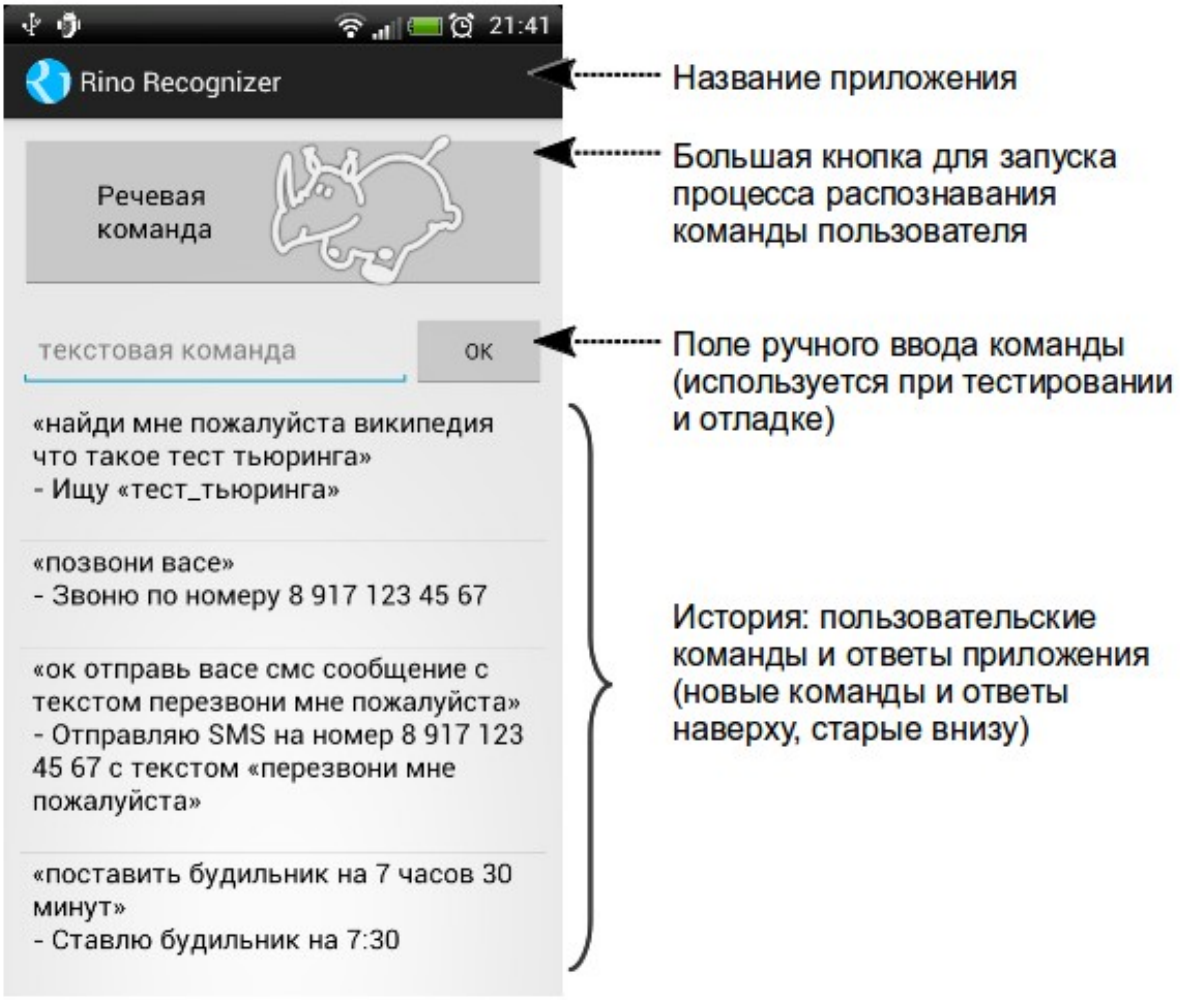

Рис. 1. Интерфейс интеллектуального помощника

При запуске приложения на экране мобильного устройства появляется главное окно интеллектуального помощника, подобное тому, что показано выше. Для голосового ввода команды пользователь нажимает большую кнопку с надписью «Речевая команда». Также интеллектуальный помощник предоставляет возможность текстового ввода команды с помощью соответствующего поля (в настоящее время эта функция используется для отладки приложения-помощника). Ниже располагается список, отображающий историю работы с приложением: каждый пункт содержит пользовательский запрос и ответ на него приложения-помощника.

2.4 Архитектура приложения

С точки зрения логической организации приложение состоит из нескольких частей: главный модуль (организует работу приложения в целом), модуль графического интерфейса, анализатор пользовательских команд и модуль, отвечающий за запуск функций телефона. Также в приложении есть специальные модули для работы с двумя базами данных: база истории и база частых контактов.

Первая база данных хранит историю пользовательских команд и ответов на них приложения-помощника. Эти данные можно использовать для анализа работы приложения с целью его дальнейшего улучшения.

Вторая база данных содержит наиболее частые контакты пользователя, записанные в специальном формате. Наличие такой базы данных увеличивает эффективность работы приложения, поскольку хранятся только частые контакты пользователей и, кроме того, о каждом контакте хранится не полная, а только необходимая приложению информация, что увеличивает скорость поиска.

Приложение-помощник использует облачный сервис распознавания речи от Google: при вызове специальной функции операционной системы Android начинается запись речи, затем полученный аудио-файл пересылается через интернет-соединение на сервера Google, где производится процесс распознавания. Приложение-помощник получает результат распознавания речевой команды в виде текстовой строки.

Общий вид архитектуры приложения показан на Рисунке 2.

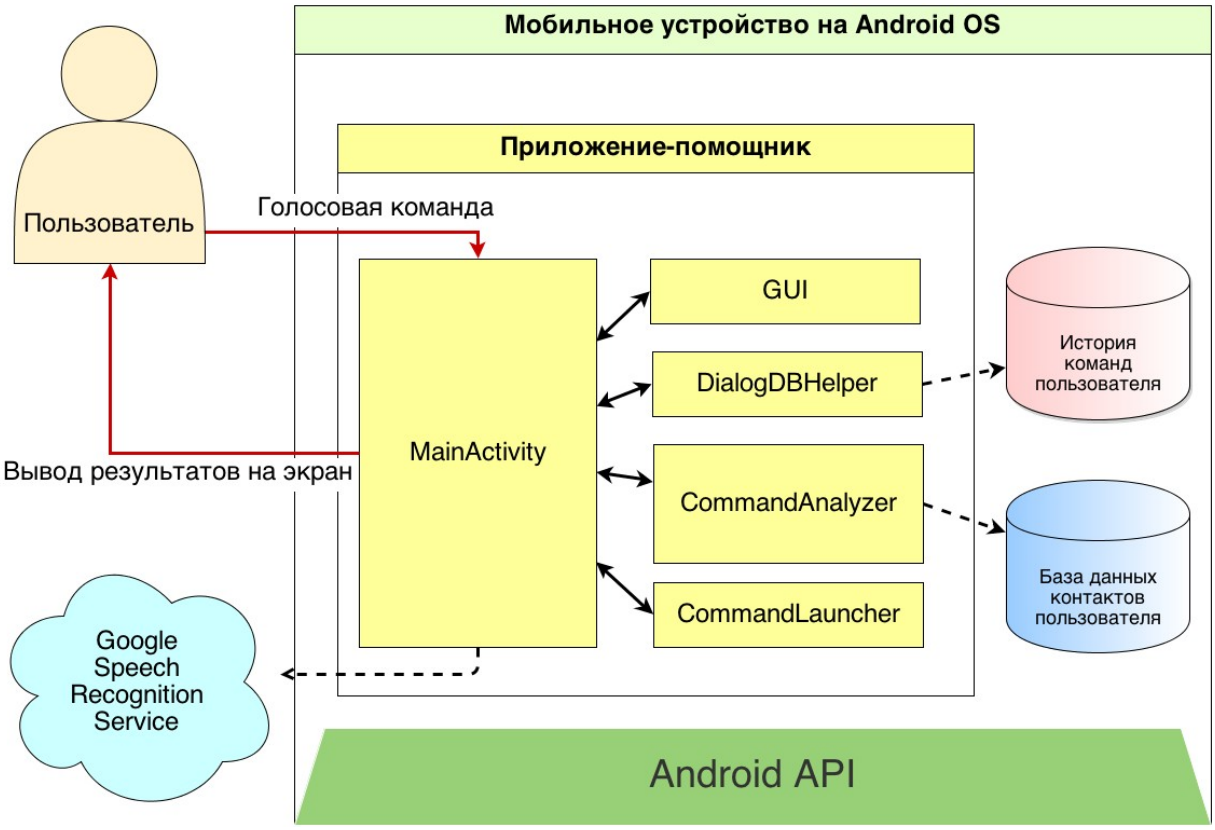

Рис. 2. Архитектура приложения

Ниже перечислены основные модули приложения с их описанием:

MainActivity – запускает основной цикл программы и организует работу остальных модулей;

GUI – задает графический интерфейс приложения;

HistoryDBHelper – загружает историю пользовательских команд из базы данных в приложение и периодически сохраняет текущую историю;

CommandAnalyser – анализирует пользовательские команды и формирует специальный объект Intent для последующего запуска одной из функций телефона. Реализация данного модуля будет подробно рассмотрена в следующей главе;

CommandLauncher – запускает определенную функцию телефона по сформированному объекту Intent.

2.5 Спецификация приложения

В качестве платформы для приложения-помощника выступает операционная система Android. Ее выбор обусловлен следующими факторами: данная ОС хорошо документирована, в ней есть встроенные средства распознавания речи и, кроме того, немаловажную роль в выборе системы сыграло наличие мобильного устройства под управлением Android OS для тестирования и отладки приложения.

Язык разработки под Android OS – Java.

Среда разработки – Android Developer Tools.

Исходные коды приложения можно найти в [8].

3. Анализ пользовательских команд с помощью SVM

После ознакомления с текущей версией приложения перейдем к более детальному рассмотрению реализации его главного модуля, отвечающего за определение типа и параметров команды. Далее для краткости данный модуль мы будем называть просто «анализатор».

Под термином «пользовательская команда» мы будем понимать текстовую строку, полученную как результат распознавания речевой команды.

Пример: «отправь смс васе с текстом перезвони»

У пользовательских команд есть особенности, которые стоит учитывать при анализе:

- все слова пишутся строчными буквами и разделяются пробелами
- знаки препинания отсутствуют (исключением являются случаи, когда соответствующий знак является частью слова, например, точка в «yandex.ru»)

3.1 Общая схема анализа команд

Каждой реализованной функции мобильного устройства в приложении-помощнике сопоставлен фрейм команды. Для запуска функции устройства необходимо выбрать фрейм команды и заполнить его поля.

На вход анализатора команд поступает текстовое представление распознанной речевой команды. В процессе анализа нужно определить тип желаемого действия и необходимые параметры для выбора и заполнения фрейма (см. Рис. 2).

Рис. 3: Заполнение фрейма "Отправка СМС"

Таким образом, работу анализатора можно разделить на три логических этапа:

1. определение типа команды
2. определение параметров команды
3. заполнение фрейма команды

При реализации первых двух этапов в данной работе предлагается использовать метод SVM. Как будет показано ниже, с помощью этого метода можно довольно просто классифицировать пользовательские команды по типам и слова команд по типам параметров. Также, применяя SVM, удается повысить устойчивость анализа команд к некоторым ошибкам распознавания речи, например, если какое-либо малоинформативное слово было распознано неправильно или же не было распознано совсем. Наконец, метод SVM имеет хорошую программную реализацию в виде библиотеки libsvm, описание которой можно найти в [5].

Далее рассмотрим процесс обучения моделей классификации и их последующее применение для определения типа команды и ее параметров.

3.2 Определение типа команды

Построение вектора признаков

Как уже говорилось ранее, в текущей версии приложения поддерживается 7 типов команд:

1. позвонить
2. отправить sms
3. проверить баланс
4. написать email
5. открыть сайт
6. искать в интернете
7. поставить будильник

На наш взгляд, наиболее естественным способом определения типа команды является поиск в ней ключевых слов, по которым далее будут строится векторы признаков для SVM. Например, для команды отправки смс ключевыми могут быть слова: «смс», «sms», «отправить», «текст».

В результате анализа примеров пользовательских запросов, собранных в работе [3], для каждого типа команды были найдены соответствующие им ключевые слова. Они хранятся в специальном текстовом файле «keywords» (см. Приложение А).

Для описания ключевых слов используются регулярные выражения, поскольку с их помощью очень просто описываются различные формы слова. Например, шаблону регулярного выражения `\w*звон\w+` соответствует каждое из слов «звонок», «звонить», «позвонить», «позвони», «перезвони». Таким образом, регулярные выражения избавляют нас от необходимости перечислять ключевые слова во всевозможных падежах, числах, временах и т. д.

Ниже приведен фрагмент файла «keywords», в котором за именем типа команды (см. первый столбец) следует регулярное выражение, описывающее ключевые слова для данного типа команды:

```
a_call      \w*звон\w+|набрать|набер\w+|набир\w+|соедин\w+|...
a_sms       смс\w*|sms|сообщение|смс|напиш\w+|напис\w+|...
a_email     e?mail|письмо|напиш\w+|напис\w+|набрать|...
a_search    найди|найти|ищи|искать|покаж\w+|показ\w*|...
a_site      покаж\w+|откр\w+
a_alarm     будильник|постав\w+|установ\w+
a_balance   баланс|покаж\w+|показ\w*|\w*смотр\w*|узна\w+|...
...
```

(приставка «a_» в обозначениях типов команд от слова «action»)

Для того, чтобы можно было применять метод SVM для классификации команд по типам, каждой пользовательской команде ставится в соответствие вектор признаков, i-ая компонента которого показывает, сколько раз в данной команде встретилось слово из i-ого множества ключевых слов. Например, для команды

«напиши смс сообщение»

начало вектора признаков будет выглядеть как

```
[ 0, 3, 1, 0, … ] ,
```

поскольку в множестве a_call нет ни одного слова из данной команды, в a_sms таких слов три («напиши», «смс» и «сообщение), в a_email – одно («напиши»), в множестве a_search ни одного.

Кроме множеств ключевых слов, непосредственно указывающих на тип команды, в том же файле описываются аналогичные множества для параметров команд (приставка «p_» от слова «parameter»):

```
p_name      Имя|Фамилия|Отчество|мам\w+|вас\w*|леш\w*|...
p_number    Номер|\d+|\+|плюс
p_email     Адрес|[^S]+@[^S]+|e?mail|письмо
p_site      Сайт|\w+\.+ru|com|org|net|su
p_time      Время|\d{1,2}|утр\w+|вечер\w*|час\w*|минут\w*|...
```

Замечание:

такие слова, как «Имя», «Номер» и другие, написанные в данном фрагменте с большой буквы, используются при обучении и отладке в качестве одного из возможных значений параметра, чтобы не приходилось придумывать реальные имена, номера телефонов, электронные адреса и т. д.

Также в файле «keywords» описываются множества слов, которые помогают при выделении текстов смс и поисковых запросов в командах (приставка «s_» от слова «sign»):

```
s_text      текст\w*
s_search    такое|значит|про|о|слов\w*|поняти*|...
```

Далее будем называть такие слова «маркерами», а тексты смс и тексты поисковых запросов – «цитатами». Например, для команды

«напиши васе смс с текстом привет как дела»

слово «текстом» будет являться маркером. Оно указывает на то, что, возможно, следующее слово является началом некоторой цитаты. В данном случае цитатой является текст смс сообщения «привет как дела».

Параметры команд и слова-маркеры косвенно указывают на тип команды, например, если в команде встретилось имя-параметр, то типом команды скорее будет телефонный вызов, чем проверка баланса, поэтому данные о вхождении слов в множества, описывающие параметры команд и слова-маркеры, также имеет смысл включать в вектор признаков.

Ниже приведен пример вектора признаков для команды

«напиши смс сообщение василию ивановичу с текстом здравствуйте»:

`[ 0, 3, 1, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 1, 0 ].`

В [6] для улучшения результатов классификации рекомендуется «масштабировать» вектора, т. е. приводить все компоненты вектора к одному и тому же диапазону, например, к [0..1]. Для мастабирования i-ого компонента вектора необходимо найти его максимальное и минимальное значения среди всех всех векторов обучающей выборки и далее вычислить его значение по формуле:

```
v[i] = (v[i] – min v[i]) / (max v[i] – min v[i])
```

После масштабирования вектор признаков для нашего примера будет выглядеть так:

```
[ 0, 0.75, 0.25, 0, 0, 0, 0, 0.67, 0, 0, 0, 0, 0, 1, 0 ].
```

Обучение и применение модели

Перед тем как применять SVM для классификации пользовательских команд по типам, нужно получить модель, обученную на заранее размеченных данных. С этой целью было собрано и размечено в полуавтоматическом режиме примерно 400 экземпляров пользовательских команд:

```
a_sms       отправь смс с текстом _напиши _мне
a_sms       перешли письмо васе с текстом _почему _не _звонишь
...
a_search    найди в интернете что такое _смс
a_search    посмотри определение слова _дзен на википедии
...
a_call      пожалуйста позвони леше
...
```

Более полный список размеченных команд приведен в Приложении Б.

На этих примерах видно, что каждой пользовательской команде предшествует метка ее типа, и, кроме того, слова, которые являются частью цитат, начинаются со знака нижнего подчеркивания. Была написана программа, которая по такой разметке автоматически строит для каждой пользовательской команды ее вектор признаков в формате, понятном для функций библиотеки libsvm (далее в «формате libsvm»). Подробнее работа этой программы будет рассмотрена ниже.

Для команд из предыдущего примера разметка в формате libsvm выглядит следующим

образом:

```
2 2:0.5 13:1 15:1
2 1:0.333333 3:1 8:0.5 10:0.5 13:1 15:0.5
...
4 2:0.5 4:1 14:1
4 4:0.333333 7:0.5 14:0.666667
...
1 1:0.333333 8:0.5
...
```

Здесь каждая строка начинается с кода типа команды («2» для смс, «4» для поиска в интернете, «1» для телефонных звонков), за которым следуют номера ненулевых компонент вектора и через двоеточие их значение. На таких данных можно обучать модель для классификации команд по типам путем вызова специальной функции библиотеки libsvm.

Обученная модель представляет собой текстовый файл с перечислением опорных векторов и дополнительной служебной информацией. Фрагмент модели для классификации команд по типам приведен в Приложении В.

После того, как модель обучена, ее можно применять для классификации пользовательских команд по типам. Для этого специальной функции приложения передаются на вход файл модели и пользовательская команда, а на выходе функция возвращает тип этой команды.

Пример:

вход – «скажи мне что такое дзен»

выход – a_search

При разбиении случайным образом коллекции размеченных пользовательских команд на обучающую и тестовую выборки в отношении примерно 1:1 точность классификации команд по типам составляет почти 100%.

3.3 Определение параметров команды

Построение вектора признаков

После того, как тип пользовательской команды найден, можно переходить к определению параметров команды. Для этого требуется получить более детальную разметку пользовательской команды: каждому слову пользовательской команды ставится в соответствие одна из меток, которая отражает «смысл» данного слова.

Ниже приведен список меток, используемых в приложении, где

1-ый столбец это название метки,

2-ой – текстовое описание метки,

3-ий – пример слова, к которому может быть поставлена данная метка:

action	команда	«позвони» или «сообщение»
p_name	параметр-имя	«васе»
p_number	параметр-тел.номер	«89171234567»
p_email	параметр-email	«vasya@yandex.ru»
p_site	параметр-url	«yandex.ru»
p_time	параметр-время	«7» или «часов»
quote	часть «цитаты»	слово из текста смс сообщения
q_mark	слово-маркер	«текстом»
other	второстепенное слово	«пожалуйста»

Пример разметки для пользовательской команды:

« отправь	пожалуйста	смс	васе	с	текстом	привет »
↓	↓	↓	↓	↓	↓	↓
action	other	action	p_name	other	q_mark	quote

Для реализации такой разметки также используется SVM. Поскольку наборы параметров для различных типов команд отличаются, было решено для каждого типа команды обучать собственную модель SVM, отвечающую за определение именно ее параметров.

Таким образом, для определения параметров команд используется 7 различных моделей, по одной модели для каждого из 7 типов команд.

Так же как и в случае с определением типа команды, для определения параметров команды необходимо сначала получить вектора признаков слов. При этом используется уже упоминавшийся файл «keywords», но подход к формированию вектора несколько отличается от рассмотренного ранее: если данное слово входит в i-ое множество ключевых слов, то i-ая компонента вектора равна 1, иначе – 0.

Ниже приведен фрагмент вектора признаков для слова «набери»:

```
[ 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 , ... ]
```

(«набери» входит в первые три множества файла «keywords» и не входит в остальные).

Далее такую информацию о принадлежности слова к определенному множеству ключевых слов будем называть «характеристикой» слова.

Кроме данных о вхождении в определенное множество ключевых слов, для правильной разметки необходимо учитывать контекст слова, поскольку контекст является основным способом выделения цитат из текста пользовательской команды.

Например, в команде

«отправь васе смс с текстом **набери васю**»

«набери» является частью цитаты (метка quote). Это можно понять по предыдущему слову – «текстом», которое является указателем на начало цитаты (метка q_mark). Если же не учитывать контекст этого слова, «набери» будет ошибочно размечено как «действие» (метка action).

Для учета контекста слова его вектор признаков включает характеристику не только самого слова, но и его соседей. Полный вектор признаков для слова «набери» из примера выше выглядит следующим образом:

```
[ 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ,
  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0 ,
  0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0 ]
```

(первая строка – характеристика для слова «набери», вторая – для слова «текстом», третья – для слова «васю»).

Тот же вектор в формате libsvm:

```
1:1 2:1 3:1 28:1 38:1
```

Если к этому вектору признаков дополнительно добавить информацию о правильной метке, которой он соответствует, то он будет выглядеть так:

```
-1 1:1 2:1 3:1 28:1 38:1
```

Здесь вектору признаков предшествует код «-1». Он используется нами как синоним метки «quote», поскольку формат libsvm поддерживает метки только целочисленного типа. Коды для других меток приведены в файле «keywords» в Приложении А.

Для большого числа пользовательских команд вручную правильно выписать такие вектора задача практически невыполнимая. С целью автоматизации этого процесса нами была создана специальная программа, которая для размеченных команд на основе файла «keywords» строит данные вектора.
Рассмотрим этот процесс подробнее на примере слова «набери» из размеченной команды

```
a_sms       отправь васе смс с текстом _набери _васю .
```

Видно, что словам «набери» и «васю» предшествует нижнее подчеркивание, которое указывает на то, что эти слова являются частью цитаты (метка «quote»), поэтому код метки равен «-1». Перед последующей проверкой слова на вхождение в различные множества ключевых слов знак нижнего подчеркивания удаляется.

Если перед словом нет знака нижнего подчеркивания, то его метка определяется при помощи файла «keywords» (см. Приложение А.):

- если слово не входит ни в одно множество ключевых слов, его метка — «other»,
- иначе, если слово входит ровно в одно множество ключевых слов, то код его метки равен значению из второго столбца соответствующей строки.

  Замечание:

  множества ключевых слов с разными кодами составлены так, что они не пересекаются, поэтому случай, когда слово входит в более чем одно множество ключевых слов, не рассматривается.

Таким образом, для слов «текстом», «_набери» и «_васю» из нашего примера будут найдены коды меток «-2» (синоним метки «q_mark»), «-1» и «-1» соответственно.

Далее, согласно процедуре, описанной выше, для каждого слова команды вычисляется

его характеристика:

```
для слова «текстом»     –     13:1
для слова «набери»      –     1:1 2:1 3:1
для слова «васю»        –     8:1
```

Добавление в вектор признаков слова характеристик соседних слов, осуществляется во время дополнительного прохода по словам команды, при этом для слов-соседей пересчитываются индексы признаков:

```
для слова «набери»      –     1:1 2:1 3:1 28:1 38:1
```

(количество признаков в характеристике слова равно 15, 15 + 13 = 28, 15 + 15 + 8 = 38).

Для первого и последнего слов команды характеристиками соответственного предыдущего и последующего слова является вектор длины 15, все компоненты которого равны 0.

Итого, для слова «_набери» мы получим следующий размеченный вектор признаков с кодом метки параметра:

```
-1 1:1 2:1 3:1 28:1 38:1
```

Обучение и применение модели

Наборы таких векторов для различных триграмм слов из пользовательских команд вместе составляют обучающую выборку для модели классификации параметров этих команд.

Если, к примеру, нам нужно обучить модель для разметки команд типа «отправить смс», то для получения размеченных векторов, естественно, нужно использовать только размеченные пользовательские команды того же типа. Их легко выделить из общего файла со всеми размеченными командами, поскольку для каждой команды ее тип указан в первом столбце:

```
...
a_sms       перешли письмо васе с текстом _почему _не _звонишь
...
```

В остальном процесс обучения моделей ничем не отличается от описанного выше случая для классификации команд по типам.

Для того, чтобы при помощи обученной модели разметить слова из пользовательской

команды, необходимо сначала представить эту команду в виде списка слов. Далее этот список вместе с информацией о типе команды передается на вход специальной функции приложения, которая в качестве результата возвращает список меток, соответствующий словам данной команды.

Пример:

слова (вход)	–	(«позвони», «пожалуйста», «василию», «ивановичу»)
метки (выход)	–	(action, other, p_name, p_name)

Ниже приведены оценки точности разметки для различных типов команд:

a_call	(телефонный вызов)	– 100%
a_sms	(отправка смс)	– 96.9%
a_email	(отправка email)	– 95.2%
a_search	(поиск в интернете)	– 98.5%
a_site	(отображение сайта в браузере)	– 100%
a_alarm	(установка будильника)	– 98.7%
a_balance	(проверка баланса)	– 95.9%

Подробнее вопрос об оценке качества работы классификатора будет рассмотрен в следующей главе.

3.4 Выделение семантических групп слов

Прежде чем переходить к заполнению фрейма, необходимо произвести еще одно действие над списками слов и меток: слова, которые стоят рядом и размечены одинаково, нужно объединить в одну группу.

Для предыдущего примера списки слов и меток тогда будут выглядеть так:

группы слов (вход)	–	(«позвони», «пожалуйста», «василию ивановичу»)
метки (выход)	–	(action, other, p_name)

Анализ собранных пользовательских команд, показал, что такое объединение корректно почти всегда. То есть, если соседним словам сопоставлены одинаковые метки, то с большой вероятностью они принадлежат одной смысловой группе (слова «василию» и «ивановичу» из пользовательского запроса выше).

Также приведем пример, когда такое объединение слов в группы все-таки не будет являться корректным:

«отправь письмо с текстом привет друзья **васе саше** и **диме**».

Если действовать по описанному выше алгоритму группировки слов, то «вася» и «саша» будут объединены в одну группу с общей меткой p_name, что по смыслу не является правильным, поскольку эти слова соответствуют разным контактам.

Однако, такие сложные случаи мы в данной работе рассматривать не будем. Как уже говорилось, описанный алгоритм объединения слов в группы корректен почти всегда, и он существенно облегчает работу по заполнению фрейма на следующем этапе.

3.5 Заполнение фрейма команды

После того, как определен тип команды и ее параметры и произведена группировка слов, можно переходить к заполнению фрейма команды.

Концепция фреймов реализована в приложении с помощью классов: так для каждой поддерживаемой функции мобильного устройства существует свой класс, в котором свойства класса соответствуют параметрам команды, а методы отвечают за заполнение данного фрейма (класса) и проверку корректности его данных.

Ниже приведен интерфейс класса SmsFrame (параметры методов опущены):

```
private class SmsFrame {
      private Uri numUri;
      private String text;

      public SmsFrame();
      public void fill();
      private boolean check();
}
```

Рассмотрим подробнее функцию заполнения фрейма. Ее общий вид для всех фреймов следующий:

```
public Intent fill(List<String> wgroups, List<ParamsType> labels)
```

Функция fill() получает на вход два согласованных списка — список сгруппированных слов, и список меток. Далее последовательно просматриваются все метки из списка, и если метка указывает на то, что соответствующая ей группа слов является параметром для команды данного типа, значение этой группы принимается за значение соответствующего слота фрейма.

Пример:

Вход функции fill() фрейма SmsFrame:

группы слов – («отправь смс», «василию ивановичу», «с», «текстом», «привет вася»)

метки – (action, p_name, other, q_mark, quote)

У команды «отправить смс» есть два параметра – телефонный номер адресата и текст сообщения. Телефонный номер может быть определен как по текстовому представлению номера, так и по имени контакта, поэтому функция fill() для определения телефонного номера ищет группы слов, которым соответствуют метки «p_number» или «p_name». Аналогично для определения текста смс данная функция ищет группу слов с меткой «quote».

Для нашего примера строка «привет вася» является цитатой и поэтому не нуждается в дополнительном анализе. Ее можно без каких-либо изменений присвоить слоту text нашего фрейма. Что касается параметра с меткой имя, для него необходим дополнительный этап, на котором определяется телефон контакта:

1. строка «василию ивановичу» передается специальной функции, которая находит в базе контактов телефона наиболее схожее имя
2. для этого имени определяется основной телефонный номер контакта
3. полученный телефонный номер возвращается как результат работы функции и присваивается слоту numUri.

Возможно, функции fill() не удастся заполнить некоторые слоты фрейма из-за того, что нужная метка не встретилась в списке, или же строковое значение параметра не удалось привести к нужному виду (например, к виду телефонного номера), и тогда получившийся фрейм будет некорректен. В этом случае приложение выведет на экран сообщение «Телефон не может выполнить это действие».

Если все данные фрейма корректны, функция fill() инициализирует объект Intent, отвечающий за запуск определенной функциональности телефона и передает его главному модулю приложения, которое уже и вызывает нужное действие телефона с помощью данного объекта. Действие выполняется и результаты работы выводятся на экран.

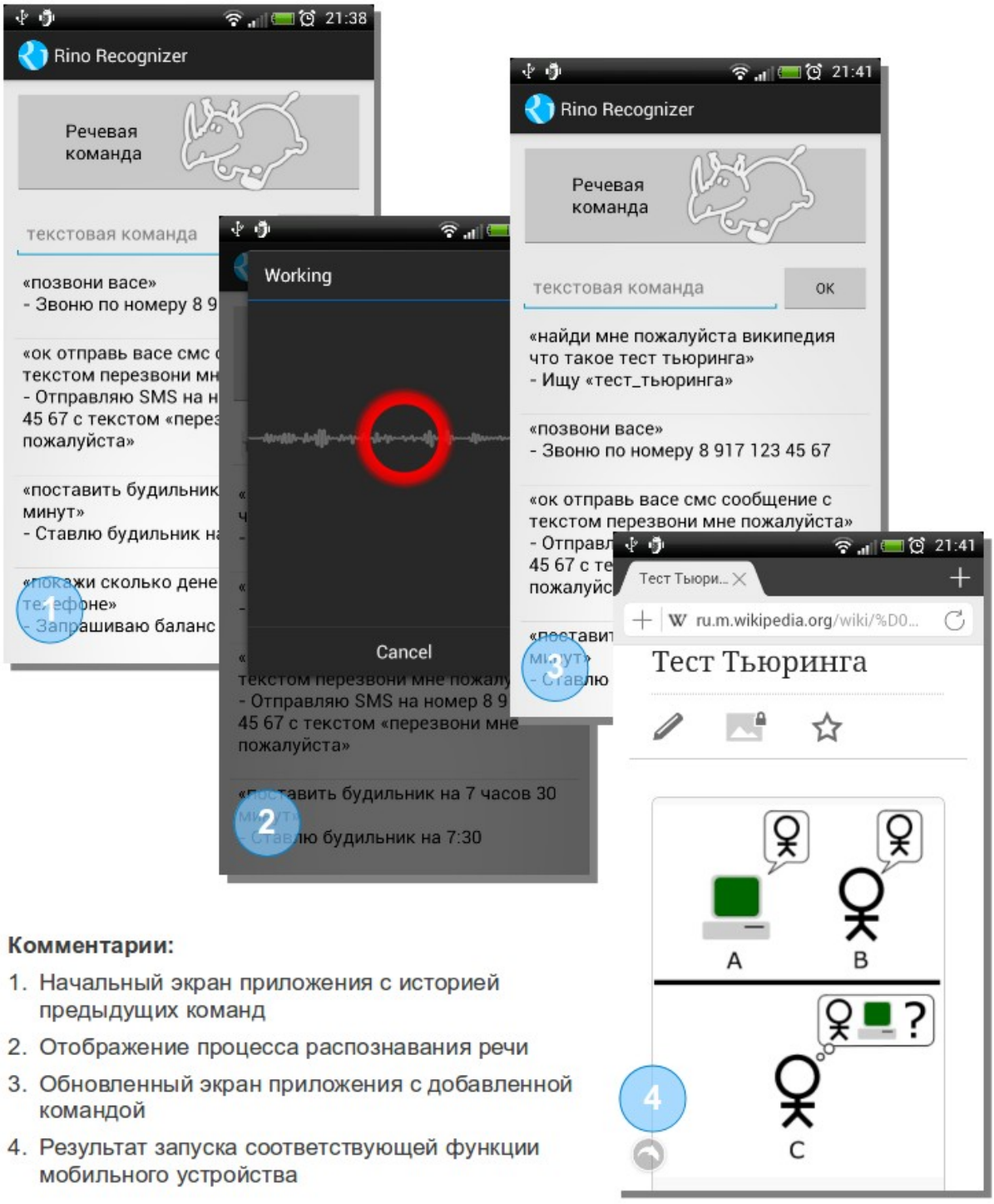

Рис. 4. Демонстрация работы приложения на примере поиска информации в интернете

5. Оценка качества работы анализатора

Для оценки точности классификации вся коллекция, состоящая из 400 размеченных пользовательских команд, случайным образом разбивалась на обучающую и тестовую выборки. Точность классификации вычислялась для следующих 3 разбиений:

обучение:	25%	50%	80%
тестирование:	75%	50%	20%

Далее эти разбиения будут соответственно обозначаться как «25/75», «50/50», «80/20».

Полученные результаты точности классификации команд и их параметров по типам для каждого данных разбиений приведены в Таблице 2:

	«25/75»	**«50/50»**	**«80/20»**
action	89.2%	96.7%	97.4%
a_call	100%	100%	100%
a_sms	92.8%	95.7%	96.7%
a_email	94.2%	93.6%	94.6%
a_search	99.3%	98.9%	100%
a_site	84.6%	100%	100%
a_alarm	93.9%	98.5%	100%
a_balance	97.5%	98%	100%
Average	**94.0%**	**97.7%**	**98.6%**

Таблица 2. Оценки точности классификации

Как и следовало ожидать, чем больше доля обучающей выборки по отношению к тестовой, тем выше точность классификации (это видно из результатов точности для первых трех разбиений).

Следует также отметить, что даже для разбиения «25/75», где обучение проводилось на выборке, состоящей всего лишь из 100 команд, что в среднем составляет 15 команд для каждого типа, получены довольно неплохие результаты точности — в среднем 94%. Это

говорит о том, что для достижения приемлемого качества классификации с помощью предложенного метода достаточно относительно небольшого числа размеченных примеров пользовательских команд.

В случае, когда потребуется добавить в приложение поддержку нового типа команды, существующую коллекцию размеченных пользовательских команд необходимо будет пополнить примерно 15-ю размеченными командами нового типа, а также привязать этот тип команды к определенной функции операционной системы, что не потребует больших усилий со стороны разработчика.

Таким образом, благодаря применению метода анализа пользовательских команд на основе SVM, удается выполнить требования языковой и функциональной расширяемости приложения, которые упоминались в главе 2.

Для того, чтобы оценить качество не только классификатора, но и самой коллекции размеченных команд, проводилось тестирование на всех имеющихся данных коллекции: в случае классификации команд по типам обучающая и тестовая выборки включали в себя все имеющиеся размеченные команды, а для классификации параметров — все команды соответствующего типа (Таблица 3):

	«100/100»
action	99.7%
a_call	100%
a_sms	96.9%
a_email	98.2%
a_search	100%
a_site	100%
a_alarm	100%
a_balance	100%
Average	**99.4%**

Таблица 3: Точность классификации параметров в случае обучения и тестирования на всей коллекции размеченных команд

Тот факт, что для некоторых типов команд не достигнута стопроцентная точность, говорит о не полной разделимости соответствующих тестовых выборок на классы. Это означает, что в тестовой выборке есть такие вектора признаков, которые очень похожи друг

на друга (или вовсе одинаковы), но при этом соответствуют разным типам команд или параметров.
Рассмотрим пример задачи определения параметров команды для отправки смс:

1-ая команда:	отправь смс на номер 8 917 123 45 67
2-ая команда:	напиши смс с текстом **позвони на номер 8 917 123 45 67**

Необходимо определить тип параметра для слова «номер». В первом случае тип параметра — action, во втором — quote.

Вектора признаков для слова «номер» в этих двух случаях будут выглядеть абсолютно одинаково, поскольку у этих слов одинаковый контекст — слова «на» и «8», а значит и классифицироваться они будут одинаково. Как следствие, в одном из этих случаев, а именно для второй команды, тип параметра будут определен неправильно.

Для облегчения анализа случаев ошибочной классификации была написана вспомогательная программа, которая для размеченных автоматически собирает необходимую информацию об ошибках. Примеры результатов работы этой программы приведены в Приложении Г.

Рассмотренные выше внутренние противоречия в коллекции размеченных команд можно решить, исключив некоторые команды из коллекции. Это приведет к большей разделимости классов, и в результате точность классификации, возможно, возрастет. Однако этот подход плох тем, что реальные пользовательские команды могут быть какими угодно, а значит, и обучать модели классификации нужно на самых разных примерах.

Другой подход состоит в добавлении новых признаков классификации, по которым удастся различать ранее похожие (или одинаковые) векторы. К примеру, таким новым признаком для определения типа параметра команды может быть позиция слова в предложении, или булево значение, показывающее, встречалось ли до этого в команде какое-либо слово, например, «текст».

Тем не менее даже без такой дополнительной настройки полученные результаты точности в 95% на выборке «50/50» на наш взгляд являются неплохим результатом.

Что касается оценки применимости метода опорных векторов к рассматриваемой задаче определения типа команд и их параметров, мы считаем, что его преимуществом по сравнению с другими методами анализа текстов на естественном языке является его применимость к произвольным формулировкам пользовательских команд, поскольку он не накладывает каких-либо синтаксических или лексических ограничений на анализируемые данные.

Так входными данными для описанного модуля-анализатора может быть сколь угодно большая пользовательская команда, к примеру:

«ну а теперь немедленно соедини меня с василием ивановичем петровым для очень серьезного разговора по душам».

Кроме того, метод SVM применим для неполных или синтаксически неверных предложений, которые иногда получаются в результате неправильного распознавания речи.

Например, результатом распознавания фразы

«Найди мне в википедии, что такое «интеллектуальный помощник» »

может быть следующая последовательность слов:

«найти википедия такое интеллектуальный помощник».

Однако несущественные ошибки распознавания, такие как неправильный падеж или отсутствие малоинформативных слов, в большинстве случаев не мешают классификатору правильно определить тип команды и ее параметры, и в результате приложение выполняет запрашиваемое действие.

Таким образом, благодаря применению метода анализа пользовательских команд на основе SVM, удается выполнить требование устойчивости приложения к ошибкам распознавания речи, а также требование его практической применимости, сформулированные в главе 2.

6. Заключение

За последние несколько лет технологии распознавания речи продвинулись далеко вперед: уже сейчас доступны онлайн-сервисы, способные распознать произнесенную фразу за доли секунды, и при этом точность распознавания оказывается довольно высокой. Этот факт делает актуальной задачу создания особых приложений для мобильных устройств — интеллектуальных помощников.

Чтобы лучше понять принципы работы интеллектуальных помощников и найти новые подходы к анализу пользовательских команд на русском языке, было решено создать собственный русскоязычный интеллектуальный помощник. К настоящему моменту разработан прототип такого приложения-помощника. В нём поддерживается работа с семью различными типами голосовых команд: телефонный вызов, отправка смс и электронных писем, поиск в интернете, отображение web-страницы в браузере, установка будильника и проверка баланса.

В данной курсовой работе была рассмотрена реализация модуля приложения, отвечающего за определение типа и параметров пользовательской команды. Для решения этой задачи было предложено использовать метод опорных векторов (SVM). В работе показаны следующие преимущества применения метода SVM к задаче определения типа пользовательской команды и ее параметров:

- повышение устойчивости анализа пользовательских команд к некоторым ошибкам распознавания речи;
- применимость метода SVM для анализа произвольных формулировок пользовательских команд, что очень важно для практической применимости приложения-помощника;
- учёт скрытых закономерностей и зависимостей при обучении классификатора на большом количестве размеченных данных.

Для обучения моделей SVM и оценки качества классификации была собрана и размечена коллекция пользовательских команд на русском языке в количестве 400 экземпляров.

При разбиении случайным образом данной коллекции на обучающую и тестовую выборки в отношении 1:1, полученные оценки точности классификации типов команд и их параметров превышают 95%, что, на наш взгляд, является неплохим показателем.

Для облегчения работы с коллекцией была написана вспомогательная программа, которая для размеченных команд автоматически строит векторы признаков, а в случае ошибочной классификации собирает необходимую информацию об ошибке. Для найденных ошибок предложены способы их исправления для дальнейшего увеличения точности классификации.

7. Список литературы и ссылки

1. *Intelligent personal assistant, Wikipedia* [Электронный ресурс] — URL: http://en.wikipedia.org/wiki/Intelligent_personal_assistant

2. *Apple - iOS 7 - Siri* [Электронный ресурс] — URL: http://www.apple.com/ios/siri/

3. Броварь И.В., Иванов Н.И. *Изучение компонент виртуальных персональных помощников для мобильных устройств,* – 2012 https://docs.google.com/file/d/0B6_tjXwNAFn0SlhJZzFocUxHR0U

4. *Автоматическая обработка текстов на естественном языке и компьютерная лингвистика : учеб. пособие* / Большакова Е. И., Клышинский Э. С., Ландэ Д. В. и др. – М.: МИЭМ, 2011. – с.181-184.

5. *LIBSVM — A Library for Support Vector Machines* [Электронный ресурс] — URL: http://www.csie.ntu.edu.tw/~cjlin/libsvm/

6. Chih-Wei Hsu, Chih-Chung Chang, Chih-Jen Lin *A Practical Guide to Support Vector Classification*, – 2010 http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf

7. *Speaktoit - Your Personal Assistant* [Электронный ресурс] — URL: http://www.speaktoit.com/

8. Репозиторий исходных кодов приложения-помощника Rino Recognizer https://github.com/nicolas-ivanov/Rino/tree/svm

8. Приложения

Приложение А.
Шаблоны регулярных выражений для описания ключевых слов
(файл «keywords»)

1-ый столбец – код семантической группы
(используется на этапе определения параметров команд)

2-ой столбец – идентификатор семантической группы

3-ий столбец – регулярное выражение, описывающее семантическую группу

```
10    a_call        \w*звон\w+|набрать|набер\w+|набир\w+|соедин\w+|телефон\w*| ↩
                    ↪ номер\w*
10    a_sms         смс\w*|sms|сообщение|смс
10    a_email       e?mail|письмо
10    a_search      найди|найти|ищи|искать|покаж\w+|показ\w*|\w*смотр\w*|узна\w+| ↩
                    ↪ скаж\w*|расска\w+|что|кто|тако\w*|значит\w*|означа\w+
10    a_site        покаж\w+|откр\w+
10    a_alarm       будильник|постав\w+|установ\w+
10    a_balance     баланс|покаж\w+|показ\w*|\w*смотр\w*|узна\w+|скаж\w*

1     p_name        Имя|Фамилия|Отчество|мам\w+|вас\w*|леш\w*|ван\w*|саш\w*| ↩
                    ↪ даш\w*|маш\w*|ан\w*|пет\w*|дим\w*|оля\w*|андр\w*| ↩
                    ↪ антон\w*|наст\w*
2     p_number      Номер|\d+|\+|плюс
3     p_email       Почта|[^S]+@[^S]+|e?mail|письмо
4     p_site        Сайт|\w+\.+ru|com|org|net|su
5     p_time        Время|утр\w+|вечер\w*|час\w*|минут\w*|завтра|полчаса

-2    s_text        текст\w*
-2    s_search      такое|значит|слов\w*|про|в|интернет\w+|википед\w+
-3    s_prepos      в|без|до|из|к|на|по|о|от|перед|при|через|с|у|за|над|об|под|про
10    s_write       напиш\w+|напис\w+|набрать|набер\w+|набир\w+|отправ\w+| ↩
                    ↪ перешл\w+|пересл\w+
```

Приложение Б.
Примеры размеченных команд для обучения и тестирования

(~150 примеров из коллекции размеченных команд)

Комментарии:

- первый столбец указывает на тип команды
- слова написанные с заглавной буквы (например, «Имя», «Номер») используются при обучении и отладке в качестве одного из возможных значений параметра (для того, чтобы не приходилось придумывать реальные имена, номера телефонов, и т. д.)
- нижние подчеркивания указывают на то, что следующее за ними слово является частью цитаты (т. е. частью тескта смс или поискового запроса)

```
########################################################################
# A collection of marked user commands for the following commands types:
# a_call     call to smb
# a_sms      send sms to smb
# a_email    send email to smb
# a_search   search smt on the web
# a_site     launch a site in browser
# a_alarm    set an alarm for particular time
# a_balance  check how much money you have on your account
########################################################################


# A_CALL section ######################################################

# commands with phone numbers
a_call 8 917 123 45 67
a_call набери 8 917 123 45 67
a_call набери телефон 8 917 123 45 67
a_call набери телефонный номер 8 917 123 45 67
a_call набери пожалуйста телефонный номер 8 917 123 45 67
...
a_call позвони по телефону плюс 7 917 123 45 67
a_call позвони по телефонному номеру плюс 7 917 123 45 67

# commands with one-word names
a_call вася
a_call позвони васе

a_call позвони пожалуйста васе
a_call позвони васе пожалуйста
a_call соедини с васей
a_call соедини меня с васей
```

```
a_call соедини пожалуйста с васей
a_call соедини меня пожалуйста с васей
a_call соедини с васей пожалуйста
a_call соедини меня с васей пожалуйста
...

# commands with 3-words names
a_call петров василий иванович
a_call позвони петрову василию ивановичу
...
a_call я хочу позвонить петрову василию ивановичу
a_call набери моего друга петрова василия ивановича
a_call набери пожалуйста моего друга петрова василия ивановича
a_call набери моего друга петрова василия ивановича пожалуйста



# A_SMS section ##########################################################

# no text, one-word name, "напиши"
a_sms  смс
a_sms  смс васе
a_sms  напиши смс васе
a_sms  напиши васе смс
...
a_sms  напиши пожалуйста смс васе
a_sms  напиши пожалуйста васе смс

# no text, two-words name, "напиши"
a_sms  смс васе петрову
a_sms  напиши смс васе петрову
a_sms  напиши васе петрову смс
...

# no text, three-words name, "напиши"
...
a_sms  смс сообщение
a_sms  смс сообщение петрову василию ивановичу
a_sms  напиши смс сообщение петрову василию ивановичу
a_sms  напиши петрову василию ивановичу смс сообщение
...

# text, name, "напиши"
a_sms  смс с текстом _здравствуйте
a_sms  смс с текстом _привет _как _дела
a_sms  смс васе с текстом _здравствуйте
a_sms  смс васе _привет _как _дела
a_sms  напиши смс васе пожалуйста с текстом _здравствуйте

# complicated text (keywords included), name, "напиши"
a_sms  смс с текстом _почему _не _звонишь
a_sms  смс васе с текстом _напиши _мне
```

```
a_sms  напиши смс васе с текстом _получил _твое _письмо
a_sms  напиши васе смс с текстом _найди _будильник

a_sms  смс сообщение с текстом _установил _новый _браузер
a_sms  смс сообщение васе с текстом _баланс _на _нуле
a_sms  напиши смс сообщение васе с текстом _саша _приехал
a_sms  напиши васе смс сообщение с текстом _позвони _на _номер _8 _917 _123 _45 _67

a_sms  напиши пожалуйста смс васе с текстом _sasha _собака _gmail.com
a_sms  напиши пожалуйста васе смс с текстом _зайди _на _yandex.ru
a_sms  отправь васе смс сообщение с текстом _завтра _буду _в _7 _часов
...

# no text, phone number
a_sms  смс на номер 8 917 123 45 67
a_sms  отправь смс на номер 8 917 123 45 67

a_sms  смс сообщение на номер 8 917 123 45 67
a_sms  напиши смс сообщение на номер 8 917 123 45 67
a_sms  напиши на номер 8 917 123 45 67 смс сообщение



# A_EMAIL section ######################################################

# no text, one-word name, "напиши"
a_email      email
a_email      email васе
a_email      напиши email васе
a_email      напиши васе email
...

# no text, two-words name, "напиши"
...
a_email      электронное письмо
a_email      электронное письмо васе петрову
a_email      напиши электронное письмо васе петрову
a_email      напиши васе петрову электронное письмо

# no text, three-words name, "напиши"
...
a_email      напиши пожалуйста электронное письмо петрову василию ивановичу
a_email      напиши пожалуйста петрову василию ивановичу электронное письмо

# text, name, "напиши"
a_email      email с текстом _здравствуйте
a_email      email с текстом _привет _как _дела
a_email      email васе с текстом _здравствуйте
a_email      email васе _привет _как _дела
a_email      напиши email васе пожалуйста с текстом _здравствуйте
```

```
# complicated text (keywords included), name, "напиши"
a_email      email с текстом _почему _не _звонишь
a_email      email васе с текстом _напиши _мне
a_email      напиши email васе с текстом _получил _твое _письмо
a_email      напиши васе email с текстом _найди _будильник

a_email      электронное письмо с текстом _установил _новый _браузер
a_email      электронное письмо васе с текстом _баланс _на _нуле
a_email      напиши электронное письмо васе с текстом _саша _приехал
a_email      напиши васе электронное письмо с текстом _позвони _на _номер ↩
             ↪ _8 _917 _123 _45 _67

a_email      напиши пожалуйста email васе с текстом _sasha _собака _gmail.com
a_email      напиши пожалуйста васе email с текстом _зайди _на _yandex.ru
a_email      отправь васе электронное письмо с текстом _завтра _буду _в _7 _часов



# A_SEARCH section ####################################################

# one-word search request
a_search     яндекс _дзен
a_search     _дзен яндекс
a_search     _дзен в яндексе

a_search     что такое _дзен
a_search     найди что такое _дзен
a_search     найди в яндексе что такое _дзен
a_search     найди что такое _дзен в яндексе

a_search     определение _конфуцианства
a_search     определение слова _конфуцианство
a_search     найди определение _конфуцианства
a_search     найди определение слова _конфуцианство

a_search     определение _конфуцианства в яндексе
a_search     определение слова _конфуцианство в яндексе
...

# two-words search request
...
a_search     найди пожалуйста определение _дзен _буддизма в яндексе
a_search     найди пожалуйста определение понятия _дзен _буддизм в яндексе

a_search     найди пожалуйста в яндексе определение _дзен _буддизма
a_search     найди пожалуйста в яндексе определение понятия _дзен _буддизм
a_search     найди пожалуйста определение _дзен _буддизма
a_search     найди пожалуйста определение понятия _дзен _буддизм

a_search     найди пожалуйста определение _дзен _буддизм в яндексе
a_search     найди пожалуйста определение понятия _дзен _буддизм в яндексе
```

```
a_search     найди пожалуйста в яндексе определение _дзен _буддизма
a_search     найди пожалуйста в яндексе определение понятия _дзен _буддизм
...

# multiple-words search request
...
a_search     найди пожалуйста в яндексе определение _философии _дзен _буддизма
a_search     найди пожалуйста в яндексе определение понятия _философия _дзен ↩
             ↪ _буддизма

a_search     найди определение понятия _философия _дзен _буддизма
a_search     найди на википедии определение понятия _философия _дзен _буддизма
a_search     найди определение понятия _философия _дзен _буддизма на википедии

a_search     найди пожалуйста определение понятия _философия _дзен _буддизма
a_search     найди пожалуйста на википедии определение понятия _философия _дзен
_буддизма
a_search     найди пожалуйста определение понятия _философия _дзен _буддизма ↩
             ↪ на википедии

# search requests with keywords
a_search     что такое _смс
a_search     найди что такое _email
a_search     найди в яндексе что такое _сайт
a_search     найди что такое _будильник в яндексе

a_search     определение слова _баланс
a_search     найди определение слова _браузер
a_search     определение слова _статья в яндексе

a_search     найди пожалуйста определение _телефонного _номера в яндексе
a_search     найди пожалуйста определение слова _sms в яндексе

a_search     найди пожалуйста в яндексе определение _электронного _письма
a_search     найди пожалуйста в яндексе определение понятия _электронная почта
a_search     найди пожалуйста определение _поисковой _системы
a_search     найди пожалуйста определение понятия _телефонный _вызов

a_search     найди пожалуйста в яндексе кто такой _вася
a_search     найди пожалуйста в яндексе что такое _плюс

a_search     найди что нибудь о _василии _ивановиче
a_search     найди на википедии что такое _собака
a_search     найди на википедии что такое яндекс

a_search     найди пожалуйста определение слова _дзен
a_search     найди пожалуйста на википедии определение слова _дзен
a_search     найди пожалуйста что такое час
```

```
# A_SITE section ########################################################
a_site yandex.ru
a_site открой yandex.ru
...
a_site пожалуйста открой yandex.ru в браузере
a_site открой пожалуйста yandex.ru в браузере
a_site открой yandex.ru в браузере пожалуйста


# A_ALARM section ########################################################
a_alarm      будильник
a_alarm      будильник на 7
a_alarm      будильник на 7 30
a_alarm      поставь будильник на 7
a_alarm      поставь будильник на 7 30
a_alarm      будильник на 7 часов
a_alarm      будильник на 7 часов 30 минут
a_alarm      поставь будильник на 7 часов
a_alarm      поставь будильник на 7 часов 30 минут
...
a_alarm      пожалуйста разбуди меня в 7 часов 30 минут
a_alarm      разбуди меня пожалуйста в 7 часов 30 минут
a_alarm      разбуди меня в 7 часов 30 минут пожалуйста


# A_BALANCE section ######################################################
a_balance    баланс
a_balance    баланс средств
a_balance    баланс телефона

a_balance    покажи баланс
a_balance    покажи баланс телефона
a_balance    пожалуйста покажи баланс
a_balance    покажи пожалуйста баланс
a_balance    покажи баланс пожалуйста
...
a_balance    какой баланс
a_balance    какой сейчас баланс
a_balance    я хочу узнать баланс
a_balance    я хочу узнать свой баланс
```

Приложение В.
Фрагмент модели libsvm для классификации команд по типам

```
svm_type c_svc
kernel_type rbf
gamma 2
nr_class 7
total_sv 91
rho 0.450056 -0.332252 0.430312 -0.179546 -0.0385735 -0.371645 -0.764802 -0.0245156
-0.523411 -0.440301 -0.681178 0.665124 0.0905238 0.309652 0.0336491 -0.560536
-0.440896 -0.72503 0.184171 -0.135873 -0.313547
label 1 2 3 4 5 6 7
probA -4.05004 -3.27128 -3.8835 -3.69011 -3.67313 -3.48831 -2.54634 -4.48846
-3.59093 -3.9247 -3.01133 -3.25867 -2.93096 -2.64671 -2.27055 -3.62171 -3.71845
-3.01791 -3.01164 -2.75766 -2.52064
probB -0.0160797 0.144003 0.200402 0.766705 0.0830334 0.136663 0.286248 0.422978
0.153799 0.501818 0.057825 -0.180343 -0.374428 -0.275473 -0.23312 -0.198579 0.198207
0.0124637 0.127166 0.0786757 -0.167929
nr_sv 16 23 7 19 8 11 7
SV
0.5 0.3430213043988293 0.5 0.2924986423240503 0.5 0.5 8:0.5
0.5 0.5 0.5 0.5 0.5 0.5 9:0.5
0.5 0 0.4598240530288463 0 0.3596548418212314 0.1856368997884134 1:0.333333 15:0.5
0.4863906600764918 0 0.1526585723912668 0.1028563125232033 0 0 1:0.666667 15:0.5
0.5 0.39714529200954 0.3143336821310739 0.3185160180737469 0.4033296009170587 0.5
1:0.333333 15:0.5
0.2288485645764056 0 0 0 0 0 1:0.333333 9:0.5 15:0.5
0.3966366016513943 0 0.3651267596331151 0 0 0 1:0.333333 8:1 15:0.5
0.5 0 0.2431050851064668 0 0 0 1:0.333333 8:1
0.5 0.3876165051138429 0.3764835059550777 0.4401732444452774 0.4682273140345504
0.287981251119958 1:0.333333 8:1 15:0.5
0.2181305182856478 0.241082512347777 0.5 0.3409940612242656 0.3646819003640135
0.1539512222703681 1:0.333333 8:1
0.005899071131055281 0 0 0 0 0 1:0.333333 8:0.5 15:0.5
0.2972055860988062 0 0.0005985116523162763 0 0.08504382483144776 0.1312026940867127
1:0.333333 8:0.5
...
```

Приложение Г.
Анализ ошибок для случая классификации параметров команд

SMS-сообщения.

1-ый столбец – слово, которое нужно классифицировать
(если перед словом стоит восклицательный знак, это показывает, что оно было классифицировано неправильно);

2-ой столбец – код метки;

3-ий столбец – вектор параметров слова;

4-ий столбец – код метки, возвращенный классификатором.

	Original text	O.label	Original params	Predicted label
	напиши	10	2:1 3:1 32:1	10
	смс	10	2:1 17:1 18:1 38:1	10
	васе	1	8:1 17:1	1
!	пожалуйста	0	23:1	-1
	с	0	43:1	0
	текстом	-2	13:1	-2
	_здравствуйте	-1	28:1	-1
	напиши	10	2:1 3:1 38:1	10
	васе	1	8:1 17:1 18:1 32:1	1
	смс	10	2:1 23:1 32:1	10
	сообщение	10	2:1 17:1 45:1	10
	с	-3	15:1 17:1 43:1	-3
	текстом	-2	13:1 30:1 31:1	-2
	_позвони	-1	1:1 28:1 45:1	-1
	_на	-1	15:1 16:1 31:1 32:1	-1
!	_номер	-1	1:1 2:1 30:1 39:1 42:1	10
!	_8	-1	9:1 12:1 16:1 17:1 39:1	2
!	_917	-1	9:1 24:1 27:1 39:1	2
!	_123	-1	9:1 24:1 39:1 42:1	2
!	_45	-1	9:1 12:1 24:1 39:1 42:1	2
!	_67	-1	9:1 12:1 24:1 27:1	2
	набери	10	1:1 2:1 3:1 38:1	10
	васе	1	8:1 16:1 17:1 18:1 32:1	1
	смс	10	2:1 23:1 32:1	10
	сообщение	10	2:1 17:1 45:1	10
	с	-3	15:1 17:1 43:1	-3
	текстом	-2	13:1 30:1 31:1	-2
	_позвони	-1	1:1 28:1 45:1	-1
	_на	-1	15:1 16:1 31:1 32:1	-1

```
! _номер          -1       1:1 2:1 30:1 39:1 42:1          10
! _8              -1       9:1 12:1 16:1 17:1 39:1          2
! _917            -1       9:1 24:1 27:1 39:1               2
! _123            -1       9:1 24:1 39:1 42:1               2
! _45             -1       9:1 12:1 24:1 39:1 42:1          2
! _67             -1       9:1 12:1 24:1 27:1               2
```

Электронные письма.

```
  Original text    O.label  Original params            Predicted label

  напиши           10       2:1 3:1 33:1                     10
  email            10       3:1 17:1 18:1 38:1               10
  васе              1       8:1 18:1                          1
! пожалуйста       0        23:1                             -1
  с                 0       43:1                              0
  текстом          -2       13:1                             -2
  _здравствуйте    -1       28:1                             -1


  email            10       3:1 38:1                         10
  васе              1       8:1 18:1 45:1                     1
  с                -3       15:1 23:1 43:1                   -3
  текстом          -2       13:1 30:1 32:1 33:1              -2
  _напиши          -1       2:1 3:1 28:1                     -1
! _мне             -1       17:1 18:1                         0


  напиши           10       2:1 3:1 33:1                     10
  email            10       3:1 17:1 18:1 38:1               10
  васе              1       8:1 18:1 45:1                     1
  с                -3       15:1 23:1 43:1                   -3
  текстом          -2       13:1 30:1                        -2
  _получил         -1       28:1                             -1
  _твое            -1       33:1                             -1
! _письмо          -1       3:1                              10


  напиши           10       2:1 3:1                          10
  пожалуйста        0       17:1 18:1 33:1                    0
  email            10       3:1 38:1                         10
  васе              1       8:1 18:1 45:1                     1
  с                -3       15:1 23:1 43:1                   -3
  текстом          -2       13:1 30:1                        -2
  _sasha           -1       28:1 40:1                        -1
! _собака          -1       10:1 41:1                         3
! _gmail.com       -1       11:1 25:1                         4
```

```
  набери          10      1:1 2:1 3:1 33:1              10
  email           10      3:1 16:1 17:1 18:1 38:1       10
  васе             1      8:1 18:1 45:1                  1
  с               -3      15:1 23:1 43:1                -3
  текстом         -2      13:1 30:1                     -2
  _получил        -1      28:1                          -1
  _твое           -1      33:1                          -1
! _письмо         -1      3:1                           10


  набери          10      1:1 2:1 3:1                   10
  пожалуйста       0      16:1 17:1 18:1 33:1            0
  email           10      3:1 38:1                      10
  васе             1      8:1 18:1 45:1                  1
  с               -3      15:1 23:1 43:1                -3
  текстом         -2      13:1 30:1                     -2
  _sasha          -1      28:1 40:1                     -1
! _собака         -1      10:1 41:1                      3
! _gmail.com      -1      11:1 25:1                      4
```