

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА БАКАЛАВРА**

**ПЕРСОНАЛЬНЫЙ АССИСТЕНТ С ГОЛОСОВЫМ УПРАВЛЕНИЕМ НА ОСНОВЕ МОБИЛЬНОГО УСТРОЙСТВА**

Работу выполнил А. В. Лемешко

[СОДЕРЖАНИЕ](#_Toc39825)

[ВВЕДЕНИЕ 4](#_Toc39825)

[1. АНАЛИТИЧЕСКАЯ ЧАСТЬ 5](#_Toc39826)

[1.1. Обзор современных средств распознавания речи 5](#_Toc39827)

[1.1.1. Краткая история технологии распознавания речи 5](#_Toc39828)

[1.1.2. «Горыныч проф.» версии 3.0 7](#_Toc39829)

##### [1.1.3. Обучающие системы «ИстраСофт» 9](#_Toc39830)

[1.1.4. Комплекс шумоочистки речи Sound Cleaner 10](#_Toc39831)

[1.1.5.Pocketsphinx 11](#_Toc39832)

[1.1.6.Google Cloud Speech API 13](#_Toc39833)

[1.1.7. Выводы по разбору аналогов 13](#_Toc39834)

[1.2. Теоретические основы технологии распознавания речи 15](#_Toc39835)

[1.2.1. Классификация систем распознавания речи 15](#_Toc39836)

[1.2.2. Методы и алгоритмы распознавания речи 15](#_Toc39837)

[1.2.3. Архитектура систем распознавания 20](#_Toc39838)

[2. ПРОЕКТНАЯ ЧАСТЬ 22](#_Toc39839)

[2.1. Требования к программе 22](#_Toc39840)

[2.2. Необходимые аппаратные и программные средства 23](#_Toc39841)

[2.2.1. Требования к устройству 23](#_Toc39842)

[2.2.2. Используемый инструментарий 23](#_Toc39843)

[2.3. Разработка Android - приложения 23](#_Toc39844)

[3. ЧАСТЬ РЕАЛИЗАЦИИ 27](#_Toc39845)

[3.1. Реализация xml-разметки 27](#_Toc39846)

[3.2.Реализация MainActivity.java 28](#_Toc39847)

[3.1.1.Метод onCreate 29](#_Toc39848)

[3.1.2. Функция promptSpeechInput() 29](#_Toc39849)

[3.1.3. Функция onActivityResult() 30](#_Toc39850)

[3.1.4. Функция GetNumberContact () 35](#_Toc39851)

[3.1.5. Функция GetNumberSms () 35](#_Toc39852)

[3.1.6. Функция addEvent() 36](#_Toc39853)

[ЗАКЛЮЧЕНИЕ 37](#_Toc39854)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 38](#_Toc39855)

[ПРИЛОЖЕНИЕ А Код программы 39](#_Toc39856)

[ПРИЛОЖЕНИЕ В Разметка приложения 46](#_Toc39857)

[ПРИЛОЖЕНИЕ С Manifest приложения 48](#_Toc39858)

##### ВВЕДЕНИЕ

Распознавание речи играет все большую роль в современном мире. На данный момент существую большое количество систем, которые реализованы на основе различных технологий распознавания. Распознавание речи получило широкое распространение как в разнообразных приложениях, так и будучи встроенными во всевозможные устройства.

Благодаря данным системам всѐ проще становится освоение компьютеров для неподготовленного человека, т.к. современные системы распознавания речи заставляют компьютеры подстраиваться под человека, а не наоборот. Голосовые команды позволяют производить различные действия с компьютером даже не прикасаясь при этом к клавиатуре или мыши. Например, для поиска в интернете конкретной информации требуется довольно долгий и утомительный процесс набора и поиска информации в сети. Но если компьютер может адекватно воспринимать речь человека, то поиск информации можно ускорить в разы.

Аналогичные устройства используются на рынке бортовых устройств управления автомобилями, на рынке мобильных устройств и КПК.

# 1. АНАЛИТИЧЕСКАЯ ЧАСТЬ

В данной части работы будут рассмотрены аналоги программ с функцией распознавания речи, выявлены их плюсы и минусы, а также подробно рассмотрены классификация, методы и алгоритмы распознавания речи.

### 1.1. Обзор современных средств распознавания речи

##### 1.1.1. Краткая история технологии распознавания речи

Первое устройство для распознавания речи появилось в 1952 году, оно могло распознавать произнесѐнные человеком цифры [1]. В 1964 году на ярмарке компьютерных технологий в Нью-Йорке было представлено устройство IBM Shoebox.

Коммерческие программы по распознаванию речи появились в начале девяностых годов. Чаще всего их используют люди, которые из-за травмы руки не в состоянии набирать большое количество текста. Эти программы переводят голос пользователя в текст, разгружая руки пользователя. Надѐжность перевода у таких программ не очень высока, но с годами она постепенно улучшается.

Увеличение вычислительных мощностей мобильных устройств позволило и для них создать программы с функцией распознавания речи. Среди таких программ стоит отметить приложение Microsoft Voice Command, которое позволяет работать со многими приложениями при помощи голоса. Например, можно включить воспроизведение музыки в плеере или создать новый документ.

Интеллектуальные речевые решения, позволяющие автоматически синтезировать и распознавать человеческую речь, являются следующей ступенью развития интерактивных голосовых систем (IVR). Использование интерактивного телефонного приложения в настоящее время не роскошь, а жизненная необходимость.

Таблица 1 - Таблица рассмотренных средств по распознаванию речи

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование | Характеристика | Плюсы | Минусы |
| «Горыныч» | Система ориентирована на ввод, но содержит  средства, позволяющие исправить неверно услышанное слово, для чего предлагается список вариантов. | С увеличением мощности компьютеров пропала временная задержка между произнесением слова и отображением его письменного варианта на экране. Бесплатное приложение. | Программа требует длительного периода тренировки и настройки под конкретного пользователя, очень зависит от оборудования, сильно чувствительна к интонации и скорости произнесения фраз |
| Обучающие системы «ИстраСофт» | В программах слова  делятся на элементарные сегменты, которые соответствуют звукам речи и не зависят ни от диктора, ни от языка. | Данный способ допускает сжатие речи в 200 раз, причем при сжатии менее чем в 40 раз качество речевого сигнала практически не ухудшается. | В основном, все приложения платные. |
| Комплекс шумоочистки  речи  Sound Cleaner | Набор программноаппаратных средств, для восстановления разборчивости речи и для очищения звуковых сигналов. | Эффективная работа в отношении шумов и искажений звука известной природы. | Сложное в освоении обычному пользователю. Платное приложение. |
| Pocketsphinx | Система  распознавания речи, которая используется для создания голосовых приложений. | Обеспечение приемлемого качества распознавания и относительную простоту внедрения. Имеется в открытом доступе. | Разное поведение на различных устройствах — от скорости распознавания до качества звука с микрофона и влияния посторонних шумов. |
| Google Cloud  Speech API | Данная технология используется для | Повышенная точность | Первые 60 минут обработки звука |
|  | управления приложениями и устройствами с помощью голосового поиска, речевых команд и голосового меню. | транскрипции продолжительных аудиозаписей и ускоренный процесс обработки данных в 3 раза по сравнению с первоначальной версией. Также была добавлена поддержка таких аудиоформатов, как WAV, OPUS и Speex. | бесплатны, далее взымается плата за каждые 15 секунд. |

##### 1.1.2. «Горыныч проф.» версии 3.0

В 1997 году вышел знаменитый «Горыныч», его появление стало своеобразной сенсацией. Ядро Dragon Naturally Speaking было настроено на особенности англоязычной речи, но даже после последовательной замены на русскоязычный лад, оно дает не более 30-40% распознавания среднего уровня лексики, даже при тщательном проговаривании. По заявлениям разработчиков компаний Dragon Systems, IBM и Lernout&Hauspie, их программы при непрерывной диктовке были способны правильно распознавать до 95% текста, но для комфортной работы точность распознавания необходимо довести до 99%.

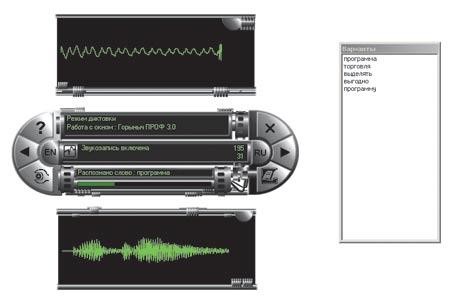


Рисунок 1 - Пример работы программы распознавания речи

Кроме того, программа требует длительного периода тренировки и настройки под конкретного пользователя, очень капризна к оборудованию, более чем чувствительна к интонации и скорости произнесения фраз, так что возможности ее обучения распознаванию различных голосов сильно различаются.

При рассмотрении новой версии этой программы ничего нового из нее получить не удалось. Даже после длительного «обучения» программы оказалось, что диктовка по-прежнему должна осуществляться строго по словам и слова нужно произносить отчетливо, что не всегда характерно для речи.

Система ориентирована в первую очередь на ввод, но содержит средства, позволяющие исправить неверно услышанное слово, для чего «Горыныч» предлагает список вариантов. Можно поправить текст и с клавиатуры, что постоянно и приходится делать. С клавиатуры вводятся и слова, отсутствующие в словаре.

Проявление прогресса в этой области только одно: из-за увеличения мощности компьютера совершенно пропала временная задержка между произнесением слова и отображением его письменного варианта на экране, а число правильных попаданий так и не увеличилось.

Очевидно, что, речевые технологии делаются все более зависимыми от того языка, с которым они работают. А это значит, во-первых, что распознавание, синтез и обработка русской речи являются тем делом, заниматься которым должны именно российские разработчики, а во-вторых, только специализированные отечественные продукты, изначально ориентированные именно на русский язык, смогут по-настоящему решить ту задачу

##### 1.1.3. Обучающие системы «ИстраСофт»

Пока технологии распознавания речи российскими разработчиками успешно применяются в основном в интерактивных обучающих системах и играх вроде «Мой говорящий словарь», Talk to Me или «Профессор Хиггинс», созданных фирмой «ИстраСофт». Используются они для контроля произношения у изучающих английский язык и аутентификации пользователя. Развивая программу «Профессор Хиггинс», сотрудники «ИстраСофт» научились членить слова на элементарные сегменты, которые соответствуют звукам речи и не зависят ни от диктора, ни от языка. При этом выделение фонем из потока слитной речи, их кодирование и последующее восстановление происходит в режиме реального времени. Указанная технология распознавания речи нашла довольно остроумное применение — она позволяет существенно сжимать файлы с диктофонными записями или речевыми сообщениями. Способ, предложенный фирмой «ИстраСофт», допускает сжатие речи в 200 раз, причем при сжатии менее чем в 40 раз качество речевого сигнала практически не ухудшается. Интеллектуальная обработка речи на уровне фонем перспективна не только как способ сжатия, но и как шаг на пути к созданию нового поколения систем распознавания речи, ведь теоретически машинное распознавание речи, то есть ее автоматическое представление в виде текста, как раз и является крайней степенью сжатия речевого сигнала.

##### 1.1.4. Комплекс шумоочистки речи Sound Cleaner

Петербуржская компания «Центр речевых технологий», которая работает в этой области с 1990 года, похоже, добилась определенных успехов. ЦРТ имеет в своем арсенале целый набор программных и аппаратных средств, предназначенных для шумоочистки и для повышения качества звуковых, и в первую очередь речевых, сигналов — это компьютерные программы, автономные устройства, платы (DSP), встраиваемые в устройства каналов записи или передачи речевой информации. «Центр речевых технологий» известен как разработчик средств шумоподавления и редактирования звука: Clear Voice, Sound Cleaner, Speech Interactive Software, Sound Stretcher и др. Специалисты фирмы принимали участие в восстановлении аудиоинформации, записанной на борту затонувшей подлодки «Курск» и на потерпевших катастрофы воздушных судах, а также в расследовании ряда уголовных дел, для которых требовалось установить содержание фонограмм речи.



Рисунок 2 - Демонстрация процесса шумоочистки

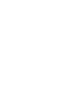
Комплекс шумоочистки речи Sound Cleaner представляет собой профессиональный набор программно-аппаратных средств, предназначенных для восстановления разборчивости речи и для очищения звуковых сигналов, записанных в сложных акустических условиях или передаваемых по каналам связи.

Естественно, Sound Cleaner эффективнее работает в отношении шумов и искажений звука известной природы, таких как типовые шумы и искажения каналов связи и звукозаписи, шумы помещений и улиц, работающих механизмов, транспортных средств, бытовой техники, голосового «коктейля», медленной музыки, электромагнитных наводок систем питания, компьютерной и другой техники, эффектов реверберации и эха. В принципе, чем равномернее и «регулярнее» шум, тем успешнее этот комплекс с ним справится.

Однако при двухканальном съеме информации Sound Cleaner существенно снижает влияние шумов любого типа — например, он имеет методы двухканальной адаптивной фильтрации, предназначенные для подавления как широкополосных нестационарных помех, так и периодических. Эти методы основаны на том, что при выделении полезного сигнала используется дополнительная информация о свойствах помехи, представленная в опорном канале.

##### 1.1.5. Pocketsphinx

Pocketsphinx — система распознавания слитной речи — разрабатывается инженерами из Университета Карнеги-Мелон и компилируется из исходников в библиотеку под ARM с помощью Android NDK. При этом можно сгенерировать JNI-обертки для использования нативных методов из кода на Java.



MIC



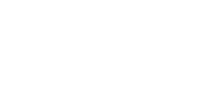
Result



File.wav

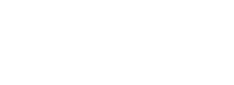


Декодер



Модуль

sphinxbas



Модуль

Pocketsphin



1



2



3



5



4



6

Рисунок 3 - Функциональная схема потоков информации в Pocketsphinx

На рисунке 3 представлена функциональная схема, которая показывает принцип приема и обработки конечной информации. 1 – это звуковой файл в формате WAV, он не обязателен в данном проекте. 3 – это микрофон, с его помощью у нас есть возможность выполнить поставленную задачу. 2 – декодер, модуль для расшифровки звукового файла, чтобы модуль Pocketsphinx 4 смог его распознать. Для работы Pocketsphinx 4 на некоторых платформах требуется еще и другой модуль, который представляет из себя динамическую библиотеку зависимостей sphinxbase.dll – это модуль 5. Конечным результатом работы является предложение в файле result.txt – 6, которое соответствует звуковому смыслу предложения, которое звучит в файле 1 или микрофоне 3, в случае данного проекта, для дальнейшего выполнения.

Pocketsphinx предоставляет возможность использовать речь во многих проектах для мобильных платформ, обеспечивая приемлемое качество распознавания и относительную простоту внедрения.

##### 1.1.6. Google Cloud Speech API

Открытая бета-версия Cloud Speech была выпущена летом 2016 года. Эта технология с простым API позволяет разработчикам преобразовывать аудио в текст. Модели нейронной сети могут распознавать более 80 языков и диалектов, а готовая транскрипция появляется сразу после проговаривания текста.

API построен на базе технологии, которая обеспечивает функцию распознавания речи в Google Assistant, Search и Now, однако в новой версии были внесены изменения для адаптации технологии под нужды пользователей Cloud.

Благодаря многочисленным отзывам команда Google смогла повысить точность транскрипции продолжительных аудиозаписей и ускорить процесс обработки данных в 3 раза по сравнению с первоначальной версией. Также была добавлена поддержка других аудиоформатов, включая WAV, OPUS и Speex.

По статистике, раньше этот API использовался чаще всего для управления приложениями и устройствами с помощью голосового поиска, речевых команд и голосового меню. Но Cloud Speech может быть использован в совершенно разных IoT-устройствах, включая автомобили, телевизоры, колонки и, конечно, телефоны и ПК.

##### 1.1.7. Выводы по разбору аналогов

В настоящее время технология распознавания речи уже достигла такого уровня, что теперь стало возможным использовать ее для организации полностью автоматизированных телефонных служб, способных работать в интерактивном голосовом режиме. Одно за другим появляются сообщения об успешном внедрении таких служб и о предоставляемых ими преимуществах.

Точность распознавания, как правило, повышается при предварительной настройке на голос конкретного пользователя, причем этим способом можно добиться распознавания речи даже тогда, когда у говорящего имеется дефект дикции или акцент. Но заметные успехи в этой области видны только в том случае, если предполагается индивидуальное применение оборудования или ПО одним, или несколькими пользователями, в крайнем случае, для каждого из которых создается свой индивидуальный «профиль».

В итоге, несмотря на все достижения, средства для распознавания слитной речи все еще допускают большое количество ошибок, нуждаются в длительной настройке, требовательны к аппаратной части и к квалификации пользователя и отказываются работать в зашумленных помещениях, хотя последнее важно, как для шумных офисов, так и для мобильных систем и эксплуатации в условиях телефонной связи.

## 1.2. Теоретические основы технологии распознавания речи

#### 1.2.1. Классификация систем распознавания речи

Системы распознавания речи можно разделить на следующие

классификации: [2]

1. по размеру словаря (ограниченный набор слов, словарь большого размера);
2. по зависимости от диктора (дикторозависимые и

дикторонезависимые системы);

1. по типу речи (слитная или раздельная речь);
2. по назначению (системы диктовки, командные системы);
3. по используемому алгоритму (нейронные сети, скрытые Марковские модели, динамическое программирование);
4. по типу структурной единицы (фразы, слова, фонемы, дифоны, аллофоны);
5. по принципу выделения структурных единиц (распознавание по шаблону, выделение лексических элементов).

Для систем автоматического распознавания речи, помехозащищѐнность обеспечивается, прежде всего, использованием двух механизмов: [3]

1. Использование нескольких, параллельно работающих, способов выделения одних и тех же элементов речевого сигнала на базе анализа акустического сигнала;
2. Параллельное независимое использование сегментного

(фонемного) и целостного восприятия слов в потоке речи.

#### 1.2.2. Методы и алгоритмы распознавания речи

Методы и алгоритмы, которые использовались до сих пор, могут быть разделены на следующие большие классы:

1) Динамическое программирование — временные динамические алгоритмы (Dynamic Time Warping).

a) Контекстно-зависимая классификация. При еѐ реализации из потока речи выделяются отдельные лексические элементы — фонемы и аллофоны, которые затем объединяются в слоги и морфемы.

1. Методы дискриминантного анализа, основанные на Байесовской дискриминации (Bayesian discrimination);
2. Скрытые Марковские модели (Hidden Markov Model);

Скрытой Марковской моделью (СММ) называется модель, состоящая из N состояний, в каждом из которых некоторая система может принимать одно из M значений какого-либо параметра. Вероятности переходов между состояниями задается матрицей вероятностей A={aij}, где aij – вероятность перехода из i-го в j-е состояние. Вероятности выпадения каждого из M значений параметра в каждом из N состояний задается вектором B={bj(k)}, где bj(k) – вероятность выпадения k-го значения параметра в j-м состоянии. Вероятность наступления начального состояния задается вектором π={πi}, где πi – вероятность того, что в начальный момент система окажется в i-м состоянии.

Таким образом, скрытой Марковской моделью называется тройка λ={A,B,π}. Использование скрытых Марковских моделей для распознавания речи основано на двух приближениях:

1. Речь может быть разбита на фрагменты, соответствующие состояниям в СММ, параметры речи в пределах каждого фрагмента считаются постоянными.
2. Вероятность каждого фрагмента зависит только от текущего состояния системы и не зависит от предыдущих состояний.

Модель называется «скрытой», так как нас, как правило, не интересует конкретная последовательность состояний, в которой пребывает система. Мы либо подаем на вход системы последовательности типа O={o1,o2,…oi} - где каждое oi – значение параметра (одно из M), принимаемое в i-й момент времени, а на выходе ожидаем модель λ={A,B,π}с максимальной вероятностью генерирующую такую последовательность, - либо наоборот подаем на вход параметры модели и генерируем порождаемую ей последовательность. И в том и другом случае система выступает как ―черный ящик‖, в котором скрыты действительные состояния системы, а связанная с ней модель заслуживает названия скрытой.

Для осуществления распознавания на основе скрытых моделей Маркова необходимо построить кодовую книгу, содержащую множество эталонных наборов для характерных признаков речи (например, коэффициентов линейного предсказания, распределения энергии по частотам и т.д.). Для этого записываются эталонные речевые фрагменты, разбиваются на элементарные составляющие (отрезки речи, в течении которых можно считать параметры речевого сигнала постоянными) и для каждого из них вычисляются значения характерных признаков. Одной элементарной составляющей будет соответствовать один набор признаков из множества наборов признаков словаря. [4]

Фрагмент речи разбивается на отрезки, в течении которых параметры речи можно считать постоянными. Для каждого отрезка вычисляются характерные признаки и подбирается запись кодовой книги с наиболее подходящими характеристиками. Номера этих записей и образуют последовательность наблюдений O={o1,o2,…oi} для модели Маркова. Каждому слову словаря соответствует одна такая последовательность. Далее A – матрица вероятностей переходов из одного минимального отрезка речи (номера записи кодовой книги) в другой минимальный отрезок речи (номер записи кодовой книги). В – вероятности выпадения в каждом состоянии конкретного номера кодовой книги (рисунок 4).

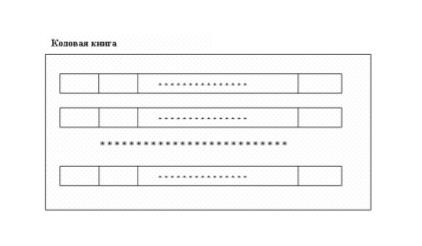


Рисунок 4 - Кодовая книга

На этапе настройки моделей Маркова мы применяем алгоритм Баума- Уэлча для имеющегося словаря и сопоставления каждому из его слов матрицы A и B.

При распознавании мы разбиваем речь на отрезки, для каждого вычисляем набор номеров кодовой страницы и применяем алгоритм прямого или обратного хода для вычисления вероятности соответствия данного звукового фрагмента определенному слову словаря. Если вероятность превышает некоторое пороговое значение – слово считается распознанным.

4) Нейронные сети (Neural networks).

Искусственная нейронная сеть — это математическая модель, а также устройства параллельных вычислений, представляющие собой систему соединѐнных и взаимодействующих между собой простых процессоров (искусственных нейронов). Как математическая модель искусственная нейронная сеть представляет собой частный случай методов распознавания образов или дискриминантного анализа. Пример нейросети изображен на рисунке 5.

Каждый процессор подобной сети имеет дело только с сигналами, которые он периодически получает, и сигналами, которые он периодически посылает другим процессорам. И тем не менее, будучи соединѐнными в достаточно большую сеть с управляемым взаимодействием, такие локально простые процессоры вместе способны выполнять довольно сложные задачи.

[5]

Понятие возникло при изучении процессов, протекающих в мозге при мышлении, и при попытке смоделировать эти процессы. Полученные модели называются искусственными нейронными сетями (ИНС).

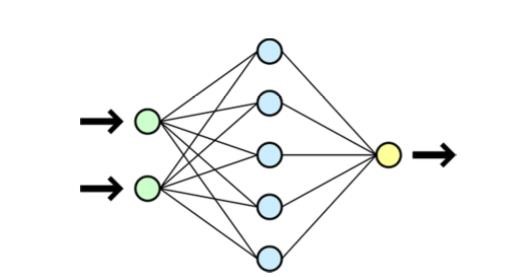


Рисунок 5 - Схема простой нейросети, где зелѐным обозначены входные элементы, жѐлтым — выходной элемент

Нейронные сети не программируются в привычном смысле этого слова, они обучаются. Возможность обучения — одно из главных преимуществ нейронных сетей перед традиционными алгоритмами. Технически обучение заключается в нахождении коэффициентов связей между нейронами. В процессе обучения нейронная сеть способна выявлять сложные зависимости между входными данными и выходными, а также выполнять обобщение. Это значит, что, в случае успешного обучения, сеть сможет вернуть верный результат на основании данных, которые отсутствовали в обучающей выборке.

Решение задачи распознавания с помощью нейронных сетей обладает значительным преимуществом перед алгоритмами, основанными на вычислении метрик – вычислительные затраты не зависят от количества слов в словаре. При увеличении длины словаря увеличивается лишь размер обучающей выборки, то есть нейронной сети требуется затратить больше времени на процесс обучения, но трудоемкость процесса распознавания не изменяется. Такая особенность позволяет оперировать с достаточно большим количеством слов в словарях. Недостатком нейросетевого подхода является отсутствие возможности добавления новых слов в словарь после окончания процесса обучения. Для разрешения этой проблемы может быть применена теория адаптивного резонанса. Нейронные сети, построенные в рамках теории адаптивного резонанса сохраняют пластичность при запоминании новых образов, и, в то же время, предотвращают модификацию старой

памяти. [6]

#### 1.2.3. Архитектура систем распознавания

Типичная архитектура статистических систем автоматической обработки речи [7]:

1. Модуль шумоочистки и отделение полезного сигнала.
2. Акустическая модель — позволяет оценить распознавание речевого сегмента с точки зрения схожести на звуковом уровне. Для каждого звука изначально строится сложная статистическая модель, которая описывает произнесение этого звука в речи.
3. Языковая модель — позволяют определить наиболее вероятные последовательности слов. Сложность построения языковой модели во многом зависит от конкретного языка. Так, для английского языка, достаточно использовать статистические модели (так называемые Nграммы). Для высокофлективных языков (языков, в которых существует много форм одного и того же слова), к которым относится и русский, языковые модели, построенные только с использованием статистики, уже не дают такого эффекта — слишком много нужно данных, чтобы достоверно оценить статистические связи между словами. Поэтому применяют гибридные языковые модели, использующие правила русского языка, информацию о части речи и форме слова и классическую статистическую модель.
4. Декодер — программный компонент системы распознавания, который совмещает данные, получаемые в ходе распознавания от акустических и языковых моделей, и на основании их объединения, определяет наиболее вероятную последовательность слов, которая и является конечным результатом распознавания слитной речи.

# 2. ПРОЕКТНАЯ ЧАСТЬ

В этой части подробнее рассмотрим необходимые шаги для создания собственной программы с функцией распознавания речи.

### 2.1. Требования к программе

Разрабатываемое приложение должно позволять:

1. При помощи голосовой команды набирать номера абонентов по их номеру телефона, либо по их имени в телефонной книге устройства.

В данном случае после активации приложения нажимается кнопка для начала распознавания, после произносится кодовая фраза «позвонить абоненту» или «набрать абонента», после чего диктуется номер телефона, либо имя абонента в телефонной книге, спустя несколько секунд устройство показывает результат запроса, после чего происходит набор заданного абонента.

1. Набирать продиктованные сообщения для последующей отправки на номер абонента или на указанное имя в телефонной книге устройства.

В данном случае после активации приложения нажимается кнопка для начала распознавания, затем произносится кодовая фраза «написать сообщение», «отправить сообщение» или «написать смс», «отправить смс», после чего устройству диктуется сперва номер абонента или его имя в телефонной книге для отправки ему сообщения, а после небольшой паузы по звуковому сигналу надо будет продиктовать само сообщение, после чего оно отправится получателю.

1. Посредством голоса оставлять заметки в календаре устройства.

В данном случае после активации приложения нажимается кнопка для начала распознавания, потом снова произносится кодовая фраза «поставить напоминание» или «оставить заметку» после чего на устройство сперва диктуется непосредственно текст заметки, а после небольшой паузы нужно произнести дату, на которую пользователь данную заметку хочет оставить.

### 2.2. Необходимые аппаратные и программные средства

##### 2.2.1. Требования к устройству

Данная программа тестировалась на смартфоне ZTE BLADE V7 LITE со следующими характеристиками:

1. Четырехъядерный процессор с частотой 1000МГц;
2. Память – 16Гб;
3. ОС Android 6.0.

В случае несоблюдения требований, работа приложения на устройстве не гарантируется.

##### 2.2.2. Используемый инструментарий

Для создания приложения потребуются следующие инструментальные средства разработки:

1. Android Studio — это интегрированная среда разработки (IDE) для работы с платформой Android;
2. Android Virtual Device – программа для эмулирования Androidприложения на компьютере для предварительных тестов.

### 2.3. Разработка Android - приложения

Для начала, после создания проекта в первую очередь надо создать макет приложения, как оно будет выглядеть на устройстве после его активации. На первое время нам нужен ишь черновой вариант, потому, особо не приукрашивая внешний вид, просто добавим кнопку для начала распознавания и несколько элементов для вывода результатов. Таким образом начальный макет-шаблон будет выглядеть так, как показано на рисунке 6.

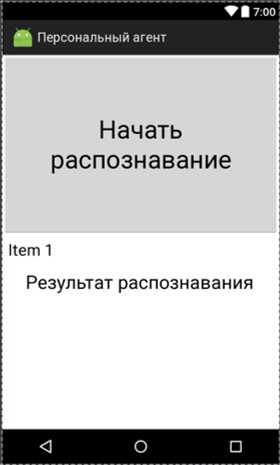


Рисунок 6 – образец начального макета Android – приложения

Рассмотрим необходимые нам классы для реализации дальнейшего кода программы:

* android.content.ActivityNotFoundException – класс выдает исключение в том случае, когда невозможен вызов запрашиваемого Activity;
* android.content.ContentResolver – класс предоставляет доступ к содержимому устройства;
* android.content.Intent – класс позволяет использовать намерения, которые передают задание новых событий в приложении и внешних программах;
* android.database.Cursor – интерфейс, который обеспечивает доступ к результирующему набору с помощью запроса к базе данных;
* android.net.Uri – класс позволяет использовать идентификатор, определяющий некоторый абстрактный или физический ресурс;
* android.os.Bundle – хранит любые примитивные типы, а так же

String и другие типы наследующие класс Parcel или интерфейс Parcelable, после чего позволяет по ключу обращаться к ним:

* android.provider.CalendarContract – через этот класс получаем разрешение на использование данных календаря из приложения;
* android.provider.ContactsContract – через этот класс получаем разрешение на использование данных о контактах в телефонной книге из приложения;
* android.speech.RecognizerIntent – класс включает поддержку распознавания речи для запуска через Intent;
* android.support.v7.app.AppCompatActivity – базовый класс для Activity;
* android.telephony.SmsManager – класс управляет smsоперациями, такими как отправка данных или сообщений;
* android.view.View – класс является основой построения пользовательского интерфейса;
* android.widget.ImageButton – класс позволяет отображать кнопку в виде картинки, которую пользователь может нажать;
* android.widget.TextView – позволяет отображать текст пользователю и, при необходимости, редактировать его;
* android.widget.Toast – отображает пользователю краткий текст, например ошибку;
* java.util.ArrayList – массив с изменяемым значением для создания списка в интерфейсе;
* java.util.GregorianCalendar – грегорианский календарь является подклассом календаря, который отвечает за стандартный ид календаря, используемый в большинстве стран мира;
* java.util.Locale – объект представляет собой какую-то определенную географическую, политическую или культурную область.

Помимо импорта данных классов в манифесте приложения так же надо внести некоторые исправления, а именно прописать доступ приложения к внутренним функциям устройства:

* чтению и записи в календаре

(android:name="android.permission.READ\_CALENDAR" android:name="android.permission.WRITE\_CALENDAR");

* набору и отправке смс

(android:name="android.permission.SEND\_SMS");

* телефонной книге

(android:name="android.permission.READ\_CONTACTS");

* набору телефонного номера

(android:name="android.permission.CALL\_PHONE").

Внеся данные изменения можно переходить к непосредственной реализации самого android-приложения.

# 3. ЧАСТЬ РЕАЛИЗАЦИИ

Android-приложение реализовано на языке Java с использованием xmlразметки на Andoid Studio 2.3.2.

### 3.1. Реализация xml-разметки

Конечный результат программы должен быть не только функциональным, но и внешне приятным, именно поэтому для окончательного варианта вносим кое-какие изменения в разметке, а именно:

* заменим громоздкую кнопку на более малый еѐ аналог, а также добавим ей иконку в виде микрофона для большей наглядности;
* надпись с кнопки перенесем немного ниже самой кнопки;
* результаты распознавания будем выводить так же по середине экрана;
* заменим стандартный белый фон на более красочный с помощью градиента.

В итоге готовое приложение у нас приняло вид, указанный на рисунке

7.

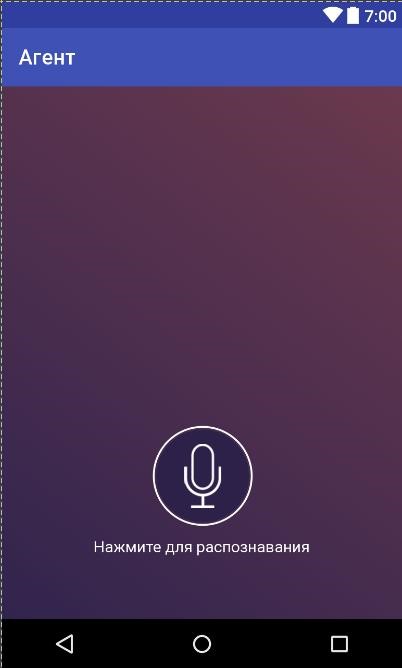


Рисунок 7 – конечный результат внешнего вида приложения

### 3.2. Реализация MainActivity.java

Тут у нас находятся все импортированные библиотеки, которые нам необходимы, а также главный класс, содержащий методы реализации программы. Тут же расположены задействованные в реализации поля, рассмотрим их:

* private TextView txtSpeechInput – поле где появляются произнесенные слова;
* private ImageButton btnSpeak – кнопка запуска распознавания;
* private final int REQ\_CODE\_SPEECH\_INPUT = 1000 – длительность в миллисекундах на возможность произнести команду;
* private boolean b\_number = false, b\_sms = false, b\_text = false, b\_calend = false, b\_calend\_text = false – флаги для проверки того какое действие требуется выполнить при произнесении команды;  private String text, dest – текст сообщения и получатель.

##### 3.1.1. Метод onCreate

В этом методе происходит построение приложения использую xmlразметку:

* txtSpeechInput = (TextView) findViewById(R.id.txtSpeechInput) – получаем идентификатор объекта типа текстового поля;
* btnSpeak = (ImageButton) findViewById(R.id.btnSpeak) – получаем идентификатор объекта типа кнопки;
* btnSpeak.setOnClickListener(new View.OnClickListener() – функция срабатывает нажатия кнопки, вызывает распознавание речи.

##### 3.1.2. Функция promptSpeechInput()

В данной функции происходит вывод текста для взаимодействия с пользователем, а также сам процесс распознавания речи:

* Intent intent = new Intent

(RecognizerIntent.ACTION\_RECOGNIZE\_SPEECH) – задается новое намерение с распознаванием речи;

* intent.putExtra(RecognizerIntent.EXTRA\_LANGUAGE\_MODEL, RecognizerIntent.LANGUAGE\_MODEL\_FREE\_FORM); intent.putExtra(RecognizerIntent.EXTRA\_LANGUAGE,

Locale.getDefault()) – задается модель распознавания слов связанных с событиями в телефоне пользователя;

* intent.putExtra(RecognizerIntent.EXTRA\_PROMPT, getString(R.string.speech\_prompt)) – выводит на экран пользователя стартовое выражение о готовности к получению команды;
* try {startActivityForResult(intent, REQ\_CODE\_SPEECH\_INPUT) – пробуем запустить распознавание, если устройство поддерживает его;
* catch (ActivityNotFoundException a) {

Toast.makeText(getApplicationContext(), getString(R.string.speech\_not\_supported),

Toast.LENGTH\_SHORT).show()} – иначе ловим ошибку.

##### 3.1.3. Функция onActivityResult()

Эта самая основная функция данной программы, тут мы получаем результаты распознавания и распределяем их по соответствующим функциям в зависимости от полученной команды:

* if (resultCode == RESULT\_OK && null != data) – проверяем, что получили не пустые данные;
* ArrayList <String> result = data.getStringArrayListExtra(RecognizerIntent.EXTRA\_RESULTS); text = result.get(0) – записываем результаты в список, после чего получаем последний распознанный текст.
* if (!b\_number && !b\_sms && !b\_text && !b\_calend &&

!b\_calend\_text) txtSpeechInput.setText("Команда '" + text + "' неверна. Попробуйте еще раз."); else txtSpeechInput.setText(text) – проверяем, является ли распознанное слово какой-либо командой, если таковой не является, то выводит ошибку, как на рисунке 8, иначе выводит нашу команду и выполняет

дальнейшие действия в зависимости от команды;

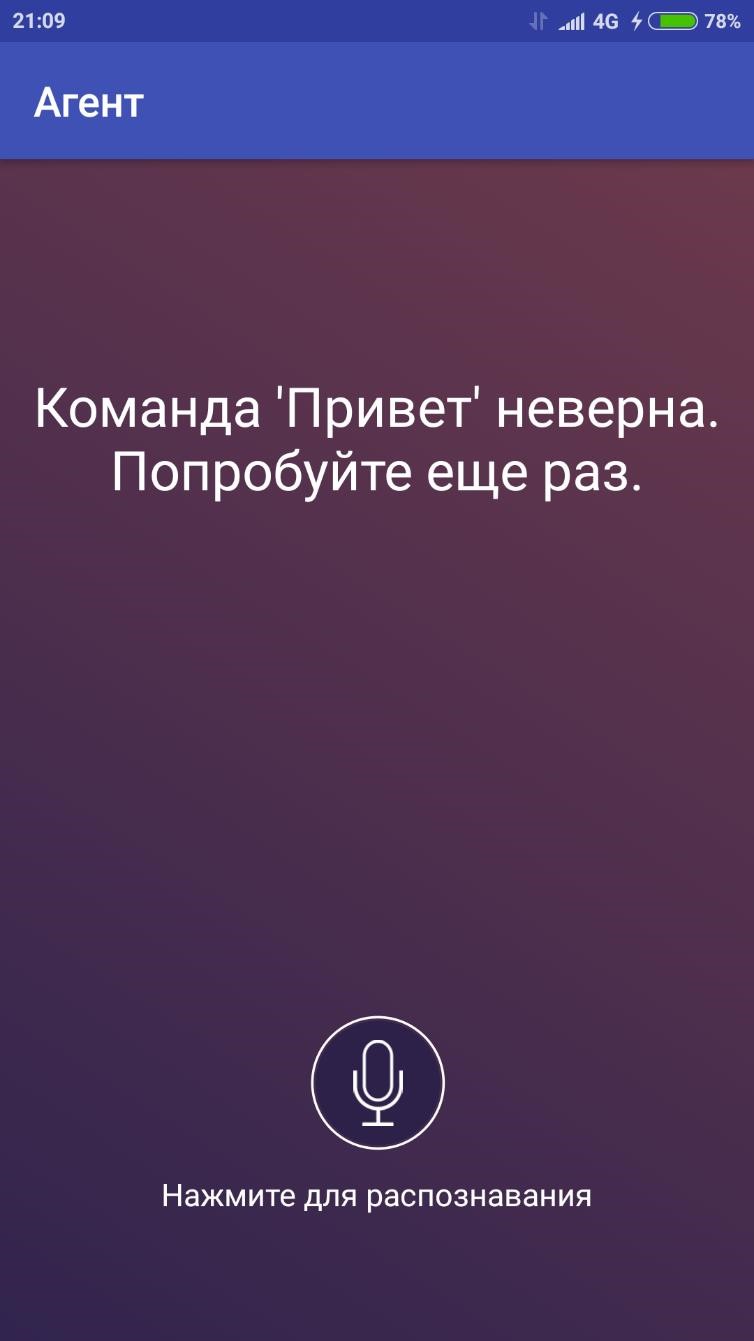


Рисунок 8 – пример ввода неверной команды

* если получена одна из команд «написать сообщение» или «отправить сообщение», программа отмечает b\_sms = true и заново запускает распознавание;
* если b\_sms = true мы сохраняем имя получателя и отмечаем b\_text = true, после чего снова идет распознавание;
* если b\_sms = true и b\_text = true проверяем, правильно ли продиктован номер, если да, пробуем найти такого абонента в телефонной книге, если такового нет, то отправка происходит просто по номеру, который был продиктован ранее, после чего снова помечаем b\_sms = false и b\_text = false;

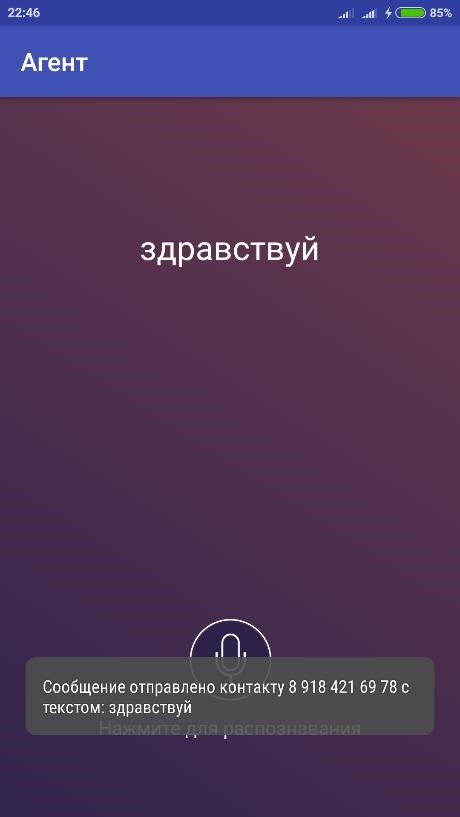
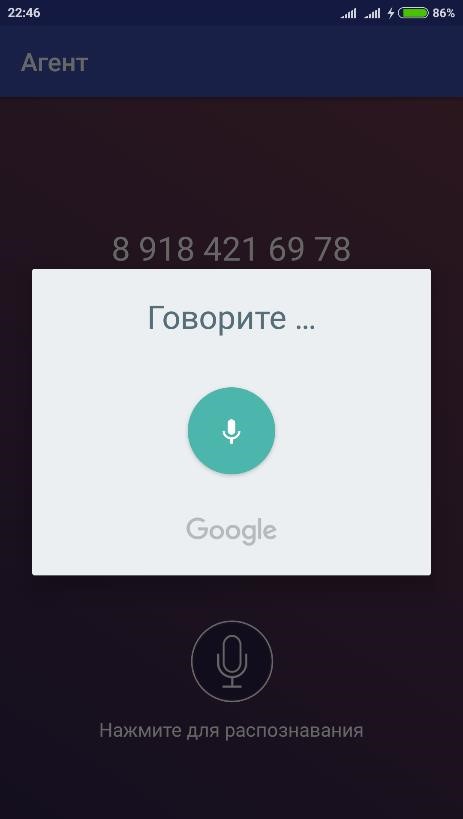
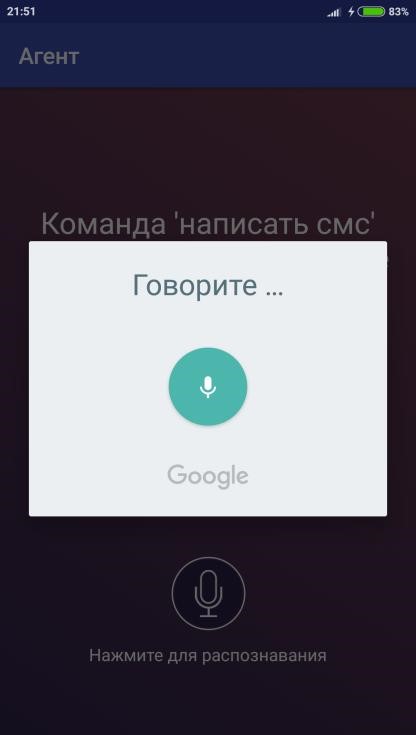


Рисунок 9 – поэтапный пример работы отправки sms-сообщения

* если получена одна из команд «позвонить абоненту» или «набрать абонента», программа отмечает b\_number = true и заново запускает распознавание;
* если b\_number = true пробуем найти такого абонента в телефонной книге, если такового нет, то происходит просто вызов по номеру, который был продиктован ранее, после чего снова помечаем b\_number = false;

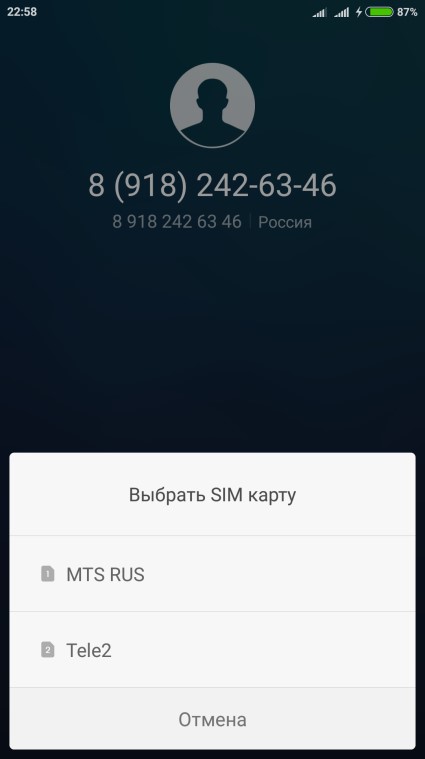
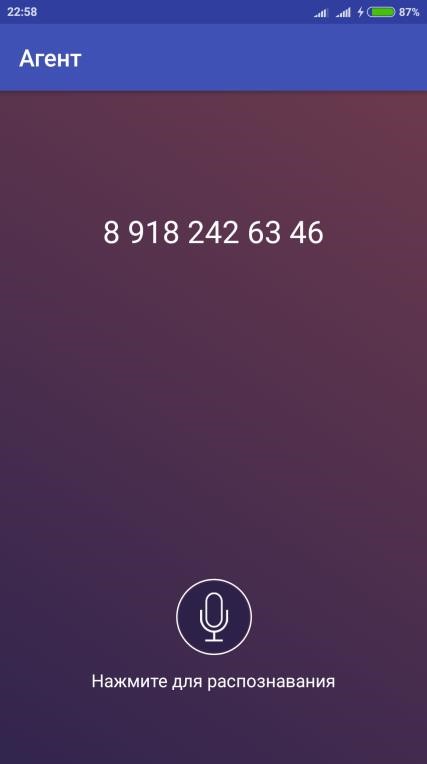
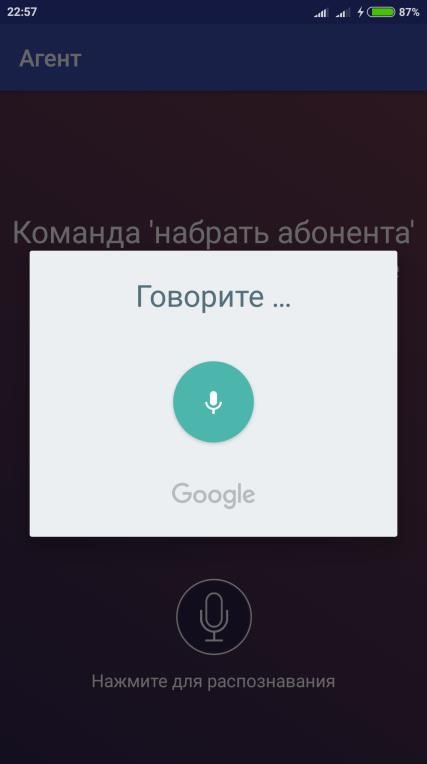


Рисунок 10 – поэтапный пример работы набора номера

* если получена одна из команд «поставить напоминание» или «оставить заметку», программа отмечает b\_calend = true и заново запускает распознавание;
* если b\_calend = true мы сохраняем сообщение для календаря и отмечаем b\_calend\_text = true, после чего снова идет распознавание;
* если b\_calend = true и b\_calend\_text = true, мы разделяем дату на день, месяц и год, и вызываем функцию добавления события в календарь, после чего снова помечаем b\_calend = false и b\_calend\_text = false;

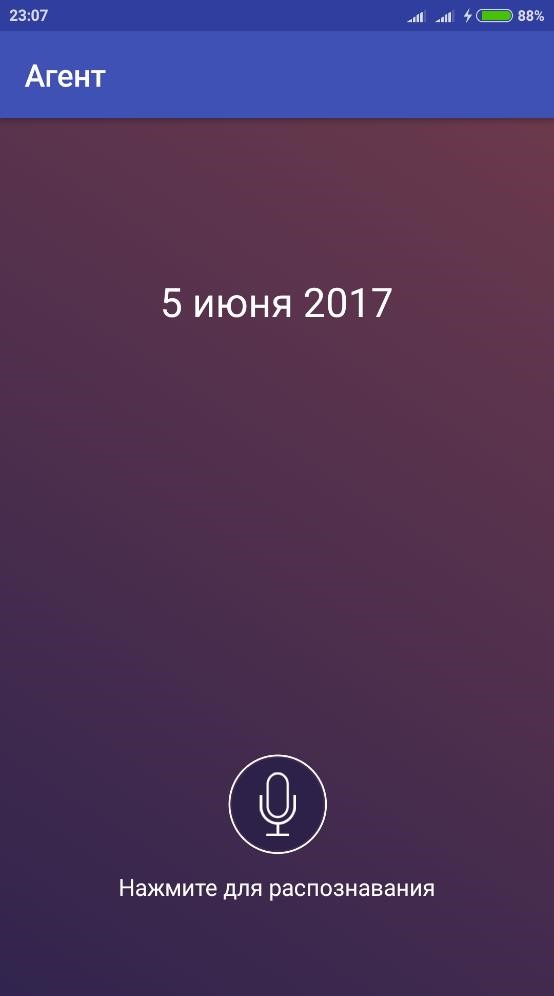
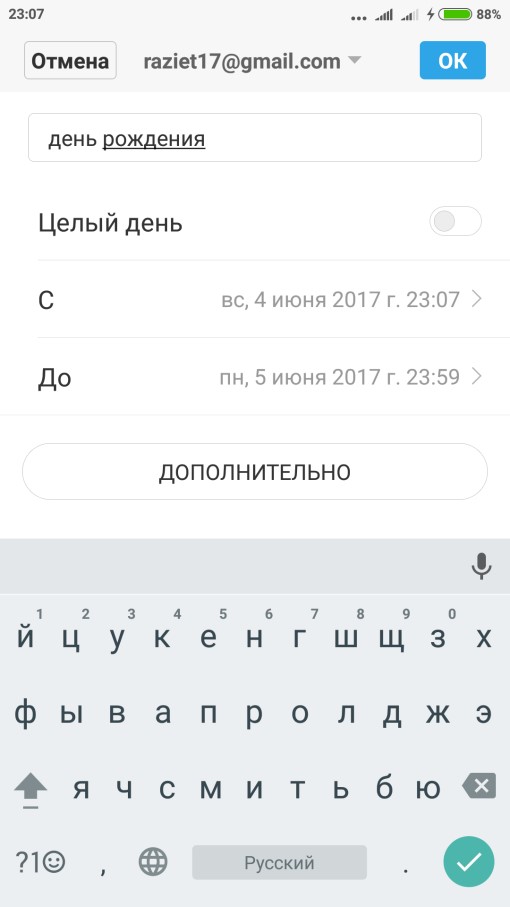
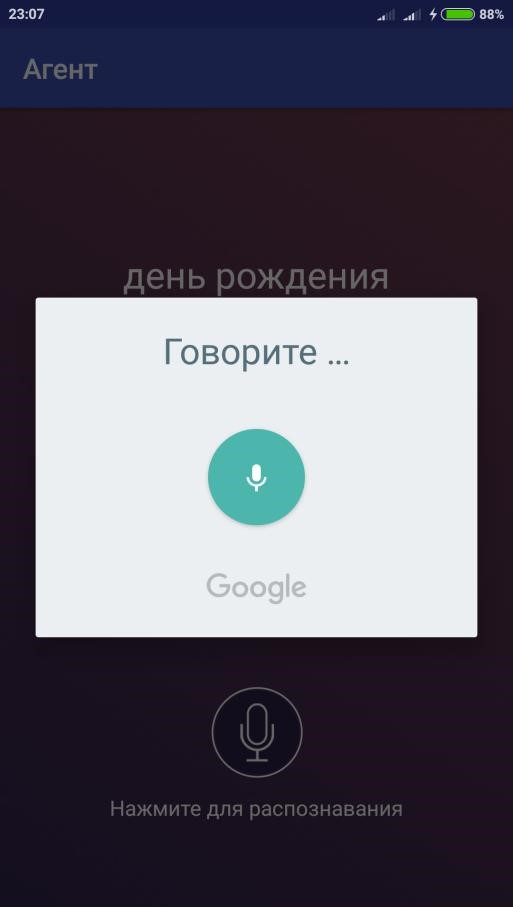
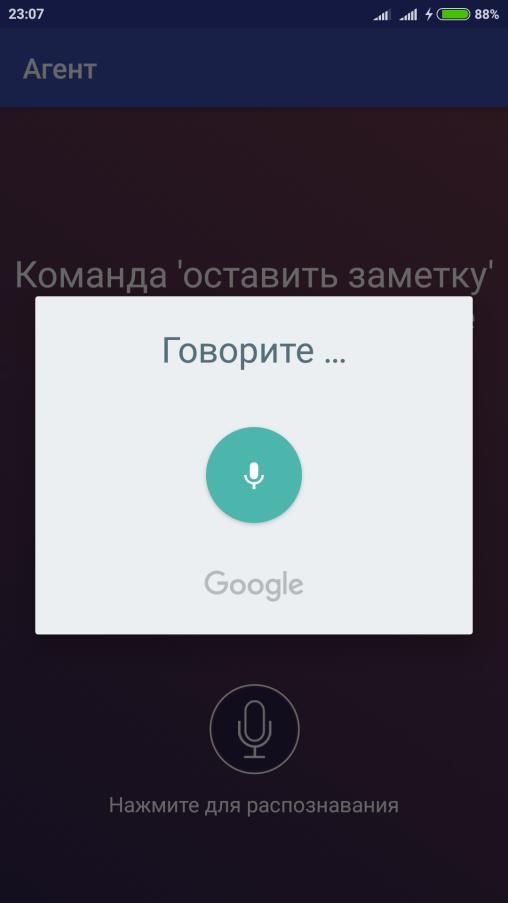


Рисунок 11 – поэтапный пример работы составления заметки в календаре

##### 3.1.4. Функция GetNumberContact ()

В этой функции получаем имя контакта по его номеру телефона, если он записан в телефонной книге для звонка этому абоненту:

* ContentResolver cr = getContentResolver(); Cursor cursor = cr.query String contactId, number, name, findName; Cursor phones; – создаем курсор для прохода по всем контактам;
* findName = nameContact.replace('ѐ', 'е'); - заменяем для простоты поиска все буквы ѐ на е;
* while (cursor.moveToNext()) – идем циклом по всем контактам;
* contactId=cursor.getString(cursor.getColumnIndex(ContactsContract. Contacts.\_ID));

phones=cr.query(ContactsContract.CommonDataKinds.Phone.CONT ENT\_URI,null,ContactsContract.CommonDataKinds.Phone.CONTA CT\_ID + " = " + contactId, null, null);phones.moveToNext() – получаем id абонента из телефонной книги;

* number=phones.getString(phones.getColumnIndex(ContactsContract. CommonDataKinds.Phone.NUMBER)) – получаем номер абонента из телефонной книги;
* name=phones.getString(phones.getColumnIndex(ContactsContract.C ommonDataKinds.Phone.DISPLAY\_NAME)) – получаем имя абонента;
* if (name.equalsIgnoreCase(findName)) { dialPhoneNumber(number);} – если имя, найденное в телефонной книге, совпадает с тем именем, что было продиктовано для набора номера, производим его вызов.

##### 3.1.5. Функция GetNumberSms ()

В этой функции получаем имя контакта по его номеру телефона, если он записан в телефонной книге для отправки ему сообщения. Функция практически идентична функции GetNumberContact(), отличием является лишь конечный вызов функции SendSMS() вместо dialPhoneNumber().

Если все данные введены верно, то программа пишет, что на номер такого-то абонента было отправлено соответствующее сообщение.

##### 3.1.6. Функция addEvent()

Функция отвечает, как можно понять по названию, за добавление какого-либо события в календарь:

* первым делом проверяем, совпадает ли дата напоминания с нынешним числом, если да, то напоминание активируется сразу и будет висеть в строке состояния;
* если дата отличается от нынешней, то задаем новое событие на нужную дату, которую мы разбили до этого, после чего на нее записываем интересующую нас заметку.

##### ЗАКЛЮЧЕНИЕ

В результате проведения работы были решены следующие задачи:

* изучена система распознавания речи на устройстве Anroid;
* спроектировано и написано Android-приложение с

распознаванием речи с учетом соответствующих изученных материалов;

* были реализованы такие функции приложения, как набор абонента, отправка сообщения на номер, а также добавление заметки в календарь;
* приложение было протестировано для соответствия поставленной в данной работе задаче и в данный момент готово к использованию.

# СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Davies, K.H., Biddulph, R. and Balashek, S. (1952) Automatic Speech Recognition of Spoken Digits, J. Acoust. Soc. Am. 24 (6) pp. 637—642
2. Чесебиев И.А. Компьютерное распознавание и порождение речи // И.А. Чесебиев. – М.: Спорт и культура, 2008 – 128 с.
3. Чекмарев А. Речевые технологии – проблемы и перспективы. // Компьютерра, №49 с. 26-43, 1997 г.
4. Фланаган Дж.Л. Анализ, синтез и восприятие речи / пер. с англ.

А. А. Пирогова. М.: Связь, 1968. 397 с.

1. Михайлов В. Г., Златоустова Л. В. Измерение параметров речи.

М.: Радио и связь, 1987. 168 с.

1. Вишнякова О. А., Лавров Д. Н. Применение преобразования Гильберта-хуанга к задаче сегментации речи // Математические структуры и моделирование. 2011. вып. 24. С. 12–18
2. О. Н. Граничин, Д. А. Дыдычкин, А. Н. Терехов

«Рандомизированный алгоритм стохастической аппроксимации в задаче распознавания отдельных слов речи», Санкт-Петербург, 2006 г. 41 с.

# ПРИЛОЖЕНИЕ А Код программы

package com.dreksers.speechrecognition;

import android.content.ActivityNotFoundException; import android.content.ContentResolver; import android.content.Intent; import android.database.Cursor; import android.net.Uri; import android.os.Bundle;

import android.provider.CalendarContract; import android.provider.ContactsContract; import android.speech.RecognizerIntent; import android.support.v7.app.AppCompatActivity; import android.telephony.SmsManager; import android.view.View; import android.widget.ImageButton; import android.widget.TextView;

import android.widget.Toast;

import java.util.ArrayList; import java.util.GregorianCalendar; import java.util.Locale;

public class MainActivity extends AppCompatActivity {

private TextView txtSpeechInput;

private ImageButton btnSpeak;

private final int REQ\_CODE\_SPEECH\_INPUT = 100; private boolean b\_number = false, b\_sms = false, b\_text = false,

b\_calend = false, b\_calend\_text = false; private String text, dest;

@Override

protected void onCreate(Bundle savedInstanceState) { super.onCreate(savedInstanceState); setContentView(R.layout.*activity\_main*);

txtSpeechInput = (TextView) findViewById(R.id.*txtSpeechInput*);

btnSpeak = (ImageButton) findViewById(R.id.*btnSpeak*);

btnSpeak.setOnClickListener(new View.OnClickListener() { @Override public void onClick(View v) {

promptSpeechInput();

}

});

}

private void promptSpeechInput() {

Intent intent = new

Intent(RecognizerIntent.*ACTION\_RECOGNIZE\_SPEECH*); intent.putExtra(RecognizerIntent.*EXTRA\_LANGUAGE\_MODEL*, RecognizerIntent.*LANGUAGE\_MODEL\_FREE\_FORM*);

intent.putExtra(RecognizerIntent.*EXTRA\_LANGUAGE*, Locale.*getDefault*()); intent.putExtra(RecognizerIntent.*EXTRA\_PROMPT*,

getString(R.string.*speech\_prompt*)); try {

startActivityForResult(intent, REQ\_CODE\_SPEECH\_INPUT);

} catch (ActivityNotFoundException a) { Toast.*makeText*(getApplicationContext(), getString(R.string.*speech\_not\_supported*),

Toast.*LENGTH\_SHORT*).show();

}

}

public void dialPhoneNumber(String phoneNumber) { Intent intent = new Intent(Intent.*ACTION\_CALL*); intent.setData(Uri.*parse*("tel:" + phoneNumber)); if (intent.resolveActivity(getPackageManager()) != null) { startActivity(intent);

}

}

public void GetNumberContact(String nameContact) {

ContentResolver cr = getContentResolver();

Cursor cursor = cr.query(ContactsContract.Contacts.*CONTENT\_URI*, null, null, null, null);

String contactId, number, name, findName; Cursor phones;

findName = nameContact.replace('ѐ', 'е'); while (cursor.moveToNext()) {

contactId =

cursor.getString(cursor.getColumnIndex(ContactsContract.Contacts.*\_ID*)); phones = cr.query(ContactsContract.CommonDataKinds.Phone.*CONTENT\_URI*, null, ContactsContract.CommonDataKinds.Phone.*CONTACT\_ID* + " = " + contactId, null, null); phones.moveToNext(); number =

phones.getString(phones.getColumnIndex(ContactsContract.CommonDataKinds.P hone.*NUMBER*));

name =

phones.getString(phones.getColumnIndex(ContactsContract.CommonDataKinds.P hone.*DISPLAY\_NAME*));

if (name.equalsIgnoreCase(findName)) { dialPhoneNumber(number);

break;

}

phones.close();

}

cursor.close();

}

public void GetNumberSms(String nameContact, String message) {

ContentResolver cr = getContentResolver();

Cursor cursor = cr.query(ContactsContract.Contacts.*CONTENT\_URI*, null, null, null, null);

String contactId, number, name, findName; Cursor phones;

findName = nameContact.replace('ѐ', 'е'); while (cursor.moveToNext()) {

contactId =

cursor.getString(cursor.getColumnIndex(ContactsContract.Contacts.*\_ID*)); phones =

cr.query(ContactsContract.CommonDataKinds.Phone.*CONTENT\_URI*, null, ContactsContract.CommonDataKinds.Phone.*CONTACT\_ID* + " = " + contactId, null, null); phones.moveToNext(); number =

phones.getString(phones.getColumnIndex(ContactsContract.CommonDataKinds.P hone.*NUMBER*));

name =

phones.getString(phones.getColumnIndex(ContactsContract.CommonDataKinds.P hone.*DISPLAY\_NAME*));

if (name.equalsIgnoreCase(findName)) { SendSMS(number, message); break;

}

phones.close();

}

cursor.close();

}

public void SendSMS(String dest\_name, String message) { SmsManager smsManager = SmsManager.*getDefault*(); try {

smsManager.sendTextMessage(dest\_name, null, message, null, null); Toast.*makeText*(getApplicationContext(), "Сообщение отправлено контакту " +

dest + " с текстом: " + text, Toast.*LENGTH\_LONG*).show();

} catch (Exception e) {

Toast.*makeText*(getApplicationContext(), "Сообщение не отправлено", Toast.*LENGTH\_LONG*).show();

}

}

public void addEvent(String message, int day, int month, int year) { GregorianCalendar calDate = new GregorianCalendar(year, month, day); long beginTime = calDate.getTimeInMillis(); if ((calDate.get(GregorianCalendar.*DAY\_OF\_MONTH*) == day) &&

(calDate.get(GregorianCalendar.*MONTH*) == month) &&

(calDate.get(GregorianCalendar.*YEAR*) == year)) { GregorianCalendar calDate2 = new GregorianCalendar(); beginTime = calDate2.getTimeInMillis();

}

Intent intent = new

Intent(Intent.*ACTION\_INSERT*).setData(CalendarContract.Events.*CONTENT\_URI*

)

.putExtra(CalendarContract.Events.*TITLE*, message)

.putExtra(CalendarContract.*EXTRA\_EVENT\_BEGIN\_TIME*, beginTime)

.putExtra(CalendarContract.*EXTRA\_EVENT\_END\_TIME*, calDate.getTimeInMillis() +

23 \* 60 \* 60 \* 1000 + 59 \* 60 \* 1000 + 59 \* 1000); if (intent.resolveActivity(getPackageManager()) != null) { startActivity(intent);

}

}

public int month(String m) { int numberMonth = 0; if (m.equals("января")) numberMonth = 0; else if (m.equals("февраля")) numberMonth = 1; else if (m.equals("марта")) numberMonth = 2; else if (m.equals("апреля")) numberMonth = 3; else if (m.equals("мая")) numberMonth = 4; else if (m.equals("июня")) numberMonth = 5; else if (m.equals("июля")) numberMonth = 6; else if (m.equals("августа")) numberMonth = 7; else if (m.equals("сентября")) numberMonth = 8; else if (m.equals("октября")) numberMonth = 9; else if (m.equals("ноября")) numberMonth = 10; else if (m.equals("декабря")) numberMonth = 11;

return numberMonth;

}

@Override

protected void onActivityResult(int requestCode, int resultCode, Intent data) { super.onActivityResult(requestCode, resultCode, data); switch (requestCode) {

case REQ\_CODE\_SPEECH\_INPUT: {

if (resultCode == *RESULT\_OK* && null != data) {

ArrayList<String> result =

data.getStringArrayListExtra(RecognizerIntent.*EXTRA\_RESULTS*); text = result.get(0);

if (!b\_number && !b\_sms && !b\_text && !b\_calend &&

!b\_calend\_text) txtSpeechInput.setText("Команда '" + text + "' неверна. Попробуйте еще раз.");

else txtSpeechInput.setText(text); if (b\_calend && b\_calend\_text) { String[] s = text.split(" "); int day = Integer.*parseInt*(s[0]); int month = month(s[1]); int year = Integer.*parseInt*(s[2]); addEvent(dest, day, month, year);

b\_calend = false;

b\_calend\_text = false;

}

if (b\_calend) { dest = text; promptSpeechInput();

b\_calend\_text = true;

}

if (b\_number) {

GetNumberContact(text); dialPhoneNumber(text); b\_number = false;

}

if (b\_sms && b\_text) { GetNumberSms(dest, text); SendSMS(dest, text); b\_text = false; b\_sms = false;

}

if (b\_sms) { dest = text; promptSpeechInput();

b\_text = true;

}

if (text.equals("написать сообщение") || text.equals("отправить сообщение") ||

text.equals("отправить смс") || text.equals("написать смс") || text.equals("отправить SMS") || text.equals("написать SMS") || text.equals("Написать сообщение") || text.equals("Отправить сообщение") ||

text.equals("Отправить смс") || text.equals("Написать смс") || text.equals("Отправить SMS") || text.equals("Написать SMS"))

{

b\_sms = true;

promptSpeechInput();

}

if (text.equals("позвонить абоненту") || text.equals("набрать абонента") ||

text.equals("позвонить по номеру") || text.equals("Позвонить абоненту") ||

text.equals("Набрать абонента") || text.equals("Позвонить по номеру")) {

b\_number = true;

promptSpeechInput();

}

if (text.equals("оставить заметку в календаре") ||

text.equals("оставить заметку") ||

text.equals("оставить напоминание") || text.equals("поставить напоминание") ||

text.equals("Оставить заметку в календаре") || text.equals("Оставить заметку") ||

text.equals("Оставить напоминание") || text.equals("Поставить напоминание")) {

b\_calend = true;

promptSpeechInput();

} } break;

}

}

}

}

# ПРИЛОЖЕНИЕ В Разметка приложения

<?xml version="1.0" encoding="utf-8"?>

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android" xmlns:tools="http://schemas.android.com/tools" android:layout\_width="match\_parent" android:layout\_height="match\_parent" android:background="@drawable/bg\_gradient"

android:orientation="vertical">

<TextView android:id="@+id/txtSpeechInput" android:layout\_width="wrap\_content" android:layout\_height="wrap\_content" android:layout\_alignParentTop="true" android:layout\_centerHorizontal="true" android:layout\_marginTop="100dp" android:gravity="center" android:textColor="@color/white" android:textSize="26dp"

android:textStyle="normal" />

<LinearLayout android:layout\_width="wrap\_content" android:layout\_height="wrap\_content" android:layout\_alignParentBottom="true" android:layout\_centerHorizontal="true" android:layout\_marginBottom="60dp"

android:gravity="center"

android:orientation="vertical">

<ImageButton

android:id="@+id/btnSpeak" android:layout\_width="wrap\_content" android:layout\_height="wrap\_content" android:background="@null"

android:src="@drawable/ico\_mic" />

<TextView android:layout\_width="wrap\_content" android:layout\_height="wrap\_content" android:layout\_marginTop="10dp" android:text="@string/tap\_on\_mic" android:textColor="@color/white" android:textSize="15dp" android:textStyle="normal" />

</LinearLayout>

</RelativeLayout>

<?xml version="1.0" encoding="utf-8"?>

<resources>

<color name="colorPrimary">#3F51B5</color>

<color name="colorPrimaryDark">#303F9F</color>

<color name="colorAccent">#FF4081</color>

<color name="white">#ffffff</color>

<color name="bg\_gradient\_start">#31244e</color>

<color name="bg\_gradient\_end">#6b394c</color>

</resources>

<resources>

<string name="app\_name">Агент</string>

<string name="speech\_prompt">Говорите &#8230;</string>

<string name="speech\_not\_supported">Извините! Ваше устройство не поддерживает распознавание речи</string>

<string name="tap\_on\_mic">Нажмите для распознавания</string> </resources>

<resources>

<!-- Base application theme. -->

<style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">

<!-- Customize your theme here. -->

<item name="colorPrimary">@color/colorPrimary</item>

<item name="colorPrimaryDark">@color/colorPrimaryDark</item>

<item name="colorAccent">@color/colorAccent</item>

</style>

</resources>

# ПРИЛОЖЕНИЕ С Manifest приложения

<?xml version="1.0" encoding="utf-8"?>

<manifest xmlns:android="http://schemas.android.com/apk/res/android" package="com.dreksers.speechrecognition">

<uses-permission android:name="android.permission.CALL\_PHONE" />

<uses-permission android:name="android.permission.READ\_CONTACTS" />

<uses-permission android:name="android.permission.SEND\_SMS"/>

<uses-permission android:name="android.permission.READ\_CALENDAR" />

<uses-permission android:name="android.permission.WRITE\_CALENDAR" />

<application

android:allowBackup="true" android:icon="@mipmap/ic\_launcher" android:label="@string/app\_name"

android:roundIcon="@mipmap/ic\_launcher\_round"

android:supportsRtl="true"

android:theme="@style/AppTheme">

<activity android:name=".MainActivity">

<intent-filter>

<action android:name="android.intent.action.MAIN" />

<category android:name="android.intent.category.LAUNCHER" />

</intent-filter>

</activity>

</application>

</manifest>