ГОСУДАРСТВЕННЫЙ КОМИТЕТ СВЯЗИ, ИНФОРМАТИЗАЦИИ И ТЕЛЕКОММУНИКАЦИОННЫХ ТЕХНОЛОГИЙ РЕСПУБЛИКИ УЗБЕКИСТАН

ТАШКЕНТСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

**АБДУЛЛАЕВ ОЛИМЖОН МАХМУДЖОНОВИЧ**

# Применение методов интеллектуального анализа данных при создании инструменталъных средств для составления базы данных фонем узбекского языка

5А330601-Программный инжиниринг

Диссертация

на соискание академической степени магистра

Научный руководитель

Хан И.В.

Ташкент – 2015

Оглавление

Введение

В последнее время наблюдается стремительное развитие информационных технологий. Одним из приоритетных направлений исследований в данной области являются задачи хранения, обработки и передачи мультимедиа данных. К сожалению, до сих пор во многих задачах анализа мультимедиа данных компьютер так и не смог окончательно заменить эксперта. Это такие задачи, как синхронный перевод, автоматическая сегментация изображений и видеопоследовательностей, автоматическая стенография. Одной из основных задач обработки мультимедиа информации является задача распознавания и анализа естественной речи человека.

В задачу анализа речи входит широкий спектр задач. Традиционно их подразделяют на три подкласса: задачи идентификации, классификации и диагностики. К задачам идентификации относят задачи верификации и идентификации дикторов. К задачам классификации относят задачи распознавания ключевых слов, распознавания слитной речи и задачи семантического анализа речи. К классу задач диагностики относят задачи определения психофизического состояния диктора. Во многих из выше перечисленных задач в последние годы был достигнут значительный прогресс. Скажем, алгоритмы идентификации или верификации дикторов широко используются при проведении криминалистических процедур или для разграничения прав доступа, благодаря высокой точности разработанных методов.

По-прежнему сохраняет свою актуальность задача распознавания речи. Область применения полученных решений довольно обширна: автоматические стенографы, автоматизированные справочные терминалы с речевым управлением, синхронные переводчики, системы сжатия и передачи речевого сигнала с высоким качеством, системы сегментации, индексации и поиска мультимедиа информации.

**Актуальность темы.** Использование речевого интерфейса актуально для таких задач, как распознавание и анализ речи, аутентификация личности по голосу (например, в интегрированных системах безопасности), возможность голосового ответа на запрос информационной системы (например, о состоянии технологического процесса), повышение эффективности преобразователей «речь-текст». Голосовой интерфейс является одним из условий создания без барьерной среды для людей с ограниченными возможностями. Все большую актуальность приобретает использование речевых технологий для создания диалога человек-машина.

Основным направлением современных речевых технологий является разработка единых алгоритмов параметризации речевого сигнала, основанных на физической модели речевого тракта и природе речевого сигнала – определение существенных параметров и разработка математических моделей, применимых как для синтеза, так и для анализа речи. Определение отличительных существенных параметров осложняется тем, что речевой сигнал имеет двойственную природу: с одной стороны, это акустический сигнал, отражающий процесс распространения энергии акустических колебаний в упругой среде, с другой стороны, речевой сигнал несёт смысловую информацию, информацию об эмоциональном состоянии диктора, а также содержит индивидуальные параметры, позволяющие отличать одного диктора от другого. В настоящее время не существует единого мнения о природе формы спектра речевых сигналов, методах их анализа и синтеза.

Современное состояние развития науки и техники позволяет создавать системы, основанные на сложном математическом аппарате, но работающие в режиме реального времени, благодаря виртуальным сетям. Данный математический аппарат даёт возможность оперирования большими объёмами информации при его несоответствии акустической теории. При этом один и тот же математический аппарат использует разные данные в системах анализа и синтеза речи.

Наибольшей точностью описания речевого сигнала обладают математические модели, основанные на физике протекающих явлений, что приводит к требованиям высокой точности, компактности (в смысле параметризации) и адекватности математических моделей речевого сигнала акустической теории речи образования.

**Степень изученности исследования.** Исследованиям проблем распознавания речи занимаются: Вычислительный центр РАН (Ю. И. Журавлев, В. Я. Чучупал), Институт проблем передачи информации РАН (В. Н. Сорокин), Институт математики СО РАН и Новосибирский государственный университет (В. М. Величко), Московский государственный университет им. М. В. Ломоносова (О. Ф. Кривнова), МГТУ им. Н. Э. Баумана (Ю. Н. Жигулевцев), Московский энергетический институт (А. И. Евсеев), Московский государственный лингвистический университет (Р. К. Потапова), Московский технический университет связи и информатики (Ю. Н. Прохоров), Санкт-Петербургский государственный университет (В. И. Галунов), Санкт-Петербургский институт информатики и автоматизации РАН. В данной области ведут исследования такие компании как IBM, Philips, Dragon Systems, Cognitive Technologies, Истрасофт, Сакрамент и др., что говорит об ее актуальности.

**Цель исследования.** Разработка программного обеспечение для интеллектуальной формирование баз данных составляющей фонем узбекского языка на базе операционной системы Android.

**Для достижение поставленной цели в диссертации решаются следующие задачи:**

- изучить общих методов интеллектуального анализа данных при применение распознавание голосовых информаций;

- провести сравнительный анализ характеристик существующих систем распознавания речи для составление баз данных фонем узбекского языка;

- реализация системы преобразования речи в текст для автоматической добавление в базу данных на базе операционных систем Android;

- оценить качество распознавания разработанной системы.

**Методы исследования.** Методы цифровой обработки звуковых данных.

**Объект исследования:** база данных фонем узбекского языка.

**Предмет исследования:** методы и алгоритмы интеллектуального анализа данных, а также, средства разработки программы для преобразование речи в текст.

**Научная новизна.**

- предложен метод применения сервиса Google Speech Voice API для разработки программы составление баз данных фонем узбекского языка;

- разработан программное обеспечение для системы преобразования речи в текст на базе операционных систем Android;

**Практическая ценность.** Разработанное программное обеспечение позволяет автоматизировать процесс ввода текстовой информации, а также, проводить экспериментальные исследования в области распознавания речи.

**Структура и объем работы.** Диссертация состоит из введения, трёх глав, заключения и приложения, содержит список литературы из __ наименований. Работа изложена на __ страницах.

# I Глава. Изучение методов интеллектуального анализа данных при распознавание речи

## 1. Общие понятия интеллектуального анализа данных

Сегодня мы являемся свидетелями активного развития технологии интеллектуального анализа данных (ИАД), появление которой связано, в первую очередь, с необходимостью аналитической обработки сверхбольших объёмов информации, накапливаемой в современных хранилищах данных. Возможность использования хорошо известных методов математической статистики и машинного обучения для решения задач подобного рода открыло новые возможности перед аналитиками, исследователями, а также теми, кто принимает решения - менеджерами и руководителями компаний.

Сложность и разнообразие методов ИАД требуют создания специализированных средств конечного пользователя для решения типовых задач анализа информации в конкретных областях. Поскольку эти средства используются в составе сложных многофункциональных систем поддержки принятия решений, они должны легко интегрироваться в подобные системы. Одним из наиболее важных и перспективных направлений применения ИАД являются бизнес-приложения, поэтому опыт канадско-американской фирмы Cognos по реализации методов ИАД в составе интегрированных интеллектуальных систем поддержки принятия решений представляет интерес как для разработчиков, так и для пользователей.

Системы ИАД применяются в научных исследованиях и образовании, в работе правоохранительных органов, производстве, здравоохранении и многих других областях. Особенно широко технология ИАД используется в деловых приложениях.

**Интеллектуальный анализ данных** — одно из новых направлений искусственного интеллекта. Этот термин является кратким и весьма неточным переводом с английского языка терминов Data Mining и Knowledge Discovery in Databases (DM&KDD). Более точный перевод — «добыча данных» и «выявление знаний в базах данных».

**Data Mining** — это процесс обнаружения в сырых данных (row data) раннее неизвестных, нетривиальных, практически полезных, доступных интерпретации знаний (закономерностей), необходимых для принятия решений в различных сферах человеческой деятельности.

**Появление технологий DM&KDD** обусловлено накоплением огромных объёмов информации в компьютерных базах данных, которые стало невыгодно хранить и которыми стало трудно пользоваться традиционными способами. Последнее обстоятельство связано со стремительным развитием вычислительной техники и программных средств для представления и обработки данных. Большие объёмы накопленных данных постоянно приходится модифицировать из-за быстрой смены аппаратного и программного обеспечения БД, при этом неизбежны потери и искажение информации. Одним из средств для преодоления подобных трудностей является создание информационных хранилищ данных, доступ к которым не будет сильно зависеть от изменения данных во времени и от используемого программного обеспечения. Другой подход ориентирован на сжатие больших объёмов данных путём нахождения некоторых общих закономерностей (знаний) в накопленной информации. Оба направления актуальны с практической точки зрения. Второй подход более интересен для специалистов в области ИИ, так как связан с решением проблемы приобретения новых знаний. Следует заметить, что наиболее плодотворным является сочетание обоих направлений.

**Наличие хранилища данных** — необходимое условие для успешного проведения всего процесса KDD. Хранилищем данных называют предметно-ориентированное, интегрированное, привязанное ко времени, неизменяемое собрание данных, используемых для поддержки процесса принятия управленческих решений. Предметная ориентация означает, что данные объединены в категории и хранятся в соответствии с теми областями, которые они описывают, а не в соответствии с приложениями, которые их используют. Такой принцип хранения гарантирует, что отчёты,

сгенерированные различными аналитиками, будут опираться на одну и ту же совокупность данных. Привязанность ко времени означает, что хранилище можно рассматривать как собрание исторических данных, т.е. конкретные значения данных однозначно связаны с определёнными моментами времени. Атрибут времени всегда явно присутствует в структурах хранилищ данных. Данные, занесённые в хранилище, уже не изменяются в отличие от оперативных систем, где присутствуют только последние, постоянно изменяемые версии данных. Для хранилищ данных характерны операции добавления, а не модификации данных. Современные средства администрирования хранилищ данных обеспечивают эффективное взаимодействие с программным инструментарием DM и KDD. В качестве примера можно привести разработки компании SAS Institute: SAS Warehouse Administrator и SAS Enterprise Miner.

**Целью интеллектуального анализа данных** (англ. Data mining, другие варианты перевода - "добыча данных", "раскопка данных") является обнаружение неявных закономерностей в наборах данных. Как научное направление он стал активно развиваться в 90-х годах XX века, что было вызвано широким распространением технологий автоматизированной обработки информации и накоплением в компьютерных системах больших объёмов данных. И хотя существующие технологии позволяли, например, быстро найти в базе данных нужную информацию, этого во многих случаях было уже недостаточно. Возникла потребность поиска взаимосвязей между отдельными событиями среди больших объёмов данных, для чего понадобились методы математической статистики, теории баз данных, теории искусственного интеллекта и ряда других областей.

Классическим считается определение, данное одним из основателей направления Григорием Пятецким-Шапиро: **DataMining - исследование и обнаружение "машиной" (алгоритмами, средствами искусственного интеллекта) в сырых данных скрытых знаний, которые ранее не были**

**известны, нетривиальны, практически полезны, доступны для интерпретации.**

Учитывая разнообразие форм представления данных, используемых алгоритмов и сфер применения, интеллектуальный анализ данных может проводиться с помощью программных продуктов следующих классов:

- Специализированных "коробочных" программных продуктов для интеллектуального анализа;
- Математических пакетов;
- Электронных таблиц (и различного рода надстроек над ними);
- Средств интегрированных в системы управления базами данных (СУБД);
- Других программных продуктов.

В рамках данного курса нас в первую очередь будут интересовать средства, интегрированные с СУБД. В качестве примера можно привести СУБД Microsoft SQLServer и входящие в ее состав службы Analysis Services, обеспечивающие пользователей средствами аналитической обработки данных в режиме online (OLAP)и интеллектуального анализа данных, которые впервые появились в MS SQLServer 2000.

Не только Microsoft, но и другие ведущие разработчики СУБД имеют в своём арсенале средства интеллектуального анализа данных.

**Задачи интеллектуального анализа данных.** В ходе проведения интеллектуального анализа данных проводится исследование множества объектов (или вариантов).

В большинстве случаев его можно представить в виде таблицы, каждая строка которой соответствует одному из вариантов, а в столбцах содержатся значения параметров, его характеризующих. Зависимая переменная - параметр, значение которого рассматриваем как зависящее от других параметров (независимых переменных). Собственно эту зависимость и необходимо определить, используя методы интеллектуального анализа данных.

**Рассмотрим основные задачи интеллектуального анализа данных.** Задача классификации заключается в том, что для каждого варианта определяется категория или класс, которому он принадлежит. В качестве примера можно привести оценку кредитоспособности потенциального заёмщика: назначаемые классы здесь могут быть "кредитоспособен" и "некредитоспособен". Необходимо отметить, что для решения задачи необходимо, чтобы множество классов было известно заранее и было бы конечным и счётным.

Задача регрессии во многом схожа с задачей классификации, но в ходе её решения производится поиск шаблонов для определения числового значения. Иными словами, предсказываемый параметр здесь, как правило, число из непрерывного диапазона.

Отдельно выделяется задача прогнозирования новых значений на основании имеющихся значений числовой последовательности (или нескольких последовательностей, между значениями в которых наблюдается корреляция). При этом могут учитываться имеющиеся тенденции (тренды), сезонность, другие факторы. Классическим примером является прогнозирование цен акций на бирже.

Тут требуется сделать небольшое отступление. По способу решения задачи интеллектуального анализа можно разделить на два класса: обучение с учителем (от англ. Supervised learning) и обучение без учителя (от англ. unsupervised learning). В первом случае требуется обучающий набор данных, на котором создается и обучается модель интеллектуального анализа данных. Готовая модель тестируется и впоследствии используется для предсказания значений в новых наборах данных. Иногда в этом же случае говорят об управляемых алгоритмах интеллектуального анализа. Задачи классификации и регрессии относятся как раз к этому типу.

Во втором случае целью является выявление закономерностей имеющихся в существующем наборе данных. При этом обучающая выборка не требуется. В качестве примера можно привести задачу анализа

потребительской корзины, когда в ходе исследования выявляются товары, чаще всего покупаемые вместе. К этому же классу относится задача кластеризации.

Также можно говорить о классификации задач интеллектуального анализа данных по назначению, в соответствии с которой, они делятся на описательные (descriptive) и предсказательные (predictive). Цель решения описательных задач - лучше понять исследуемые данные, выявить имеющиеся в них закономерности, даже если в других наборах данных они встречаться не будут. Для предсказательных задач характерно то, что в ходе их решения на основании набора данных с известными результатами строится модель для предсказания новых значений.

Но вернёмся к перечислению задач интеллектуального анализа данных.

**Задача кластеризации** - заключается в делении множества объектов на группы (кластеры) схожих по параметрам. При этом, в отличие от классификации, число кластеров и их характеристики могут быть заранее неизвестны и определяться в ходе построения кластеров исходя из степени близости объединяемых объектов по совокупности параметров.

Другое название этой задачи - сегментация. Например, интернет-магазин может быть заинтересован в проведении подобного анализа базы своих клиентов, для того, чтобы потом сформировать специальные предложения для выделенных групп, учитывая их особенности.

Кластеризация относится к задачам обучения без учителя (или "неуправляемым" задачам).

Задача определения взаимосвязей, также называемая задачей поиска ассоциативных правил, заключается в определении часто встречающихся наборов объектов среди множества подобных наборов. Классическим примером является анализ потребительской корзины, который позволяет определить наборы товаров, чаще всего встречающиеся в одном заказе (или в одном чеке). Эта информация может потом использоваться при размещении

товаров в торговом зале или при формировании специальных предложений для группы связанных товаров.

Данная задача также относится к классу "обучение без учителя".

Анализ последовательностей или сиквенциальный анализ одними авторами рассматривается как вариант предыдущей задачи, другими - выделяется отдельно. Целью, в данном случае, является обнаружение закономерностей в последовательностях событий. Подобная информация позволяет, например, предупредить сбой в работе информационной системы, получив сигнал о наступлении события, часто предшествующего сбою подобного типа. Другой пример применения - анализ последовательности переходов по страницам пользователей web-сайтов.

Анализ отклонений позволяет отыскать среди множества событий те, которые существенно отличаются от нормы. Отклонение может сигнализировать о каком-то необычном событии (неожиданный результат эксперимента, мошенническая операция по банковской карте …) или, например, об ошибке ввода данных оператором.

В таблице 1.1 приведены примеры задач интеллектуального анализа данных из различных областей. Более подробно некоторые из них будут рассмотрены в последующих разделах курса.

Таблица 1.1. Примеры применения интеллектуального анализа данных

| | Информационные технологии | Торговля | Финансовая сфера |
|---|---|---|---|
| Классификация | | | Оценка кредитоспособности |
| Регрессия | | | Оценка допустимого кредитного лимита |
| Прогнозирование | | Прогнозирование продаж | Прогнозирование цен акции |
| Кластеризации | | Сегментация клиентов | Сегментация клиентов |
| Определения взаимосвязей | | Анализ потребительской корзины | |
| Анализ последовательностей | Анализ переходов по страницам web-сайта | | |
| Анализ отклонений | Обнаружение вторжений | | Выявление мошенничества |

| | | | |
|---|---|---|---|
| | в информационные системы | | с банковскими картами |

## 2. Исследование алгоритмов интеллектуального анализа данных. Службы Analysis Services

**Алгоритм интеллектуального анализа данных** — это набор эвристики и вычислений, который создаёт модель интеллектуального анализа данных из данных. Чтобы создать модель, алгоритм сначала анализирует предоставленные данные, осуществляя поиск определённых закономерностей и тенденций. Алгоритм использует результаты этого анализа для выбора оптимальных параметров создания модели интеллектуального анализа данных. Затем эти параметры применяются ко всему набору данных, чтобы выявить пригодные к использованию закономерности и получить подробную статистику.

Модель интеллектуального анализа данных, создаваемая алгоритмом из предоставленных данных, может иметь различные формы, включая следующие.

- Набор кластеров, описывающих связи вариантов в наборе данных.
- Дерево решений, которое предсказывает результат и описывает, какое влияние на этот результат оказывают различные критерии.
- Математическую модель, прогнозирующую продажи.
- Набор правил, описывающих группирование продуктов в транзакции, а также вероятности одновременной покупки продуктов.

Эти алгоритмы являются реализациями некоторых из наиболее популярных методов, используемых в интеллектуальном анализе данных. Все алгоритмы интеллектуального анализа данных Майкрософт настраиваются, они полностью программируются через API-интерфейсы или компоненты интеллектуального анализа данных служб SQL Server Integration Services.

Кроме того, поддерживается использование сторонних алгоритмов, соответствующих спецификации OLE DB для интеллектуального анализа данных. Имеется также возможность разрабатывать собственные алгоритмы, которые можно зарегистрировать в качестве служб, а затем использовать в платформе интеллектуального анализа данных SQL Server.

**Выбор правильного алгоритма** для использования в конкретной аналитической задаче может быть достаточно сложным. В то время как можно использовать различные алгоритмы для выполнения одной и той же задачи, каждый алгоритм выдает различный результат, а некоторые алгоритмы могут выдавать более одного типа результатов. Например, можно использовать алгоритм дерева принятия решений (Майкрософт) не только для прогнозирования, но также в качестве способа уменьшения количества столбцов в наборе данных, поскольку дерево принятия решений может идентифицировать столбцы, не влияющие на конечную модель интеллектуального анализа данных.

**Выбор алгоритма по типу.** Службы Службы Analysis Services включают следующие типы алгоритмов.

- Алгоритмы классификации осуществляют прогнозирование одной или нескольких дискретных переменных на основе других атрибутов в наборе данных.
- Регрессивные алгоритмы осуществляют прогнозирование одной или нескольких непрерывных переменных, например прибыли или убытков, на основе других атрибутов в наборе данных.
- Алгоритмы сегментации делят данные на группы или кластеры элементов, имеющих схожие свойства.
- Алгоритмы взаимосвязей осуществляют поиск корреляции между различными атрибутами в наборе данных. Наиболее частым применением этого типа алгоритма является создание правил взаимосвязи, которые могут использоваться для анализа потребительской корзины.

- Алгоритмы анализа последовательностей обобщают часто встречающиеся последовательности в данных, например поток данных в Интернете.

Однако ничто не заставляет пользователя ограничиваться одним алгоритмом в своих решениях. Опытные аналитики часто используют один алгоритм для выявления наиболее эффективных входных данных (то есть переменных), после чего применяют другой алгоритм для прогнозирования определённого результата на основе этих данных. Интеллектуальный анализ данных SQL Server позволяет на базе одной структуры интеллектуального анализа построить много моделей таким образом, что в рамках одного решения для интеллектуального анализа данных можно было использовать алгоритм кластеризации, модель дерева решений, а также модель упрощённого алгоритма Байеса для получения разных представлений данных. Несколько алгоритмов в одном решении также можно использовать для выполнения отдельных задач. Например, с помощью регрессии можно получать финансовые прогнозы, а с помощью алгоритма нейронной сети выполнять анализ факторов, влияющих на объем продаж.

Выбор алгоритма по задаче. Чтобы облегчить выбор алгоритмов для решения определённой задачи, в следующей таблице приведены типы задач, для решения которых обычно используется каждый алгоритм.

| Примеры задач | Подходящие алгоритмы Майкрософт |
|---|---|
| Прогнозирование дискретного атрибута<br>• Пометка клиентов из списка потенциальных покупателей как хороших и плохих кандидатов.<br>• Вычисление вероятности отказа сервера в течение следующих шести месяцев.<br>• Классификация вариантов развития болезней пациентов и исследование связанных факторов. | Алгоритм дерева принятия решений (Майкрософт)<br>Упрощенный алгоритм Байеса (Майкрософт)<br>Алгоритм кластеризации (Майкрософт)<br>Алгоритм нейронной сети (Майкрософт) |
| Прогнозирование непрерывного атрибута<br>• Прогноз продаж на следующий год.<br>• Прогноз количества посетителей сайта с учётом прошлых лет и сезонных тенденций.<br>• Формирование оценки риска с учётом | Алгоритм дерева принятия решений (Майкрософт)<br>Алгоритм временных рядов (Майкрософт)<br>Алгоритм линейной регрессии (Майкрософт) |

| | |
|---|---|
| демографии. | |
| Прогнозирование последовательности<br>• Анализ маршрута перемещения по веб-сайту компании.<br>• Анализ факторов, ведущих к отказу сервера.<br>• Отслеживание и анализ последовательностей действий во время посещения поликлиники с целью формулирования рекомендаций по общим действиям. | Алгоритм кластеризации последовательностей (Майкрософт) |
| Нахождение групп общих элементов в транзакциях.<br>• Использование анализа потребительской корзины для определения мест размещения продуктов.<br>• Выявление дополнительных продуктов, которые можно предложить купить клиенту.<br>• Анализ данных опроса, проведённого среди посетителей события, с целью выявления того, какие действия и стенды были связаны, чтобы планировать будущие действия. | Алгоритм взаимосвязей (Майкрософт)<br>Алгоритм дерева принятия решений (Майкрософт) |
| Нахождение групп схожих элементов<br>• Создание профилей рисков для пациентов на основе таких атрибутов, как демография и поведение.<br>• Анализ пользователей по шаблонам просмотра и покупки.<br>• Определение серверов, которые имеют аналогичные характеристики использования. | Алгоритм кластеризации (Майкрософт)<br>Алгоритм кластеризации последовательностей (Майкрософт) |

## 3. Применение методов ИАД при параметризации, анализа и классификации голосовой речи. Скрытие Марковские модели

Основой моделирования речевого сигнала на фонетическом уровне является построение иерархической структуры состоящей из элементов, которые получили название минимальных речевых единиц (МРЕ).

В большинстве случаев, в качестве таких единиц используются аллофоны, дифоны, тифоны, слоги и фонемы. Аллофон – набор звуков, имеющих одинаковое признаковое описание. Дифоны – переход между двумя аллофонами без их стационарных участков, чаще всего переход

согласный-гласный или гласный-согласный. Трифон – последовательность из трех аллофонов, позволяющая учитывать коартикуляционное воздействие предыдущего и последующего звуков на текущий звук. Фонема – совокупность аллофонов, имеющих одинаковые функции в речеобразовании и не несущие семантических различий. Слог – ядро гласного звука и функционально и формально связанные с ним соседние согласные звуки.

В качестве МРЕ могут быть так же использованы и слова. Но для распознавания русского языка использование слов в качестве МРЕ ведёт к большим расходом вычислительных ресурсов, в силу того, что слово в русском языке обладает порядка 100 словоформ, все из которых являются возможными МРЕ. Кроме того, для устойчивого распознавания в словаре для каждой МРЕ могут хранится признаковые описания всего класса МРЕ, что ведёт к дополнительному расходу ресурсов. Так же, значительно усложняется процесс обучения готовой системы распознавания, построенной с таким использованием такого подхода, так как каждому диктору необходимо произнести каждую МРЕ несколько раз для получения устойчивых эталонов.

С учётом вышеизложенного, можно определить основные требования к МРЕ:

1. Словарь МРЕ должен обладать минимальным возможным размером.

2. Алгоритм сегментации речевого сигнала на МРЕ должен по возможности затрачивать минимальные временно-аппаратные ресурсы.

3. Алгоритм классификации каждой МРЕ также должен минимизировать затраты.

4. МРЕ должны иметь устойчивую классификацию на всем словаре.

Данным требованием удовлетворяют МРЕ, представляющие собой участки речевого сигнала фиксированной длительности, соответствующие фазам фонем или самим фонемам. Количество фонем в русском языке равно 42, из них 6 гласных и 36 согласных. Акустические свойства фонем

определяются артикуляторными особенностями их образования – местом и способом.

Место образования гласных фонем обусловлено положением тела языка и губ. Место образования согласных фонем определяется положением щели в ротовой полости, а также заднее или переднее положение языка. Способ образования фонем характеризует динамические и энергетические характеристики речевого образа.

Сложность использования фонем в качестве МРЕ заключается в том, что в речевом сигнале, соответствующем разговорной речи, фонемы в «чистом» виде не встречаются по причине того, что фонемы способны изменять свои акустико-артикуляторные параметры в зависимости от окружения. Таким образом, в разговорной речи возникают модификации фонем – аллофоны, число которых резко увеличивается, в сравнении с числом фонем, а именно 480 для гласных и 8800 для согласных. Аллофоны можно разделить на позиционные и комбинаторные. Комбинаторные аллофоны возникают в результате влияния фонетического окружения на текущую фонему и наложение процессов артикуляции – эффект коартикуляции. Позиционные аллофоны возникают в результате изменения звучания фонемы в зависимости от положения к ударному слогу или другим фонемам – эффект редукции. Кроме того, фонемы расположенные рядом на плоскости «место-способ» имеют схожие признаковые описания, как следствие, распознающая система имеет низкую точность классификации схожих фонем, в результате чего, возникают ошибки «замены» для фонем одной группы. Традиционно, разделение близкорасположенных фонем не выделяют в отдельный этап идентификации фонем, а разделимость образов повышают использованием более информационных признаковых описаний.

Преимущество использования фонем в качестве МРЕ очевидно – малый размер словаря и простота фонетической модели. Для построения малого словаря в исследовательских целях нет необходимости в использовании большой базы данных для обучения, что так же является

значительным преимуществом, в силу высоких материальных затрат, необходимых для создания большой обучающей базы.

Построение векторов признаков речевых сигналов на основе вейвлет-преобразования. Признаком, называется отображение $f : X \to D_f$, где $D_f$ - пространство возможных значений признака. Вектор $x = (f_1, ..., f_n)$, $x \in X^n$ называется вектором признаков., отождествляемым с самим объектом, и является математическим описанием образа в системах классификации. Пространство $X^n$ называется пространством признаков. В зависимости от пространства возможных значений признаков существует несколько обобщённых типов признаков таких, как бинарные, номинальные, порядковые и количественные. Наиболее часто используются количественные признаки, пространством возможных значений которых является пространство рациональных чисел.

В качестве критерия выбора используемых признаков принят принцип наибольшей информативности признака, для получения более устойчивых алгоритмов классификации.

Традиционно, вектора признаков речевых сигналов получают в результате спектрального анализа исследуемого сигнала с использованием преобразования Фурье На данный момент ведутся исследования по извлечению векторов признаков с использованием вейвлет преобразований, однако значительных результатов в данной области на сегодня не достигнуто. Для построения векторов признаков широко используются знания о психоакустическом восприятии человеком звуковых сигналов.

В рамках данной работы разработан следующий алгоритм извлечения векторов признаков для речевых сигналов на основе вейвлет-анализа Определим набор двумерных фильтров в пространстве «частота-время»:

$H = \{H_{mn} : m = 1...3, \quad n = 1...N\}$, (1) где

$$H_{mn}(t, \omega) = \begin{cases} 1/\tau\omega, & t \in [(n-1)\tau, m\tau], \omega \in [\omega(n-1), \omega n] \\ 0, & иначе \end{cases} \quad (2)$$

в свою очередь $\tau$, $\varpi$ - ширина фильтра во временной и частотной области соответственно, N – параметр, определяемый экспериментально.

Ширина фильтра во временной области может быть найдена из следующего выражения:

$\tau = T/3$, (3) где T – длительность фонемы.

Ширина фильтра в частотной области может быть найдена из выражения

$\varpi = \Omega/N$, (4)

где $\Omega$ - ширина частотной области вейвлет образа.

Тогда вектор признаков может быть сформирован как

$x = (S_1, \ldots S_N, \Delta_1, \ldots \Delta_N)$, (5)

где $S_i = \sum_t \sum_\omega W(t, \omega) H_{2i}(t, \omega)$, (6)

$\Delta_i = \sum_t \sum_\omega W(t, \omega) H_{3i}(t, \omega) + \sum_t \sum_\omega W(t, \omega) H_{1i}(t, \omega)$ (7)

Параметры $\Delta_i$, $i = 1...N$ введены для учета динамических процессов в начале и конце фонемы, обусловленных эффектами редукции и коартикуляции.

Основные подходы к решению задачи распознавания речевых сигналов. Существует большое множество методов решения задачи распознавания речевых сигналов, все они могут быть разделены на два наиболее общих подхода – дискриминантный и структурный. Исторически первым был дискриминатный подход, который в литературе так же называют эталонным или теорико-информационным. Суть данного подхода – формирование пространства признаков речевых образов, в котором схожие речевые образы формируют генеральные совокупности – таксоны или кластеры. Для описания собственных областей таких кластеров используются функции плотности вероятности, которые в своих реализациях приобретают экстремальные значения. Параметры, а также внешний вид

функций плотностей вероятностей определяются в ходе обучения на обучающей выборке. Принадлежность поступившего речевого образа к какому-либо конкретному кластеру в ходе процесса распознавания определятся при помощи решающего правила, которое в большинстве случаев записывается в виде дискриминантной функции.

Данный подход обладает рядом недостатков. Во-первых, в силу ограниченности мощности обучающей выборки приводит к использованию оценок вместо истинно статистических характеристик функций плотности вероятностей для каждого кластера, что влечет за собой нарушение условий оптимальности классификаторов, построенных на статистических решающих критериях, а, следовательно, и к ошибкам распознавания. Во-вторых, данный метод не может напрямую применяться к речевым сигналам в задачах распознавания слитной речи в силу высокой вариативности естественной речи и, как следствие, невозможности составления актуальной обучающей выборки со всеми возможными прецедентами.

Данных недостатков лишен структурный подход. Структурный подход – это метод распознавания речевых образов на основе теории формальных грамматик, когда конечный речевой сигнал представляется в виде иерархического набора структурных единиц.

Точность определения отдельной минимальной акустико-фонетической единицы речи, как правило, не высока и не превышает 80% [9], а значит большой вклад в точность окончательного распознавания вносят принятые фонетические, синтаксические и лексические модели языка. Основным преимуществом структурного подхода является тот факт, что акустико-фонетических единиц на несколько порядков меньше, чем всех возможных словоформ, что значительно уменьшает временные затраты полученных алгоритмов, в сравнении дискриминантным подходом.

Идентификация минимальных речевых единиц. Задача классификации МРЕ представляет собой классическую задачу распознавания образов, которая может быть сформулирована следующим образом. Пусть имеются

$X$ - множество признаковых описаний МРЕ, $Y$ - множество наименований классов МРЕ, $y^*: X \to Y$ - целевая зависимость, значения которой известны для объектов обучающей выборки $X^l = (x_i, y_i)_{i=L}^l$. Требуется построить алгоритм $a: X \to Y$, который будет аппроксимировать целевую зависимость на всем пространстве $X$.

В данной работе в качестве алгоритма классификации был выбран МОВ, что отличается от широко распространенного подхода с использованием скрытых Марковских моделей. Данный подход аргументирован тем фактом, что СММ фактически не является классификатором и не обладает разделяющей способностью. В ходе обучения СММ минимизируются внутриклассовые расстояния, но не максимизируются межклассовые расстояния, в силу чего алгоритм классификации не позволяют распознавать фонемы расположенные рядом на плоскости классификации «место-способ». Предполагается, что МОВ обеспечит более высокую точность классификации близко расположенных фонем в силу максимизации межклассовых отступов в процессе обучения.

**Скрытые Марковские модели** – являются мощным инструментом, позволяющим распознавать речевой сигнал с высоким качеством. К недостаткам СММ среди прочих причисляют и отсутствие средств моделирования длительности звуков. Не секрет, что параметры произносимых звуков обладают значительной вариативностью. Как одно и то же слово, произнесенное одним и тем же диктором, может состоять из разных наборов звуков, так и «одинаковые» звуки могут различаться между собой. Длительность звуков только один из изменяющихся параметров. Значительная вариативность длительности звуков речи делает задачу моделирования и учета в процессе распознавания речи данного параметра актуальной.

Зависимость вероятности появления фонем от их длительности. В результате анализа сегментированных записей речи были построены

гистограммы вероятностей появления фонем различной длительности. Звуки короче 36 мс и длиннее 720 мс считались не валидными, и при построении гистограмм не учитывались. Диапазон допустимых длительностей был разбит на 57 отрезков по 12 мс каждый.

Выяснилось, что зависимости для разных фонем существенно различаются. Были выявлены три основных формы зависимостей вероятностей появления фонем от их длительности, получившие условные названия «ко-са» «нормальное» и «шляпа».

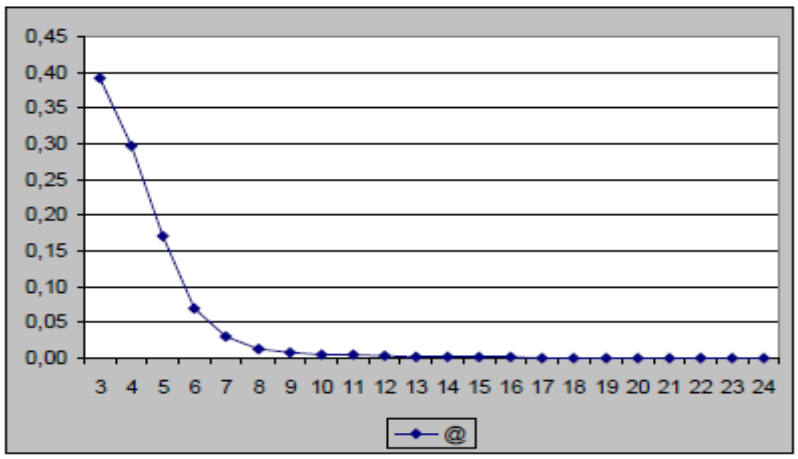



Рис. 1. Зависимость «коса»

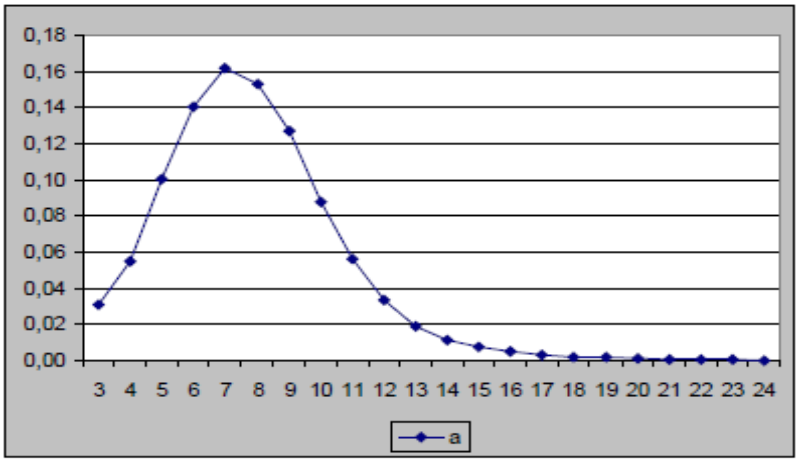



Рис. 2. Зависимость «нормальное»

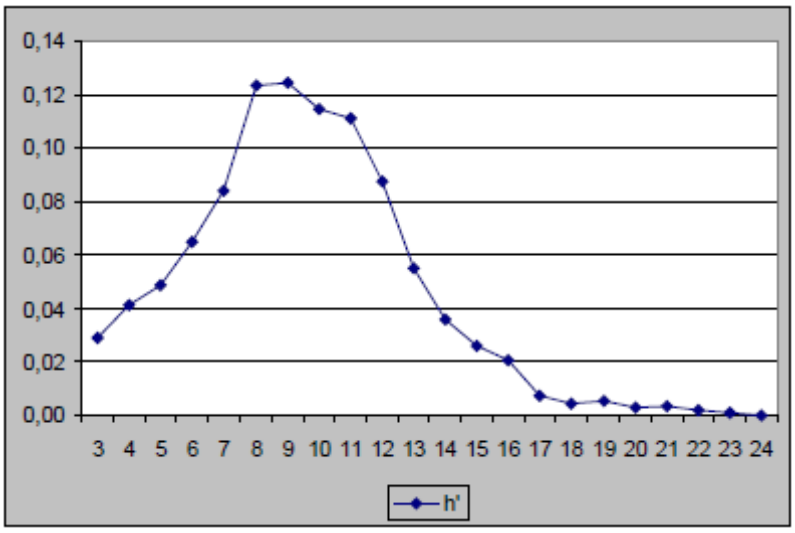



Рис. 3. Зависимость «шляпа»

Наиболее часто встречается распределение вида «нормальное» - им описывается около 2/3 всех зависимостей, еще около 1/3 фонем описываются зависимостью типа «коса», и только для трех фонем характерна зависимость типа «шляпа».

Эксперименты показали, что зависимость вероятности появления фона от его длительности с высокой точностью моделируется модифицированной формулой закона нормального распределения. Видно, что получена из закона нормального распределения путем введения трех параметров:

$$f(x) = \frac{1}{\sqrt{2\pi(D - a_2)}} e^{-\frac{(x-m+a_1)^2}{2(D-a_3)}}$$

Изменение параметра a1 приводит к сдвигу кривой (при увеличении – влево, при уменьшении – вправо); параметр a2 регулирует вертикальное растяжение/сжатие (увеличение параметра растягивает кривую, уменьшение - сжимает); параметр a3 управляет горизонтальным растяжением/сжатием (увеличение a3 сжимает кривую, уменьшение - растягивает).

Для подбора параметров законов распределений фонем была разработана программа, реализующая метод градиентного спуска с переменным шагом. В качестве минимизируемой целевой функции используется квадратичная ошибка:

$$E_i(a_1, a_2, a_3) = \sum_{j=3}^{24} (fe_j - f_j(a_1, a_2, a_3))^2$$

$fe$ – экспериментальное значение вероятности появления фонемы в отрезке j;

$f_j(a_1, a_2, a_3)$ – моделируемое по (1) значение вероятности появления фонемы в отрезке j.

Есть две причины выбора целевой функции вида (2): важнее добиться высокой точности описания для больших значений функции распределения, чем для меньших; в некоторых отрезках j значение fej может равняться нулю из-за того, что количество реализаций данного фона мало.

В таблице 1 представлены параметры законов распределений, использованных в качестве примера форм зависимостей (рис. 1-3).

Таблица 1. Параметры законов распределения

| Фонема | m | D | a1 | a2 | a3 | E |
|---|---|---|---|---|---|---|
| @ | 4,296139 | 4,004516 | 1,873084 | 3,059915 | 0,233991 | 0,000195 |
| a | 7,877618 | 8,097394 | 0,470001 | 1,867001 | 1,985900 | 0,000318 |
| h’ | 9,718665 | 17,485761 | 0,460000 | 6,694901 | 6,969100 | 0,000741 |

Полученный комплект зависимостей и есть модель длительности звуков речи. Дополнительно следует заметить, что при моделировании длительность бралась не в мс, а в интервалах.

В таблице 2 приведены результаты тестирования системы распознавания на пяти грамматиках. В тестировании использовалось более 15 тысяч файлов с записями команд (от 2 до 4 тысяч команд на грамматику).

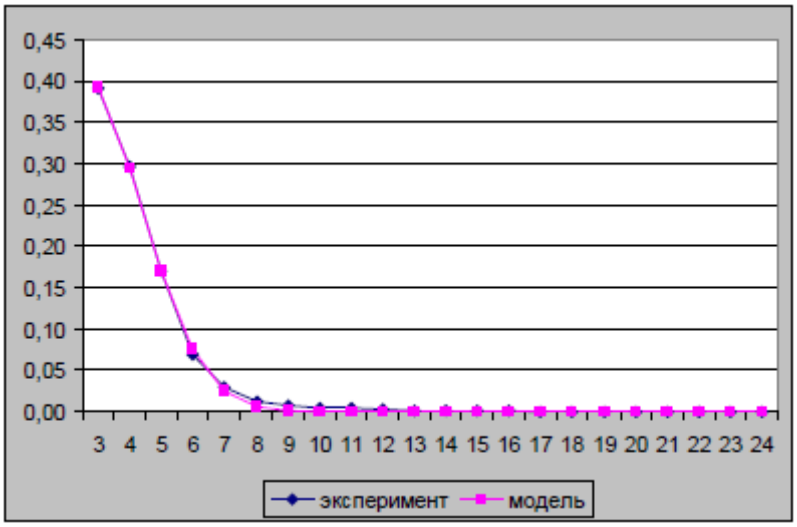



Рис. 4. распределения для фонемы «@»

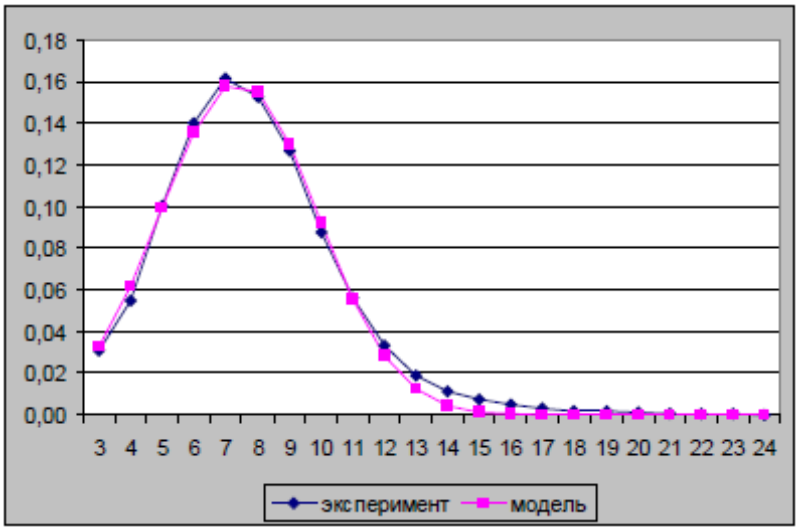



Рис. 5 распределения для фонемы «a»

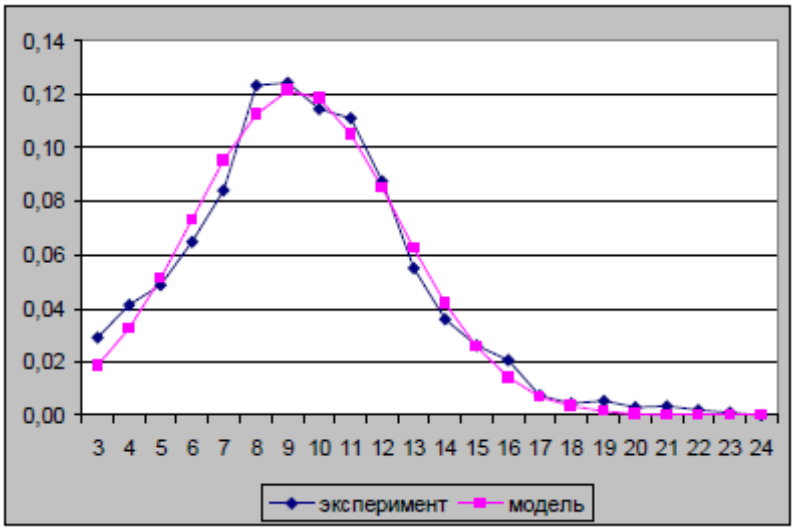



Рис. 6. распределения для фонемы «h’»

Видно, что на четырёх грамматиках происходит увеличение процента правильно распознанных команд при подключении моделей длительностей фонем, и на всех грамматиках среднее время вычислений сокращается на 10-15%. Падение процента на грамматике «пинкоды» объясняется недостаточной точностью моделирования.

**Развитие модели.** Длительность звуков речи зависит от множества факторов: коммуникативная ситуация и условия фонации, стиль речи, наличие акцентов и диалектов, социально-экономические факторы, эмоциональное состояние, анатомические особенности диктора. Кроме того, длительности звуков зависят от контекста, позиции в слове и фразе. Остановимся подробнее на некоторых общих закономерностях, учет которых может оказаться полезным для повышения точности распознавания речи.

В результате исследования зависимости длительности звуков от длины высказывания было установлено, что с увеличением длительности высказывания средняя длительность звуков уменьшается. Также было установлено, что начальные сегменты оказываются короче конечных, причем длительность зависит от количества звуков, оставшихся до конца высказывания.

В также отмечается, что все исследованные факторы, вызывающие изменение длительности как гласных, так и согласных звуков, действуют независимо, и для предсказания длительности звука фразы может использоваться принцип суперпозиции.

Полученные выводы подтверждаются порождающими моделями длительности звуков речи, с успехом используемые в системах синтеза речи по тексту.

Модель, описывающая длительности гласных звуков на основе принципа «несжимаемости», имеет вид (3). Предполагается, что в ударной позиции длительность i-го гласного не может быть меньше величины Timin, а влияние контекста учитывается коэффициентом k.

$$T_i = k(T_i^0 - T_{i\min}) + T_{i\min}$$

где $T_i^0$ – собственная длительность гласного в ударной позиции. )0(iT

Предлагается также модификация модели (3) учитывающая сокращение длительности гласных звуков в зависимости от их числа и положения в высказывании (4):

$$T_i = \alpha^a \beta^b \left(T_i^{(0)} - T_{i\min}\right) + T_{i\min}$$

где α и β - коэффициенты укорочения, a – количество слогов после данно-го звука, b – количество слогов перед данным звуком.

Модели (3) и (4) предназначены исключительно для гласных звуков. В качестве общей модели длительности звуков в высказывании, короче 15 звуков, предлагается следующее уравнение:

$$T_j = \left[T_j^{(0)} - k_n(N - j)\right] \cdot \left[1 - (1 - k_T)\frac{N - j + 1}{N}\right] \cdot \frac{T}{N}, \text{ где}$$

kn – коэффициент сжатия, требуемого по условиям экономии оперативной памяти;

kT – коэффициент темпа речи;

N – количество звуков в высказывании;

T – длительность высказывания.

Исходя из приведенных выше результатов, было решено учесть в модели длительности зависимость от положения звука во фразе. Для различных звуков речи были построены экспериментальные законы распределе-ния вероятности появления звука заданной длительности в заданной пози-ции

высказывания. Примеры полученных законов распределения пред-ставлены на рис. 7 и 8.

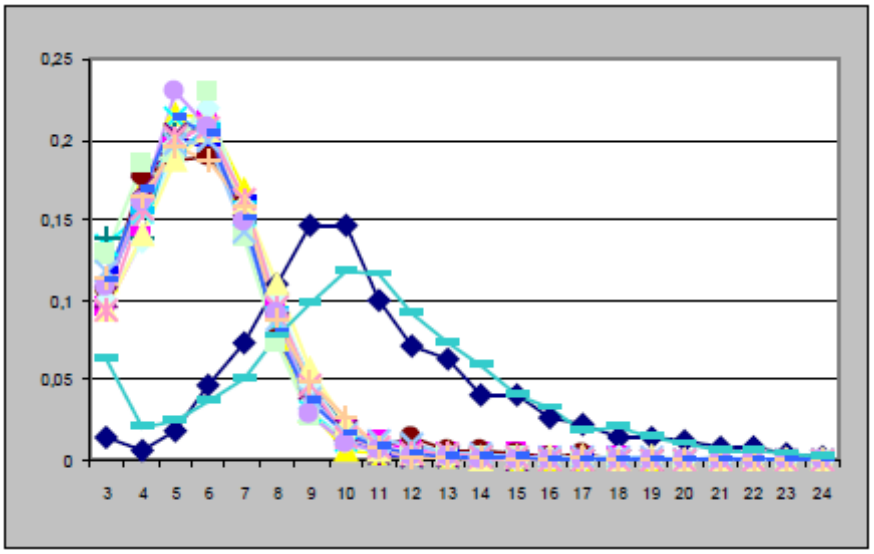



Рис. 7. Для фонемы «a1»

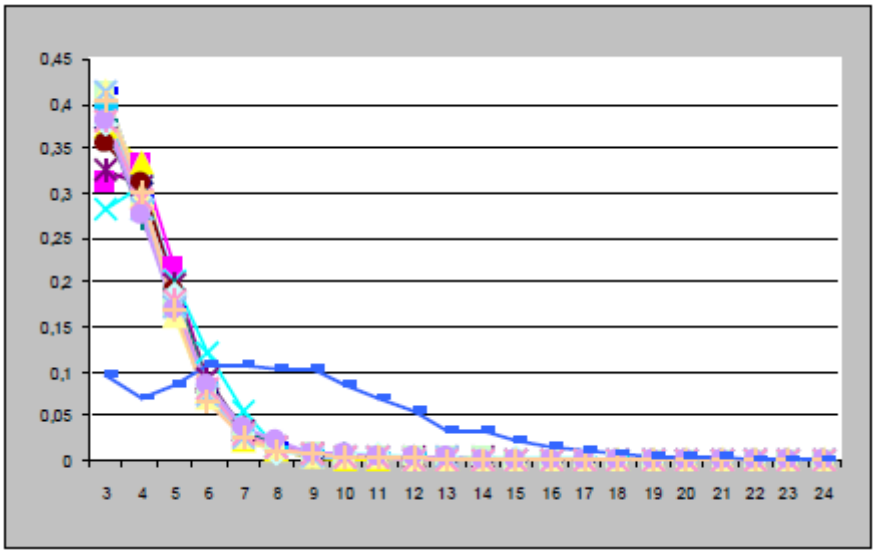



Рис. 8. Для фонемы «@»

Из рисунка видно, что за исключением одной (для «@») или двух (для «a1») кривых, распределения практически совпадают. Выделяющиеся зависимости соответствуют начальным и конечным звукам (на «@» выделяется только одна зависимость, т.к. начальных «@» не существует).

Полученные семейства распределений позволяют сделать вывод о том, что:

• зависимость длительности звука от положения в высказывании не так велико, как принято считать;

• законы распределения длительностей для начальных и конечных фонем высказываний существенно отличаются от законов распределений звуков, расположенных внутри высказываний.

Порождающие модели длительностей звуков речи (3-5) также предполагают, зависимость длительности звуков речи от количества звуков в высказывании. Для проверки этой гипотезы были определены средние значения длительностей звуков в высказываниях, содержащих от 2 до 78 звуков. Полученные зависимости длительности звуков от количества звуков в высказывании представлены рис. 9 – 12.

Как для ударных, так и для безударных гласных, характерно падение средней длительности с увеличением количества звуков в высказывании. Основное падение происходит на высказываниях, содержащих до 16 (30) звуков. При дальнейшем увеличении длины высказываний длительность звуков продолжает медленно убывать. Также видно, что длительности безударных гласных меньше длительностей ударных.

Из рис. 10 видно, что для мягких и твердых сонант характерно сокращение длительности с увеличением количества звуков в высказывании. Основное падение происходит на высказываниях, содержащих от 22 до 30 звуков. При дальнейшем увеличении длины высказываний длительность сонант продолжает медленно убывать. Также видно, что твердость и мягкость практически не влияют на длительности сонант.

Из рис. 11 видно, что для твердых и мягких сонант характерно сокращение длительности с увеличением количества звуков в высказывании. Основное падение происходит на высказываниях, содержащих до 22 звуков. На высказываниях в 23 – 35 звуков происходит некоторое повышение длительности взрывных, после чего продолжает медленно убывать.

Для шипящих и аффрикат не наблюдается столь гладкой картины, как для других типов звуков. (Возможно, причина кроется в их меньшей

частотности.) Однако графики (рис. 12) позволяют проследить общую тенденцию уменьшения длительности звуков с увеличением их количества в высказывании.

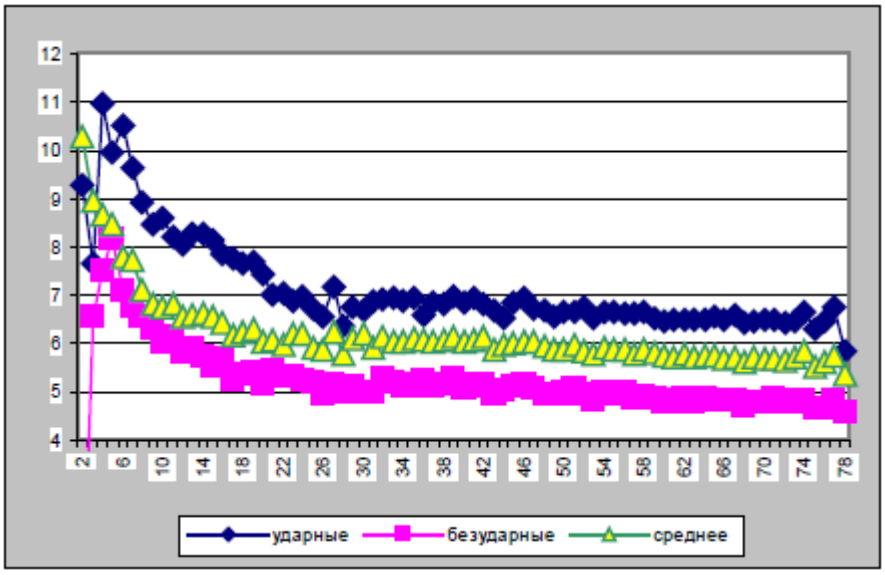



Рис. 9. Ударные и безударные гласные

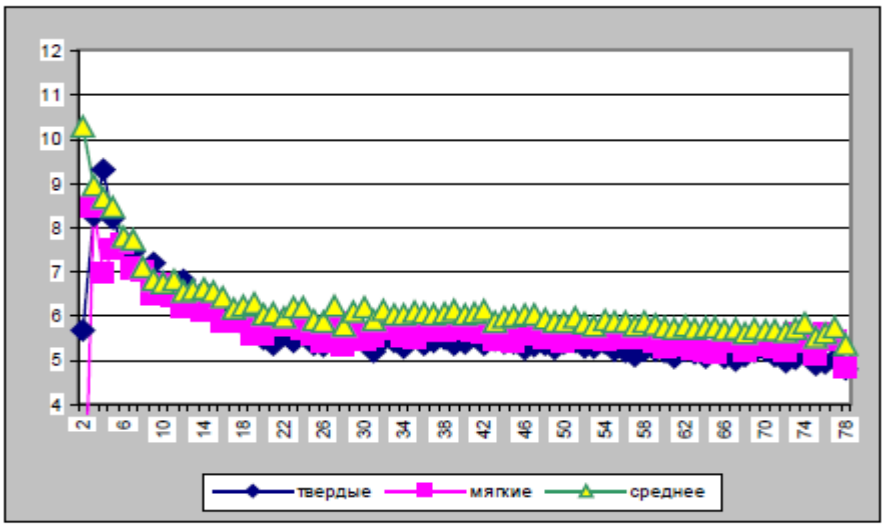



Рис. 10. Твердые и мягкие сонанты

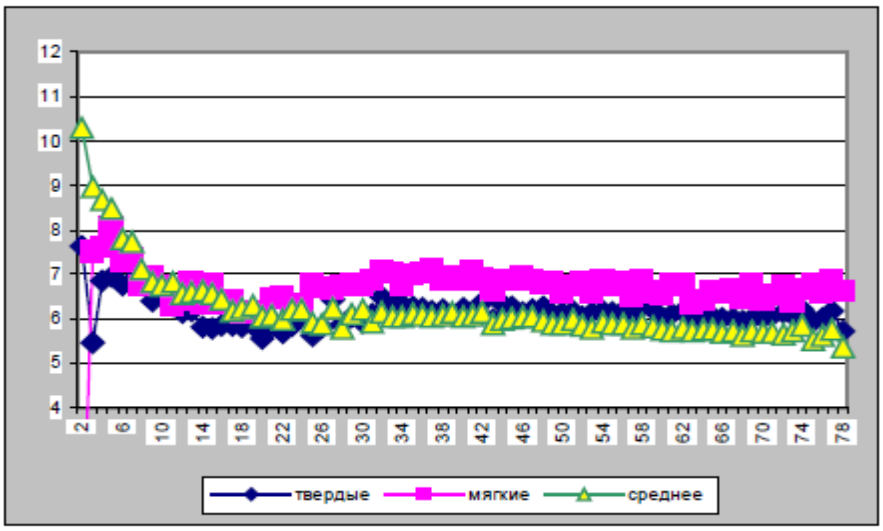



Рис. 11. Твердые и мягкие взрывные

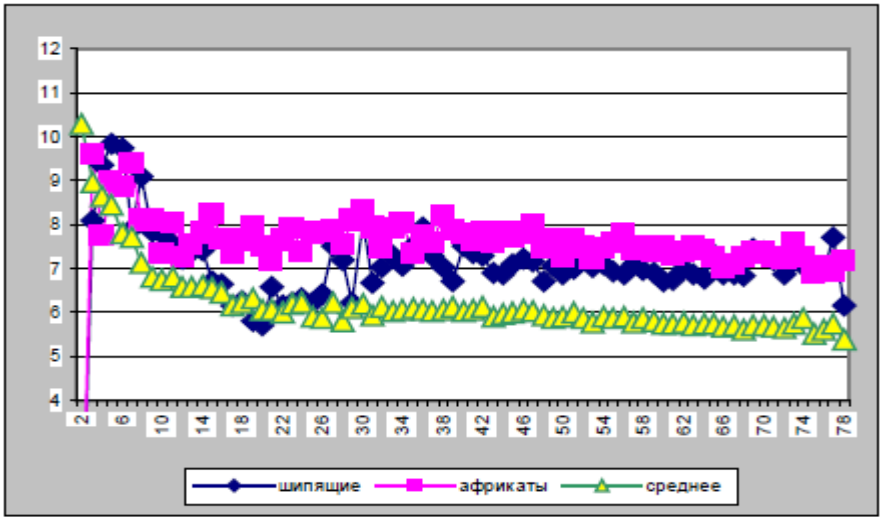



Рис. 12. Шипящие и аффрикаты

Зависимости, представленные на рисунках 9 – 12, подтверждают:

• наличие связи между количеством звуков во фразе и их длительностью;

• существование общей тенденции убывания длительности звуков с уве-личением их числа в высказывании;

• уникальность вида зависимости для звуков разных типов.

Сказанное выше свидетельствует о целесообразности учета длитель-ности высказываний (выраженной в количестве звуков речи) в модели длительностей звуков. В качестве примера приведем экспериментальные зависимости вероятностей появления фонемы «а1» заданной длительности в словах с различным количеством звуков (рис. 13).

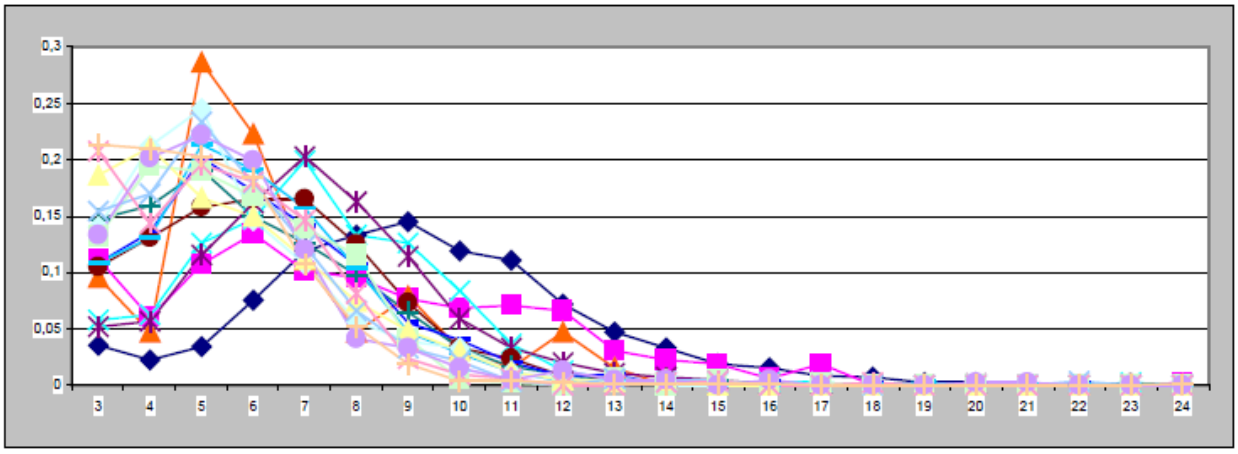



Рис. 13. Экспериментальные зависимости вероятностей появления фонемы «a1»

В с помощью коэффициента k, учитывается влияние контекста на длительность звука. В приводятся правила для коррекции длительности звуков в зависимости от контекста, используемые в системах синтеза речи по тексту. Приведем некоторые из них:

• следующий за ударным звуком согласный звук укорачивается на 25 мс; звук, следующий за укорачиваемым согласным, также укорачивается на 25 мс;

• гласный, предшествующий звонкому согласному, удлиняется на 20 мс;

Предшествующий щелевому согласному гласный удлиняется на 15 мс; и т.д.

С учетом того, что модель описывает зависимость длительности от количества звуков в высказывании, и общего количества всех возможных контекстов, построение распределений для всех возможных контекстов не представляется возможным. В большинстве случаев для построения распределений просто не хватает данных.

## Выводы по первой главе

В проделанной работе были получены следующие результате:

- Общие понятия интеллектуального анализа данных;
- Исследованы алгоритмы интеллектуального анализа данных;

- Изучен применение методов ИАД при параметризации и анализа речевых информации;

# II Глава. Экспериментальное исследование характеристик разработанных технологии и алгоритмов распознавание речи

## 1. Перспективы развития систем распознавания речи

Системы распознавания голоса – это вычислительные системы, которые могут определять речь говорящего из общего потока. Эта технология связана с технологией распознавания речи, которая преобразует произнесенные слова в цифровые текстовые сигналы, путем проведения процесса распознавания речи машинами. Обе эти технологии используются параллельно: с одной стороны для идентификации голоса конкретного пользователя с другой стороны для идентификация голосовых команд посредством распознавания речи. Распознавание голоса используется в биометрических целях безопасности, чтобы определить голос конкретного человека. Эта технология стала очень популярной в мобильном банкинге, который требует идентификации подлинности пользователей, а также для других голосовых команд, чтобы помочь им совершать сделки.

Некоторые из основных факторов мирового рынка распознавания речи:

• Увеличение спроса на услуги голосовой биометрии

• Более широкое использование идентификации диктора для судебно-медицинских целей

• Спрос на распознавания речи в военных целях

• Высокий спрос для распознавания голоса в сфере здравоохранения

Увеличение спроса на услуги голосовой биометрии

Изначально, слово «биометрия» встречалось только в медицинской теории. Тем не менее, стали возрастать потребности в безопасности с использованием биометрических технологий среди предприятий и государственных учреждений. Использование биометрических технологий – один из ключевых факторов на мировом рынке распознавания речи. Распознавание голоса используется проверки подлинности человека, так как

голос каждого человека индивидуален. Это обеспечит высокий уровень точности и безопасности. Распознавание голоса имеет большое значение в финансовых институтах, таких как банк, а так же на предприятиях в сфере здравоохранения. В настоящее время сегмент распознавания речи составляет 3,5% от доли технологий биометрии на мировом рынке, но это доля имеет постоянный рост. Также низкая стоимость биометрических устройств увеличивает спрос со стороны малого и среднего бизнеса.

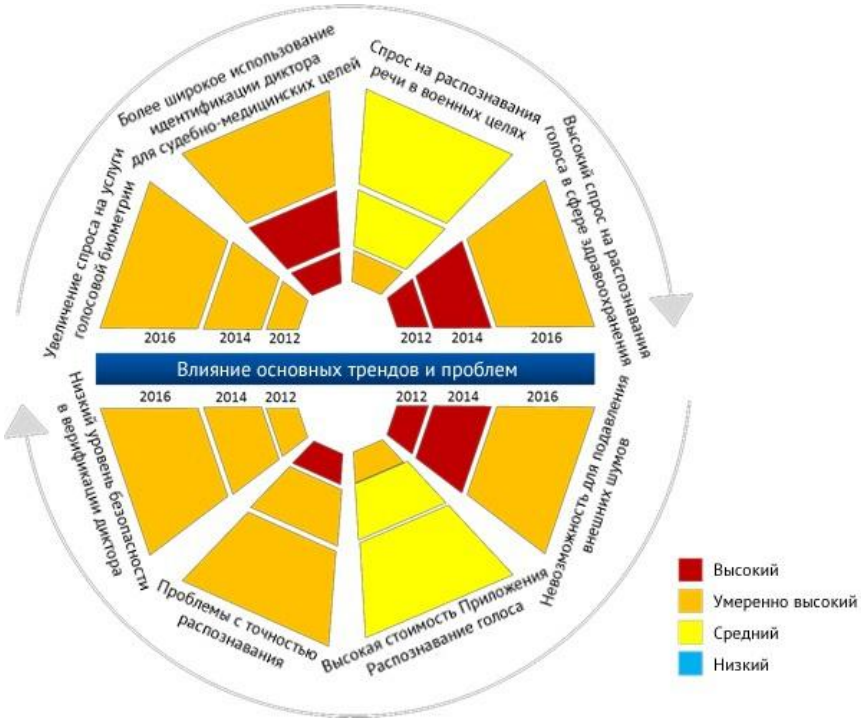
Более широкое использование идентификации диктора для судебно-медицинских целей

Использование технологии идентификации диктора для судебно-медицинских целей является одной из главных движущих сил на мировом рынке распознавания голоса. Происходит сложный процесс определения, соответствует ли голос лица, подозреваемого в совершении преступления, голосу из судебно-медицинских образцов. Данная технология позволяет правоохранительным органам выявлять преступников по одной из самых уникальных характеристик человека, его голосу, тем самым предлагая относительно высокий уровень точности. Судебно-медицинские эксперты проводят анализ соответствия голоса подозреваемого образцам до тех пор, пока не будет найден преступник. В последнее время эта технология используется, чтобы помочь решать некоторые уголовные дела.

**Высокий спрос для распознавания голоса в сфере здравоохранения.** Биометрические технологии, такие как сосудистое распознавание, распознавание голоса и сканирование сетчатки глаза широко внедряются в сферу здравоохранения. Распознавание голоса, как ожидается, станет одним из основных режимов идентификации в медицинских учреждениях. Многие компании здравоохранения в США, обращаясь к стандартам Health Insurance Portability and Accountability Act (HIPAA), также применяют биометрические технологии, такие как распознавание голоса, распознавание отпечатков пальцев для более безопасной и эффективной регистрации пациента, накопления информации пациента, защиты

медицинских записей пациента. Также учреждения клинических испытаний внедряют распознавания голоса для выявления лиц, набранных для клинических испытаний. Таким образом, голосовая биометрия является одним из основных режимов для идентификации клиента в сфере здравоохранения в Азиатско-Тихоокеанском регионе.

**Требования рынка**




Влияние основных четырёх трендов и проблем на мировой рынок распознавания показано на рисунке

Влияние проблем и трендов оценивается на основе интенсивности и длительности их воздействия на текущий рынок. Классификация величины воздействия:

• Низкий – незначительное или нулевое влияние на рынок

• Средний – средний уровень влияния на рынок

• Умеренно высокий– значительное влияние на рынок

• Высокий – очень сильное воздействие с радикальным влиянием на рост рынка

Несмотря на рост трендов мировой рынок распознавания голоса продолжает сталкиваться с некоторыми серьёзными тормозами роста. Одна

из важных проблем – трудность подавления окружающего шума. Хотя рынок распознавания речи стал свидетелем нескольких технологических достижений, неспособность подавлять окружающий шум все еще остается препятствием на пути к признанию приложений распознавания голоса. Еще одной проблемой для этого рынка является высокая стоимость приложений распознавания голоса.

Некоторые из основных задач, стоящих перед мировым рынком распознавания голоса:

• Невозможность подавления внешних шумов

• Высокая стоимость приложения распознавание голоса

• Проблемы с точностью распознавания

• Низкий уровень безопасности в верификации диктора

**Невозможность подавления внешних шумов.** Несмотря на технический прогресс в сфере распознавания голоса, шумы продолжает оставаться одной из основных проблем на мировом рынке распознавания голоса. Кроме того, голосовая биометрия отличается особенной чувствительностью по сравнению с другими видами биометрии. Приложения распознавания голоса, голосовой биометрии и распознавания речи оказываются очень чувствительными к шуму окружающей среды. В результате, любое шумовое нарушение препятствует точности распознавания. Также нарушается автоматизированный ответ на голосовую команду. Неспособность подавить окружающий шум является единственным фактором, который не дает системам распознавания голоса достичь высоких результатов и занять высокий процент доли на мировом рынке биометрических технологий.

**Высокая стоимость приложений распознавания голоса.** Одной из основных проблем, препятствующих развитию технологий распознавания речи, является потребность в больших инвестиционных вложениях, требуемых для разработки и реализации. Крупномасштабное развертывание технологии распознавания голоса на предприятии является трудоемким

процессом и требует огромных инвестиций. Экономия на бюджете приводит к ограничению тестирования технологии, следовательно, любой сбой может привести к большим потерям на предприятии. Поэтому альтернативные распознаванию голоса варианты, такие как swipe card и keypad по-прежнему активно используются во многих компаниях, особенно среди малого и среднего бизнеса, в силу их экономической эффективности. Таким образом, приложения распознавания голоса требуют больших материальных вложений, включая стоимость интеграционной системы, дополнительного оборудования и другие затраты.

**Проблемы с точностью распознавания.** На мировом рынке распознавания голоса единой проблемой является невысокие показатели точности распознавания, не смотря на то, что в настоящее время системы распознавания голоса способны распознавать различные языки и определять подлинность голоса. Так как система включает в себя сложный процесс согласования баз данных с произносимыми командами и интегрированной технологией распознавания речи и голосовой верификации, даже незначительная ошибка в любой часть процесса может привести к неверному результату. Погрешность в распознавании речи является одним из основных ограничений в приложениях распознавания голоса. Однако некоторые производители начали разработку систем с очень низким уровнем погрешности в распознавании голоса. Они разработали системы с менее чем 4% неточных результатов (например, измерения голосовой биометрии неверно идентифицируют и отвергают голос человека, у которого есть доступ).

**Увеличение слияний и поглощений.** На мировом рынке распознавания голоса наблюдаются серьезные тенденции слияния и поглощения. Доминирующий лидер рынка Nuance Communications Inc., который держит более чем 50% доли на рынке, приобрел большое количество маленьких компаний на рынке распознавания речи. Из этого следует, что приобретение – это новый подход к росту компании, в

результате чего у Nuance шесть приобретений в 2007 году. Эта тенденция, как ожидается, сохранится и в ближайшие несколько лет в связи с наличием многочисленных мелких игроков, которые могут быть приобретены более крупными компаниями как Nuance. Поскольку рынок является технологически ориентированным, то небольшие компании разрабатывают инновационные решения. Но из-за нехватки ресурсов эти компании не в состоянии увеличить масштабы своего бизнеса. Таким образом, крупные компании, такие как Nuance, используют процесс поглощения в качестве основной стратегии для выхода на новые рынки и отрасли. Например, Nuance приобрела Loquendo Inc. Для того, чтобы войти в регион EMEA.

## 2. Поиск оптимальной системы анализ речи с открытыми и закрытыми исходными кодами

Согласно лингвистическим особенностям человеческой речи, дополнительные артикуляционные данные позволяют более точно выявить речь диктора и автоматически разбить звуковую волну на отдельные фрагменты. Также, при общем анализе аудиовизуального голосового сигнала во временной динамике имеется перспектива фиксирования открытых и закрытых слогов, звонких, шипящих, ударных, безударных гласных/согласных и другие речевые единицы.

Среди самых распространённых систем распознавания речи можно выделить 2 типа лицензий. К лицензии BSD относятся следующие продукты аудио-распознавания речи: CMU Sphinx, Julius. К лицензии GPL относятся: Simon software, iATROS, RWTH ASR (как разновидность Q Public License (QPL) лицензии), SHoUt, VoxForge (как разновидность — Open source acoustic models and speech corpus, то есть речевой модели как корпуса). Рассмотрим их более подробно:

Рис. 1. Эмблема CMU Sphinx

**CMU Sphinx** – также для краткости просто называемая Sphinx, главным образом была написана группой разработчиков систем распознавания речи университета Карнеги Меллон. Она включает в себя серию распознавателей речи (Sphinx 2-4) и тренировщика акустической модели (Sphinx train).

**Sphinx – это дикторонезависимый** распознаватель непрерывной речи, который использует Скрытую Марковскую модель и n-граммную статистическую языковую модель. Она была разработана Кей-Фу Ли. Sphinx имеет возможности распознавания продолжительной речи, дикторонезависимый огромный словарь распознавания, то есть те возможности, которые в 1986 году вызвали большие разногласия в среде распознавания речи. Sphinx в историческом развитии примечателен тем, что в своем развитии затмил все предыдущие версии по своей производительности.

Текущие цели развития включают в себя:

• Развитие новых акустических моделей для тренировки;

• Реализация системы речевой адаптации (MLLR)

• Улучшения менеджмента конфигурации

• Реализация ConfDesigner – графической системы дизайна

**PocketSphinx –** эта версия Sphinx может быть встроена в любые другие системы на базе ARM процессора. PocketSphinx активно развивается и встраивается в различные системы с арифметикой фиксированной запятой и в эффективные модели на базе смешанной модели вычислений.


Рис. 2 Эмблема Julius

**Julius** – это высокопроизводительный распознаватель непрерывной речи с большим словарем (large vocabulary continuous speech recognition),

декодер программного обеспечения для исследования в области связанной речи и разработки. Он идеально подходит для декодирования в режиме почти реального времени на большинстве существующих компьютеров, с словарем 60 тысяч слов, используя задачи тиаграмма слова и контекстно независимую Скрытую марковскую модель. Главная особенность проекта заключается в полной встраиваемости. Это также безопасная модуляция может быть независима от модельных структур и различных типов Скрытых Марковских моделей, которая поддерживает общее состояние трифонов и связанной смеси-моделей со множеством микстур, фонем и утверждений. Стандартные форматы активны за счет свободного моделирования инструментария. Главная платформа системы Linux и другие UNIX подобные станции, также система работает на Windows. Julius имеет открытый исходный код и распространяется с BSD типом лицензии.


Рис. 3 Эмблема RWTH ASR

**RWTH ASR (кратко RASR)** – это инструментарий распознавания речи с открытым исходным кодом. Инструментарий включает в себя технологию умения распознавать речь для создания автоматических систем распознавания речи. Данная технология развивается Технологическим центром Естественного языка и Образцовой распознавательной группой в Рейнско-Вестфальском техническом университете Ахена.

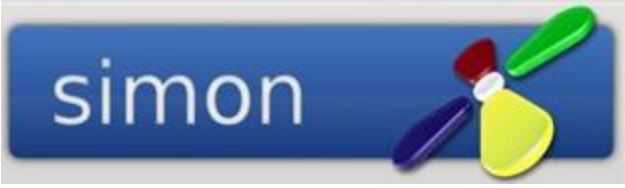



Рис. 4. Эмблема Simon

**Simon** – система распознавания речи, основанная на речевых движках Julius и HTK. Система Simon спроектирована таким образом, что она довольна удобна для работы с различными языками и разного рода

диалектами. При этом реакция распознавания речи полностью настраиваемая и она не подходит для исключительного распознавания единичных голосовых запросов и не может быть сконфигурирована под нужды пользователей.

Чтобы легко использовать систему необходимо выполнить определенные «сценарии». Пакеты Simon сконфигурированы для специальных задач. Среди возможных сценариев Simon, например «Firefox» (запуск и управление браузером «Firefox») или «пакет управления окном» (закрытие, движение, изменение размеров окон) и так далее. Сценарии легко могут быть создаваться пользователями и раздаваться в сообществе через Get Hot New Stuff систему. На сегодняшнее время написано более 39 сценариев и опубликовано 3 языка на репозитарии opendesktop.org


Рис. 5. Эмблема iATROS

**iATROS** – это новое исполнение системы распознавания речи предыдущего поколения ATROS, которая подходит для распознавания как речи, так и для рукописного варианта текста. iATROS основан на модулярной структуре и может использоваться как для построения дифференцированных моделей, чья цель осуществить Ветибри поиск на основе скрытой Марковской модели. iATROS обеспечивает стандартный инструментарий для распознавания речи как в режиме офлайн, так и онлайн (основываясь на ALSA модулях).

iATROS состоит из 2-х модулей предварительной обработки (для речевого сигнала и изображений написанных от руки) и модуля ядра распознавания. Предварительная обработка данных и черты извлечения модулей обеспечиваются векторами распознавания модулей, которые используют Скрытые Марковские модели и языковые модели, которые исполняются поиском предположений из лучших систем распознавания речи.

Все эти модули выполнены на языке программирования “C”.


Рис. 6. Эмблема SHoUt

SHoUt – это инструментарий, написанный Marjin Huijbregts в Нидерландском институте звуков и видео, который подходит для распознавания продолжительной речи с большим словарем данных. Инструментарий состоит из приложения для тренировки статистических моделей и для (не)/речевого детекционирования, диаризации и декодирования речи.


Рис. 7. Эмблема VoxForge

VoxForge – это свободный речевой корпус и языковая акустическая модель, представленная на Open Source хранилище данных. VoxForge был собран как хранилище транскрипций речи, имеющий свободный GPL корпус для использования с речевыми движками, которые имеют открытый исходный код. Речевые аудиофайлы могут быть собраны в акустические модели для использования с системами распознавания речи с открытым исходным кодом типа Julius, Sphinx и HTK (но HTK имеет ограничение по дистрибьюции).


Рис. 8. Эмблема HTK

HTK – это инструментарий для распознавания речи, использующий скрытую марковскую модель. Его главным образом предназначают для распознавания речи, однако он также используется для других иных

приложений распознавания, в которых используется скрытая Марковская модель (включая речевой синтез, распознавание символов и ДНК упорядочивание).

**Что касается определения — закрытый исходный код,** то необходимо сказать – оно означает, что распространяются только бинарные (откомпилированные) версии программы и лицензия подразумевает отсутствие доступа к исходному коду программы, что затрудняет создание модификаций программы. Доступ к исходному коду третьим лицам обычно предоставляется при подписании соглашения о неразглашении.

ПО с закрытым исходным кодом является проприетарным (собственническим) ПО. Однако необходимо иметь в виду, что фраза «закрытый исходный код» можно трактовать по-разному. Так как она может подразумевать лицензии, в которых исходный код программ недоступен. Однако если считать ее антонимом открытого кода, то она относится к программному обеспечению, не подходящему под определение лицензии открытого ПО, что имеет несколько другой смысл. Одним из таковых спорных моментов стало то, как трактовать понятия интерфейса программирования приложений.

**Dragon Mobile SDK.** Сам инструментарий называется NDEV. Чтоб получить необходимый код и документацию, надо зарегистрироваться на сайте в «программе сотрудничества».

Инструментарий (SDK) содержит в себе компоненты и клиента, и сервера. Диаграмма иллюстрирует их взаимодействие на верхнем уровне:

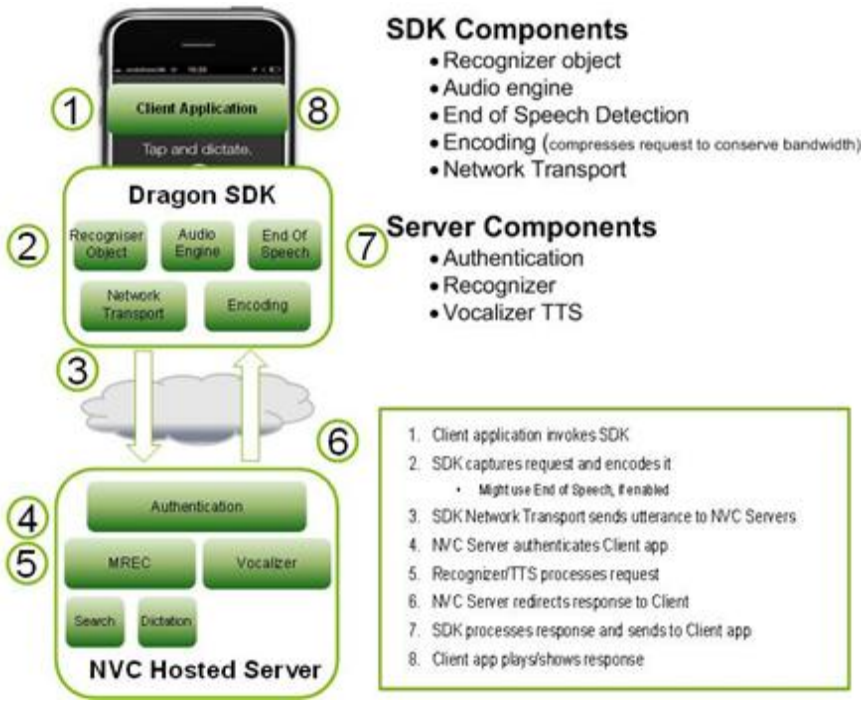



Рис. 1. Принцип работы технологии Dragon Mobile SDK

Комплект Dragon Mobile SDK состоит из различных примеров кода и шаблонов проектов, документации, а также программной платформы (фреймворка), упрощающей интеграцию речевых сервисов в любое приложение.

Платформа Speech Kit framework позволяет легко и быстро добавлять в приложения сервисы распознавания и синтеза (TTS, Text-to-Speech) речи. Данная платформа также обеспечивает доступ к компонентам обработки речи, находящимся на сервере, через асинхронные «чистые» сетевые API, сводя к минимуму накладные расходы и потребляемые ресурсы.

Платформа Speech Kit является полнофункциональным высокоуровневым «фреймворком», который автоматически управляет всеми низкоуровневыми сервисами.

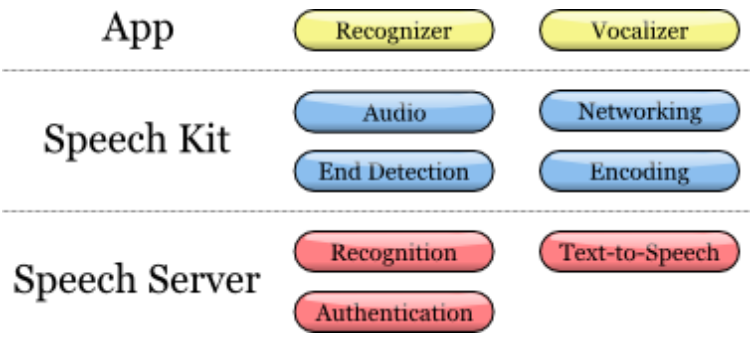


Рис. 2. Архитектура Speech Kit

Платформа выполняет несколько согласованных процессов:

1. Осуществляет полное управление аудио системой для записи и воспроизведения

2. Сетевой компонент управляет подключениями к серверу и автоматически восстанавливает соединения с истекшим временем ожидания при каждом новом запросе

3. Детектор окончания речи определяет, когда пользователь закончил говорить, и при необходимости автоматически останавливает запись

4. Кодирующий компонент сжимает и распаковывает потоковую аудиозапись, снижая требования к полосе пропускания и уменьшая среднее время задержки.

Система серверов отвечает за большинство операций, входящих в цикл обработки речи. Процесс распознавания или синтеза речи выполняется целиком на сервере, обрабатывая или синтезируя аудио-поток. Кроме того, сервер осуществляет аутентификацию в соответствии с конфигурацией разработчика.

Платформа Speech Kit является сетевым сервисом и нуждается в некоторых базовых настройках перед началом использования классов распознавания или синтеза речи.

Данная установка выполняет две основные операции:

Во-первых, она определяет и авторизует ваше приложение.

Во-вторых, — устанавливает соединение с речевым сервером, — это позволяет производить быстрые запросы на речевую обработку и, следовательно, повышает качество обслуживания пользователей.

Распознавание речи

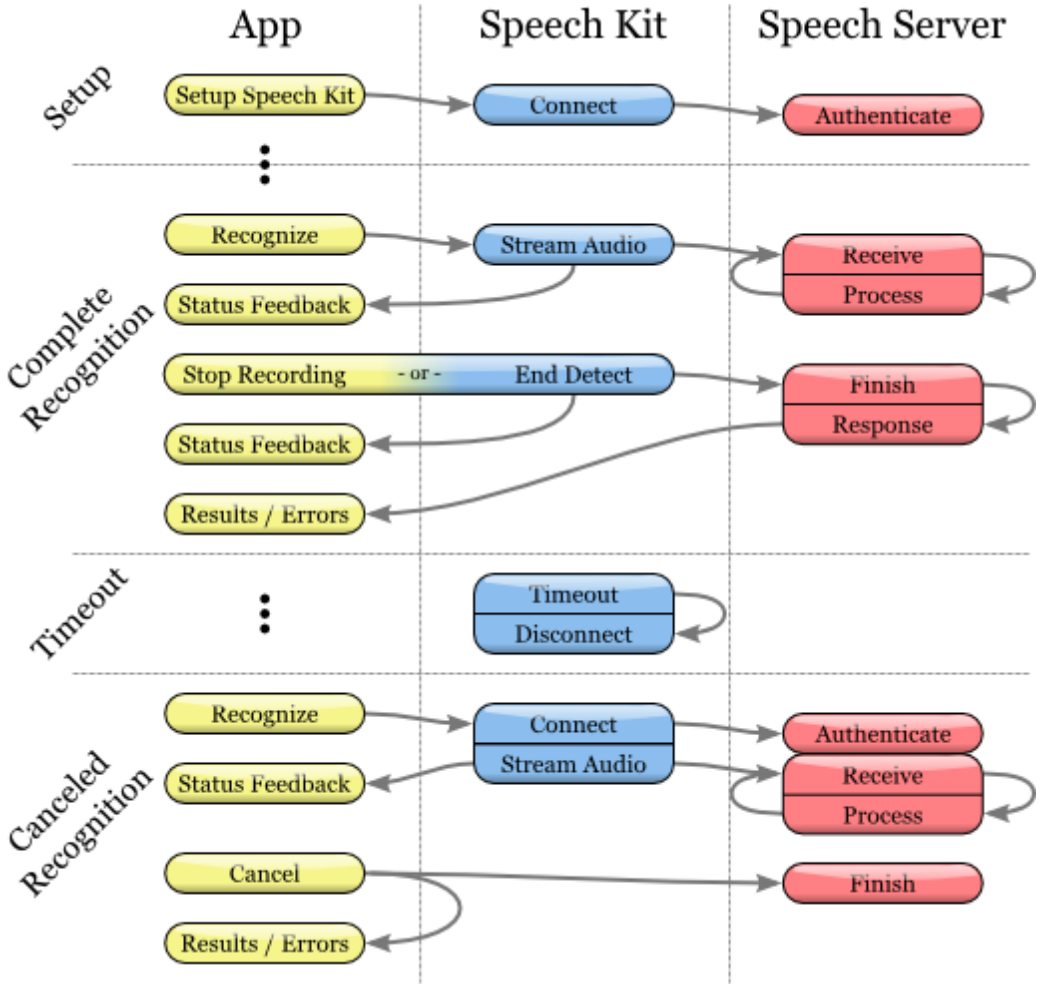



Рис. 3. Процесс распознавания речи

Пионер отрасли распознавания речи показал отличные результаты, особенно на английском языке. Однако большим его недостатком следует считать ограниченный бесплатный функционал: только 10 тысяч запросов в сутки — которых для работы нашего приложения очень скоро стало недостаточно. За больший доступ следует платить.

Google Speech Recognition API

Google™ Voice Search

Рис. 4. Логотип Google Voice Search

Это продукт компании Google, который позволяет вводить голосовой поиск с помощью технологии распознавания речи. Технология интегрирована в мобильные телефоны и компьютеры, где можно ввести информация с помощью голоса. С 14 июня 2011 года Google объявила об интеграции речевого движка в Google Search и с тех пор он работает в стабильном режиме с этого времени. Эта технология на персональных компьютерах поддерживается только браузером Google Chrome. Функция включена по умолчанию в сборках dev-канала, но может быть включена вручную добавлением командного флага. Есть также функция голосового управления для введения речевых команд на телефонах с ОС Android.

Первоначально Google Voice Search — поддерживал короткие поисковые запросы длиной 35-40 слов. Необходимо было для отправки запроса включать и выключать микрофон, что было очень не естественно для использования (такая функция еще осталось в строке поиска Google, нужно нажать лишь на микрофон). Однако, в конце февраля 2013 года в браузер Chrome была добавлена возможность распознавания непрерывной речи и фактически Google Voice Search трансформировался в Speech Input (можно попробовать технологию на примере набора текста в Google Translate). Технологию можно экспериментально протестировать например также здесь. Ознакомиться с полной документацией можно здесь. Заметим лишь, что если раньше многие разработчики грешили тем, что незаконно с помощью различных уловок вклинивались в канал распознавания Google Speech API, то сейчас во время частых изменений API с мая 2014 года процесс доступа к API фактически стал легализован, так как для работы с базой данных системы распознавания речи достаточно зарегистрировать учетную запись в Google Developers и потом можно работать с системой в рамках правового поля.

Yandex Speech Kit


Рис. 5. Лого Yandex Speech Kit

Сразу замечу, что сам лично я с данной библиотекой не работал. Расскажу лишь об опыте программиста, который работал с нами. Он говорил, что очень тяжелая для его восприятия документация и система имеет ограничение по количеству запросов: 10 000 в сутки, поэтому в итоге мы не стали использовать базу данных от Яндекса. Хотя по уверению разработчиков — этот инструментарий является номером 1 для русского языка и, что исследовательская группа компании, которая работала одна в Швейцарии, другая в Москве смогла сделать технологический прорыв в этой области. Однако с таким решением достаточно тяжело выходить на международный рынок по словам Григория Бакунова, так как «многое в области распознавания речи с точки зрения патентования принадлежит известной Nuance и Яндекс одним из последних сумел зацепиться за вагон уходящего вперед поезда развития систем распознавания речи.»

Microsoft Speech API


Рис. 6. Лого Microsoft Speech API

Майкрософт тоже в последнее время стал активно развивать речевые технологии. Особенно после анонсирования голосового ассистента Cortana и разработки автоматический технологии синхронного телеперевода с английского на немецкий язык и наоборот для Skype. На текущий момент существуют 4 варианта использования:

1. Для Windows и Windows Server 2008. Можно добавить речевой движок для Windows приложения используя управляемый или нативный код, который можно взять с API и управлять речевым движком, который встроен в Windows и Windows Server 2008.

2. Speech Platforms. Встраивание платформы в приложения, которые используют распространяемые Microsoft диструбутивы (языковые пакеты с функцией распознавания речи или же средства перевода текста в речь).

3. Embedded. Встроенные решения, которые позволяют человеку взаимодействовать с устройствами используя голосовые команды. Например управление автомобилями Форд с помощью голосовых команд в ОС WIndows Automotive

## 3. Распознавание речи на основе искусственных нейронных сетей

Создание естественных для человека средств общения с компьютером является в настоящее время важнейшей задачей современной науки, при этом речевой ввод информации осуществляется наиболее удобным для пользователя способом. Разработка технологии распознавания речи учёные начали с освоения методики выделения информативных признаков, описывающих речевой сигнал. Затем приступили к решению задачи классификации речевых сигналов наборами информативных признаков.

Существуют следующие подходы к выделению информативных признаков, описывающих речевой сигнал:

- метод линейного предсказания;
- спектральный анализ.

Спектральный анализ отличается от линейного предсказания тем, что оценки среднего значения усреднённого шума вычитаются из спектра, вычисленного по зашумлённым данным.

Наиболее часто употребляются два подхода к классификации и распознаванию:

- мера близости параметров (такая функция называется метрикой);
- нейронные сети.

Второй подход не использует вспомогательных функций, но моделирует процесс распознавания в биологических системах. Этот подход представляется более перспективным в настоящее время.

В системах распознавания речи выделяются две основные подсистемы:

- подсистема предварительной обработки речевых сигналов;
- подсистема классификации речевых сигналов.

На рис. 1 показана схема предварительной обработки речевых сигналов. В настоящей работе представлены модель распознавания речи на основе искусственных нейронных сетей.

Похожая статья: Intelligent Evolutionary Studio — программное обеспечение для машинного обучения искусственных нейронных сетей

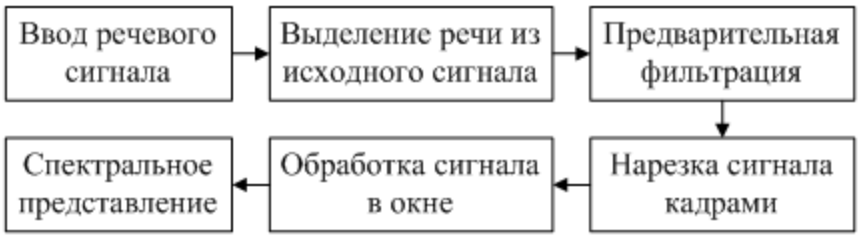



Рис. 1 - Схема предварительной обработки речевых сигналов Модель распознавания речи на основе искусственных нейронных сетей

Пусть речевой сигнал как входные данные нейронной сети. После обработки звуковых данных получен массив сегментов сигналов. Каждый сегмент соответствует набору чисел, характеризующих амплитудные спектры сигнала. Для подготовки к вычислению для сигнала выхода нейронной сети необходимо записать все наборы чисел в таблицу, строка которой – это набор чисел каждого кадра.

Таблица 1 – Описание набора признаков речевого сигнала

| Кадр | 1-ое значение | 2-ое значение | … | I-ое значение |
|---|---|---|---|---|
| 1-ый кадр | $x_{11}$ | $x_{12}$ | … | $x_{1I}$ |
| 2-ый кадр | $x_{21}$ | $x_{22}$ | … | $x_{2I}$ |
| … | … | … | … | … |

| | | | | |
|---|---|---|---|---|
| N-ый кадр | $x_{N1}$ | $x_{N2}$ | … | $x_{NI}$ |

I – Количество значений одного набора чисел

N – Количество наборов чисел (кадр сигнала после нарезки)

Количество входных и выходных нейронов известно. Каждый из входных нейронов соответствует одному набору чисел. А на выходном слое только один нейрон, выход которого соответствует желаемому значению распознавания сигнала.

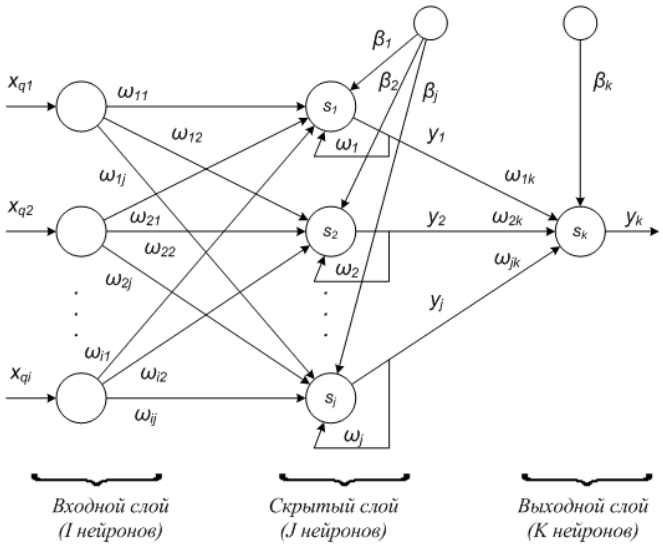



Рис. 1 – Структура нейронной сети с одной обратной связью

$y_j$ – выход j–го нейрона слоя;

$\omega_{ij}$ – весовой коэффициент связи, соединяющей i–ый нейрон с j–ым нейроном;

$\omega_j$ – весовой коэффициент обратной связи j–го нейрона;

$\beta_j$ – смещение j–го нейрона слоя.

Для вычисления выхода нейронной сети необходимо выполнить следующие последовательные шаги:

Шаг 1: Инициировать все контексты всех нейронов скрытого слоя .

Шаг 2: Подать первый набор чисел на вход нейронной сети . Вычислить для него выходы скрытого слоя.

$$y_j = f(\sum_{i=1}^{I} \omega_{ij} x_{1i} + \beta_j + \omega_j x_j)$$

где f(x) – нелинейная активационная функция $y_j = \frac{1}{1+e^{-\alpha s_j}}$.

Похожая статья: Создание и обучение нейронных сетей в системе Matlab

Шаг 3: Если текущий набор чисел не является последним, то переход на шаг 5, иначе переход на шаг 4.

Шаг 4: Записать выходы нейронов скрытого слоя на контексты $x_j = y_j$, где . Переход к шагу 2 для следующего набора чисел.

Шаг 5: Вычислить выход нейрона выходного слоя.

$$y_k = f(\sum_{j=1}^{J} \omega_{jk} y_j + \beta_k)$$

Рассмотрим задачу, которая состоит в распознавании чисел от 0 до 9. Для распознавания одного числа нужно построить собственную нейронную сеть. И так должно построить 10 нейронных сетей. Надиктована база из 250 слов (числа от 0 до 9) с различными вариациями произношения. База случайным образом разделялась на две равные части – обучающую и тестирующую выборки. При обучении нейронной сети распознаванию одного числа, например 5, желаемый выход этой нейронной сети должен быть единицей для обучающей выборки с числом 5, а остальные – нулю.

Обучение нейронной сети осуществляется путём последовательного предъявления обучающей выборки, с одновременной подстройкой весов в соответствии с определённой процедурой, пока ошибка настройки по всему множеству не достигнет приемлемого низкого уровня. Функции ошибки в системе будет вычисляться по следующей формуле:

$$E = \frac{1}{2N} \sum_{i=1}^{N} (y_{ki} - d_i)^2$$

где N – количество обучающих выборок, обработанных нейронной сетью примеров;

– реальный выход нейронной сети;

– желаемый (идеальный) выход нейронной сети.

Для каждого слова из тестовой выборки реальные выходы вычисляются 10 нейронными сетями распознавания разных чисел. Нейронная сеть, которая имеет максимальное выходное значение, и является нейронной сетью распознавания данного слова. И слово, распознанное нейронной сетью, является результатом распознания. Применение генетических алгоритмов для обучения нейронных сетей

Алгоритм обучения нейронной сети: необходимо произвести итерационную подстройку матрицы весов, постепенно уменьшая ошибку в векторах на выходе. Для обучения данной нейронной сети не может использоваться алгоритм с обратным распространением ошибки и его аналоги. Впервые в 1989 году Дэвид Монтана и Лоуренс Дэвис использовали генетические алгоритмы в качестве средства подстройки весов скрытых и выходных слоёв для фиксированного набора связей.

Рассмотрим как используются генетические алгоритмы для подстройки весов скрытых и выходных слоёв. Каждая хромосома (нейронная сеть) представляет собой вектор из весовых коэффициентов. Хромосома состоит из генов, которые могут иметь числовые значения. Приспособленность соответствует функции ошибки E.

Популяцией называют набор хромосом (решений). Эволюция популяций – это чередование поколений, в которых хромосомы изменяют свои признаки, чтобы каждая новая популяция наилучшим способом приспосабливалась к внешней среде.

Для генерации новых популяций к начальной популяции применяются основные генетические операторы:

- Оператор селекции осуществляет отбор хромосом в соответствии со значениями их функции приспособленности.
- Оператор скрещивания определяет передачу признаков родителей потомкам.

- Оператор мутации предназначен для того, чтобы поддерживать разнообразие особей популяции.
- Оператор инверсии заключается в том, что хромосома делится на две части, и затем они меняются местами.

Теперь, зная как интерпретировать значения генов, перейдем к описанию функционирования генетического алгоритма. Рассмотрим схему функционирования генетического алгоритма в его классическом варианте.

Шаг 1: Инициировать начальный момент времени t=0. Случайным образом сформировать начальную популяцию, состоящую из особей.

Шаг 2: Вычислить приспособленность каждой особи и популяции в целом. Значение этой функции определяет, насколько хорошо подходит особь, описанная данной хромосомой, для решения задачи.

Шаг 3: Выбрать одну особь из популяции.

Шаг 4: С определенной вероятностью скрещивания выбрать вторую особь из популяции и произвести оператор скрещивания над двумя хромосомами.

Шаг 5: С определенной вероятностью мутации выполнить оператор мутации над новой хромосомой.

Шаг 6: С определенной вероятностью инверсии выполнить оператор инверсии над новой хромосомой.

Шаг 7: Поместить полученную хромосому в новую популяцию.

Шаг 8: Если выполнилось условие останова, то завершить работу, иначе увеличить номер текущей эпохи t=t+1и переход на шаг 3.

Наибольшую роль в успешном функционировании алгоритма играет этап отбора родительских хромосом на шагах 3 и 4. Другой важный момент – определение критериев остановки.

## 4. Алгоритм распознавания речи на основе Mel-frequency cepstral coefficients - MFCC

В последнее время наблюдается значительный рост интереса к технологиям, связанным с распознаванием речи. Можно назвать несколько причин этого роста, в частности, значительное рост вычислительных возможностей и обучающего материала.

Постановка задачи. Существуют множество методов распознавания речи, в подавляющем большинстве случаев они основаны на методах статистического анализа и теории вероятностей (Hidden Markov Model, Gaussian Mixture Model и т.п.). Как известно, компания google предоставляет бесплатный сервис по распознаванию коротких речевых сообщений. На основе этого сервиса было даже предложено распознавание речи при помощи микроконтроллера: Распознавание речи на STM32F4-Discovery. Однако, возникает вопрос: есть ли возможность сделать свою систему распознавания речи, пусть даже на довольно ограниченном по размеру словаре, без использования «внешних» сервисов, при этом чтобы она работала быстро и с приемлемым качеством?

**Основная идея.** Итак, для распознавания речи будем использовать MFCC. Чтобы не вдаваться в подробности скажу, что относиться к ним стоит лишь как к некоторому фильтру, на входе которого — фонограмма, на выходе — набор векторов (коэффициенты), который мы и будем распознавать как некоторое слово или набор слов. Справедливости ради стоит отметить, что существуют множество других акустических признаков, использующихся для распознавания речи: Perceptual linear predictive (PLP), Linear prediction cepstral coefficient (LPCC), Linear frequency cepstral coefficients (LFCC).

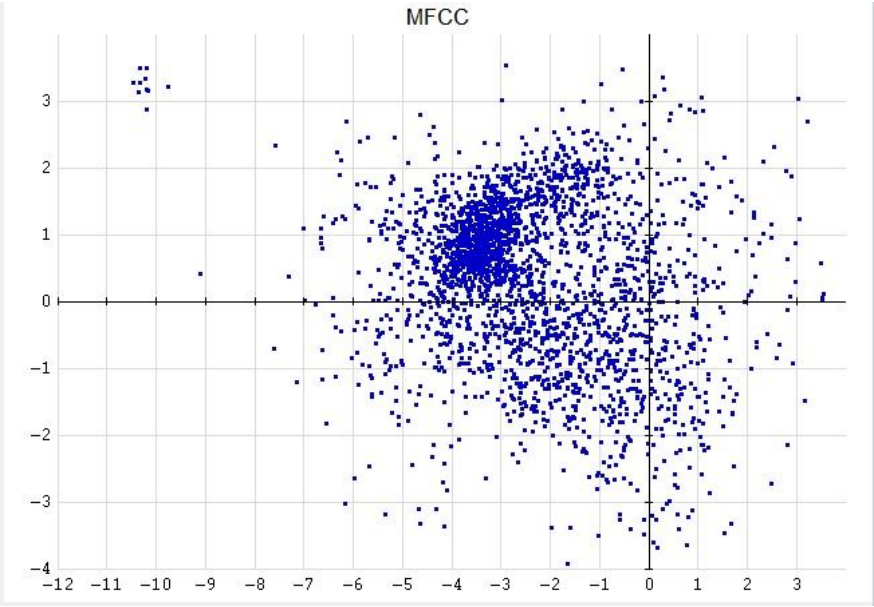
Основная идея заключается в использовании линейного дискриминантного анализа для идентификации слова. Однако, он применим лишь для векторов одинаковой размерности. Т.к. слова могут быть различной длины, возникает вопрос: каким образом преобразовать последовательность произвольного числа MFCC-векторов в вектор фиксированной размерности?

Можно поступить следующим образом: находить места «сгущения» распределения этих векторов и в качества результирующего вектора брать конкатенацию векторов, являющихся центрами «сгущений». Такой конкатенированный вектор будем называть супер вектором средних, а сами центры — средними значениями. При этом в качестве «отправной точки» будем использовать супер вектор средних, полученный на всех MFCC-векторах всей базы обучения. Преобразовав таким образом последовательность MFCC-векторов в один супер вектор средних фиксированной размерности, мы можем применять различные методы классификации.

Очевиден принципиальный недостаток такого подхода: не учитывается динамика распределения MFCC-признаков по времени, следовательно, система априори не способна различать, к примеру, слова «главрыба» и «абырвалг», т.к. общее распределение MFCC-векторов таких слов будет примерно одинаковым (соответственно, центры «сгущений» будут совпадать).

**Описание алгоритма.** В качестве базы обучения будем использовать множество файлов, каждый из которых представляет собой набор MFCC-векторов, полученных из фонограммы с записью того или иного слова. При этом файлы с записью одного и того же слова должны быть объединены в одну группу.

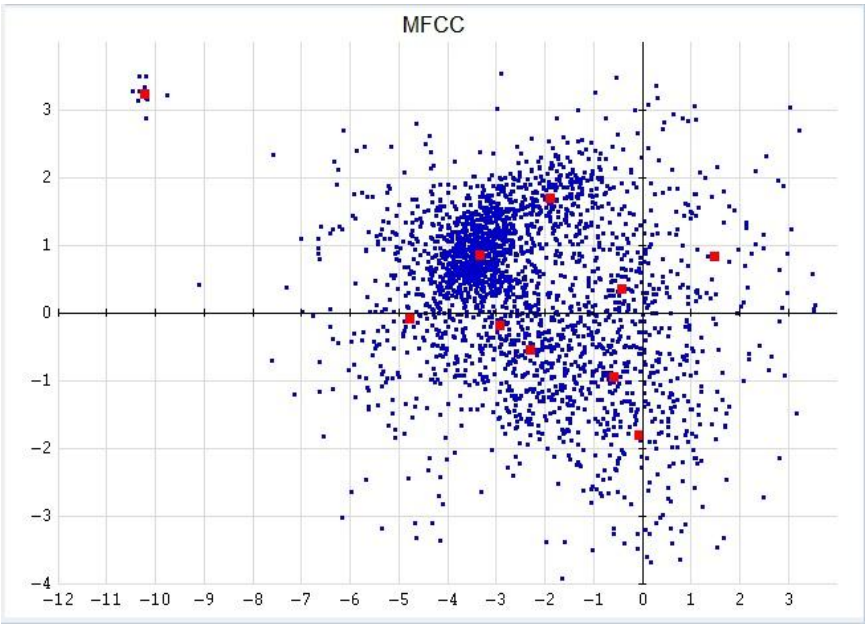
Вот как выглядит распределение первых двух компонент MFCC-векторов всей базы обучения:




Алгоритм состоит из следующих этапов:

Находим супер вектор средних для всей базы обучения при помощи алгоритма K-средних.

Пример работы алгоритма K-средних для K=10 представлен на рисунке:



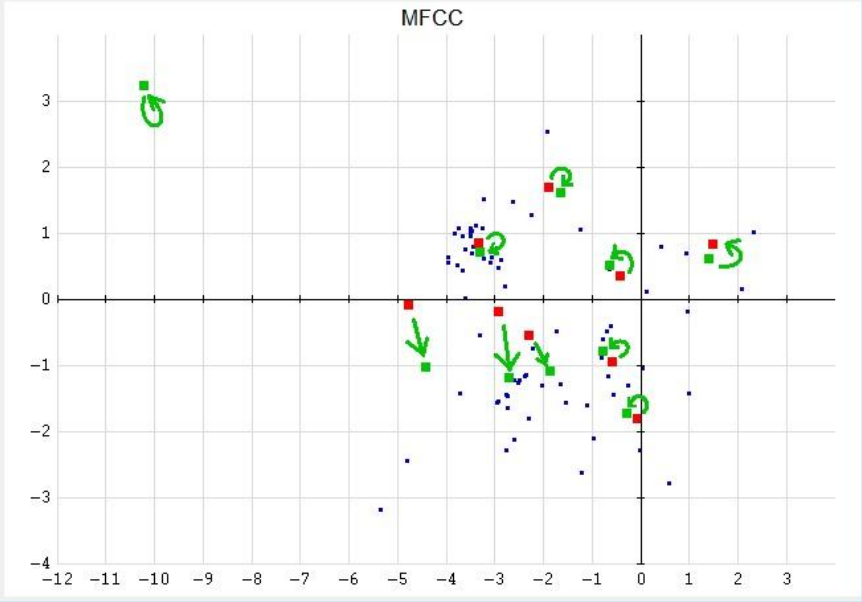

где большие красные квадраты и есть искомые средние значения.

Для каждого файла базы находим собственные средние значения по формуле:

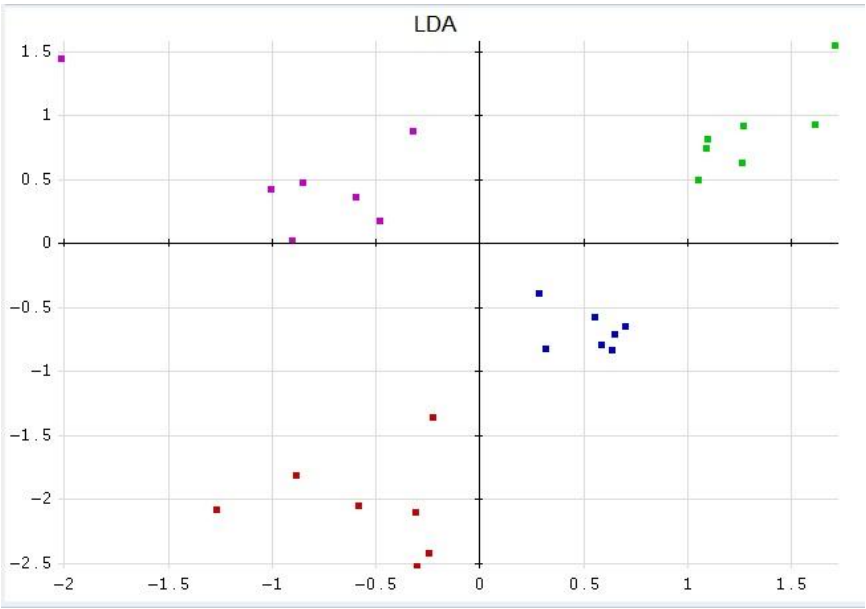
Mk = a * Mk0 + (1 — a) * Mk', k = 1:K

где Mk0 — среднее значение, найденное в п.1, Mk' — среднее значение, полученное в результате применения одной итерации алгоритма K-средних для MFCC-векторов файла с использованием в качестве начального значения Mk0, a = R/(R + Nk), где R — коэффициент «чувствительности», Nk — число MFCC-векторов, соответствующие среднему значению Mk'. Найденные таким образом средние значения будем называть адаптированными средними значениями.

Пример адаптированных средних значений для файла представлен на рисунке:




Имея теперь вместо исходных фонограмм адаптированные супер вектора средних, проводим LDA для N классов (каждый класс соответствует одному слову). В результате мы должны получить матрицу, состоящую из векторов нового базиса, при проекции на который исходные адаптированные супер вектора средних должны достаточно хорошо разделяться. Пример для N=4:




Проецируем все адаптированные супер вектора средних на новый базис и находим средние значения и СКО (среднее квадратичное отклонение) проекций для каждого класса.

Для определения принадлежности тестовой фонограммы тому или иному классу (т.е. распознавания), выполняем для неё пп. 2 и 4, далее находим расстояния полученной проекции до средних значений всех классов (можно дополнительно нормировать их на соответствующее СКО). Минимальное расстояние и будет соответствовать классу, к которому принадлежит тестовая фонограмма.

Создание собственной системы распознавания слов состоит из следующих этапов:

**Запись фонограмм для обучения и тестирования.** Для записи можно воспользоваться любой программой, умеющей записывать звук и сохранять его в формате WAVE. Я рекомендую использовать бесплатную программу Audacity.

Разработанная система не умеет выделять речевые сегменты, поэтому при записи нужно стараться, чтобы в фонограмме присутствовала только речь. Чем качественнее используется микрофон, тем качественнее получается система. Записывать необходимо в моно-режиме с частотой дискретизации 16000.

**Построение MFCC-векторов.** Для построения MFCC-векторов можно использовать бесплатную библиотеку SPro 5.0. Я взял на себя ответственность, немного перебрал эту библиотеку, исправил парочку ошибок и сделал сборку программы sfbcep.exe под windows (см. папку ../spro-5.0). 32-разрядная версия этой программы лежит в папке ../tools. Для построения MFCC-векторов я использовал следующие параметры:

**Обучение и тестирование системы.** Для обучения и тестирования системы я написал программу wrsystem на языке C++. Полный исходный код находится в папке ../wrsystem. 32-разрядную версию этой программы можно взять в папке ../tools.

Релизация алгоритма LDA была позаимствована из библиотеки ALGLIB.

Программа wrsystem имеет два режима работы: обучение (в случае наличия параметра --learn) и тестирование. Эта программа принимает на вход три основных параметра:

— Путь к файлу с описанием базы обучения (тестирования) (параметр --base). Пример файла с описанием базы лежит в папке ../base, также описание формата можно посмотреть, запустив программу с параметром --help.

— Путь к бинарному файлу, хранящему результат обучения системы (параметр -system). В режиме обучения этот файл создается, в режиме тестирования — считывается.

— Путь к файлу, в который записывается результаты тестирования системы на указанной базе: матрица перепутывания и значение WER (Word Error Rate) (параметр --test_results).

## Выводы по второй главе

В результате проделанной работе во второй главе были рассмотрены следующие вопросы:

- Перспективы развития систем распознавания речи;

- Поиск оптимальной системы анализ речи с открытыми и закрытыми исходными кодами, для дальнейшей интеграции;
- Распознавание речи на основе искусственных нейронных сетей;
- Алгоритм распознавания речи на основе Mel-frequency cepstral coefficients – MFCC;

## III Глава. Программная реализация составление баз данных фонем узбекского языка

### 1. Методология применение алгоритма Google Speech Voice API в целях программной реализации

Последнее время большой интерес у пользователей вызывает возможность распознавания речи в телефонах. Огромная заслуга в популяризации этого направления принадлежит компании Aple, однако Google также располагает подобными технологиями. Собственно этой теме и будет посвящена данная статья. Мы разработаем приложение, которое будет распознавать речь пользователя и воспроизводить результат с помощью голосового движка “Text To Speech” (TTS). Отметим, что распознавание происходит на серверах Google, поэтому для работы приложению необходимо разрешить использовать коммуникационные возможности. Кроме того, распознавание речи не работает на эмуляторе. Тестировать программу необходимо на реальном устройстве.

На самом деле работать с распознаванием и синтезом речи в Android очень просто. Все сложные вычисления скрыты от нас в довольно элегантную библиотеку с простым API. Вы сможете осилить этот урок, даже если имеете весьма поверхностные знания о программировании для Android.

Давайте создадим новый проект в Eclipse. Для наших нужд понадобится версия SDK не меньше 8. Опишем в общих чертах создаваемую программу. При запуске приложения пользователю будет показана кнопка, после нажатия на которую пользователю будет предложено надиктовать фразу. Затем будет осуществлено распознавание и будет показан список возможных вариантов. Поскольку технологии распознавания речи далеки от совершенства, программа не может ручаться за точность результата, именно поэтому будет предложено несколько вариантов. После того, как пользователь выберет один из них, будет запущен генератор голоса, который воспроизведет выбранную фразу.

Нам понадобится несколько текстовых строк, объявим их в фале “res/values/strings.xml”

```
<resources>
<string name="intro">Press the button to speak!</string>
<string name="app_name">SpeechRepeat</string>
<string name="speech">Speak now!</string>
<string name="word_intro">Suggested words&#8230;</string>
</resources>
```

Откроем файл “res/layout/main.xml” и зададим шаблон дизайна приложения. Для этого переключимся из графического в XML редактор и изменим содержимое файла

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
android:orientation="vertical"
android:background="#ff330066"
android:paddingBottom="5dp" >
</LinearLayout>
Добавим в Linear Layout элемент Text View
<TextView android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="@string/intro"
android:padding="5dp"
android:textStyle="bold"
android:textSize="16dp"
android:gravity="center"
android:textColor="#ffffff33" />
```

обратите внимание, TextView ссылается на строку intro, которую мы задали в файле strings.xml.

После Text View добавим кнопку

```
<Button android:id="@+id/speech_btn"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:text="@string/speech" />
```

Пользователь будет нажимать эту кнопку, чтобы начать говорить. Кнопка имеет параметр id, через который ее можно вызвать из Java кода. После нажатия на кнопку пользователю показывается сообщение. Нам также понадобится TextView для вывода слов с предложениями

```
<TextView android:layout_width="fill_parent"
```

```
android:layout_height="wrap_content"
android:padding="5dp"
android:text="@string/word_intro"
android:textStyle="italic" />
```

TextView будет использовать строковый ресурс. Нам также понадобится список для вариантов слов

```
<ListView android:id="@+id/word_list"
android:layout_width="fill_parent"
android:layout_height="0dip"
android:layout_weight="1"
android:paddingLeft="10dp"
android:paddingTop="3dp"
android:paddingRight="10dp"
android:paddingBottom="3dp"
android:layout_marginLeft="20dp"
android:layout_marginRight="20dp"
android:layout_marginTop="5dp"
android:layout_marginBottom="5dp"
android:background="@drawable/words_bg" />
```

ListView будет заполняться данными в процессе работы программы, поэтому для доступа к этому компоненту также требуется ID. Обратите также внимание на наличие ресурса drawable. Вы должны сохранить файл words_bg.xml в папке res

```
<shape xmlns:android="http://schemas.android.com/apk/res/android"
android:dither="true">
<gradient
android:startColor="#ff000000"
android:endColor="#ff000000"
android:centerColor="#00000000"
android:angle="180" />
<corners android:radius="10dp" />
<stroke
android:width="2dp"
android:color="#66ffffff" />
</shape>
```

Ничего особенного. Вы можете настроить дизайн ListView по своему усмотрению. Нам осталось задать еще один элемент пользовательского интерфейса – шаблон для элемента ListView. Создайте новый файл res/layout/word.xml со следующим содержанием

```
<TextView xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="fill_parent"
```

```
android:layout_height="fill_parent"
android:gravity="center"
android:padding="5dp"
android:textColor="#ffffffff"
android:textSize="16dp" >
</TextView>
```

Таким образом, каждый элемент списка представляет собой просто Text View.

Если Вы все сделали правильно, то при запуске должно получиться следующее


Программируем распознавание речи в Android

После того, как шаблон будущего приложения создан, можно перейти к кодированию. Откройте java файл главной Activity и добавьте в начало файла

```
import java.util.ArrayList;
import java.util.List;
import java.util.Locale;
import android.app.Activity;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.content.pm.ResolveInfo;
import android.os.Bundle;
```

```
import android.speech.RecognizerIntent;
import android.speech.tts.TextToSpeech.OnInitListener;
import android.speech.tts.TextToSpeech;
import android.util.Log;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.ListView;
import android.widget.Toast;
import android.widget.TextView;
```

Изменим немного декларацию главного класса

public class SpeechRepeatActivity extends Activity implements OnClickListener, OnInitListener {

OnInitListener необходим для работы TTS движка. Внутри класса добавим объявления переменных перед методом onCreate

```
//переменная для проверки возможности
//распознавания голоса в телефоне
private static final int VR_REQUEST = 999;
//ListView для отображения распознанных слов
private ListView wordList;
//Log для вывода вспомогательной информации
private final String LOG_TAG = "SpeechRepeatActivity";
//***здесь можно использовать собственный тег***
//переменные для работы TTS
//переменная для проверки данных для TTS
private int MY_DATA_CHECK_CODE = 0;
//Text To Speech интерфейс
private TextToSpeech repeatTTS;
```

Внутри метода onCreate автоматически сгенерирован код, вызывающий метод родительского класса и устанавливающий главный контекст вывода.

```
/вызов суперкласса
super.onCreate(savedInstanceState);
//установка контекста вывода
setContentView(R.layout.main);
```

Создадим переменные для работы с кнопкой и списком распознанных слов

```
Button speechBtn = (Button) findViewById(R.id.speech_btn);
```

```
wordList = (ListView) findViewById(R.id.word_list);
```

Далее необходимо проверить поддерживается ли возможность распознавания голоса телефоном

```
//проверяем, поддерживается ли распознование речи
PackageManager packManager = getPackageManager();
List<ResolveInfo> intActivities = packManager.queryIntentActivities(new
  Intent(RecognizerIntent.ACTION_RECOGNIZE_SPEECH), 0);
if (intActivities.size() != 0) {
// распознавание поддерживается, будем отслеживать событие щелчка по кнопке
speechBtn.setOnClickListener(this);
}
else
{// распознавание не работает. Заблокируем
// кнопку и выведем соответствующее
// предупреждение.
speechBtn.setEnabled(false);
Toast.makeText(this, "Oops - Speech recognition not supported!",
Toast.LENGTH_LONG).show();
}
```

Мы запрашиваем среду, поддерживается ли Recognizer Intent. Если поддерживается, мы говорим приложению, что нужно отслеживать щелчок пользователя по кнопке. Если интент не поддерживается, мы блокируем кнопку и выводим соответствующее сообщение пользователю.

Напишем код, обрабатывающий нажатие на кнопку. Внутри класса после метода OnCreate добавим метод OnClick.

```
public void onClick(View v) {
if (v.getId() == R.id.speech_btn) {
  // отслеживаем результат
  listenToSpeech();
}}
```

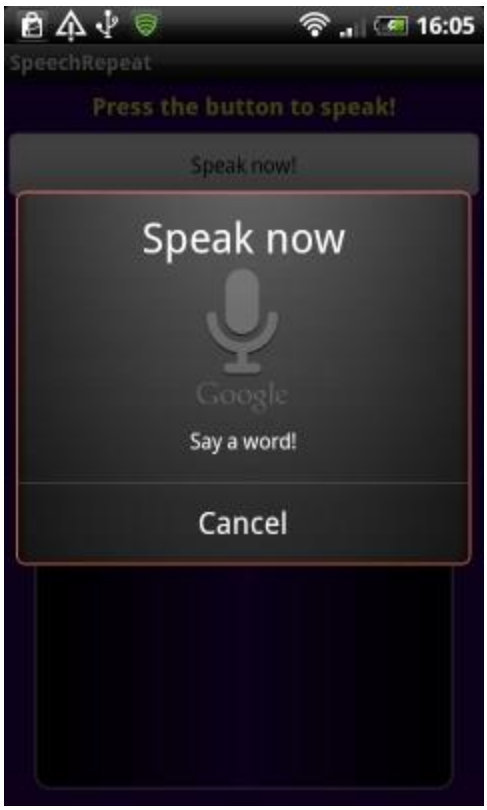
Как видите, при нажатии на кнопку мы вызываем метод listenToSpeech().

```
private void listenToSpeech() {
//запускаем интент, распознающий речь и передаем ему требуемые данные
Intent listenIntent = new Intent(RecognizerIntent.ACTION_RECOGNIZE_SPEECH);
//указываем пакет
listenIntent.putExtra(RecognizerIntent.EXTRA_CALLING_PACKAGE,
    getClass().getPackage().getName());
//В процессе распознования выводим сообщение
listenIntent.putExtra(RecognizerIntent.EXTRA_PROMPT, "Say a word!");
//устанавливаем модель речи
```

```
  listenIntent.putExtra(RecognizerIntent.EXTRA_LANGUAGE_MODEL,
    RecognizerIntent.LANGUAGE_MODEL_FREE_FORM);
//указывае число результатов, которые могут быть получены
listenIntent.putExtra(RecognizerIntent.EXTRA_MAX_RESULTS, 10);
//начинаем прослушивание
startActivityForResult(listenIntent, VR_REQUEST);
}
```

Большая часть приведенного кода стандартна для программ, использующих распознавание голоса. Обратите внимание на параметр EXTRA_PROMPT. Он задает строку-приглашение для пользователя. Параметр EXTRA_MAX_RESULTS определяет максимальное число вариантов распознавания. В конце концов, мы вызываем startActivityForResult. Результат его работы будет передан в метод onActivityResult.

На следующем скриншоте показан экран в момент распознавания речи.



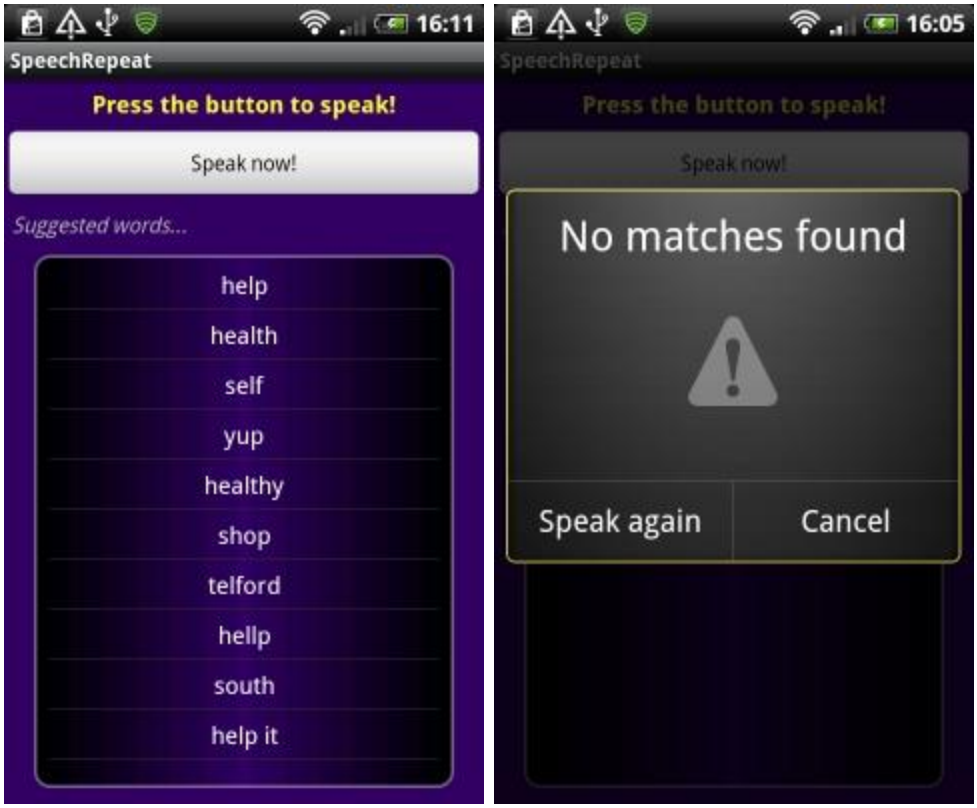

Определим метод onActivityResult

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data)
{
//проверяем результат распознавания речи
```

```
        if (requestCode == VR_REQUEST && resultCode == RESULT_OK)
        {  //Добавляем распознанные слова в список результатов
          ArrayList<String> suggestedWords =
               data.getStringArrayListExtra(RecognizerIntent.EXTRA_RESULTS);
          //Передаем список возможных слов через ArrayAdapter компоненту ListView
          wordList.setAdapter(new     ArrayAdapter<String>     (this,     R.layout.word,
suggestedWords));}//tss код здесь//вызываем метод родительского класса
        super.onActivityResult(requestCode, resultCode, data);}
```

Обратите внимание, при проверке результата мы сравниваем переменную requestCode с константой VR_REQUEST, которую использовали ранее при вызове метода startActivityForResult. Таким образом, мы рассматриваем только результаты от нашего запроса. В метод возвращается 10 вариантов распознанных слов, которые мы записываем в список ArrayList. Этот список мы используем в ArrayAdapter компонента List View.

Если приложение справилось с задачей и смогло что-то распознать, вы увидите похожий н показанный на левом скриншоте результат. Если приложению не удалось распознать фразу, будет показано сообщение, как на правом скриншоте



Вот, собственно и все. Распознавание голоса в Android – довольно простая задача. Мы вызываем интент RecognizerIntent с требуемыми нами параметрами. Результат возвращается в onActivityResult.

Генерация речи в Android

Перейдем ко второй части нашего приложения, связанного с генерацией речи. Мы хотим, чтобы телефон проговаривал фразу из списка результатов. Мы должны определить строку, на которую щелкнул пользователь. Вернемся к методу onCreate и добавим в конец этого метода код

```
        //засекаем щелчок пользователя по слову из списка
        wordList.setOnItemClickListener(new OnItemClickListener() {
        //метод вызывается в ответ на щелчок по слову
        public void onItemClick(AdapterView<?> parent, View view, int position, long
id)
        {  //записываем в переменную TextView строки
          TextView wordView = (TextView)view;
          //получаем строку с текстом
          String wordChosen = (String) wordView.getText();
          //выводим ее в лог для отладки
          Log.v(LOG_TAG, "chosen: "+wordChosen);
          //выводим Toast сообщение
          Toast.makeText(SpeechRepeatActivity.this, "You said: "+wordChosen,
              Toast.LENGTH_SHORT).show();
        }});
```

Мы используем метод setOnItemClickListener чтобы установить отслеживание щелчков для каждой строки. Внутри нового объекта OnItemClickListener мы описали метод onItemClick, который вызывается в ответ на щелчок по строке списка. Выбранная строка передается, как View в этот метод. Поскольку при проектировании шаблона приложения мы указали, что наш список состоит из TextView, мы преобразуем полученное значение в объект TextView и достаем из него строковое значение. Мы записываем это слово в лог и показываем пользователю Toast сообщение.

Если Вас не интересует процесс генерации речи, Вы можете остановиться и протестировать приложение.

Для генерации речи необходимо настроить движок TTS. Добавим код в конец метода onCreate

```
//подготовка движка TTS для проговаривания слов
Intent checkTTSIntent = new Intent();
//проверка наличия TTS
checkTTSIntent.setAction(TextToSpeech.Engine.ACTION_CHECK_TTS_DATA);
//запуск checkTTSIntent интента
startActivityForResult(checkTTSIntent, MY_DATA_CHECK_CODE);
```

Как и в случае распознавания, результат интента возвращается в метод onActivityResult. В этом методе перед строкой super.onActivityResult(requestCode, resultCode, data); добавьте

```
//returned from TTS data check
if (requestCode == MY_DATA_CHECK_CODE)
{ //все необходимые приложения установлены, создаем TTS
if (resultCode == TextToSpeech.Engine.CHECK_VOICE_DATA_PASS)
  repeatTTS = new TextToSpeech(this, this);
//движок не установлен, предположим пользователю установить его
else{   //интент, перебрасывающий пользователя на страницу TSS в Google Play
  Intent instalTTSIntent = new Intent();
  installTTSIntent.setAction(TextToSpeech.Engine.ACTION_INSTALL_TTS_DATA);
  startActivity(installTTSIntent); } }
```

Таким образом, мы проверяем наличие TTS движка, и если он не установлен – предлагаем пользователю установить соответствующую программу.

Чтобы завершить настройку TTS, добавим метод onInit, который вызывается при успешной инициализации TTS.

```
public void onInit(int initStatus) {
if (initStatus == TextToSpeech.SUCCESS)
  repeatTTS.setLanguage(Locale.UK); //Язык
}
```

**Здесь мы устанавливаем язык генератора речи.** Таким образом, одновременно с Toast сообщением пользователь услышит сгенерированную речь. Отметим ещё раз, что эмулятор не поддерживает распознавание речи, поэтому тестировать программу необходимо на телефоне.

## 2. Исследование и обоснование среды разработок под Android OS

**Исследование сред разработок под Android OS.** Обзор сред программирования. Прежде чем начать разрабатывать приложения под Android, рассмотрим существующие инструменты, подходящие для этих целей. Можно выделить необходимые инструменты, без которых разработка мобильных приложений под Android просто невозможна. С другой стороны, существует большое количество вспомогательных систем, в какой-то мере упрощающих процесс разработки.

К обязательным инструментам относится Android SDK - набор средств программирования, который содержит инструменты, необходимые для создания, компиляции и сборки мобильного приложения.

Рассмотрим кратко наиболее важные инструменты, входящие в состав Android SDK:

- SDK Manager - инструмент, позволяющий загрузить компоненты Android SDK. Показывает пакеты Android SDK и их статус: установлен (Installed), не установлен (Not Installed), доступны обновления (Update available).

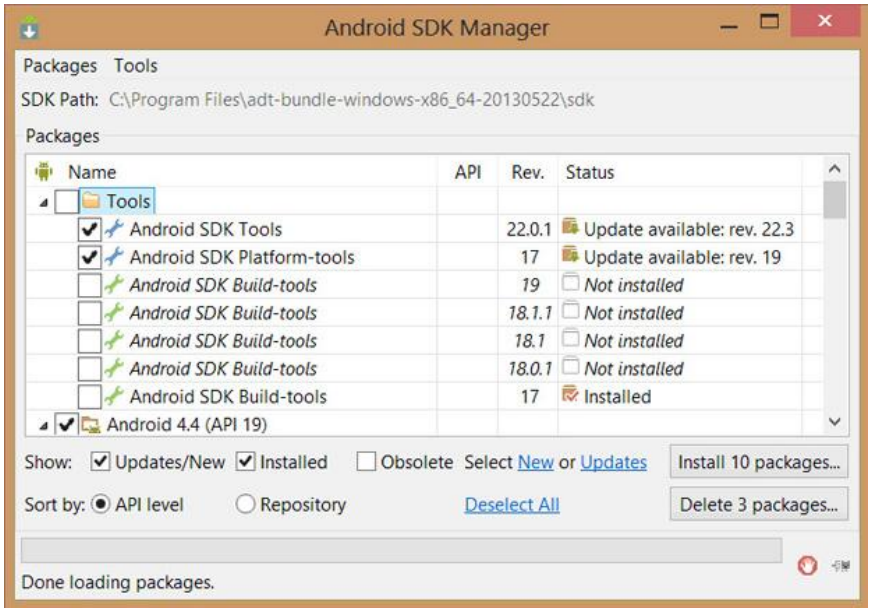



Рис. 2.1. Android SDK Manager

- Debug Monitor - самостоятельный инструмент, предоставляющий графический интерфейс к нескольким инструментам, предназначенным для анализа и отладки Android приложений:
  - DDMS (Dalvik Debug Monitor Server) предоставляет услуги переброса портов, захват экрана устройства, информацию о потоках и динамической памяти устройства, вывод информации о действиях Android в реальном времени (logcat) и многое другое.
  - Hierarchy Viewer позволяет отлаживать и оптимизировать пользовательский интерфейс Android приложения.
  - Tracer for OpenGL ES - инструмент для анализа OpenGL|ES кода, используемого в мобильном приложении, позволяет захватывать команды OpenGL|ES и демонстрировать их по отдельным кадрам, что помогает понять как исполняются графические команды.

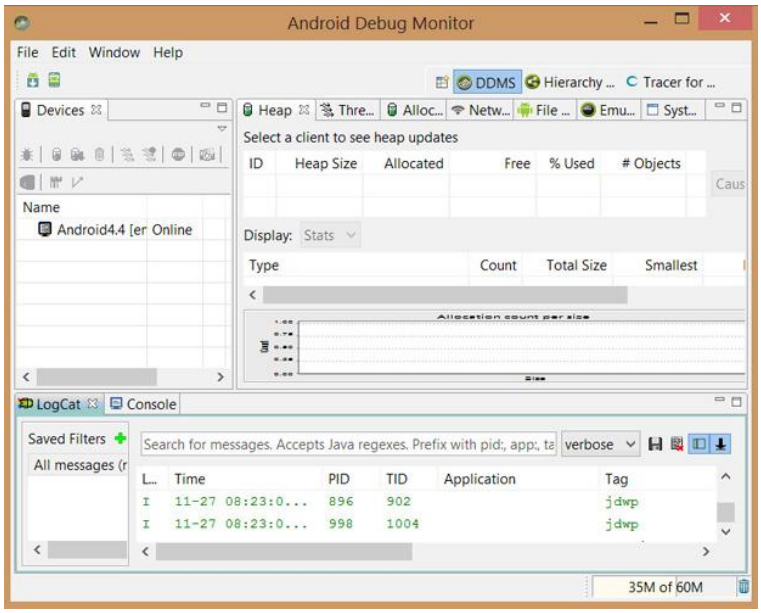



Рис. 2.2. Окно инструмента Monitor

- Android Emulator (emulator) - виртуальное мобильное устройство, которое создается и работает на компьютере разработчика, используется для разработки и тестирования мобильных приложений без привлечения реальных устройств.

- AVD Manager - предоставляет графический интерфейс для создания виртуальных Android устройств (AVDs), предусмотренных Android Emulator, и управления ими.

(В ЛР№1 подробно рассматривается создание и использование виртуального устройства).

- Android Debug Bridge (adb) - гибкий инструмент, позволяющий управлять состоянием эмулятора или реального Android устройства, подключенного к компьютеру. Также может использоваться для установки Android приложения (.apk файл) на реальное устройство.

Мы рассмотрели основные инструменты, входящие в состав Android SDK, разумеется, не все и недостаточно подробно. Для более серьезного изучения инструментов имеет смысл обратиться к сайту разработчиков (http://developer.android.com/tools/help/index.html). Для разработки мобильных приложений под Android уверенного владения инструментами из SDK вполне достаточно. Если же возникают какие-то вопросы, дополнительные инструкции по созданию проектов, компиляции, запуску из командной строки содержатся в руководстве от Google (http://developer.android.com/training/basics/firstapp/index.html).

В современных условиях разработка ПО в большинстве случаев ведется с использованием интегрированных сред разработки (IDE).IDE имеют несомненные достоинства: процесс компиляции, сборки и запуска приложения обычно автоматизирован, в связи с чем для начинающего разработчика создать свое первое приложение труда не составляет. Но чтобы заниматься разработкой всерьез, необходимо потратить силы и время на изучение возможностей самой среды. Рассмотрим IDE, пригодные для разработки под Android[1].

Для начала поговорим о двух средах разработки, которые рекомендует Google: Android IDE (ADT) и Android Studio.

Android IDE - среда разработки под Android, основанная на Eclipse. Предоставляет интегрированные инструменты для разработки, сборки и

отладки мобильных приложений. В данном курсе Android IDE выбрана в качестве основной среды разработки. Возможности этой среды более подробно рассмотрены в первой лабораторной работе. Также там даны рекомендации по установке и настройке среды, созданию и запуску первого приложения как на эмуляторе, так и на реальном устройстве.

Android Studio - среда разработки под Android, основанная на IntelliJ IDEA. Подобно Android IDE, она предоставляет интегрированные инструменты для разработки и отладки. Дополнительно ко всем возможностям, ожидаемым от IntelliJ, в AndroidStudio реализованы:

- поддержка сборки приложения, основанной на Gradle;
- специфичный для Android рефакторинг и быстрое исправление дефектов;
- lint инструменты для поиска проблем с производительностью, с юзабилити, с совместимостью версий и других;
- возможности ProGuard (утилита для сокращения, оптимизации и обфускации кода) и подписи приложений;
- основанные на шаблонах мастера для создания общих Android конструкций и компонентов;
- WYSIWYG редактор, работающий на многих размерах экранов и разрешений, окно предварительного просмотра, показывающее запущенное приложение сразу на нескольких устройствах и в реальном времени;
- встроенная поддержка облачной платформы Google.

Загрузить последнюю версию Android Studio, а также получить рекомендации по установке, настройке и началу работы можно тут:http://developer.android.com/sdk/installing/studio.html.

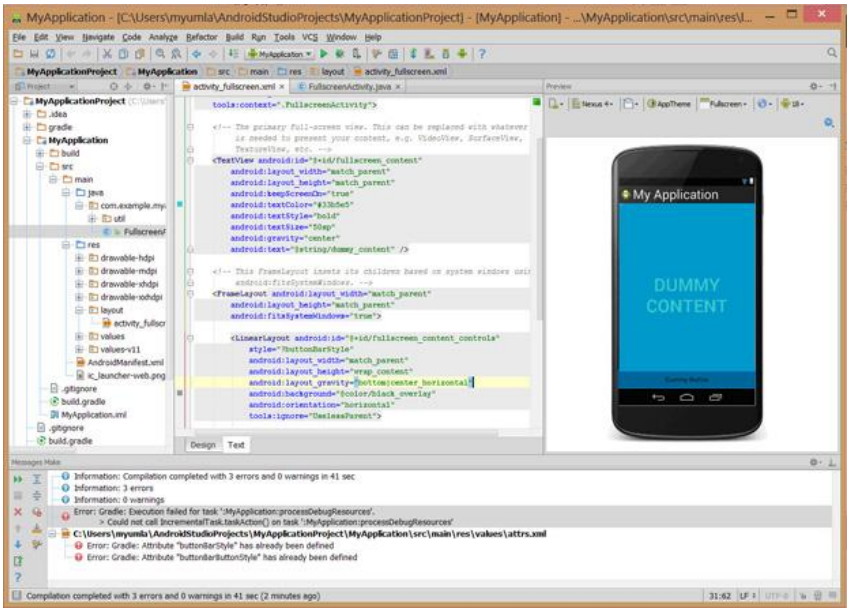



Рис. 2.3. Среда разработки Android Studio

Перейдем к рассмотрению других инструментов, пригодных для разработки мобильных приложений под Android. Начнем с инструментов от Intel - Intel XDK и Intel Beacon Mountain.

Intel XDK позволяет легко разрабатывать кроссплатформенные мобильные приложения; включает в себя инструменты для создания, отладки и сборки ПО, а также эмулятор устройств; поддерживает разработку для Android, Apple iOS, Microsoft Windows 8, Tizen; поддерживает языки разработки: HTML5 и JavaScript.

Последняя тема данного курса полностью посвящена изучению нового поколения инструментальных средств разработки мобильных HTML5-приложений и Intel XDK, предполагается разработка мобильного приложения с использованием этих инструментов.

Intel Beacon Mountain - среда разработки, позволяющая создавать приложения для устройств, работающих под управлением ОС Android. Предоставляет инструменты необходимые для проектирования, разработки, отладки и оптимизации приложений под Android. Освобождает разработчика от необходимости поддерживать систему разработки в актуальном состоянии, следит за обновлениями и добавляет их в среду

разработки по мере появления. Поддерживает разработку для целевых платформ на основе процессоров Intel Atom и ARM.

Beacon Mountain построена на основе Android IDE (Eclipse, Android ADT, Android SDK), для более серьезной разработки и оптимизации добавлены следующие инструменты Intel:

- Intel* Hardware Accelerated Execution Manager (Intel* HAXM) - аппаратно поддерживаемый процессор виртуализации, использующий технологию виртуализации Intel* (Intel* VT) для ускорения работы эмулятора в среде разработки.
- Intel* Graphics Performance Analyzers (Intel* GPA) System Analyzer поддерживает мобильные устройства с процессором Intel Atom под управлением ОС Android. Позволяет разработчикам оптимизировать загруженность системы при использовании процедур OpenGL, предоставляя возможность получать множество системных метрик в реальном времени, отображающих загруженность CPU, GPU и OpenGL ES API. Разработчик может запустить несколько графических экспериментов для выявления узких мест в обработке графики.
- Intel* Integrated Performance Primitives (Intel* IPP) Preview - библиотека оптимизированной обработки данных и изображений, поддерживающая мобильные устройства с платформой Intel под управлением ОС Android. Preview версия является частью полной версии Intel IPP, которая тоже поддерживает ОС Android.

Эмуляция (англ. emulation) в вычислительной технике - комплекс программных, аппаратных средств или их сочетание, предназначенное для копирования (или эмулирования) функций одной вычислительной системы (гостя) на другой, отличной от первой, вычислительной системе (хосте) таким образом, чтобы эмулированное поведение как можно ближе соответствовало поведению оригинальной системы (гостя). Целью является максимально точное воспроизведение поведения в отличие от разных форм

компьютерного моделирования, в которых имитируется поведение некоторой абстрактной модели (Википедия).

Эмулятор - виртуальное мобильное устройство, которое запускается на компьютере. При помощи эмулятора можно разрабатывать и тестировать приложения без использования реальных устройств.

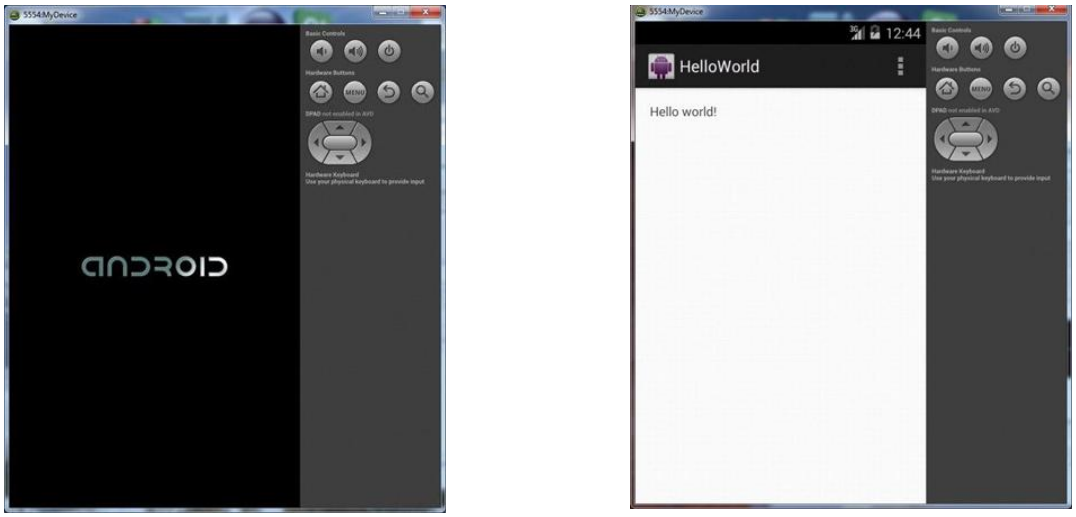



Рис. 2.5. Эмулятор Android SDK в процессе запуска и приложение "Hello, world!"

К достоинствам использования эмуляторов можно отнести простоту их использования и нулевую стоимость. Разработчику не нужно покупать огромное количество устройств с различными характеристиками, чтобы проверить работоспособность приложения на различных смартфонах. Достаточно создать несколько эмуляторов с требуемыми характеристиками и запустить на них приложение.

К сожалению, эмуляторы имеют и ряд недостатков:

- Требуют много системных ресурсов.
- Из-за различий в архитектуре процессоров компьютера и смартфона медленно запускаются. Современные персональные компьютеры построены на архитектурах x86 и x64, а большинство процессоров смартфонов на Android - ARM. Процесс эмуляции одной архитектуры на другой чрезвычайно сложен и происходит довольно медленно.
- В некоторых случаях стандартного эмулятора недостаточно. Речь идет о возможностях смартфонов, которыми обычные компьютеры не

обладают (например, наличие датчика gps или акселерометра). В таких случаях полноценную отладку можно провести только с использованием реального устройства.

**Установка среды разработки Android Studio.** Разработка Android-приложений, как и в случае с любыми другими приложениями, начинается с установки среды разработки.

Мы разделим этот этап на три шага:

Установка Java;

Установка Android Studio;

Установка эмулятора.

Шаг 1. Установка Java.

Подробно данный шаг описан в уроке #1 основного курса Java:

http://study-java.ru/uroki-java/urok-1-ustanovka-java

Шаг 2. Установка Android Studio.

Последнюю рабочую версию Android Studio скачиваем по ссылке:

https://developer.android.com/sdk/installing/studio.html

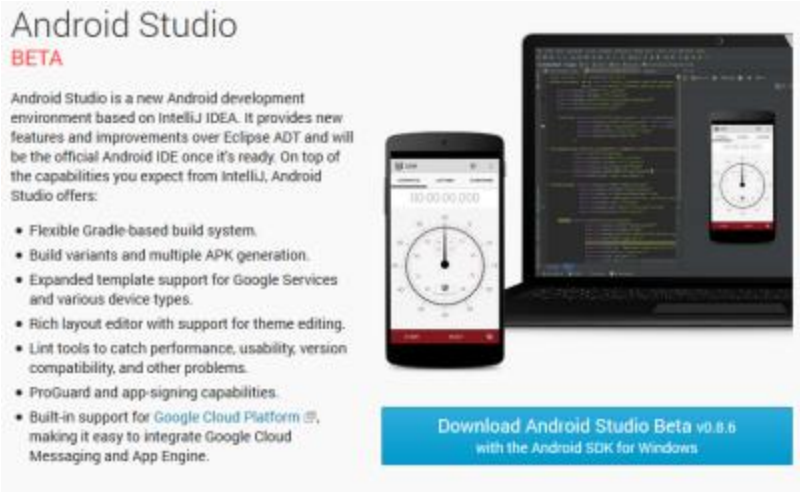



Рис. 2.7. Свойства среды программирование

Сайт сам подбирает подходящий дистрибутив для версии вашей операционной системы. Все, что требуется, — это нажать «Download Android Studio» и согласиться с условиями установки Android SDK.

Установка сопровождается подсказками «Мастера установки», и если у вас не возникает проблем на данном этапе, то вы можете перейти сразу к шагу 3.


Рис.2.8. Первый шаг установки

«Мастер» предложит вам возможные пути установки.

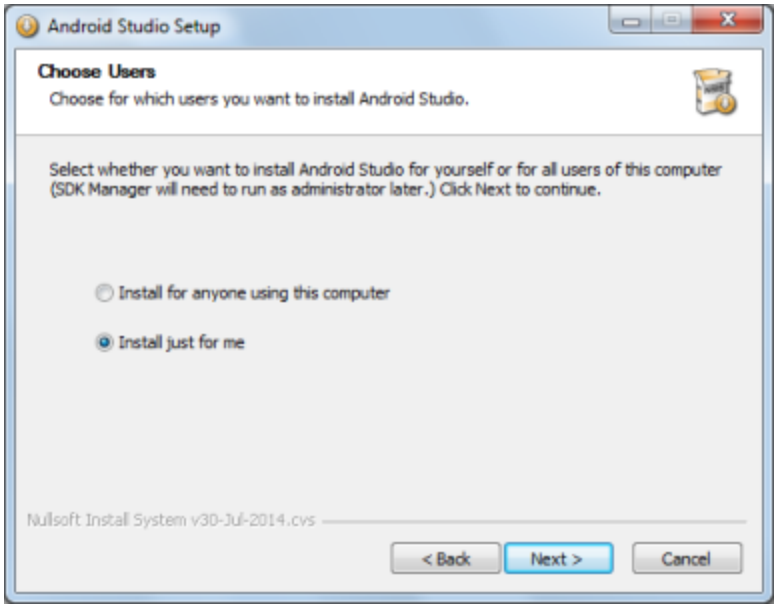



Рис.2.9. Второй шаг установки (права пользователей)

Для того, чтобы было понятно отличие предлагаемых вариантов:

Install for anyone using this computer: «Диск:\Program Files\Android\android-studio» Install just for me: «Диск:\Users\Имя_пользователя\AppData\Local\Android\android-studio»

Удостоверимся в том, что нам хватает места на жестком диске, а место, куда мы хотим произвести установку, указано верно.

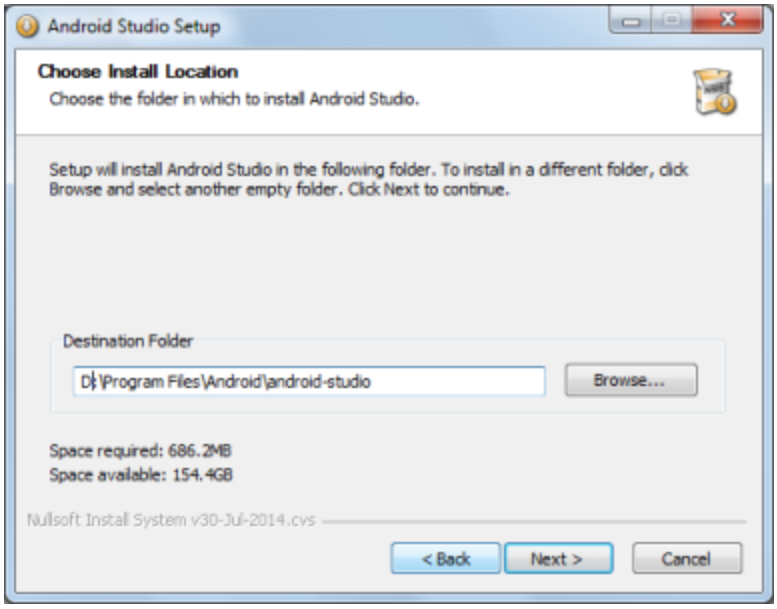



Рис.2.10. Третий шаг установки (выбирание пути установки)

Нажимаем «Next». Начнется установка

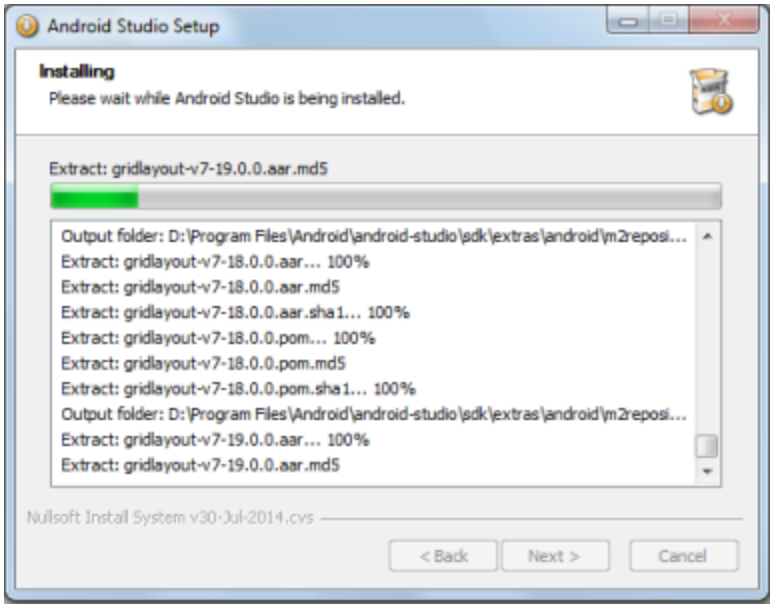



Рис.2.11. Четвертый шаг установки (процесс установки)

«Мастер» уведомит вас о завершении установки.

Android Studio готов к запуску.

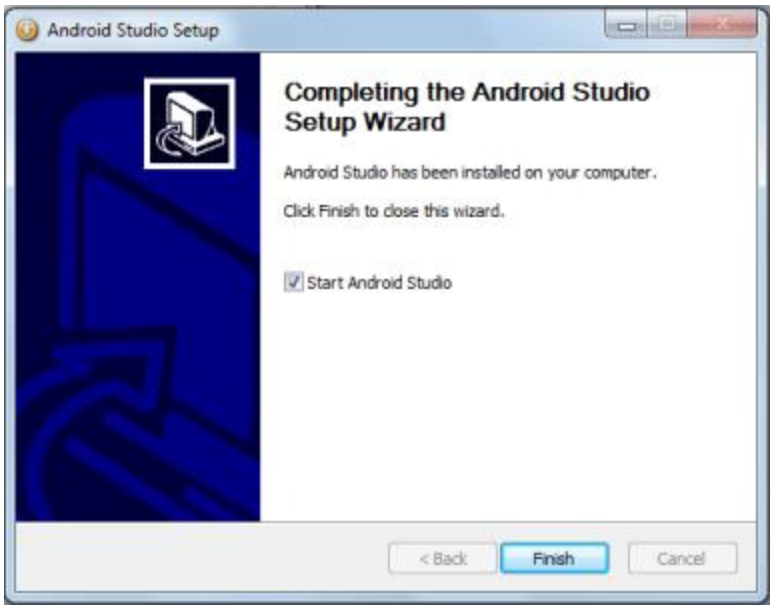



Рис.2.12. Пятый последний шаг установки

Шаг 3. Установка эмулятора.

Тестировать будущие Android-приложения можно двумя способами:

- при помощи Android-устройства;
- при помощи эмулятора.

Подробно мы рассмотрим оба эти способа в следующем уроке, а сейчас установим эмулятор Genymotion.

Рис.2.13. Логотип Genymotion

В пакете Android Studio имеется собственный эмулятор, однако его производительность оставляет желать лучшего. Установка дополнительного эмулятора не является необходимым условием написания приложения под Android. Это просто дополнительное удобство.

Скачиваем эмулятор по ссылке (требуется регистрация):

https://cloud.genymotion.com/page/launchpad/download/

Для этого необходимо выбрать подходящий дистрибутив под вашу операционную систему.

Например, в случае с Windows рекомендуется выбирать «Windows 32/64 bits (with VirtualBox)»:

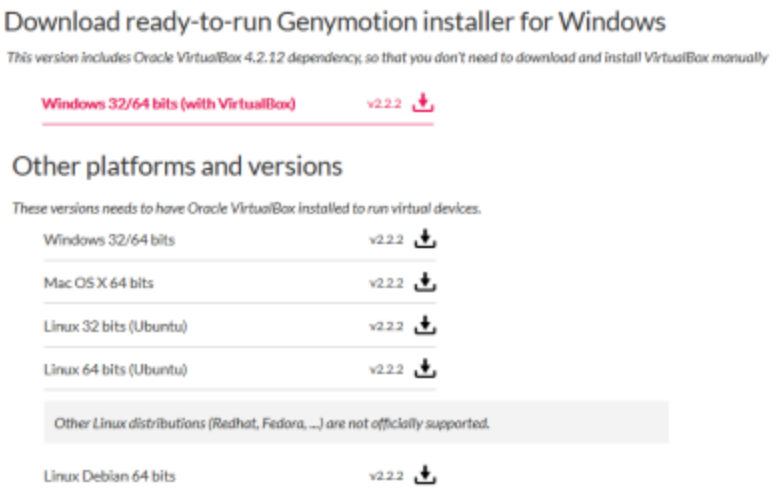



Рис.2.14. Скачивание Genymotion

Запускаем установщик и следуем подсказкам «Мастера установки». По-умолчанию директорией установки является: «C:\Program Files\Genymobile\Genymotion». Вне зависимости от выставленного пути, его необходимо запомнить.

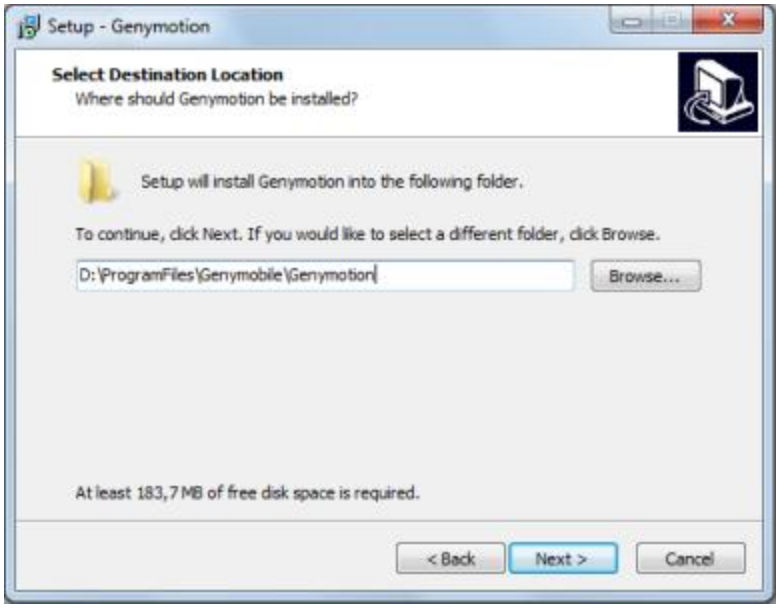



Рис.2.14. Выбирание пути установки

Запускаем Android Studio:


Рис. 2.15. Запуск Андроид

Выберем «File->Settings»:

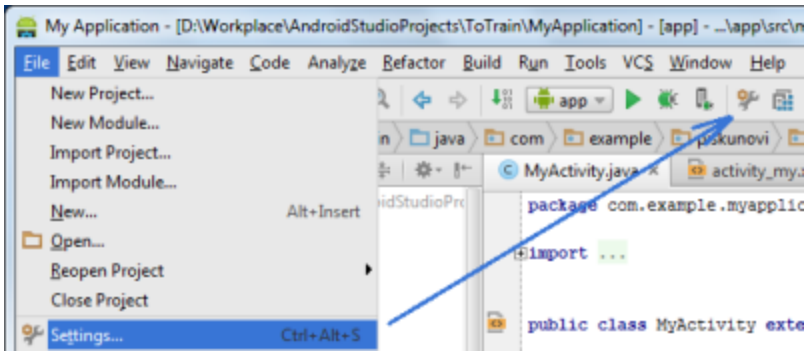



Рис. 2.16. Основное меню Андроид

В настройках IDE («IDE Settings») выбираем раздел Plugins, в котором нажмем кнопку «Browse repositories…».

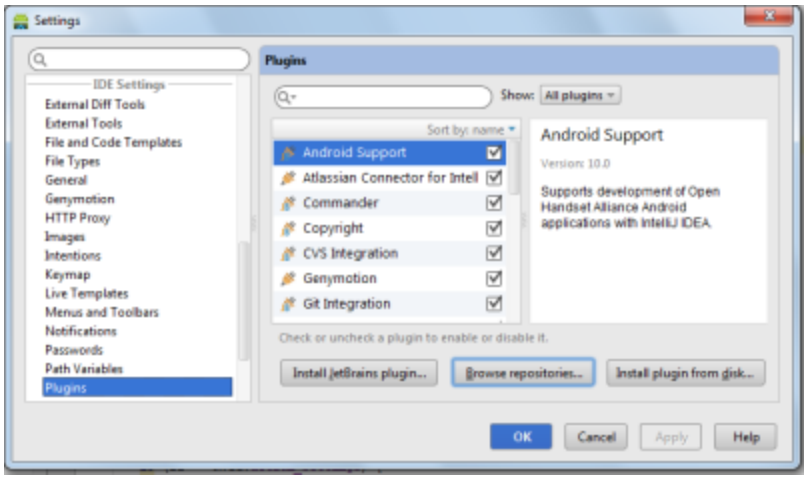



Рис. 2.17. Создание нового проекта

В поисковой строке напишем «Genymotion», а в отобразившемся справа окне нажмем кнопку «Install plugin».

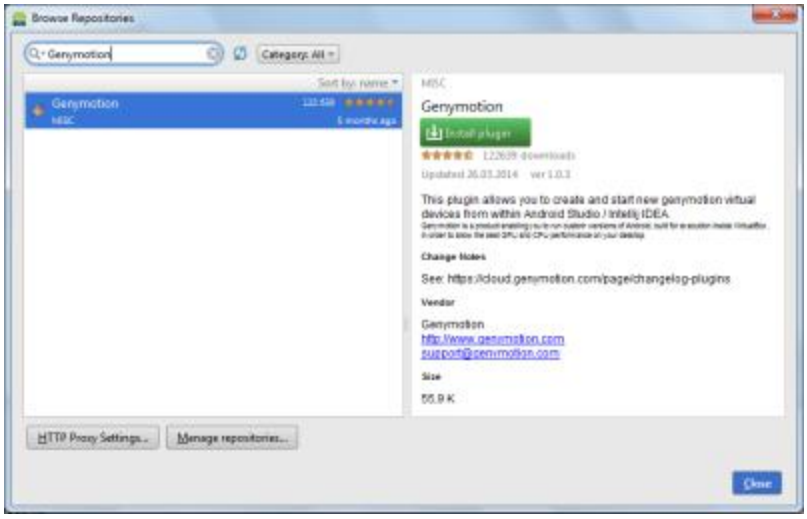



Рис. 2.18. Установка плагинов

После этого среда разработки предупредит о необходимости перезапуска. Нажимаем «Close» после чего Android Studio перезапустится.

На этом установка Android Studio завершена!

Примечание: Отметим, что процесс «studio.exe» съедает в среднем 400Mb памяти, что может привести к не комфортным условиям работы на устройствах с менее чем 3Gb RAM.

**Процесс разработки приложений в среде Android Studio. Язык программирование JAVA.**

Ниже приводится создание и запуск приложение.

Новый проект

Android Virtual Device (AVD)

Запуск приложения.

Шаг 1. Новый проект.

Запустить Android Studio и выбрать «New Project».

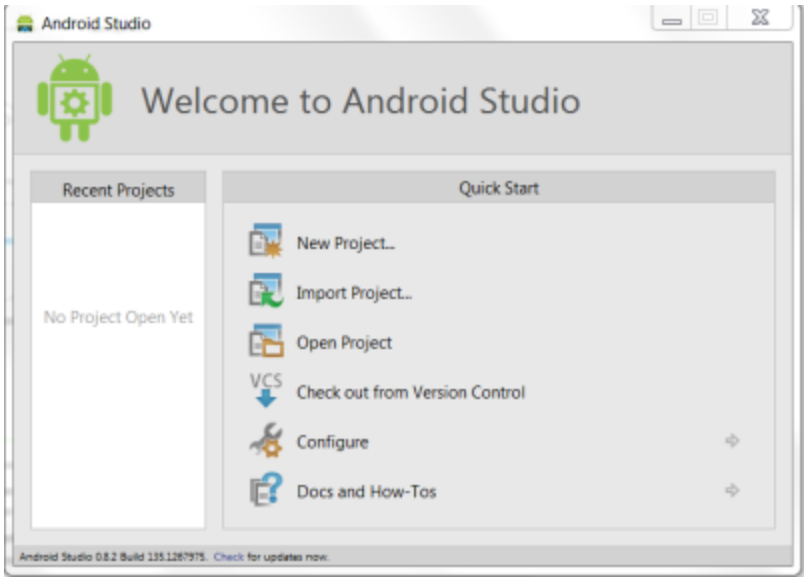



Рис. 2.19. Создание проекта

В новом окне отобразится форма настройки нашего проекта. Здесь мы можем указать имя приложения, домен компании, имя пакета и путь до проекта.

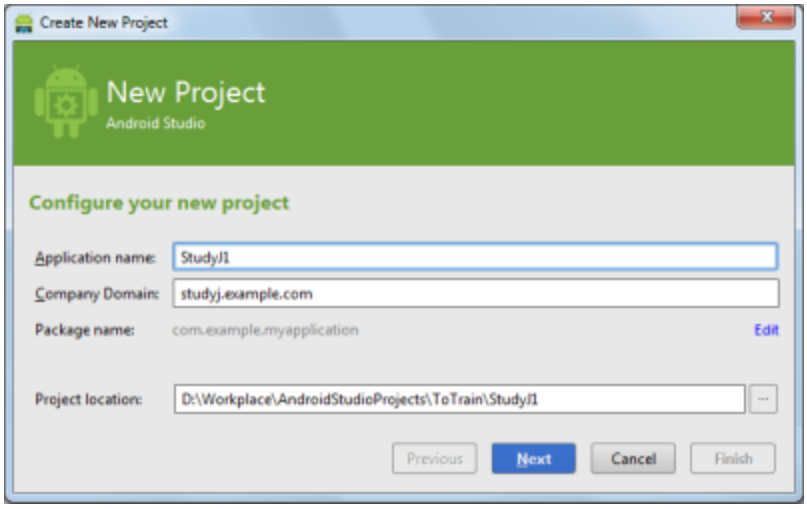



Рис. 2.20. Настройка нового проекта

Обычно компании используют свои зарегистрированные доменные имена в качестве основания для имен пакетов — например, com.example.myapplication для пакетамyapplication, созданный программистом из studyj.example.com.

Нажимаем «Next».

Теперь нам требуется выбрать форм-факторы устройств, на которых будет работать наше приложение. В качестве минимума SDK рекомендуется

выбрать API 10, Android 2.3.3 (Gingerbread). Это обусловлено тем, что приложение будет поддерживаться большинством существующих Android устройств (хоть и в ущерб некоторой функциональности).

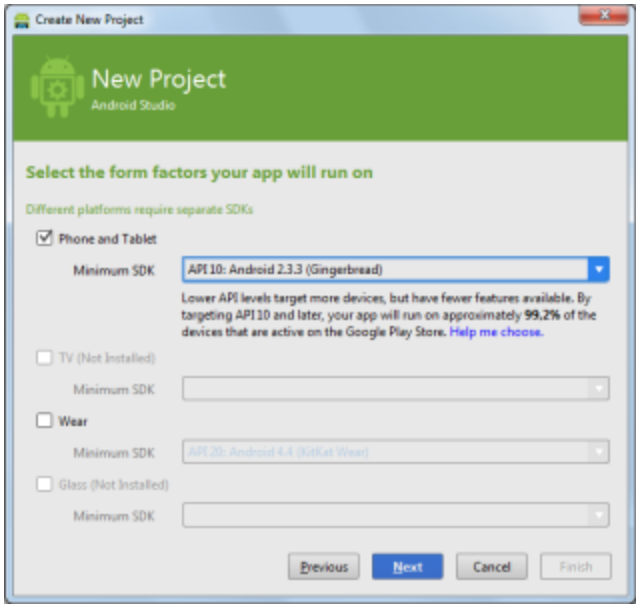



Рис. 2.21. Настройка устройство для компиляции проекта

После того как мы нажмем «Next», нам предложат выбрать activity (активность) нашего проекта из предложенных вариантов. Выбираем «Blank Activity» и снова жмем «Next».


Рис. 2.21. Выбирание тип проекта

Настройки активности оставляем по-умолчанию.

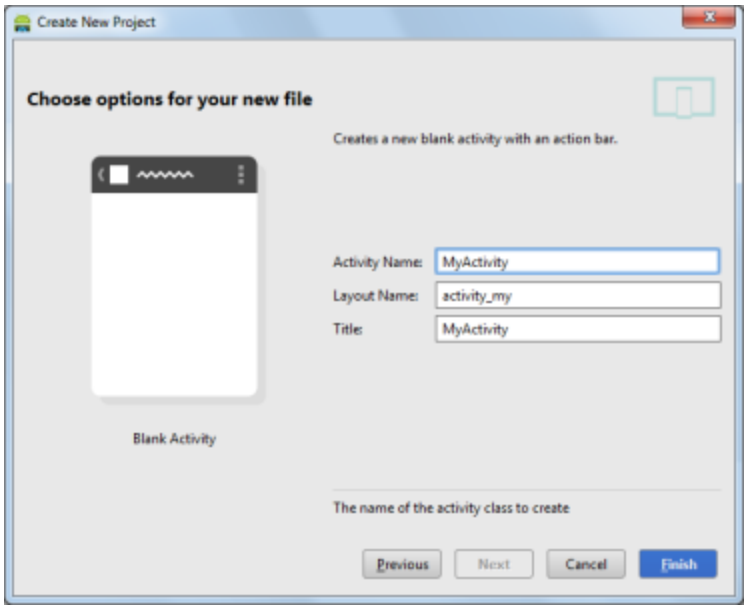



Рис. 2.22. Последний шаг создание проекта

Нажимаем кнопку «Finish», чтобы закончить создание проекта.

Примечание: Если у вас возникает ошибка SDK, проверьте build.gradle в папке вашего приложения. Убедитесь, что compileSdkVersion, minSdkVersion, и targetSdkVersionсовпадают со значениями 19, 10, 19 соответственно.

Шаг 2. Android Virtual Device (AVD).

Запустите Android Studio и нажмите на иконку AVD Manager в панели инструментов (рис. 2.6). Вы можете открыть его точно также перейдя во вкладку Tools->Android->AVD Manager.

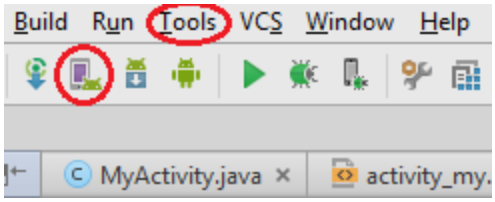



Рис. 2.22. Настройка устройство

После запуска AVD Manager создайте новый AVD и перенесите в него настройки.

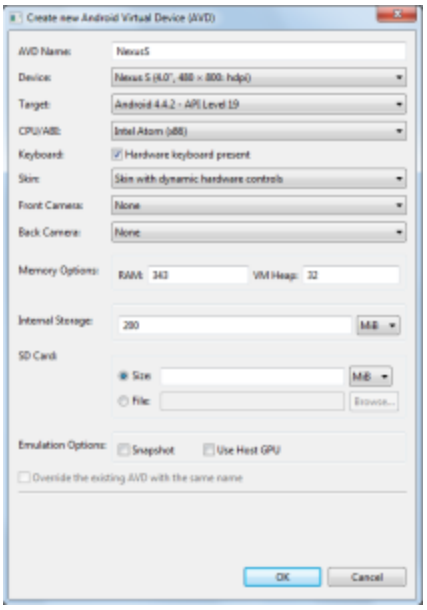



Рис. 2.23. Настройка устройство

В результате будет создан виртуальное устройство, характеристики которого отобразятся в новом окне. Нажимаем кнопку «Ок», наш AVD создан и теперь отображается в панели AVD Manager.

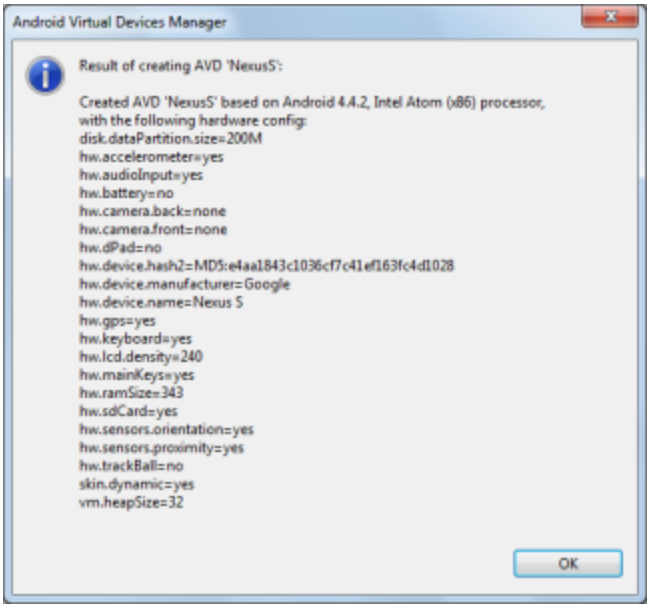



Рис. 2.24. Настройка устройство

Шаг 3. Запуск приложения.

Для того, чтобы убедиться, что все шаги, сделанные раннее, были выполнены верно давайте запустим приложение кликнув по зеленой иконке «Run» в панели инструментов или воспользуемся вкладкой меню «Run».

Выберите «NexusS» в списке запускаемых эмуляторов, когда вас об этом попросят. Наберитесь терпения, первый запуск эмулятора может занять до 20 минут. Именно из-за этого в прошлом уроке мы установили Genymotion.

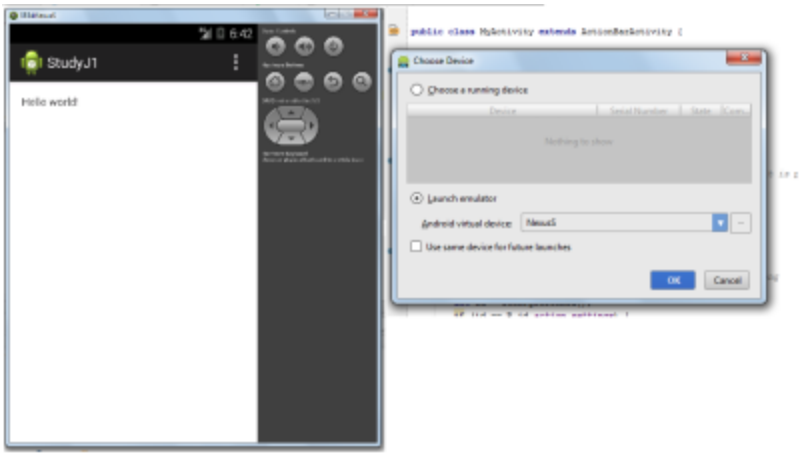


Рис. 2.25. Запуск эмулятора

Для того, чтобы воспользоваться эмулятором Genymotion нам необходимо нажать на иконку «Genymotion Device Manager» на панели инструментов.

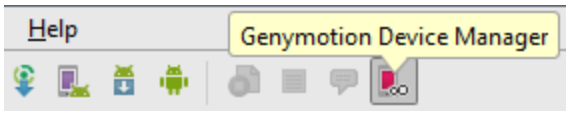
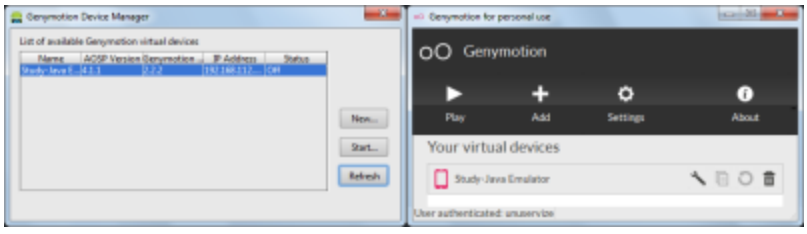



Рис. 2.26. Менежер устройство

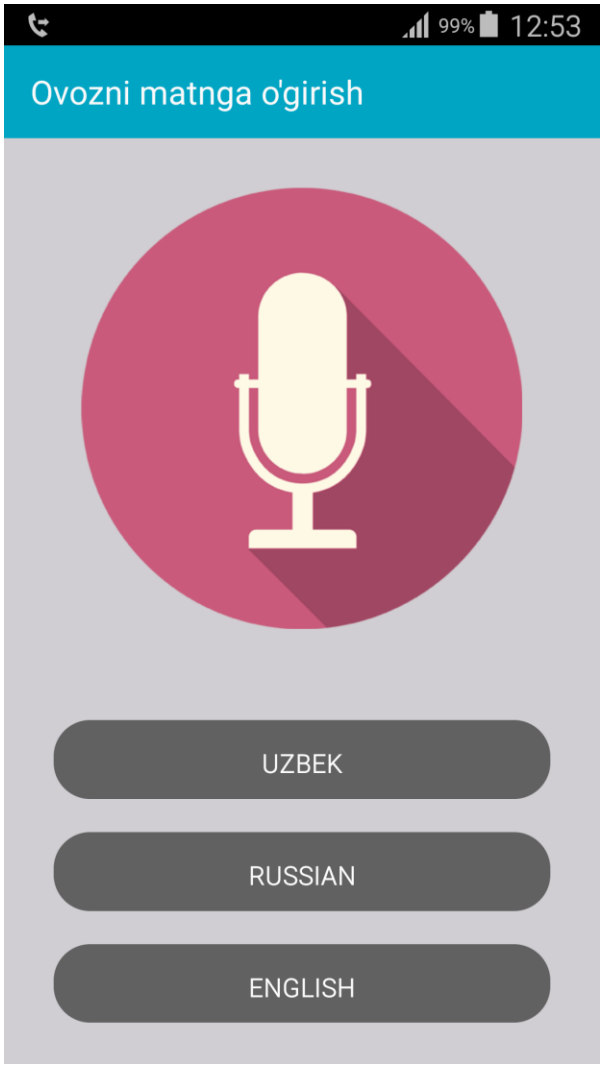
Нас попросят указать папку Genymotion, в которую он был установлен. После того, как папка установки указана, вас попросят авторизоваться при помощи логина и пароля, которыми вы пользовались на сайте.

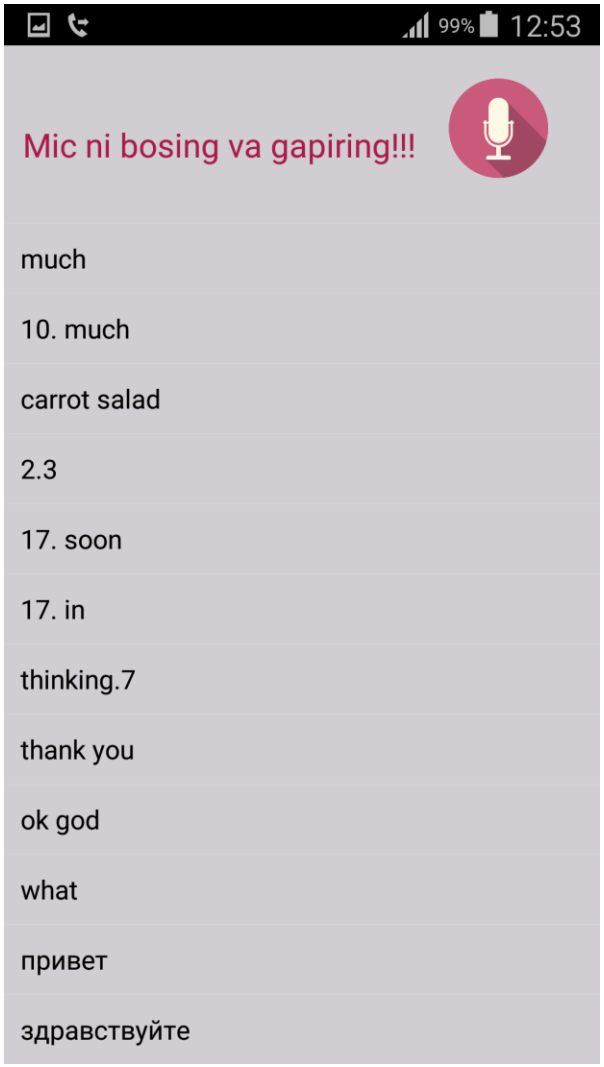
Следующие шаги напоминают создание и запуск AVD, за одним исключением: устройство Genymotion должно быть запущено до запуска приложения, в ином случае оно не будет предложено для выбора в списке доступных устройств.

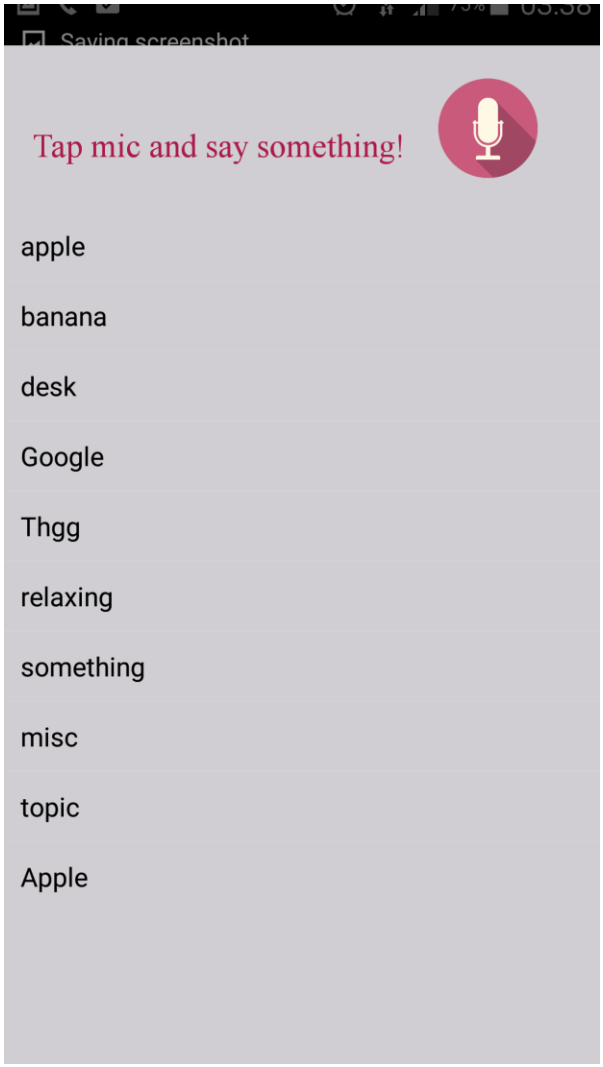


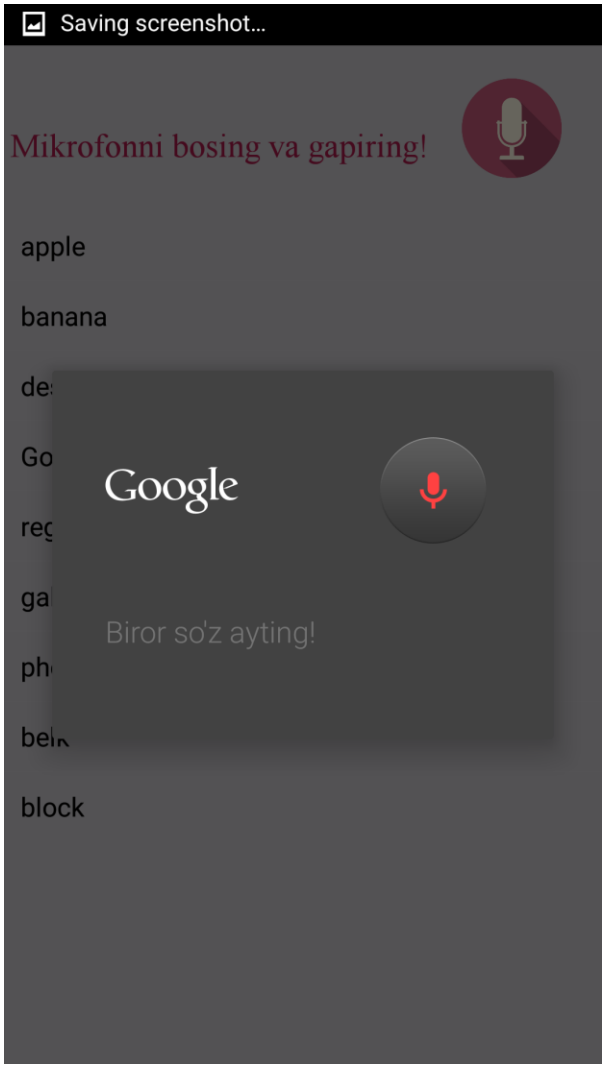
3. Разработка системы «Speech to Text» для интеллектуальной формирование баз данных составляющей фонем узбекского языка

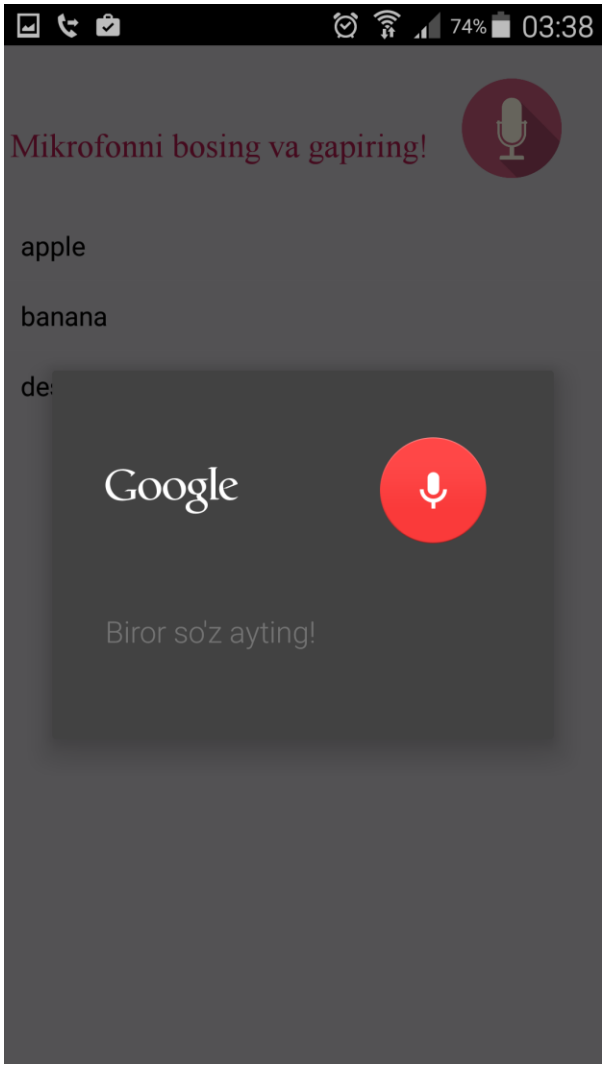
Ниже приводят все виды разработанной системы для распознавание и составление баз данных фонем узбекского языка:

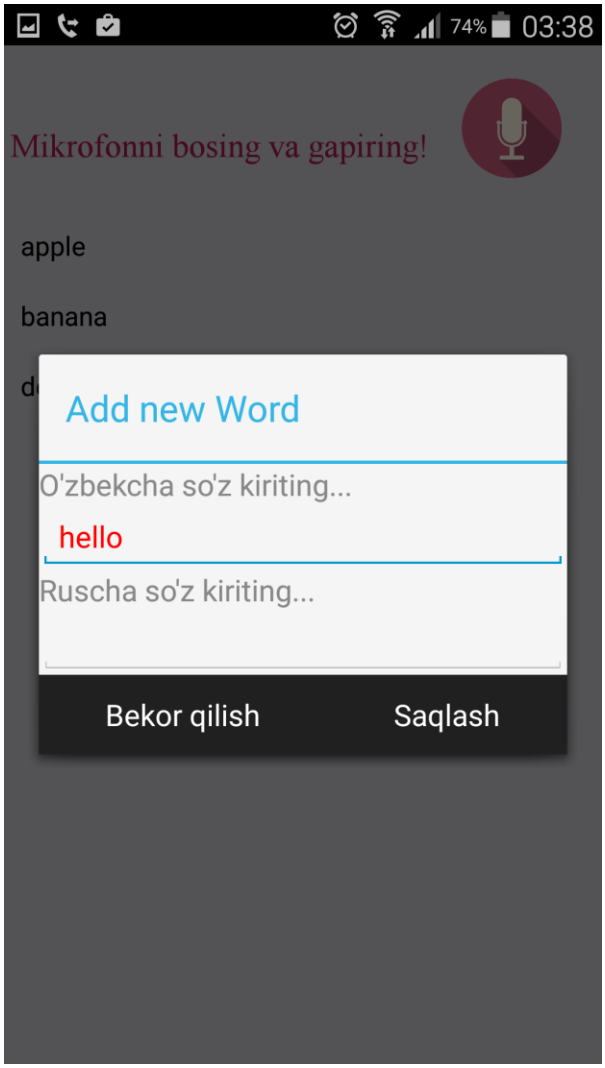


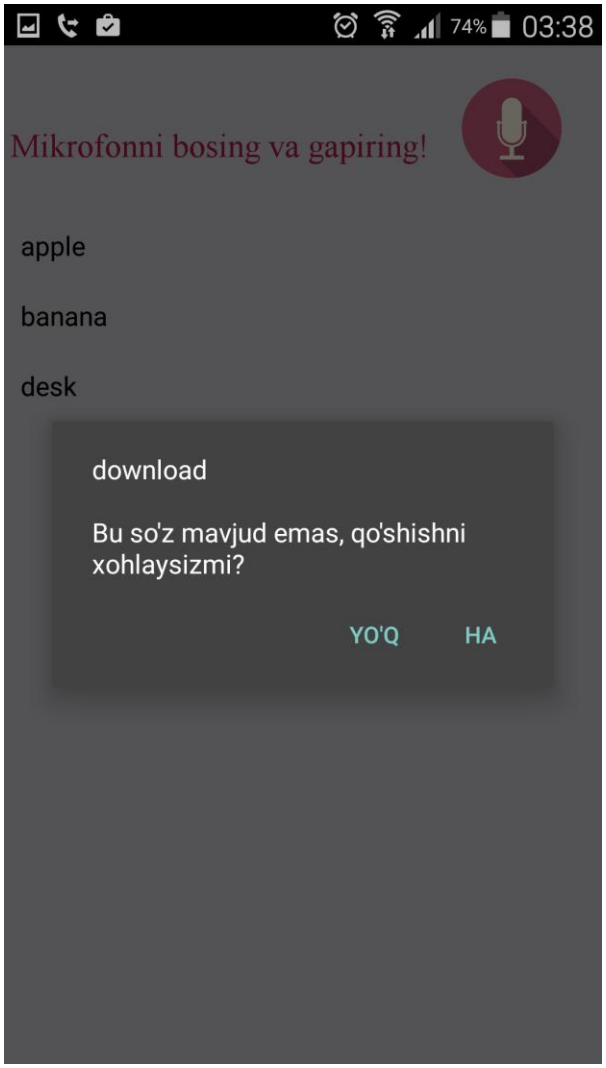


99%
12:53
Mic ni bosing va gapiring!!!
much
10. much
carrot salad
2.3
17. soon
17. in
thinking.7
thank you
ok god
what
привет
здравствуйте


Saving screenshot
Tap mic and say something!
apple
banana
desk
Google
Thgg
relaxing
something
misc
topic
Apple


Saving screenshot...
Mikrofonni bosing va gapiring!
apple
banana
Google
Biror so'z ayting!
block


74%
03:38
Mikrofonni bosing va gapiring!
apple
banana
Google
Biror so'z ayting!




## Выводы по третьей главе

В этой главе приведён следующие результаты работ:

- Методология применение алгоритма Google Speech Voice API в целях программной реализации;
- Исследование и обоснование среды разработок под Android OS;
- Разработка системы «Speech to Text» для интеллектуальной формирование баз данных составляющей фонем узбекского языка.

## Заключение

В результате проделанной работе были получено были рассмотрены теоретические основы разработки распознающий системы для составление база данных фонем узбекского языка с применением методов интеллектуального анализа данных.

Изучены методы и алгоритмы распознавание голосовых информаций.

Было изучено технология применение сервиса Google Speech Voice API в разработке программных средств для составление баз данных. Был изучен технология разработки ПО на базе операционных систем Андроид.

Проведён сравнительный анализ систем распознавания голоса.

Исследовано разработать алгоритм распознавание речи на основе Mel-frequency cepstral coefficients – MFCC, а также, разработана программа для системы преобразования речи в текст для автоматической добавление в базу данных на базе операционных систем Android;

## Список использованной литературы


1. Определение типов лицензий на Boston Software Foundation: www.fsf.org/news/agplv3-pr
2. Ася Власова Как украсть Linux? (рус.) (24.06.2008) — о FOSS-лицензиях и их применении в России:www.osp.ru/cio/2008/06/4987902/
3. Kai Fu Li, Hsiao-Wuen Hon. An overview of the Sphinx Speech Recognition
4. Rybach, D.; C. Gollan, G. Heigold, B. Hoffmeister, J. Lööf, R. Schlüter, H. Ney (September 2009). «The RWTH Aachen University Open Source Speech Recognition System». Interspeech-2009: 2111–2114.
5. Peter Grasch: simon: Open Sourcing Speech Recognition with KDE technology:www.desktopsummit.org/program/sessions/simon-open-sourcing-speech-recognition-kde-technology
6. Interactive Analysis, Transcription and Translation of Old Text Documents:prhlt.iti.upv.es/page/projects/multimodal/idoc/iatros
7. SHoUT speech recognition toolkit: www.digibic.eu/techprofile.asp?slevel=0z84z101&parent_id=101&renleewtsapf=
8. Frequently Asked Questions (and Answers) about Copyright: www.chillingeffects.org/copyright/faq.cgi#QID805
9. Stoughton, Nick (April 2005). «Update on Standards» (PDF). USENIX. Retrieved 2009-06-04.
10. Kai Fu Li, Speech Input API Specification. Editor's Draft 18 October 2010 Latest Editor's Draft: dev.w3.org/… Editors: Satish Sampath, Google Inc. Bjorn Bringert, Google Inc.
11. Голосовой поиск в Google Chrome: habrahabr.ru/post/111201/
12. Официальная страница Dragon Mobile SDK: dragonmobile.nuancemobiledeveloper.com/public/index.php
13. Герберт Шилдт. Java 8. Полное руководство, 9-е издание = Java 8. The Complete Reference, 9th Edition. — М.: «Вильямс», 2015.

14. Кей С. Хорстманн. Java SE 8. Вводный курс = Java SE 8 for the Really Impatient. — М.: «Вильямс», 2014.

15. Фрэд Лонг, Дхрув Мохиндра, Роберт С. Сикорд, Дин Ф. Сазерленд, Дэвид Свобода. Руководство для программиста на Java: 75 рекомендаций по написанию надежных и защищенных программ = Java Coding Guidelines: 75 Recommendations for Reliable and Secure Programs. — М.: «Вильямс», 2014.

16. Кей С. Хорстманн, Гари Корнелл. Java. Библиотека профессионала, том 1. Основы. 9-е издание = Core Java, Volume I: Fundamentals (9th Edition). — М.: «Вильямс», 2013.

17. Барри Берд. Java 8 для чайников = Java For Dummies, 6th edition. — М.: «Диалектика», 2015.

18. Джеймс Гослинг, Билл Джой, Гай Стил, Гилад Брача, Алекс Бакли. Язык программирования Java SE 8. Подробное описание, 5-е издание = The Java Language Specification, Java SE 8 Edition (5th Edition) (Java Series). — М.: «Вильямс», 2015.

19. Джошуа Блох. Java. Эффективное программирование = Effective Java. — М.: Лори, 2002.

20. Монахов Вадим. Язык программирования Java и среда NetBeans. — 3-е изд. — СПб.: БХВ-Петербург, 2011.

21. Брюс Эккель. Философия Java = Thinking in Java. — 3-е изд. — СПб.: Питер, 2003.

22. Голощапов А. Google Android: программирование для мобильных устройств. — СПб.: БХВ-Петербург, 2010.

23. Коматинэни С., Маклин Д., Хэшими С. Google Android: программирование для мобильных устройств = Pro Android 2. — 1-е изд. — СПб.: Питер, 2011.

24. Сатия Коматинени, Дэйв Маклин. Android 4 для профессионалов. Создание приложений для планшетных компьютеров и смартфонов = Pro Android 4. — М.:Вильямс.

25. Роджерс Р., Ломбардо Д. Android. Разработка приложений. — М.: ЭКОМ Паблишерз, 2010.

26. Донн Фелкер. Android: разработка приложений для чайников = Android Application Development For Dummies. — М.: Диалектика, 2011.

# Приложение