УЗБЕКСКОЕ АГЕНТСТВО СВЯЗИ И ИНФОРМАТИЗАЦИИ
ТАШКЕНТКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

К защите.
Зав.кафедрой
_____
_____2010 г

Выпускная квалификационная работа бакалавра

на тему «Создание системы распознавания узбекской речи»

Выпускник: Газиев И.А
Руководитель: Рахматов Ф

Ташкент 2010

УЗБЕКСКОЕ АГЕНТСТВО СВЯЗИ И ИНФОРМАТИЗАЦИИ
ТАШКЕНТСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

Факультет: Информационные технологии

Кафедра: Компьютерные системы

Направление (специальность): 5811300-“Сервис” (информационный сервис)

У Т В Е Р Ж Д А Ю

Зав. кафедрой_____
«_____» _____2010 г.

ЗАДАНИЕ

на выпускную квалификационную работу
Газиев Ислам Авазбекович
(фамилия, имя, отчество)

1. Тема работы: Создание системы распознавания узбекской речи.
2. Утверждена приказом по университету от «___»_____ 2010 г. №____
3. Срок сдачи законченной работы: 31.05.10 г.
4. Исходные данные к работе: Текст на узбекском языке, аудиозапись узбекской речи, дистрибутив Sphinx-4.
5. Содержание расчётно – пояснительной записи (перечень подлежащих разработке вопросов): Введение. Обзор методов компьютерного распознавания речи. Типичная структура системы распознавания речи. Методы обработки речевых сигналов. Основные параметры и характеристики речевого сигнала. Метод выделения признаков речевых сигналов. Алгоритмы распознавания речи. Существующие программы и их сравнительный анализ. Изучение структуры системы Sphinx-4. Обучающая система SphinxTrain. Создание языковой модели. Разработка программы распознавания узбекской речи. Связь библиотеки с приложением. Описание программы. Руководство пользователя. Основные системные требования. Заключение. Литература. Приложение.
6. Перечень графического материала: Структура распознавания речи. График формант с частотами. Устройства выделения признаков речевых сигналов. Нейронные сети. Структура декодера Sphinx-4. Блок-схема работы SphinxTrain. Пользовательский интерфейс. График звукового сигнала.
7. Дата выдачи задания: 15.01.10 г.

Руководитель _____ Задание принял _____
(подпись) (подпись)

8. Консультант по отдельным разделам выпускной работы

Раздел	Ф.И.О Руководителя	Подпись дата	
		Задание выдал	Задание получил
Основная часть	Рахматов Ф.А.		
Безопасность жизнедеятельности	Абдуллаева С.М.		

9. График выполнения работы

№	Наименование раздела работы	Срок выполнения	Отметка руководителя выполнении
1.	Обзор методов компьютерного распознавания речи	15.01.10-26.02.10	
2.	Система распознавания речи SPHINX	27.02.10-01.04.10	
3.	Разработка системы распознавания узбекской речи	05.04.10-10.05.10	
4.	Безопасность жизнедеятельности	11.05.10-15.05.10	
5.	Заключение	17.05.10-31.05.10	

Выпускник _____ «____» _____2010 г.
(подпись)

Руководитель_____ «____» _____2010 г.
(подпись)

Данная выпускная квалификационная работа посвящена разработке системы распознавания узбекской речи. Программа, обработав речевой сигнал, выдает результаты в текстовом виде. Были созданы словарь, акустическая и лингвистическая модели узбекского языка. Работа выполнена с помощью системы CMU Sphinx-4 под ОС Linux, а библиотека была собрана в среде NetBeans на языке Java, что позволяет системе являться кросс-платформой. Также имеется демонстрационный пакет приложений, который был написан на языке Visual C#.

Мазкур битирув малакавий иши ўзбек тили нутқини идрок этиш тизимининг лойихалаштирилишига бағишланган. Дастур нутқ сигналини қайта ишлагандан сўнг натижани матн кўринишида қайтаради. Ишни бажараш жараёнида ўзбек тили луғати, акустик ва лингвистик моделлари яратилди. Иш Linux ОТ остида CMU Sphinx-4 тизими ёрадмади бажарилган, пакет эса NetBeans муҳитида Java дастурлаш тилида йиғилган. Бу эса ўз навбатида тизимга кросс-платформа бўлишига имкон беради. Шунингдек, Visual C# дастурлаш тилида ёзилган дастурлар пакети мавжуд.

This graduation thesis deals with developing an Uzbek speech recognition system. After the program handles speech signal, it returns result in text form. During realization of the work, vocabulary, acoustic and linguistic models of Uzbek language were created. This work was developed under Linux OS with the help of CMU Sphinx-4 system, and a library was built in the NetBeans environment in Java language, that allows the system to be a cross-platform. Also there is a demo package, which includes programs written in Visual C# language.

Содержание

ВВЕДЕНИЕ……... 7

1. ОБЗОР МЕТОДОВ КОМПЬЮТЕРНОГО РАСПОЗНАВАНИЯ РЕЧИ.. 10

1.1. Типичная структура системы распознавания речи…….. 10

1.2. Методы обработки речевых сигналов……... 13

1.3. Основные параметры и характеристики речевого сигнала……. 19

1.4. Метод выделения признаков речевых сигналов…….. 24

1.5. Алгоритмы распознавания речи…… 32

1.4.1. Нейронные сети…….. 32

1.4.2. Алгоритмы обратного распространения…….. 35

1.4.3. Скрытая Марковская модель…… 37

1.6. Существующие программные продукты распознавания речи и их сравнительный анализ…… 38

1.6.1. Бесплатное программное обеспечение…… 38

1.6.2. Коммерческое программное обеспечение……... 40

1.6.3. Сравнительная таблица……. 43

2. СИСТЕМА РАСПОЗНАВАНИЯ РЕЧИ SPHINX……. 44

2.1. Общие сведения…….. 44

2.2. Изучение структуры системы Sphinx-4…… 46

2.3. Обучающая система акустической модели SphinxTrain……. 50

2.4. Этапы построения языковой модели узбекской языка. Создание лингвистической модели. Создание словаря……... 52

2.5. Создание акустической модели……. 56

3. РАЗРАБОТКА ПРОГРАММЫ РАСПОЗНАВАНИЯ УЗБЕКСКОЙ РЕЧИ UzLang…… 64

3.1. Связь библиотеки с приложением……. 64

3.2. Описание программы…… 65

3.3. Руководство пользователя……. 67

3.4. Основные системные требования……. 69

4. БЕЗОПАСНОСТЬ ЖИЗНЕДЕЯТЕЛЬНОСТИ……. 70
4.1. Влияние метеорологических условий производственной среды на организм человека…… 70
4.2. Чрезвычайные ситуации. Защита предприятия в чрезвычайных ситуациях и ликвидация последствий…….. 73
ЗАКЛЮЧЕНИЕ…… 79
СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ……..... 80
ПРИЛОЖЕНИЕ…… 81

Введение

Информационные технологии, это наиболее быстроразвивающаяся отрасль XXI века, не зря называют XXI век, веком информационных технологий. С каждым годом рынок информационно-коммуникационных технологий все совершенствуется, обновляется и пополняется новыми разработками и новинками. То, что до недавнего времени считалось нереальным, сейчас используется практически везде, информационно-коммуникационные технологии (ИКТ) внедряются во все сферы общественной жизни и эффективно применяются в жизни общества, тем самым заметно упрощают рабочие, учебные, развлекательные процессы.

Рынок информационных технологий Узбекистана развивается быстрыми темпами, все совершенствуется и внедряется во все сферы экономики и жизни общества.

Развитие информационно-коммуникационных технологий, является важнейшим фактором поднятия благосостояния и экономического роста, становится одним из основных приоритетов государственной политики Узбекистана.

Именно поэтому Президент Республики Узбекистан И.А. Каримов в своей книге «Мировой финансово-экономический кризис, пути и меры по его преодолению в условиях Узбекистана» особо отметил динамичное развитие услуг в сфере информационно-коммуникационных технологий, которые за последние четыре года в среднем увеличиваются ежегодно на 50 процентов. В результате доля сферы услуг в ВВП возросла в 2008 году до 45,3 процента против 42,5 процента в 2007 году [1].

Также общее видение развития ИКТ в Узбекистане отражено в выступлении Президента Республики Узбекистан на сессии Парламента страны в мае 2001 года. В заявлении Президент призвал правительство

разработать общую стратегию развития ИКТ в поддержку социального, культурного и экономического будущего страны.

По мере развития коммуникационных, информационный и компьютерных систем становится все более очевидным, что использование этих систем намного расширится, если станет возможным использование человеческой речи при работе непосредственно с компьютером, и в частности станет возможным управление машиной обычным голосом в реальном времени, а также ввод и вывод информации в виде обычной человеческой речи.

В настоящее время научное сообщество вкладывает гигантское количество денег в развитие ноу-хау и научно-исследовательские разработки для решения проблем автоматического распознавания и понимания речи. Это стимулируется практическими требованиями, связанными с созданием системы военного и коммерческого назначения.

Несомненно наше правительство сейчас четко осознает важность ИКТ для достижения своих целей развития. Но все же работы над распознаванием речи в Узбекистане пока не приобретают должной актуальности. Распознаванием речи в нашей стране, в частности узбекской, занимаются единицы. А их результаты не находят практического применения, так как эти системы оказываются достаточно дорогостоящими.

Можно выделить несколько проблем рождающих данную тенденцию.

Одной из них является недостаточная осведомленность руководителей коммерческих и государственных организаций о возможностях распознавания речи. Не все представляют, какие могут открыться возможности при достижении цели. Другие же, в силу своей неграмотности в базовой области компьютерных систем, могут поставить перед разработчиками невыполнимые задачи, не имея понятия, какие ресурсы требуются для создания даже небольшой системы распознавания речи, полагая, что компьютерная техника – это полностью интеллектуальная

система. Всё это приводит к снижению интереса обоих сторон – заказчиков и разработчиков. Следовательно, речевые технологии перестают развиваться как коммерческое предложение.

Самой же больной и актуальной проблемой является нехватка кадров, что способствует сильному торможению развития сферы распознавания речи. Как известно, разработка такой системы требует анализа и применения сложнейших алгоритмов. Не так много людей, которые могут в достаточной мере изучить такие понятия как преобразование Фурье, скрытая Марковская модель, векторное квантование, алгоритмы обратного распространения, Витерби, Баума-Велча или EM-процедура и прочее. Еще меньше людей, способные обучать этому с точки зрения программирования.

Целью данной выпускной квалификационной работы является поиск и применение альтернативных способов разработки системы распознавания узбекской речи, которые могли бы в некоторой степени решить вышеперечисленные проблемы.

1. Обзор методов компьютерного распознавания речи

1.1. Типичная структура системы распознавания речи

Распознавание речи – это многоуровневая задача распознавания образов, в которой акустические сигналы анализируются и структурируются в иерархию структурных элементов (например, фонем), слов, фраз и предложений. Каждый уровень иерархии может предусматривать некоторые временные константы, например, возможные последовательности слов или известные виды произношения, которые позволяют уменьшить количество ошибок распознавания на более низком уровне. Чем больше мы знаем (или предполагаем) априорной информации о входном сигнале, тем качественнее мы можем его обработать и распознать.

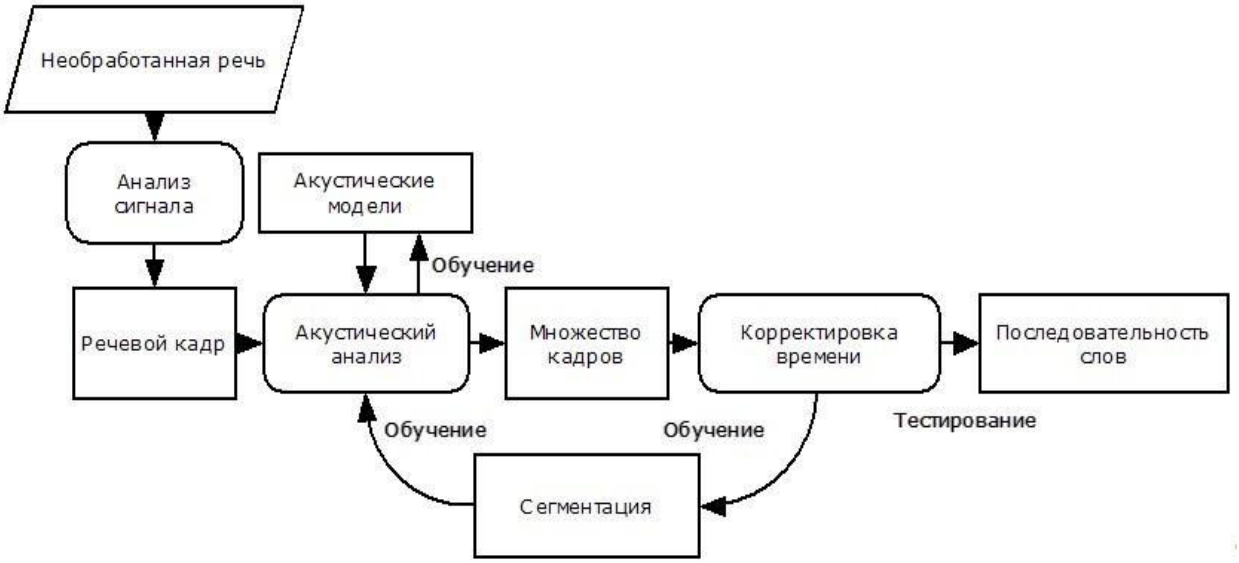

Рис. 1.1.1. Структура стандартной системы распознавания речи показана на рисунке. Рассмотрим основные элементы этой системы.

- **Необработанная речь.** Обычно, поток звуковых данных, записанный с высокой дискретизацией (20 КГц при записи с микрофона либо 8 КГц при записи с телефонной линии).

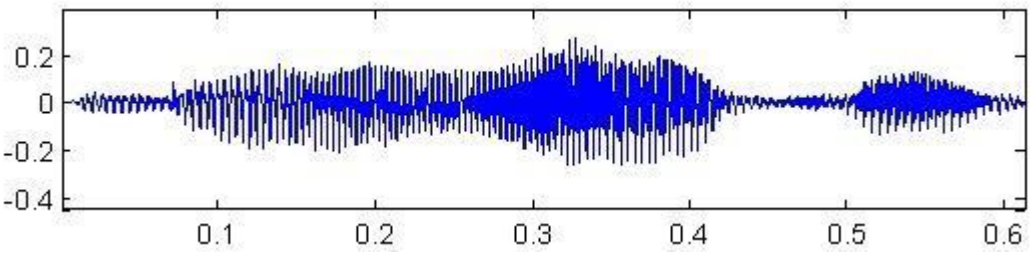

Рис.1.1.2. Поток звуковых данных.

- **Анализ сигнала.** Поступающий сигнал должен быть изначально трансформирован и сжат, для облегчения последующей обработки. Есть различные методы для извлечения полезных параметров и сжатия исходных данных в десятки раз без потери полезной информации. Наиболее используемые методы:
    1. *анализ Фурье;*
    2. *линейное предсказание речи;*
    3. *кепстральный анализ.*
- **Речевые кадры.** Результатом анализа сигнала является последовательность речевых кадров. Обычно, каждый речевой кадр – это результат анализа сигнала на небольшом отрезке времени (порядка 10 мс.), содержащий информацию об этом участке (порядка 20 коэффициентов). Для улучшения качества распознавания, в кадры может быть добавлена информация о первой или второй производной значений их коэффициентов для описания динамики изменения речи.

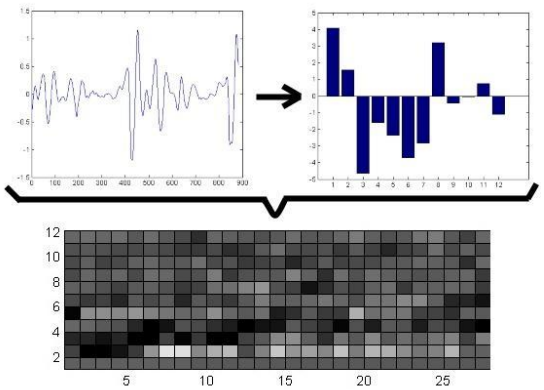

Рис.1.1.3. Речевые кадры.

- **Акустические модели.** Для анализа состава речевых кадров требуется набор акустических моделей. Рассмотрим две наиболее распространенные из них.
    1. *Шаблонная модель.* В качестве акустической модели выступает каким-либо образом сохраненный пример распознаваемой структурной единицы (слова, команды). Вариативность распознавания такой моделью достигается путем сохранения различных вариантов произношения одного и того же элемента (множество дикторов много раз повторяют одну и ту же команду). Используется, в основном, для распознавания слов как единого целого (командные системы).
    2. *Модель состояний.* Каждое слово моделируется как последовательность состояний указывающих набор звуков, которые возможно услышать в данном участке слова, основываясь на вероятностных правилах. Этот подход используется в более масштабных системах.
- **Акустический анализ.** Состоит в сопоставлении различных акустических моделей к каждому кадру речи и выдает матрицу сопоставления последовательности кадров и множества акустических моделей. Для шаблонной модели, эта матрица представляет собой Евклидово расстояние между шаблонным и распознаваемым кадром (т.е. вычисляется, как сильно отлечается полученный сигнал от записанного шаблона и находится шаблон, который больше всего подходит полученному сигналу). Для моделей, основанных на состоянии, матрица состоит из вероятностей того, что данное состояние может сгенерировать данный кадр.
- **Корректировка времени.** Используется для обработки временной вариативности, возникающей при произношении слов (например, “растягивание” или “съедание” звуков).

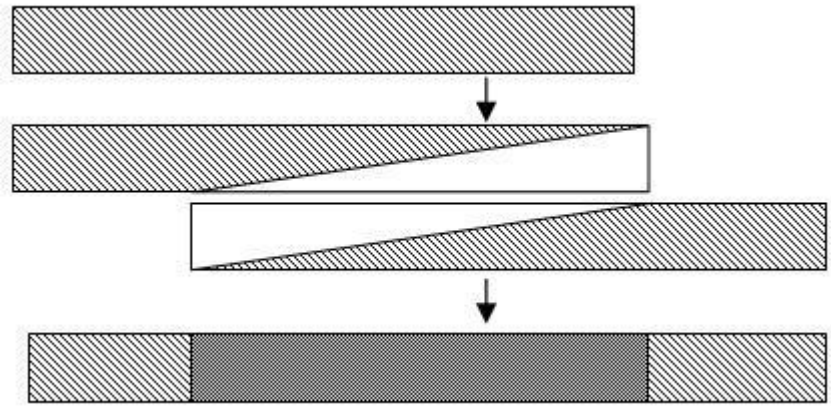

Рис.1.1.4. Корректировка времени.

- **Последовательность слов.** В результате работы, система распознавания речи выдает последовательность (или несколько возможных последовательностей) слов, которая, наиболее вероятно, соответствует входному потоку речи.

1.2. Методы обработки речевых сигналов

Под сигналом будем понимать некую скалярную функцию, зависящую от одного или нескольких аргументов. На практике наиболее часто речь идет о зависимости некоторой величины от времени.

Эта функция может быть задана аналитически (при помощи формулы) или в виде таблицы соответствия ее значений значениям аргумента, идущим подряд на оси времени. Если таковая функция задается непрерывной, то и сигнал называют непрерывным. Если функция дискретна – то и сигнал дискретен. Для дискретных отсчетов обычно временной отрезок делится на равные интервалы, которые затем нумеруются и номера служат аргументами функции-сигнала. Каждую пару “значение аргумента - значение функции” для дискретных сигналов называют отсчетом.

Под системой будем понимать нечто, например процесс, которое получает на входе сигнал и выдает сигнал на выходе.

Большинство систем в природе попадают под понятие линейных. Линейной называется такая система, которая обладает свойствами аддитивности и однородности. Зачастую для линейных систем выставляется также требование обладать свойством инвариантности относительно смещения.

Однородность системы означает то, что если она преобразует входной сигнал x[n] в выходной сигнал y[n], то она должна переводить сигнал kx[n] в сигнал ky[n] для любой постоянной k. То есть усиление или ослабление в k раз входного сигнала должно привести к усилению или ослаблению в такое же число раз выходного сигнала.

Аддитивность системы означает то, что если она преобразует входной сигнал x1[n] в выходной сигнал y1[n], а входной сигнал x2[n] в y2[n], то она должна переводить входной сигнал x1[n]+ x2[n] в выходной сигнал y1[n] +y2[n].

Инвариантность системы относительно смещения означает, что если она преобразует входной сигнал x1[n] в выходной сигнал y1[n], то она должна переводить сигнал x2[n]=x[n+s] в сигнал y2[n]=y[n+s] для любых целых s.

Примерами линейных систем могут служить среды распространения звуковых или электромагнитных волн, электронные схемы, например усилители и фильтры, и многие другие системы.

Декомпозицией называется разложение сигнала на составляющие, сумма которых равна исходному сигналу. Наиболее распространенными видами декомпозиции являются импульсное разложение и разложение Фурье.

Импульсное разложение делит сигнал длиной N отсчетов на N сигналов по N отсчетов. Каждый из результирующих сигналов содержит один отсчет равный значению отсчета с этим номером в исходном сигнале и N-1 отсчетов равных нулю. Для первого сигнала ненулевой отсчет будет иметь номер 1, для второго – номер 2 и так далее. Импульсное разложение

позволяет рассматривать сигнал по одной точке. Зная, как линейная система реагирует на один одиночный импульс мы, пользуясь свойствами однородности, и инвариантности к смещению, можем получить реакцию системы на все остальные импульсы, смещая выходной сигнал на смещение любого импульса относительно импульса-образца и умножая выходной сигнал на отношение амплитуды импульса к амплитуде импульса-образца. Полученные таким образом сигналы мы можем сложить и, поскольку линейная система обладает свойством аддитивности, сделать вывод о том что результат эквивалентен тому, который мы получили бы, если бы пропустили через систему сигнал до того, как подвергли его импульсному разложению. Это метод называется сверткой и будет рассмотрен совсем скоро.

Разложение Фурье делит исходный сигнал на составляющие, половина которых является косинусоидальными, а половина – синусоидальными сигналами. Разложение Фурье является важным, во-первых, потому, что для линейных систем синусоида на входе дает синусоиду на выходе (причем той же частоты – отличаться могут только амплитуда и фаза), а во-вторых, из-за того, что разложение Фурье имеет хорошо разработанный математический аппарат. Ну и кроме того, а может быть, и в первую очередь, потому, что очень многие сигналы в природе имеют синусоидальную форму. Это относится и к интересующим нас звуковым волнам.

Дискретное преобразование Фурье

Необходимо подробно остановиться на разложении Фурье, дискретном преобразовании Фурье и алгоритмах его быстрого вычисления.

В 1807 году французский математик и физик Жан-Батист Жозеф Фурье выдвинул идею о том, что любой непрерывный периодический сигнал может быть представлен как сумма должным образом выбранных синусоидальных волн. И хотя он оказался прав не на все сто процентов, ибо суммированием синусоид нельзя получить сигнал с уголковыми вершинами, все же

подобным разложением можно получить очень хорошо приближенные значения даже для таких сигналов, а уже тем паче для других.

Нас будет интересовать частный случай подобного разложения, а именно дискретное преобразование Фурье (ДПФ или в английском варианте FFT – Fast Fourier Transformation), потому что мы имеем дело не с непрерывным (аналоговым) сигналом, а с дискретным сигналом, представляющим собой набор цифровых отсчетов, следующих с определенной частотой дискретизации.

Дискретное преобразование Фурье (в англоязычной литературе DFT, Discrete Fourier Transform) — это одно из преобразований Фурье, широко применяемых в алгоритмах цифровой обработки сигналов (его модификации применяются в сжатии звука в MP3, сжатии изображений в JPEG и др.), а также в других областях, связанных с анализом частот в дискретном (к примеру, оцифрованном аналоговом) сигнале. Дискретное преобразование Фурье требует в качестве входа дискретную функцию. Такие функции часто создаются путём дискретизации (выборки значений из непрерывных функций). Дискретные преобразования Фурье помогают решать частные дифференциальные уравнения и выполнять такие операции, как свёртки. Дискретные преобразования Фурье также активно используются в статистике, при анализе временных рядов. Преобразования бывают одномерные, двумерные и даже трёхмерные.

Прямое преобразование:

$$X_k = \sum_{n=0}^{N-1} x_n e^{-\frac{2\pi i}{N} kn} \qquad k = 0, \ldots, N-1$$

Обратное преобразование:

$$x_n = \frac{1}{N} \sum_{k=0}^{N-1} X_k e^{\frac{2\pi i}{N} kn} \qquad n = 0, \ldots, N-1.$$

Обозначения:

- N — количество значений сигнала, измеренных за период, а также количество компонентов разложения;

- $x_n, \quad n = 0, \ldots, N-1$, — измеренные значения сигнала (в дискретных временных точках с номерами $n = 0, \ldots, N-1$, которые являются входными данными для прямого преобразования и выходными для обратного;
- $X_k, \quad k = 0, \ldots, N-1$, — N комплексных амплитуд синусоидальных сигналов, слагающих исходный сигнал; являются выходными данными для прямого преобразования и входными для обратного; поскольку амплитуды комплексные, то они обозначают одновременно и амплитуду и фазу;
- $\frac{|X_k|}{N}$ — обычная (вещественная) амплитуда k-го синусоидального сигнала;
- $arg(X_k)$ — фаза k-го синусоидального сигнала (аргумент комплексного числа);
- k — частота k-го сигнала, равная $\frac{k}{T}$, где T — период времени, в течение которого брались входные данные.

Из последнего видно, что преобразование раскладывает сигнал на синусоидальные составляющие (которые называются гармониками) с частотами от N колебаний за период до одного колебания за период. Поскольку частота дискретизации сама по себе равна N отсчётов за период, то высокочастотные составляющие не могут быть корректно отображены — возникает муаров эффект. Это приводит к тому, что первая половина из N комплексных амплитуд, фактически, является зеркальным отображением второй и не несёт значительного смысла.

Рассмотрим некоторый периодический сигнал $x(t)$ с периодом равным T. Разложим его в ряд Фурье:

$$x(t) = \sum_{k=-\infty}^{+\infty} c_k e^{i\omega_k t}, \qquad \omega_k = \frac{2\pi k}{T}$$

Проведем дискретизацию сигнала так, чтобы на периоде было N отсчетов. Дискретный сигнал представим в виде отсчетов: $x_n = x(t_n)$, где $t_n = \frac{n}{N}T$, тогда эти отсчеты через ряд Фурье запишутся следующим образом:

$$x_n = \sum_{k=-\infty}^{+\infty} c_k e^{i\omega_k t_n} = \sum_{k=-\infty}^{+\infty} c_k e^{\frac{2\pi i}{N}kn}$$

Используя соотношение: $e^{\frac{2\pi i}{N}(k+mN)n} = e^{\frac{2\pi i}{N}kn}$, получаем:

$$x_n = \sum_{k=0}^{N-1} X_k e^{\frac{2\pi i}{N}kn}, \quad \text{где} \quad X_k = \sum_{l=-\infty}^{+\infty} c_{k+lN}$$

Таким образом мы получили **обратное дискретное преобразование Фурье**.

Умножим теперь скалярно выражение для x_n на $e^{-\frac{2\pi i}{N}mn}$ и получим:

$$\sum_{n=0}^{N-1} x_n e^{-\frac{2\pi i}{N}mn} = \sum_{k=0}^{N-1}\sum_{n=0}^{N-1} X_k e^{\frac{2\pi i}{N}(k-m)n} = \sum_{k=0}^{N-1} X_k \frac{1-e^{2\pi i(k-m)}}{1-e^{\frac{2\pi i(k-m)}{N}}} = \sum_{k=0}^{N-1} X_k N\delta_{km}$$

Здесь использованы: а) разность 2-х сумм членов (экспонент) геометрических прогрессий с коэффициентами, меньшими единицы, и б) выражение символа Кронекера как предела отношения функций Эйлера для комплексных чисел. Отсюда следует, что:

$$X_k = \frac{1}{N}\sum_{n=0}^{N-1} x_n e^{-\frac{2\pi i}{N}kn}$$

Эта формула описывает **прямое дискретное преобразование Фурье**.

В литературе принято писать множитель $\frac{1}{N}$ в обратном преобразовании, и поэтому обычно пишут формулы преобразования в следующем виде:

$$X_k = \sum_{n=0}^{N-1} x_n e^{-\frac{2\pi i}{N}kn}, \qquad x_n = \frac{1}{N}\sum_{k=0}^{N-1} X_k e^{\frac{2\pi i}{N}kn}$$

Дискретное преобразование Фурье является линейным преобразованием, которое переводит вектор временных отсчётов $\vec{x}$ в вектор спектральных отсчётов той же длины. Таким образом преобразование может быть реализовано как умножение квадратной матрицы на вектор:

$$\vec{X} = \hat{A}\vec{x}$$

матрица А имеет вид:

$$\hat{A} = \begin{pmatrix} 1 & 1 & 1 & 1 & \dots & 1 \\ 1 & e^{-\frac{2\pi i}{N}} & e^{-\frac{4\pi i}{N}} & e^{-\frac{6\pi i}{N}} & \dots & e^{-\frac{2\pi i}{N}(N-1)} \\ 1 & e^{-\frac{4\pi i}{N}} & e^{-\frac{8\pi i}{N}} & e^{-\frac{12\pi i}{N}} & \dots & e^{-\frac{2\pi i}{N}2(N-1)} \\ 1 & e^{-\frac{6\pi i}{N}} & e^{-\frac{12\pi i}{N}} & e^{-\frac{18\pi i}{N}} & \dots & e^{-\frac{2\pi i}{N}3(N-1)} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & e^{-\frac{2\pi i}{N}(N-1)} & e^{-\frac{2\pi i}{N}2(N-1)} & e^{-\frac{2\pi i}{N}3(N-1)} & \dots & e^{-\frac{2\pi i}{N}(N-1)^2} \end{pmatrix}$$

Элементы матрицы задаются следующей формулой:

$$A(m,n) = \exp\left(-2\pi i \frac{(m-1)(n-1)}{N}\right)$$

Свойства

1. линейность

   $ax(n) + by(n) \longleftrightarrow aX(k) + bY(k)$
2. сдвиг по времени

   $x(n-m) \longleftrightarrow X(k)e^{-\frac{2\pi i}{N}km}$
3. периодичность

   $X(k + rN) = X(k), r \in \mathbb{Z}$
4. выполняется Теорема Парсеваля
5. обладает спектральной плотностью

   $S(k) = |x(k)|^2$
6. $x(n) \in \mathbb{R}$

   $X(0) \in \mathbb{R}$

   $N \mod 2 = 0 \Rightarrow X(N/2) \in \mathbb{R}$

   Стоит отметить, что нулевая гармоника является суммой значений сигнала.

1.3. Основные параметры и характеристики речевого сигнала

Имея цифровое представление речевого сигнала, мы можем задуматься о метриках, то есть параметрах этого сигнала, с помощью которых программа может распознавать звуки, слова и предложения приблизительно с тем же

результатом, который дают здоровый слуховой аппарат и здоровый мозг человека. То есть нам нужны параметры, которые позволяют:

1. отделить речь от промежутков “молчания”,
2. определить конкретный звук и сложить из звуков слова,
3. определить начало и конец фразы, предложения и их характер (повествование, вопрос).
4. определить особенности речи диктора.

Параметры речевого сигнала, как правило, быстро меняются в течение времени, поэтому принято снимать их на отрезке звукового сигнала 10-20 мс, считая что сигнал на таком отрезке примерно стационарен (постоянен).

Основные параметры речевого сигнала таковы:

1) Кратковременная энергия речевого сигнала

$$E=(1/N)\sum_{k=1}^{N} S_k^2$$

где N-число отсчетов в кадре (обычная длина кадра 10-20мс), S_k – значение k-го отсчета.

2) Число нулей интенсивности (мгновенная частота).

$$Z=(1/2)\sum_{k=1}^{N} |sign(S_k)-sign(S_{k-1})|$$

где

sign(S)=1, при S>=0

sign(S)=-1, при S<0

Кратковременная энергия звукового сигнала и число нулей интенсивности используются для выделения пауз в речи. Мгновенная частота используется также при классификации шипящих и гласных звуков.

3) Форманты речевого сигнала.

Важнейшим параметром, характеризующим спектр (распределение энергии или амплитуды по частотам) речевого сигнала являются форманты, которые определяют как концентрацию энергии в ограниченной частотной области. Форманта характеризуется частотой, шириной и амплитудой. За частоту форманты принимают частоту максимальной амплитуды в пределах форманты. Другими словами, форманта – это некоторый амплитудный всплеск на графике спектра, а его частота – частота пика этого всплеска.

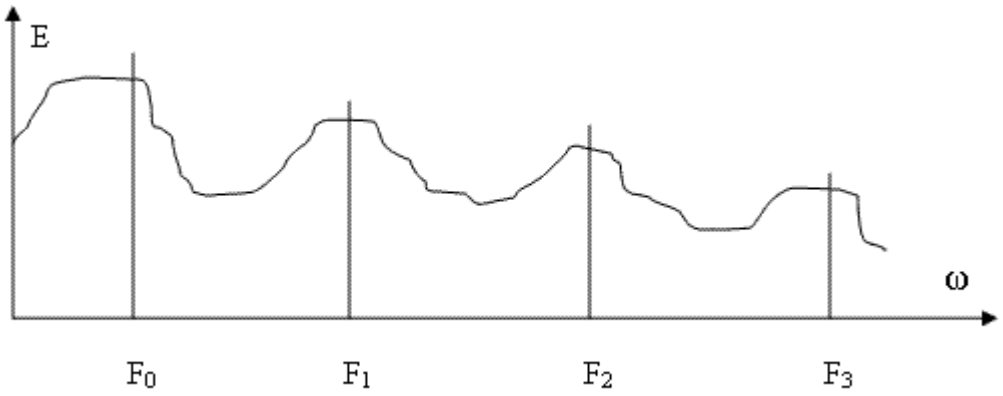

Рис.1.3.1. Амплитудный всплеск.

На рисунке видно четыре форманты с частотами F0, F1, F2, F3. Форманта F0 называется также частотой основного тона речевого сигнала. Форманты F1 – F6 определяют концентрацию энергии речевого сигнала по частоте и характеризуют вокализованные (как правило, гласные) звуки. Форманты начиная с F3 несут информацию о дикторе и широко используются в системах распознавания диктора.

Поговорим об алгоритме выделения формант в речевом сигнале. Как и остальные параметры речевого сигнала, форманты быстро меняются в течение времени, поэтому принято снимать их на отрезке звукового сигнала 10-20 мс, считая что сигнал на таком отрезке примерно стационарен (постоянен).

На первом этапе, естественно, берется преобразование Фурье от исходного сигнала, чтобы получить сигнал спектра.

Известно, что первая форманта (частота основного тона) лежит в промежутке от 50 Гц (у взрослого мужчины) до 300 Гц (у детей и некоторых женщин). Кроме того известно, что амплитуда каждой последующей форманты ниже, чем предыдущей, а их частоты равны частоте основного тона умноженного на целое число (2 для F1, 3 для F2 и так далее).

Таким образом мы можем найти максимальное значение амплитуды сигнала на отрезке от 50 Гц до 300 Гц, после чего умножить эту частоту на 2, найти максимальное значение амплитуды в некоторой окрестности этой точки, проверить является ли она максимумом амплитуды в некоторой большей окрестности. Если это предположение верно, то мы нашли частоту второй форманты (F1). Если нет – второй форманты у сигнала нет.

Далее умножаем частоту основного тона на три и повторяем все вышеприведенные шаги для третьей форманты. Потом повторяем для всех остальных формант. Как правило, вычисляют от 4 до 6 формант.

4) Коэффициенты линейного предсказания.

Суть линейного предсказания в нахождении коэффициентов ak (k=1..p) для формулы:

$$x[n] = \sum_{k=1}^{p} (a_k \, x[n-k]) \qquad (1.3.1)$$

и последующем использовании этой формулы. Другими словами мы должны построить линейный многочлен, позволяющий с хорошей точностью вычислять значение любого отсчета в сигнале по значениям предыдущих p отсчетов. Коэффициенты ak и называются коэффициентами линейного предсказания.

Фактически, имея некоторый сигнал, мы имеем статистическую выборку которую можно представить в виде таблицы:

x[n-p]	**x[n-p+1]**	**x[n-p+2]**	**. . .**	**x[n-1]**	**x[n]**
x[0]	x[1]	x[2]	. . .	x[p-1]	x[p]
x[1]	x[2]	x[3]	. . .	x[p]	x[p+1]
x[2]	x[3]	x[4]	. . .	x[p+1]	x[p+2]
. . .	. . .	. . .	. . .	. . .	. . .
. . .	. . .	. . .	. . .	. . .	. . .
x[N-p-1]	x[N-p]	x[N-p+1]	. . .	x[N-2]	x[N-1]

То есть нахождение коэффициентов линейного предсказания сводится к вычислению коэффициентов линейной регрессии для данной статистической выборки и мы можем пользоваться методами математической статистики.

Минимизируем сумму квадратов ошибок для каждого из вычисляемых отсчетов.

Ошибка для отсчета x[n] равна

$$\delta[n] = x[n] - \sum_{k=1}^{p} (a_k x[n-k]) \qquad (1.3.2)$$

А минимизируемая функция равна

$$E = \sum_{n=0}^{N-1} \delta^2[n] = \sum_{n=0}^{N-1} x[n] - \sum_{k=1}^{p} (a_k x[n-k]))^2 = \sum_{n=0}^{N-1} x^2[n] - 2 \sum_{n=0}^{N-1} x[n] \sum_{k=1}^{p} (a_k x[n-k]))+$$

$$+ \sum_{n=0}^{N-1} (\sum_{k=1}^{p} (a_k x[n-k]))^2 = \sum_{n=0}^{N-1} x^2[n] - 2 \sum_{k=1}^{p} (a_k \sum_{n=0}^{N-1} (x[n] x[n-k])) +$$

$$+ \sum_{j-1}^{p} \sum_{k=1}^{p} a_k a_j \sum_{n=0}^{N-1} (x[n-k]x[n-j])) \qquad (1.3.3)$$

Продифференцируем E по a_k и приравняем частные производные нулю для нахождения экстремума:

$$dE/da_k = \sum_{n=0}^{N-1} (x[n]\, x[n-k])) + \sum_{j=1}^{p} a_j \sum_{n=0}^{N-1} (x[n-k]x[n-j]))=0 \qquad (1.3.4)$$

Заменив для удобства восприятия j на i, a k на j получим систему p линейных уравнений с p неизвестными:

$$\sum_{i=1}^{p} a_i c_{ij}=c_{0j} \qquad (1.3.5)$$

где

$$c_{ij}=c_{ji}= \sum_{n=0}^{N-1} x[n-i]x[n-j]) \qquad (1.3.6)$$

Эта система называется системой уравнений Юла-Уокера. Погрешность найденных коэффициентов оценивается как:

$$E = c_{00}-2\sum_{i=1}^{p} a_i c_{0i}+\sum_{i=1}^{p} a_i \sum_{j=1}^{p} a_j c_{ij} = c_{00} - \sum_{i=1}^{p} a_i c_{0i} \qquad (1.3.7)$$

5) Распределение энергии сигнала по частотным группам.

В результате преобразования Фурье мы получаем комплексные коэффициенты разложения: действительная часть соответствует ReX, мнимая – ImX.

При этом нас будут интересовать только N/2 первых значений полученного массива коэффициентов (остальные будут равны нулю). Как известно, комплексное число можно представить в виде абсолютного значения и фазы (полярная форма):

Абсолютное значение:

$MagX[k]=sqrt(Re^2X[k]+Im^2X[k])$

здесь sqrt - квадратный корень

Фаза:

PhaseX[k]=arctan(ImX[k]/ReX[k]).

Эта форма в нашем случае полезна по той причине, что для человеческого слуха, оказывается, фаза практически не имеет значения. То есть оперируя только абсолютными значениями, мы уменьшим вдвое количество входных параметров распознающей системы, не потеряв при этом значимой информации.

Но даже такое число параметров является слишком большим, что приводит к чрезмерному усложнению процесса обучения распознающей системы, что будет понятно, когда будут рассматриваться алгоритмы распознавания. Например, при частоте дискретизации 16000 мы имеем 8000 частот, то есть 8000 входных параметров.

На помощь приходит то, что человеческий слух имеет свойство образовывать частотные группы. То есть мы можем заменить абсолютные значения амплитуд в частотном домене на некоторую величину, характеризующую суммарную амплитуду частот, попадающих в определенную группу. Число групп может порядка нескольких десятков, при этом группы в высоких частотах должны иметь больший диапазон, чем в низких, так как человеческий слух более точно распознает частоты в низком поддиапазоне звуковых волн и менее точно – по мере увеличения частоты.

В качестве входящих параметров распознавания можно использовать распределение энергии сигнала по этим частотным группам. Значение распределения для i-й частотной группы будет вычисляться по формуле:

$$P[i]=\left(\sum_{k=b[i]}^{N[i]} Mag^2X[k]\right) / \left(\sum_{k=1}^{N/2} Mag^2X[k]\right) \qquad (1.3.8)$$

где b[i] - индекс первой частоты в i-й частотной группе, n[i] - индекс последней частоты в i-й частотной группе, N/2 – общее число частотных групп. Этот показатель хорош еще и тем, что является нормализованным, безразмерным, ведь человеческий слух реагирует не на абсолютные значения амплитуд каких-либо частот, а на их соотношения друг с другом.

1.4. Метод выделения признаков речевых сигналов

Ниже будет предложена следующая структурная схема устройства выделения признаков речевых сигналов (рисунок 1.4.1).

Она состоит из следующих блоков:

1 - микрофон;

2 – блок выделения огибающей;

3 – блок определения начала и конца слова;

4 – блок выделения конечной разности;

5 – блок выделения количества звуков;

6 – линия задержки;

7 – блок выделения интервалов;

8 – блок анализа;

9 – блок данных;

10 – печатающее устройство.

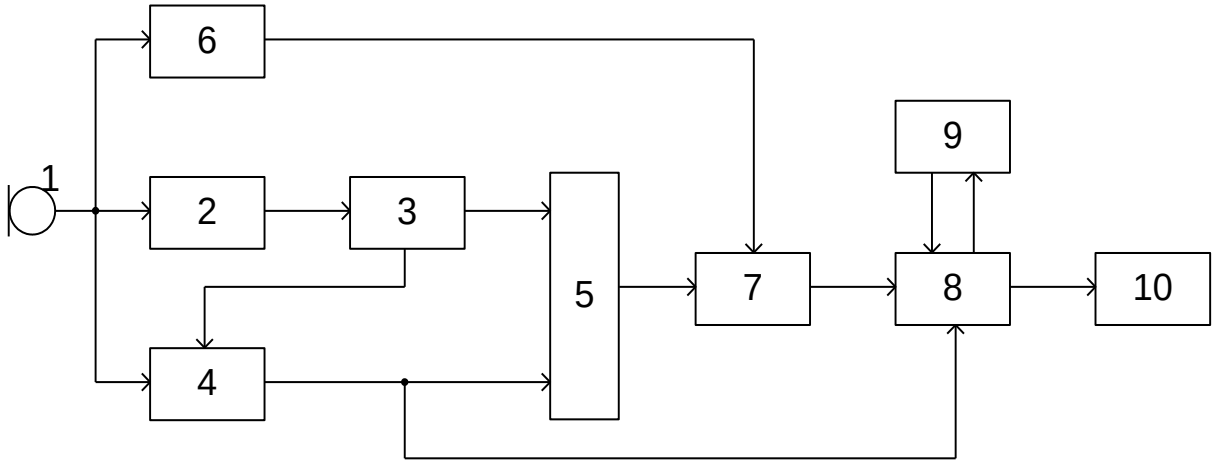

Рис.1.4.1 - Структурная схема устройства выделения признаков речевых сигналов

Задача распознавания речи может быть сведена к задаче распознавания отдельных звуков с последующим использованием алгоритмов,

учитывающих особенности произношения, словопостроения и словосочетания фраз отдельных индивидуумов.

В этом случае задача выделения звуков речи может рассматриваться как задача распознавания образов, количество которых ограничено, хотя и достигает нескольких десятков. При этом сама задача классификации предъявляемых образцов звуков может быть сведена к задаче многоальтернативной проверки гипотез. При этом система распознавания звуков речи может строиться с использованием принципов "обучения с учителем", т.е. предварительного набора информационной базы классифицированных данных, с которыми производится сравнение поступающих на анализ сигналов. Процедура распознавания звуков речи должна учитывать особенности их реализации. Во-первых, эти реализации у каждого звука имеют свой вид. Во-вторых, имеют ограниченную протяженность во времени.

Методы анализа речевых сигналов можно рассматривать с помощью модели, в которой речевой сигнал является откликом системы с медленно изменяющимися параметрами на периодическое или шумовое возбуждающее колебание (рисунок 1.4.2).

Выходной сигнал голосового тракта определяется сверткой функции возбуждения и импульсного отклика линейного, изменяющегося во времени фильтра, моделирующего голосовой тракт. Таким образом, речевой сигнал s(t) выражается следующим образом:

$$S(t) = \int_{-\infty}^{t} e(\tau) v(t, \tau) d\tau ,$$

где e(t) - функция возбуждения, v(t,τ) - отклик голосового тракта в момент t на дельта-функцию, подаваемую на вход в момент τ.

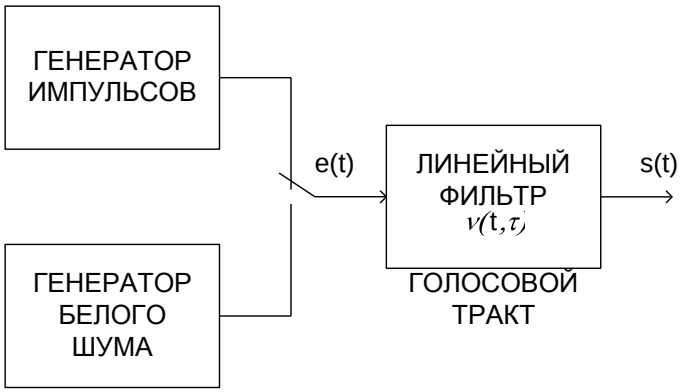

Рис.1.4.2 - Схема функциональной модели формирования речи

Речевой сигнал можно промоделировать откликом линейной системы с переменными параметрами (голосового тракта) на соответствующий возбуждающий сигнал. При неизменной форме голосового тракта выходной сигнал равен свертке возбуждающего сигнала и импульсного отклика голосового тракта. Однако все разнообразие звуков получается путем изменения формы голосового тракта. Если форма голосового тракта изменяется медленно, то на коротких интервалах времени выходной сигнал логично по-прежнему аппроксимировать сверткой возбуждающего сигнала и импульсного отклика голосового тракта. Поскольку при создании различных звуков форма голосового тракта изменяется, огибающая спектра речевого сигнала будет, конечно, тоже изменяться с течением времени. Аналогично при изменении периода сигнала, возбуждающего звонкие звуки, частотный разнос между гармониками спектра будет изменяться. Следовательно, необходимо знать вид речевого сигнала на коротких отрезках времени и характер его изменения во времени.

В системах анализа речевых сигналов обычно пытаются разделить возбуждающую функцию и характеристики голосового тракта. Далее в зависимости от конкретного способа анализа получают параметры, описывающие каждую компоненту.

В частотной области спектр коротких отрезков речевого сигнала можно представить в виде произведения огибающей, характеризующей состояние голосового тракта, и функции, описывающей тонкую структуру, которая

характеризует возбуждающий сигнал. Поскольку основным параметром сигнала, возбуждающего звонкий звук, является разнос гармоник основного тона, а характеристики голосового тракта с достаточной полнотой определяются частотами формант, то при анализе весьма удобно исходить из представления речи в частотной области. При создании различных звуков форма голосового тракта и возбуждающий сигнал изменяются, при этом изменяется и спектр речевого сигнала. Следовательно, спектральное представление речи должно основываться на кратковременном спектре, получаемом из преобразования Фурье.

Рассмотрим дискретизированный речевой сигнал, представленный последовательностью s(n). Его кратковременное преобразование Фурье S(ω,n) определяется как

$$S(\omega,n)=\sum_{k=-\infty}^{\infty}s(k)h(n-k)e^{-j\omega k} \qquad (1.4.1)$$

Данное выражение описывает преобразование Фурье взвешенного отрезка речевого колебания, причем весовая функция h(n) сдвигается во времени.

Линейное предсказание является одним из наиболее эффективных методов анализа речевых сигналов. Этот метод становится доминирующим при оценке основных параметров речевых сигналов, таких как период основного тона, форманты, спектр, а также при сокращенном представлении речи с целью ее низкоскоростной передачи и экономного хранения. Важность метода обусловлена высокой точностью получаемых оценок и относительной простотой вычисления.

Основной принцип метода линейного предсказания состоит в том, что текущий отсчет речевого сигнала можно аппроксимировать линейной комбинацией предшествующих отсчетов. Коэффициент предсказания при этом определяется однозначно минимизацией среднего квадрата разности между отсчетами речевого сигнала и их предсказанными значениями (на

конечном интервале). Коэффициенты предсказания - это весовые коэффициенты, используемые в линейной комбинации. Метод линейного предсказания можно применять для сокращения объема цифрового речевого сигнала.

Основной целью обработки речевых сигналов является получение наиболее удобного и компактного представления содержащейся в них информации. Точность представления определяется той информацией, которую необходимо сохранить или выделить. Например, цифровая обработка может применяться для выяснения, является ли данное колебание речевым сигналом. Сходная, но несколько более сложная задача состоит в том, чтобы классифицировать колебания на вокализованную речь, невокализованную речь и паузу (шум).

В основе большинства методов обработки речи лежит представление о том, что свойства речевого сигнала с течением времени медленно изменяются. Это предположение приводит к методам кратковременного анализа, в которых сегменты речевого сигнала выделяются и обрабатываются так, как если бы они были короткими участками отдельных звуков с отличающимися свойствами.

Одним из наиболее известных методов анализа речи во временной области можно назвать метод, предложенный Л.Рабинером и Р.Шафером в /3/. Он основан на измерении кратковременного среднего значения сигнала и кратковременной функции среднего числа переходов через нуль. Как отмечалось выше, амплитуда речевого сигнала существенно изменяется во времени. Подобные изменения амплитуды хорошо описываются с помощью функции кратковременной энергии сигнала. В общем случае определить функцию энергии можно как

$$E_n = \sum_{m=-\infty}^{\infty} [x(m)w(n-m)]^2$$

Это выражение может быть переписано в виде

$$E_n = \sum_{m=-\infty}^{\infty} x^2(m)h(n-m), \qquad (1.4.2)$$

где $h(n) = w^2(n)$

Выбор импульсной характеристики h(n) или окна составляет основу описания сигнала с помощью функции энергии.

Чтобы понять, как влияет выбор временного окна на функцию кратковременной энергии сигнала, предположим, что h(n) в (1.4.2) является достаточно длительной и имеет постоянную амплитуду; значение En будет при этом изменяться во времени незначительно. Такое окно эквивалентно фильтру нижних частот с узкой полосой пропускания. Полоса фильтра нижних частот не должна быть столь узкой, чтобы выходной сигнал оказался постоянным. Для описания быстрых изменений амплитуды желательно иметь узкое окно (короткую импульсную характеристику), однако слишком малая ширина окна может привести к недостаточному усреднению и, следовательно, к недостаточному сглаживанию функции энергии. Влияние ширины временного окна на точность измерения кратковременного среднего значения (средней энергии):

если N (ширина окна в отсчетах) мало (порядка периода основного тона и менее), то En будет изменяться очень быстро, в соответствии с тонкой структурой речевого колебания,

если N велико (порядка нескольких периодов основного тона), то En будет изменяться медленно и не будет адекватно описывать изменяющиеся особенности речевого сигнала.

Это означает, что не существует единственного значения N, которое в полной мере удовлетворяло бы перечисленным требованиям, так как период основного тона изменяется от 10 отсчетов (при частоте дискретизации 10 кГц) для высоких детских и женских голосов и до 250 отсчетов для очень низких мужских. N выберем равным 100, 200, 300 отсчетов при частоте дискретизации 8 кГц.

Основное назначение En состоит в том, что эта величина позволяет отличить вокализованные речевые сегменты от невокализованных. Значение функции кратковременного среднего значения сигнала для невокализованных сегментов значительно меньше, чем для вокализованных.

Характерной особенностью метода анализа речевых сигналов является бинарное квантование входного речевого сигнала. Возможность выделения параметров сигналов, подвергшихся бинарному квантованию, показана в [2]. Используемая математическая модель речевого сигнала имеет вид:

$$S(t) = A(t) \cdot e^{j\Psi(t)}, \qquad (1.4.3)$$

где A(t) - закон изменения амплитуды речевого сигнала, Ψ(t) - полная фазовая функция речевого сигнала.

Закон изменения амплитуды сигнала не является достаточно информативным параметром для оценки речевого сообщения, так как он не является постоянным для одного и того же слова или фразы, произнесенных с различной интонацией и громкостью. В качестве информативной характеристики речевого сигнала в предлагаемом методе полагается полная фазовая функция речевого сигнала. Полная фазовая функция речевого сигнала представляется в виде разложения в ряд Тейлора:

$$\Psi(t) = \frac{\Psi^{(0)}(t_0)}{0!}t^0 + \frac{\Psi^{(1)}(t_0)}{1!}t^1 + \frac{\Psi^{(2)}(t_0)}{2!}t^2 + \frac{\Psi^{(3)}(t_0)}{3!}t^3 + \ldots \qquad (1.4.4)$$

Выражение (1.4) можно переписать следующим образом

$$\Psi(t) = \mu_0 + \mu_1 t + \frac{\mu_2 t^2}{2} + \frac{\mu_3 t^3}{6} + \ldots \qquad (1.4.5)$$

1.4. Алгоритмы распознавания речи

1.4.1. Нейронные сети

Основу нейронной сети составляют как правило однотипные элементы, имитирующие работу биологического нейрона, и называемые обычно так же.

Каждый из нейронов в каждый момент времени находится, как и биологический нейрон, в некотором текущем состоянии. Он имеет группу однонаправленных входных связей-синапсов, идущих от входа в сеть или от других нейронов. Кроме того он имеет одну однонаправленную выходную связь-аксон.

Схематически нейрон можно представить так:

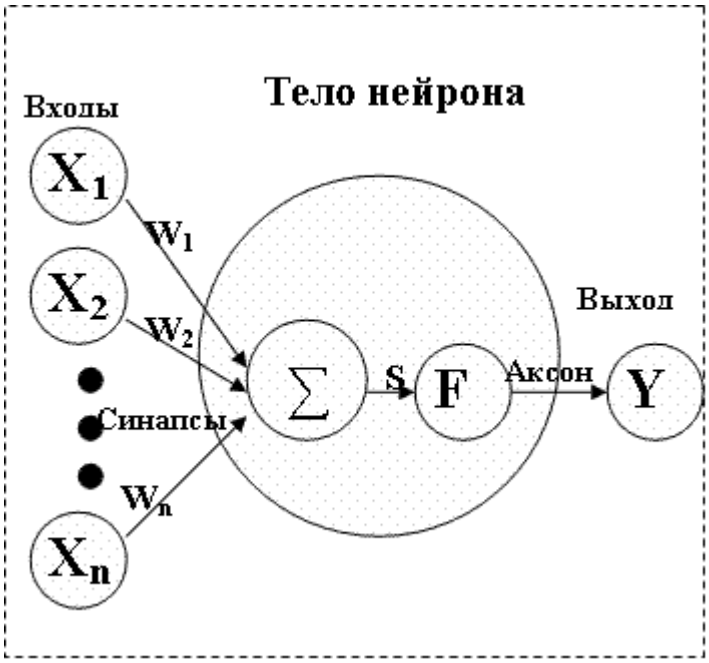

Рис.1.4.1. Нейрон в схематической форме.

Синаптические связи характеризуются весами wi. Текущее состояние S нейрона равно взвешенной сумме входов:

$$S = \sum_{i=1}^{n} X_i w_i$$

Но вернемся к однослойному персептрону. Он обнаружил ряд положительных свойств, которые и заставили многих ученых обратить свой взор на исследование нейронных сетей. Главными из обнаруженных свойств персептрона была способность к обучению и распознаванию. То есть оказалось возможным в ряде случаев установить то, как можно настроить веса синапсов персептрона, чтобы при различных комбинациях значений входов получать заранее установленные, “правильные” значения выходов. То

есть однослойный персептрон оказался способным воспроизводить некоторые математические функции.

Однако эйфория по поводу этих открытий оказалась недолгой: были обнаружены существенные ограничения в способностях однослойного персептрона к обучению и распознаванию.

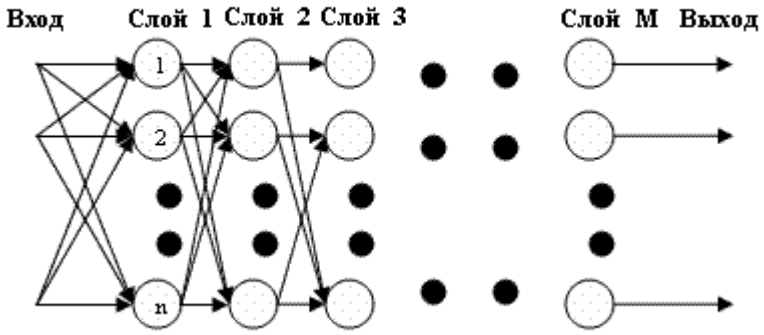

Рис.1.4.1. Многослойный персептрон.

Во-первых было доказана его неспособность воспроизводить некоторые простые функции, например ИСКЛЮЧАЮЩЕЕ ИЛИ.

Если Tj – пороговое значение j-го выхода, то однослойный персептрон описывается уравнениями:

$$T_j = \sum_{i=1}^{n} x_i w_{1,j,i}$$

где n – число нейронов в слое. В n-мерном пространстве эти функции оказываются прямыми. Точки n-мерного пространства входов для значений дающих разные значения (0 или 1) выхода (то есть F>Tj или F<=Tj) должны лежать по разные стороны от таких прямых. Но для многих функций (например ИСКЛЮЧАЮЩЕЕ ИЛИ) это невыполнимо.

Все было бы очень грустно, если бы не существовало многослойных нейронных сетей. Уже при количестве слоев равном двум сеть описывается уравнениями:

$$T_j = \sum_{i=1}^{n} \left(\sum_{i=1}^{n} x_i w_{1,j,i} \right) w_{2,j,i}$$

1.4.2. Алгоритмы обратного распространения

Сложнее обстоит дело с многослойными сетями, так как изначально неизвестны желаемые выходы слоев сети (за исключением последнего) и их невозможно обучить, руководствуясь только величиной ошибок на выходе сети, как это было с однослойной сетью.

Наиболее приемлемым вариантом решения проблемы стала идея распространения сигнала ошибки от выхода сети к ее входу, слой за слоем. Алгоритмы, реализующие обучение сети этой схеме, получили название алгоритмов обратного распространения. Наиболее распространенный вариант этого алгоритма мы и рассмотрим и в дальнейшем применим в программной реализации задачи.

Алгоритм требует дифференцируемости активационной (или как ее по-другому называют, сжимающей) функции на всей оси абсцисс. По этой причине, функция единичного скачка не может использоваться и в качестве сжимающей функции обычно применяют упомянутый выше сигмоид (логистическую функцию), хотя существуют и другие варианты.

Рассматриваем “классический вариант” многослойной сети, где синаптические связи могут определяться любыми действительными числами, а выход нейрона – действительными числами из интервала от 0 до 1. В качестве активационной функции используем сигмоид. Число слоев произвольное.

1. Определяем M матриц весовых коэффициентов W размером NxN, где M – число слоев, N – число нейронов в одном слое. $W_{i,j,k}$ будет обозначать вес j-го входа k-го нейрона в i-м слое. Инициализируем матрицы некоторыми малыми случайными (не одинаковыми) значениями.

2. Подаем на входы сети определенные значения X, для которых известны правильные значения выходов сети Y*.

3. Вычисляем значения выходов сети для текущего состояния матриц W. То есть для входного вектора X вычисляется выходной вектор Y. Для этого необходимо последовательно вычислить выход для каждого слоя сети с первого по последний. Для i-го слоя в векторном виде это можно записать так:

$$O_i=F(XW_i), \text{ если i – не первый слой.}$$

$$O_i=F(O_{i-1}W_i), \text{ если i – не первый слой.}$$

где O_i – вектор выхода i-го слоя, F – активационная функция, X – вектор входов, O_{i-1} – вектор выхода (i-1)-го слоя, W_i – матрица весовых коэффициентов i-го слоя.

4. Вычисляем вектор $\Delta Y=Y-Y^*$

5. Если ΔY меньше заданной погрешности, переходим к шагу 9.

6. Для слоя с номером M (т.е. в последнем слое) производим следующие операции:

6.1. Для всех нейронов в слое с номера 1 по N производим следующие операции:

6.1.1. Для всех весов нейрона с номера 1 по N производим следующие операции:

6.1.1.1. Рассчитываем вектор $\delta M=X(1-X)\Delta Y$

6.1.1.2. Рассчитываем величину $\Delta W_{M,j,k}=\eta\delta_{M,k}O_{i-1,j}$,

где η – коэффициент скорости обучения (от 0.01 до 1.0)

6.1.1.3. Корректируем величину весового коэффициента, добавляя к $W_{M,j,k}$ величину $\Delta W_{M,j,k}$

7. Для слоев с номером M-1 по первый последовательно производим следующие операции:

7.1. Для всех нейронов в слое с номера 1 по N производим следующие операции:

7.1.1. Для всех весов нейрона с номера 1 по N производим следующие операции:

7.1.1.1. Рассчитываем вектор

$$\delta_i = O_{i+1}(1-O_{i+1})[\sum_{k=1}^{N} \delta_{i+1,j} W_{i+1,\,j,\,k}]$$

7.1.1.2. Рассчитываем величину $\Delta W_{i,j,k} = \eta \delta_{i,k} O_{i-1,j}$,

7.1.1.3. Корректируем величину весового коэффициента, добавляя к WM, j, k величину WM,j,k

8. Переход к шагу 3.

9. Конец (обучение окончено).

1.4.3. Скрытая Марковская модель

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

Таким образом, скрытой марковской моделью называется тройка $\lambda=\{A,B,\pi\}$. Использование скрытых марковских моделей для распознавания речи основано на двух приближениях:

1) Речь может быть разбита на фрагменты, соответствующие состояниям в СММ, параметры речи в пределах каждого фрагмента считаются постоянными.

2) Вероятность каждого фрагмента зависит только от текущего состояния системы и не зависит от предыдущих состояний.

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

1.6. Существующие программные продукты распознавания речи и их сравнительный анализ

1.6.1. Бесплатное программное обеспечение

XVoice

XVoice - это распознаватель непрерывной речи, который может быть использован различными XWindow-приложениями. Он позволяет пользователям установить макросы. Это прекрасная программа с определенным будущем. Она выполняет распознавание с достаточной точностью.

XVoice требует от пользователя загрузки и установки IBM's (бесплатный) ViaVoice для Linux (см. коммерческое ПО). Программа также требует корректной настройки ViaVoice для правильной работы. Требуется дополнительное ПО: Lesstif/Motif (libXm). Важно также отметить, что, поскольку эта программа взаимодействует с X Window, необходимо оставить открытым X-ресурс на вашем компьютере, иначе нужно будет осторожно

использовать, если программа используется на сетевой или многопользовательской машине.

Это программное обеспечение предназначено для пользователей. RPM доступен.

CVoiceControl/kVoiceControl

CVoiceControl (Console Voice Control – консольное голосовое управление) начинал свою жизнь как KVoiceControl (KDE Voice Control). Это базовая система распознавания речи, которая позволяет пользователю выполнять команды Linux с помощью голосовых команд.CVoiceControl заменил KVoiceControl.

Программное обеспечение включает в себя утилиту для регулирования громкости микрофона, редактор словаря для добавления новых команд и высказываний и систему распознавания речи. CVoiceControl является отличной отправной точкой для опытных пользователей, желающих начать работу с APP. Это не самый лучший программный продукт для пользователя, но если обучение прошло хорошо, он может быть очень полезным.

Это программное обеспечение предназначено для разработчиков.

GVoice

GVoice - речевая ASR библиотека, которая использует IBM ViaVoice (бесплатную) SDK для управления Gtk/GNOME-приложениями. Она включает в себя библиотеки для инициализации, движки распознавания, манипуляции словарем и панель управления. Дальнейшая разработка этой программы приостановлено несколько лет назад.

Это программное обеспечение предназначено для разработчиков.

ISIP

Институт сигналов и обработки информации при Государственном университете Миссисипи разработал движок распознавания речи Инструмент включает в себя интерфейс ввода, декодер, модуль обучения.

Это программное обеспечение предназначено для разработчиков.

CMU Sphinx

Sphinx разработан открытым исходным кодом группой CMU. Система довольно большая и включает в себя множество инструментов и информации. Хотя система постоянно находится в разработке, но все в нее входят такие компоненты как трейнер, распознаватель, акустическая модель, языковая модель и документация.

Это программное обеспечение предназначено для разработчиков.

ПО СММ Майерза

Это программное обеспечение от Ричарда Майерза (Richard Myers) с алгоритмом скрытой Марковской модели написанная на языке C++. Оно поставляется с инструментами примеров и обучения для СММ, который были описаны в книге Л. Рабинера (L. Rabiner) «Основы распознавания речи» ("Fundamentals of Speech Recognition").

Это программное обеспечение предназначено для разработчиков.

1.6.2. Коммерческое программное обеспечение

Dragon Naturally Speaking 7.0 Preferred

За свою долгую историю Dragon прошел весь нелегкий путь от солдата до маршала. Вначале пользователю будет предложено откалибровать уровень звука из микрофона и надиктовать компьютеру ряд уже готовых текстов для более тонкой подстройки Dragon Naturally Speaking под ваши тембр, интонацию и произношение. И наконец, интерактивный tutorial, где пользователя обучают базовым голосовым командам.

Accuracy Center, который позволяет оптимизировать пользовательский профиль и научить, как пополнять словарь популярными нео-логизмами. Возможны и более экзотические действия вроде распознавания текстового

содержимого wav-файла (в том числе и с Pocket PC или напрямую с линейного выхода аудиоплаты). Кроме того, Dragon Na-turally Speaking умеет запускать различные программы, переключаться между ними и даже управлять рядом их функций (например, начинать/приостанавливать воспроизведение музыки в медиапроигрывателе или напрямую работать с меню). Ну а в состав версий Preferred и Pro-fessional дополнительно входит собственный речевой движок Real-Speech 2, один из наиболее совершенных на сегодня.

Intelligent Voice Recognition System (IVOS)

Самая скромная (по размерам дистрибутива) программа в обзоре проявила себя на удивление достойно и в значительной мере оправдала свое громкое название.. IVOS позволяет: а) распознавать речь и преобразовывать ее в текст в любом Windows-совместимом текст-процессоре; б) управлять своим ПК с помощью разнообразных голосовых команд, а также создавать свои собственные; в) озвучивать электронные книги с помощью внешних голосовых движков. Плюс, разумеется, такие мелочи, как извлечение текста из Wav-файлов, удобная, не отягощающая экран панель управления программой и демократичная (по сравнению с Dragon) цена. После регистрации пользователю становится доступна технология VoiceTouch, позволяющая обучать ПК собственным устным приказам.

Эффективность исполнения команд на удивление высока. А вот уровень распознавания непрерывной речи пониже. Надо отметить, что IVOS, как и многие другие программы распознавания речи, кроме Dragon, использует для таких целей модуль Speech API от Microsoft, и ее результативность в данной области напрямую зависит от творческих успехов этой корпорации. Тем не менее, добиться качественной работы от IVOS можно уже сейчас, начитав программе все наличествующие в ее запасе обучающие тексты. Конечно, до уровня Dragon Naturally Speaking в итоге она не дотянет, но набирать не слишком сложные документы ей вполне под силу.

Realize Voice 4.0

Realize Voice, в отличие от ранее рассмотренного Dragon Naturally Speaking, не совсем хорошо приспособлена к стенографированию (хотя такая функция в ее арсенале и имеется), но с другой стороны блестяще справляется с голосовыми командами. Что примечательно, исключительно глубоких знаний в области английского не нужно - благодаря умному модулю эвристического анализатора программа без особых проблем найдет общий язык практически с любым диктором. Спектр функций Realize Voice довольно широк: от запуска исполняемых файлов и ярлыков программ до работы с корреспонденцией и сложными макросами. Весь процесс полностью автоматизирован и выполняется буквально мгновенно. Правда, некоторые неудобства вызывает невозможность использования в названии команд цифр - к примеру, запустить DOOM 3 с помощью голосового приказа удастся, лишь переименовав его ярлык в "DOOM Three". То же, кстати, касается и кириллицы? Впрочем, в подобном случае всегда можно прибегнуть к ручной настройке программы, напрямую указав путь к интересующему файлу/документу/графическому изображению и т. д. Здесь уже название файла и его координаты никакого значения не имеют - будь он хоть абвгд.exe, да и *Рабочий стол* заполнять ярлыками не придется. Весьма порадовал и набор встроенных системных команд для работы с Windows - хоть он и не слишком велик, но перемещаться между открытыми окнами, эмулировать действие самых распространенных клавиш (*Spacebar, Insert, Home* и т. д.) [12].

1.6.3. Сравнительная таблица

Для того чтобы выбрать именно ту систему, которая удовлетворяла бы всем требованиям, необходимо было провести их сравнительный анализ. Сравнение систем приводится в этой таблице:

Таблица 1.6.1. Сравнительная характеристика систем APP.

Система распознавания речи	Бесплатно	Работа под Windows	Работа под Linux	Возможность добавления нового языка	Интегрирование в собственные приложения	Распознавание непрерывной речи	Требовательность к системным ресурсам
Dragon Naturally Speaking	Нет	Да	Нет	Нет	Нет	Да	Высокая
IVOS	Нет	Да	Нет	Нет	Нет	Нет	Средняя
IBM ViaVoice	Частично	Да	Да	Нет	Да	Да	Средняя
Microsoft Speech API	Да	Да	Нет	Нет	Да	Нет	Низкая
XVoice	Да	Нет	Да	Нет	Нет	Нет	Низкая
CVoiceControl	Да	Нет	Да	Нет	Нет	Нет	Средняя
CMU Sphinx	Да	Да	Да	Да	Да	Да	Средняя

Вышеуказанная таблица наглядно показывает, что система CMU Sphinx подходит по всем критериям, для достижения цели данной выпускной квалификационной работы.

2. Система распознавания речи SPHINX

2.1. Общие сведения

Система Sphinx - дикторонезависимая система распознавания непрерывной речи, которая использует скрытую Марковскую акустическую модель и n-грамную статическую модель, которая была разработана Кай-фу Ли (Kai-Fu Lee). Sphinx демонстрирует выполнимость распознавания непрерывной дикторонезависимой речи с объемным словарем, осуществимость которой была под сомнением до сегодняшнего дня.

В 2000 г. разработчики создали несколько компонентов распознавания речи с открытым исходным кодом, включая Sphinx 2, а позже и Sphinx 3 (в 2001). Речевые декодеры поставляются с акустической моделью и демонстративными приложениями. Существующие ресурсы встраиваются в дополнительные приложения для обучения акустической модели, компиляции лингвистической модели и словарь доменного произношения [8].

Разработанная группой Sphinx университета Карнеги Мелона в сотрудничестве с Sun Microsystems Laboratories, Mitsubishi Electric Research Labs (MERL) и Hewlett Packard (HP), при участии Калифорнийского университета Санта-Крус (UCSC) и Массачусетского технологического института (MIT) современная система распознавания речи Sphinx-4 полностью написана на языке программирования Java™ [9].

На рис. 2.1.1 представлена обобщенная структура системы Sphinx, взаимодействие компонентов которой продемонстрирована стрелками. Рис. 2.1.2 демонстрирует еще более обобщенную схему системы распознавания речи.

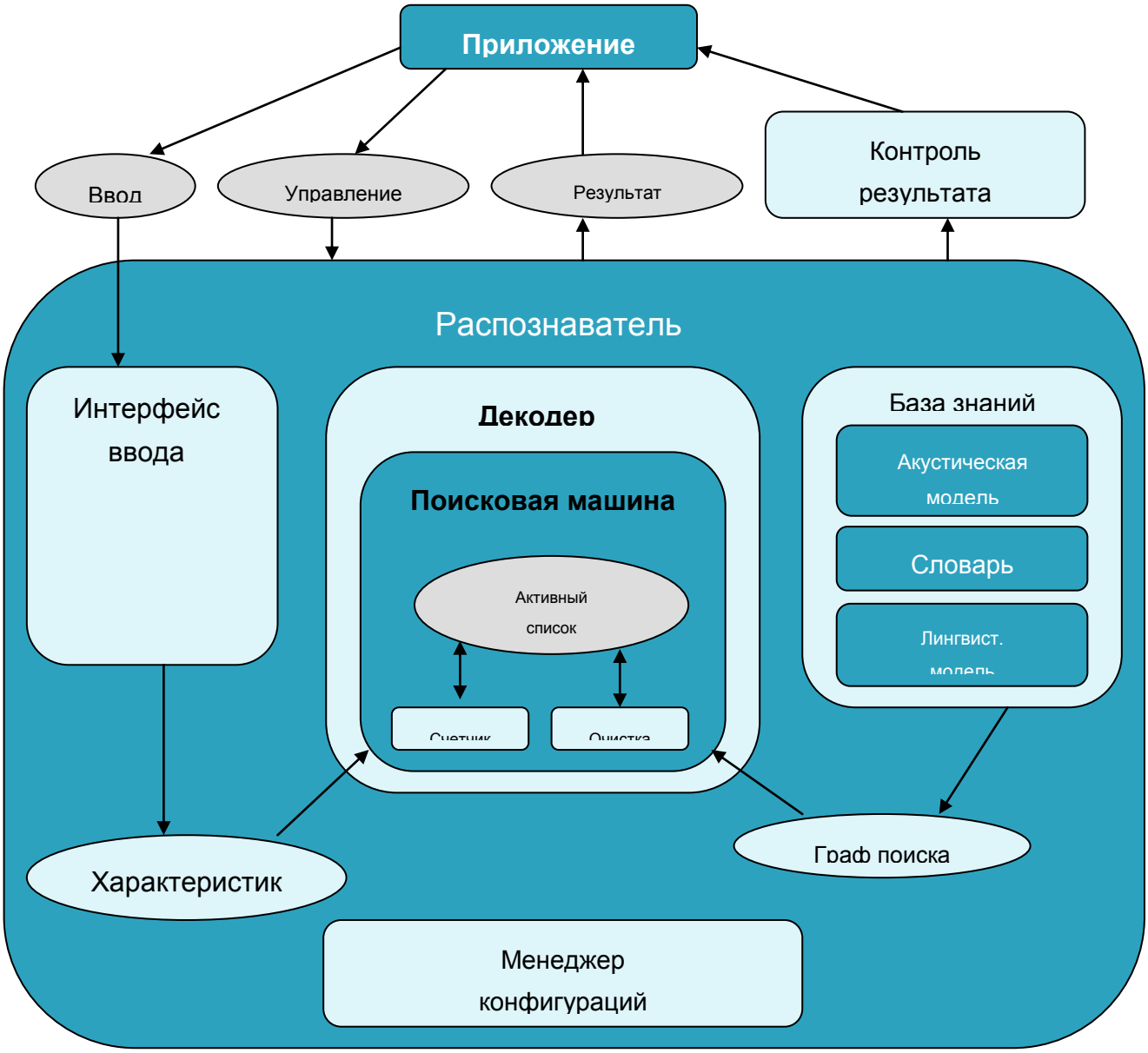

Рис. 2.1.1. Структура декодера Sphinx-4 Decoder. Основные модули: внешний интерфейс, декодер и языковая модель. Дополнительные модули: менеджер конфигураций и инструментарий. Связь между модулями и между приложением.

Механизм распознавания речи требует два основных компонента: акустическую модель, создаваемую при помощи аудио записей речи и их транскрипции (свод речевых правил), и их компиляцию в статическое представление звуков, которые будут составлять слова (через процесс, называемый «обучение»). К ним требуется также лингвистическая модель

или файл грамматики. Файл грамматики содержит набор определенных комбинаций слов. Лингвистическая модель используется приложениями диктовки, тогда как грамматика используется для голосовых команд и управления, или же в телефонной IVR (интерактивный речевой ответ).

Рис. 2.1.2. Более обобщенная структура системы распознавания речи.

Аудиозапись может быть закодирована с различной частотой дискретизации (8 kHz, 16 kHz, 32 kHz, 44.1 kHz, 48 kHz и 96 kHz) и разной разрядностью (8 бит, 16 бит или 32 бита). Для высокой точности распознавания аудиозапись должна быть записана с той же частотой дискретизации и разрядностью, что и речь, которая привлекается к распознаванию[10].

2.2. Изучение структуры системы Sphinx-4

Акустическая модель позволяет оценить распознавание речевого сегмента с точки зрения схожести на звуковом уровне. Современная акустическая модель системы Sphinx для, так называемого пофонемного распознавания, основана на использовании скрытых Марковских моделей (Hidden Markov Models — HMM).

Идея заключается в том, что для каждого звука изначально строится сложная статистическая модель, которая описывает произнесение этого звука в речи. Для того чтобы акустическая модель учитывала произнесение звуков людьми разного пола, возраста, с разным тембром и акцентом, акустическую модель «тренируется» на специально подобранных и отсегментированных

речевых базах большого объема, включающих речь сотен различных людей. В результате, несколько тысяч моделей фонем в разных фонетических контекстах являются основой дикторонезависимого пофонемного распознавания речи на определенном языке.

Модели языка. Использования чисто акустической информации недостаточно для осуществления качественного распознавания речи. Например, в реальных условиях (при наличии посторонних шумов и искажений речевого сигнала) ни одни, даже самые точные, акустические модели не смогут отличить слово «крюк» от слова «трюк».

В такой ситуации важна информация о контексте (теме разговора) и, что еще более важно, о тех словах, которые уже были распознаны ранее. Например, если ранее было распознано слово «железный», то в этой ситуации гораздо вероятнее ожидать произнесения слова «крюк», чем «трюк». Подобная оценка и осуществляется лигвистической моделью.

Модели языка бывают двух основных видов: на основании грамматик и статистические.

Статистическая лингвистическая модель определяет вероятность последовательности m слов с помощью распределения вероятностей $P(w_1, \ldots, w_m)$.

Моделирование языка используется во многих приложениях обработки естественных языков, таких, как распознавание речи, машинный перевод, маркировка части речи, анализ и поиск информации.

В распознавании речи и сжатии данных, такая модель представляет собой фиксацию свойств языка, а также предсказывание следующего слово в последовательности речи.

При поиске информации, модель языка связана с данными из коллекции. При запросе Q в качестве входных данных, полученные данные ранжируются на основе вероятностей, которые будет генерировать условия запроса модель языка данных, $P(Q \mid Md)$.

Оценка вероятности последовательностей может затрудниться в корпусах, в которых фразы или предложения могут быть сколь угодно долго и, следовательно, некоторые последовательности не наблюдаются в процессе подготовки модели языка (проблема разреженности данных избыточного обучения). По этой причине эти модели часто аппроксимируются с помощью использования сглаженных N-грамм моделей.

В n-граммной модели вероятность $P(w_1, \ldots, w_m)$ наблюдения предложения w1,...,wm аппроксимируется как

$$P(w_1, \ldots, w_m) = \prod_{i=1}^{m} P(w_i | w_1, \ldots, w_{i-1}) \approx \prod_{i=1}^{m} P(w_i | w_{i-(n-1)}, \ldots, w_{i-1})$$

Здесь предполагается, что вероятность наблюдения слова w_i в истории контекста предыдущих i-1 слов можно проаппроксимировать вероятностью его наблюдения в сокращенной истории контекста предыдущих n-1 слов (n-ый порядок Марковского свойства).

Условия вероятности могут быть вычислены из n-граммного подсчета частот:

$$P(w_i | w_{i-(n-1)}, \ldots, w_{i-1}) = \frac{count(w_{i-(n-1)}, \ldots, w_{i-1}, w_i)}{count(w_{i-(n-1)}, \ldots, w_{i-1})}$$

Слова биграммной и триграммной лигвистической модели указываются как n-граммная лигвистическая модель с n=2 и n=3 соответственно.

Пример. В биграммной (n=2) лигвистической модели вероятность предложения *I saw the red house* аппроксимируется следующим обрзом:

$$P(I, saw, the, red, house) \approx P(I| < s >)P(saw|I)P(the|saw)P(red|the)P(house|red)$$

тогда как в триграммной (n=3)лигвистической модели аппроксимация выглядит как

$$P(I, saw, the, red, house) \approx P(I| < s >, < s >)P(saw| < s >, I)P(the|I, saw)P(red|saw, the)P(house|the, red)$$

Следует учесть, что контекст первых слов заполнены метками о начале предложения (обычно обозначаются <s>).

Орфоэпический словарь CMU (также известный как cmudict) является открытым орфоэпическим словарем, созданный при университете Карнеги Меллона (CMU). Он может использоваться в качестве словаря как для система синтеза речи Festival так и для системы распознавания речи Sphinx.

Формат базы знаний. База знаний хранится в виде текстового файла в формате *слово-два_пробела-произношение.* Если существует несколько произношений для одного слова, все последующие записи сопровождаются индексом в круглых скобках. Произношения кодируются с помощью модифицированной формы системы Arpabet. Разница состоит в ударениях гласных с уровнями 0, 1, 2; однако не все записи имеют ударения. Например, ниже представлены возможные произношения для слова «encyclopedia» [3,4]:

ENCYCLOPEDIA AH0 N S AY2 K L AH0 P IY1 D IY0 AH0
ENCYCLOPEDIA(2) AH0 N S AY2 K L OW0 P IY1 D IY0 AH0

Декодер. Программный компонент системы распознавания, который совмещает данные, получаемые в ходе распознавания от акустической и языковой моделей, и на основании их объединения, определяет наиболее вероятную последовательность слов, которая и является конечным результатом распознавания.

На первый взгляд декодер — наименее нагруженный в научном плане компонент системы распознавания. Однако быстрый и надежный декодер является главным фактором успеха любой прикладной системы распознавания. Создание такого декодера — сложнейшая техническая задача, потребовавшая высочайшей квалификации разработчиков [11].

2.3. Обучающая система акустической модели SphinxTrain

SphinxTrain - тренер акустических моделей с открытым исходным кодом Университета Карнеги Меллон. Он содержит сценарии и инструкции, необходимые для построения моделей для распознавателя CMU Sphinx Recognizer.

Моделирование контекстно-зависимых фонем с несвязанным состоянием: некоторые требования к памяти.

Моделирование контексто-независимых фонем с развязанным состоянием требует большого объема свободного места на жестком диске (рис.2.3.1).

Полунепрерывная модель

Для обучения 5 состояний/CMM моделей для 10,000 трифонов:
5 состояний/трифон = 50,000 состояний
Каждый четырехпоточный набор = 1024 плавающих точек /состояние
*Состояние имеет всего 4*256 миксов*
весы = 205МБ буфера для 50,000 состояний

Таким образом, трифонов можно обучать 12 раз столько, сколько возможно, когда имеется полунепрерывная модель для одного и того же объема памяти. Если возможно использование большого количества Трифонов для обучения (а следовательно больше информации), деревья решений будет совершеннее, и в итоге повышается результативность распознавания речи.

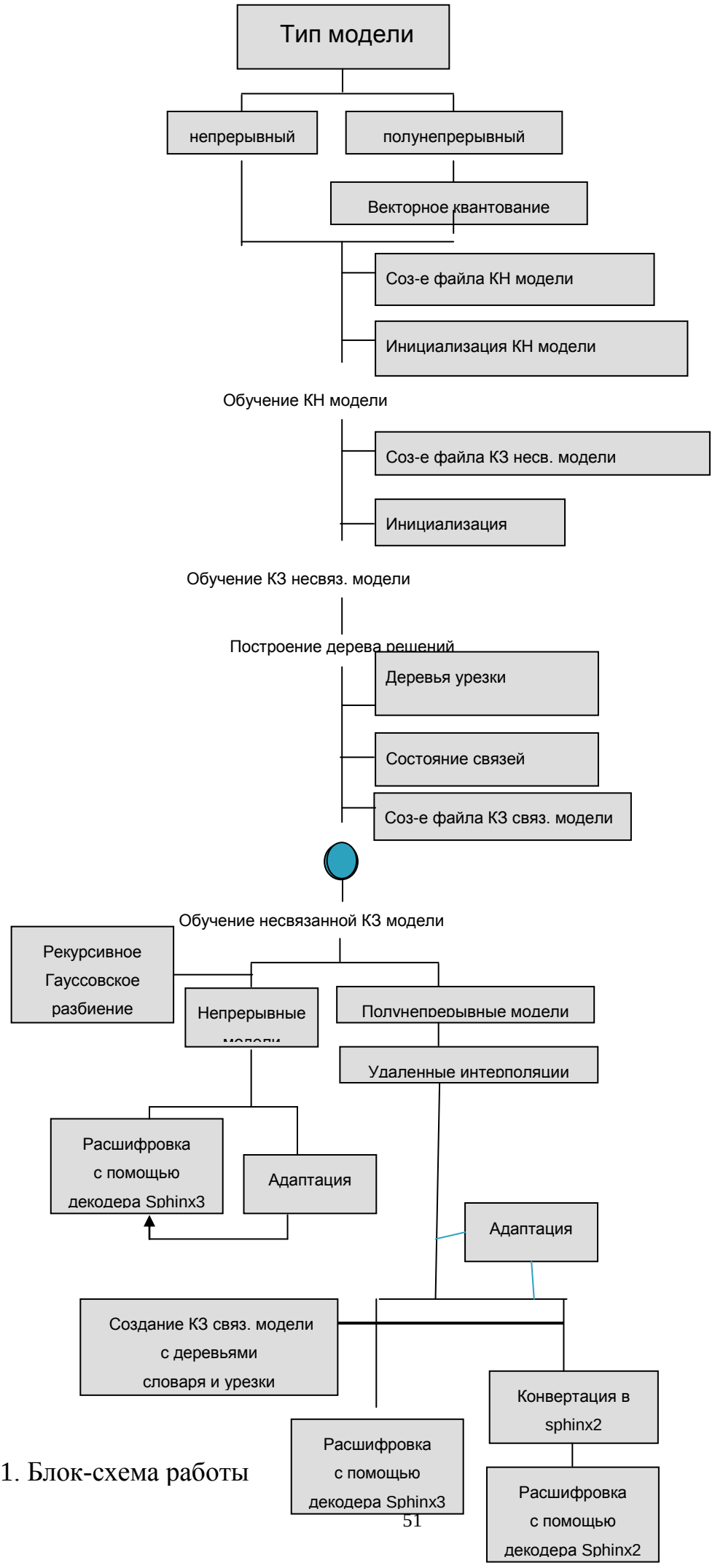

Рис.2.3.1. Блок-схема работы

2.4. Этапы построения языковой модели узбекской языка. Создание лингвистической модели. Создание словаря.

0. Естественно, прежде всего, было загружено и установлено на операционной системе Linux всё необходимое программное обеспечение для полноценной обработки данных. Это:

- ANT;
- Sphinx-4;
- SphinxTrain
- cmuclmtk;
- cygwin;
- python;
- gcc;
- perl.

I. Первым и пожалуй ответственным этапом является *подготовка текста*, который в дальнейшем читался диктором для создания аудиозаписи речи. От содержания текста зависит содержание словаря и строение лингвистической модели. Контекст распознаваемой речи зависит от тематики текста.

II. Важнейшим этапом создания акустической модели любого языка является *запись речи* по подготовленному тексту. При обучении нового языка SphinxTrain использовал именно эти файлы с записью для того чтобы построить акустическую модель. Следовательно, качество речи и записи должно было максимально приближено к идеальному. А для того чтобы работало дикторонезависимое распознавание речи, речь была записана различными голосами. Чем больше голосов для одних и тех же предложений, тем более дикторонезависима система. Второстепенным фактором качества распознавания является продолжительность записи. Запись желательно должна производиться профессиональным диктором, речь которого является равномерным, эмоционально-устойчивым. Для того чтобы алгоритм сработал, запись должна произвестись не менее 1-2 часов. Поэтому длину текста необходимо подобрать так, чтобы длина записи соответствовала требованию длины записи. Для сравнения: встроенная демонстрационная

акустическая модель для распознавания непрерывно диктуемых чисел английского языка записывалось около 1000 часов с 400 голосами.

III. После того, как запись была создана, ее необходимо было *разбить по предложениям*. Каждое разбитое предложение сохранялось отдельным файлом. Частота дискретизации и разрядность устанавливалась по определенным критериям.

IV. Далее был создан файл речевых правил (файл транскрипций), который содержит транскрипции для каждого предложения конкретной аудиозаписи(файл назван *uzlang.trans*). Для каждой транскрипции указаны начало и конец предложения. В конце указаны имена файлов (без расширения) аудиозаписи данного предложения. Пример:

Файл транскрипций содержит:

<s>quyosh nuri xonani yoritmoqda</s>(2)

<s>ertaga ob havo yaxshi bo'lishi aniq</s>(3)

Файлы:

2.wav и *3.wav*

V. Следующий этап – это создание словаря. Словарь содержит (без повтора) все слова, которые находятся в файле транскрипций в алфавитном порядке. После каждого слова с помощью фонем прописывались их произношение. Фонемы указывались по определенному правилу. Пример:

AMALGA AH M AE L G AH

AMALIYOT AE M AH L IY AY AH T

AMMO AE M OH

ANIQAE N AH K

ANIQLANGAN AH N IH K L AE N G AH N

Файл назван *uzlang.dic.*

VI. Здесь в отдельном файле дополнений создаются знаки начала предложения (<s>), конца предложения (</s>) и знак тишины (<sil>). Файл пауз содержит следующее:

<s> SIL

</s> SIL

<sil> SIL

- SIL

Файл назван *uzlang.filler*.

VII. На данном этапе создается одна из важнейших составляющих языковой модели – лингвистическая модель. Для этого был использован набор инструментальных средств cmuclmtk (The CMU-Cambridge Statistical Language Modeling Toolkit v2). Данный набор запускался на ОС Linux. Инструменты были использованы по определенному алгоритму. Блок-схема этого алгоритма приведена ниже:

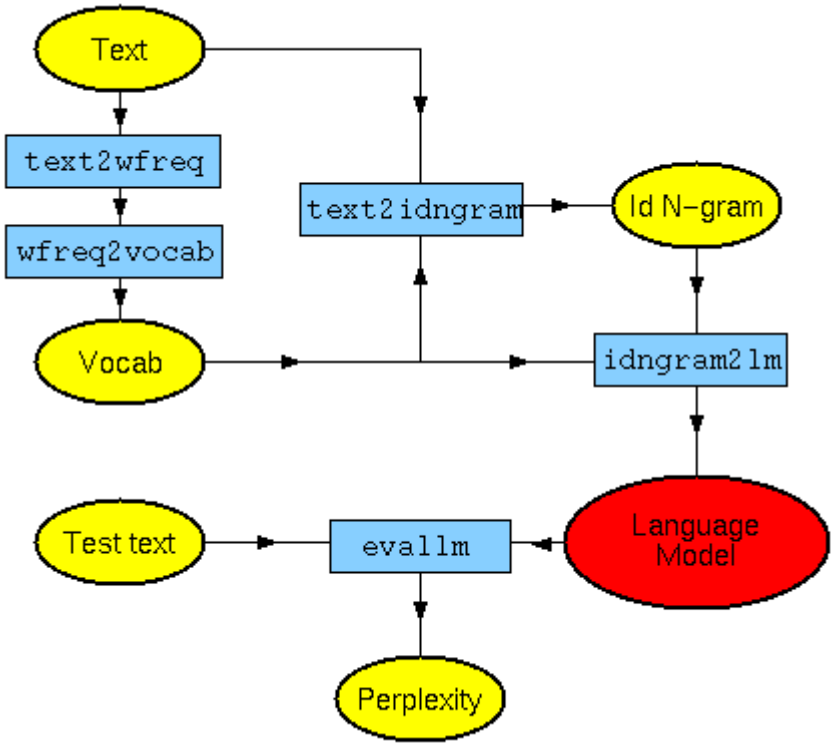

Рис. X. Алгоритм использования каждого инструмента cmuclmtk.

Команды, которые были использованы для вызова определенного инструмента:

1) text2wfreq uzlang.text uzlang.wfreq

2) wfreq2vocab uzlang.wfreq uzlang.vocab

3) text2wngram -n 3 uzlang.text uzlang.wngram

4) text2idngram -vocab uzlang.vocab uzlang.text uzlang.idngram

5) ngram2mgram -n 3 -m 2 [-binary | -ascii | **-words**]
uzlang.ngram uzlang.mgram

6) wngram2idngram -vocab uzlang.vocab uzlang.wngram uzlang.idngram

7) idngram2stats uzlang.idngram uzlang.stats

8) mergeidngram -ascii_output uzlang.idngram_1 uzlang.idngram_2
uzlang.idngram_3

9) idngram2lm -idngram uzlang.idngram
-vocab uzlang.vocab
-arpa uzlang.arpa | -binary uzlang.binlm

10) binlm2arpa -binary uzlang.binlm -arpa uzlang.arpa

Полученный файл лингвистической модели был назван *uzlang.lm*.

VIII. На уровне папки SphinxTrain создана еще одна папка (*uzlang*), которая после определенных процедур будет содержать результаты работы. На уровне этой папки была запущена следующая Perl-команда:
perl ../SphinxTrain/scripts_pl/setup_SphinxTrain.pl –task uzlang

X. Создана папка *uzlang/etc*. В ней создан файл, содержащий список всех аудиофайлов(*uzlang.fiileids*). Затем туда же перенесены файлы *uzlang.trans, uzlang.lm, uzlang.dic, uzlang.filler*.

IX. На этом этапе из словаря извлекаются фонемы из словаря, а также знаки пауз, и внедряются в отдельный файл (без повтора) с помощью команды
perl ./scripts_pl/make_phoneset.pl etc/uzlang.dic etc/uzlang.filler > etc/uzlang.phone

Пример вывода приведен ниже:

HH

F

GG

E

KK

PP

BB

ZZ

SCH

X. Далее необходимо было изменить настройку для файлов аудиозаписи в файле конфигураций *uzlang/etc/sphinx_train.cfg* следующим образом:

$CFG_WAVFILE_EXTENSION = 'wav';

$CFG_WAVFILE_TYPE = 'wav';

XI. Для того чтобы указать обучающему инструменту названия файлов аудиозаписи необходимо было выполнить следующую команду:

perl ./scripts_pl/make_feats.pl -ctl etc/uzlang.fileids

XII. Данный этап является наиболее решающим, так как здесь SphinxTrain начинает обучение системы на основе полученных данных. Для этого была использована следующая команда:

perl ./scripts_pl/RunAll.pl

2.5. Создание акустической модели

Для того чтобы использовать созданные модели инструментом SphinxTrain, необходимо было заархивировать данные JAR-файл. Достоинство данной процедуры является в том, что JAR-файл может упростить внедрение этой библиотеки в приложение с помощью ссылок в файле конфигураций.

В дистрибутиве Sphinx-4 содержится скрипт build.xml с ANT-объектами, которые позволяют преобразовывать модели SphinxTrain в JAR-файл. Ниже приведены несколько этапов, которые были проделаны для того, чтобы построить акустическую модель узбекского языка:

1. Создание директории модели. Все файлы модели размещаются в директории "*sphinx4/models/acoustic*" дистрибутива Sphinx-4. Для нашей модели узбекского языка была создана папка *uzlang*:
   sphinx4> cd models/acoustic
   sphinx4/models/acoustic> mkdir uzlang

2. Копирование файлов модели. Скопированы все файлы модели в директорию *sphinx4/models/acoustic/uzlang/*. После копирования всех необходимых файлов, содержимое директории *uzlang* выглядит следующим образом:
   uzlang.cd_cont_1000_8/
   uzlang.cd_cont_1000_8/means
   uzlang.cd_cont_1000_8/mixture_weights
   uzlang.cd_cont_1000_8/variances
   uzlang.cd_cont_1000_8/transition_matrices
   dict/
   dict/uzlang.dic
   dict/uzlang.filler
   dict/uzlang.phone
   etc/
   etc/uzlang.1000.mdef
   etc/uzlang.alltriphones.mdef
   etc/uzlang.ci.mdef

Все файлы, находящиеся в каталоге *uzlang.cd_cont_1000_8* бинарные

3. Создание файла свойств *model.props*. Создан текстовый файл model.props в папке *sphinx4/models/acoustic/uzlang/*. В нем указаны следующие параметры:

   description = Uzbek Language Model

   modelClass = edu.cmu.sphinx.model.acoustic.uzlang_8gau_13dCep_16k_40mel_130Hz_6800Hz.Model

   modelLoader = edu.cmu.sphinx.model.acoustic.uzlang_8gau_13dCep_16k_40mel_130Hz_6800Hz.ModelLoader

   isBinary = true

   featureType = 1s_c_d_dd

   vectorLength = 39

   sparseForm = false

   numberFftPoints = 512

   numberFilters = 40

   gaussians = 8

   minimumFrequency = 130

   maximumFrequency = 6800

   sampleRate = 16000

   dataLocation = uzlang.cd_cont_1000_8

   modelDefinition = etc/uzlang.1000.mdef

Описание свойств:

Свойство	Описание
modelClass	**"edu.cmu.sphinx.model.acoustic.${значение}.Model"**. ${значение} может использоваться в следующем формате:

	1. the name of the data set used to train the models (uzlang) 2. число гауссов (8gau). 3. число точек данных кепстр (13dCep) 4. частота дискретизации тренировочных данных (16k) 5. число мел-фильтров (40mel) 6. минимальная частота (130Hz) 7. максимальная частота (6800Hz) Таким образом название JAR-файлу было присвоено uzlang_8gau_13dCep_16k_40mel_130Hz_6800Hz.jar.
modelLoader	Класс загрузки модели
dataLocation	Директория, в которой находятся файлы модели: "uzlang.cd_cont_1000_8". Это место связи класса modelLoader внутри JAR-файла.
modelDefinition	Определяет расположение.mdef-файла: "etc/uzlang.1000.mdef". Это расположение касается класса modelLoader за пределами JAR-файла.
isBinary	Определяет бинарные или ASCII файлы модели (файлы means, variances, mixture_weights и transition_matrices).
featureType	Название SphinxTrain для типа функции генерируется из обучающих данных, здесь имя 1s_c_d_dd, означает кепстра дельта кепстра и двойная дельта кепстра. Только модели подготовки 1s_c_d_dd и s3_1x39 поддерживаются Sphinx -4.
vectorLength	Длина вектора характеристик, значение которого обычно 39
sparseForm	Переходящие ли матрицы акустической модели в редкой форме, т. е. исключая нули без перехода состояний.
numberFftPoints	Число БПФ точек, которые используются при создании тренировочных данных.
numberFilters	Число фильтров, которые используются при создании тренировочных данных.

gaussians	Число гауссов модели
maximumFrequency	Максимальная частота мел-фильтров, которые используются при создании тренировочных данных.
minimumFrequency	Минимальная частота мел-фильтров, которые используются при создании тренировочных данных.
sampleRate	Частота дискретизации.

При вызове созданной библиотеки в консоле можно увидеть эти данные. Пример:

sphinx4> java -jar lib/uzlang_8gau_13dCep_16k_40mel_130Hz_6800Hz.jar

Uzbek Language Model

dataLocation: uzlang.cd_cont_1000_8
description: Uzbek Language Model
featureType: 1s_c_d_dd
gaussians: 8
isBinary: true
maximumFrequency: 6800
minimumFrequency: 130
modelClass:
edu.cmu.sphinx.model.acoustic.uzlang_8gau_13dCep_16k_40mel_130Hz_6800Hz.Model
modelDefinition:
etc/uzlang_8gau_13dCep_16k_40mel_130Hz_6800Hz.4000.mdef
modelLoader:
edu.cmu.sphinx.model.acoustic.uzlang_8gau_13dCep_16k_40mel_130Hz_6800Hz.ModelLoader
numberFftPoints: 512
numberFilters: 40

sampleRate: 16000

sparseForm: false

vectorLength: 39

4. Добавление ANT-свойств в build.xml. Были произведены изменения в файле *build.xml*, с помощью которого строятся JAR-файлы акустических моделей. Сперва было определено имя акустической модели. Поэтому после секции, где она начинается с "For generating the WSJ..." необходимо было добавить следующее:

   <property name="uzlang_name"

   value="uzlang_8gau_13dCep_16k_40mel_130Hz_6800Hz"/>

   <property name="uzlang_data_dir" value="models/acoustic/uzlang "/>

5. Создание модельных классов в *build.xml*. После поиска ANT-объекта "create_all_model_classes" было добавлено в контейнер следующее:

   <antcall target="create_my_model_classes">

   <param name="my_model_name" value="${uzlang_name}"/>

   </antcall>

6. Удаление модельных классов в *build.xml*. Эти параметры необходимо добавлять, для того чтобы при желании можно было удалить библиотеки. В контейнере "delete_all_model_classes" было добавлены нижеприведенные строки:

   <antcall target="delete_my_model_classes">

   <param name="my_model_name" value="${uzlang_name}"/>

   </antcall>

7. Создание JAR-модели в build.xml. В ANT-объекте "create_all_models" добавляется последняя запись ANT-вызова:

   <antcall target="create_my_model">

   <param name="my_model_data_dir" value="${toy_data_dir}"/>

<param name="my_model_name" value="${toy_name}"/>
</antcall>

Это был последним шагом редактирования *build.xml* файла.

8. Сборка. Из корневой директории Sphinx-4 была запущена команда "*ant*". Она построила на основе файла build.xml и модельных данных библиотеку акустической модели узбекского языка для распознавания речи. Она находится в папке "lib".

9. Определение модельных классов в файле конфигураций. В файле конфигураций приложения Sphinx-4 необходимо указать адрес акустической модели и словаря:

<component name="uzlang" type=
"edu.cmu.sphinx.model.acoustic.uzlang_8gau_13dCep_16k_40mel_130Hz_6800Hz.Model">
<property name="loader" value="sphinx3Loader"/>
<property name="unitManager" value="unitManager"/>
</component>

<component name="sphinx3Loader"

type="edu.cmu.sphinx.model.acoustic.uzlang_8gau_13dCep_16k_40mel_130Hz_6800Hz.ModelLoader">
<property name="logMath" value="logMath"/>
<property name="unitManager" value="unitManager"/>
</component>

Также необходимо указать местонахождение словаря в JAR-файле:

<component name="dictionary"
type="edu.cmu.sphinx.linguist.dictionary.FullDictionary">
<property name="dictionaryPath"

value="resource:/edu.cmu.sphinx.model.acoustic.uzlang_8gau_13dCep_16k_40mel_130Hz_6800Hz.Model!/edu/cmu/sphinx/model/acoustic/uzlang_8gau_13dCep_16k_40mel_130Hz_6800Hz/dict/uzlang.dic"/>

...

</component>

10. Добаление JAR-файла в приложение. Наконец добавлена библиотека uzlang_8gau_13dCep_16k_40mel_130Hz_6800Hz.jar в Java-приложение.

3. Разработка программы распознавания узбекской речи UzLang

3.1. Связь библиотеки с приложением

После всей проделанной работы в результате была получена Java-библиотека, которая предназначена для внедрения в любое Java-приложение. Это является одним из больших достоинств системы Sphinx-4. Так, если разработать текстовый процессор, можно включить эту библиотеку в состав приложения в качестве дополнительного модуля диктовки, что позволит облегчить ввод текста. Или же это может позволить встроить в Интернет-мессенджер для быстрой передачи сообщений.

Само Java-приложение, разработанное для квалификационной работы, не имеет интерфейса, так как оно работает в фоновом режиме. Структура приложения приведена ниже:

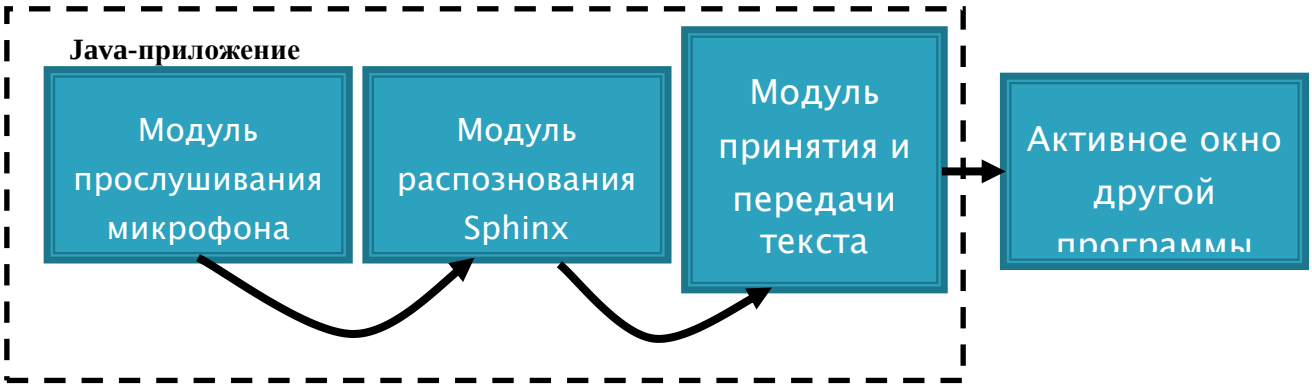

Рис. 3.1.2. Структура Java-приложения.

3.2. Описание программы

В качестве примера можно привести консольное приложения, которое выполняется в среде разработки NetBeans или Eclipse (рис.3.2.1).

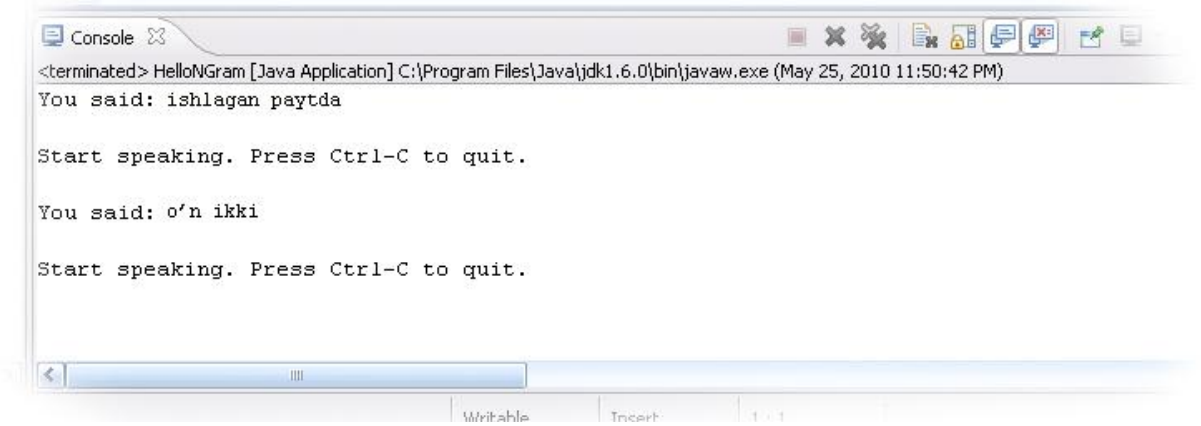

Рис.3.2.1. Консольный вид приложения.

После ее запуска, приложение начинает принимать голосовые сигналы, т.е. прослушивать микрофон. В системе имеется модуль определения начала и окончания речи. После того как речь сказана, программа ее обрабатывает, затем передает результат в системный вывод (рис. 3.2.1).

В данной выпускной квалификационной работе было создано Java-приложение UzLang, которое работает в фоновом режиме. Распознанный текст передается в текстовое поле любого активного окна (рис. 3.2.2).

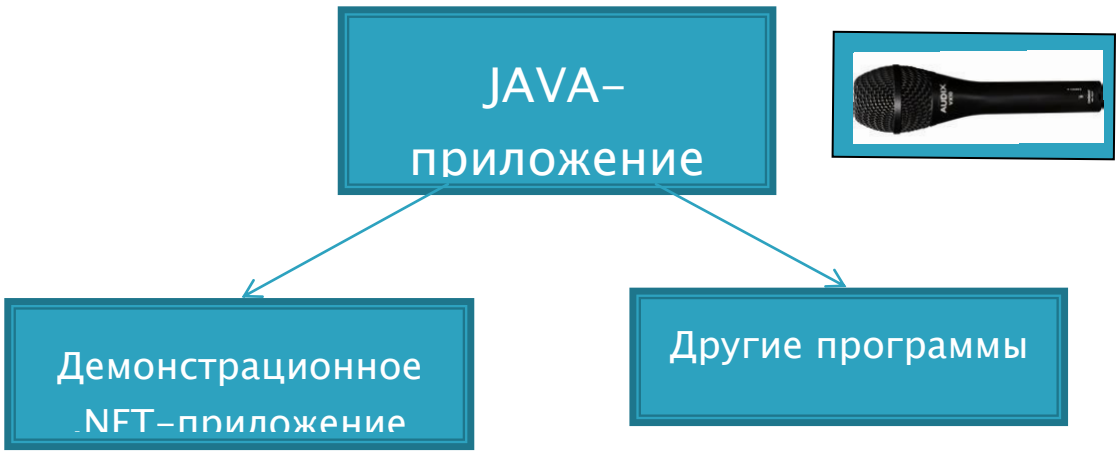

Рис. 3.2.2. Передача распознонного текста в другие приложения.

Данную библиотеку также можно использовать в других приложениях по назначению. Пример такой программы рассматривается ниже.

Имеется пакетный файл, который запускает из директории своего же уровня Java-приложение и .NET-приложение. Распознанный текст Java-приложение отправляет в активное окно.

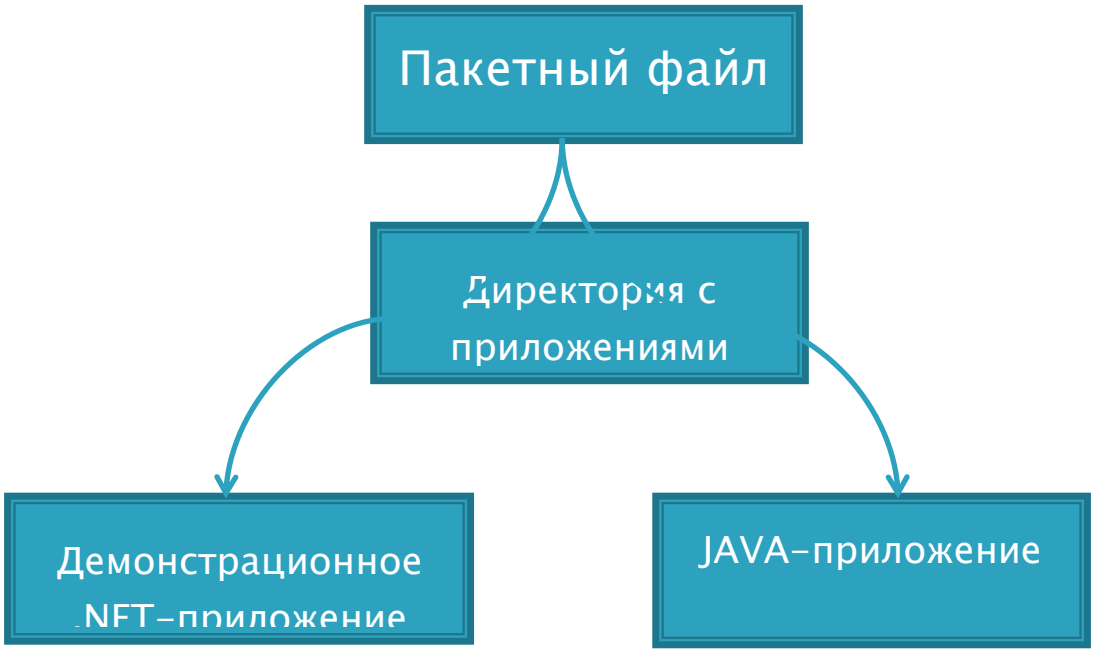

Рис.3.2.3. Связь приложений.

3.3. Руководство пользователя.

В качестве демонстрационного пакета была создана программа, которая, как и все текстовые редакторы, принимает распознанный текст. Приложение имеет модуль графического отображения звука, MS Agent.

После запуска приложения на экране появляется форма, показанная на рис.3.3.1.

В меню «Файл» имеются 5 команд: «Создать», «Открыть», «Сохранить», «Сохранить как…», «Выход». Из меню «Правка» текст можно редактировать четырьмя командами: «Вырезать», «Копировать», «Вставить», «Выделить все». Меню «Формат» имеет команду «Шрифт», который изменят вид текста.

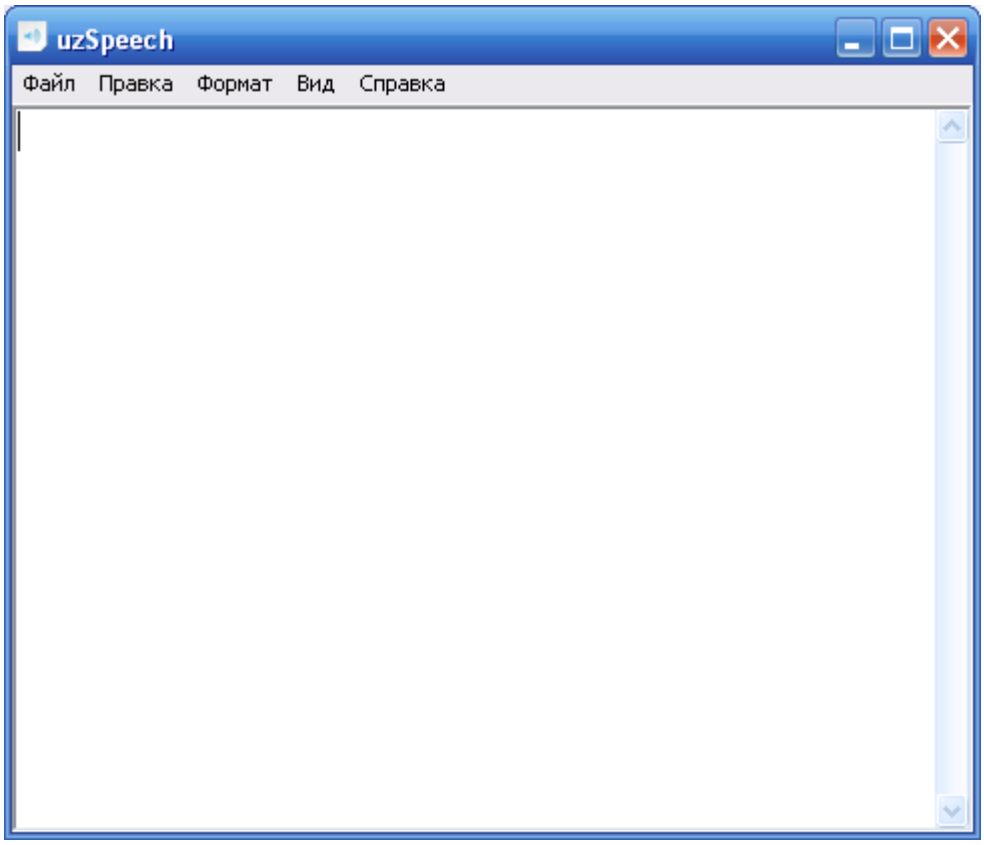

Рис.3.3.1. Форма при запуске,

Также имеются меню дополнительных функции таких как «График», «Агент» и прочее. Вид с агентом показан на рис. 3.3.2.

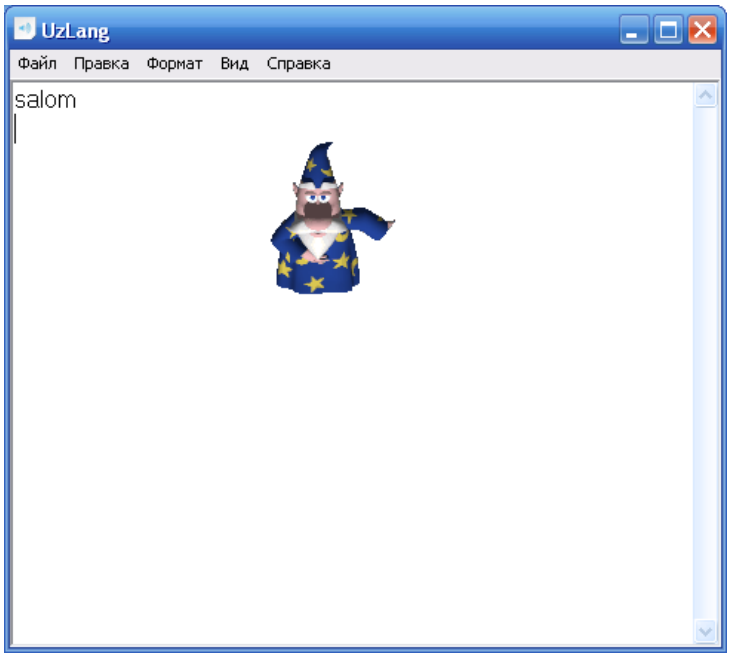

Рис.3.3.2. Вид интерфейса с агентом.

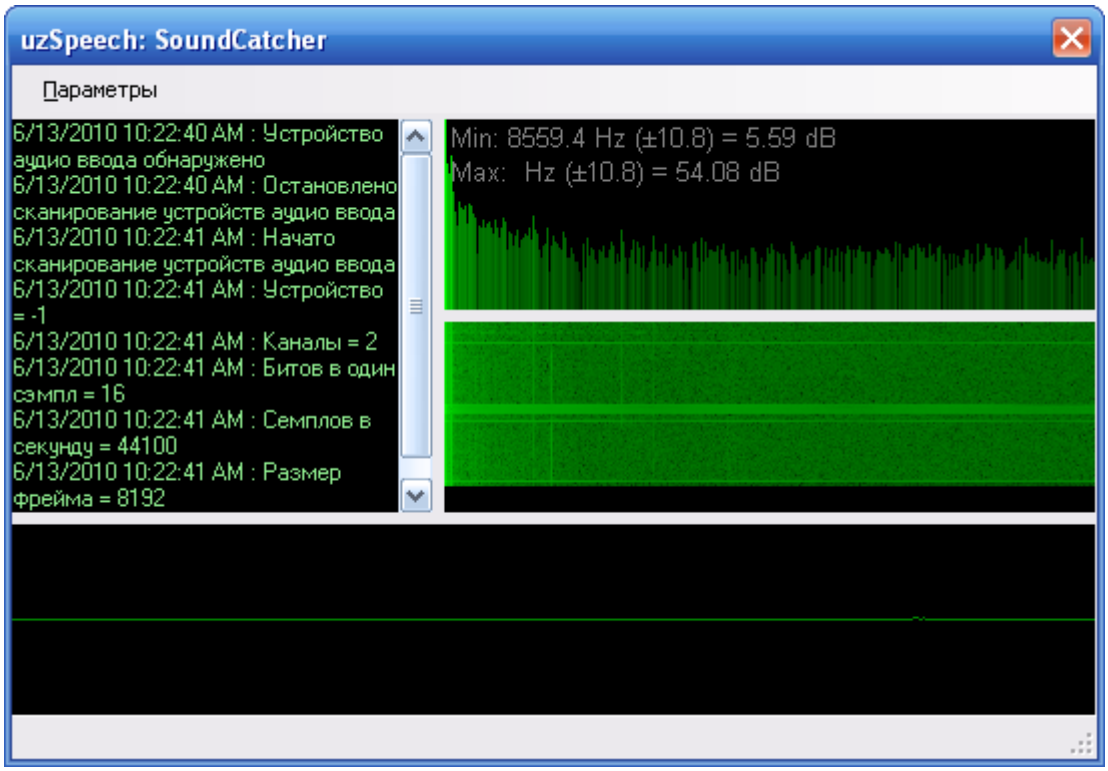

Рис. Изображение звукового сигнала в графическом виде.

3.4. Основные системные требования.

Для полноценной работы программы необходимы следующие ресурсы.

Аппаратное обеспечение:

- Центральный процессор: Двуядерный 2.0 GHz (Dual Core);
- ОЗУ: 2 Гб;
- Свободное пространство жесткого диска: 10 Мб;
- Устройства аудио ввода-вывода: микрофон, источник звука (желательно);

Программное обеспечение:

- ОС: MS Windows XP/Vista/7; Linux, MacOSx, Solaris (только для консольного приложения);
- Java™ SE Runtime Environment ≥ version 1.5;
- .NET Framework ≥ 3.0.

4. Безопасность жизнедеятельности.

4.1. Влияние метеорологических условий производственной среды на организм человека.

Производственный микроклимат или метеорологические условия, определяются состоянием температуры, влажности и движения воздуха производственных помещений, а также тепловым излучением от нагретого оборудования и обрабатываемых материалов.

Производственный микроклимат, как правило, отличается большой изменчивостью, неравномерностью по горизонтали и вертикали, разнообразием сочетаний температуры и влажности движения воздуха и интенсивности излучения. Многообразие это определяется особенностями технологии производства, климатическими особенностями местности, конфигурацией зданий, организацией воздухообмена с внешней атмосферой и т. п.

По характеру воздействия микроклимата на работающих производственные помещения могут быть: с преоблададающим охлаждающим действием и с относительно нейтральным (не вызывающим значительных изменений терморегуляции) действием микроклимата. Существующим санитарным законодательством все цехи делятся на горячие, где избыточные тепловыделения превышают 20 ккал. на один кубический метр объема помещения в час и холодные, где тепло выделения ниже этой величины.

В организме человека непрерывно происходят окислительные реакции, связанные с образованием тепла. Вместе с тем непрерывно происходит и отдача тепла в окружающую среду.

Совокупность процессов, обуславливающих теплообмен между организмом и внешней средой, в результате которого температура тела

поддерживается примерно на одинаковом уровне называется терморегуляцией.

Теплоотдача организма во внешнюю среду зависит от температуры окружающей среды, от количества выделяемой организмом влаги (пота) вследствие затрат тепла на испарение, от тяжести выполняемой работы и физического состояния человека. При высокой температуре воздуха и облучении кровеносные сосуды поверхности тела расширяются; при этом происходит перемещение крови в организме к периферии (поверхности тела). Вследствие такого перераспределения крови теплоотдача с поверхности тела значительно увеличивается. Однако, отдача тепла с поверхности тела путем усилений конвекции и излучения может происходить только при внешней температуре до 30°С. Если температура воздуха выше этого предела, большая часть тепла уже отдается путем испарения влаги с поверхности кожи, а при температуре воздуха, близкой к температуре поверхности тела, теплоотдача происходит только за счет испарения пота. При этом организм теряет большое количество влаги, а вместе с ней и солей, играющих важную роль в жизнедеятельности организма. Так, например, при выполнении тяжелой физической работы в помещении с температурой 30°С потери влаги человеком достигают 10—12 л. в смену.

Иначе реагирует человеческий организм на понижение температуры окружающего воздуха: кровеносные сосуды кожи сокращаются, скорость протекания крови через кожу замедляется и отдача тепла путем конвекции и излучения уменьшается.

Влажность воздуха также оказывает большое влияние на терморегуляцию организма. Повышенная относительная влажность воздуха в помещении (свыше 85%) затрудняет терморегуляцию организма, так как отдача тепла путем испарения пота с поверхности тела будет крайне затруднена.

Особенно неблагоприятные условия наступают для терморегуляции организма в том случае, когда наряду с повышенной влажностью в

помещении поддерживается также и высокая температура (свыше 30°С); наступает быстрое утомление, расслабление организма и прекращение потовыделения. Нарушение терморегуляции ведет к тяжелым последствиям, головокружению, тошноте, потере сознания, тепловому удару.

Движение воздуха способствует увеличению отдачи тепла с поверхности тела путем конвекции, а следовательно, улучшает терморегуляцию организма в жарком помещении, но является неблагоприятным фактором при низкой температуре окружающего воздуха в холодное время года.

Советское законодательство строго регламентирует метеорологические условия в рабочей зоне производственных помещений. Рекомендуемыми нормами метеорологические условия должны обеспечивать такое состояние физических процессов в организме, при котором поддерживалось бы устойчивое благоприятное тепловое состояние организма в течение длительного времени без снижения работоспособности человека и без резких изменений функционального состояния отдельных органов и систем.

Температура воздуха в производственных помещениях должны быть в зависимости от тяжести работ в холодный и переходный период от 17° до 21 °, в теплый — не превышать температуру наружного воздуха на 3—5° и не подниматься выше 28°. Относительная влажность — в пределах 40—60%, скорость движения воздуха, как правило, не более 0,2—0,3 м/сек [6].

Нормальные метеорологические условия обеспечиваются следующими мероприятиями:

- защита от источника излучения;
- обеспечение оптимального воздухообмена;
- механизация тяжелых работ;
- применение индивидуальных средств защиты;

4.2. Чрезвычайные ситуации. Защита предприятия в чрезвычайных ситуациях и ликвидация последствий.

Повседневная деятельность и отдых, способ существования человека определяются понятием жизнедеятельность. Стремление человечества к повышению комфортности среды жизнедеятельности, к обеспечению защиты от естественных негативных воздействий привело во многих регионах нашей планеты к разрушению биосферы и созданию нового типа среды обитания — техносферы. Техносфера — это регион биосферы в прошлом, преобразованный людьми с помощью прямого или косвенного воздействия технических средств с целью наилучшего соответствия своим материальным и социально-экономическим потребностям (регион города или промышленной зоны, производственная или бытовая среда). Практически все урбанизированное население нашей планеты проживает в техносфере, где условия обитания отличаются от биосферных прежде всего повышенным влиянием на человека негативных техногенных и антропогенных факторов.

В условиях техносферы негативные воздействия обусловлены влиянием элементов техносферы (машины, оборудование, сооружения и т.п.) и действиями человека. Вследствие роста объемов антропогенных выбросов и сбросов непрерывно ухудшается качество атмосферного воздуха и гидросферы, повсеместно наблюдается деградация земель. Создание техносферы не исключило, а усугубило воздействие на человека явлений природного характера (землетрясения, наводнения, ураганы и т.д.). Число природных катастроф в последние годы непрерывно растет. Все это привело к тому, что обеспечение безопасности жизнедеятельности стало актуальной социальной задачей.

Безопасность — это свойство системы «человек — среда обитания» сохранять условия взаимодействия с минимальной возможностью возникновения ущерба людским, природным и материальным ресурсам. Безопасность жизнедеятельности может быть обеспечена только при

комфортном (с оптимальными условиями) или допустимом (гарантирующим невозможность возникновения и развития негативных процессов у человека и в среде обитания) состояниях взаимодействия человека со средой. Управление безопасностью жизнедеятельности в Российской Федерации строится на действии многоуровневой системы законодательных, подзаконных и нормативно-правовых актов, а также директивной документации организаций.

Под чрезвычайной ситуацией (ЧС) понимается такое состояние объекта, определенной территории или акватории, при котором в результате возникновения источника чрезвычайной ситуации нарушаются нормальные условия жизни и деятельности людей, возникает угроза их жизни или здоровья, наносится ущерб имуществу населения, экономике и окружающей природной среде. Под источником чрезвычайной ситуации понимают опасное природное явление, аварию или опасное техногенное происшествие, крупномасштабное инфекционное заболевание людей, животных или растений, а также применение современных средств массового поражения, в результате которого произошла или может возникнуть чрезвычайная ситуация.

Причинами возникновения чрезвычайных ситуаций могут быть:

- аварии — чрезвычайные события с техногенными причинами;
- стихийные бедствия — чрезвычайные события природного происхождения;
- катастрофы — аварии и стихийные бедствия, повлекшие за собой многочисленные человеческие жертвы, значительный материальный ущерб или другие тяжелые последствия.

В общем случае в своем развитии чрезвычайные ситуации проходят пять основных фаз:

1. Фаза накопления отклонений объекта от нормального протекания процесса, характеризующаяся продолжительностью, возможностью фиксации отклонения и принятием профилактических мер.

2. Фаза инициирования события (аварии, стихийного бедствия, катастрофы). Эта фаза скоротечна и характеризуется отсутствием времени для осуществления эффективных действий для предотвращения чрезвычайной ситуации.
3. Фаза непосредственного развития и протекания процесса чрезвычайной ситуации. В это время происходит непосредственное воздействие поражающих факторов на людей, объекты и природную среду. Эта фаза носит длительный характер, а в случае промышленной аварии процесс ее протекания определяется не начальными событиями, а структурой производства и используемыми технологиями, что затрудняет прогнозирование развития ЧС.
4. Фаза действия остаточных факторов поражения, в течение которой возможно распространение поражающих факторов за пределы объекта поражения.
5. Фаза ликвидации последствий чрезвычайной ситуации, которая может быть начата в ходе протекания третьей фазы. В ходе ликвидации последствий ЧС могут привлекаться силы и средства организации — объекта поражения, а при широких масштабах поражения — войска гражданской обороны РФ, силы МЧС РФ, воинские подразделения Министерства обороны и т.д.

Концепция гражданской защиты населения предусматривает: защиту населения и территорий и гражданскую оборону. Защита населения, объектов народного хозяйства и окружающей среды (гражданская защита) от действия чрезвычайных ситуаций любого происхождения, а также постоянная готовность к ликвидации их последствий достигается:

- уменьшением возможных масштабов источников аварий, катастроф и стихийных бедствий;
- локализацией и сокращением времени действия существующих поражающих факторов;

- снижением опасности поражения людей путем установления требований к размещению опасных объектов, планированию населенных пунктов, строительству устойчивых зданий и сооружений;
- повышением устойчивости функциональных объектов экономики и жизнеобеспечения;
- проведением аварийно-спасательных и других неотложных работ;
- ликвидацией последствий ЧС и реабилитацией населения, территорий и окружающей среды.

К категории опасных производственных объектов относят производства, на которых:

- получаются, используются, перерабатываются, образуются, хранятся, транспортируются или уничтожаются взрывчатые, окисляющие, воспламеняющиеся, горючие или токсичные вещества;
- используется оборудование, работающее под давлением более 0,7 Мпа или при температуре нагрева воды более 115°С;
- используются стационарно установленные грузоподъемные машины, эскалаторы, канатные дороги, фаникулеры;
- получают расплавы черных и цветных металлов и сплавы на их основе;
- ведутся горные работы, работы по обогащению полезных ископаемых, а также работы в подземных условиях.

Обязательным условием принятия решения о начале строительства, расширения, реконструкции, технического перевооружения, консервации или ликвидации опасного производственного объекта является положительное заключение экспертизы промышленной безопасности проектной документации, утвержденное федеральным органом исполнительной власти, специально уполномоченным в области промышленной безопасности, или его территориальным органом. Технические устройства, в том числе иностранного производства, применяемые на опасном производственном объекте, подлежат сертификации на соответствие требованиям

промышленной безопасности в порядке, установленным законодательством Республики Узбекистан.

Организация, эксплуатирующая опасный производственный объект, обязана: иметь лицензию на право эксплуатации объекта, комплектовать штаты обслуживающего персонала подготовленными и аттестованными работниками, организовывать и осуществлять производственный контроль за соблюдением требований промышленной безопасности, обеспечивать проведение экспертизы промышленной безопасности зданий, освидетельствование технических устройств и сооружения, осуществлять мероприятия по локализации и ликвидации последствий аварий, разрабатывать декларацию промышленной безопасности.

В целях обеспечения готовности к действиям по локализации чрезвычайных ситуаций и ликвидации их последствий организация, эксплуатирующая опасный производственный объект, обязана:

- планировать и осуществлять мероприятия по локализации и ликвидации последствий аварий на опасном производственном объекте;
- заключать с профессиональными аварийно-спасательными службами или с профессиональными аварийно-спасательными формированиями договоры на обслуживание, а в случаях, предусмотренных законодательством Российской Федерации, создавать собственные профессиональные аварийно-спасательные службы или формирования, а также нештатные аварийно-спасательные формирования из числа собственных работников;
- иметь резервы финансовых средств и материальных ресурсов для локализации и ликвидации последствий аварий в соответствии с законодательством Российской Федерации;
- обучать персонал действиям в случае возникновения аварии или инцидента на опасном производственном объекте;

- создавать системы наблюдения, оповещения, связи и поддержки действий в случае аварии и поддерживать указанные системы в пригодном к использованию состоянии.

Организация, эксплуатирующая опасный производственный объект, обязана страховать ответственность за причинение вреда жизни или имуществу других лиц и окружающей природной среде в случае возникновения аварии на опасном производственном объекте [7].

Заключение

В ходе выполнения выпускной квалификационной работы были достигнуты следующие результаты:

1. Были изучены методы обработки речевых сигналов, а также способы извлечения основных параметров и характеристик речевого сигнала.

2. Также рассмотрены способы и технологии применения алгоритмов для распознавания речи.

3. Произведен сравнительный анализ существующих коммерческих и бесплатных систем распознавания речи.

4. На основе вышеуказанного анализа была выбрана, а далее изучена система распознавания речи CMU Sphinx.

5. Были разработаны словарь, акустическая и лингвистическая модели узбекского языка.

6. Разработана программа распознавания узбекской речи, обеспечивающая перевод речи в текст.

Технологии распознавания речи считаются одними из наиболее перспективных в мире. По мере развития компьютерных систем становится все более очевидным, что использование этих систем намного расширится, если станет возможным использование человеческой речи при работе непосредственно с компьютером, и в частности станет возможным управление машиной обычным голосом в реальном времени, а также ввод и вывод информации в виде обычной человеческой речи.

Значение созданной программы состоит в том, что она дает возможность переводить произнесенную узбекскую речь в текстовый вид, которой аналогов на сегодняшний день, можно сказать, не существует. Такая возможность позволяет в значительной степени сократить время ввода данных. Данную технологию распознавания речи в дальнейшем можно использовать в таких сферах как военное дело, медицина, робототехника, информационные технологии, а также стенография.

Список использованной литературы.

1. *Каримов И.А*. Мировой финансово-экономический кризис, пути и меры по его преодолению в условиях Узбекистана – Ташкент: Узбекистан, март 2009.
2. *Сергиенко А. Б.* Цифровая обработка сигналов. — 2-е. — Спб: Питер, 2006. — С. 751. — ISBN 5-469-00816-9
3. *J. M. Ponte and W. B. Croft* (1998). "A Language Modeling Approach to Information Retrieval". Research and Development in Information Retrieval. pp. 275–281.
4. *F. Song and W. B. Croft* (1999). "A General Language Model for Information Retrieval". Research and Development in Information Retrieval. pp. 279–280.
5. *М. А. Павлейно, В. М. Ромаданов* Спектральные преобразования в MatLab. — СПб: 2007. — С. 160. — ISBN 978-5-98340-121-1
6. Защита объектов народного хозяйства от оружия массового поражения: Справочник / *Г. П. Демиденко, Е. П. Кузьменко и др.* Под ред. Г. П. Демиденко. – 2-е издание перераб. и доп. – К.: Вышая школа, 1989.–287 с.
7. *Миценко І. М.* Забезпечення життєдіяльності людини в навколишньому середовищі. – Кіровоград, 1998.–292 с.
8. http://en.wikipedia.org/wiki/CMU_Sphinx
9. http://cmusphinx.sourceforge.net/sphinx4/.
10. http://en.wikipedia.org/wiki/Acoustic_Model
11. http://www.speechpro.ru/techno/recognition
12. http://en.wikipedia.org/wiki/List_of_speech_recognition_software
13. http://www.speech.cs.cmu.edu

Приложение

//Основной класс Java-приложения

```
package demo.sphinx.hellongram_ru;

import edu.cmu.sphinx.frontend.util.Microphone;

import edu.cmu.sphinx.recognizer.Recognizer;

import edu.cmu.sphinx.result.Result;

import edu.cmu.sphinx.util.props.ConfigurationManager;

import edu.cmu.sphinx.util.props.PropertyException;

import java.io.BufferedReader;

import java.io.File;

import java.io.IOException;

import java.io.InputStreamReader;

import java.net.URL;

import org.jivesoftware.smack.XMPPConnection;

import org.jivesoftware.smack.XMPPException;

public class HelloNGram {

   public static void main(String[] args) {

      try {           URL url;

         if (args.length > 0) {

            url = new File(args[0]).toURI().toURL();  } else {

            url = HelloNGram.class.getResource("uzbek_config.xml");

         }

         System.out.println("Загурзка...");

         ConfigurationManager cm = new ConfigurationManager(url);

         Recognizer recognizer = (Recognizer) cm.lookup("recognizer");
```

```
    Microphone microphone = (Microphone) cm.lookup("microphone");
    recognizer.allocate();
    Client cl = new Client();
    try {           cl = Connect();
                    } catch (XMPPException e) {
                            e.printStackTrace();                    }
    if (microphone.startRecording()) {
       while (true) {    System.out.println
                ("Говорить. Нажмите Ctrl-C для выхода.\n");
          Result result = recognizer.recognize();
          if (result != null) {
             String resultText = result.getBestResultNoFiller();
             System.out.println("You said: " + resultText + "\n");
          } else {
             System.out.println("Не возможно распознать.\n");
          }             }          } else {
       System.out.println("Невозможно связаться с микрофоном.");
       recognizer.deallocate();
       System.exit(1);
    }       } catch (IOException e) {
    System.err.println("Problem when loading HelloNGram: " + e);
    e.printStackTrace();
} catch (PropertyException e) {
    System.err.println("Problem configuring HelloNGram: " + e);
    e.printStackTrace();
```

```
    }    }
private static Client Connect() throws XMPPException{
    Client c = new Client();
    XMPPConnection.DEBUG_ENABLED = false;
    c.login("islamgv13", "13GreatPeople05");
    c.displayBuddyList();
    return c;            }}
```

//Класс буферизации результата

```
package copytext;
import java.awt.Robot; import java.awt.Toolkit;
import java.awt.datatransfer.Clipboard;
import java.awt.datatransfer.ClipboardOwner;
import java.awt.datatransfer.StringSelection;
import java.awt.datatransfer.Transferable; import java.awt.event.KeyEvent;

public class CopyText implements ClipboardOwner {
   public static void main(String[] args) {
      CopyText ct=new CopyText();  ct.setstring(Result);
      ct.sendKey(KeyEvent.VK_CONTROL, KeyEvent.VK_V);    }
   public void setstring(String data){
      StringSelection ss=new StringSelection(data);
      Clipboard cb=Toolkit.getDefaultToolkit().getSystemClipboard();
      Transferable td = new StringSelection(data);
      Toolkit.getDefaultToolkit().getSystemClipboard().setContents(td, this);    }
   public void sendKey(int ctrlValue, int keyValue){
      Robot robot = null;       try{          robot=new Robot();       }
      catch(Exception ex){          System.out.print(ex.ToString());       }
      robot.setAutoWaitForIdle(true);
      if (ctrlValue != -1) robot.keyPress(ctrlValue);
      robot.keyPress(keyValue);  robot.keyRelease(keyValue);
      if (ctrlValue != -1) robot.keyRelease(ctrlValue);    }}
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!--   Sphinx-4 Configuration file-->
<!--  biship  configuration file                    -->
<config>
   <!-- frequently tuned properties                    -->
   <property name="absoluteBeamWidth"  value="-1"/>
   <!--<property name="relativeBeamWidth"  value="1E-80"/>-->
   <property name="relativeBeamWidth"  value="1E-70"/>
   <property name="absoluteWordBeamWidth" value="-1"/>
   <property name="relativeWordBeamWidth" value="1E-40"/>
   <property name="wordInsertionProbability" value="0.2"/>
   <property name="languageWeight" value="10.0"/>
   <property name="silenceInsertionProbability" value=".1"/>
   <property name="frontend" value="epFrontEnd"/>
   <property name="recognizer" value="recognizer"/>
   <property name="showCreations" value="false"/>
   <!-- word recognizer configuration                    -->
   <component name="recognizer"
type="edu.cmu.sphinx.recognizer.Recognizer">
      <property name="decoder" value="decoder"/>
   </component>
   <!-- The Decoder   configuration                    -->
   <component name="decoder" type="edu.cmu.sphinx.decoder.Decoder">
      <property name="searchManager" value="wordPruningSearchManager"/>
   </component>
```

```
    <!-- The Search Manager                          -->
    <component name="wordPruningSearchManager"
type="edu.cmu.sphinx.decoder.search.WordPruningBreadthFirstSearchManager">
        <property name="logMath" value="logMath"/>
        <property name="linguist" value="lexTreeLinguist"/>
        <property name="pruner" value="trivialPruner"/>
        <property name="scorer" value="threadedScorer"/>
        <property name="activeListManager" value="activeListManager"/>
        <property name="growSkipInterval" value="0"/>
        <property name="checkStateOrder" value="false"/>
        <property name="acousticLookaheadFrames" value="1.7"/>
        <property name="relativeBeamWidth" value="${relativeBeamWidth}"/>
          <property name="keepAllTokens" value="false"/>
    </component>
    <component name="activeList"
        type="edu.cmu.sphinx.decoder.search.PartitionActiveListFactory">
          <property name="logMath" value="logMath"/>
          <property name="absoluteBeamWidth" value="${absoluteBeamWidth}"/>
          <property name="relativeBeamWidth" value="${relativeBeamWidth}"/>
    </component>
    <!-- The Active Lists                           -->
    <component name="activeListManager"
            type="edu.cmu.sphinx.decoder.search.SimpleActiveListManager">
        <propertylist name="activeListFactories">
             <item>standardActiveListFactory</item>
```

```
            <item>wordActiveListFactory</item>
            <item>wordActiveListFactory</item>
            <item>standardActiveListFactory</item>
            <item>standardActiveListFactory</item>
            <item>standardActiveListFactory</item>
        </propertylist>    </component>
        <component name="standardActiveListFactory"
            type="edu.cmu.sphinx.decoder.search.PartitionActiveListFactory">
        <property name="logMath" value="logMath"/>
        <property name="absoluteBeamWidth" value="${absoluteBeamWidth}"/>
        <property name="relativeBeamWidth" value="${relativeBeamWidth}"/>
    </component>
        <component name="wordActiveListFactory"
            type="edu.cmu.sphinx.decoder.search.PartitionActiveListFactory">
        <property name="logMath" value="logMath"/>
        <property name="absoluteBeamWidth"
value="${absoluteWordBeamWidth}"/>
        <property name="relativeBeamWidth"
value="${relativeWordBeamWidth}"/>
    </component>
        <!-- The Pruner                                    -->
    <component name="trivialPruner"
            type="edu.cmu.sphinx.decoder.pruner.SimplePruner"/>
    <!-- TheScorer                                      -->
    <component name="threadedScorer"
```

```
        type="edu.cmu.sphinx.decoder.scorer.ThreadedAcousticScorer">
    <property name="frontend" value="${frontend}"/>
    <property name="isCpuRelative" value="true"/>
    <property name="numThreads" value="0"/>
    <property name="minScoreablesPerThread" value="10"/>
    <property name="scoreablesKeepFeature" value="true"/>
</component>
<!-- The linguist  configuration                    -->
<component name="lexTreeLinguist"
        type="edu.cmu.sphinx.linguist.lextree.LexTreeLinguist">
    <property name="logMath" value="logMath"/>
    <property name="acousticModel" value="uzlang"/>
    <property name="languageModel" value="trigramModel"/>
    <property name="dictionary" value="dictionary"/>
    <property name="addFillerWords" value="false"/>
    <property name="fillerInsertionProbability" value="1E-10"/>
    <property name="generateUnitStates" value="false"/>
    <property name="wordInsertionProbability"
        value="${wordInsertionProbability}"/>
    <property name="silenceInsertionProbability"
        value="${silenceInsertionProbability}"/>
    <property name="languageWeight" value="${languageWeight}"/>
    <property name="unitManager" value="unitManager"/>
</component>
<!-- The Dictionary configuration                    -->
```

```
    <component name="dictionary"
        type="edu.cmu.sphinx.linguist.dictionary.FastDictionary">
        <property name="dictionaryPath"

          value="resource:/demo.sphinx.hellongram_ru.HelloNGram!/demo/sphinx/hellongram_ru/uzlang.dic"/>
          <property name="fillerPath"

          value="resource:/edu.cmu.sphinx.model.acoustic.uzlang_8gau_13dCep_16k_40mel_130Hz_6800Hz.Model!/edu/cmu/sphinx/model/acoustic/uzlang_8gau_13dCep_16k_40mel_130Hz_6800Hz/dict/uzlang.filler"/>
        <property name="addSilEndingPronunciation" value="false"/>
        <property name="wordReplacement" value="&lt;sil&gt;"/>
        <property name="unitManager" value="unitManager"/>
    </component>
    <!-- The Language Model configuration                       -->
    <component name="trigramModel"
        type="edu.cmu.sphinx.linguist.language.ngram.SimpleNGramModel">
        <property name="location"

          value="resource:/demo.sphinx.hellongram_ru.HelloNGram!/demo/sphinx/hellongram_ru/uzlang.lm"/>
        <property name="logMath" value="logMath"/>
        <property name="dictionary" value="dictionary"/>
        <property name="maxDepth" value="3"/>
        <property name="unigramWeight" value=".7"/>
```

```
    </component>
    <!-- The acoustic model configuration                    -->
    <component name="uzlang"
type="edu.cmu.sphinx.model.acoustic.uzlang_8gau_13dCep_16k_40mel_130Hz_
6800Hz.Model">
        <property name="loader" value="uzlangLoader"/>
        <property name="unitManager" value="unitManager"/>
    </component>
        <component name="uzlangLoader"
type="edu.cmu.sphinx.model.acoustic.uzlang_8gau_13dCep_16k_40mel_130Hz_
6800Hz.ModelLoader">
        <property name="logMath" value="logMath"/>
        <property name="unitManager" value="unitManager"/>
    </component>
    <!-- The unit manager configuration                      -->
    <component name="unitManager"
            type="edu.cmu.sphinx.linguist.acoustic.UnitManager"/>
    <!-- The live frontend configuration                     -->
    <component name="epFrontEnd" type="edu.cmu.sphinx.frontend.FrontEnd">
        <propertylist name="pipeline">
            <item>microphone </item>
            <item>dataBlocker </item>
            <item>speechClassifier </item>
            <item>speechMarker </item>
            <item>nonSpeechDataFilter </item>
```

```
        <item>premphasizer </item> <item>windower </item>
        <item>fft </item><item>melFilterBank </item>
        <item>dct </item> <item>liveCMN </item>
        <item>featureExtraction </item>
      </propertylist>    </component>
   <component name="dataBlocker"
type="edu.cmu.sphinx.frontend.DataBlocker"/>
   <component name="speechClassifier"
          type="edu.cmu.sphinx.frontend.endpoint.SpeechClassifier">
      <property name="threshold" value="13"/>
   </component>
   <component name="nonSpeechDataFilter"
          type="edu.cmu.sphinx.frontend.endpoint.NonSpeechDataFilter"/>
   <component name="speechMarker"
          type="edu.cmu.sphinx.frontend.endpoint.SpeechMarker">
      <property name="speechTrailer" value="50"/>
   </component>
   <component name="premphasizer"
             type="edu.cmu.sphinx.frontend.filter.Preemphasizer"/>
   <component name="windower"
       type="edu.cmu.sphinx.frontend.window.RaisedCosineWindower"/>
   <component name="fft"
      type="edu.cmu.sphinx.frontend.transform.DiscreteFourierTransform"/>
   <component name="melFilterBank"
      type="edu.cmu.sphinx.frontend.frequencywarp.MelFrequencyFilterBank"/>
```

```
    <component name="dct"
            type="edu.cmu.sphinx.frontend.transform.DiscreteCosineTransform"/>
    <component name="featureExtraction"
                    type="edu.cmu.sphinx.frontend.feature.DeltasFeatureExtractor"/>
        <component name="liveCMN"
                type="edu.cmu.sphinx.frontend.feature.LiveCMN"/>
        <component name="microphone"
                type="edu.cmu.sphinx.frontend.util.Microphone">
       <property name="closeBetweenUtterances" value="false"/>
    </component>
    <!--  Miscellaneous components                          -->
    <component name="logMath" type="edu.cmu.sphinx.util.LogMath">
       <property name="logBase" value="1.0001"/>
       <property name="useAddTable" value="true"/>
    </component>
</config>
```

```
using System; using System.Data; using System.IO;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;

namespace uzSpeech
{
        public class frmNotepad : System.Windows.Forms.Form
        {

      private bool vis = false;
                public frmNotepad()
                {       InitializeComponent();    }

                private void InitializeComponent()
                {
          this.components = new System.ComponentModel.Container();
          System.ComponentModel.ComponentResourceManager resources = new
System.ComponentModel.ComponentResourceManager(typeof(frmNotepad));
          this.mainMenuNotepad = new
System.Windows.Forms.MainMenu(this.components);
          this.menuItemFile = new System.Windows.Forms.MenuItem();
          this.menuItemNew = new System.Windows.Forms.MenuItem();
          this.menuItemOpen = new System.Windows.Forms.MenuItem();
```

```
this.menuItemSeparator1 = new System.Windows.Forms.MenuItem();
this.menuItemSave = new System.Windows.Forms.MenuItem();
this.menuItemSaveAs = new System.Windows.Forms.MenuItem();
this.menuItem7 = new System.Windows.Forms.MenuItem();
this.menuItem6 = new System.Windows.Forms.MenuItem();
this.menuItemEdit = new System.Windows.Forms.MenuItem();
this.menuItemCut = new System.Windows.Forms.MenuItem();
this.menuItemCopy = new System.Windows.Forms.MenuItem();
this.menuItemPast = new System.Windows.Forms.MenuItem();
this.menuItemSelectAll = new System.Windows.Forms.MenuItem();
this.menuItemFormat = new System.Windows.Forms.MenuItem();
this.menuItemFont = new System.Windows.Forms.MenuItem();
this.menuItem1 = new System.Windows.Forms.MenuItem();
this.menuItem2 = new System.Windows.Forms.MenuItem();
this.menuItem8 = new System.Windows.Forms.MenuItem();
this.menuItem9 = new System.Windows.Forms.MenuItem();
this.menuItem3 = new System.Windows.Forms.MenuItem();
this.menuItem4 = new System.Windows.Forms.MenuItem();
this.menuItem5 = new System.Windows.Forms.MenuItem();
this.txtBody = new System.Windows.Forms.TextBox();
this.contextMenuTxtBody = new System.Windows.Forms.ContextMenu();
this.ContextFont = new System.Windows.Forms.MenuItem();
this.ContextSeparator2 = new System.Windows.Forms.MenuItem();
this.ContextCut = new System.Windows.Forms.MenuItem();
this.ContextCopy = new System.Windows.Forms.MenuItem();
```

```
            this.ContextPaste = new System.Windows.Forms.MenuItem();
            this.ContextSeparator1 = new System.Windows.Forms.MenuItem();
            this.ContextSelectAll = new System.Windows.Forms.MenuItem();
            this.dlgOpenFile = new System.Windows.Forms.OpenFileDialog();
            this.dlgSaveFile = new System.Windows.Forms.SaveFileDialog();
            this.dlgFont = new System.Windows.Forms.FontDialog();
            this.axAgent1 = new AxAgentObjects.AxAgent();
            ((System.ComponentModel.ISupportInitialize)(this.axAgent1)).BeginInit();
            this.SuspendLayout();
            this.mainMenuNotepad.MenuItems.AddRange(new
System.Windows.Forms.MenuItem[] {
            this.menuItemFile,  this.menuItemEdit,   this.menuItemFormat,
            this.menuItem1,  this.menuItem3});
            this.menuItemFile.Index = 0;
            this.menuItemFile.MenuItems.AddRange(new
System.Windows.Forms.MenuItem[] {
            this.menuItemNew,  this.menuItemOpen,
            this.menuItemSeparator1,  this.menuItemSave,
            this.menuItemSaveAs,  this.menuItem7,
            this.menuItem6});  this.menuItemFile.Text = "Файл";
            this.menuItemNew.Index = 0;
            this.menuItemNew.Shortcut = System.Windows.Forms.Shortcut.CtrlN;
            this.menuItemNew.Text = "&Создать...";
            this.menuItemNew.Click += new
System.EventHandler(this.menuItemNew_Click);
```

```
        this.menuItemOpen.Index = 1;
        this.menuItemOpen.Shortcut = System.Windows.Forms.Shortcut.CtrlO;
        this.menuItemOpen.Text = "&Открыть...";
        this.menuItemOpen.Click += new
System.EventHandler(this.menuItemOpen_Click);
        this.menuItemSeparator1.Index = 2;
        this.menuItemSeparator1.Text = "-";
        this.menuItemSave.Enabled = false;
        this.menuItemSave.Index = 3;
        this.menuItemSave.Shortcut = System.Windows.Forms.Shortcut.CtrlS;
        this.menuItemSave.Text = "С&охранить";
        this.menuItemSave.Click += new
System.EventHandler(this.menuItemSave_Click);
        this.menuItemSaveAs.Index = 4;
        this.menuItemSaveAs.Text = "Сохранить как...";
        this.menuItemSaveAs.Click += new
System.EventHandler(this.menuItemSaveAs_Click);
        this.menuItem7.Index = 5; this.menuItem7.Text = "-";
        this.menuItem6.Index = 6;
        this.menuItem6.Text = "Выход";
        this.menuItem6.Click += new
System.EventHandler(this.menuItem6_Click);
        this.menuItemEdit.Index = 1;
        this.menuItemEdit.MenuItems.AddRange(new
System.Windows.Forms.MenuItem[] {
        this.menuItemCut,  this.menuItemCopy,
```

```
            this.menuItemPast, this.menuItemSelectAll});
            this.menuItemEdit.Text = "Правка";
            this.menuItemCut.Index = 0;
            this.menuItemCut.Shortcut = System.Windows.Forms.Shortcut.CtrlX;
            this.menuItemCut.Text = "&Вырезать";
            this.menuItemCut.Click += new
System.EventHandler(this.menuItemCut_Click);
            this.menuItemCopy.Index = 1;
            this.menuItemCopy.Shortcut = System.Windows.Forms.Shortcut.CtrlC;
            this.menuItemCopy.Text = "&Копировать";
            this.menuItemCopy.Click += new
System.EventHandler(this.menuItemCopy_Click);
            this.menuItemPast.Index = 2;
            this.menuItemPast.Shortcut = System.Windows.Forms.Shortcut.CtrlV;
            this.menuItemPast.Text = "В&ставить";
            this.menuItemPast.Click += new
System.EventHandler(this.menuItemPast_Click);
            this.menuItemSelectAll.Index = 3;
            this.menuItemSelectAll.Shortcut =
System.Windows.Forms.Shortcut.CtrlA;
            this.menuItemSelectAll.Text = "В&ыделить все";
            this.menuItemSelectAll.Click += new
System.EventHandler(this.menuItemSelectAll_Click);
            this.menuItemFormat.Index = 2;
            this.menuItemFormat.MenuItems.AddRange(new
System.Windows.Forms.MenuItem[] {
```

```
            this.menuItemFont});
            this.menuItemFormat.Text = "Формат";
            this.menuItemFont.Index = 0;
            this.menuItemFont.Text = "Шрифт...";
            this.menuItemFont.Click += new
System.EventHandler(this.menuItemFont_Click);
            this.menuItem1.Index = 3;
            this.menuItem1.MenuItems.AddRange(new
System.Windows.Forms.MenuItem[] {
            this.menuItem2,
            this.menuItem8,
            this.menuItem9});
            this.menuItem1.Text = "Вид";
            this.menuItem2.Index = 0;
            this.menuItem2.Text = "Строка состояния";
            this.menuItem8.Index = 1;
            this.menuItem8.Text = "График входного звука";
            this.menuItem8.Click += new
System.EventHandler(this.menuItem8_Click);
            this.menuItem9.Index = 2;
            this.menuItem9.Text = "Агент";
            this.menuItem9.Click += new
System.EventHandler(this.menuItem9_Click);
            this.menuItem3.Index = 4;
            this.menuItem3.MenuItems.AddRange(new
System.Windows.Forms.MenuItem[] {
```

```
            this.menuItem4,
            this.menuItem5});
            this.menuItem3.Text = "Справка";
            this.menuItem4.Index = 0;
            this.menuItem4.Text = "Помощь";
            this.menuItem5.Index = 1;
            this.menuItem5.Text = "О программе";
            this.txtBody.ContextMenu = this.contextMenuTxtBody;
            this.txtBody.Dock = System.Windows.Forms.DockStyle.Fill;
            this.txtBody.Font = new System.Drawing.Font("Microsoft Sans Serif",
12F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
((byte)(0)));
            this.txtBody.Location = new System.Drawing.Point(0, 0);
            this.txtBody.Multiline = true;
            this.txtBody.Name = "txtBody";
            this.txtBody.ScrollBars = System.Windows.Forms.ScrollBars.Vertical;
            this.txtBody.Size = new System.Drawing.Size(480, 381);
            this.txtBody.TabIndex = 0;
            this.txtBody.TextChanged += new
System.EventHandler(this.txtBody_TextChanged);
            this.txtBody.KeyDown += new
System.Windows.Forms.KeyEventHandler(this.txtBody_KeyDown);
            this.contextMenuTxtBody.MenuItems.AddRange(new
System.Windows.Forms.MenuItem[] {
            this.ContextFont,
            this.ContextSeparator2,
```

```
            this.ContextCut,
            this.ContextCopy,
            this.ContextPaste,
            this.ContextSeparator1,
            this.ContextSelectAll});
            this.ContextFont.Index = 0;
            this.ContextFont.Text = "Шрифт...";
            this.ContextFont.Click += new
System.EventHandler(this.menuItemFont_Click);
            this.ContextSeparator2.Index = 1;
            this.ContextSeparator2.Text = "-";
            this.ContextCut.Index = 2;
            this.ContextCut.Text = "Вырезвать";
            this.ContextCut.Click += new
System.EventHandler(this.menuItemCut_Click);
            this.ContextCopy.Index = 3;
            this.ContextCopy.Text = "Копировать";
            this.ContextCopy.Click += new
System.EventHandler(this.menuItemCopy_Click);
            this.ContextPaste.Index = 4;
            this.ContextPaste.Text = "Вставить";
            this.ContextPaste.Click += new
System.EventHandler(this.menuItemPast_Click);
            this.ContextSeparator1.Index = 5;
            this.ContextSeparator1.Text = "-";
            this.ContextSelectAll.Index = 6;
```

```
            this.ContextSelectAll.Text = "Выделить все";
            this.ContextSelectAll.Click += new
System.EventHandler(this.menuItemSelectAll_Click);
            this.dlgSaveFile.FileName = "doc1";
            this.dlgFont.FontMustExist = true;
            this.dlgFont.ScriptsOnly = true;
            this.axAgent1.Enabled = true;
            this.axAgent1.Location = new System.Drawing.Point(122, 93);
            this.axAgent1.Name = "axAgent1";
            this.axAgent1.OcxState =
((System.Windows.Forms.AxHost.State)(resources.GetObject("axAgent1.OcxStat
e")));
            this.axAgent1.Size = new System.Drawing.Size(32, 32);
            this.axAgent1.TabIndex = 0;
            this.axAgent1.TabStop = false;
            this.axAgent1.Visible = false;
            this.AutoScaleBaseSize = new System.Drawing.Size(5, 13);
            this.ClientSize = new System.Drawing.Size(480, 381);
            this.Controls.Add(this.txtBody);
            this.Controls.Add(this.axAgent1);
            this.Icon = ((System.Drawing.Icon)(resources.GetObject("$this.Icon")));
            this.Menu = this.mainMenuNotepad;
            this.Name = "frmNotepad";
            this.Text = "uzSpeech";
            this.Load += new System.EventHandler(this.frmNotepad_Load);
```

```
((System.ComponentModel.ISupportInitialize)(this.axAgent1)).EndInit();
this.ResumeLayout(false);   this.PerformLayout();                }
        [STAThread]
        static void Main()
        {                       Application.Run(new frmNotepad());          }
    private void txtBody_TextChanged(object sender, System.EventArgs e)
        {m_bModified = true; menuItemSave.Enabled = true;}
        private void CheckChanged(object sender, System.EventArgs e)
        {       if(m_bModified)           {
dlgResult = MessageBox.Show("Сохранить документ?",
"uzSpeech",MessageBoxButtons.YesNo,MessageBoxIcon.Exclamation);
                        if(dlgResult == DialogResult.Yes)
                        {       menuItemSave_Click(sender,e);        }
                }       }
        private void menuItemNew_Click(object sender, System.EventArgs e)
        {
                CheckChanged(sender,e);
                m_strFileName = "";txtBody.Clear();
                m_bModified = false;
        }

private void menuItemSaveAs_Click(object sender, System.EventArgs e)
        {       dlgSaveFile.FileName = "mydoc" +
m_intDocNumber.ToString() + ".txt";
                dlgResult = dlgSaveFile.ShowDialog();
```

```
                if(dlgResult == DialogResult.Cancel)          return;
                try
                {       m_strFileName = dlgSaveFile.FileName;
                        m_sw = new StreamWriter(m_strFileName);
                        m_sw.Write (txtBody.Text);     m_sw.Close();
                        menuItemSave.Enabled = true;
                        m_intDocNumber++;      m_bModified = false;
                        this.Text = "uzSpeech: " + m_strFileName;
                }       catch(Exception err)       {
        MessageBox.Show(err.Message,"Ошибка",MessageBoxButtons.OK,Messa
geBoxIcon.Error); }}
private void menuItemSave_Click(object sender, System.EventArgs e)
                {
                        if(m_strFileName == "")
                        {       menuItemSaveAs_Click(sender,e);
                                return;}
                        m_sw = new StreamWriter(m_strFileName);
                        m_sw.Write (txtBody.Text);
                        m_sw.Close();m_bModified = false;
                }
        private void menuItemOpen_Click(object sender, System.EventArgs e)
                {
                        CheckChanged(sender,e);
                        dlgResult = dlgOpenFile.ShowDialog();
                        if(dlgResult == DialogResult.Cancel) return;
```

```
                try
                {
                        m_strFileName = dlgOpenFile.FileName;
                        StreamReader sr = new StreamReader(m_strFileName);
                        txtBody.Text = sr.ReadToEnd();
                        sr.Close();    m_bModified = false;
                }
                catch(Exception err)
                {
        MessageBox.Show(err.Message,"Ошибка",MessageBoxButtons.OK,Messa
geBoxIcon.Error);
                }}
            private void menuItemCut_Click(object sender, System.EventArgs e)
            {
                    Clipboard.SetDataObject(txtBody.SelectedText);
                    txtBody.SelectedText = "";}
private void menuItemCopy_Click(object sender, System.EventArgs e)
            {       txtBody.Copy();    }
            private void menuItemPast_Click(object sender, System.EventArgs e)
            {       txtBody.Paste();    }

private void menuItemSelectAll_Click(object sender, System.EventArgs e)
            {txtBody.SelectAll();}
            private void menuItemFont_Click(object sender, System.EventArgs e)
```

```
            {
                dlgResult = dlgFont.ShowDialog();
                if(dlgResult == DialogResult.Cancel)          return;
                FontFamily fmFontName = dlgFont.Font.FontFamily;
                float fFontSize = dlgFont.Font.Size;
                FontStyle fsStyle = dlgFont.Font.Style;
                Font font;    try     {font = new
Font(fmFontName,fFontSize,txtBody.Font.Style ^ fsStyle);
                    txtBody.Font = font;       }
                catch(ArgumentException)
                {       return;}
            }
    private void menuItem6_Click(object sender, EventArgs e)
    {          Application.Exit();       }

    private void menuItem9_Click(object sender, EventArgs e)
    {
       if (!menuItem9.Checked&&!vis)
       {
          agent1.Left = (short)this.Left; agent1.Top = (short)this.Top;
          menuItem9.Checked = true; agent1.Show(false);
          agent1.Play("Greet");  vis = true;  }
       else      {
          menuItem9.Checked = false;
          agent1.Play("Wave");
```

```
        agent1.Hide(false); vis = false;  }  }
private void menuItem8_Click(object sender, EventArgs e)
{
    SoundCatcher.FormMain f = new SoundCatcher.FormMain(); f.Show();
}
private void txtBody_KeyDown(object sender, KeyEventArgs e)
{
    string[] coms = null;
    if (e.KeyCode == Keys.Enter)
    {
        coms = txtBody.Text.Split("\n".ToCharArray());
       AgentPlay(coms[coms.Length - 1].ToUpper());
    }  }
private void AgentPlay(string command)
{
    object sender = null; EventArgs e = null;
    switch (command)
    {
        case "SALOM":
            {  menuItem9_Click(sender,e); vis = true;  break; }
        case "XAYR":
            {menuItem9_Click(sender, e); vis = false; break; }
        case "YOZ": {agent1.Play("Write"); break; }
        case "O'QI": {  agent1.Play("Read"); break; }
        case "ISHLA": {  agent1.Play("Process");  break; }
```

```
        case "QIDIR": { agent1.Play("Search"); break;  }
        case "TUR": {agent1.Play("Idle1_1");break; }
        case "HA":  {  agent1.Play("Acknowledge"); break;    }
        case "KO'Z": {  agent1.Play("Blink"); break; }
        case "TOMI KETGAN":  { agent1.Play("Confused");break;  }
        case "HAFA":{ agent1.Play("Sad"); break; }
        case "XAYRON":     { agent1.Play("Surprised"); break; }
       case "GRAFIK": {   menuItem8_Click(sender, e);  break; }
        default: { agent1.Think(command); break; }  }  }
private void frmNotepad_Load(object sender, EventArgs e)
{
   axAgent1.Characters.Load("agentAssistant", "merlin.acs");
   agent1 = axAgent1.Characters.Character("agentAssistant");
}}}
```