МИНИСТЕРСТВО ПО РАЗВИТИЮ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И КОММУНИКАЦИЙ РЕСПУБЛИКИ УЗБЕКИСТАН

НУКУССКИЙ ФИЛИАЛ ТАШКЕНТСКОГО УНИВЕРСИТЕТА ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

ФАКУЛЬТЕТ: Компьютерный инжиниринг

КУРСОВАЯ РАБОТА

По предмету: «Разработка программного обеспечения встроенных систем»

На тему: «Распознавания речи»

Выполнила: **Кадирбергенов Н.**

Принял: **Татлымуратов Н.**

Содержание

Введение

В настоящее время все острее встает необходимость создать высокоточные инструменты работы для ЭВМ с аудиопотоком. Места применения таких систем можно найти повсеместно. Цели так же могут быть различными от военных и инженерных до социальных и личных. Одним из наиболее успешных инструментов в настоящее время являются Скрытые Марковские Модели (СММ). Не смотря на то, что данная работа посвящена именно этому методу, в ней так же будут упомянуты другие методы работы с аудиопотоком, с целью их сравнения.

Предполагалось, что когда компьютер научится понимать человеческую речь, мы быстро сможем создать искусственный интеллект. Но точность систем распознавания речи достигла своего пика в 1999 году и с тех пор застыла на месте. Академические тесты 2006 года констатируют факт: системы общего профиля так и не преодолели уровень 80%, тогда как у человека этот показатель составляет 96-98%.

Сложность задачи можно себе представить. По некоторым оценкам, количество возможных предложений в человеческом языке составляет 10570. В документированных источниках зафиксирована лишь малая их часть, так что систему невозможно научить, даже если «скормить» ей все тексты, созданные людьми.

У многих слов в языке — сотни или тысячи значений. Выбор конкретного значения зависит от контекста, то есть от окружающих слов. В устной речи он ещё зависит от выражения лица или от интонации.

Наш мозг способен генерировать текст совершенно произвольно, используя интуитивно понятные правила функциональной грамматики и усвоенную с возрастом семантическую парадигму каждого слова. Эти правила описывают, какие слова могут сочетаться друг с другом и каким образом (через какие функциональные элементы). Значение каждого слова зависит от значения предыдущего слова, а в сложных случаях наш мозг распознаёт речь лишь по обрывкам фраз, зная контекст.

Базовые правила функциональной грамматики понятны каждому человеку, но их никак не удаётся формализовать, чтобы стало понятно и компьютеру. А без этого никак. Когда компьютер пытается распознать ранее не встречавшиеся ему предложения, он неизбежно будет допускать ошибки в распознавании, если у него нет грамматического парсера и словаря с семантическими парадигмами, встроенного в человеческий мозг.

Например, российские лингвисты когда-то попытались составить семантическую парадигму одного простого предлога русского языка (кажется, ПРИ). Они дошли до нескольких сотен значений, каждое из которых допускает свой набор последующих элементов. И это был явно не полный список.

По грамматике предлогов проводятся целые научные конференции (некоторые учёные всю жизнь изучают предлог ПО и не могут до конца раскрыть его тайны). А ведь подобное описание требуется для каждой морфемы человеческого языка, включая приставки и суффиксы. Только после этого можно будет приступить к программированию компьютерных систем распознавания речи. По силам ли человечеству эта задача? Ведь нужно учесть ещё, что парадигма каждого элемента человеческой речи постоянно меняется, ведь язык живёт своей жизнью и всё время эволюционирует. Как компьютерная система сможет самообучаться?

Самый поверхностный анализ опубликованных текстов в интернете компанией Google позволил выявить триллион объектов. Это лишь мизерная часть морфем, из которых состоит наша речь. Google выложил 24-гигабайтный архив с текстами во всеобщий доступ и прекратил дальнейшие публикации по этой теме.

Проект MindNet по созданию «универсального парсера» компания Microsoft начала в 1991 году. Они пытались построить универсальную карту всех возможных взаимосвязей между словами. На проект потратили много сил и финансовых средств, но были вынуждены практически прекратить исследования в 2005 году.

Можно поставить точку и начинать всё сначала, только другим способом (гораздо более сложным). Язык необходимо формализовать в рамках единой

функциональной грамматики, универсальной для всех языков, и без серьёзной помощи лингвистов тут не обойтись, если задача вообще решаема.

Профессор Роберт Фортнер из Media Research Institute считает, что создатели систем распознавания речи окончательно зашли в тупик. Программисты сделали всё что смогли, и у них не получилось. Спустя несколько десятилетий они поняли, что человеческая речь — не просто набор звуков. Акустический сигнал не несёт достаточно информации для распознавания текста.

Недостатки, имеющиеся у существующих в настоящее время систем распознавания речи могут объяснятся неполным соответствием между реальной речью и математическими моделями, лежащими в основе используемых методов. Качество системы распознавания речи определяется многими параметрами. В частности, большую роль играют точность распознавания, устойчивость системы к шумам, степень зависимости от диктора, зависимость от параметров микрофона. Построения полной математической модели, учитывающей все необходимые параметры, представляется сложной задачей. На сегодня аппарат скрытых Марковских моделей (СММ) является дефактно стандартом в области речевых технологий, используемым как для распознавания речи, так и для ее синтеза.

В основе применения СММ лежат рекурсивные процедуры, обладающие вычислительной сложностью. относительно количества состояний модели N и длины наблюдаемой последовательности Т. При работе с большим словарем и использовании трифонов в качестве моделей фонем число состояний достигает сотен, а длина наблюдаемой последовательности при распознавании слитной речи может быть, в принципе, неограниченной. При этом от систем автоматического распознавания речи (АРР) часто требуется, чтобы они работали в режиме реального времени, поэтому повышение быстродействия для таких систем является актуальной проблемой. В основе применения скрытых марковских моделей лежат рекурсивные процедуры, обладающие вычислительной сложностью. При этом от систем автоматического распознавания речи часто требуется, чтобы они работали в режиме реального времени, поэтому повышение быстродействия для таких систем является актуальной задачей. Материалы и методы. Одним из путей решения

данной задачи является реализация аппаратной поддержки вычислений в ассоциативной осцилляторной среде. Она обладает малыми аппаратными затратами из-за простоты базовых клеточных ансамблей и выполняемых ими функций и высоким быстродействием, не зависящим от длины наблюдаемой последовательности и количества состояний скрытых марковских моделей, благодаря массовому параллелизму и конвейерному характеру вычислений.

Целью данной работы является получение практических навыков работы с аудиоданными, в частности в области распознавания речи и поиска ключевых слов. Разработать собственную программу, запустить ее и проанализировать полученные данные. Провести сравнение полученных результатов с результатами других известных приложений и методов.

1.Основы распознавания речи

Не смотря на то, что названием данной работы является поиск ключевых слов, тема распознавания речи не могла не появиться в данной работе. Более того, поиск целиком и полностью базируется на распознавании, ведь чтобы искать что-то в речи необходимо чтобы это была именно речь, а не набор неясных шумов.

Начнём с того, что наша речь — это последовательность звуков. Звук в свою очередь — это суперпозиция (наложение) звуковых колебаний (волн) различных частот. Волна же, как нам известно из физики, характеризуются двумя атрибутами — амплитудой и частотой.

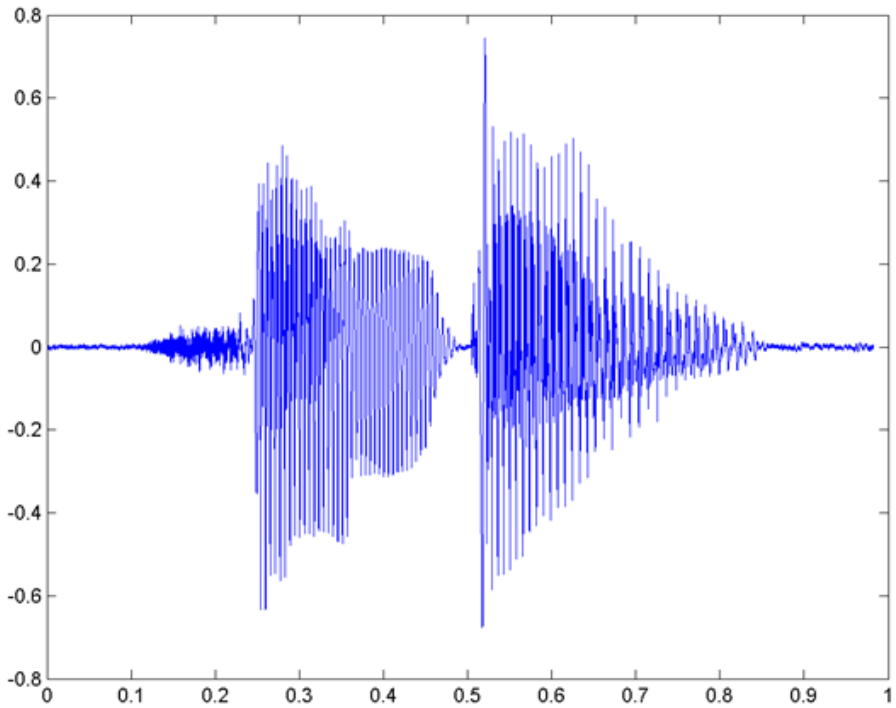

Рисунок 1.1. Вид сигнала.

Для того, что бы сохранить звуковой сигнал на цифровом носителе, его необходимо разбить на множество промежутков и взять некоторое «усредненное» значение на каждом из них.

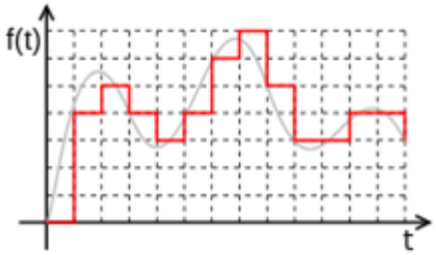

Рисунок 1.2. Среднее значение.

Таким вот образом механические колебания превращаются в набор чисел, пригодный для обработки на современных ЭВМ. Отсюда следует, что задача распознавания речи сводится к «сопоставлению» множества численных значений (цифрового сигнала) и слов из некоторого словаря (русского языка, например).

Входные данные:

Допустим у нас есть некоторый файл/поток с аудиоданными. Прежде всего, нам нужно понять, как он устроен и как его прочесть. Давайте рассмотрим самый простой вариант — WAV файл.

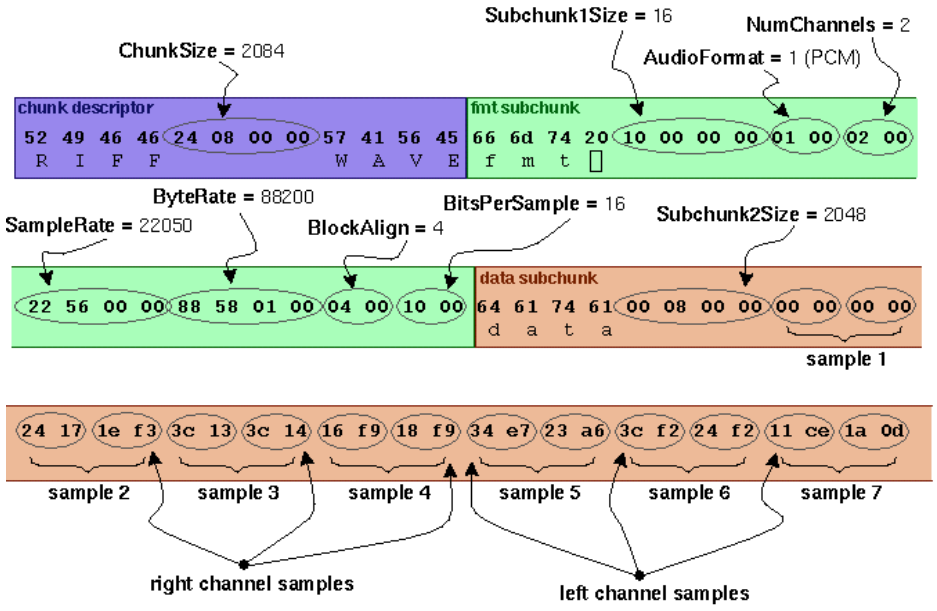

Рисунок 1.3. Строение исходного файла.

[2] Формат подразумевает наличие в файле двух блоков. Первый блок — это заголовка с информацией об аудиопотоке: битрейте, частоте, количестве каналов, длине файла и т.д. Второй блок состоит из «сырых» данных — того самого цифрового сигнала, набора значений амплитуд.

Логика чтения данных в этом случае довольно проста. Считываем заголовок, проверяем некоторые ограничения (отсутствие сжатия, например), сохраняем данные в специально выделенный массив.

1.1 Распознавание

Чисто теоретически, теперь мы можем сравнить (поэлементно) имеющийся у нас образец с каким-нибудь другим, текст которого нам уже известен. То есть попробовать «распознать» речь, однако этот метод нам не подходит.

Наш подход должен быть устойчив (ну хотя бы чуть-чуть) к изменению тембра голоса (человека, произносящего слово), громкости и скорости произношения. Поэлементным сравнением двух аудиосигналов этого, естественно, добиться нельзя.

Первым делом разобьём наши данные по небольшим временным промежуткам — фреймам. Причём фреймы должны идти не строго друг за другом, а “внахлёст”. Т.е. конец одного фрейма должен пересекаться с началом другого.

Фреймы являются более подходящей единицей анализа данных, чем конкретные значения сигнала, так как анализировать волны намного удобней на некотором промежутке, чем в конкретных точках. Расположение же фреймов “внахлёст” позволяет сгладить результаты анализа фреймов, превращая идею фреймов в некоторое “окно”, движущееся вдоль исходной функции (значений сигнала).

Опытным путём установлено, что оптимальная длина фрейма должна соответствовать промежутку в 10мс, «нахлёст» — 50%. С учётом того, что средняя длина слова (по крайней мере, в моих экспериментах) составляет 500мс — такой шаг даст нам примерно 500 / (10 * 0.5) = 100 фреймов на слово.

Разбиение слов.

Первой задачей, которую приходится решать при распознавании речи, является разбиение этой самой речи на отдельные слова. Для простоты предположим, что в нашем случае речь содержит в себе некоторые паузы (промежутки тишины), которые можно считать “разделителями” слов.

В таком случае нам нужно найти некоторое значение, порог — значения выше которого являются словом, ниже — тишиной. Вариантов тут может быть несколько:

1) задать константой (сработает, если исходный сигнал всегда генерируется при одних и тех же условиях, одним и тем же способом);

2) кластеризовать значения сигнала, явно выделив множество значений соответствующих тишине (сработает только если тишина занимает значительную часть исходного сигнала);

3) проанализировать энтропию;

Как вы уже догадались, речь сейчас пойдёт о последнем пункте:) Начнём с того, что энтропия — это мера беспорядка, “мера неопределённости какого-либо опыта” (с). В нашем случае энтропия означает то, как сильно “колеблется” наш сигнал в рамках заданного фрейма.

Для того, что бы подсчитать энтропию конкретного фрейма выполним следующие действия:

предположим, что наш сигнал пронормирован и все его значения лежат в диапазоне [-1;1];

построим гистограмму (плотность распределения) значений сигнала фрейма: рассчитаем энтропию, как

$$E = \sum_{i=0}^{N-1} P[i] * \log_2(P[i]) \qquad (1,1)$$

И так, мы получили значение энтропии. Но это всего лишь ещё одна характеристика фрейма, и для того, что бы отделить звук от тишины, нам по

прежнему нужно её с чем-то сравнивать. В некоторых статьях рекомендуют брать порог энтропии равным среднему между её максимальным и минимальным значениями (среди всех фреймов). Однако, в моём случае такой подход не дал сколь либо хороших результатов.

К счастью, энтропия (в отличие от того же среднего квадрата значений) — величина относительно самостоятельная. Что позволило мне подобрать значение её порога в виде константы (0.1).

Тем не менее проблемы на этом не заканчиваются: (Энтропия может проседать по середине слова (на гласных), а может внезапно вскакивать из-за небольшого шума. Для того, что бы бороться с первой проблемой, приходится вводить понятие “минимально расстояния между словами” и “склеивать” близ лежачие наборы фреймов, разделённые из-за проседания. Вторая проблема решается использованием “минимальной длины слова” и отсечением всех кандидатов, не прошедших отбор (и не использованных в первом пункте).

Если же речь в принципе не является “членораздельной”, можно попробовать разбить исходный набор фреймов на определённым образом подготовленные подпоследовательности, каждая из которых будет подвергнута процедуре распознавания. Но это уже совсем другая история.

MFCC

И так, мы у нас есть набор фреймов, соответствующих определённому слову. Мы можем пойти по пути наименьшего сопротивления и в качестве численной характеристики фрейма использовать средний квадрат всех его значений (Root Mean Square). Однако, такая метрика несёт в себе крайне мало пригодной для дальнейшего анализа информации.

Вот тут в игру и вступают Мел-частотные кепстральные коэффициенты (Mel-frequency cepstral coefficients). Согласно Википедии (которая, как известно, не врёт) MFCC — это своеобразное представление энергии спектра сигнала. Плюсы его использования заключаются в следующем:

Используется спектр сигнала (то есть разложение по базису ортогональных [ко]синусоидальных функций), что позволяет учитывать волновую “природу” сигнала при дальнейшем анализе;

Спектр проецируется на специальную mel-шкалу, позволяя выделить наиболее значимые для восприятия человеком частоты;

Количество вычисляемых коэффициентов может быть ограничено любым значением (например, 12), что позволяет “сжать” фрейм и, как следствие, количество обрабатываемой информации;

Давайте рассмотрим процесс вычисления MFCC коэффициентов для некоторого фрейма. Представим наш фрейм в виде вектора

$$x[k], 0 \leq k < N$$

где N — размер фрейма.

Разложение в ряд Фурье

Первым делом рассчитываем спектр сигнала с помощью дискретного преобразования Фурье (желательно его “быстрой” FFT реализацией).

$$X[k] = \sum_{n=0}^{N-1} x[n] * e^{-2*\pi*i*k*n/N}, 0 \leq k < N$$

Так же к полученным значениям рекомендуется применить оконную функцию Хэмминга, что бы “сгладить” значения на границах фреймов.

$$H[k] = 0.54 - 0.46 * \cos(2 * \pi * k/(N-1))$$

То есть результатом будет вектор следующего вида:

$$X[k] = X[k] * H[k], 0 \leq k < N$$

Важно понимать, что после этого преобразования по оси X мы имеем частоту (hz) сигнала, а по оси Y — магнитуду (как способ уйти от комплексных значений).

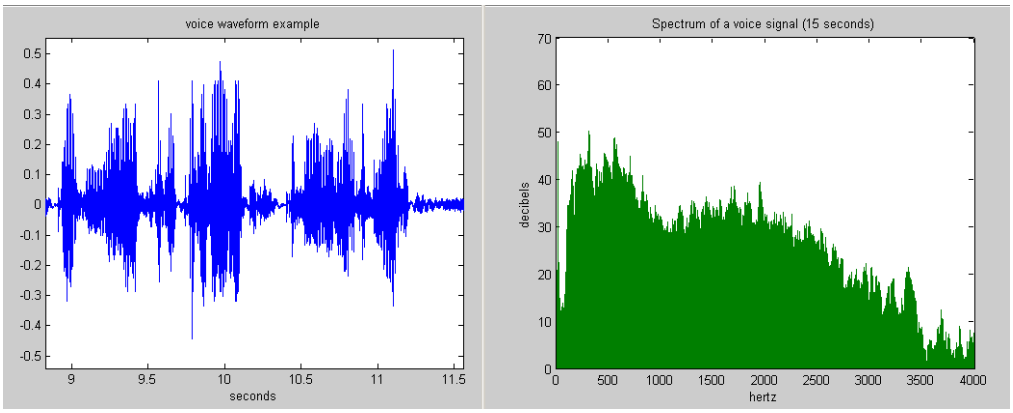

Рисунок 1.4. Преобразование.

1.2 Расчёт mel-фильтров

Начнём с того, что такое mel. mel — это “психофизическая единица высоты звука”, основанная на субъективном восприятии среднестатистическими людьми. Зависит в первую очередь от частоты звука (а так же от громкости и тембра). Другими словами, эта величина, показывающая, на сколько звук определённой частоты “значим” для нас. Преобразовать частоту в мел можно по следующей формуле (запомним её как «формула-1»):

$$M = 1127 * \log(1 + F/700)$$

Обратное преобразование выглядит так (запомним её как «формула-2»):

$$F = 700 * (e^{M/1127} - 1)$$

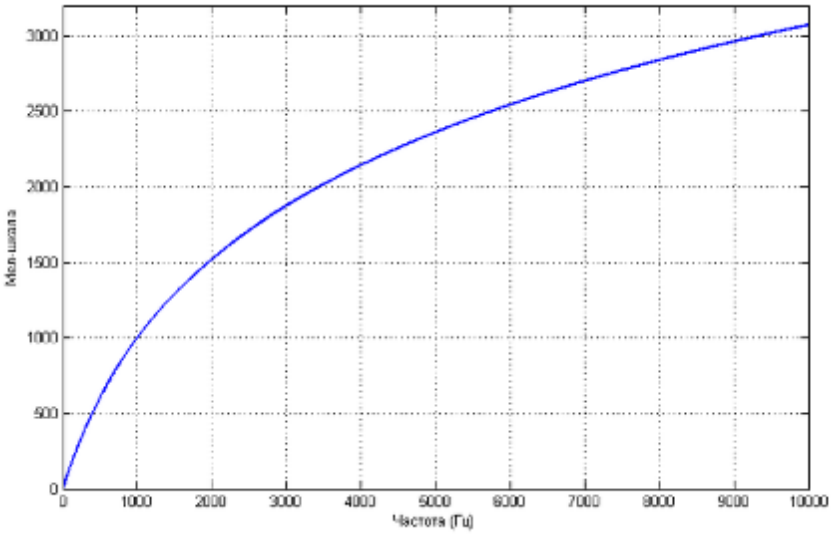

Рисунок 1.5. График зависимости mel / частота

Но вернёмся к нашей задаче. Допустим у нас есть фрейм размером 256 элементов. Мы знаем (из данных об аудиоформате), что частота звука в данной фрейме 16000hz. Предположим, что человеческая речь лежит в диапазоне от [300; 8000]hz. Количество искомых мел-коэффициентов положим M = 10 (рекомендуемое значение).

Для того, что бы разложить полученный выше спектр по mel-шкале, нам потребуется создать “гребёнку” фильтров. По сути, каждый mel-фильтр это треугольная оконная функция, которая позволяет просуммировать количество энергии на определённом диапазоне частот и тем самым получить mel-коэффициент. Зная количество мел-коэффициентов и анализируемый диапазон частот мы можем построить набор таких вот фильтров.

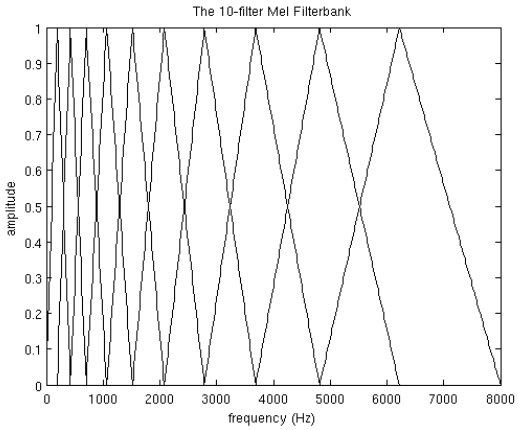

Рисунок 1.6. Фильтры.

Обратите внимание, что чем больше порядковый номер мел-коэффициента, тем шире основание фильтра. Это связано с тем, что разбиение интересующего нас диапазона частот на обрабатываемые фильтрами диапазоны происходит на шкале мелов.

Но мы опять отвлеклись. И так для нашего случая диапазон интересующих нас частот равен [300, 8000]. Согласно формуле-1 в на мел-шкале этот диапазон превращается в [401.25; 2834.99].

Далее, для того, что бы построить 10 треугольных фильтров нам потребуется 12 опорных точек:

m[i] = [401.25, 622.50, 843.75, 1065.00, 1286.25, 1507.50, 1728.74, 1949.99, 2171.24, 2392.49, 2613.74, 2834.99]

Обратите внимание, что на мел-шкале точки расположены равномерно. Переведём шкалу обратно в герцы с помощью формулы-2:

h[i] = [300, 517.33, 781.90, 1103.97, 1496.04, 1973.32, 2554.33, 3261.62, 4122.63, 5170.76, 6446.70, 8000]

Как видите теперь шкала стала постепенно растягиваться, выравнивая тем самым динамику роста “значимости” на низких и высоких частотах.

Теперь нам нужно наложить полученную шкалу на спектр нашего фрейма. Как мы помним, по оси X у нас находится частота. Длина спектра 256 — элементов, при этом в него умещается 16000hz. Решив нехитрую пропорцию можно получить следующую формулу:

f(i) = floor((frameSize+1) * h(i) / sampleRate)(1,8)

что в нашем случае эквивалентно f(i) = 4, 8, 12, 17, 23, 31, 40, 52, 66, 82, 103, 128.

Вот и всё! Зная опорные точки на оси X нашего спектра, легко построить необходимые нам фильтры по следующей формуле:

$$H_m(k) = \begin{cases} 0 & k < f(m-1) \\ \dfrac{k - f(m-1)}{f(m) - f(m-1)} & f(m-1) \le k \le f(m) \\ \dfrac{f(m+1) - k}{f(m+1) - f(m)} & f(m) \le k \le f(m+1) \\ 0 & k > f(m+1) \end{cases}$$

Применение фильтров, логарифмирование энергии спектра

Применение фильтра заключается в попарном перемножении его значений со значениями спектра. Результатом этой операции является mel-коэффициент. Поскольку фильтров у нас M, коэффициентов будет столько же.

$$S[m] = \log(\sum_{k=0}^{N-1} |X[k]|^2 * H_m[k]), 0 \le m < M$$

Однако, нам нужно применить mel-фильтры не к значениям спектра, а к его энергии. После чего прологарифмировать полученные результаты. Считается, что таким образом понижается чувствительность коэффициентов к шумам.

Косинусное преобразование

Дискретное косинусное преобразование (DCT) используется для того, что бы получить те самые “кепстральные” коэффициенты. Смысл его в том, что бы “сжать” полученные результаты, повысив значимость первых коэффициентов и уменьшив значимость последних.

В данном случае используется DCTII без каких-либо домножений на (scale factor).

$$C[l] = \sum_{m=0}^{M-1} S[m] * \cos(\pi * l * (m + \frac{1}{2})/M)), 0 \leq l < M$$

Теперь для каждого фрейма мы имеем набор из M mfcc-коэффициентов, которые могут быть использованы для дальнейшего анализа.

1.3 Алгоритм распознавания

Вот тут, дорогой читатель, тебя и ждёт главное разочарование. В интернетах мне довелось увидеть множество высокоинтеллектуальных (и не очень) споров о том, какой же способ распознавания лучше.

На данный момент, предлагаю остановится на гораздо менее эффективном, но в разы более простом способе.

И так, вспомним, что наша задача заключается в распознавании слова из некоторого словаря. Для простоты, будем распознавать называния первых десять цифр: “один“, “два“, “три“, “четыре“, “пять“, “шесть“, “семь“, “восемь“, “девять“, “десять“.

Теперь возьмем в руки айфон/андроид и пройдёмся по L коллегам с просьбой продиктовать эти слова под запись. Далее поставим в соответствие (в какой-нибудь локальной БД или простом файле) каждому слову L наборов mfcc-коэффициентов соответствующих записей.

Это соответствие мы назовём “Модель”, а сам процесс — Machine Learning! На самом деле простое добавление новых образцов в базу имеет крайне слабую связь с машинным обучением… Но уж больно термин модный.

Теперь наша задача сводится к подбору наиболее “близкой” модели для некоторого набора mfcc-коэффициентов (распознаваемого слова). На первый взгляд задачу можно решить довольно просто:

для каждой модели находим среднее (евклидово) расстояние между идентифицируемым mfcc-вектором и векторами модели;

выбираем в качестве верной ту модель, среднее расстояние до которой будет наименьшим;

Однако, одно и тоже слово может произносится как Андреем Малаховым, так и каким-нибудь его эстонским коллегой. Другими словами размер mfcc-вектора для одного и того же слова может быть разный.

К счастью, задача сравнения последовательностей разной длины уже решена в виде Dynamic Time Warping алгоритма. Этот алгоритм динамического программирования прекрасно расписан как в буржуйской Wiki, так и на православном Хабре.

Единственное изменение, которое в него стоит внести — это способ нахождения дистанции. Мы должны помнить, что mfcc-вектор модели — на самом деле последовательность mfcc-“подвекторов” размерности M, полученных из фреймов. Так вот, DTW алгоритм должен находить дистанцию между последовательностями эти самых “подвекторов” размерности M. То есть в качестве значений матрицы расстояний должны использовать расстояния (евклидовы) между mfcc-“подвекторами” фреймов.

2. Скрытые марковские модели

Скрытая марковская модель — статистическая модель, имитирующая работу процесса, похожего на марковский процесс с неизвестными параметрами, и задачей ставится разгадывание неизвестных параметров на основе наблюдаемых. Полученные параметры могут быть использованы в дальнейшем анализе, например, для распознавания образов. СММ может быть рассмотрена как простейшая байесовская сеть доверия.

Ма́рковский проце́сс — случайный процесс, эволюция которого после любого заданного значения временного параметра t не зависит от эволюции, предшествовавшей t, при условии, что значение процесса в этот момент фиксировано («будущее» процесса не зависит от «прошлого» при известном «настоящем»; другая трактовка (Вентцель): «будущее» процесса зависит от «прошлого» лишь через «настоящее»).

Процесс Маркова — модель авторегрессии первого порядка

AR(1): $X_t = c + \alpha X_{t-1} + \varepsilon_t$

Марковская цепь — частный случай марковского процесса, когда пространство его состояний дискретно (т.е. не более чем счетно).

Рассмотрим простой пример марковского случайного процесса. По оси абсцисс случайным образом перемещается точка. В момент времени ноль точка находится в начале координат и остается там в течении одной секунды. Через секунду бросается монета — если выпал герб, то точка X перемещается на одну единицу длины вправо, если цифра — влево. Через секунду снова бросается монета и производится такое же случайное перемещение, и так далее. Процесс изменения положения точки («блуждания») представляет собой случайный процесс с дискретным временем (t=0, 1, 2, …) и счетным множеством состояний. Такой случайный процесс называется марковским, так как следующее состояние точки зависит только от настоящего (текущего) состояния и не зависит от прошлых

состояний (неважно, каким путем и за какое время точка попала в текущую координату).

Структура скрытой марковской модели

В обычной марковской модели состояние видимо наблюдателю, поэтому вероятности переходов — единственный параметр. В скрытой марковской модели мы можем следить лишь за переменными, на которые оказывает влияние данное состояние. Каждое состояние имеет вероятностное распределение среди всех возможных выходных значений. Поэтому последовательность символов, сгенерированная СММ, даёт информацию о последовательности состояний.

Диаграмма, представленная ниже, показывает общую структуру СММ. Овалы представляют собой переменные со случайным значением. Случайная переменная x(t) представляет собой значение скрытой переменной в момент времени t. Случайная переменная y(t) — это значение наблюдаемой переменной в момент времени t. Стрелки на диаграмме символизируют условные зависимости.

Из диаграммы становится ясно, что значение скрытой переменной x(t) (в момент времени t) зависит только от значения скрытой переменной x(t-1) (в момент t-1). Это называется свойством Маркова. Хотя в то же время значение наблюдаемой переменной y(t) зависит только от значения скрытой переменной x(t) (обе в момент времени t).

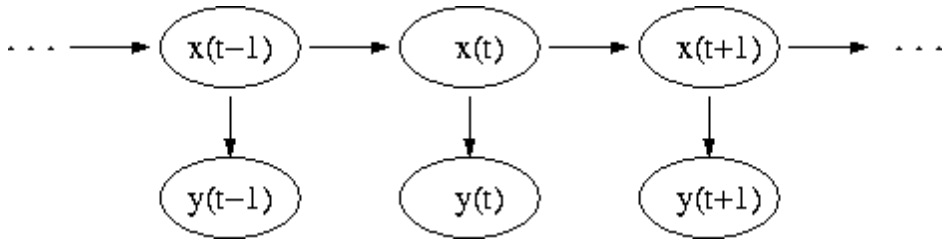

Рисунок 2.1 Схема переходов Марковского процесса.

Вероятность увидеть последовательность $Y = y(0), y(1), \ldots, y(L-1)$ длины L равна

$$P(Y) = \sum_{X} P(Y \mid X) P(X), \quad (2,2)$$

здесь сумма пробегает по всем возможным последовательностям скрытых узлов $X = x(0), x(1), \ldots, x(L-1)$. Метод подсчёта полным перебором значений P(Y) — очень трудоёмкий для многих задач из реальной жизни в силу того, что количество возможных последовательностей скрытых узлов очень велико. Но применение процедуры прямого-обратного хода позволяет существенно увеличить скорость вычислений.

2.1 Алгоритмы

Алгоритм «прямого-обратного» хода — алгоритм для вычисления апостериорных вероятностей последовательности состояний при наличии последовательности наблюдений. Иначе говоря, алгоритм, вычисляющий вероятность специфической последовательности наблюдений. Алгоритм применяется в трёх алгоритмах скрытых Марковских моделей.

Алгоритм включает три шага:

1) вычисление прямых вероятностей
2) вычисление обратных вероятностей
3) вычисление сглаженных значений

Прямые и обратные шаги часто называют «прямым проходом по сообщению» и «обратным проходом по сообщению». Где сообщениями выступают ряд последовательных наблюдений. Формулировка происходит из способа, которым алгоритм обрабатывает данную последовательность наблюдений. Сначала алгоритм продвигается, начинаясь с первого наблюдения в последовательности и идя в последнее, и затем возвращаясь назад к первому. При каждом наблюдении в вероятностях последовательности, которые будут использоваться для вычислений при следующем наблюдении, вычислены. Во время обратного прохода алгоритм одновременно выполняет шаг сглаживания. сглаживание — это процесс

вычисления распределения вероятностей значений переменных в прошлых состояниях при наличии свидетельств вплоть до нынешнего состояния. Этот шаг позволяет алгоритму принимать во внимание все прошлые наблюдения для того, чтобы вычислить более точные результаты.

Алгоритм Витерби — алгоритм поиска наиболее подходящего списка состояний (называемого путём Витерби), который в контексте цепей Маркова получает наиболее вероятную последовательность произошедших событий.

Является алгоритмом динамического программирования. Применяется в алгоритме свёрточного декодирования Витерби.

Алгоритм был предложен Эндрю Витерби в 1967 году как алгоритм декодирования свёрточного кода, передаваемого по сетям с наличием шума. Алгоритм получил широкое применение в декодировании свёрточных кодов мобильных телефонов стандартов GSM и CDMA, dial-up модемах и беспроводных сетях стандарта 802.11. Также он широко используется в распознавании речи, синтезе речи, компьютерной лингвистике и биоинформатике. К примеру, при распознавании речи звуковой сигнал воспринимается как последовательность событий и строка текста есть «скрытый смысл» акустического сигнала. Алгоритм Витерби находит наиболее вероятную строку текста по данному сигналу.

Алгоритм делает несколько предположений: наблюдаемые и скрытые события должны быть последовательностью. Последовательность чаще всего упорядочена по времени. Две последовательности должны быть выровнены: каждое наблюдаемое событие должно соответствовать ровно одному скрытому событию вычисление наиболее вероятной скрытой последовательности до момента t должно зависеть только от наблюдаемого события в момент времени t, и наиболее вероятной последовательности до момента $t - 1$.

Алгоритм Баума — Велша используется в информатике и статистике для нахождения неизвестных параметров скрытой марковской модели (HMM). Он использует алгоритм прямого-обратного хода и является частным случаем обобщённого EM-алгоритма.

2.2 СММ в распознавании речи

Самое быстрое и эффективное взаимодействие между людьми происходит посредством устной речи. С помощью речи могут быть переданы различные чувства и эмоции, а главное — полезная информация. Необходимость создания компьютерных интерфейсов звукового ввода-вывода не вызывает сомнений, поскольку их эффективность основана на практически неограниченных возможностях формулировки в самых различных областях человеческой деятельности.

Первая электронная машина, синтезирующая английскую речь, была представлена в Нью-Йорке на торговой выставке в 1939 году и называлась voder, но звук, который она воспроизводила, был крайне нечетким. Первое же устройство для распознавания речи вышло в свет в 1952 втором году и было способно распознавать цифры.

При процессе распознавания речи можно выделить следующие сложности: произвольный, наивный пользователь; спонтанная речь, сопровождаемая аграмматизмами и речевым «мусором»; наличие акустических помех и искажений; наличие речевых помех.

Из всего многообразия методов в данной статье мы рассмотрим возможность создания статистической модели посредством скрытых Марковских моделей (СММ).

Part-Of-Speech tagging

При анализе естественного языка первым шагом необходимо определить: к какой части речи относится каждое из слов в предложении. В английском языке задача на этом этапе называется Part-Of-Speech tagging. Каким образом мы можем определить часть речи отдельного члена предложения? Рассмотрим предложение на английском языке: «The can will rust». Итак, the –определенный артикль или частица «тем»; can – может одновременно являться и модальным глаголом, и существительным, и глаголом; will – модальный глагол, существительное и глагол; rust – существительное или глагол. В статистическом подходе необходимо

построить таблицу вероятностей использования слов в каждом грамматическом значении. Эту задачу можно решить на основе тестовых текстов, проанализированных вручную. И сразу можно выделить одну из проблем: слово «can» в большинстве случаев используется в качестве глагола, но иногда оно может являться и существительным. Учитывая этот недостаток, была создана модель, принимающая во внимание тот факт, что после артикля последует прилагательное или существительное:

$$\arg\max_{t1..tm} \prod_{t=1}^{n} p(w_i|t_i)p(t_i|t_{i-1}) \quad (2,3)$$

Где: t – таг (существительное, прилагательное и т.д.)

w – слово в тексте (rust, can …)

p(w|t) – вероятность того, что слово w соответствует тагу t

p(t1|t2) – вероятность того, что t1 идет после t2

Из предложенной формулы видно, что мы пытаемся подобрать таги так, чтобы слово подходило тагу, и таг подходил предыдущему тагу. Данный метод позволяет определить, что «can» выступает в роли существительного, а не как модального глагола.

Эргодическая Марковская модель на практике

Каждая вершина в данной схемы обозначает отдельную часть речи, в которой записываются пары (слово; вероятность, что слово относится именно к этой части речи). Переходы показывают возможную вероятность следования одной части речи за другой. Так, например, вероятность того, что подряд будут идти 2 артикля, при условии, что встретится артикль, будет равна 0,0016. Данный этап распознавания речи очень важен, так как правильное определение грамматической структуры предложения позволяет подобрать верную грамматическую конструкцию для экспрессивной окраски воспроизводимого предложения.

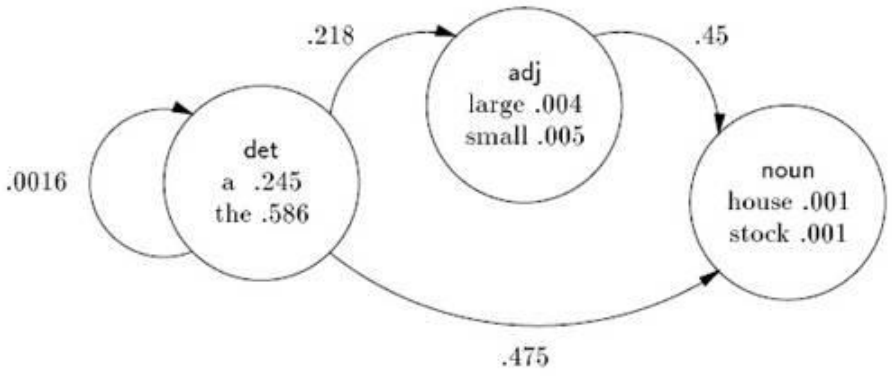

Рисунок 2.2 CMM модель.

N-граммные модели

Также существуют n-граммные модели распознавания речевого потока. Они основаны на предположении, что вероятность употребления очередного слова в предложении зависит только от n-1 слов. Сегодня наиболее популярные биграммные и триграммные модели языка. Поиск в таких моделях происходит по большой таблице (корпусу). Несмотря на быстро работающий алгоритм, такие модели не способны уловить семантические и синтаксические связи, если зависимые слова находятся на расстоянии 5 слов друг от друга. Использование же n-граммных моделей, где n больше чем 5, требует огромных мощностей.

Как уже отмечалось выше, самой популярной моделью на сегодняшний день является триграммная модель. Условная вероятность наблюдения предложения w1, … wn приближена к:

$$P(w1,\ldots,wn) = \Pi P(wi \mid w1,\ldots,w2) \approx \Pi P(wi \mid wi-(n-1), \ldots, wi-1)0$$

Например, рассмотрим предложение «I want to go home». Вероятность этого предложения можно вычислить от счета частоты n-грамма (в этом примере возьмем n=3):

$$P(I, want, to, go, home) \approx P(I)*P(want|I)*P(to|I, want)*P(go|want, to)\times$$
$$\times P(home|\ to, go)$$

Стоит отметить, дальнодействующую триграммную модель, в которой анализ ведется не только по двум предшествующим словам, а по любой паре слов, находящихся рядом. Такая триграммная модель может пропускать малоинформативные слова, тем самым улучшая предсказуемость сочетаемости в модели.

3. Отчеты об опытно-экспериментальной работе, сравнение и анализ результатов

В ходе экспериментальной части мной были исследованы методы визуализации и обработки аудиоданных. В частности вейвлет преобразования.

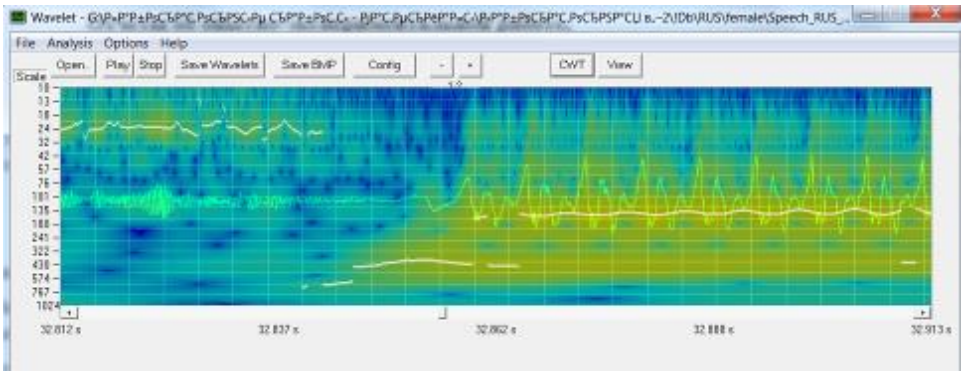

Рисунок 3.1 Визуализация данных.

Мною была написана простейшая программа для распознавания речи, а также были найдены и проанализированы результаты чужих экспериментов.

Рисунок 3.2. Интерфейс программы

Данная программа принимает на вход простые wave файлы и производит их обработку. Детальный обзор ее алгоритма рассмотрен в 1 главе данной работы.

Таким образом, можно выделить следующие этапы обработки:

1) Выбор семантической единицы для разбиения на промежутки аудио потока.

2) Разбиение на фреймы

3) Разбиение слов

4) Определение порога тишины

5) Вычисление коэффициентов

6) Разложение в ряд Фурье (или вейвлеты)

7) Расчет фильтров

8) Распознавание

Эксперименты

Результаты же тестов на выборке из 3х экземпляров для каждого слова в несинтетических условиях показали, мягко говоря, не лучший результат — 55% верных распознаваний.

Результаты сравнения представлены в таблице 3.1.

Таблица 3.1

Наименование метода(алгоритма)	Наименование программы	Мультиязычность	Приблизительный процент точного распознавания, %
--	Google services (search and other)	Да	95±10
--	Voice Digger	Нет	75-95
искусственные нейронные сети	--	Да	85-90
Решетка слогов	--	Да	88
N-best	--	Да	80
СММ	--	ДА	55 (Моя версия. Сильно зависит от размера словаря обучения и предварительной обработки звуков.)
СММ	--	Да	80-95
Фонетический стенограф	--	Да	92

Разумеется, результаты лишь примерные и сравнение данных полученных при разных условиях является не совсем честно. Процент может меняться

абсолютно от любых факторов. Будь то окружающие шумы, состояние диктора, объем обучения алгоритма и даже язык. Теоретически английская речь должна быть более расположена к распознаванию в ввиду того, что она жестко структурирована, слова в ней слабо поддаются изменениям (приставки, окончания, суффиксы…), а также алфавит и словесная база значительно меньше. С другой стороны, в русском языке звуки более четкие и слабо похожи друг на друга.

Заключение

Мною была проведена опытно-экспериментальная работа по работе с аудиоданными. Разработана программная база для дальнейших исследований в области распознавания речи и поиска ключевых слов в ней. Однако достигнутые практически результаты оставляют желать лучшего по множественным причинам. В дальнейшем мною будет продолжена работа над проблемами, возникшими в курсовой работе.

Список использованной литературы

1) Свободная библиотека Wikipedia
2) Свободная база знаний Habrahabr
3) Свободная база знаний Geektimes
4) База статей Санкт-Петербургского ВУЗа ИМТО
5) Пилипенко В.В. Використання фонетичного стенографа при розтзнаванш мовлення з великих словнишв / В.В. Пилипенко // Тезисы 12-й международной конференции «Автоматика - 2005». - Харьков, 2005. - С. 73.
6) Ле Н.В. Распознавание речи на основе искусственных нейронных сетей [Текст] / Н.В. Ле, Д.П. Панченко // Технические науки в России и за рубежом: материалы междунар. науч. конф. (г. Москва, май 2011 г.).
7) Янь Цзинбинь, Хейдоров.И.Э., Алиев Р.М. KEYWORD SEARCH USING SYLLABLE LATTICE
8) S. Young, G. Evermann, D. Kershaw and others. The HTK Book - Cambridge University Engineering Department, 2002
9) Vintsiuk Taras K. Generalized Automatic Phonetic Transcribing of Speech Signals / Taras K. Vintsiuk // Труды Пятой Всеукраинской международной конференции «Оброблення сигналів і зображень та розпізнавання образів» / УАсО1РО. - Київ, 2000.