Применение речевого диалога при управлении техническими средствами.

Автор: Ясенков Ярослав Геннадьевич
лицей №1580 при МГТУ им.Н.Э.Баумана, класс 11-3.

Научный руководитель: Жигулёвцев Юрий Николаевич
МГТУ им. Н.Э.Баумана
доцент, к.т.н.

Москва - 2015

Оглавление

Анотация

В настоящее время научное сообщество вкладывает огромное количество денег в научно-исследовательские разработки для решения проблем автоматического распознавания речи. Это стимулируется практическими требованиями, ведь системам распознавания речи можно найти широкое применение как в военной, так и в гражданской сфере. Некоторые компании смогли достичь определённых успехов. Самым качественным распознавателем английской речи стал продукт компании Dragon Systems, называемый Dragon Naturally Speaking Preferred. Он обеспечивает безошибочность распознавания на уровне 95%. Широкую известность в последнее время получил сервис Google Voice Search. Изначально этим сервисом поддерживались только короткие поисковые запросы длиной до 35-40 слов и только английский язык. Сегодня доступно распознавание непрерывной речи и множество языков, включая русский. В 2014 году был запущен Google Speech API, доступный для широкого круга разработчиков. Сервис основан на облачной технологии, то есть на компьютере или мобильном устройстве происходит только запись звука. После обработки записи голосового сообщения, компьютер или смартфон отправляет данные на сервер Google, где происходит непосредственное распознавание, после чего распознанный запрос возвращается на устройство в виде текста. [3] Среди распознавателей русской речи, хотелось бы отметить сервис компании Yandex, SpeechKit. Работа этого сервиса организована аналогично сервису Google Speech API. Но между ними есть существенные различия:

-SpeechKit изначально поддерживает русский язык,

-Яндекс предоставляет русскоязычную документацию

-Google Speech Recognition API предоставляется только на коммерческой основе (количество бесплатных запросов ограничено 50 в сутки)

-SpeechKit Cloud API для исследовательских целей предоставляется бесплатно

Именно из-за этих различий я выбрал сервис SpeechKit Cloud для своей работы.

Введение

Существует множество способов организации общения человека с машиной. Но наиболее эффективным методом, а также наиболее естественным для человека является речевой диалог. Именно поэтому внедрение речевого общения с машиной очень актуальная задача. Применение такого способа управления несёт множество преимуществ: остаются свободными руки оператора, не имеют значения условия освещения, механической вибрации, проста методика наблюдения за реакцией на команду, в систему «человек-машина могут быть включены люди с физическими недостатками. [2]

Речевое общение может быть отнесено к одной из проблем искусственного интеллекта. Полное решение проблемы речевого диалога возможно лишь в рамках создания искусственного интеллекта, но множество практических задач решено уже на современном уровне. Сложившаяся к настоящему времени структура задач может быть представлена в виде схемы (рисунок 1) [1].

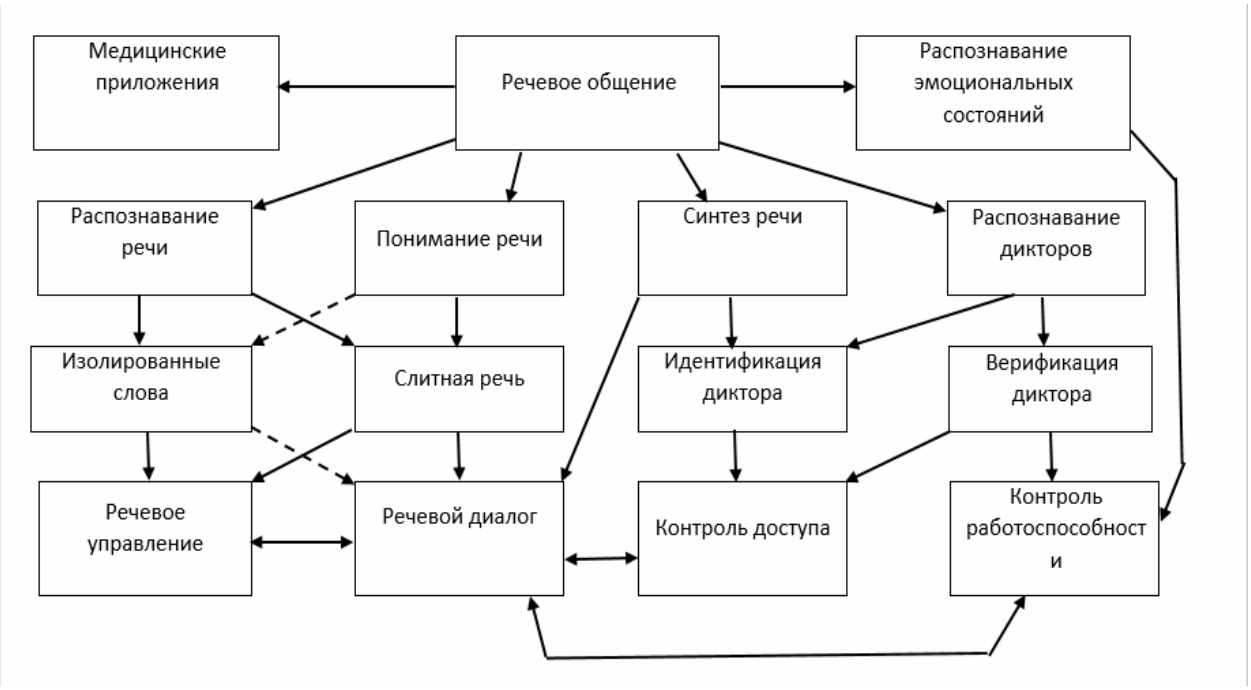


Рисунок 1

Как видно из схемы, основными направлениями исследований в области речевого общения помимо распознавания речи являются работы по синтезу

речевых сообщений, а также по распознаванию индивидуальности говорящего: идентификации (определению, кто говорит из известного круга лиц) и верификации (подтверждению личности говорящего). Много внимания также уделяется и медицинским приложениям проблемы речевого общения, позволяющим осуществлять диагностику психических и соматических заболеваний, приводящих к тем или иным отклонениям в речевой функции.

Основные варианты приложения методов речевого общения, предназначенные для использования в автоматизированных системах управления(АСУ) движущимися объектами, представлены нижним рядом блоков на рис.1. Двунаправленные связи указывают на возможность организации управления, контроля доступа и работоспособности в режиме диалога (с помощью тестов, предлагаемых системой) [1]

Описание процесса распознавания.

Рассмотрим, как происходит распознавание. Звук – это физическое явление, представляющее собой распространение в виде упругих волн механических колебаний в твёрдой, жидкой или газообразной среде. [5] Основными характеристиками звуковых колебаний являются частота и амплитуда. При цифровой записи звука производится его временная дискретизация и квантование. Дискретизация представляет собой разбивку непрерывной звуковой волны на короткие участки, для каждого из которых измеряется величина амплитуды. Квантование заключается в разбиении диапазона амплитуд на подуровни. Затем каждому измеренному значению амплитуды присваивается номер подуровня, которому соответствует это значение. То есть звук записывается в виде массива значений амплитуд. Теоретически, можно сравнивать полученную запись поэлементно с образцом, текст которого уже известен. Но такой подход совершенно неустойчив к малейшему изменению тембра голоса, громкости и скорости произношения. В реальной системе распознавания происходят следующие преобразования, которые можно представить в виде схемы (рисунок 6).

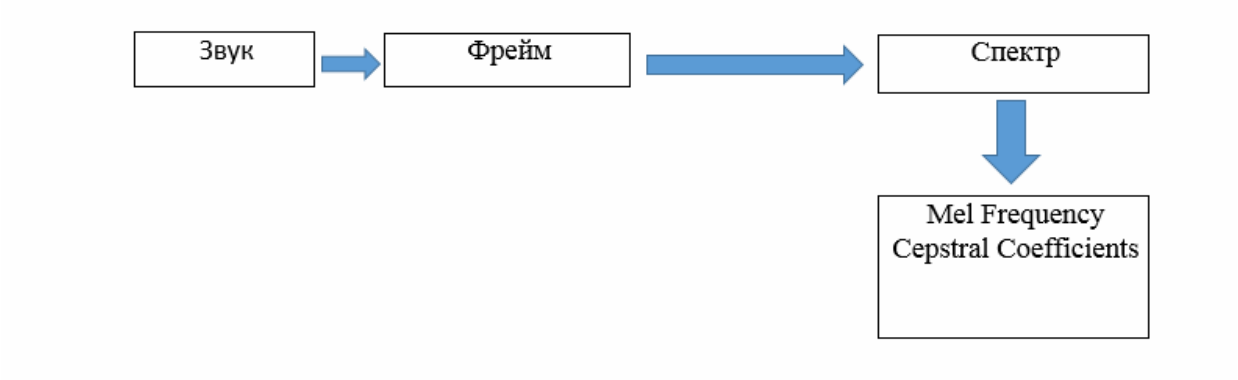


Рисунок 6

Сначала звук разбивается на небольшие временные промежутки – фреймы. Один фрейм – это участок длиной в 10 мс, каждый следующий фрейм накладывается на предыдущий с «нахлёстом» 50%, такое наложение позволяет сгладить результаты анализа. В качестве численной характеристики фрейма

можно использовать средний квадрат его значений, но такая характеристика несет крайне мало информации. Для последующего анализа используются Мел-частотные кепстральные коэффициенты (Mel-frequency cepstral coefficients).[6]

Рассмотрим процесс вычисления MFCC для некоторого фрейма.

Как описывалось выше, сигнал в дискретном виде можно записать так, $x[k], 0 \leq k < N$, где N – размер фрейма.

Далее к записанному сигналу применяем преобразование Фурье, чтобы получить спектр сигнала.

$$X_a[k] = \sum_{n=0}^{N-1} x[n] e^{\frac{-2\pi i}{N} kn}, \quad 0 \leq k < N$$

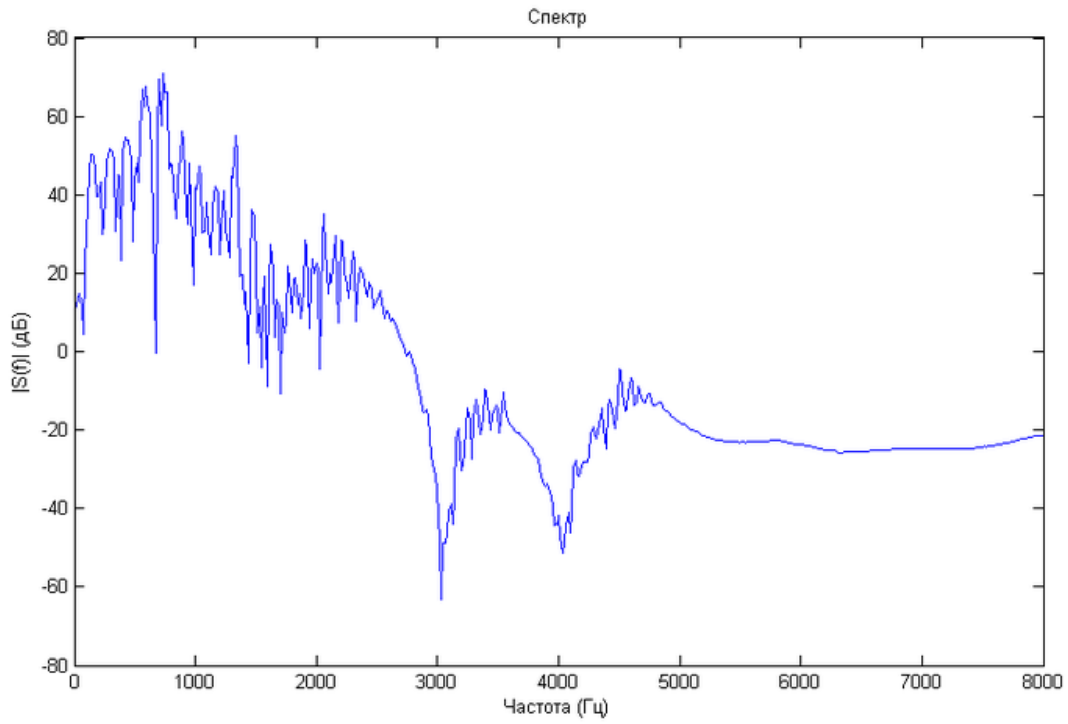


Рисунок 9

Для того, чтобы 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.

Если диапазон интересующих нас частот звука равен [300;8000], то по формуле $M = 1127 * \log(1 + F/700)$ на мел-шкале этот диапазон превращается в [401.25; 2834.99]. Далее, для того, чтобы 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]. Теперь с помощью формулы $F = 700 * (e^{M/1127} - 1)$переведём шкалу обратно в герцы.

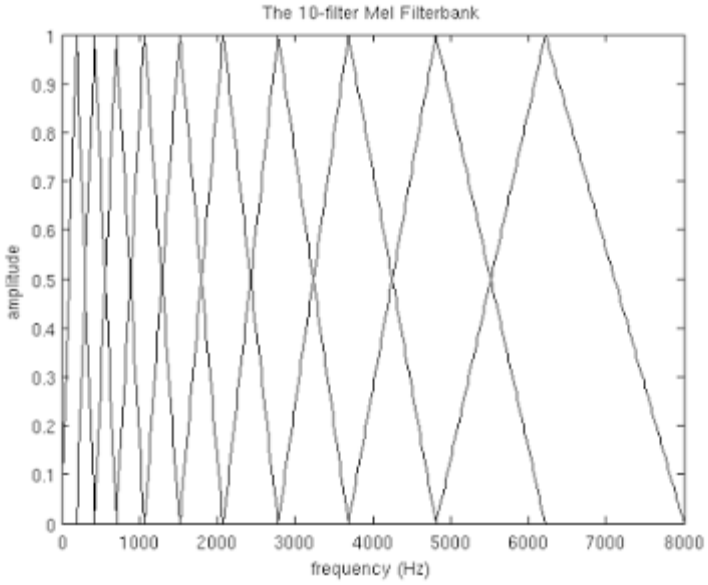


Рисунок 7

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

На рисунке 7 заметно, что окна «собираются» в области низких частот, обеспечивая более высокое «разрешение» там, где оно необходимо для распознавания.

Теперь необходимо наложить полученную шкалу на спектр фрейма. Применив формулу f(i) = [(frameSize+1) * h(i) / sampleRate] получим опорные точки. Зная опорные точки, легко построить необходимые фильтры по 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 полученного фильтра заключается в попарном перемножении его значений со значениями спектра.

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

Результатом этой операции является спектральный mel-коэффициент. Дискретное косинусное преобразование (DCT) используется для того, чтобы получить кепстральные коэффициенты.

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

Теперь для каждого фрейма имеется набор из M MFCC-коэффициентов, которые могут быть использованы для дальнейшего анализа.

В современных системах распознавания речи, в частности Yandex SpeechKit, для анализа используются скрытые марковские модели (СММ). Скрытая марковская модель позволяет соотнести набор наблюдаемых величин (звук) и скрытые состояния (фонемы). Схематически СММ изображена на рисунке 8. y1, y2, y3 – наблюдаемые значения, x1,x2,x3 – скрытые состояния, a12, a23, a21 – вероятности переходов, b1,b2,b3 – вероятности результатов. Чтобы однозначно задать скрытую марковскую модель необходимо знать распределение вероятностей состояний и матрицу вероятностей переходов. Распределение вероятностей состояний на данном фрейме возвращает акустическая модель (сравнивающая полученные MFCC с коэффициентами известной фонемы). А матрицу вероятностей переходов получается задать на основе статистических данных, так известно, что одни сочетания фонем произносятся легко и встречаются часто, другие сложнее для произношения и на практике используются реже. Таким образом, имея словарь, в котором задано

множество марковских моделей, можно определить какая из гипотез распознавания наиболее вероятна.

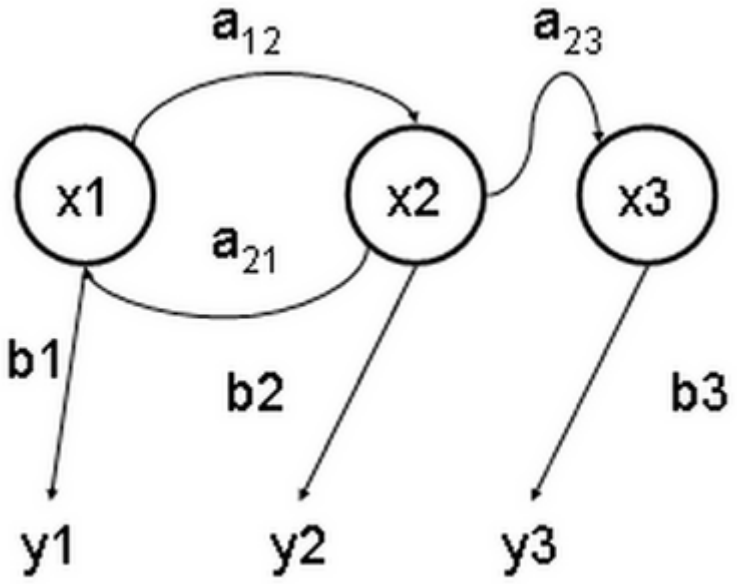


Рисунок 8

Качество распознавания

В зависимости от тематики голосовых запросов сервис SpeechKit обеспечивает достаточно высокое качество, так для queries (короткие фразы на различную тематику) безошибочность заявлена на уровне 88%, для maps (адреса, географические объекты) 95%, для notes (свободно диктуемый текст) 82%. Достижение 100% безошибочности на сегодняшний день затруднено по целому ряду причин.

Наиболее серьёзные проблемы возникают при условиях [8]:

-спонтанная речь, сопровождаемая аграмматизмами и речевым «мусором»

-наличие акустических помех и искажений, в том числе меняющихся

-наличие речевых помех

В своей работе я исследовал зависимость качества распознавания сервиса Yandex.Speechkit от акустических помех (шума).

Описание работы

Количество приложений распознающих русскую речь мало на рынке программного обеспечения, а эффективность многих из них находится на низком уровне. Поэтому в моей работе и ставилась цель – создать эффективную систему распознавания речи. В качестве инструмента разработки было решено использовать среду Visual Studio. В качестве языка программирования был выбран C++.

Структуру программы можно представить в виде схемы (рисунок 2).

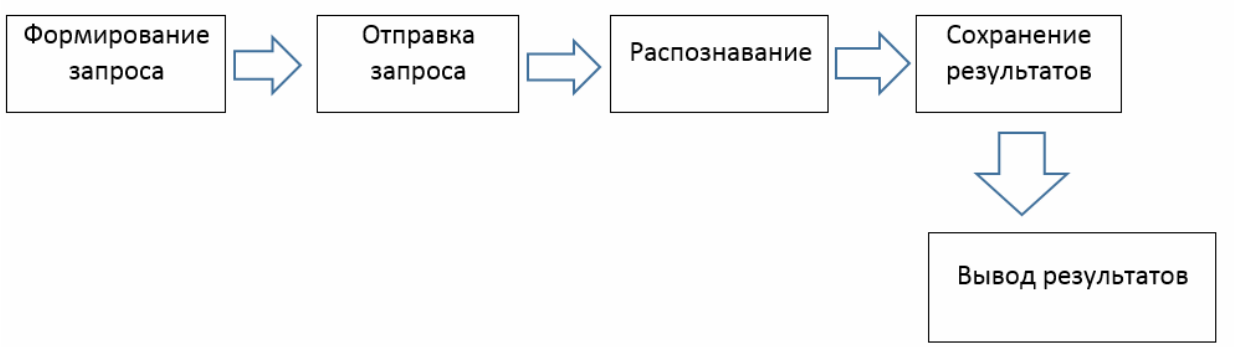


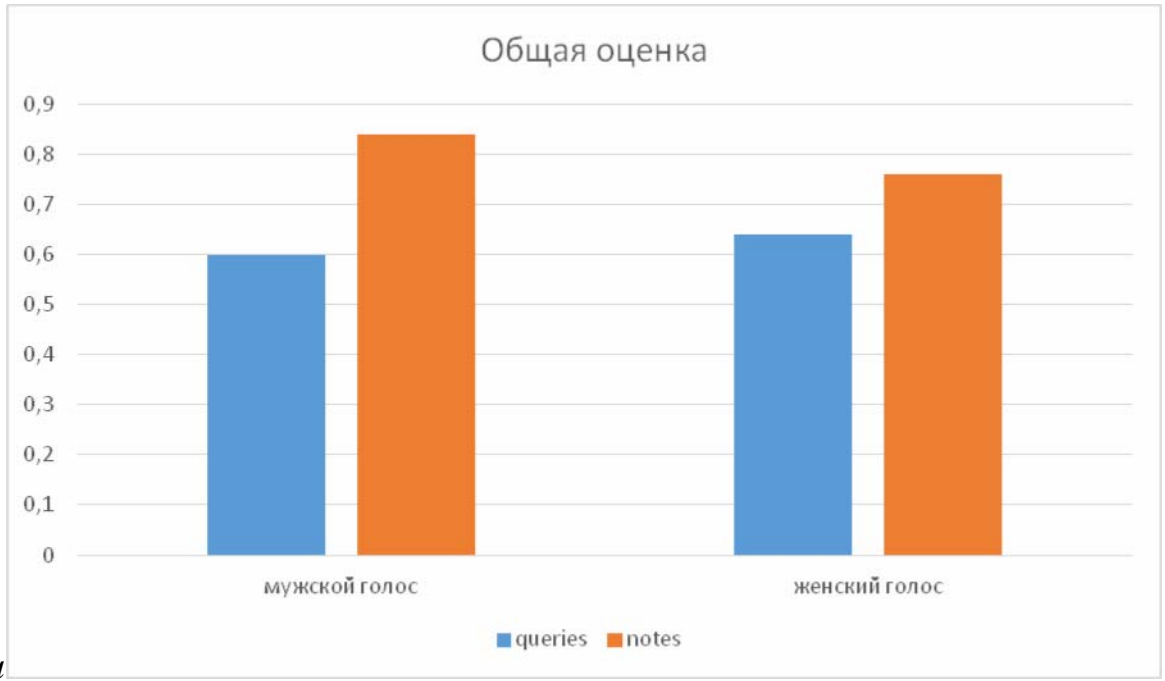
Рисунок 2

На начальном этапе разработки было решено использовать возможности библиотеки Libcurl для отправки речевых сообщений. Библиотека libcurl – это свободно распространяемая и простая в использовании клиентская библиотека, позволяющая работать со множеством различных протоколов с синтаксисом URL, чем и объясняется её применение. Библиотека libcurl была собрана посредством компилятора Visual Studio и подключена к проекту.

Для отправки речевого сообщения необходимо сначала записать аудиофайл при помощи звукового редактора. На начальном этапе разработки название записанного файла приходилось вводить в консоли, что затрудняет пользование приложением. Сейчас эта задача решена при помощи библиотеки MFC. Создано диалоговое окно, в котором по нажатию кнопки пользователь выбирает необходимый для отправки файл. В случае если распознавание прошло успешно, будет выведен результат с оценкой достоверности, в ином случае будет выведено сообщение «не распознано».

В ходе работы были сделаны независимые оценки качества распознавания. Для проведения оценки были использованы таблицы фраз из ГОСТ Р 50840-95. Использованные фразы состоят из 3-4 слов, произносятся в нормальном темпе речи (4 слога в секунду) в среднем за 2,5 секунды и носят бытовой характер. Уровень звукового давления в помещении, в котором проводился эксперимент составлял 26 дБ, что соответствует условиям указанного стандарта при проведении оценки. Результаты оценки можно представить в виде диаграммы.

Результаты анали-



за

Рисунок 3

Для языковой модели queries процент распознанных слов составил 62%, для модели notes 80%. Несколько худшие результаты чем заявлено можно объяснить несколькими причинами: непрофессиональные дикторы, использование иного записывающего оборудования (микрофон ноутбука), неидеальные шумовые условия.

Для изучения влияния шума на качество распознавания были отобраны только точно распознанные фразы. Для оценки использовались разные виды шума. Так зависимость качества от уровня звукового давления белого шума представлена на рисунке 4.

Рисунок 4

При наложении на аудиозапись белого шума с уровнем -20 дБ качество падает до 75%, при дальнейшем повышении уровня звука до -17 дБ наблюдается падение в 2 раза, вплоть до -11 дБ. При -11 дБ правильно распознанных фраз нет, лишь отдельные слова. При наложении розового шума распознавание затруднено уже при -20 дБ. Наложение броуновского (коричневого) шума никак не влияет на качество при уровнях ниже 0 дБ (максимально возможный уровень шума в редакторе cool edit pro, 0 дБ – это максимальный уровень громкости, который данное устройство может отобразить без искажений).

В ходе исследования также была обнаружена зависимость качества распознавания от голоса говорящего. Она представлена на рисунке 5. Так для мужского голоса на фоне белого шума количество распознанных фраз почти в два раза больше.

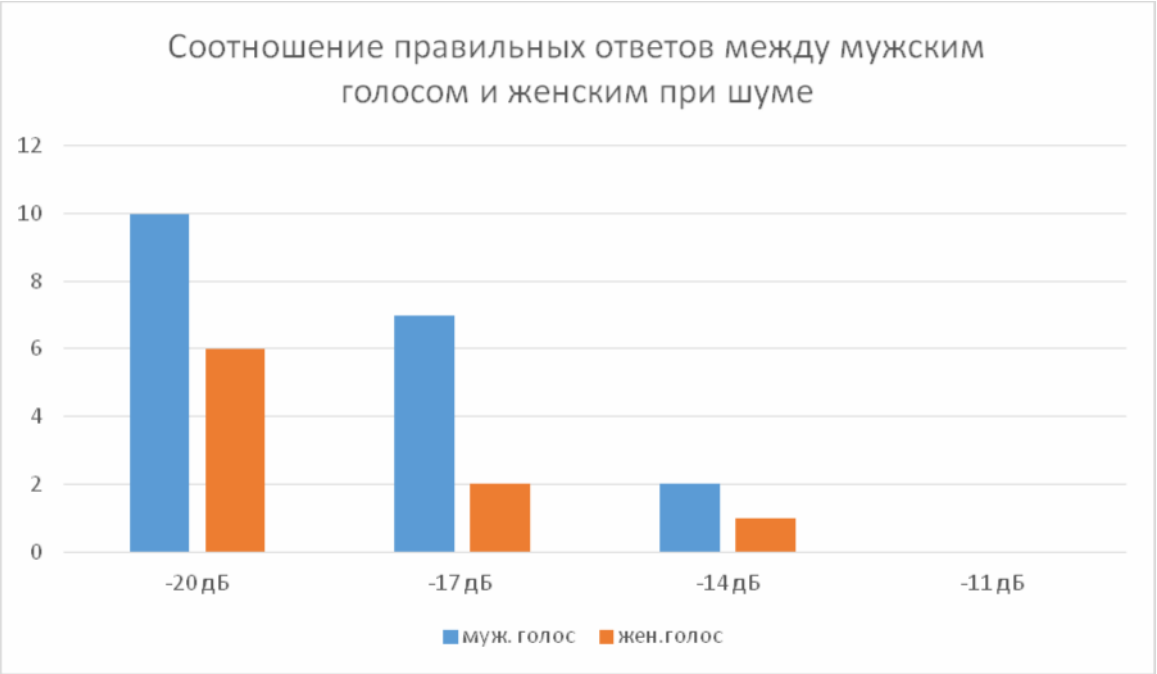


Рисунок 5

Как заявляют разработчики SpeechKit при распознавании учитывается распределение вероятностей, которое дает языковая модель (n-gram language model). То есть наличие фиксированного контекста «мама мыла ...» задает распределение вероятностей для следующего слова, которое отражает как семантику, так и морфологию. [4]

Для исследования семантического уровня было решено использовать длинные запросы, длиной около 50 слов (около 25 секунд), представляющие собой фрагменты радиопередачи.

Было выявлено, что, как и указывалось разработчиками, для диктовки длинных текстов необходима модель notes. Так для одного фрагмента при модели notes количество слов в неправильной форме (неправильный падеж или число) составляет 4, а количество слов, распознанных неверно 1. Для модели queries 6 слов имеют неправильную форму и 5 слов распознаны неправильно. Из всего этого можно сделать вывод что языковая модель на сегодняшний день ещё недостаточно натренирована.

Выводы.

1. Рассмотрен процесс распознавания речи на примере сервиса компании Yandex.

Реализован эффективный распознаватель речи, основанный на облачной технологии и не требующий больших вычислительных мощностей. Проанализированы возможности сервиса Yandex SpeechKit в ходе которых выявлены несколько худшие результаты распознавания: 62% против заявленных 88% у модели queries, и 80% против 82% у модели notes. Выявлена сильная зависимость от шума, так при уровне белого шума в -11 дБ распознавание прекращается, хотя человек легко понимает сказанное. Наложение розового шума (близкого к природным шумам) вызывает ухудшение результатов уже при уровне в -20 дБ.

Применение такого распознавателя возможно в разных сферах:
-управление мультимедийной и навигационной системой автомобиля
-голосовое управление такими устройствами как персональный компьютер или смартфон,
-управление умным домом
На сегодняшний день применение такой системы на борту летательных аппаратов затруднено по ряду причин:
-недостаточно высокое качество распознавания в реальных условиях
-недостаточная устойчивость к разного вида акустическим шумам
-сложности в локальном размещении системы.

Список литературы:

1. В.Н. Плотников, В.А.Суханов, Ю.Н.Жигулевцев «Речевой диалог в системах управления»;
2. Р.К.Потапова, «Речевое управление роботом»
3. http://old.kv.by/index2010381103.htm
4. Блог компании Яндекс http://habrahabr.ru/company/yandex/blog/198556
5. https://ru.wikipedia.org/wiki/%D0%97%D0%B2%D1%83%D0%BA
6. http://habrahabr.ru/post/226143/
7. Д.Либерти «Освой самостоятельно C++ за 21 день.»
8. «Система управления распознаванием речевой информации» К.П.Нарян

Приложение.

```
#include "stdafx.h"
#include "MFCApplication2.h"
#include "MFCApplication2Dlg.h"
#include "afxdialogex.h"
#include <curl/curl.h>
#include <iostream>
#include <fstream>
#include <sstream>
using namespace std;

#ifdef _DEBUG
#define new DEBUG_NEW
#endif


// Диалоговое окно CAboutDlg используется для описания сведений о приложении

class CAboutDlg : public CDialogEx
{
public:
	CAboutDlg();

// Данные диалогового окна
	enum { IDD = IDD_ABOUTBOX };

	protected:
	virtual void DoDataExchange(CDataExchange* pDX);    // поддержка DDX/DDV

// Реализация
protected:
	DECLARE_MESSAGE_MAP()
};

CAboutDlg::CAboutDlg() : CDialogEx(CAboutDlg::IDD)
{
}

void CAboutDlg::DoDataExchange(CDataExchange* pDX)
{
	CDialogEx::DoDataExchange(pDX);
}

BEGIN_MESSAGE_MAP(CAboutDlg, CDialogEx)
END_MESSAGE_MAP()


// диалоговое окно CMFCApplication2Dlg



CMFCApplication2Dlg::CMFCApplication2Dlg(CWnd* pParent /*=NULL*/)
	: CDialogEx(CMFCApplication2Dlg::IDD, pParent)
{
	m_hIcon = AfxGetApp()->LoadIcon(IDR_MAINFRAME);
}

void CMFCApplication2Dlg::DoDataExchange(CDataExchange* pDX)
{
```

```
	CDialogEx::DoDataExchange(pDX);
}

BEGIN_MESSAGE_MAP(CMFCApplication2Dlg, CDialogEx)
	ON_WM_SYSCOMMAND()
	ON_WM_PAINT()
	ON_WM_QUERYDRAGICON()
	ON_BN_CLICKED(IDC_BUTTON1, &CMFCApplication2Dlg::OnBnClickedButton1)
END_MESSAGE_MAP()


// обработчики сообщений CMFCApplication2Dlg

BOOL CMFCApplication2Dlg::OnInitDialog()
{
	CDialogEx::OnInitDialog();

	// Добавление пункта "О программе..." в системное меню.

	// IDM_ABOUTBOX должен быть в пределах системной команды.
	ASSERT((IDM_ABOUTBOX & 0xFFF0) == IDM_ABOUTBOX);
	ASSERT(IDM_ABOUTBOX < 0xF000);

	CMenu* pSysMenu = GetSystemMenu(FALSE);
	if (pSysMenu != NULL)
	{
		BOOL bNameValid;
		CString strAboutMenu;
		bNameValid = strAboutMenu.LoadString(IDS_ABOUTBOX);
		ASSERT(bNameValid);
		if (!strAboutMenu.IsEmpty())
		{
			pSysMenu->AppendMenu(MF_SEPARATOR);
			pSysMenu->AppendMenu(MF_STRING, IDM_ABOUTBOX, strAboutMenu);
		}
	}

	// Задает значок для этого диалогового окна.  Среда делает это автоматически,
	//  если главное окно приложения не является диалоговым
	SetIcon(m_hIcon, TRUE);			// Крупный значок
	SetIcon(m_hIcon, FALSE);		// Мелкий значок

	// TODO: добавьте дополнительную инициализацию

	return TRUE;  // возврат значения TRUE, если фокус не передан элементу управления
}

void CMFCApplication2Dlg::OnSysCommand(UINT nID, LPARAM lParam)
{
	if ((nID & 0xFFF0) == IDM_ABOUTBOX)
	{
		CAboutDlg dlgAbout;
		dlgAbout.DoModal();
	}
	else
	{
		CDialogEx::OnSysCommand(nID, lParam);
	}
}

// При добавлении кнопки свертывания в диалоговое окно нужно воспользоваться приведенным
ниже кодом,
//  чтобы нарисовать значок.  Для приложений MFC, использующих модель документов или
представлений,
```

```
//  это автоматически выполняется рабочей областью.

void CMFCApplication2Dlg::OnPaint()
{
	if (IsIconic())
	{
		CPaintDC dc(this); // контекст устройства для рисования

		SendMessage(WM_ICONERASEBKGND, reinterpret_cast<WPARAM>(dc.GetSafeHdc()), 0);

		// Выравнивание значка по центру клиентского прямоугольника
		int cxIcon = GetSystemMetrics(SM_CXICON);
		int cyIcon = GetSystemMetrics(SM_CYICON);
		CRect rect;
		GetClientRect(&rect);
		int x = (rect.Width() - cxIcon + 1) / 2;
		int y = (rect.Height() - cyIcon + 1) / 2;

		// Нарисуйте значок
		dc.DrawIcon(x, y, m_hIcon);
	}
	else
	{
		CDialogEx::OnPaint();
	}
}

// Система вызывает эту функцию для получения отображения курсора при перемещении
//  свернутого окна.
HCURSOR CMFCApplication2Dlg::OnQueryDragIcon()
{
	return static_cast<HCURSOR>(m_hIcon);
}

//функция записи данных ответа
size_t write_response_data(char *ptr, size_t size, size_t nmemb, void *userdata)
{
	std::stringstream* s = (std::stringstream*)userdata;
	size_t n = size * nmemb;  s->write(ptr, n);
	return n;
}

//функция, определяющая размер отправляемого потока
size_t read_request_data(char *ptr, size_t size, size_t nmemb, void *userdata)
{
	std::ifstream* f = (std::ifstream*)userdata;
	size_t n = size * nmemb;  f->read(ptr, n);
	size_t result = f->gcount();  return result;
}


	//utf8->Unicode(переводим utf8 в Юникод)
	static wchar_t* utf8_to_unicode__dontForgetDeleteArr(const char *utf8_string)
	{
		wchar_t* pRes = 0;
		int res_len = 0;

		//тест на возможность преобразования
		res_len = MultiByteToWideChar(CP_UTF8, 0, utf8_string, -1, 0, 0);
		if (!res_len)return 0;

		//выделяем память
		pRes = new wchar_t[res_len];
		if (!pRes)return 0;
```

```
	//преобразование
	if (!MultiByteToWideChar(CP_UTF8, 0, utf8_string, -1, pRes, res_len))
	{
		delete[] pRes;
		return 0;
	}

	return pRes;
}

//unicode->1251
static char * unicode_to_1251__dontForgetDeleteArr(const wchar_t *unicode_string)
{
	char* pRes = 0;
	int res_len = 0;

	//тест на возможность преобразования
	res_len = WideCharToMultiByte(1251, 0, unicode_string, -1, 0, 0, 0, 0);
	if (!res_len)return 0;

	//выделяем память
	pRes = new char[res_len];
	if (!pRes)return 0;

	//преобразование
	if (!WideCharToMultiByte(1251, 0, unicode_string, -1, pRes, res_len, 0, 0))
	{
		delete[] pRes;
		return 0;
	}

	return pRes;
}


void CMFCApplication2Dlg::OnBnClickedButton1()
{
	// TODO: Add your control notification handler code here
	CFileDialog ldFile(TRUE);
	if (ldFile.DoModal() == IDOK)
	{

		CString m_strResults = ldFile.GetPathName();//получаем путь к файлу

		char fp[1000];
		char a[1000];
		wchar_t* unicode_string = 0;
		char* cp1251_string = 0;


		CString b;
		b = "";

		CString FILE_PATH = m_strResults;

		CURL *curl = NULL;
		curl = curl_easy_init();//инициализируем сеанс cURL
		if (curl)
		{
		curl_easy_setopt(curl, CURLOPT_HEADER, 0);//не выводим заголовок
		curl_easy_setopt(curl, CURLOPT_POST, 1);//выбираем post запрос
	      curl_easy_setopt(curl, CURLOPT_VERBOSE, 0);//не выводим доп.информацию
			curl_easy_setopt(curl, CURLOPT_IPRESOLVE, CURL_IPRESOLVE_V4);
```

```
                struct curl_slist *headers = NULL;

                headers = curl_slist_append(headers, "Content-Type: audio/x-wav");
            curl_easy_setopt(curl, CURLOPT_HTTPHEADER, headers);//указываем заголовок

                        curl_easy_setopt(curl, CURLOPT_USERAGENT, "Dalvik/1.2.0
(Linux; U; Android 2.2.2; LG-P990 Build/FRG83G");//содержимое заголовка Useragent
                        curl_easy_setopt(curl, CURLOPT_URL,
"asr.yandex.net/asr_xml?key=uniquekey4&uuid=12345678123456781234567812345678&topic=notes&
lang=ru-RU");//выбираем загружаемый URL

                        std::ifstream fileStream(FILE_PATH, std::ifstream::binary);
                        fileStream.seekg(0, fileStream.end);
                        int length = fileStream.tellg();
                        fileStream.seekg(0, fileStream.beg);

                        curl_easy_setopt(curl, CURLOPT_READFUNCTION,
&read_request_data);//ограничиваем количество данных для чтения
                        curl_easy_setopt(curl, CURLOPT_POSTFIELDSIZE, length);//задаем
размер файла
                        curl_easy_setopt(curl, CURLOPT_READDATA,
&fileStream);//считываем файл

                        std::stringstream contentStream;



                       curl_easy_setopt(curl, CURLOPT_WRITEFUNCTION,
&write_response_data);//получаем ответ
                        curl_easy_setopt(curl, CURLOPT_WRITEDATA,
&contentStream);//выводим ответ


                        CURLcode code = curl_easy_perform(curl);//отправка данных

                        unsigned httpCode;
                        curl_easy_getinfo(curl, CURLINFO_HTTP_CODE,
&httpCode);//получение данных о передаче
                        std::stringstream msg;
                        msg << "Http code is " << httpCode;

                        std::ofstream fout;
                        fout.open("vyv.txt");//сохраняем ответ в файле

                        fout << contentStream.str();
                        fout.close();

                        curl_free(headers);//освобождаем память libcurl
                        curl_easy_cleanup(curl);//завершаем сессию

                        std::ifstream fin;
                        fin.open("vyv.txt");//открываем файл для чтения

                        //выводим необходимую информацию
                        boolean markerzapisi = false;
                        boolean markerkonez = false;
                        char pred=*"o";
                        char s;
                        int i;
                        int k = 0;
                        for ( i = 1; markerkonez != true;k++)
                        {
                               fin.get(s);//перебираем все символы файла
```

```
                                    if ((s == *"v") && (pred == *"<")) markerzapisi =
true;//если встретилось сочетание <v
                                    if ((s == *"/") && (pred == *"<"))
                                    {
                                          markerzapisi = false;//если встретилось </ то не
записываем
                                          markerkonez = true;
                                    }
                                    if ((markerzapisi == true) && (markerkonez != true))
                                    {
                                          a[i] = s;//то записываем
                                          i++;
                                    };
                                    pred = s;
                                    if ((k == 90) && (markerzapisi == false))//если не
встретилось подтверждение распознания, то выводим нераспознано
                                    {
                                          markerkonez = true;
                                          b = "не распознано";
                                    }

                              }

                              //убираем пустые символы
                              for (int k = i;k != sizeof(a); k++)
                              {
                                    a[k] = *"";
                              }

                              //конвертируем текст для корректного вывода
                              char* cp1251_string = 0;
                              for (;;)
                              {

                                    unicode_string =
utf8_to_unicode__dontForgetDeleteArr(a);
                                    if (!unicode_string)
                                    {
                                          AfxMessageBox("Не удалось конвертировать в uni-
code!");
                                          break;
                                    }

                                    cp1251_string = uni-
code_to_1251__dontForgetDeleteArr(unicode_string);

                                    if (!cp1251_string)
                                    {
                                          AfxMessageBox("Не удалось конвертировать из uni-
code!");
                                          break;
                                    }

                                    break;
                              }

                              //выводим результат, если успешно распознано
                              if (b != "не распознано")
                               this->SetDlgItemText(IDC_EDIT1, cp1251_string);
                              if (b == "не распознано")
                                    this->SetDlgItemTextA(IDC_EDIT1, b);
                              if (unicode_string)
                              {
                                    delete[] unicode_string;
```

```
                unicode_string = 0;
        }
        if (cp1251_string)
        {
                delete[] cp1251_string;
                cp1251_string = 0;
        }
}
```