Учреждение образования

«Белорусский государственный университет информатики и радиоэлектроники»

Кафедра интеллектуальных информационных технологий

ОТЧЁТ

по лабораторной работе №4-5 по дисциплине «ЕЯзИИС»

на тему: «Разработка систем анализа речи»

Выполнили студенты группы 821701: Поживилко П.С.
Витушко Л. Д.

Проверил: Крапивин Ю.Б.

Минск, 2021

Цель работы: освоить на практике основные принципы создания систем анализа и синтеза речи.

Основные задачи:

1. Изучить основы создания систем анализа/синтеза речи.
2. Закрепить навыки программирования на языке высокого уровня (C#, Java, Python…).

Ход работы.

Для распознавания речи использовалась модель vosk, а точнее ее маленькая русская модель. Она способна распознавать речь в автономно, не требуя интернет соединения, сам vosk toolkit поддерживает около 20 языков, также есть возможность изменять модель под свои нужды или использовать свою. В качестве примера была написана программа распознающая речь и выполняющая несколько команд, а именно это старт и стоп, открыть/создать/записать файл, а также сделать запрос на википедию. Запрос предварительно нормализуется с помощью pymorphy.

```
recognized_data = rec.Result() if rec.AcceptWaveform(data) else rec.PartialResult()
result = ''
if 'старт' in recognized_data:
    start = 1
if start:
    if 'text' in json.loads(recognized_data):
        result = json.loads(recognized_data)['text']
        if result != '':
            print(result)
            words = nltk.word_tokenize(result)
            for i, word in enumerate(words):
                words[i] = morph.normal_forms(word)[0]
            print(words)
            if 'википедия' in words:
                results = wikipedia.summary(' '.join(words[1:]), sentences=5)
                print(results)
            if 'создать файл' in ' '.join(words):
                open(words[2], 'w')
            if 'открыть файл' in ' '.join(words):
                file = open(words[2], 'r')
                print(file.read())
            if 'записать файл' in ' '.join(words):
                file = open(words[2], 'w')
                file.write(' '.join(words[3:]))
                file.close()

if 'стоп' in recognized_data:
    break
```

Рисунок 1. Код программы

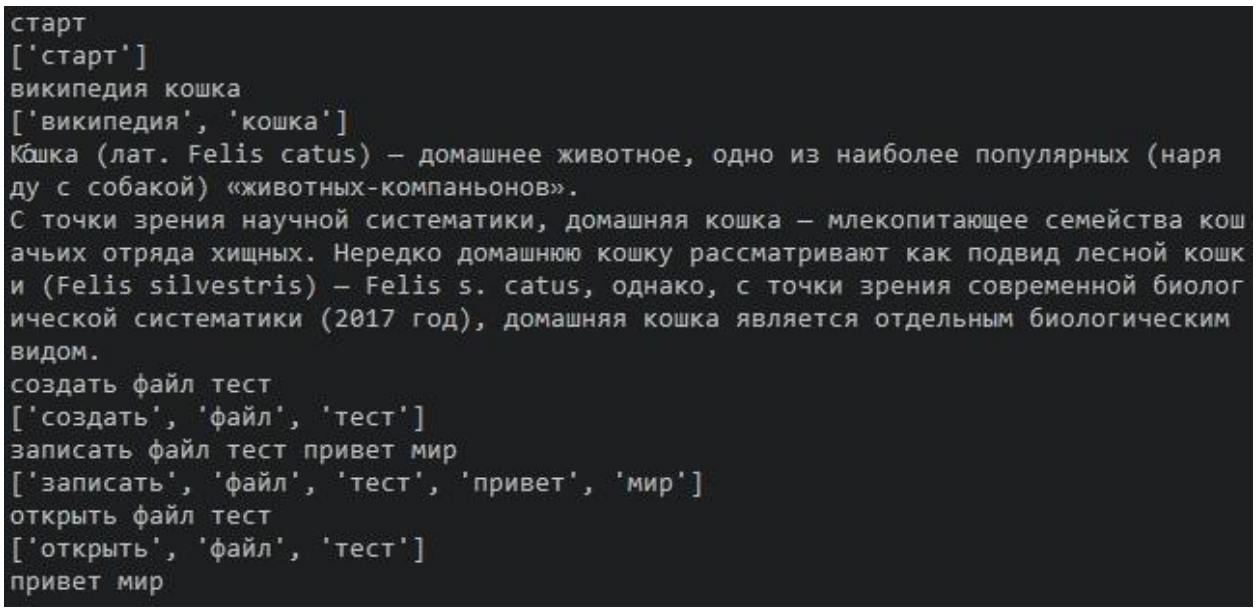

Рисунок 2. Результат работы программы

Точно не написано, как работает vosk, однако известно, что модели в нем скорее всего (возможно не все) представлены в виде графов. Происходит разделение аудио файла на отдельные куски («chunk»), затем эти куски хешируют и добавляют в базу, а при распознновании обходят базу и предполагают, что было сказано.

Для синтеза речи использовался Java Speech API.

Java Speech API (JSAPI) - это интерфейс для кроссплатформенной поддержки распознавателей команд и управления, систем диктовки и синтезаторов речи. Хотя JSAPI определяет только интерфейс, существует несколько реализаций, в данной лабораторной используется реализация FreeTTS.

Через Java Speech API поддерживаются две основные речевые технологии: синтез речи и распознавание речи.

Синтез речи обеспечивает обратный процесс создания синтетической речи из текста, сгенерированного приложением, апплетом или пользователем. Это часто называют технологией преобразования текста в речь.

Основные этапы создания речи из текста заключаются в следующем:

1. Анализ структуры: Обрабатывает входной текст, чтобы определить, где начинаются и заканчиваются абзацы, предложения и другие структуры. Для большинства языков на этом этапе используются данные пунктуации и форматирования.

2. Предварительная обработка текста: Анализирует входной текст на наличие специальных конструкций языка. В английском языке требуется особый режим для сокращений, сокращений, дат, времени, чисел, сумм в валюте, адресов электронной почты и многих других форм. Другие языки требуют специальной обработки для этих форм, и большинство языков имеют другие специализированные требования.

Результатом этих первых двух шагов является устная форма письменного текста. (ред.)

Распознавание речи предоставляет компьютерам возможность прослушивать разговорную речь и определять, что было сказано. Другими словами, он обрабатывает аудиовход, содержащий речь, путем преобразования его в текст.

Основные этапы типичного распознавателя речи заключаются в следующем:

3. Грамматический дизайн: Определяет слова, которые может произносить пользователь, и шаблоны, в которых они могут произноситься.
4. Обработка сигнала: Анализирует спектральные (т.е. частотные) характеристики входящего аудио.
5. Распознавание фонем: Сравнивает образцы спектра с образцами фонем распознаваемого языка.
6. Распознавание слов: Сравнивает последовательность вероятных фонем со словами и шаблонами слов, заданными активными грамматиками.
7. Генерация результата: Предоставляет приложению информацию словах, обнаруженных распознавателем во входящем аудио.

```java
import java.util.Locale;
import java.util.Scanner;

import javax.speech.AudioException;
import javax.speech.Central;
import javax.speech.EngineException;
import javax.speech.synthesis.Synthesizer;
import javax.speech.synthesis.SynthesizerModeDesc;

public class Main {

    public static void main(String[] args) throws EngineException, AudioException, InterruptedException {
        Scanner scanner = new Scanner(System.in);
        scanner.useDelimiter("\n");
        System.setProperty(
            "freetts.voices",
            "com.sun.speech.freetts.en.us" + ".cmu_us_kal.KevinVoiceDirectory");
        Central.registerEngineCentral(
            "com.sun.speech.freetts"
                + ".jsapi.FreeTTSEngineCentral");
        Synthesizer synthesizer = Central.createSynthesizer(new SynthesizerModeDesc(Locale.US));
        synthesizer.allocate();

        while (true) {
            System.out.println("write");
            String str = scanner.nextLine();
            synthesizer.resume();
            synthesizer.speakPlainText(str, null);
            synthesizer.waitEngineState(Synthesizer.QUEUE_EMPTY);
        }

    }

}
```

Рисунок 3. Код программы для синтеза речи