重 庆 大 学

学 生 实 验 报 告

实验课程名称 多媒体技术

开课实验室 DS1501

学 院 软件学院 年级 2025 专业班 软件工程 1 班

学 生 姓 名 学 号

开 课 时 间 2024 至 2025 学年第 二 学期

总 成 绩	
教师签名	桑军

软件学院制

《多媒体技术》实验报告

开课实验室：DS1501 **2025 年 9 月 27 日**

学院	软件学院	年级、专业、班	2323 级软件工程 1 班	姓名		成绩	
课程名称	多媒体技术	实验项目名　称	语音合成/语音合成	指导教师	桑军		
教师评语	教师签名：桑军 年　月　日						

一、实验目的

了解 Microsoft Speech SDK 的用法，对 TTS 和 SR 有一般性的认识，了解如何进行文本语音转换和语音识别的一般性编程。

二、实验原理

熟悉 Microsoft Speech SDK

TTS(Text-To-Speech,文本语音转换)顾名思义，将文本转找成语音进行输出。目前 TTS 技术已相对比较成熟，TTS 引擎也比较多，比较有名的有 Microsoft Speech SDK 和 IBM 的 ViaVoice 还有开源的 Festival 等。

微软的 Microsoft Speech SDK 种主要包括两方面的内容，一个是语音识别，一个是语音合成，虽然微软的这套引擎不是业界标准，不过其应用范围还是很广的，作为一个入门级的应用还是很有价值的。目前微软的这套引擎支持多种语言的识别和朗读，不过对于不同的语言要下载不同的语言包。

对于微软的 Microsoft Speech API 下面简称 SAPI，包括下面几个部分：

(1)Voice Commands API。对应用程序进行控制，一般用于语音识别系统中。识别某个命令后，会调用相关接口是应用程序完成对应的功能。如果程序想实现语音控制，必须使用此组 对象。

(2)Voice Dictation API。听写输入，即语音识别接口。

(3)Voice Text API。完成从文字到语音的转换，即语音合成。

(4)Voice Telephone API。语音识别和语音合成综合运用到电话系统之上，利用此接口可以建立一个电话应答系统，甚至可以通过电话控制计算机。

(5)Audio Objects API。封装了计算机发音系统。

上面这一串可能已经令你头晕目眩了，列出来的目的不是说你要知道这么多，而是当你需要搜寻资料的时候可以用相应的关键词去找。

在微软的这套 SDK 中，最关键的是 SpVoice 类，通过这个类声明出来的对象调用 TTS 引擎，可以实现朗读功能，SpVoice 类主要有属性有以下的：

Voice：表示发音类型，相当于进行朗读的人，通常我们可以通过安装相应的语音引擎来增加相应的语音。

Rate：语音朗读速度，取值范围为-10 到+10。数值越大，速度越快。

Volume：音量，取值范围为 0 到 100。数值越大，音量越大。

SpVoice 主要方法有以下几种：

Speak()：完成将文本信息转换为语音并按照指定的参数进行朗读，该方法有 Text 和 Flags 两个参数，分别指定要朗读的文本和朗读方式(同 步或异步等)。

GetVoices()：获取系统中的语音，用于指定 SpVoice 的 Voice 属性。

Pause()：暂停使用该对象的所有朗读进程。该方法没有参数。

Resume()：恢复该对象所对应的被暂停的朗读进程。该方法没有参数。

没有标该方法没有参数的都是有参数的。

三、实验内容

研究如何在程序中使用 Microsoft Speech SDK（可以从微软网站上自由下载）的 TTS（Text To Speech）功能，并编写 一个中英文文本阅读器软件。也可以设计一个基于 TTS 的更具想象力的应用软件。将研究结果和程序设计方法总结成实验报告。

研究如何在程序中使用 Microsoft Speech SDK（可以从微软网站上自由下载）的 SR（语音识别）功能，并编写一个基于 SR 的应用软件。将研究结果和程序设计方法总结成实验报告。

。

四、实验工具

Python

百度 aip AI

五、实验步骤

1.首先注册一个百度 AI 平台账号，完成验证后申请一个支持语音识别和语音合成的新应用，记下返回的 AppID，API Key，Secret Key（由于我们采用的免费体验版本，会有一些次数和长度的限制，但是在测试阶段各个功能完全可行）

2.配置环境库，编写后端代码和简单的 html 交互文件

语音合成提供了百度 API 支持的 7 中声线，语音识别支持普通话，英语，粤语，重庆方言，以及 WAV, PCM, AMR, M4A 四种文件格式

App.py:

```
from flask import Flask, render_template, request, send_file, jsonify
from aip import AipSpeech
import os
import uuid
from datetime import datetime
import wave
import contextlib

app = Flask(__name__)

# 百度 AI 配置 - 请替换为你的实际密钥
APP_ID = '你的 APP_ID'
API_KEY = '你的 API_KEY'
SECRET_KEY = '你的 SECRET_KEY'

# 初始化 AipSpeech 对象
client = AipSpeech(APP_ID, API_KEY, SECRET_KEY)

# 确保目录存在
AUDIO_DIR = os.path.join(os.path.dirname(__file__), 'static', 'audio')
UPLOAD_DIR = os.path.join(os.path.dirname(__file__), 'static', 'uploads')
os.makedirs(AUDIO_DIR, exist_ok=True)
os.makedirs(UPLOAD_DIR, exist_ok=True)

# 发音人选项
VOICE_OPTIONS = {
    0: "女声",
    1: "男声",
    3: "情感合成-度逍遥",
    4: "情感合成-度丫丫",
    5: "情感合成-度小娇",
```

```
    103: "情感合成-度米朵",
    106: "情感合成-度博文"
}

# 语音识别语言选项
LANGUAGE_OPTIONS = {
    1537: "普通话",
    1737: "英语",
    1637: "粤语",
    1837: "四川话"
}


def get_audio_duration(file_path):
    """获取音频文件时长"""
    try:
        with contextlib.closing(wave.open(file_path, 'r')) as f:
            frames = f.getnframes()
            rate = f.getframerate()
            duration = frames / float(rate)
            return duration
    except:
        return 0


@app.route('/')
def index():
    return render_template('index.html',
                           voice_options=VOICE_OPTIONS,
                           language_options=LANGUAGE_OPTIONS)


# 现有的语音合成路由
@app.route('/synthesize', methods=['POST'])
def synthesize_speech():
    try:
        data = request.json
        text = data.get('text', '')
        speed = int(data.get('speed', 5))
        pitch = int(data.get('pitch', 5))
        volume = int(data.get('volume', 5))
        person = int(data.get('person', 0))

        if len(text) > 1024:
            return jsonify({
                'success': False,
```

```
                'message': '文本长度不能超过 1024 个字符'
            })

        filename                                                                    =
f"{datetime.now().strftime('%Y%m%d%H%M%S')}_{uuid.uuid4().hex[:8]}.mp3"
        filepath = os.path.join(AUDIO_DIR, filename)

        result = client.synthesis(
            text,
            'zh',
            1,
            {
                'spd': speed,
                'pit': pitch,
                'vol': volume,
                'per': person
            }
        )

        if not isinstance(result, dict):
            with open(filepath, 'wb') as f:
                f.write(result)

            return jsonify({
                'success': True,
                'message': '语音合成成功',
                'audio_url': f'/static/audio/{filename}'
            })
        else:
            return jsonify({
                'success': False,
                'message': f'合成失败: {result}'
            })

    except Exception as e:
        return jsonify({
            'success': False,
            'message': f'服务器错误: {str(e)}'
        })


# 新增语音识别路由
@app.route('/recognize', methods=['POST'])
def recognize_speech():
    try:
        # 检查是否有文件上传
```

```
        if 'audio_file' not in request.files:
            return jsonify({
                'success': False,
                'message': '没有上传文件'
            })

        audio_file = request.files['audio_file']

        # 检查文件名
        if audio_file.filename == '':
            return jsonify({
                'success': False,
                'message': '未选择文件'
            })

        # 检查文件格式
        allowed_extensions = {'wav', 'pcm', 'amr', 'm4a'}
        file_extension = audio_file.filename.rsplit('.', 1)[1].lower() if '.' in
audio_file.filename else ''

        if file_extension not in allowed_extensions:
            return jsonify({
                'success': False,
                'message': f'不支持的文件格式。请上传以下格式：{",
".join(allowed_extensions)}'
            })

        # 保存上传的文件
        filename                                                                =
f"{datetime.now().strftime('%Y%m%d%H%M%S')}_{uuid.uuid4().hex[:8]}.{file_extension}"
        filepath = os.path.join(UPLOAD_DIR, filename)
        audio_file.save(filepath)

        # 获取识别语言参数
        language = request.form.get('language', '1537')

        # 读取音频文件
        def get_file_content(file_path):
            with open(file_path, 'rb') as fp:
                return fp.read()

        # 调用百度语音识别 API
        result = client.asr(
            get_file_content(filepath),
            'wav' if file_extension == 'wav' else file_extension,
            16000,  # 采样率
```

```
            {
                'dev_pid': int(language),  # 语言模型
            }
        )

        # 处理识别结果
        if result['err_no'] == 0:
            recognized_text = result['result'][0]

            # 获取音频时长
            duration = get_audio_duration(filepath)

            return jsonify({
                'success': True,
                'message': '语音识别成功',
                'text': recognized_text,
                'duration': round(duration, 2),
                'file_size': os.path.getsize(filepath)
            })
        else:
            error_msg = result['err_msg']
            # 删除上传的文件
            if os.path.exists(filepath):
                os.remove(filepath)

            return jsonify({
                'success': False,
                'message': f'识别失败: {error_msg}'
            })

    except Exception as e:
        return jsonify({
            'success': False,
            'message': f'服务器错误: {str(e)}'
        })


# 音频文件服务路由
@app.route('/static/audio/<filename>')
def serve_audio(filename):
    return send_file(os.path.join(AUDIO_DIR, filename))


@app.route('/static/uploads/<filename>')
def serve_upload(filename):
    return send_file(os.path.join(UPLOAD_DIR, filename))
```

```
if __name__ == '__main__':
    app.run(debug=True, host='0.0.0.0', port=5000)
```

html 文件：

```
<!DOCTYPE html>
<html lang="zh-CN">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>语音合成与识别系统</title>
    <style>
        * {
            box-sizing: border-box;
            margin: 0;
            padding: 0;
            font-family: 'Arial', sans-serif;
        }

        body {
            background-color: #f5f5f5;
            padding: 20px;
            line-height: 1.6;
        }

        .container {
            max-width: 900px;
            margin: 0 auto;
            background: white;
            padding: 30px;
            border-radius: 10px;
            box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
        }

        h1 {
            text-align: center;
            margin-bottom: 30px;
            color: #333;
        }

        .tabs {
            display: flex;
            margin-bottom: 20px;
            border-bottom: 1px solid #ddd;
```

```
        }

        .tab {
            padding: 10px 20px;
            cursor: pointer;
            border: 1px solid transparent;
            border-bottom: none;
            border-radius: 5px 5px 0 0;
            margin-right: 5px;
        }

        .tab.active {
            background-color: #4CAF50;
            color: white;
        }

        .tab-content {
            display: none;
        }

        .tab-content.active {
            display: block;
        }

        .form-group {
            margin-bottom: 20px;
        }

        label {
            display: block;
            margin-bottom: 8px;
            font-weight: bold;
            color: #555;
        }

        textarea {
            width: 100%;
            height: 120px;
            padding: 10px;
            border: 1px solid #ddd;
            border-radius: 5px;
            resize: vertical;
            font-size: 16px;
        }

        .slider-container {
```

```
            display: flex;
            align-items: center;
            gap: 10px;
        }

        .slider-container input[type="range"] {
            flex: 1;
        }

        .slider-value {
            min-width: 30px;
            text-align: center;
            font-weight: bold;
        }

        select, input[type="file"] {
            width: 100%;
            padding: 10px;
            border: 1px solid #ddd;
            border-radius: 5px;
            font-size: 16px;
        }

        button {
            display: block;
            width: 100%;
            padding: 12px;
            background-color: #4CAF50;
            color: white;
            border: none;
            border-radius: 5px;
            font-size: 16px;
            cursor: pointer;
            transition: background-color 0.3s;
        }

        button:hover {
            background-color: #45a049;
        }

        button:disabled {
            background-color: #cccccc;
            cursor: not-allowed;
        }

        .result {
```

```
            margin-top: 20px;
            padding: 15px;
            border-radius: 5px;
            display: none;
        }

        .success {
            background-color: #dff0d8;
            border: 1px solid #d6e9c6;
            color: #3c763d;
        }

        .error {
            background-color: #f2dede;
            border: 1px solid #ebccd1;
            color: #a94442;
        }

        .audio-player {
            margin-top: 15px;
            width: 100%;
        }

        .loading {
            display: none;
            text-align: center;
            margin-top: 10px;
        }

        .loading-spinner {
            border: 4px solid #f3f3f3;
            border-top: 4px solid #3498db;
            border-radius: 50%;
            width: 30px;
            height: 30px;
            animation: spin 2s linear infinite;
            margin: 0 auto;
        }

        .file-info {
            margin-top: 10px;
            padding: 10px;
            background-color: #f9f9f9;
            border-radius: 5px;
            display: none;
        }
```

```
        @keyframes spin {
            0% { transform: rotate(0deg); }
            100% { transform: rotate(360deg); }
        }
    </style>
</head>
<body>
    <div class="container">
        <h1>语音合成与识别系统</h1>

        <div class="tabs">
            <div class="tab active" data-tab="synthesis">语音合成</div>
            <div class="tab" data-tab="recognition">语音识别</div>
        </div>

        <!-- 语音合成标签页 -->
        <div class="tab-content active" id="synthesis-tab">
            <div class="form-group">
                <label for="text">输入要合成的文本：</label>
                <textarea id="text" placeholder="请输入要转换为语音的文本（最多 1024
个字符）"></textarea>
                <div id="char-count" style="text-align: right; margin-top: 5px; color:
#666;">0/1024</div>
            </div>

            <div class="form-group">
                <label for="speed">语速：</label>
                <div class="slider-container">
                    <input type="range" id="speed" min="0" max="9" value="5">
                    <span class="slider-value" id="speed-value">5</span>
                </div>
            </div>

            <div class="form-group">
                <label for="pitch">音调：</label>
                <div class="slider-container">
                    <input type="range" id="pitch" min="0" max="9" value="5">
                    <span class="slider-value" id="pitch-value">5</span>
                </div>
            </div>

            <div class="form-group">
                <label for="volume">音量：</label>
                <div class="slider-container">
                    <input type="range" id="volume" min="0" max="15" value="5">
```

```
                <span class="slider-value" id="volume-value">5</span>
            </div>
        </div>

        <div class="form-group">
            <label for="person">发音人：</label>
            <select id="person">
                {% for key, value in voice_options.items() %}
                <option value="{{ key }}">{{ value }}</option>
                {% endfor %}
            </select>
        </div>

        <button id="synthesize-btn">合成语音</button>

        <div class="loading" id="synthesis-loading">
            <div class="loading-spinner"></div>
            <p>正在合成语音，请稍候...</p>
        </div>

        <div class="result" id="synthesis-result"></div>
    </div>

    <!-- 语音识别标签页 -->
    <div class="tab-content" id="recognition-tab">
        <div class="form-group">
            <label for="audio_file">选择音频文件：</label>
            <input type="file" id="audio_file" accept=".wav,.pcm,.amr,.m4a">
            <div style="margin-top: 5px; color: #666; font-size: 14px;">
                支持格式: WAV, PCM, AMR, M4A
            </div>
        </div>

        <div class="form-group">
            <label for="recognition-language">识别语言：</label>
            <select id="recognition-language">
                {% for key, value in language_options.items() %}
                <option value="{{ key }}">{{ value }}</option>
                {% endfor %}
            </select>
        </div>

        <button id="recognize-btn">识别语音</button>

        <div class="loading" id="recognition-loading">
            <div class="loading-spinner"></div>
```

```
                <p>正在识别语音，请稍候...</p>
            </div>

            <div class="file-info" id="file-info"></div>

            <div class="result" id="recognition-result"></div>
        </div>
    </div>

    <script>
        document.addEventListener('DOMContentLoaded', function() {
            // 标签页切换
            const tabs = document.querySelectorAll('.tab');
            const tabContents = document.querySelectorAll('.tab-content');

            tabs.forEach(tab => {
                tab.addEventListener('click', function() {
                    const tabId = this.getAttribute('data-tab');

                    // 移除所有 active 类
                    tabs.forEach(t => t.classList.remove('active'));
                    tabContents.forEach(tc => tc.classList.remove('active'));

                    // 添加 active 类到当前标签
                    this.classList.add('active');
                    document.getElementById(`${tabId}-tab`).classList.add('active');
                });
            });

            // 语音合成相关代码
            const textInput = document.getElementById('text');
            const charCount = document.getElementById('char-count');
            const speedSlider = document.getElementById('speed');
            const speedValue = document.getElementById('speed-value');
            const pitchSlider = document.getElementById('pitch');
            const pitchValue = document.getElementById('pitch-value');
            const volumeSlider = document.getElementById('volume');
            const volumeValue = document.getElementById('volume-value');
            const personSelect = document.getElementById('person');
            const synthesizeBtn = document.getElementById('synthesize-btn');
            const synthesisLoading = document.getElementById('synthesis-loading');
            const synthesisResult = document.getElementById('synthesis-result');

            // 更新字符计数
            textInput.addEventListener('input', function() {
                const count = textInput.value.length;
```

```
            charCount.textContent = `${count}/1024`;

            if (count > 1024) {
                charCount.style.color = 'red';
            } else {
                charCount.style.color = '#666';
            }
        });

        // 更新滑块值显示
        speedSlider.addEventListener('input', function() {
            speedValue.textContent = speedSlider.value;
        });

        pitchSlider.addEventListener('input', function() {
            pitchValue.textContent = pitchSlider.value;
        });

        volumeSlider.addEventListener('input', function() {
            volumeValue.textContent = volumeSlider.value;
        });

        // 语音合成
        synthesizeBtn.addEventListener('click', function() {
            const text = textInput.value.trim();

            if (!text) {
                showResult('请输入要合成的文本', false, synthesisResult);
                return;
            }

            if (text.length > 1024) {
                showResult('文本长度不能超过 1024 个字符', false,
synthesisResult);
                return;
            }

            // 显示加载动画，禁用按钮
            synthesisLoading.style.display = 'block';
            synthesizeBtn.disabled = true;
            synthesisResult.style.display = 'none';

            // 发送请求到后端
            fetch('/synthesize', {
                method: 'POST',
                headers: {
```

```
                    'Content-Type': 'application/json'
                },
                body: JSON.stringify({
                    text: text,
                    speed: speedSlider.value,
                    pitch: pitchSlider.value,
                    volume: volumeSlider.value,
                    person: personSelect.value
                })
            })
            .then(response => response.json())
            .then(data => {
                // 隐藏加载动画，启用按钮
                synthesisLoading.style.display = 'none';
                synthesizeBtn.disabled = false;

                if (data.success) {
                    showResult(
                        `${data.message}<br>
                        <audio class="audio-player" controls>
                            <source src="${data.audio_url}" type="audio/mpeg">
                            您的浏览器不支持音频播放
                        </audio>
                        <br>
                        <a href="${data.audio_url}" download>下载音频文件</a>`,
                        true,
                        synthesisResult
                    );
                } else {
                    showResult(data.message, false, synthesisResult);
                }
            })
            .catch(error => {
                // 隐藏加载动画，启用按钮
                synthesisLoading.style.display = 'none';
                synthesizeBtn.disabled = false;
                showResult('网络错误: ' + error.message, false, synthesisResult);
            });
        });

        // 语音识别相关代码
        const audioFileInput = document.getElementById('audio_file');
        const languageSelect = document.getElementById('recognition-language');
        const recognizeBtn = document.getElementById('recognize-btn');
        const                    recognitionLoading                    =
document.getElementById('recognition-loading');
```

```
            const recognitionResult = document.getElementById('recognition-result');
            const fileInfo = document.getElementById('file-info');

            // 显示文件信息
            audioFileInput.addEventListener('change', function() {
                const file = this.files[0];
                if (file) {
                    fileInfo.innerHTML = `
                        <strong>文件名:</strong> ${file.name}<br>
                        <strong>文件大小:</strong> ${(file.size / 1024).toFixed(2)}
KB<br>
                        <strong>文件类型:</strong> ${file.type || '未知'}
                    `;
                    fileInfo.style.display = 'block';
                } else {
                    fileInfo.style.display = 'none';
                }
            });

            // 语音识别
            recognizeBtn.addEventListener('click', function() {
                const file = audioFileInput.files[0];

                if (!file) {
                    showResult('请选择音频文件', false, recognitionResult);
                    return;
                }

                // 显示加载动画，禁用按钮
                recognitionLoading.style.display = 'block';
                recognizeBtn.disabled = true;
                recognitionResult.style.display = 'none';

                // 创建表单数据
                const formData = new FormData();
                formData.append('audio_file', file);
                formData.append('language', languageSelect.value);

                // 发送请求到后端
                fetch('/recognize', {
                    method: 'POST',
                    body: formData
                })
                .then(response => response.json())
                .then(data => {
                    // 隐藏加载动画，启用按钮
```

```
                    recognitionLoading.style.display = 'none';
                    recognizeBtn.disabled = false;

                    if (data.success) {
                        let resultHTML = `${data.message}<br><br>`;
                        resultHTML += `<strong>识别结果:</strong> ${data.text}<br>`;

                        if (data.duration) {
                            resultHTML += `<strong>音频时长:</strong> ${data.duration} 秒<br>`;
                        }

                        if (data.file_size) {
                            resultHTML += `<strong>文件大小:</strong> ${(data.file_size / 1024).toFixed(2)} KB`;
                        }

                        showResult(resultHTML, true, recognitionResult);
                    } else {
                        showResult(data.message, false, recognitionResult);
                    }
                })
                .catch(error => {
                    // 隐藏加载动画，启用按钮
                    recognitionLoading.style.display = 'none';
                    recognizeBtn.disabled = false;
                    showResult('网络错误: ' + error.message, false, recognitionResult);
                });
            });

            // 显示结果
            function showResult(message, isSuccess, resultElement) {
                resultElement.innerHTML = message;
                resultElement.className = isSuccess ? 'result success' : 'result error';
                resultElement.style.display = 'block';
            }
        });
    </script>
</body>
</html>
```

3.测试

语音合成部分测试了中英文不同字符，语音识别主要做了 wav 文件格式的测试，运行良好

六、实验结果及分析

效果图：

语音合成：

在实际运行过程中还可以选择不同的语速和声调，音量，发音人即声线选择，包括以下七种：

女声

男声

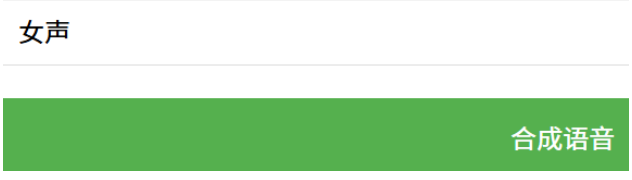
情感合成-度逍遥

情感合成-度丫丫

情感合成-度小娇

情感合成-度米朵

情感合成-度博文



语音合成功能总体而言比较好，能够生成中，英文的声音和不同声线的声音

语音识别：

而识别功能在测试过程中发现较差，一方面由于免费的申请限制比较多，如 wav 文件不能压缩所以会时常导致文件过大而无法识别，另一方面，识别的内容也不是很准确