实验 10：使用公有云实现智能化功能

一、实验目的

1、理解公有云 API 服务的基本概念和调用流程。

2、掌握在 Java 应用程序中集成腾讯云语音识别 SDK 的方法。

3、能够通过编程调用云端 AI 服务，实现本地音频文件的文字转写功能。

4、培养学生将云服务与现有系统结合，实现功能增强的工程实践能力。

二、实验学时

2 学时

三、实验类型

综合性

四、实验需求

1、硬件

每人配备计算机 1 台，建议优先使用个人计算机开展实验。

2、软件

安装 IntelliJ IDEA Community。

3、网络

本地主机能够访问互联网和实验中心网络。

4、工具

无。

五、实验任务

1、注册腾讯云账号并开通语音识别服务。

2、在 Java 项目中调用腾讯云 SDK，编写代码实现语音识别功能。

3、结合现有系统功能将语音识别与系统功能相结合。

六、实验内容及步骤

1、注册腾讯云账号并开通服务

步骤 1： 访问腾讯云官网（https://cloud.tencent.com/）并注册个人账号。完成实名认证（通常可选择微信扫码认证）。

图 1 腾讯云官网

图 2 语音识别产品页

步骤 2： 登录腾讯云控制台，在顶部搜索栏中搜索“语音识别”，进入产品页面。

步骤 3： 点击“立即使用”或“开通服务”按钮，开通语音识别服务。首次开通通常会有免费资源包，如“新客 0 元体验包”，请留意领取。

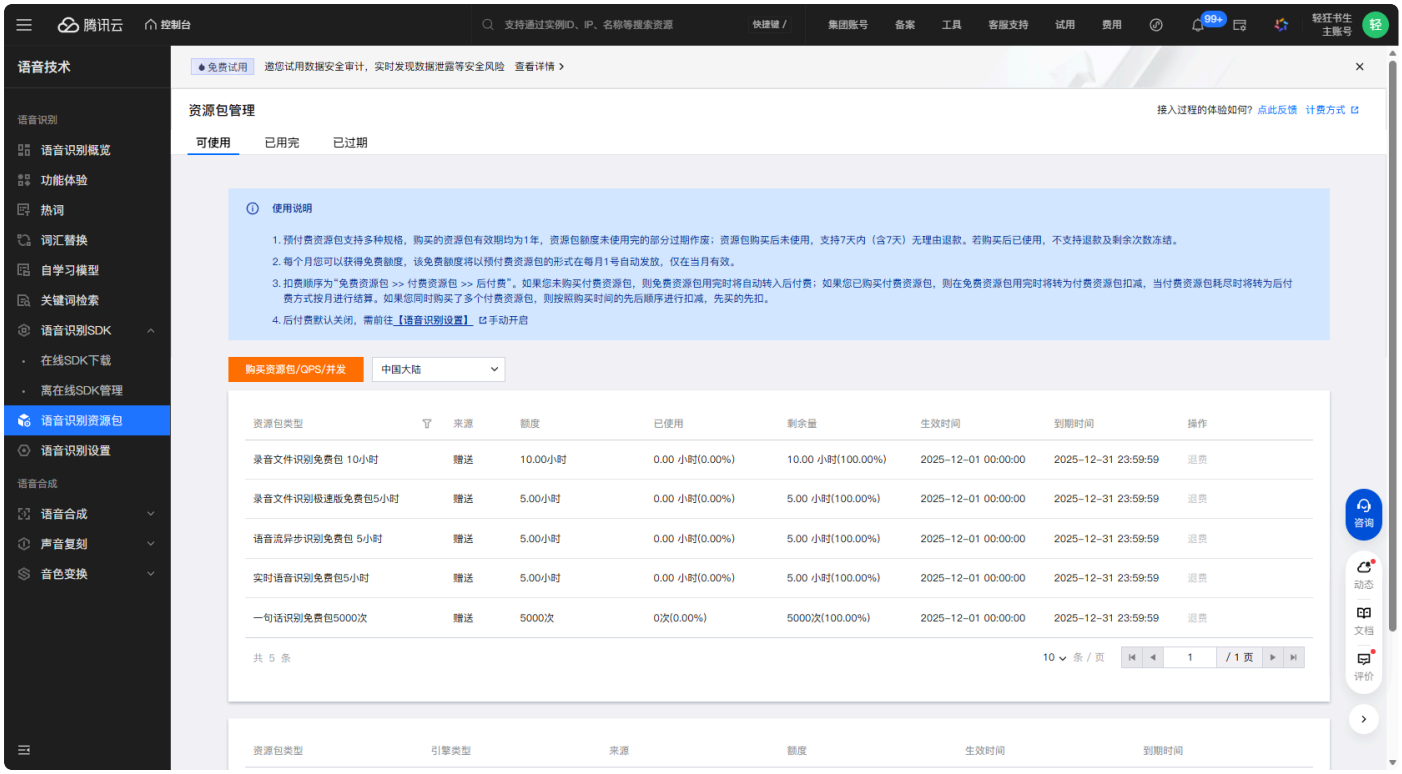

图 3 腾讯云提供的免费额度

2、调用公有云实现语音识别

步骤 1：获取 API 密钥

（1）进入腾讯云控制台（https://console.cloud.tencent.com/cam/capi），访问 API 密钥管理页面。

（2）创建一个新的密钥或使用现有密钥。**请妥善保管 SecretId 和 SecretKey，切勿上传到代码仓库。**

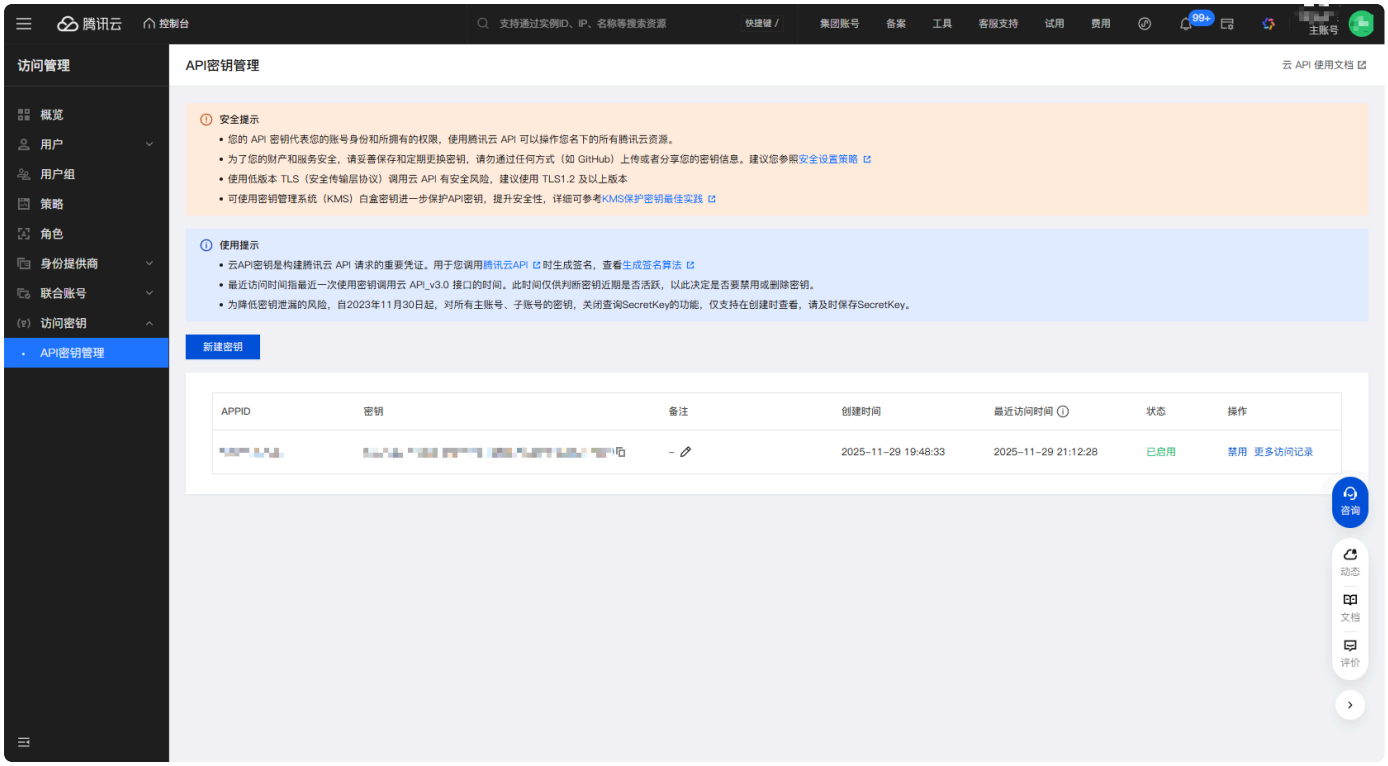

图 4 获取 API 密钥

步骤 2：创建 Java 项目并引入 SDK

> 腾讯云语音识别支持通过 SDK 接入和通过 API 接入两种方式，推荐通过 SDK 接入，因为语音识别涉及音频传输，通过 SDK 接入更简单、便捷，可省去鉴权、协议对接过程。
>
> - 通过 SDK 接入：https://cloud.tencent.com/document/product/1093/52554
> - 通过 API 接入：https://cloud.tencent.com/document/product/1093/35637

（1）在 IntelliJ IDEA 中创建一个新的 Maven 项目。

（2）打开 pom.xml 文件，添加腾讯云语音识别（ASR）SDK 的依赖。需要在腾讯云官方文档中找到最新版本的依赖配置。

XML/HTML

```xml
<dependencies>
    <!-- 示例依赖，请以腾讯云官方文档为准 -->
    <dependency>
        <groupId>com.tencentcloudapi</groupId>
        <artifactId>tencentcloud-sdk-java-asr</artifactId>
        <version>最新版本号</version>
    </dependency>
</dependencies>
```

（3）点击 IDEA 的 Maven 面板的刷新按钮，以下载相关依赖。

步骤 3：编写 Java 代码调用识别服务

（1）在项目中创建一个类，例如 TencentASRDemo。

（2）编写代码，核心流程包括：

- 实例化客户端： 使用 SecretId 和 SecretKey 创建认证对象和客户端对象。
- 构建请求参数： 创建识别请求对象，设置必要的参数，如引擎类型（EngineModelType，例如 16k_zh 表示 16k 采样率的中文普通话）、音频文件的本地路径或 URL。
- 发送请求并获取响应： 通过客户端对象发送请求，并接收响应对象。
- 处理响应结果： 从响应对象中解析出识别出的文本。

示例代码框架如下（仅供参考，请以官方 SDK 文档为准）。

Java

```java
import com.tencentcloudapi.common.Credential;
import com.tencentcloudapi.common.profile.ClientProfile;
import com.tencentcloudapi.common.profile.HttpProfile;
import com.tencentcloudapi.asr.v20190614.AsrClient;
import com.tencentcloudapi.asr.v20190614.models.*;

import java.io.File;
import java.nio.file.Files;
import java.util.Base64;
import java.util.Scanner;

public class TencentASRDemo {

    // 请替换为您的实际密钥
    private static final String SECRET_ID = "您的SecretId";
    private static final String SECRET_KEY = "您的SecretKey";
    private static final String REGION = "ap-beijing"; // 地域

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        try {
            System.out.println("=== 腾讯云语音识别演示程序 ===");
            System.out.print("请输入音频文件路径: ");
            String audioPath = scanner.nextLine();

            // 检查文件是否存在
            File audioFile = new File(audioPath);
            if (!audioFile.exists()) {
                System.out.println("错误: 文件不存在! ");
                return;
            }

            System.out.println("开始语音识别处理...");

            // 1. 实例化认证对象
            Credential cred = new Credential(SECRET_ID, SECRET_KEY);

            // 2. 实例化HTTP和客户端配置对象
```

```
            HttpProfile httpProfile = new HttpProfile();
            httpProfile.setEndpoint("asr.tencentcloudapi.com");
            ClientProfile clientProfile = new ClientProfile();
            clientProfile.setHttpProfile(httpProfile);

            // 3. 实例化ASR客户端
            AsrClient client = new AsrClient(cred, REGION, clientProfil
e);

            // 4. 读取音频文件并转换为Base64
            byte[] audioDataBytes = Files.readAllBytes(audioFile.toPath
());
            String audioData = Base64.getEncoder().encodeToString(audio
DataBytes);

            System.out.println("音频文件大小: " + audioDataBytes.length
+ " 字节");

            // 5. 实例化请求对象，设置参数
            CreateRecTaskRequest req = new CreateRecTaskRequest();
            req.setEngineModelType("16k_zh"); // 16k中文普通话模型
            req.setChannelNum(1L); // 单声道
            req.setResTextFormat(0L); // 识别结果文本格式
            req.setSourceType(1L); // 语音数据源: 语音数据
            req.setData(audioData);
            req.setDataLen((long) audioDataBytes.length);

            // 可选: 设置热词（提高特定词汇识别准确率）
            // req.setHotwordId("您的热词表ID");

            // 6. 发送识别请求
            System.out.println("正在提交识别任务...");
            CreateRecTaskResponse resp = client.CreateRecTask(req);
            String taskId = resp.getData().getTaskId();
            System.out.println("任务创建成功，任务ID: " + taskId);

            // 7. 轮询查询识别结果
            System.out.println("等待识别结果...");
            String recognitionResult = pollRecognitionResult(client, ta
skId);

```

```
76              // 8. 输出最终结果
77              System.out.println("\n=== 语音识别结果 ===");
78              System.out.println(recognitionResult);
79
80          } catch (Exception e) {
81              System.err.println("处理过程中发生错误:");
82              e.printStackTrace();
83          } finally {
84              scanner.close();
85          }
86      }
87
88      /**
89       * 轮询查询识别结果
90       * @param client ASR客户端
91       * @param taskId 任务ID
92       * @return 识别结果文本
93       */
94      private static String pollRecognitionResult(AsrClient client, Strin
    g taskId) {
95          int maxRetry = 30; // 最大重试次数
96          int retryInterval = 2000; // 重试间隔2秒
97
98          for (int i = 0; i < maxRetry; i++) {
99              try {
100                 // 等待一段时间再查询
101                 Thread.sleep(retryInterval);
102
103                 // 创建查询请求
104                 DescribeTaskStatusRequest statusReq = new DescribeTaskS
    tatusRequest();
105                 statusReq.setTaskId(taskId);
106
107                 // 发送查询请求
108                 DescribeTaskStatusResponse statusResp = client.Describe
    TaskStatus(statusReq);
109                 TaskStatus data = statusResp.getData();
110                 int status = data.getStatus().intValue();
111
112                 System.out.println("查询进度: " + (i + 1) + "/" + maxRetr
    y +
```

```
                                    ", 任务状态: " + getStatusText(status));

                // 根据状态码处理
                if (status == 2) { // 成功
                    return data.getResult() != null ? data.getResult()
    : "识别结果为空";
                } else if (status == 1) { // 排队中/识别中
                    continue; // 继续轮询
                } else if (status == 3) { // 失败
                    return "识别失败，错误信息: " + data.getErrorMsg();
                }

            } catch (Exception e) {
                System.err.println("查询任务状态时发生错误: " + e.getMessag
    e());
                return "查询失败: " + e.getMessage();
            }
        }

        return "识别超时，请稍后手动查询任务ID: " + taskId;
    }

    /**
     * 获取状态码对应的文本描述
     * @param status 状态码
     * @return 状态描述
     */
    private static String getStatusText(int status) {
        switch (status) {
            case 0: return "等待处理";
            case 1: return "识别中";
            case 2: return "识别完成";
            case 3: return "识别失败";
            default: return "未知状态(" + status + ")";
        }
    }
}
```

3、结合系统实现智能增强

步骤 1：功能设计

结合系统功能实现基于语音识别的智能增强，例如记录患者主诉信息的时候，支持通过录音的方式记录主诉，然后调用腾讯云人工智能 - 语音识别接口实现内容识别辅助记录，并可通过录音文件识别请求中的【情绪识别能力】（https://cloud.tencent.com/document/product/1093/37823）获取患者提供主诉时候的情绪能量值。

步骤 2：编码实现

（1）引入腾讯云官方 SDK 依赖。建立项目基础结构，配置 API 认证信息。创建配置文件管理各类参数，确保系统的灵活性和安全性。

（2）重点实现语音识别服务的调用封装，包括音频文件读取、Base64 编码、API 请求构建等功能。需要正确处理各种音频格式，设置合适的超时机制和重试策略，保证服务的稳定性。

（3）开发业务处理逻辑，包括语音内容识别、情绪分析结果解析等。需要设计合理的数据结构来存储和传递分析结果，确保信息的完整性和准确性。

（4）实现友好的命令行交互界面，提供清晰的进度反馈和状态提示。优化结果展示格式，使主诉记录和情绪分析结果易于阅读和理解。同时需要完善错误处理和异常提示机制。

步骤 3：测试优化

验证每个功能模块的正确性。重点测试音频处理、API 调用、情绪分析等核心功能。确保系统在各种情况下都能正常工作，特别是边界情况和异常场景。

七、实验考核

1、本课程实验考核方案

本课程实验考核采用【实验智能评】【实验随堂查】方式开展，根据不同的实验内容选择不同的考核方式。

【实验智能评】：实验完成后提交 GitLab，通过自动化代码评审工具进行评分。

【实验随堂查】：在实验课上通过现场演示的方式向实验指导教师进行汇报，并完成现场问答交流。

2、本实验考核要求

本实验考核方式：实验智能评

实验 10-12 作为本课程第 3 次实验考核。

考核要求：

（1）学生通过 GitLab 提交实验成果：{此部分说明需要提交的内容}。

（2）由 GitLab 根据成果和交流情况综合评分。