离线语音识别

一、离线语音识别优势

- 支持超过100个国家语言的实时同声传译，同步输出和客户母语对应的实时同传翻译文本和语音合成数据
- **业界领先的声纹识别，返回结果字段中精准识别说话人ID**
- **庞大的声纹数据库，海量音视频文件中毫秒级返回说话人的片段音视频信息**
- **业界领先的降噪算法，返回的音视频文件，可包含降噪后的清晰语音**
- **业界领先的高精准中英文等语音识别算法**

二、离线语音识别API文档

1、接口说明

离线语音识别为超过1分钟的音频而设计。内置标点，说话人分离，语气词过滤等功能，并能通过规则替换的功能实现禁忌词替换。

1.1 支持的音频格式有 WAV/MP3/WMA/FLAC/AMR/OPUS/M4A/AAC

1.2 支持音频详情：

- 音频时长：不超过5小时
- 音频大小：不超过600 MB
- 音频声道：单声道、双声道
- 采样率：支持8000、16000、44100和48000 Hz
- 采样精度：16 bits、8 bits
- 编码格式：文件格式和音频解码需结合查看，如下表第一行：只支持pcm封装或wav封装的pcm音频，不支持其他文件格式封装的pcm音频。 不同的音频解码格式需在请求体中填写不同的aue.

文件格式	音频解码	audio_encode
.wav	pcm	pcm
.m4a .aac	aac	aac
.mp3	mpeg2	mpeg2
.opus	opus	opus
.flac	flac	flac

注意
a. opus编码的音频不支持44100 Hz的采样率
b. 仅 wav封装的pcm格式音频支持 8 bits采样精度

2 长语音转写-创建转写任务

- 接口描述： 使用音频文件地址创建音频转写任务。
- 请求格式： application/json
- 请求方式：POST
- URL: https://asr-prod.abcpen.com/v1/asr/long

2. 1 请求参数

2.1.1 Header

Header中输入正确的账号、时间戳及签名（X-App-Key、X-Timestamp、X-App-Signature），鉴权方式

2.1.2 Body体

Body

参数名	类型	是否必须	描述
audio_url	string	是	文件的URL，如：https://zos.abcpen.com/tts/zmeet/20221023/3058bca8-52cb-11ed-961e-00155dc6cbed.mp3
sd	string	否	说话人区分；如使用说话人区分，设置为True，否则设置为False
callback	string	否	支持http / https。填写后可通过回调获取结果。为空可使用查询获取结果。
audio_encode	string	否	设置关于文件的编码和码率等。
speaker_number	integer	否	说话人分离功能默认为2，两位位说话人； 0 表示盲分。2-4时，表示指定了说话人的数量，返回speakerId。
words_output	string	否	逐字输出开关。 默认false，逐句输出。 true，开启后带有逐字模式。
audio_sample_rate	interger	否	采样频率, 支持选项有： ["8000", "16000", "44100", "48000"]， 从中任选一个符合客户输入的语音采样频率值

2.1.3 返回数据实例

- 创建任务返回结果：返回包含任务task_id的JSON字符串

```
{"code":"0","data":{"task_id":"8ba73ba7-08e3-4590-a379-5f77f7b6508d"},"msg":"success"}
```

3、查询任务

- 接口描述： 使用音频文件地址创建音频转写任务。
- 请求格式： application/json
- 请求方式：POST
- URL: https://asr-prod.abcpen.com/v1/asr/long

3. 1 请求参数

3.1.1 Header

Header中输入正确的账号、时间戳及签名（X-App-Key、X-Timestamp、X-App-Signature），鉴权方式参考下述“signa生成"

3.1.2 Body体

Body

参数名	类型	是否必须	描述
task_id	string	是	离线语音识别的任务id号，如 “8ba73ba7-08e3-4590-a379-5f77f7b6508d”

3.1.3 返回结果实例

- 返回的是JSON字符串：

{'code': '0', 'data': {'data': {'speechResult': {'onebest': '除常规的宠物食品，宠物用具外，宠物殡葬，宠物医疗，宠物保险，宠物服务，线下门店等在外界看来颇为小众的领域，也陆续拿到动辄上千万甚至过亿的投资。近两年，我国一人户家庭数超1.25亿，占比超25%。 2021年，我国饲养猫狗的人群超6800万，其中有近一半 是90后宠物主。当传统家庭结构发生变化，越来越多年轻人把情感倾注到宠物身上，宠物从单纯的陪伴者晋升为亲密家人，而这些毛孩子们也像人类幼崽一样，有着吃喝拉撒睡，医疗，托管，保险等各种需求，愿意为之付费的宠物主们就 这样浇灌出一个千亿消费市场。', 'duration': 53928, 'detail': [{'sentences': '除常规的宠物食品，', 'wordBg': '320', 'wordEd': '2320', 'speakerId': '1'}, {'sentences': '宠物用具外，', 'wordBg': '2320', 'wordEd': '3720', 'speakerId': '1'}, {'sentences': '宠物殡葬，', 'wordBg': '3720', 'wordEd': '4840', 'speakerId': '1'}, {'sentences': '宠物医疗，', 'wordBg': '4840', 'wordEd': '6080', 'speakerId': '1'}, {'sentences': ' 宠物保险，', 'wordBg': '6080', 'wordEd': '7280', 'speakerId': '1'}, {'sentences': '宠物服务，', 'wordBg': '7280', 'wordEd': '8000', 'speakerId': '1'}, {'sentences': '线下门店等在外界看来颇为小众的领域，', 'wordBg': '8000', 'wordEd': '11750', 'speakerId': '1'}, {'sentences': '也陆续拿到动辄上千万甚至过亿的投资。', 'wordBg': '11750', 'wordEd': '14930', 'speakerId': '0'}, {'sentences': '近两年，', 'wordBg': '15420', 'wordEd': '16460', 'speakerId': '0'}, {'sentences': '我国一人户家庭数超1.25亿，', 'wordBg': '16460', 'wordEd': '20340', 'speakerId': '0'}, {'sentences': '占比超25%。', 'wordBg': '20340', 'wordEd': '22300', 'speakerId': '0'}, {'sentences': '2021年，', 'wordBg': '22300', 'wordEd': '23060', 'speakerId': '0'}, {'sentences': '我国饲养猫狗的人群超6800万，', 'wordBg': '23060', 'wordEd': '26740', 'speakerId': '0'}, {'sentences': '其中有近一半是90后宠物主。', 'wordBg': '26740', 'wordEd': '29200', 'speakerId': '0'}, {'sentences': '当传统家庭结构发生变化，', 'wordBg': '29700', 'wordEd': '32300', 'speakerId': '0'}, {'sentences': '越来越多年轻人把情感倾注到宠物身上，', 'wordBg': '32300', 'wordEd': '36100', 'speakerId': '0'}, {'sentences': '宠物从单纯的陪伴者晋升为亲密家人，', 'wordBg': '36100', 'wordEd': '39780', 'speakerId': '0'}, {'sentences': '而这些毛孩子们也像人类幼崽一样，', 'wordBg': '39780', 'wordEd': '42640', 'speakerId': '0'}, {'sentences': '有着吃喝拉撒睡，', 'wordBg': '43070', 'wordEd': '44870', 'speakerId': '0'}, {'sentences': '医疗，', 'wordBg': '44870', 'wordEd': '45790', 'speakerId': '0'}, {'sentences': '托管，', 'wordBg': '45790', 'wordEd': '46510', 'speakerId': '1'}, {'sentences': '保险等各种需求，', 'wordBg': '46510', 'wordEd': '48270', 'speakerId': '1'}, {'sentences': '愿意为之付费的宠物主们就这样浇灌出一个千亿消费市场。', 'wordBg': '48270', 'wordEd': '53010', 'speakerId': '0'}]}}, 'task_id': '8ba73ba7-08e3-4590-a379-5f77f7b6508d', 'spk_id_text': '', 'audio_denoise_url': '', 'status': {'spk_id_status': 0, 'denoise_status': 0, 'asr_spk_status': 0}}, 'msg': 'success'}

4、 signa生成

a. python示例（）

```python
import hashlib
import hmac
import time
import base64

def get_signature_flytek(ts, app_id, app_secret):
    tt = (app_id + ts).encode('utf-8')
    md5 = hashlib.md5()
    md5.update(tt)
```

```
    baseString = md5.hexdigest()
    baseString = bytes(baseString, encoding='utf-8')

    apiKey = app_secret.encode('utf-8')
    signa = hmac.new(apiKey, baseString, hashlib.sha1).digest()
    signa = base64.b64encode(signa)
    signa = str(signa, 'utf-8')
    return signa
```

b. Java示例

- Java示例(具体参考github Java目录代码)

```
    // 生成握手参数
    public static String getHandShakeParams(String appId, String secretKey) {
        String ts = System.currentTimeMillis() / 1000 + "";
        String signa = "";
        try {
            signa = EncryptUtil.HmacSHA1Encrypt(EncryptUtil.MD5(appId + ts),
secretKey);
            return "?appid=" + appId + "&ts=" + ts + "&signa=" +
URLEncoder.encode(signa, "UTF-8");
        } catch (Exception e) {
            e.printStackTrace();
        }

        return "";
    }
```

- Java基础工具类

```
package com.abcpen.ai.rtasr.util;

import java.io.UnsupportedEncodingException;
import java.nio.charset.StandardCharsets;
import java.security.InvalidKeyException;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.security.SignatureException;

import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;

import org.apache.commons.codec.binary.Base64;

public class EncryptUtil {

    /**
     * 加密数字签名（基于HMACSHA1算法）
     *
     * @param encryptText
```

```
     * @param encryptKey
     * @return
     * @throws SignatureException
     */
    public static String HmacSHA1Encrypt(String encryptText, String encryptKey)
throws SignatureException {
        byte[] rawHmac = null;
        try {
            byte[] data = encryptKey.getBytes(StandardCharsets.UTF_8);
            SecretKeySpec secretKey = new SecretKeySpec(data, "HmacSHA1");
            Mac mac = Mac.getInstance("HmacSHA1");
            mac.init(secretKey);
            byte[] text = encryptText.getBytes(StandardCharsets.UTF_8);
            rawHmac = mac.doFinal(text);
        } catch (InvalidKeyException e) {
            throw new SignatureException("InvalidKeyException:" + e.getMessage());
        } catch (NoSuchAlgorithmException e) {
            throw new SignatureException("NoSuchAlgorithmException:" +
e.getMessage());
        }
        String oauth = new String(Base64.encodeBase64(rawHmac));

        return oauth;
    }

    public final static String MD5(String pstr) {
        char[] md5String = {'0', '1', '2', '3', '4', '5', '6', '7', '8', '9', 'a',
'b', 'c', 'd', 'e', 'f'};
        try {
            byte[] btInput = pstr.getBytes();
            MessageDigest mdInst = MessageDigest.getInstance("MD5");
            mdInst.update(btInput);
            byte[] md = mdInst.digest();
            int j = md.length;
            char[] str = new char[j * 2];
            int k = 0;
            for (int i = 0; i < j; i++) { // i = 0
                byte byte0 = md[i]; // 95
                str[k++] = md5String[byte0 >>> 4 & 0xf]; // 5
                str[k++] = md5String[byte0 & 0xf]; // F
            }

            return new String(str);
        } catch (Exception e) {
            return null;
        }
    }
}
```

- 加密规则说明

1.获取baseString，baseString由app_id和当前时间戳ts拼接而成，假如app_id为595f23df，ts为1512041814，则baseString为

> 595f23df1512041814

2.对baseString进行MD5，假如baseString为上一步生成的595f23df1512041814，MD5之后则为

> 0829d4012497c14a30e7e72aeebe565e

3.以app_secret为key对MD5之后的baseString进行HmacSHA1加密，然后再对加密后的字符串进行base64编码。
假如app_secret为d9f4aa7ea6d94faca62cd88a28fd5234，MD5之后的baseString为上一步生成的0829d4012497c14a30e7e72aeebe565e，
则加密之后再进行base64编码得到的signa为

> IrrzsJeOFk1NGfJHW6SkHUoN9CU=

备注：

- app_secret：接口密钥，在应用中添加实时语音转写服务时自动生成，调用方注意保管；
- signa的生成公式：HmacSHA1(MD5(app_id + ts), app_secret)，具体的生成方法参考本git实例代
- 请求示例

```
{'ts': '1688972361', 'appid': 'test1', 'signa': '6b6VF7IR38Yk3DB651juuKM1Stg=',
'audio_url': 'https://zos.abcpen.com/tts/zmeet/20221023/3058bca8-52cb-11ed-961e-
00155dc6cbed.mp3', 'audio_encode': 'mpeg2', 'audio_sample_rate': '48000',
'has_participle': 'false'}
```

3、错误码

错误码	描述	说明	处理方式
0	success	成功	
-1	in progress	识别中	请继续重试
-2	audio encode error	音频编码错误	请编码成正确的格式，再提交请求
10105	illegal access	没有权限	检查apiKey，ip，ts等授权参数是否正确
10106	invalid parameter	无效参数	上传必要的参数， 检查参数格式以及编码
10107	illegal parameter	非法参数值	检查参数值是否超过范围或不符合要求

错误码	描述	说明	处理方式
10109	audio url is not valid http(s) url	audio_url不是http[s]链接	长语音识别的时候，audio_url必须是http[s]链接
10110	no license	无授权许可	检查参数值是否超过范围或不符合要求
10700	engine error	引擎错误	提供接口返回值，向服务提供商反馈
10701	Audio encode error, only support pcm, aac, mpeg2, opus and flac	音频编码错误	支持pcm, aac, mpeg2, opus 和 flac这几种编码，请选择其中一种
10702	Audio sample error, only support 8000、16000、44100 and 48000 Hz	音频采样率错误	支持 8000、16000、44100 和 48000 Hz，请选择其中一种
10202	websocket connect error	websocket连接错误	检查网络是否正常
10204	websocket write error	服务端websocket写错误	检查网络是否正常，向服务提供商反馈
10205	websocket read error	服务端websocket读错误	检查网络是否正常，向服务提供商反馈
16003	basic component error	基础组件异常	重试或向服务提供商反馈
10800	over max connect limit	超过授权的连接数	确认连接数是否超过授权的连接数