ALIBABA CLOUD

Alibaba Cloud

智能语音交互
自学习平台

文档版本：20201013

阿里云

法律声明

阿里云提醒您在阅读或使用本文档之前仔细阅读、充分理解本法律声明各条款的内容。如果您阅读或使用本文档，您的阅读或使用行为将被视为对本声明全部内容的认可。

1. 您应当通过阿里云网站或阿里云提供的其他授权通道下载、获取本文档，且仅能用于自身的合法合规的业务活动。本文档的内容视为阿里云的保密信息，您应当严格遵守保密义务；未经阿里云事先书面同意，您不得向任何第三方披露本手册内容或提供给任何第三方使用。

2. 未经阿里云事先书面许可，任何单位、公司或个人不得擅自摘抄、翻译、复制本文档内容的部分或全部，不得以任何方式或途径进行传播和宣传。

3. 由于产品版本升级、调整或其他原因，本文档内容有可能变更。阿里云保留在没有任何通知或者提示下对本文档的内容进行修改的权利，并在阿里云授权通道中不时发布更新后的用户文档。您应当实时关注用户文档的版本变更并通过阿里云授权渠道下载、获取最新版的用户文档。

4. 本文档仅作为用户使用阿里云产品及服务的参考性指引，阿里云以产品及服务的“现状”、“有缺陷”和“当前功能”的状态提供本文档。阿里云在现有技术的基础上尽最大努力提供相应的介绍及操作指引，但阿里云在此明确声明对本文档内容的准确性、完整性、适用性、可靠性等不作任何明示或暗示的保证。任何单位、公司或个人因为下载、使用或信赖本文档而发生任何差错或经济损失的，阿里云不承担任何法律责任。在任何情况下，阿里云均不对任何间接性、后果性、惩戒性、偶然性、特殊性或刑罚性的损害，包括用户使用或信赖本文档而遭受的利润损失，承担责任（即使阿里云已被告知该等损失的可能性）。

5. 阿里云网站上所有内容，包括但不限于著作、产品、图片、档案、资讯、资料、网站架构、网站画面的安排、网页设计，均由阿里云和/或其关联公司依法拥有其知识产权，包括但不限于商标权、专利权、著作权、商业秘密等。非经阿里云和/或其关联公司书面同意，任何人不得擅自使用、修改、复制、公开传播、改变、散布、发行或公开发表阿里云网站、产品程序或内容。此外，未经阿里云事先书面同意，任何人不得为了任何营销、广告、促销或其他目的使用、公布或复制阿里云的名称（包括但不限于单独为或以组合形式包含“阿里云”、“Aliyun”、“万网”等阿里云和/或其关联公司品牌，上述品牌的附属标志及图案或任何类似公司名称、商号、商标、产品或服务名称、域名、图案标示、标志、标识或通过特定描述使第三方能够识别阿里云和/或其关联公司）。

6. 如若发现本文档存在任何错误，请与阿里云取得直接联系。

通用约定

格式	说明	样例
危险	该类警示信息将导致系统重大变更甚至故障，或者导致人身伤害等结果。	危险 重置操作将丢失用户配置数据。
警告	该类警示信息可能会导致系统重大变更甚至故障，或者导致人身伤害等结果。	警告 重启操作将导致业务中断，恢复业务时间约十分钟。
注意	用于警示信息、补充说明等，是用户必须了解的内容。	注意 权重设置为0，该服务器不会再接受新请求。
说明	用于补充说明、最佳实践、窍门等，不是用户必须了解的内容。	说明 您也可以通过按Ctrl+A选中全部文件。
>	多级菜单递进。	单击**设置**> **网络**> **设置网络类型**。
粗体	表示按键、菜单、页面名称等UI元素。	在**结果确认**页面，单击**确定**。
`Courier字体`	命令或代码。	执行 `cd /d C:/window` 命令，进入Windows系统文件夹。
斜体	表示参数、变量。	`bae log list --instanceid` *Instance_ID*
[] 或者 [a|b]	表示可选项，至多选择一个。	`ipconfig [-all|-t]`
{} 或者 {a|b}	表示必选项，至多选择一个。	`switch {active|stand}`

目录

1.热词

1.1. 概述

在语音识别服务中，如果您的业务领域有部分词汇默认识别效果不够好，可以考虑使用热词功能，将这些词添加到词表从而改善识别结果。

热词分类

- 名称（人名/地名）

  目前名称类热词只支持人名和地名。一个词表中只能包含人名或只能包含地名。

- 业务专属

  可以是姓名、公司名称，或某领域专有名词如人名“王晓铭”（通常会被识别为“王小明”）、司法领域的“被上诉人”、电商领域的“包邮”。

> 说明
>
> - 热词不支持标点符号，不建议过长。
> - 两类热词可各添加10组，每组128个。
> - 每个项目可关联1个人名、1个地名和1个业务专属类热词。
> - 如果您有大量热词的添加需求，请使用自学习模型。详情请参见自学习模型。

管控台与POP API设置热词的区别

- 在管控台上配置项目热词与项目Appkey绑定，无需在代码中指定热词表；POP API仅支持创建业务专属热词表，需要您在客户端代码中调用SDK接口设置业务专属热词表ID，该热词表才能生效。
- 在管控台上配置项目热词时，每个热词占一行，无需设置热词权重；使用POP API创建业务专属热词表，需要指定每个热词的权重。

应用举例

为了提高电影名称识别率，将如下电影名称作为热词添加到项目中。

肖申克的救赎
霸王别姬
这个杀手不太冷
阿甘正传
美丽人生
泰坦尼克号
千与千寻
辛德勒的名单
盗梦空间
机器人总动员
忠犬八公的故事
三傻大闹宝莱坞
海上钢琴师
放牛班的春天
楚门的世界
教父
龙猫
星际穿越

1.2. 创建热词

本文为您介绍如何在控制台创建热词。

前提条件

已开通智能语音交互服务，详情请参见开通服务。

使用限制

- 目前仅支持中文热词识别。
- 文件为TXT格式，100 KB以内，`UTF-8（无BOM）`编码。
- 每行一个热词，最多128行，每个热词不超过10个字。
- 词语中的数字需要按照发音替换为对应的汉字。例如：“58.9元”需要替换为“五十八点九元”。
- 语料中请不要出现除空格、制表符、换行、换页之外的其他特殊字符。

操作步骤

1. 登录智能语音交互管理控制台。
2. 在左侧导航栏单击**自学习平台 > 热词**。
3. 在**热词**页面，单击**创建热词**。
4. 在**添加热词组**弹框中，输入热词组名称、上传热词文件，完成后单击**确定**。

   您也可以通过以下方式添加热词组：

i. 在左侧导航栏单击**全部项目**。

ii. 在**我的所有项目**页面，单击项目列**操作**栏下的**项目功能配置**。

iii. 在**热词**区域，单击右侧的**创建热词**。

5. 在**项目功能配置**页面，选择需要关联的热词，完成后单击**发布上线**。

1.3. 使用POP API创建业务专属热词

本文为您介绍如何使用POP API，在客户端自行操作训练业务专属热词，而无需依赖控制台的设置。

您可通过POP API执行如下操作：

- 创建词表：`CreateAsrVocab`
- 获取词表：`GetAsrVocab`
- 更新词表：`UpdateAsrVocab`
- 删除词表：`DeleteAsrVocab`
- 列举所有词表：`ListAsrVocab`

调用限制

- 默认最多创建10个词表。
- 每个词表最多添加128个词，每个词语最长10个字。
- 业务专属热词必须为`UTF-8`编码，不能包含标点、特殊字符。
- 业务专属词对应的权重取值范围为[-6,5]之间的整数。

> **说明**
>
> 取值大于0增大该词语被识别的概率，小于0减小该词语被识别的概率。
>
> 取值为-6：表示尽量不要识别出该词语。
>
> 取值为2：常用值。
>
> 如果效果不明显可以适当增加权重，但是当权重较大时可能会起到负面效果，导致其他词语识别不准确。

创建词表

一个词表就是一类业务专属热词的组合。

输入参数：

提交创建词表的请求时，需要设置输入参数到请求的Body中，如下表所示。

名称	类型	是否必选	说明
Name	String	是	词表名称。

名称	类型	是否必选	说明

WordWeights	String	是	词表里的词和对应的权重，为JSON的Map格式字符串。例如：
			{
			"苹果":3,
			"西瓜":3
			}
			key为String类型的热词，value为Int类型的热词对应的权重。

Description	String	否	词表描述信息。

输出参数：

返回的输出参数为JSON格式的字符串。

```
{
 "VocabId": "0074ac87db754e0fbd3465c60d86****",
 "RequestId": "77C00AE4-A646-4A41-B6FF-F06C19FA****"
}
```

- 返回HTTP状态：200表示成功，更多状态码请查阅HTTP状态码。
- 返回JSON格式字符串参数：

名称	类型	说明
RequestId	String	请求ID。
VocabId	String	词表ID，作为后续词表的获取、更新、删除使用。

获取词表

根据词表的ID获取对应的词表信息。

输入参数：

提交获取词表的请求时，需要设置输入参数到请求的Body中，如下表所示。

名称	类型	是否必选	说明
Id	String	是	词表ID（创建词表时获取的VocabId）。

输出参数：

返回的输出参数为JSON格式的字符串。

```
{
 "RequestId": "A590423E-FEBC-4AA0-A520-4DA77292****",
 "Vocab": {
  "Name": "测试词表",
  "Md5": "58c732d3b31eb564c275371d46fc****",
  "Description": "测试词表描述",
  "CreateTime": "2018-11-26 17:19:40",
  "UpdateTime": "2018-11-26 17:19:40",
  "Id": "6118b2a057d1440bb253382a7617****",
  "WordWeights": {
   "西瓜": 3,
   "苹果": 3
  },
  "Size": 323
 }
}
```

- 返回HTTP状态：200表示成功，更多状态码请查阅HTTP状态码。
- 返回JSON格式字符串参数：

名称	类型	说明
RequestId	String	请求ID。
Vocab	Vocab对象	词表对象。

其中，Vocab对象的参数描述如下表所示。

名称	类型	说明
Id	String	词表ID（与创建词表时获取的VocabId相同）。
Name	String	词表名称。
Description	String	词表描述信息。
Size	Int	词表编译后的大小。
Md5	String	词表编译后的md5值。
CreateTime	String	词表创建时间。
UpdateTime	String	词表更新时间。
WordWeights	Map	词表中的业务专属热词和对应的权重。

更新词表

根据词表的ID可以更新对应的词表信息，包括词表名称、词表描述信息、词表的词和权重。

输入参数：

提交更新词表的请求时，需要设置输入参数到请求的Body中，如下表所示。

名称	类型	是否必选	说明
Id	String	是	词表Id（创建词表时获取的VocabId）。
Name	String	是	更新后的词表名称。
WordWeights	String	是	更新后的词表里业务专属热词和对应的权重，为JSON的Map格式字符串。
Description	String	否	更新后的词表描述信息。

输出参数：

返回的输出参数为JSON格式的字符串。

```
{
 "RequestId": "829E373C-9E23-4DEF-A979-002F140B****"
}
```

- 返回HTTP状态：200表示成功，更多状态码请查阅HTTP状态码。
- 返回JSON格式字符串参数：

名称	类型	说明
RequestId	String	请求ID。

删除词表

根据词表的ID删除对应的词表。

输入参数：

提交删除词表请求时，需要设置输入参数到请求的Body中，如下表所示。

名称	类型	是否必选	说明
Id	String	是	词表Id（创建词表时获取的VocabId）。

输出参数：

返回的输出参数为JSON格式的字符串。

```
{
 "RequestId": "75CCBD40-BC19-4227-9140-0F42806B****"
}
```

- 返回HTTP状态：200表示成功，更多状态码请查阅HTTP状态码。
- 返回JSON格式字符串参数：

名称	类型	说明
RequestId	String	请求ID。

列举词表

列举指定页的词表信息。

> 说明
>
> 为了防止响应Body过大，获取的词表信息不包含具体的专属热词和对应的权重信息。

输入参数：

提交列举词表请求时，需要设置输入参数到请求的Body中，如下表所示。

名称	类型	是否必选	说明
PageNumber	Int	否	页号，默认值为1，取值大于0。
PageSize	Int	否	每页包含的词表数量，默认值为10，取值范围为[10,100]。

> 说明
>
> 服务端根据词表的修改时间降序排列，如果词表过多，一次获取全部的词表，导致HTTP响应体过大，可能会被中间的网关或者代理节点拦截。因此采用分页的方式，每次请求获取指定的一页词表。PageSize指定了每页的词表数量，不足一页按一页处理，PageNumber指定了要获取第几页的词表。

输出参数：

返回的输出参数为JSON格式的字符串。

```
{
 "Page": {
  "PageNumber": 1,
  "PageSize": 10,
  "TotalItems": 5,
  "TotalPages": 1,
  "Content": [{
   "Name": "测试词表_1",
   "Md5": "eafaaf1d73b17c9d35c64d600e07****",
   "Description": "测试词表描述_1",
   "CreateTime": "2018-11-26 17:51:41",
   "UpdateTime": "2018-11-26 17:51:41",
   "Id": "266df2819a9d4d96a07c5c5d39b6****",
   "Size": 323
  }, {
   "Name": "测试词表_2",
   "Md5": "f32c10fd8569cb3712576a0ea988****",
   "Description": "测试词表描述_2",
   "CreateTime": "2018-11-26 17:51:41",
   "UpdateTime": "2018-11-26 17:51:41",
   "Id": "0fa718759c034f67bb3e394d2fd9****",
   "Size": 323
  }]
 },
 "RequestId": "CB7B4AB4-5C16-4617-8B91-519A130E****"
}
```


- 返回HTTP状态：200表示成功，更多状态码请查阅HTTP状态码。
- 返回JSON格式字符串参数：

名称	类型	说明
RequestId	String	请求ID。
Page	Page对象	输入参数中指定获取页，没有指定默认获取第一页。

其中Page对象的参数描述如下表所示。

名称	类型	说明
Content	List< Vocab >	词表数组，参见获取词表中输出参数Vocab，不包含WordWeights参数。
PageNumber	Int	页号，与输入参数相同。
PageSize	Int	页包含的词表数量，与输入参数相同。
TotalPages	Int	总页数。
TotalItems	Int	总词表数量。

错误码

在提交创建词表、获取词表、更新词表、删除词表、列举词表的请求时，获取的HTTP响应状态码如果不是 `200` ，则表示请求失败。失败信息以JSON格式字符串包含在HTTP响应的Body中，具体的错误码如下表所示。

错误码	说明	解决方案
SLP.PAGE_NUMBER_INVALID	调用列举接口时设置了无效的页号。	请检查列举的页号是否在有效取值范围：[1, ∞)。
SLP.PAGE_SIZE_INVALID	调用列举接口时设置了无效的页大小。	请检查列举的页大小是否在有效取值范围：[10, 100]。
SLP.NOT_FOUND	词表ID不存在。	请检查使用的词表ID是否正确。
SLP.PARAMETER_ERROR	请求参数设置错误。	请参考返回的具体错误消息提示，检查参数设置。
SLP.EXCEED_LIMIT	创建词表数量超过上限。	请确认现有的词表数量限制，默认每个用户允许创建10个业务专属热词表。

错误码	说明	解决方案
SLP.ASR_VOCAB_ERROR	其他错误。	与词表相关的其他错误，请参考错误消息提示。

以创建词表超出数量限制的错误响应为例：

```
{
 "RequestId": "848C33E3-5A74-4BF8-9BE6-B78576C6****",
 "HostId": "nls-slp.cn-shanghai.aliyuncs.com",
 "Code": "SLP.EXCEED_LIMIT",
 "Message": "Vocab count has reached the limit! (max: 10)"
}
```

示例代码

说明

- 示例使用了阿里云Java SDK的CommonRequest提交请求，采用的是RPC风格的POP API调用方式。
- 阿里云SDK的详细介绍请参见安装及使用Java SDK。
- Java SDK CommonRequest的使用方法请参见使用CommonRequest进行调用。
- 您只需要依赖阿里云Java SDK核心库与阿里开源库fastjson。阿里云Java SDK的核心库版本支持3.5.0及以上（如果版本在4.0.0及以上，需要增加其对应的第三方依赖，根据错误提示补充即可）。

```
<dependency>
    <groupId>com.aliyun</groupId>
    <artifactId>aliyun-java-sdk-core</artifactId>
    <version>3.7.1</version>
</dependency>
<dependency>
    <groupId>com.alibaba</groupId>
    <artifactId>fastjson</artifactId>
    <version>1.2.49</version>
</dependency>
```

```
import com.alibaba.fastjson.JSONObject;
import com.aliyuncs.CommonRequest;
import com.aliyuncs.CommonResponse;
```

```
import com.aliyuncs.DefaultAcsClient;
import com.aliyuncs.IAcsClient;
import com.aliyuncs.exceptions.ClientException;
import com.aliyuncs.http.MethodType;
import com.aliyuncs.http.ProtocolType;
import com.aliyuncs.profile.DefaultProfile;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
public class AsrVocabPopApiDemo {
    /**
     * 地域信息
     * 常量内容，固定值。
     */
    private static final String REGION_ID = "cn-shanghai";
    private static final String DOMAIN = "nls-slp.cn-shanghai.aliyuncs.com";
    private static final ProtocolType PROTOCOL_TYPE = ProtocolType.HTTPS;
    /**
     * POP API信息
     * 常量内容，固定值。
     */
    private static final String API_VERSION = "2018-11-20";
    private static final String ACTION_CREATE_ASR = "CreateAsrVocab";
    private static final String ACTION_GET_ASR_VOCAB = "GetAsrVocab";
    private static final String ACTION_LIST_ASR_VOCAB = "ListAsrVocab";
    private static final String ACTION_UPDATE_ASR_VOCAB = "UpdateAsrVocab";
    private static final String ACTION_DELETE_ASR_VOCAB = "DeleteAsrVocab";
    /**
     * 参数设置key
     * 常量内容，固定值。
     */
    private static final String KEY_VOCAB_ID = "VocabId";
    private static final String KEY_ID = "Id";
    private static final String KEY_NAME = "Name";
    private static final String KEY_DESCRIPTION = "Description";
    private static final String KEY_WORD_WEIGHTS = "WordWeights";
    private static final String KEY_VOCAB = "Vocab";
    private static final String KEY_PAGE = "Page";
    private static final String KEY_PAGE_NUMBER = "PageNumber";
    private static final String KEY_PAGE_SIZE = "PageSize";
```

```
// 阿里云鉴权client
private static IAcsClient client;
static class Vocab {
    public String Id;
    public String Name;
    public String Description;
    public int Size;
    public String Md5;
    public String CreateTime;
    public String UpdateTime;
    public Map<String, Integer> WordWeights = new HashMap<String, Integer>();
}
static class Page {
    class VocabContent {
        public String Id;
        public String Name;
        public String Description;
        public int Size;
        public String Md5;
        public String CreateTime;
        public String UpdateTime;
    }
    public int PageNumber;
    public int PageSize;
    public int TotalItems;
    public int TotalPages;
    public List<VocabContent> Content = new ArrayList<VocabContent>();
}
public AsrVocabPopApiDemo(String akId, String akSecret) {
    DefaultProfile profile = DefaultProfile.getProfile(REGION_ID, akId, akSecret);
    client = new DefaultAcsClient(profile);
}
private CommonRequest newRequest(String action) {
    CommonRequest request = new CommonRequest();
    request.setDomain(DOMAIN);
    request.setProtocol(PROTOCOL_TYPE);
    request.setVersion(API_VERSION);
    request.setMethod(MethodType.POST);
    request.setAction(action);
    return request;
```

```
}
/**
 * 创建词表
 *
 * @param name 词表名称，必填。
 * @param description 词表描述信息，可选。
 * @param wordWeights 词表里的词和对应的权重，JSON的Map格式，必填。
 *
 * @return String 创建的词表Id。
 */
String createAsrVocab(String name, String description, String wordWeights) {
    CommonRequest request = newRequest(ACTION_CREATE_ASR);
    request.putBodyParameter(KEY_NAME, name);
    request.putBodyParameter(KEY_DESCRIPTION, description);
    request.putBodyParameter(KEY_WORD_WEIGHTS, wordWeights);
    CommonResponse response = null;
    try {
        response = client.getCommonResponse(request);
    } catch (ClientException e) {
        e.printStackTrace();
    }
    if (response.getHttpStatus() != 200) {
        System.out.println(response.getData());
        System.out.println("创建词表失败，HTTP错误码：" + response.getHttpStatus());
        return null;
    }
    JSONObject result = JSONObject.parseObject(response.getData());
    String vocabId = result.getString(KEY_VOCAB_ID);
    return vocabId;
}
/**
 * 获取词表
 *
 * @param vocabId 词表Id。
 *
 * @return Vocab 获取的词表对象。
 */
Vocab getAsrVocab(String vocabId) {
    CommonRequest request = newRequest(ACTION_GET_ASR_VOCAB);
    request.putBodyParameter(KEY_ID, vocabId);
    CommonResponse response = null;
```

```
    try {
        response = client.getCommonResponse(request);
    } catch (ClientException e) {
        e.printStackTrace();
    }
    if (response.getHttpStatus() != 200) {
        System.out.println(response.getData());
        System.out.println("获取词表失败，HTTP错误码：" + response.getHttpStatus());
        return null;
    }
    JSONObject result = JSONObject.parseObject(response.getData());
    String vocabJson = result.getString(KEY_VOCAB);
    Vocab vocab = JSONObject.parseObject(vocabJson, Vocab.class);
    return vocab;
}
/**
 * 更新词表
 *
 * @param vocabId 待更新的词表Id。
 * @param name 更新后的词表名称。
 * @param description 更新后的词表描述。
 * @param wordWeights 更新后的热词和权重。
 *
 * @return boolean 更新词表是否成功。
 */
boolean updateAsrVocab(String vocabId, String name, String description, String wordWeights) {
    CommonRequest request = newRequest(ACTION_UPDATE_ASR_VOCAB);
    request.putBodyParameter(KEY_ID, vocabId);
    request.putBodyParameter(KEY_NAME, name);
    request.putBodyParameter(KEY_DESCRIPTION, description);
    request.putBodyParameter(KEY_WORD_WEIGHTS, wordWeights);
    CommonResponse response = null;
    try {
        response = client.getCommonResponse(request);
    } catch (ClientException e) {
        e.printStackTrace();
    }
    if (response.getHttpStatus() != 200) {
        System.out.println(response.getData());
        System.out.println("更新词表失败，HTTP错误码：" + response.getHttpStatus());
        return false;
```

```
        return false;
    }
    return true;
}
/**
 * 删除词表
 *
 * @param vocabId 词表Id。
 *
 * @return boolean 删除词表是否成功。
 * */
boolean deleteAsrVocab(String vocabId) {
    CommonRequest request = newRequest(ACTION_DELETE_ASR_VOCAB);
    request.putBodyParameter(KEY_ID, vocabId);
    CommonResponse response = null;
    try {
        response = client.getCommonResponse(request);
    } catch (ClientException e) {
        e.printStackTrace();
    }
    if (response.getHttpStatus() != 200) {
        System.out.println(response.getData());
        System.out.println("删除词表失败, HTTP错误码：" + response.getHttpStatus());
        return false;
    }
    return true;
}
/**
 * 列举词表
 * 如果不指定获取的页号，默认获取第1页。
 * 如果不指定每页的词表数量，默认每页10个词表。
 *
 * @return Page 所有词表信息。
 */
Page listAsrVocab() {
    CommonRequest request = newRequest(ACTION_LIST_ASR_VOCAB);
    request.putBodyParameter(KEY_PAGE_NUMBER, 1);
    request.putBodyParameter(KEY_PAGE_SIZE, 10);
    CommonResponse response = null;
    try {
        response = client.getCommonResponse(request);
```

```
        } catch (ClientException e) {
            e.printStackTrace();
        }
        if (response.getHttpStatus() != 200) {
            System.out.println(response.getData());
            System.out.println("列举词表失败，HTTP错误码：" + response.getHttpStatus());
            return null;
        }
        JSONObject result = JSONObject.parseObject(response.getData());
        String pageJson = result.getString(KEY_PAGE);
        Page page = JSONObject.parseObject(pageJson, Page.class);
        return page;
    }
    public static void main(String[] args) {
        if (args.length < 2) {
            System.err.println("FileASRDemo need params: <AccessKey Id> <AccessKey Secret>");
            return;
        }
        String accessKeyId = args[0];
        String accessKeySecret = args[1];
        AsrVocabPopApiDemo demo = new AsrVocabPopApiDemo(accessKeyId, accessKeySecret);
        // 词表ID
        String vocabId = null;
        /**
         * 创建词表
         */
        String name = "测试词表";
        String description = "测试词表描述";
        String wordWeights = "{\"苹果\": 3, \"西瓜\": 3}";
        vocabId = demo.createAsrVocab(name, description, wordWeights);
        if (vocabId != null) {
            System.out.println("创建词表成功，词表Id：" + vocabId);
        }
        else {
            System.out.println("创建词表失败！");
            return;
        }
        /**
         * 获取词表
         */
        Vocab vocab = demo.getAsrVocab(vocabId);
```

```
if (vocab != null) {
    System.out.println("获取词表成功：" + JSONObject.toJSONString(vocab));
}
else {
    System.out.println("获取词表失败！");
}
/**
 * 更新词表
 */
name = "测试词表2";
description = "测试词表描述2";
wordWeights = "{\"苹果\": 2, \"西瓜\": 2}";
boolean isUpdated = demo.updateAsrVocab(vocabId, name, description, wordWeights);
if (isUpdated) {
    System.out.println("更新词表成功：" + JSONObject.toJSONString(demo.getAsrVocab(vocabId)));
}
else {
    System.out.println("更新词表失败！");
}
/**
 * 删除词表
 */
boolean isDeleted = demo.deleteAsrVocab(vocabId);
if (isDeleted) {
    System.out.println("删除词表成功！");
}
else {
    System.out.println("删除词表失败！");
}
/**
 * 列举所有词表
 */
// 创建多个词表
for (int i = 0; i < 10; i++) {
    name = "测试词表_" + String.valueOf(i);
    description = "测试词表描述_" + String.valueOf(i);
    JSONObject jsonObject = new JSONObject();
    jsonObject.put("苹果", 2);
    jsonObject.put("西瓜", 2);
    wordWeights = jsonObject.toJSONString();
    demo.createAsrVocab(name, description, wordWeights);
```

```
            demo.createAsrVocab(name, description, wordWeights);
        }
        // 列举创建的词表
        Page page = demo.listAsrVocab();
        if (page != null) {
            System.out.println("列举词表成功：" + JSONObject.toJSONString(page));
        }
        else {
            System.out.println("列举词表失败！");
            return;
        }
        // 删除所有的词表
        for (int i = 0; i < page.Content.size(); i++) {
            demo.deleteAsrVocab(page.Content.get(i).Id);
        }
        page = demo.listAsrVocab();
        if (page != null) {
            System.out.println("删除所有词表：" + JSONObject.toJSONString(page));
        }
    }
}
```

1.4. 使用SDK设置业务专属热词

本文为您介绍在一句话识别、实时语音识别和录音文件识别SDK示例中如何设置业务专属热词。

通过管控台配置的业务专属热词表与项目appkey绑定，无需自行设置；通过POP API训练获取的业务专属热词表，需要在SDK中设置其词表ID。

一句话识别

在一句话识别中，需要通过设置高级参数 `vocabulary_id` 指定业务专属热词表ID。

Java SDK

> **说明**
>
> 首先需要了解Java SDK的基本使用方法，详情请参见Java SDK。

SDK中没有设置 `vocabulary_id` 参数的方法，需要通过SpeechRecognizer类中的addCustomedParam方法进行设置：

```
public void addCustomedParam(String key, Object value);
```

在示例中创建SpeechRecognizer对象recognizer后（调用start方法之前），调用该方法设置业务专属热词：

```
SpeechRecognizer recognizer = new SpeechRecognizer(client, getRecognizerListener());
... // 省略其他设置
recognizer.addCustomedParam("vocabulary_id", "您的业务专属热词表ID");
...
```

iOS SDK

> 说明
> 首先需要了解iOS SDK的基本使用方法，详情请参见iOS SDK。

通过RequestParam对象的setParams方法设置热词ID。

```
...
NSMutableDictionary *userParams = [[NSMutableDictionary alloc]init];
[userParams setValue:@"您的业务专属热词表ID" forKey:@"vocabulary_id"];
[_recognizeRequestParam setParams:userParams];
...
```

Android SDK

> 说明
> 首先需要了解Android SDK的基本使用方法，详情请参见Android SDK。

通过SpeechRecognizer对象的setParams方法设置热词ID。

```
...
String userParamString = "{\"vocabulary_id\":\"您的泛热词表ID\"}";
speechRecognizer.setParams(userParamString);
```

RESTful API

> 说明
> 首先需要了解RESTful API的基本使用方法，详情请参见RESTFUL API。

`vocabulary_id` 参数作为HTTP的请求参数追加到其他请求参数之后，以Java示例举例如下，其他语言设置方法与Java类似。

```
String url = "http://nls-gateway.cn-shanghai.aliyuncs.com/stream/v1/asr";
String request = url;
request = request + "?appkey=" + appkey;
request = request + "&vocabulary_id=" + "您的业务专属热词表ID";
...
```

实时语音识别

在实时语音识别中，需要通过设置高级参数 `vocabulary_id` 指定业务专属热词表ID。

Java SDK

> 说明
>
> 首先需要了解Java SDK的基本使用方法，详情请参见java SDK。

SDK中没有设置 `vocabulary_id` 参数的方法，需要通过SpeechTranscriber类中的addCustomedParam方法进行设置：

```
public void addCustomedParam(String key, Object value);
```

在示例中创建SpeechTranscriber的对象transcriber后（调用start方法之前），调用该方法设置业务专属热词：

```
SpeechTranscriber transcriber = new SpeechTranscriber(client, getTranscriberListener());
... // 省略其他设置
transcriber.addCustomedParam("vocabulary_id", "您的业务专属热词表ID");
...
```

C++ SDK

> 说明
>
> 首先需要了解C++ SDK的基本使用方法，详情请参见C++ SDK。

SDK中没有设置 `vocabulary_id` 参数的方法，需要通过SpeechTranscriberRequest类中的setPayloadParam方法进行设置：

```
/**
 * @brief 参数设置
 * @param value： JSON Map格式的字符串。
 * @return 成功则返回0，否则返回-1。
 */
int setPayloadParam(const char value);
```

在示例中创建SpeechTranscriberRequest的对象request后（调用start方法之前），调用该方法设置业务专属热词：

```
SpeechTranscriberRequest* request = NlsClient::getInstance()->createTranscriberRequest(callback);
... // 省略其他设置
request->setPayloadParam("{\"vocabulary_id\":\"您的业务专属热词表ID\"}");
...
```

iOS SDK

> 说明
> 首先需要了解iOS SDK的基本使用方法，详情请参见iOS SDK。

通过RequestParam对象的setParams方法设置热词ID。

```
...
NSMutableDictionary *userParams = [[NSMutableDictionary alloc]init];
[userParams setValue:@"您的业务专属热词表ID" forKey:@"vocabulary_id"];
[_transRequestParam setParams:userParams];
...
```

Android SDK

> 说明
> 首先需要了解Android SDK的基本使用方法，详情请参见Android SDK。

通过SpeechTranscriber对象的setParams方法设置热词ID。

```
...
String userParamString = "{\"vocabulary_id\":\"您的业务专属热词表ID\"}";
speechTranscriber.setParams(userParamString);
```

录音文件识别

在录音文件识别中，需要通过设置高级参数 `vocabulary_id` 指定业务专属热词表ID。

> 说明
> 首先需要了解录音文件识别接口基本使用方法，详情请参见接口说明。

`vocabulary_id` 参数和其他输入参数一样，以JSON字符串形式设置到HTTP请求Body中，JSON格式如下。

```
{
 "app_key": "您的项目appkey",
 "vocabulary_id": "您的业务专属热词表ID",
 "file_link": "https://aliyun-nls.oss-cn-hangzhou.aliyuncs.com/asr/fileASR/examples/nls-sample-16k.wav"
}
```

以Java SDK为例，其他语言SDK的参数设置方法与Java类似。

```
CommonRequest postRequest = new CommonRequest();
 ... // 省略其他设置
/**
 * 设置请求参数，以JSON字符串形式设置到请求Body中。
 */
JSONObject taskObject = new JSONObject();
// 设置appkey
taskObject.put(KEY_APP_KEY, appKey);
// 设置音频文件访问链接
taskObject.put(KEY_FILE_LINK, "https://aliyun-nls.oss-cn-hangzhou.aliyuncs.com/asr/fileASR/examples/nls-sample-16k.wav");
taskObject.put("vocabulary_id", "您的业务专属热词表ID");
String task = taskObject.toJSONString();
System.out.println(task);
// 设置以上JOSN字符串为Body参数
postRequest.putBodyParameter(KEY_TASK, task);
...
```

2.语言模型定制

2.1. 概述

阿里云官方对某些场景（包括通用、教育、司法、医疗等）进行了大量语音识别训练，提供了高准确率场景模型。如果您需要的语音识别服务场景不在所提供的模型范围内，或者需要对标准模型进行更进一步优化，可以通过自学习平台，达成优化目的。

通过使用阿里云语音自学习工具，您可以在操作界面上传训练语料文本，并选择对应领域的语言基础模型，对训练语料做模型训练，从而有效提高该场景的语音识别率。尤其针对专有名词和高频词汇，有较好的优化效果。

控制台与POP API设置自学习模型的区别

使用控制台训练和管理自学习模型，可以界面化操作，在控制台**项目功能配置**中，单击**切换场景**，选择自学习模型，发布上线后将与appkey绑定，而无需在代码中自行设置。

使用POP API创建的自学习模型，需要您在客户端代码中调用SDK的接口设置自学习模型的ID，该模型才能生效。

训练语料说明

调用限制

- 训练数据为领域相关的文本，与待识别语音数据越接近，优化效果越好。
- 以文本方式保存，使用 `UTF-8（无BOM）` 格式编码，文件大小不超过10 MB。
- 一句话或者一个被加强调优的关键词单独一行，控制每行的长度在500个字符以内。
- 文本中的数字需要按照发音替换为对应的汉字。例如，“58.9元”需要转换为“五十八点九元”。
- 文件中需要至少有一行为句子（大于4个词）。
- 只采用逗号（，）、句号（。）、问号（？）和感叹号（！），句尾需要加标点。像书名号（《》）、双引号（“”）等标点应去除。

优化建议

对于识别不准确的关键词，可以将含该词的句子或者关键词（一个关键词在训练文本中独占一行）多复制几行，例如10行。如果效果仍不满意，可以适当增加复制行数。

> **说明**
>
> - 需要首先排除关键词识别不准确，不是发音不清晰或者音频质量不好造成的。
> - 建议经过识别试错，谨慎提供训练语料，避免相同发音的其他内容识别错误。

应用举例

下载训练语料，以阿里巴巴简介为例：

一九九九年九月，马云带领下的十八位位创始人在杭州的公寓中正式成立了阿里巴巴集团，集团的首个网站是英文全球批发贸易市场阿里巴巴。

同年阿里巴巴集团推出专注于国内批发贸易的中国交易市场。

一九九九年十月，阿里巴巴集团从数家投资机构融资五百万美元。

一九九九年十月，阿里巴巴集团从数家投资机构融资五百万美元。

二零零零年一月，阿里巴巴集团从软银等数家投资机构融资两千万美元。

二零零零年一月，阿里巴巴集团从软银等数家投资机构融资两千万美元。

二零零零年九月，阿里巴巴集团举办首届西湖论剑，汇聚互联网界的商业和意见领袖讨论业界重要议题。

如果“融资”、“互联网”等是业务关键词，可以将含这两个词的句子多复制几遍。

训练流程如下：

1. 选择基础模型：采用通用模型（具体选择何种模型可根据实际场景进行调整）。
2. 训练语料采集：请将如上训练语料保存至训练文本。如果需要自行设置训练语料，请根据标点做裁剪，将每句话保存为训练文本中的一行。
3. 操作训练模型：通过自学习服务提交语料并训练之后，采用训练出的模型，能够有效识别出训练语料中的词汇，获得理想的识别效果。

2.2. 定制语言模型

本文为您介绍如何在控制台创建定制模型并应用模型。

前提条件

已开通智能语音交互服务，详情请参见开通服务。

> 说明
>
> - 该功能免费开放给所有开通智能语音交互免费版和商用版的用户。
> - 最多支持创建10个模型。

操作步骤

1. 登录智能语音交互管理控制台。
2. 单击左侧导航栏**自学习平台 > 语言模型定制**。在**语言模型定制**页的**模型**页签下，单击**创建模型**。
3. 输入模型名称并上传测试集（可选），完成后单击**下一步**。

   

   您可以直接上传或选择已有测试集，上传测试集有如下优势：

   i. 自动对创建的语言模型进行测试，量化自学习语言模型的定制效果。

   ii. 系统可以根据测试集分析并推荐最优基础模型，确保最佳的语言模型定制效果。
4. 选择基础模型，完成后单击**下一步**。

   
5. 上传语料，完成后单击**确定**。

上传或选择已有训练语料后，系统会自动开始进行模型定制训练，此时**模型状态**显示**训练中**。

6. 创建自动化测试任务。

如果您在步骤3中已上传测试集，当**模型状态**变为**模型上线**时，系统会自动创建测试任务。否则需要您单击模型**操作**栏下的**自动化测试**手动创建测试任务。

7. 应用定制模型。

当**模型状态**为**模型上线**时，您就可以在项目中应用该模型。

i. 在左侧导航栏选择**全部项目**，在**我的所有项目**页面，单击项目列**操作**栏下的**项目功能配置**。

ii. 在**语音识别**下，选择您已定制好的模型场景。

iii. 单击**发布上线**。

2.3. 使用POP API创建自学习模型

本文为您介绍如何使用POP API，在客户端自行操作训练自学习模型，不需要依赖管控台的设置。

您可通过POP API执行如下操作：

- 数据集（训练语料）管理
  - 创建数据集：`CreateAsrLmData`
  - 查询数据集：`GetAsrLmData`
  - 删除数据集：`DeleteAsrLmData`
  - 列举数据集：`ListAsrLmData`
- 自学习模型管理
  - 创建自学习模型：`CreateAsrLmModel`
  - 查询自学习模型：`GetAsrLmModel`
  - 删除自学习模型：`DeleteAsrLmModel`
  - 列举自学习模型：`ListAsrLmModel`
- 自学习模型的训练与发布
  - 添加数据集到自学习模型：`AddDataToAsrLmModel`
  - 从自学习模型中删除数据集：`RemoveDataFromAsrLmModel`
  - 训练自学习模型：`TrainAsrLmModel`
  - 上线自学习模型：`DeployAsrLmModel`
  - 下线自学习模型：`UndeployAsrLmModel`

开通服务

使用自学习服务的POP API接口训练自学习模型，需要首先在管控台开通智能语音交互服务，具体开通步骤，请参见定制语言模型。

> 说明
>
> 在管控台的项目功能配置中，请设置项目当前模型与训练自学习模型使用的基础模型一致。

自学习模型服务信息如下表。

名称	值
地域ID	cn-shanghai
域名	nls-slp.cn-shanghai.aliyuncs.com
协议类型	HTTPS
POP API版本	2018-11-20

准备训练语料

请阅读概述中的训练语料说明，根据调用限制和优化建议准备训练语料。

指标	说明
自学习模型数量	永久免费，最多创建10个。
数据集的数量	默认最多可创建100个。
单个数据集的大小	最大10 MB。
单个自学习模型使用数据集的数量	固定为10个。
生效时间	对于总的数据集在10 MB以下可以在5分钟内生效，对于总的数据集较大的情况在半小时内生效。

训练流程

训练过程步骤如下：

1. 准备训练语料，将训练语料导入自学习服务，并创建为训练数据集。

   数据集操作的状态转换

状态说明：

状态	说明
Fetching	正在将训练数据从URL导入到自学习系统中。
FetchingFailed	复制数据集出现错误，请检查训练数据的URL是否正确（只支持阿里云OSS的HTTP/HTTPS链接）。
Ready	数据集导入成功。

2. 创建一个自学习模型，并将数据集添加到该模型。
3. 启动自学习模型训练，并等待模型训练完成。

   训练自学习模型的状态转换

   状态说明：

状态	说明
Empty	自学习模型新创建，还未训练过。
Training	自学习模型正在训练中。
TrainingFailed	自学习模型训练失败。
Ready	已上线的自学习模型下线成功。
Deploying	自学习模型正在上线或下线。
Deployed	自学习模型已上线。

接口说明

数据集管理：数据集即训练语料，与管控台中上传的训练语料相同。

创建数据集

调用动作：CreateAsrLmData

输入参数：

提交创建数据集的请求时，需要设置输入参数到请求体中。

参数名称	类型	是否必选	说明
Name	String	是	创建的数据集名称。
Url	String	是	数据集位置，只支持阿里云OSS的HTTP/HTTPS链接，需要自学习服务能访问和下载训练数据。
Description	String	否	数据集描述信息。

输出参数：

服务端返回响应体为JSON格式的字符串。

```
{
   "RequestId": "C71B7CAA-18D6-4012-AC3D-425BA1CB****",
   "DataId": "9934e10f19044282825508cbc7c8****"
}
```

参数说明：

- 返回HTTP状态200表示成功，更多状态码请查阅HTTP状态码。
- 返回JSON格式字符串参数：

参数化名称	类型	说明
RequestId	String	请求ID。
DataId	String	数据集ID，作为后续模型训练使用。

> 说明
> 通过轮询数据集的方式，直到状态为Ready，表示数据集已经创建成功。

查询数据集

调用动作：GetAsrLmData

输入参数：

提交查询数据集的请求时，需要设置输入参数到请求体中。

参数名称	类型	是否必选	说明
DataId	String	是	要查询的数据集ID。

输出参数：

服务端返回响应体为JSON格式的字符串。

```
{
  "Data": {
    "Name": "测试数据集",
    "Status": "Ready",
    "Md5": "38fc072ac60796a84ce1a0b13f78****",
    "Description": "通过POP-API创建数据集",
    "Url": "https://aliyun-nls.oss-cn-hangzhou.aliyuncs.com/asr/fileASR/SLP/SLPTest.txt",
    "CreateTime": "2019-02-11 14:40:35",
    "UpdateTime": "2019-02-11 14:40:35",
    "Id": "9934e10f19044282825508cbc7c8****",
    "Size": 5991
  },
  "RequestId": "C88130E6-F3B5-4F3E-9BF5-9C617DDD****"
}
```

参数说明：

- 返回HTTP状态：200表示成功，更多状态码请查阅HTTP状态码。
- 返回JSON格式字符串参数：

参数名称	类型	说明
RequestId	String	请求ID
Data	Data对象	数据集对象

其中，Data对象的参数描述：

参数名称	类型	说明
Id	String	数据集ID，数据集的唯一标识，即创建数据集时获取的DataId。

参数名称	类型	说明
Name	String	数据集名称
Description	String	数据集描述信息
Size	Integer	数据集大小
Md5	String	数据集MD5值
Url	String	创建数据集使用的URL
Status	String	数据集状态。可能的状态Fetching、FetchingFailed、Ready。
CreateTime	String	数据集创建时间
UpdateTime	String	数据集更新时间
ErrorMessage	String	错误消息（Status为错误状态时出现）

删除数据集

调用动作：DeleteAsrLmData

输入参数：

提交删除数据集的请求时，需要设置输入参数到请求体中。

参数名称	类型	是否必选	说明
DataId	String	是	需要删除的数据集ID

输出参数：

服务端返回响应体为JSON格式的字符串。

```
{
  "RequestId": "7130914d32a3441db06747523675****"
}
```

参数说明：

返回HTTP状态：200表示成功，更多状态码请查阅HTTP状态码。

> 说明
>
> 需要删除的数据集状态必须为Ready。

列举数据集

调用动作：ListAsrLmData

输入参数：

提交列举数据集的请求时，需要设置输入参数到请求体中。

参数名称	类型	是否必选	说明
PageNumber	Int	否	页号：从1开始编号，默认值是1。
PageSize	Int	否	页大小：从1到100，默认值10。
ModelId	String	否	模型ID，用于搜索被指定模型使用到的数据。默认列出所有数据集。

输出参数：

服务端返回响应体为JSON格式的字符串。

```
{
  "RequestId": "7130914d32a3441db06747523675****",
  "Page": {
    "Content": [{
      "Id": "1b64bee9994749f2a67eadac6379****",
      "Name": "示例数据集",
      "Description": "这是一个示例数据集",
      "Size": 7777404,
      "Md5": "39326cf690e384735355a385ec1e****",
      "Url": "slp/tmp/demo-data-lm.txt",
      "Status": "Ready",
      "CreateTime": "2018-10-31 17:20:39",
      "UpdateTime": "2018-10-31 17:20:39"
    }],
    "TotalPages": 1,
    "TotalItems": 1,
    "PageNumber": 1,
    "PageSize": 10
  }
}
```


参数说明：

- 返回HTTP状态：200表示成功，更多状态码请查阅HTTP状态码。
- 返回JSON格式字符串参数。

参数名称	类型	说明
RequestId	String	请求ID
Page	Page对象	输入参数中指定获取的页，没有指定默认获取第一页。

其中，Page对象的参数描述如下。

参数名称	类型	说明
Content	List< Data >	数据集数组，参考获取数据集的输出参数Data。

参数名称	类型	说明
TotalPages	Integer	总页数
TotalItems	Integer	总数据集数
PageNumber	Integer	页号，与输入参数相同。
PageSize	Integer	页包含的数据集数量，与输入参数相同。

自学习模型管理

创建自学习模型

调用动作：CreateAsrLmModel

输入参数：

提交请求时，需要设置输入参数到请求体中。

参数名称	类型	是否必选	说明
Name	String	是	模型名称
BaseId	String	是	基础模型ID，创建成功之后不可修改，并在管控台的**项目功能配置**中，将当前模型设置为此基础模型。
Description	String	否	模型描述信息。

其中，支持训练自学习模型的基础模型BaseId如下表所示，模型的详细信息，请在管控台的项目模型中查看。

模型	BaseId
通用中文识别模型（中文普通话 16K）	universal
电话客服及质检模型（中文普通话 8K）	customer_service_8k

模型	BaseId
电商语音购物模型（中文普通话 16K）	e_commerce
政法庭审识别模型（中文普通话 16K）	law_politics
电话客服及质检模型 （中文地方口音 8k）	dialect_customer_service_8k
电话客服及质检模型 (中文粤语 8k)	cantonese_customer_service_8k
英文识别模型 (英语 16k)	english

输出参数：

服务端返回响应体为JSON格式的字符串。

```
{
  "ModelId": "dbb6b71ff3e54b45a600ee5157a2****",
  "RequestId": "945C59DF-B3D9-4F22-808E-76752FF3****"
}
```

参数说明：

- 返回HTTP状态：200表示成功，更多状态码请查阅HTTP状态码。
- 返回JSON格式字符串参数。

参数名称	类型	说明
RequestId	String	请求ID
ModelId	String	创建的自学习模型ID，作为后续的模型训练使用。

> 说明
>
> 创建自学习模型成功的状态为Empty。

查询自学习模型

调用动作：GetAsrLmModel

输入参数：

提交请求时，需要设置输入参数到请求的体中。

参数名称	类型	是否必选	说明
ModelId	String	是	要查询的自学习语言模型ID

输出参数：

服务端返回响应体为JSON格式的字符串。

```
{
    "Model": {
        "Name": "测试自学习模型",
        "Status": "Empty",
        "Description": "测试自学习模型描述",
        "CreateTime": "2019-02-12 10:11:57",
        "UpdateTime": "2019-02-12 10:11:57",
        "Id": "dbb6b71ff3e54b45a600ee5157a2****",
        "BaseId": "common",
        "Size": 0
    },
    "RequestId": "6CE24FF7-B7C8-4B9F-B0EB-FE4AF20B****"
}
```

参数说明：

- 返回HTTP状态：200表示成功，更多状态码请查阅HTTP状态码。
- 返回JSON格式字符串参数。

参数名称	类型	说明
RequestId	String	请求ID
Model	Model对象	自学习模型对象

其中，Mode对象的参数描述。

参数名称	类型	说明

参数名称	类型	说明
Id	String	模型ID，模型的唯一标识，与创建自学习模型的ModelId相同。
Name	String	模型名称
Description	String	模型描述信息
BaseId	String	基础模型ID
Size	Integer	模型大小
Status	String	模型状态。可能的状态：Empty、Training、TrainingFailed、Ready、Deploying和Deployed。
CreateTime	String	模型创建时间
UpdateTime	String	模型更新时间
ErrorMessage	String	错误消息（Status为错误状态时出现）

删除自学习模型

> **注意**
>
> 删除自学习模型时，请确定您的应用没有正在使用该模型，否则将没有该模型的识别效果。

调用动作：DeleteAsrLmModel

输入参数：

提交请求时，需要设置输入参数到请求体中。

参数名称	类型	是否必选	说明
ModelId	String	是	需要删除的自学习语言模型ID

输出参数：

服务端返回响应体为JSON格式的字符串。

```
{
  "RequestId": "7130914d32a3441db06747523675****"
}
```

参数说明：

返回HTTP状态：200表示成功，更多状态码请查阅HTTP状态码。

> 说明
>
> 需要删除的自学习模型状态不能为Training和Deploying。

列举自学习模型

调用动作：ListAsrLmModel

输入参数：

提交请求时，需要设置输入参数到请求体中。

参数名称	类型	是否必选	说明
PageNumber	Int	否	页号：从1开始编号，默认值是1。
PageSize	Int	否	页大小：从1到100，默认值10。
DataId	String	否	数据集ID，用于搜索使用了指定数据的模型。默认列出所有模型。

输出参数：

服务端返回响应体为JSON格式的字符串。

```
{
  "RequestId": "7130914d32a3441db06747523675****",
  "Page": {
    "Content": [{
      "Id": "demo-model",
      "Name": "示例模型",
      "Description": "这是一个示例模型",
      "Size": 0,
      "Status": "Empty",
      "CreateTime": "2018-11-01 17:05:21",
      "UpdateTime": "2018-11-01 17:05:21",
      "BaseId": "common"
    }],
    "TotalPages": 1,
    "TotalItems": 1,
    "PageNumber": 1,
    "PageSize": 10
  }
}
```


参数说明：

- 返回HTTP状态：200表示成功，更多状态码请查阅HTTP状态码。
- 返回JSON格式字符串参数。

参数名称	类型	说明
RequestId	String	请求ID
Page	Page对象	输入参数中指定获取的页，没有指定默认获取第一页。

其中，Page对象的参数描述：

参数名称	类型	说明
Content	List< Model >	Model对象数组，参考获取模型的输出参数Model。
TotalPages	Integer	总页数

参数名称	类型	说明
TotalItems	Integer	总自学习模型数
PageNumber	Integer	页号，与输入参数相同。
PageSize	Integer	页包含的自学习模型数量，与输入参数相同。

自学习模型的训练与发布

添加数据集到自学习模型

> **注意**
>
> 请勿重复添加同一数据集ID到同一自学习模型。

调用动作：AddDataToAsrLmModel

输入参数：

提交请求时，需要设置输入参数到请求体中。

参数名称	类型	是否必选	说明
ModelId	String	是	自学习模型ID
DataId	String	是	数据集ID

输出参数：

服务端返回响应体为JSON格式的字符串。

```
{
  "RequestId": "9B232563-12C0-4242-AA27-C250E1BB****"
}
```

参数说明：

返回HTTP状态：200表示成功，更多状态码请查阅HTTP状态码。

> **说明**
>
> 添加的数据集状态必须为Ready，自学习模型的状态不能为Training和Deploying。

从自学习模型中删除数据集

调用动作：RemoveDataFromAsrLmModel

输入参数：

提交请求时，需要设置输入参数到请求体中。

参数名称	类型	是否必选	说明
ModelId	String	是	自学习模型ID
DataId	String	是	数据集ID

输出参数：

服务端返回响应体为JSON格式的字符串。

```
{
  "RequestId": "7130914d32a3441db06747523675****"
}
```

参数说明：

返回HTTP状态：200表示成功，更多状态码请查阅HTTP状态码。

> 说明
>
> 需要删除的自学习模型状态不能为Training和Deploying。

开始训练自学习模型

调用动作：TrainAsrLmModel

输入参数：

提交请求时，需要设置输入参数到请求体中。

参数名称	类型	是否必选	说明
ModelId	String	是	自学习模型ID

输出参数：

服务端返回响应体为JSON格式的字符串。

```
{
  "RequestId": "3D922A91-68AA-4260-AFE4-C429832F****"
}
```

参数说明：

返回HTTP状态：200表示成功，更多状态码请查阅HTTP状态码。

> 说明
>
> - 开始训练的自学习模型，状态不能为Deploying。
> - 自学习模型训练完成后，默认发布上线，状态为Deployed。

上线自学习模型

调用动作：DeployAsrLmModel

输入参数：

提交请求时，需要设置输入参数到请求体中。

参数名称	类型	是否必选	说明
ModelId	String	是	自学习模型ID

输出参数：

服务端返回响应体为JSON格式的字符串。

```
{
  "RequestId": "D9DDA978-5D68-45A4-B840-E4BC45C7****"
}
```

参数说明：

返回HTTP状态：200表示成功，更多状态码请查阅HTTP状态码。

> 说明
>
> 需要上线的自学习模型状态必须为Ready。

下线自学习模型

> 注意
>
> 下线自学习模型时，请确定您的应用没有正在使用该模型，否则将没有该模型的识别效果。

调用动作：UndeployAsrLmModel

输入参数：

提交请求时，需要设置输入参数到请求体中。

参数名称	类型	是否必选	说明
ModelId	String	是	自学习模型ID

输出参数：

服务端返回响应体为JSON格式的字符串。

```
{
   "RequestId": "8417BA8E-2428-41D2-A849-396A0897****"
}
```

参数说明：

返回HTTP状态：200表示成功，更多状态码请查阅HTTP状态码。

> 说明
>
> 需要下线的自学习模型状态必须为Deployed。

各状态下的操作

> 说明
>
> NO 表示该状态不允许的操作， - 表示该状态允许的操作。

状态/操作	Fetching	FetchingFailed	Ready	Empty	Training	Training Failed	Ready	Deploying	Deployed
CreateAsrLmData	-	-	-	-	-	-	-	-	-
ListAsrLmData	-	-	-	-	-	-	-	-	-
GetAsrLmData	-	-	-	-	-	-	-	-	-

状态/操作	Fetching	FetchingFailed	Ready	Empty	Training	Training Failed	Ready	Deploying	Deployed
DeleteAsrLmData	NO	NO	-	-	NO	-	-	-	-
CreateAsrLmModel	-	-	-	-	-	-	-	-	-
ListAsrLmModel	-	-	-	-	-	-	-	-	-
GetAsrLmModel	-	-	-	-	-	-	-	-	-
DeleteAsrLmModel	-	-	-	-	NO	-	-	NO	-
AddDataToAsrLmModel	NO	NO	-	-	NO	-	-	NO	-
RemoveDataFromAsrLmModel	-	-	-	-	NO	-	-	-	-
TrainAsrLmModel	-	-	-	-	-	-	-	NO	-
DeployAsrLmModel	-	-	-	NO	NO	NO	-	NO	NO

状态/操作	Fetching	FetchingFailed	Ready	Empty	Training	Training Failed	Ready	Deploying	Deployed
UndeployAsrLmModel	-	-	-	NO	NO	NO	NO	NO	-

错误码

以上操作在出现错误时，服务端返回的响应体中包含了错误信息。您可以根据错误信息提示，检查程序是否存在问题。

错误名称	说明
SLP.ASR_MODEL_ERROR	自学习模型相关错误。
SLP.NOT_FOUND	指定的ID无效，无法根据ID找到指定的资源。
SLP.PARAMETER_ERROR	创建资源时设置了无效的参数。
SLP.EXCEED_LIMIT	资源数量超过限制，无法创建新的资源。

以NOT_FOUND错误为例：

```
{
  "RequestId": "E70F51F6-23E3-4681-B954-ABF32B89****",
  "HostId": "nls-slp.cn-shanghai.aliyuncs.com",
  "Code": "SLP.NOT_FOUND",
  "Message": "Model not found!"
}
```

参数名称	类型	说明
RequestId	String	请求ID
HostId	String	自学习服务端的域名

参数名称	类型	说明
Code	String	错误码
Message	String	错误信息描述

代码示例

> 说明
>
> - 本示例以Java语言为例，使用了阿里云Java SDK的CommonRequest提交请求，采用RPC风格的POP API调用。阿里云SDK的详细介绍参见阿里云SDK开发指南，Java SDK CommonRequest的使用方法参见使用CommonRequest进行调用。
> - 鉴权SDK使用过程中，所有的接口调用均通过阿里云账号完成鉴权操作。通过传入阿里云账号的AccessKey ID和AccessKey Secret（获取方法参见开通服务），调用阿里云Java SDK，创建IAcsClient对象，全局唯一。
> - 示例中包含了三个类：
>   - AsrLmModelPopApiDemo：POP API测试类，包含main函数，用于测试接口，您可根据此类，进行集成时的状态检查。
>   - AsrLmData：数据集类，封装了数据集相关操作的数据对象和接口调用。
>   - AsrLmModel：自学习模型类，封装了自学习模型相关操作的数据对象和接口调用。

添加Java依赖：只需要依赖阿里云Java SDK的核心库与阿里云开源库fastjson。阿里云Java SDK的核心库版本可以选择3.5.0至3.7.x的版本。

```
<dependency>
  <groupId>com.aliyun</groupId>
  <artifactId>aliyun-java-sdk-core</artifactId>
  <version>3.7.1</version>
</dependency>
<dependency>
  <groupId>com.alibaba</groupId>
  <artifactId>fastjson</artifactId>
  <version>1.2.49</version>
</dependency>
```

- **POP API测试类**

```
import com.alibaba.fastjson.JSONObject;
import com.aliyuncs.DefaultAcsClient;
import com.aliyuncs.IAcsClient;
import com.aliyuncs.profile.DefaultProfile;
```

```
import com.aliyuncs.profile.DefaultProfile;
public class AsrLmModelPopApiDemo {
    private static String REGION = "cn-shanghai";
    private static final String STATUS_FETCHING = "Fetching";
    private static final String STATUS_FETCHINGFAILED = "FetchingFailed";
    private static final String STATUS_READY = "Ready";
    private static final String STATUS_EMPTY = "Empty";
    private static final String STATUS_TRAINING = "Training";
    private static final String STATUS_TRAININGFAILED = "TrainingFailed";
    private static final String STATUS_DEPLOYING = "Deploying";
    private static final String STATUS_DEPLOYED = "Deployed";
    private static IAcsClient client;
    private AsrLmData asrLmData;
    private AsrLmModel asrLmModel;
    public AsrLmModelPopApiDemo(String akId, String akSecret) {
        DefaultProfile profile = DefaultProfile.getProfile(REGION, akId, akSecret);
        client = new DefaultAcsClient(profile);
        asrLmData = new AsrLmData(client);
        asrLmModel = new AsrLmModel(client);
    }
    /****************************** 数据集管理 ******************************/
    // 创建数据集
    public String createAsrLmData(String name, String fileUrl, String description) {
        String dataId = asrLmData.createAsrLmData(name, fileUrl, description);
        if (null == dataId) {
            return dataId;
        }
        // 轮询数据集，检查状态是否为Ready，即数据集导入成功。
        while (true) {
            AsrLmData.LmData data = asrLmData.getAsrLmData(dataId);
            if (null == data) {
                dataId = null;
                break;
            }
            if (data.Status.equals(STATUS_FETCHING)) {
                System.out.println("正在将数据集导入到自学习服务中，dataId: " + dataId);
                try {
                    Thread.sleep(100);
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
```

```
        }
        else if (data.Status.equals(STATUS_FETCHINGFAILED)) {
            System.out.println("复制数据集出现错误，dataId: " + dataId);
            asrLmData.deleteAsrLmData(dataId);
            dataId = null;
            break;
        }
        else if (data.Status.equals(STATUS_READY)) {
            System.out.println("数据集导入成功，dataId: " + dataId);
            break;
        }
    }
    return dataId;
}
// 查询数据集
public AsrLmData.LmData getAsrLmData(String dataId) {
    return asrLmData.getAsrLmData(dataId);
}
// 删除数据集
public boolean deleteAsrLmData(String dataId) {
    AsrLmData.LmData data = asrLmData.getAsrLmData(dataId);
    if (null == data) {
        return false;
    }
    if (!data.Status.equals(STATUS_READY)) {
        System.out.println("数据集状态不允许进行删除操作，status: " + data.Status);
        return false;
    }
    return asrLmData.deleteAsrLmData(dataId);
}
// 列举数据集
public AsrLmData.LmDataPage listAsrLmData() {
    return asrLmData.listAsrLmData();
}
/****************************** 自学习模型管理 ******************************/
// 创建自学习模型
public String createAsrLmModel(String name, String baseId, String description) {
    String modelId = asrLmModel.createAsrLmModel(name, baseId, description);
    if (null == modelId) {
        return modelId;
    }
```

```
    // 轮询自学习模型，检查模型状态是否是Empty，即新创建的自学习模型。
    while (true) {
        AsrLmModel.LmModel model = asrLmModel.getAsrLmModel(modelId);
        if (null == model) {
            modelId = null;
            break;
        }
        if (model.Status.equals(STATUS_EMPTY)) {
            break;
        }
        else {
            System.out.println("创建自学习模型失败，modelId: " + modelId);
            asrLmModel.deleteAsrLmModel(modelId);
            modelId = null;
            break;
        }
    }
    return modelId;
}
// 获取自学习模型
public AsrLmModel.LmModel getAsrLmModel(String modelId) {
    return asrLmModel.getAsrLmModel(modelId);
}
// 删除自学习模型
public boolean deleteAsrLmModel(String modelId) {
    AsrLmModel.LmModel model = asrLmModel.getAsrLmModel(modelId);
    if (null == model) {
        return false;
    }
    if (model.Status.equals(STATUS_TRAINING) || model.Status.equals(STATUS_DEPLOYING)) {
        System.out.println("自学习模型状态不允许进行删除操作，status: " + model.Status);
        return false;
    }
    return asrLmModel.deleteAsrLmModel(modelId);
}
// 列举自学习模型
public AsrLmModel.LmModelPage listAsrLmModel() {
    return asrLmModel.listAsrLmModel();
}
/************************** 自学习模型的训练与发布 **************************/
// 添加数据集到自学习模型
```

```
// 添加数据集到自学习模型
public boolean addDataToAsrLmModel(String dataId, String modelId) {
    AsrLmData.LmData data = asrLmData.getAsrLmData(dataId);
    if (null == data) {
        return false;
    }
    if (!data.Status.equals(STATUS_READY)) {
        System.out.println("数据集状态不允许进行添加到自学习模型操作，status: " + data.Status);
        return false;
    }
    AsrLmModel.LmModel model = asrLmModel.getAsrLmModel(modelId);
    if (null == model) {
        return false;
    }
    if (model.Status.equals(STATUS_TRAINING) || model.Status.equals(STATUS_DEPLOYING)) {
        System.out.println("自学习模型状态不允许进行添加数据集操作，status: " + model.Status);
        return false;
    }
    return asrLmModel.addDataToAsrLmModel(dataId, modelId);
}
// 从自学习模型中删除数据集
public boolean removeDataFromAsrLmModel(String dataId, String modelId) {
    // 列举指定模型使用到的数据集，判断待删除的数据集是否已经添加到该模型。
    boolean isAdded = false;
    AsrLmData.LmDataPage page = asrLmData.listAsrLmData(1, 10, modelId);
    if (page != null && page.Content.size() > 0) {
        for (int i = 0; i < page.Content.size(); i++) {
            if (dataId.equals(page.Content.get(i).Id)) {
                isAdded = true;
                break;
            }
        }
    }
    if (!isAdded) {
        System.out.println("待删除的数据集没有添加到指定模型，不能进行删除操作！");
        return false;
    }
    // 检查模型状态是否允许删除数据集操作
    AsrLmModel.LmModel model = asrLmModel.getAsrLmModel(modelId);
    if (null == model) {
        return false;
```

```
    }
    if (model.Status.equals(STATUS_TRAINING)) {
        System.out.println("自学习模型状态不允许进行删除数据集操作，status: " + model.Status);
        return false;
    }
    return asrLmModel.removeDataFromAsrLmModel(dataId, modelId);
}
// 训练自学习模型
public boolean trainAsrLmModel(String modelId) {
    AsrLmModel.LmModel model = asrLmModel.getAsrLmModel(modelId);
    if (null == model) {
        return false;
    }
    if (model.Status.equals(STATUS_DEPLOYING)) {
        System.out.println("自学习模型状态不允许进行训练操作，status: " + model.Status);
        return false;
    }
    boolean isTrain = asrLmModel.trainAsrLmModel(modelId);
    if (!isTrain) {
        return isTrain;
    }
    // 轮询自学习模型，直到状态为Deployed，即自学习模型训练成功且并已上线。
    while (true) {
        model = asrLmModel.getAsrLmModel(modelId);
        if (null == model) {
            isTrain = false;
            break;
        }
        if (model.Status.equals(STATUS_TRAINING) || model.Status.equals(STATUS_DEPLOYING)) {
            if (model.Status.equals(STATUS_TRAINING)) {
                System.out.println("自学习模型正在训练中，modelId: " + modelId);
            }
            else {
                System.out.println("自学习模型正在上线，modelId: " + modelId);
            }
            try {
                Thread.sleep(5000);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
```

```
        else if (model.Status.equals(STATUS_TRAININGFAILED)) {
            System.out.println("自学习模型训练失败, modelId: " + modelId);
            isTrain = false;
            break;
        }
        else if (model.Status.equals(STATUS_DEPLOYED)) {
            System.out.println("自学习模型训练成功并已上线，modeId: " + modelId);
            isTrain = true;
            break;
        }
        else {
            System.out.println("自学习模型状态不允许进行训练操作，Status: " + model.Status);
            isTrain = false;
            break;
        }
    }
    return isTrain;
}
// 上线自学习模型
public boolean deployAsrLmModel(String modelId) {
    AsrLmModel.LmModel model = asrLmModel.getAsrLmModel(modelId);
    if (null == model) {
        return false;
    }
    if (!model.Status.equals(STATUS_READY)) {
        System.out.println("自学习模型状态不允许进行上线操作，status: " + model.Status);
        return false;
    }
    boolean isDeployed = asrLmModel.deployAsrLmModel(modelId);
    if (!isDeployed) {
        return isDeployed;
    }
    // 轮询自学习模型，检查状态是否是Deployed，即自学习模型已上线。
    while (true) {
        model = asrLmModel.getAsrLmModel(modelId);
        if (null == model) {
            isDeployed = false;
            break;
        }
        if (model.Status.equals(STATUS_DEPLOYING)) {
            System.out.println("自学习模型正在上线, modelId: " + modelId);
```

```
            System.out.println("自学习模型正在上线，modelId: " + modelId);
            try {
                Thread.sleep(100);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
        else if (model.Status.equals(STATUS_DEPLOYED)) {
            System.out.println("自学习模型已经上线，modelId: " + modelId);
            isDeployed = true;
            break;
        }
        else {
            System.out.println("自学习模型的状态不允许上线操作，Status: " + model.Status);
            isDeployed = false;
            break;
        }
    }
    return isDeployed;
}
// 下线自学习模型
public boolean undeployAsrLmModel(String modelId) {
    AsrLmModel.LmModel model = asrLmModel.getAsrLmModel(modelId);
    if (null == model) {
        return false;
    }
    if (!model.Status.equals(STATUS_DEPLOYED)) {
        System.out.println("自学习模型的状态不允许进行下线操作，status: " + model.Status);
        return false;
    }
    boolean isUnDeployed = asrLmModel.undeployAsrLmModel(modelId);
    if (!isUnDeployed) {
        return isUnDeployed;
    }
    // 轮询自学习模型，检查状态是否是Ready，即自学习模型下线完成。
    while (true) {
        model = asrLmModel.getAsrLmModel(modelId);
        if (null == model) {
            isUnDeployed = false;
            break;
        }
```

```
        if (model.Status.equals(STATUS_DEPLOYING)) {
            System.out.println("自学习模型正在下线, modelId: " + modelId);
            try {
                Thread.sleep(100);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
        else if (model.Status.equals(STATUS_READY)) {
            System.out.println("自学习模型下线完成，modelId: " + modelId);
            isUnDeployed = true;
            break;
        }
        else {
            System.out.println("自学习模型的状态不允许进行下线操作，Status: " + model.Status);
            isUnDeployed = false;
            break;
        }
    }
    return isUnDeployed;
  }
  public static void main(String[] args) {
    if (args.length < 2) {
        System.err.println("AsrLmModelPopApiDemo need params: <AccessKey Id> <AccessKey Secre
t>");
        return;
    }
    String accessKeyId = args[0];
    String accessKeySecret = args[1];
    AsrLmModelPopApiDemo demo = new AsrLmModelPopApiDemo(accessKeyId, accessKeySecret);
    /****************************** 数据集管理 ******************************/
    // 数据集ID
    String dataId;
    // 创建数据集
    String name = "测试数据集";
    String fileUrl = "https://aliyun-nls.oss-cn-hangzhou.aliyuncs.com/asr/fileASR/SLP/SLPTest.txt"
;
    String description = "通过POP-API创建数据集";
    dataId = demo.createAsrLmData(name, fileUrl, description);
    if (dataId != null) {
        System.out.println("创建数据集成功，数据集ID: " + dataId);
```

```
}
else {
    System.out.println("创建数据集失败！");
}
// 查询数据集
AsrLmData.LmData data = demo.getAsrLmData(dataId);
if (data != null) {
    System.out.println("获取数据集成功：" + JSONObject.toJSONString(data));
}
else {
    System.out.println("获取数据集失败！");
}
// 列举所有数据集
AsrLmData.LmDataPage page = demo.listAsrLmData();
if (page != null) {
    System.out.println("列举词表成功：" + JSONObject.toJSONString(page));
}
else {
    System.out.println("列举词表失败！");
    return;
}
/************************* 自学习模型管理 *********************************/
// 自学习模型ID
String modelId;
// 创建自学习模型
String modelName = "测试自学习模型";
// 使用的是通用模型，请根据业务需求选择合适的基础模型。
String baseId = "universal";
String modelDescription = "测试自学习模型描述";
modelId = demo.createAsrLmModel(modelName, baseId, modelDescription);
if (modelId != null) {
    System.out.println("创建自学习模型成功，模型ID：" + modelId);
}
else {
    System.out.println("创建自学习模型失败！");
}
// 查询自学习模型
AsrLmModel.LmModel model = demo.getAsrLmModel(modelId);
if (model != null) {
    System.out.println("获取自学习模型成功：" + JSONObject.toJSONString(model));
}
```

```
}
else {
    System.out.println("获取自学习模型失败！");
}
// 列举自学习模型
AsrLmModel.LmModelPage modelPage = demo.listAsrLmModel();
if (modelPage != null) {
    System.out.println("列举自学习模型成功：" + JSONObject.toJSONString(modelPage));
}
else {
    System.out.println("列举自学习模型失败！");
}
/****************************** 自学习模型的训练与发布 ******************************/
// 添加数据集到自学习模型
boolean isAdded = demo.addDataToAsrLmModel(dataId, modelId);
if (isAdded) {
    System.out.println("添加数据集到自学习模型成功！");
}
else {
    System.out.println("添加数据集到自学习模型失败！");
}
// 训练自学习模型
boolean isTrian = demo.trainAsrLmModel(modelId);
if (isTrian) {
    System.out.println("训练自学习模型成功！");
}
else {
    System.out.println("训练自学习模型失败！");
}
// 下线自学习模型
boolean isUnDeployed = demo.undeployAsrLmModel(modelId);
if (isUnDeployed) {
    System.out.println("自学习模型下线成功！");
}
else {
    System.out.println("自学习模型下线失败！");
}
// 上线自学习模型
boolean isDeployed = demo.deployAsrLmModel(modelId);
if (isDeployed) {
    System.out.println("自学习模型上线成功！");
```

```
        }
        else {
            System.out.println("自学习模型上线失败！");
        }
        /***************************** 清理数据集和模型，请根据需求操作 *****************************/
        // 1. 下线自学习模型，需要时开启。
        isUnDeployed = demo.undeployAsrLmModel(modelId);
        if (isUnDeployed) {
            System.out.println("自学习模型下线成功！");
        }
        else {
            System.out.println("自学习模型下线失败！");
        }
        // 2. 从自学习模型中删除数据集，需要时开启。
        boolean isDeleted = demo.removeDataFromAsrLmModel(dataId, modelId);
        if (isDeleted) {
            System.out.println("从自学习模型中删除数据集成功！");
        }
        else {
            System.out.println("从自学习模型中删除数据集失败！");
        }
        // 3. 删除数据集，需要时开启。
        boolean isDeletedData = demo.deleteAsrLmData(dataId);
        if (isDeletedData) {
            System.out.println("删除数据集成功！");
        }
        else {
            System.out.println("删除数据集失败！");
        }
        // 4. 删除自学习模型，需要时开启。
        boolean isDeletedModel = demo.deleteAsrLmModel(modelId);
        if (isDeletedModel) {
            System.out.println("删除自学习模型成功！");
        }
        else {
            System.out.println("删除自学习模型失败！");
        }
    }
}
```

- 数据集类

```
import com.alibaba.fastjson.JSONObject;
import com.aliyuncs.CommonRequest;
import com.aliyuncs.CommonResponse;
import com.aliyuncs.IAcsClient;
import com.aliyuncs.exceptions.ClientException;
import com.aliyuncs.http.MethodType;
import com.aliyuncs.http.ProtocolType;
import java.util.ArrayList;
import java.util.List;
public class AsrLmData {
    public static class LmData {
        public String Name;
        public String Status;
        public String Description;
        public String Url;
        public String CreateTime;
        public String UpdateTime;
        public String Id;
        public String ErrorMessage;  // 只有Status为错误状态时有该信息。
        public int Size;
    }
    public static class LmDataPage {
        public int PageNumber;
        public int PageSize;
        public int TotalItems;
        public int TotalPages;
        public List<LmData> Content = new ArrayList<LmData>();
    }
    private static final String VERSION = "2018-11-20";
    private static final String DOMAIN = "nls-slp.cn-shanghai.aliyuncs.com";
    private static ProtocolType PROTOCOL_TYPE = ProtocolType.HTTPS;
    private static final String KEY_NAME = "Name";
    private static final String KEY_URL = "Url";
    private static final String KEY_DESCRIPTION = "Description";
    private static final String KEY_DATA_ID = "DataId";
    private static final String KEY_DATA = "Data";
    private static final String KEY_PAGE = "Page";
    private static final String KEY_PAGE_NUMBER = "PageNumber";
    private static final String KEY_PAGE_SIZE = "PageSize";
    private static final String KEY_MODEL_ID = "ModelId";
```

```
private IAcsClient client;
private CommonRequest newRequest(String action) {
    CommonRequest request = new CommonRequest();
    request.setDomain(DOMAIN);
    request.setProtocol(PROTOCOL_TYPE);
    request.setVersion(VERSION);
    request.setMethod(MethodType.POST);
    request.setAction(action);
    return request;
}
public AsrLmData(IAcsClient client) {
    this.client = client;
}
/**
 * 创建数据集
 * @param name 创建的数据集名称，必填。
 * @param fileUrl 数据集文件的链接，需要服务端可下载，必填。
 * @param  description 数据集描述信息，可选。
 * @return String 创建的数据集ID。
 */
public String createAsrLmData(String name, String fileUrl, String description) {
    CommonRequest request = newRequest("CreateAsrLmData");
    request.putBodyParameter(KEY_NAME, name);
    request.putBodyParameter(KEY_URL, fileUrl);
    request.putBodyParameter(KEY_DESCRIPTION, description);
    CommonResponse response = null;
    try {
        response = client.getCommonResponse(request);
    } catch (ClientException e) {
        e.printStackTrace();
    }
    System.out.println("CreateAsrLmData: " + response.getData());
    if (response == null || response.getHttpStatus() != 200) {
        System.out.println(response.getData());
        System.out.println("创建数据集失败，HTTP错误码：" + response.getHttpStatus());
        return null;
    }
    JSONObject result = JSONObject.parseObject(response.getData());
    String dataId = result.getString(KEY_DATA_ID);
    return dataId;
}
```

```
/**
 * 查询数据集
 * @param dataId 数据集ID
 * @return LmData 获取的数据集对象
 */
public LmData getAsrLmData(String dataId) {
    CommonRequest request = newRequest("GetAsrLmData");
    request.putBodyParameter(KEY_DATA_ID, dataId);
    CommonResponse response = null;
    try {
        response = client.getCommonResponse(request);
    } catch (ClientException e) {
        e.printStackTrace();
    }
    System.out.println("GetAsrLmData: " + response.getData());
    if (response == null || response.getHttpStatus() != 200) {
        System.out.println(response.getData());
        System.out.println("获取数据集失败，HTTP错误码：" + response.getHttpStatus());
        return null;
    }
    JSONObject result = JSONObject.parseObject(response.getData());
    String dataJson = result.getString(KEY_DATA);
    LmData data = JSONObject.parseObject(dataJson, LmData.class);
    return data;
}
/**
 * 删除数据集
 * @param dataId 需要删除的数据集ID，必填。
 * @return 是否删除成功。
 */
public boolean deleteAsrLmData(String dataId) {
    CommonRequest request = newRequest("DeleteAsrLmData");
    request.putBodyParameter(KEY_DATA_ID, dataId);
    CommonResponse response = null;
    try {
        response = client.getCommonResponse(request);
    } catch (ClientException e) {
        e.printStackTrace();
    }
    System.out.println("DeleteAsrLmData: " + response.getData());
    if (response == null || response.getHttpStatus() != 200) {
```

```
        System.out.println(response.getData());
        System.out.println("删除数据集失败，HTTP错误码：" + response.getHttpStatus());
        return false;
      }
      return true;
    }
    /**
     * 列举数据集
     * @param pageNumber 页号，从1开始编号，可选，默认值是1。
     * @param pageSize  页大小，从1到100，可选，默认值是10。
     * @param modelId 模型ID，用于搜索被指定模型使用到的数据集，可选，默认列出所有数据集。
     * @return 数据集信息。
     */
    public LmDataPage listAsrLmData(int pageNumber, int pageSize, String modelId) {
      CommonRequest request = newRequest("ListAsrLmData");
      request.putBodyParameter(KEY_PAGE_NUMBER, pageNumber);
      request.putBodyParameter(KEY_PAGE_SIZE, pageSize);
      request.putBodyParameter(KEY_MODEL_ID, modelId);
      CommonResponse response = null;
      try {
        response = client.getCommonResponse(request);
      } catch (ClientException e) {
        e.printStackTrace();
      }
      System.out.println("ListAsrLmData: " + response.getData());
      if (response == null || response.getHttpStatus() != 200) {
        System.out.println(response.getData());
        System.out.println("列举数据集失败，HTTP错误码：" + response.getHttpStatus());
        return null;
      }
      JSONObject result = JSONObject.parseObject(response.getData());
      String pageJson = result.getString(KEY_PAGE);
      LmDataPage page = JSONObject.parseObject(pageJson, LmDataPage.class);
      return page;
    }
    public LmDataPage listAsrLmData() {
      return listAsrLmData(1, 10, null);
    }
}
```

- 自学习模型类

```
import com.alibaba.fastjson.JSONObject;
import com.aliyuncs.CommonRequest;
import com.aliyuncs.CommonResponse;
import com.aliyuncs.IAcsClient;
import com.aliyuncs.exceptions.ClientException;
import com.aliyuncs.http.MethodType;
import com.aliyuncs.http.ProtocolType;
import java.util.ArrayList;
import java.util.List;
public class AsrLmModel {
    public static class LmModel {
        public String Name;
        public String Status;
        public String Description;
        public String CreateTime;
        public String UpdateTime;
        public String Id;
        public String BaseId;
        public String ErrorMessage;  // 只有Status为错误状态时有该信息。
        public int Size;
    }
    public static class LmModelPage {
        public int PageNumber;
        public int PageSize;
        public int TotalItems;
        public int TotalPages;
        public List<LmModel> Content = new ArrayList<LmModel>();
    }
    private static final String VERSION = "2018-11-20";
    private static final String DOMAIN = "nls-slp.cn-shanghai.aliyuncs.com";
    private static ProtocolType PROTOCOL_TYPE = ProtocolType.HTTPS;
    private static final String KEY_NAME = "Name";
    private static final String KEY_BASE_ID = "BaseId";
    private static final String KEY_DESCRIPTION = "Description";
    private static final String KEY_MODEL_ID = "ModelId";
    private static final String KEY_MODEL = "Model";
    private static final String KEY_PAGE = "Page";
    private static final String KEY_PAGE_NUMBER = "PageNumber";
    private static final String KEY_PAGE_SIZE = "PageSize";
```

```
    private static final String KEY_DATA_ID = "DataId";
    private IAcsClient client;
    private CommonRequest newRequest(String action) {
        CommonRequest request = new CommonRequest();
        request.setDomain(DOMAIN);
        request.setProtocol(PROTOCOL_TYPE);
        request.setVersion(VERSION);
        request.setMethod(MethodType.POST);
        request.setAction(action);
        return request;
    }
    public AsrLmModel(IAcsClient client) {
        this.client = client;
    }
    /**
     * 创建模型
     * @param name 自学习模型名称，必填。
     * @param baseId 基础模型ID，创建成功后不可修改，必填。
     * @param description 自学习模型描述信息，可选。
     * @return 创建的模型ID。
     */
    public String createAsrLmModel(String name, String baseId, String description) {
        CommonRequest request = newRequest("CreateAsrLmModel");
        request.putBodyParameter(KEY_NAME, name);
        request.putBodyParameter(KEY_BASE_ID, baseId);
        request.putBodyParameter(KEY_DESCRIPTION, description);
        CommonResponse response = null;
        try {
            response = client.getCommonResponse(request);
        } catch (ClientException e) {
            e.printStackTrace();
        }
        System.out.println("CreateAsrLmModel: " + response.getData());
        if (response == null || response.getHttpStatus() != 200) {
            System.out.println(response.getData());
            System.out.println("创建自学习模型失败，HTTP错误码：" + response.getHttpStatus());
            return null;
        }
        JSONObject result = JSONObject.parseObject(response.getData());
        String modelId = result.getString(KEY_MODEL_ID);
        return modelId;
```

```
}
/**
 * 查询自学习模型
 * @param modelId 模型ID
 * @return 获取的模型对象
 */
public LmModel getAsrLmModel(String modelId) {
    CommonRequest request = newRequest("GetAsrLmModel");
    request.putBodyParameter(KEY_MODEL_ID, modelId);
    CommonResponse response = null;
    try {
        response = client.getCommonResponse(request);
    } catch (ClientException e) {
        e.printStackTrace();
    }
    System.out.println("GetAsrLmModel: " + response.getData());
    if (response == null || response.getHttpStatus() != 200) {
        System.out.println(response.getData());
        System.out.println("查询自学习模型失败，HTTP错误码：" + response.getHttpStatus());
        return null;
    }
    JSONObject result = JSONObject.parseObject(response.getData());
    String modelJson = result.getString(KEY_MODEL);
    LmModel model = JSONObject.parseObject(modelJson, LmModel.class);
    return model;
}
/**
 * 删除自学习模型
 * @param modelId 需要删除的自学习模型ID
 * @return 是否删除成功
 */
public boolean deleteAsrLmModel(String modelId) {
    CommonRequest request = newRequest("DeleteAsrLmModel");
    request.putBodyParameter(KEY_MODEL_ID, modelId);
    CommonResponse response = null;
    try {
        response = client.getCommonResponse(request);
    } catch (ClientException e) {
        e.printStackTrace();
    }
    System.out.println("DeleteAsrLmModel: " + response.getData());
```

```
        System.out.println("DeleteAsrLmModel: " + response.getData());
        if (response == null || response.getHttpStatus() != 200) {
            System.out.println(response.getData());
            System.out.println("删除自学习模型失败，HTTP错误码：" + response.getHttpStatus());
            return false;
        }
        return true;
    }
    /**
     * 列举自学习模型
     * @param pageNumber 页号，从1开始，可选，默认值是1。
     * @param pageSize 页大小，从1到100，可选，默认值是10。
     * @param dataId 数据集ID，用于搜索使用了指定数据集的模型，可选，默认列出所有自学习模型。
     * @return 自学习模型信息。
     */
    public LmModelPage listAsrLmModel(int pageNumber, int pageSize, String dataId) {
        CommonRequest request = newRequest("ListAsrLmModel");
        request.putBodyParameter(KEY_PAGE_NUMBER, pageNumber);
        request.putBodyParameter(KEY_PAGE_SIZE, pageSize);
        request.putBodyParameter(KEY_DATA_ID, dataId);
        CommonResponse response = null;
        try {
            response = client.getCommonResponse(request);
        } catch (ClientException e) {
            e.printStackTrace();
        }
        System.out.println("ListAsrLmModel: " + response.getData());
        if (response == null || response.getHttpStatus() != 200) {
            System.out.println(response.getData());
            System.out.println("列举自学习模型失败，HTTP错误码：" + response.getHttpStatus());
            return null;
        }
        JSONObject result = JSONObject.parseObject(response.getData());
        String pageJson = result.getString(KEY_PAGE);
        LmModelPage page = JSONObject.parseObject(pageJson, LmModelPage.class);
        return page;
    }
    public LmModelPage listAsrLmModel() {
        return listAsrLmModel(1, 10, null);
    }
    /**
```

```
     * 添加数据集到自学习模型
     * @param dataId 需要添加的数据集ID
     * @param modelId 自学习模型ID
     * @return 是否添加成功
     */
    public boolean addDataToAsrLmModel(String dataId, String modelId) {
        CommonRequest request = newRequest("AddDataToAsrLmModel");
        request.putBodyParameter(KEY_DATA_ID, dataId);
        request.putBodyParameter(KEY_MODEL_ID, modelId);
        CommonResponse response = null;
        try {
            response = client.getCommonResponse(request);
        } catch (ClientException e) {
            e.printStackTrace();
        }
        System.out.println("AddDataToAsrLmModel: " + response.getData());
        if (response == null || response.getHttpStatus() != 200) {
            System.out.println(response.getData());
            System.out.println("添加数据集到自学习模型失败，HTTP错误码：" + response.getHttpStatus());
            return false;
        }
        return true;
    }
    /**
     * 从自学习模型中删除数据集
     * @param dataId 需要删除的数据集
     * @param modelId 自学习模型ID
     * @return 是否删除成功
     */
    public boolean removeDataFromAsrLmModel(String dataId, String modelId) {
        CommonRequest request = newRequest("RemoveDataFromAsrLmModel");
        request.putBodyParameter(KEY_DATA_ID, dataId);
        request.putBodyParameter(KEY_MODEL_ID, modelId);
        CommonResponse response = null;
        try {
            response = client.getCommonResponse(request);
        } catch (ClientException e) {
            e.printStackTrace();
        }
        System.out.println("RemoveDataFromAsrLmModel: " + response.getData());
        if (response == null || response.getHttpStatus() != 200) {
```

```
            System.out.println(response.getData());
            System.out.println("从自学习模型中删除数据集失败，HTTP错误码：" + response.getHttpStatus());
            return false;
        }
        return true;
    }
    /**
     * 开始训练自学习模型
     * @param modelId 需要开始训练的自学习模型ID
     * @return 是否开始训练成功
     * */
    public boolean trainAsrLmModel(String modelId) {
        CommonRequest request = newRequest("TrainAsrLmModel");
        request.putBodyParameter(KEY_MODEL_ID, modelId);
        CommonResponse response = null;
        try {
            response = client.getCommonResponse(request);
        } catch (ClientException e) {
            e.printStackTrace();
        }
        System.out.println("TrainAsrLmModel: " + response.getData());
        if (response == null || response.getHttpStatus() != 200) {
            System.out.println(response.getData());
            System.out.println("开始训练自学习模型失败，HTTP错误码：" + response.getHttpStatus());
            return false;
        }
        return true;
    }
    /**
     * 上线自学习模型
     * @param modelId 需要上线的自学习模型ID
     * @return 是否上线成功
     */
    public boolean deployAsrLmModel(String modelId) {
        CommonRequest request = newRequest("DeployAsrLmModel");
        request.putBodyParameter(KEY_MODEL_ID, modelId);
        CommonResponse response = null;
        try {
            response = client.getCommonResponse(request);
        } catch (ClientException e) {
            e.printStackTrace();
```

```
            e.printStackTrace();
        }
        System.out.println("DeployAsrLmModel: " + response.getData());
        if (response == null || response.getHttpStatus() != 200) {
            System.out.println(response.getData());
            System.out.println("上线自学习模型失败，HTTP错误码：" + response.getHttpStatus());
            return false;
        }
        return true;
    }
    /**
     * 下线自学习模型
     * @param modelId 需要下线的自学习模型ID
     * @return 是否下线成功
     */
    public boolean undeployAsrLmModel(String modelId) {
        CommonRequest request = newRequest("UndeployAsrLmModel");
        request.putBodyParameter(KEY_MODEL_ID, modelId);
        CommonResponse response = null;
        try {
            response = client.getCommonResponse(request);
        } catch (ClientException e) {
            e.printStackTrace();
        }
        System.out.println("UndeployAsrLmModel: " + response.getData());
        if (response == null || response.getHttpStatus() != 200) {
            System.out.println(response.getData());
            System.out.println("下线自学习模型失败，HTTP错误码：" + response.getHttpStatus());
            return false;
        }
        return true;
    }
}
```

2.4. 使用SDK 2.0设置自学习模型

本文为您介绍如何在SDK中使用POP API设置自学习模型。

使用POP API训练自学习模型

使用POP API训练获取的自学习模型，需要在SDK中设置其模型ID才可以使用。下面介绍在一句话识别、实时语音识别、录音文件识别中如何设置自学习模型。

一句话识别

在一句话识别中，需要通过设置高级参数 `customization_id` 指定自学习模型ID。

- Java SDK

  > **说明**
  >
  > 请首先阅读Java SDK，了解Java SDK的基本用法。

  由于SDK中没有 `customization_id` 参数对应的set方法，需要通过SpeechRecognizer类中的addCustomedParam方法进行设置：

  ```
  public void addCustomedParam(String key, Object value);
  ```

  示例中创建SpeechRecognizer的实例对象recognizer后（调用start方法之前），调用该方法设置自学习模型ID：

  ```
  SpeechRecognizer recognizer = new SpeechRecognizer(client, getRecognizerListener());
  ... // 省略其他设置。
  recognizer.addCustomedParam("customization_id", "您的自学习模型ID");
  ```

- C++ SDK

  > **说明**
  >
  > 请首先阅读C++ SDK，了解C++ SDK的基本用法。

  由于SDK中没有 `customization_id` 参数对应的set方法，需要通过SpeechRecognizerRequest类中的setPayloadParam方法进行设置：

  ```
  /**
   * @brief 参数设置。
   * @param value JSON Map格式的字符串。
   * @return 成功则返回0，否则返回-1。
   */
  int setPayloadParam(const char value);
  ```

  在示例中创建SpeechRecognizerRequest的实例对象request后（调用start方法之前），调用该方法设置自学习模型ID：

  ```
  SpeechRecognizerRequest *request = NlsClient::getInstance()->createRecognizerRequest(callback)
  ;
  ... // 省略其他设置。
  request->setPayloadParam("{\"customization_id\":\"您的自学习模型ID\"}");
  ```

- iOS SDK

> **说明**
>
> 请先阅读iOS SDK，了解iOS SDK的基本用法。

通过RequestParam对象的setParams方法设置自学习模型ID。

```
NSMutableDictionary *userParams = [[NSMutableDictionary alloc]init];
[userParams setValue:@"您的自学习模型ID" forKey:@"customization_id"];
[_recognizeRequestParam setParams:userParams];
```

- Android SDK

> **说明**
>
> 请先阅读Android SDK，了解Android SDK的基本用法。

通过SpeechRecognizer对象的setParams方法设置自学习模型ID。

```
String userParamString = "{\"customization_id\":\"您的自学习模型ID\"}";
speechRecognizer.setParams(userParamString);
```

- RESTful API

> **说明**
>
> 请首先阅读RESTFUL API，了解RESTful API的基本用法。

`customization_id` 参数作为HTTP的请求参数继续追加到其他请求参数之后。以Java举例如下，其他语言设置 `customization_id` 参数与Java示例相同。

```
String url = "http://nls-gateway.cn-shanghai.aliyuncs.com/stream/v1/asr";
String request = url;
request = request + "?appkey=" + appkey;
request = request + "&customization_id=" + "您的自学习模型ID";
```

实时语音识别

在实时语音识别中，需要通过设置高级参数 `customization_id` 设置自学习模型ID。

- Java SDK

> **说明**
>
> 请首先阅读Java SDK，了解Java SDK的基本用法。

由于SDK中没有 `customization_id` 参数对应的set方法，需要通过SpeechTranscriber类中的addCustomedParam方法进行设置：

```
public void addCustomedParam(String key, Object value);
```

在示例中创建SpeechTranscriber的实例对象transcriber后（调用start方法之前），调用该方法设置自学习模型ID：

```
SpeechTranscriber transcriber = new SpeechTranscriber(client, getTranscriberListener());
... // 省略其他设置。
transcriber.addCustomedParam("customization_id", "您的自学习模型ID");
```

- C++ SDK

> 说明
>
> 请首先阅读C++ SDK，了解C++ SDK的基本用法。

由于SDK中没有 customization_id 参数对应的set方法，需要通过SpeechTranscriberRequest类中的setPayloadParam方法进行设置：

```
/**
 * @brief 参数设置。
 * @param value JSON Map格式的字符串。
 * @return 成功则返回0，否则返回-1。
 */
int setPayloadParam(const char value);
```

在示例中创建SpeechTranscriberRequest的实例对象request后（调用start方法之前），调用该方法设置自学习模型ID：

```
SpeechTranscriberRequest* request = NlsClient::getInstance()->createTranscriberRequest(callback
);
... // 省略其他设置。
request->setPayloadParam("{\"customization_id\":\"您的自学习模型ID\"}");
```

- iOS SDK

> 说明
>
> 请先阅读iOS SDK，了解iOS SDK的基本用法。

通过RequestParam对象的setParams方法设置自学习模型ID。

```
NSMutableDictionary *userParams = [[NSMutableDictionary alloc]init];
[userParams setValue:@"您的自学习模型ID" forKey:@"customization_id"];
[_transRequestParam setParams:userParams];
```

- Android SDK

> **说明**
>
> 请先阅读Android SDK，了解Android SDK的基本用法。

通过SpeechTranscriber对象的setParams方法设置自学习模型ID。

```
String userParamString = "{\"customization_id\":\"您的自学习模型ID\"}";
speechTranscriber.setParams(userParamString);
```

录音文件识别

在录音文件识别中，需要通过设置高级参数 `customization_id` 设置自学习模型ID。

> **说明**
>
> 请首先阅读录音文件识别接口说明，了解录音文件识别的基本用法。

`customization_id` 参数和其他输入参数一样，以JSON格式的字符串设置到HTTP请求的Body中。

```
{
  "app_key": "yourAppkey",
  "customization_id": "您的自学习模型ID",
  "file_link": "https://aliyun-nls.oss-cn-hangzhou.aliyuncs.com/asr/fileASR/examples/nls-sample-16k.
wav"
}
```

以Java SDK为例，其他语言SDK的 `customization_id` 参数设置方法与Java SDK相同。

```
CommonRequest postRequest = new CommonRequest();
... // 省略其他设置。
/**
 * 设置文件转写请求参数，以JSON字符串的格式设置到请求体中。
 */
JSONObject taskObject = new JSONObject();
// 设置appkey。
taskObject.put(KEY_APP_KEY, appKey);
// 设置音频文件访问链接。
taskObject.put(KEY_FILE_LINK, "https://aliyun-nls.oss-cn-hangzhou.aliyuncs.com/asr/fileASR/examples/nls-sample-16k.wav");
taskObject.put("customization_id", "您的自学习模型ID");
String task = taskObject.toJSONString();
System.out.println(task);
// 设置以上JOSN字符串为Body参数。
postRequest.putBodyParameter(KEY_TASK, task);
```

3.自动化测试