微信语音SDK(iOS版)上手指南

使用说明

本文档主要介绍微信语音SDK（iOS版）使用方法，利用SDK可以直接完成从录音到网络传输、云端语音识别、结果获取等一系列动作。

本文属于入门级文档，旨在帮助开发者快速学习iOS SDK的使用并应用到自身开发工作。具体API可查询《iOS开发手册》。

获取应用授权码

请到“开发者应用登记页面”进行登记，登记并选择移动应用进行设置后，将该应用提交审核，只有审核通过的应用才能进行开发。

注册完毕，我们会在7个工作日内完成审核工作。

下载iOS SDK

进入“资源中心”下载iOS SDK，压缩包中包括Demo+SDK+开发文档。其中的Demo使用SDK的各功能API；SDK包括3个.h头文件，2个.a库文件，1个.bundle资源文件 。

Demo介绍

为了更好的理解微信语音SDK的使用，下面将通过一个简单的实例来讲解一下 SDK各个关键API接口的使用。

1. 工程配置

i. 导入 SDK 文件

WXVoiceSDK.h //语音识别

WXSpeechSynthesizer.h //语音合成

WXSpeechRecognizerWithUI.h //语音识别+UI(无 UI 使用时可不选)

libWXVoiceSDK.a //库文件(必选)

libmtaWXOsdk.a //库文件(必选)

WXResourceBundle.bundle //UI 资源(无 UI 使用时可不选)

ii. 引入系统库

AudioToolbox.framework

SystemConfiguration.framework

CoreTelephony.framework

AVFoundation.framework

libz.dylib

libsqlite3.dylib

iii. 设置 Build Settings

C++ Standard Library: libstdc++ 或 Compiler Default

Compile Sources As: Objective-C++ 或 将使用 SDK 的文件扩展名改为.mm

如果仍有 std::字样的报错，可以引用 libstdc++.6.0.9.dylib 或最新版

iv. 与其它 SDK 冲突

如果与其它含有 libmtaWXOsdk.a 文件的 SDK 冲突，则删除多余的 libmtaWXOsdk.a 文件即可。

2. 语音识别无UI调用流程

i. 初始设置

```
WXVoiceSDK * speechRecognizer= [WXVoiceSDK sharedWXVoice];
speechRecognizer.delegate = self;
speechRecognizer.silTime = 0.5f;
[speechRecognizer setAppID:@"--------------------"];
```

ii. 开始识别

```
- (BOOL)start{
        return [[WXVoiceSDK sharedWXVoice] startOnce];
}
```

或使用语法

```
- (BOOL)start{
        return [[WXVoiceSDK sharedWXVoice] startOnceWithGrammarString:_grammarTextView.text
andType:[self grammarType]];
}
```

iii. 识别到结果的回调

```
- (void)voiceInputResultArray:(NSArray *)array{
    if (array && array.count>0) {
        WXVoiceResult *result=[array objectAtIndex:0];
        [_speechRecognizerView setResultText:result.text];
    }else{
        [_speechRecognizerView setResultText:@""];
    }
}
```

iv. 出现错误的回调

```
- (void)voiceInputMakeError:(NSInteger)errorCode{
        [_speechRecognizerView setErrorCode:errorCode];
}
```

3. 语音识别无UI截图

图1

图2

4. 语法识别截图

图3

图4

5. 语音识别有UI调用流程

i. 创建 UI 对象（请不要创建多个）

```
_wxssui = [[WXSpeechRecognizerWithUI alloc] initWithDelegate:self andAppID:@"***appID***"];
```

ii. 开始识别

```
- (BOOL)clickedStartBtn{
    return [_wxssui showAndStart];
}
```

或使用语法

```
- (BOOL)clickedStartBtn{
    return [_wxssui showAndStartOnceWithGrammarString:_grammarTextView.text andType:[self grammarType]];
}
```

iii. 识别到结果的回调

```
- (void)voiceInputResultArray:(NSArray *)array{
    WXVoiceResult *result=[array objectAtIndex:0];
    [_ctrView setText:result.text];
}
```

iv. 如果允许屏幕方向改变

```
- (BOOL)shouldAutorotateToInterfaceOrientation:(UIInterfaceOrientation)toInterfaceOrientation{
    [_wxssui orientationChanged];
    return YES;
}
- (BOOL)shouldAutorotate{
    return YES;
}
- (NSUInteger)supportedInterfaceOrientations{
    [_wxssui orientationChanged];
    return UIInterfaceOrientationMaskAll;
}
```

6. 语音识别有UI截图

图5

图6

7. 语音合成SDK调用流程

i. 初始设置

```
WXSpeechSynthesizer * speechSynthesizer = [WXSpeechSynthesizer sharedSpeechSynthesizer];
[speechSynthesizer setDelegate:self];
[speechSynthesizer setAppID:@"***appID***"];
[speechSynthesizer setVolumn:1.0];
```

ii. 开始合成

```
- (BOOL)startWithText:(NSString *)text{
        return [[WXSpeechSynthesizer sharedSpeechSynthesizer] startWithText:text];
}
```

iii. 得到合成语音数据的回调

```
- (void)speechSynthesizerResultSpeechData:(NSData *)speechData speechFormat:(int)speechFormat{
        //amr格式无法直接播放，请自行转码
        [_player playNewData:speechData];
        if (_isPause) {
                [_player pause];
        }
}
```

iv. 出现错误的回调

```
- (void)speechSynthesizerMakeError:(NSInteger)error{
        [_ssView reSetView];
[WXErrorMsg showErrorMsg:[NSString stringWithFormat:@"错误码：%d",error] onView:self.view];
}
```

8. 语音合成界面截图

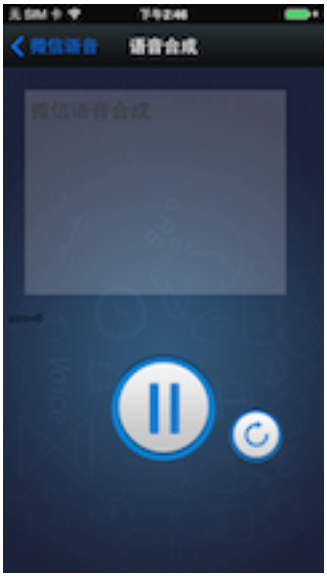

图7

更多内容请参考《iOS开发手册》和WXVoiceSDKDemo，WXVoiceSDKWithUIDemo。

无 UI 的使用规范

微信语音开放平台免费为你的应用提供语音识别服务，你可以根据自己的风格自由制定UI，但需在语音采集识别的窗口正确、完整的标注“**Powered by 微信智能**”或“**语音技术由微信智能提供**”的字样。参考如下弹窗：

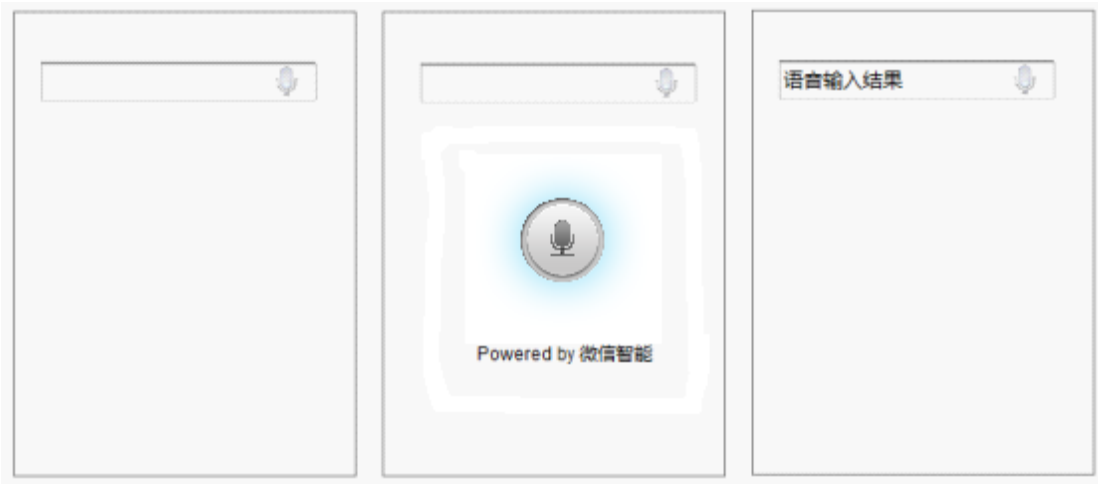

图 8

附录

ABNF 语法手册

语音识别语法用来告知语音识别器（Recognizer）所要识别的语句形式。它包括：

- 用户将要说的词汇（word）
- 将要出现的词汇的模式（pattern）
- 每个词汇的语言

语法文档被编译成识别网络后，将被送往语音识别器。语音识别器提取输入语音的特征信息并在识别网络上进行路径匹配，最终识别出用户说话的内容。因此语法是语音识别系统的输入之一，它是现阶段语音识别得以应用的必要条件。

```
&DEQI 4JB XWI-; >

odqj xdj h}k-fq>

p r gh yr lfh>

ur r w' edvlfFp g>

sxedf ' edvlfFp g @查询`' doqdp hv路况>

' doqdp hv @-成乐  京昆  广深,高速`>
```

例如，开发一个高速路况查询的简单语音识别系统，可以定义如下的语法：

目前，语音识别语法可以采用的是 ABNF 形式，ABNF 形式简练，便于用户理解和书写。

ABNF 文档包括两个部分，文档首部（header）和主体(body)。文档首部定义了文档的各种属性，而文档的主体则具体定义了用户说话的内容和模式。文档首部包括：

- ABNF 文档自标识头
- 语言
- 模式
- 根规则

注意：文档首部必须出现在文档的开头部分，也就是说一旦出现了第一个规则定义，即宣告文档首部的结束，出现在文档主体中的文档首部声明，作为文档主体（规则扩展）看待，通常会引起编译器报错。用户务必注意这一点。

1. ABNF 文档自标识头（Self-Identifying Header，强制）

ABNF 文档必须在其第一行包含一个形式如下的自标识头：

#ABNF *VersionNumber CharEncoding ;*

文档自标识头定义了文档的版本和编码格式。其中：

VersionNumber 指定语法文档版本号，目前版本号必须是1.0。

CharEncoding 指定语法文档字符编码类型，其默认值是 GB2312。

开发语法时，请确保文档自标识头声明的字符编码类型和文件的真实字符编码类型是统一的。

例如，如果声明的文档编码格式是 UTF-8的编码格式，文档的内容必须采用这一格式。因为语法编译系统依靠这一声明的字符编码，把文档转换成统一的 Unicode 格式。因此，如果声明的字符编码和文件真实的字符编码不一致，那么转换会出错。

讯飞语音识别系统的语法编译子系统可以支持 ISO-8859-1、GB2312、GBK、UTF-8、UTF-16LE、UTF-16BE 等多种格式的文本编码方式。

推荐中国大陆用户使用 GB2312编码类型，并在语法中预先声明它。

```
&DEQI 4B J E5645>
```

2. 语言声明（Language，可选）

在进行中文识别时，推荐使用的语言声明为 zh-cn。ABNF 语言声明具有以下形式：

```
language zh-cn;
```

3. 模式（Mode，强制）

目前，语法引擎仅支持 voice 模式。

```
p r gh yr lf h>
```

整个语法主体的规则扩展展开是一棵或多棵语法树，在应用中推荐为语法指定一个根规则，根规则可以看作是整个语法树的根，同时根规则也定义了外部引用该语法的默认引用规则。外部引用的根规则必须定义成 public 的。注意一个语法文档能且只能定义一个根规则：

ABNF 根规则声明具有如下形式：

```
ur r w' edvlf Fp g>
```

语法文档中可以使用注释。ABNF 形式的注释如下：

```
// C++/Java-style single-line comment
/* C/C++/Java-style comment */
/** Java-style documentation comment */
```

语音识别语法是通过规则定义（Rule Definition）和规则引用（Rule Reference）来组成语法主体（Body）的。规则引用的各种组合通称为规则扩展（Rule Expansion）。规则扩展是一个正则表达式。一个规则定义用“=”把规则名称和规则内容联系起来，规则名称具有“$+字符串的形式”，而规则内容就是所谓的规则扩展。规则扩展的最基本的结构是顺序、选择和循环。

```
^vf r s h`' ux dhQdp h @ ux dhH( s dqvlr q>
```

在高速公路路况查询的语法中：

```
s xe df ' edvlf Fp g @ 查询`' doqdp hv 路况>

' doqdp hv @

+成乐  京昆  广深 , 高速`>
```

从而造成编译错误。一个规则定义可以连续引用多个规则名，记号名以及它们的各种组合：

包括顺序结构、选择结构、重复结构、可选结构。

7. 顺序结构

一个规则定义可以连续引用多个规则名，记号名以及它们的各种组合。序列相当于程序设计中的顺序结构。例如 ABNF 形式：

这是一个测试用例 // token 的序列

$action $object //规则引用的序列

(查询 $allnames 路况) //用括号来封装

8. 选择结构

在规则扩展中选择结构表示说话时只可能覆盖其中的一条路径，它相当于程序设计语言中的选择结构。选择结构的在 ABNF 中用“|”来分隔多个选择分支：

```
' dqqqdp hv @成乐  京昆  广深 >
```

重复结构用来在语法中表示需要重复说出的内容，它特别适合表示诸如数字串识别等有一些简单词语反复出现构成的语法结构。

ABNF 形式的重复结构可以由在被引用语法结构（规则、记号或它们的任意组合）后加上重复标签<min – max>来设定。min 表示最小重复次数，max 表示最大重复次数。

ABNF 形式的重复结构表示的一个例子如下：

```
' uxdhqdp h @' glj lw? 5-; A>

' glj lw@4  5  6  7  8  9  :  ;  <  3>
```

10. 可选结构：

在高速公路路况查询的语法中，查询一词可出现，也可不出现，用[]表示其可选性：

```
sxedof ' edvlfFp g @'查询` ' dqqqdp hv 路况>
```

在语法开发的过程中，用户还必须注意下面这些问题：

- 不要出现规则的递归定义，及一个规则的定义直接或者间接地引用自己。实用语法不需要递归结构。
- 开发的过程中考虑准确性和性能的统一。例如：稍微长一些的词语有利于提高识别率；能够合并的部分尽可能合并已减少语法网络的大小，这样可以有效地提高识别系统的性能。