# 第3章 文本相似度计算

文本相似度计算是自然语言处理的基本任务。文本相似度一般是指文本在语义上的相似程度，可以广泛应用于自然语言处理任务的多个领域。如在机器翻译领域可以用于评价翻译结果的准确程度，在搜索引擎领域可以用于衡量检索文本与被检索文本之间的相关程度，在文本分类领域可以用于评价文本间内容的相似程度，在问答领域可以用于评定用户输入问题与问答库中问题的相似程度及问题与答案的相关程度等。本章主要介绍常见的文本相似度算法及用于比较文本相似度的模型。

本章主要涉及到的知识点有：

* 文本相似度计算介绍。
* 基于字符的文本相似度算法。
* 基于语义的文本相似度算法。
* 基于深度学习的文本相似度计算。
* 相似度匹配模型介绍及实现。

注意：本章代码地址：xxx

3.5 相似度比较模型实现

3.5.1 pairwise模型

3.5.2 pointwise模型

## 3.1 文本相似度计算介绍

在信息爆炸时代，如何从海量信息中获取需要的信息成为一种亟待解决的问题，为了解决这个问题而出现的搜索引擎、推荐系统、问答系统、文本分类、信息检索等。这些应用场景的关键技术就是文本相似度计算。

文本相似度在不同的场景中有不同的含义，狭义的文本相似即为字符串的相似程度，一般认为两个文本所包含的字符及字符间的排列顺序越相似，则文本的相似度越高。随着自然语言处理技术的发展，当前的文本相似也包括文本所描述的语义相似的情况。

文本相似度计算方法的不同本质是文本表示方法的不同。我们将文本相似度算法一般分为两类，一类为基于字符（charter-base-model）即直接基于将待比较文本的字符串信息进行计算。另一类为基于向量空间（VSM vector-space-model）即将待比较的文本转换为向量表示，然后在向量空间内进行比较。

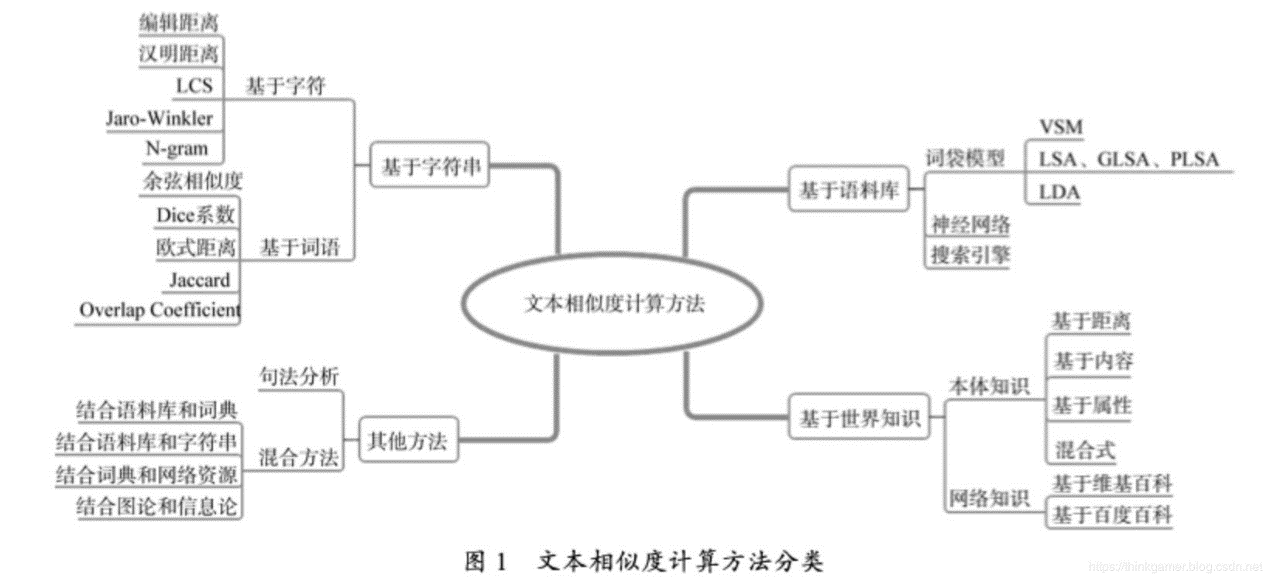


图2.1 Siri交互页面

## 3.2 基于字符的文本相似度计算

本节主要介绍我们接下来开发所依赖的开发环境及开发工具的安装与配置。本书中代码全部采用Python语言开发，使用的开发工具为PyCharm。接下来主要介绍这两部分的安装与配置。

### 2.2.1 Python环境安装

首先在官网<https://www.python.org/downloads/>下载Python的安装包。

本文中的操作系统为Windows 10，选用的Python版本为Python3.7.6，为了避免项目运行中出现版本冲突的情况，推荐使用Python3.7及以上版本。

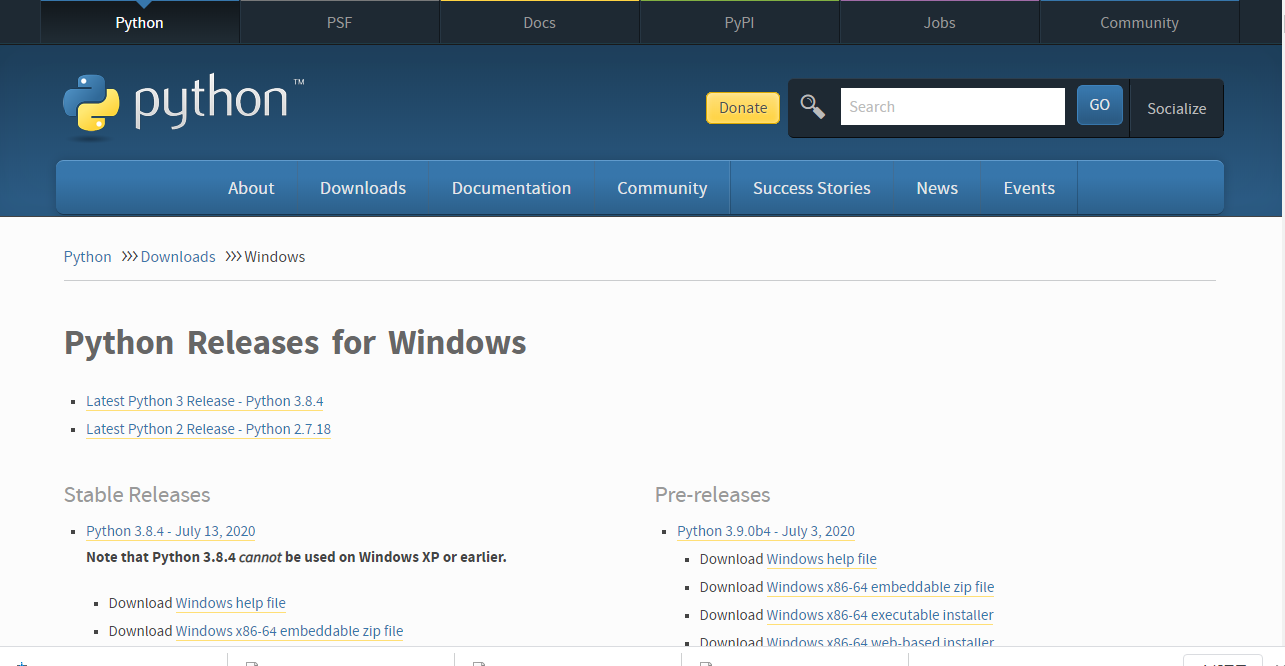


图2.4 Python官网

在该页面选择相应的安装包，下载后双击安装。

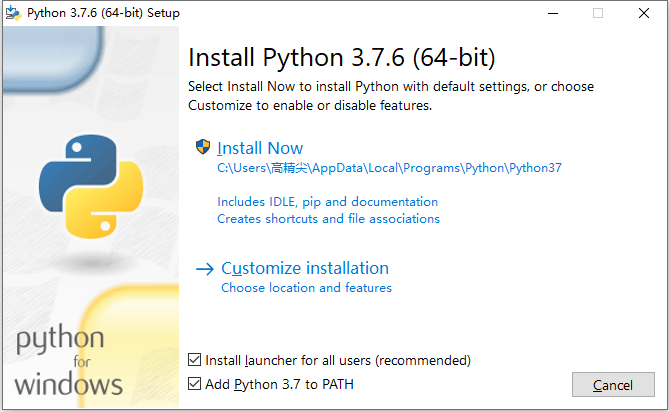


图2.5 Python安装界面

安装Python过程中，勾选“Add Python 3.7 to PATH”将Python的安装位置添加到系统环境变量PATH中，安装完成后可以通过命令提示符打开。同时安装过程可以选用“Customize installation”自定义安装方式，将Python安装到指定路径下。

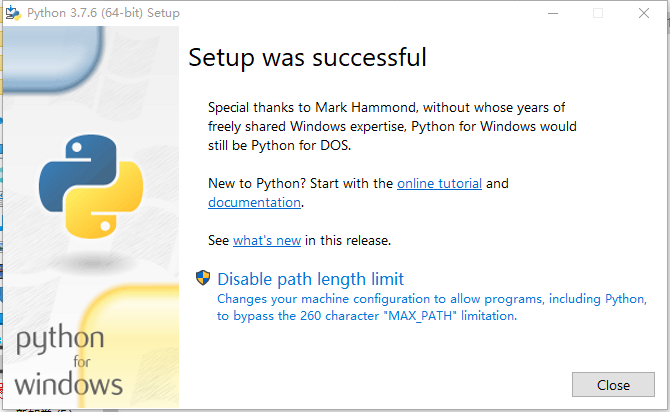


图2.6 Python安装成功

Python安装成功后，可以通过命令提示符进行测试。打开命令提示符，输入“python”命令，可以看到我们安装的python版本为3.7.6，以下为测试效果。

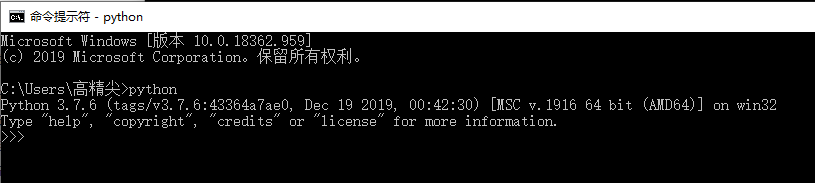


图2.7 Python安装测试效果

### 2.2.2 PyCharm安装配置

为了提高我们的编程效率，我们选用JetBrains的Python IDE—PyCharm作为开发工具。PyCharm带有一整套可以帮助用户使用Python语言开发时可以提高开发人员效率的工具，如调试、语法高亮、Project管理、代码跳转、智能提示、自动完成、单元测试、版本控制等。

首先在官网下载安装包。官网地址为：

<https://www.jetbrains.com/zh-cn/pycharm/download/#section=windows>

这里选用社区开源版，下载PyCharm2020.1.3版本。下载完成后双击安装。

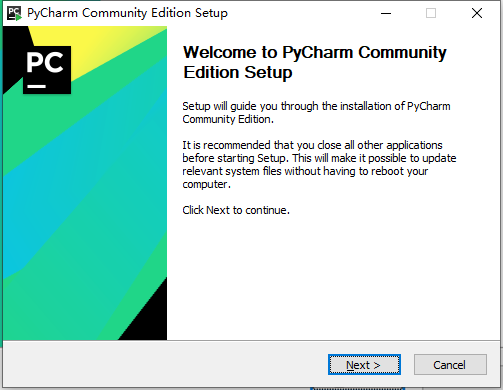


图2.8 PyCharm安装

安装过程中勾选相应信息。

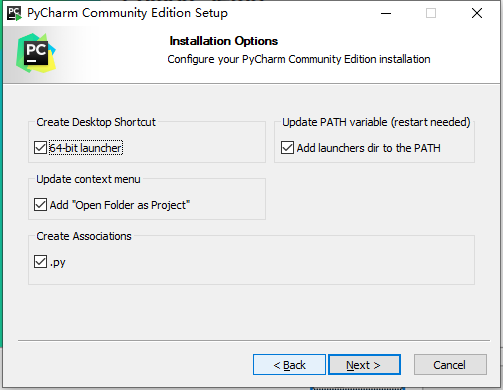


图2.9 PyCharm 配置安装信息

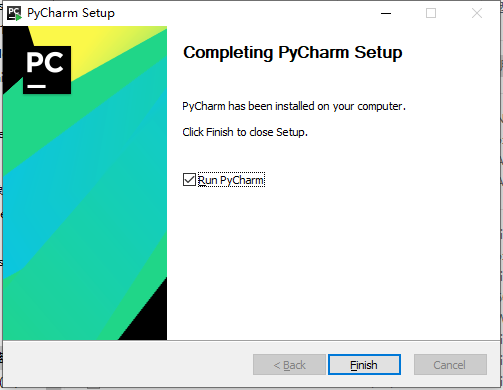


图2.10 PyCharm安装成功

PyCharm安装成功后，点击进入，新建一个Python项目。

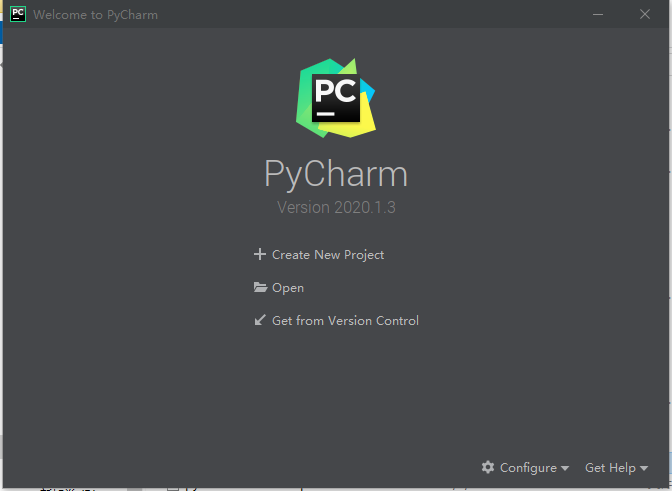


图2.11 PyCharm新建项目

新建项目后，配置Python解析器，配置成功后，即可进行项目实际开发。

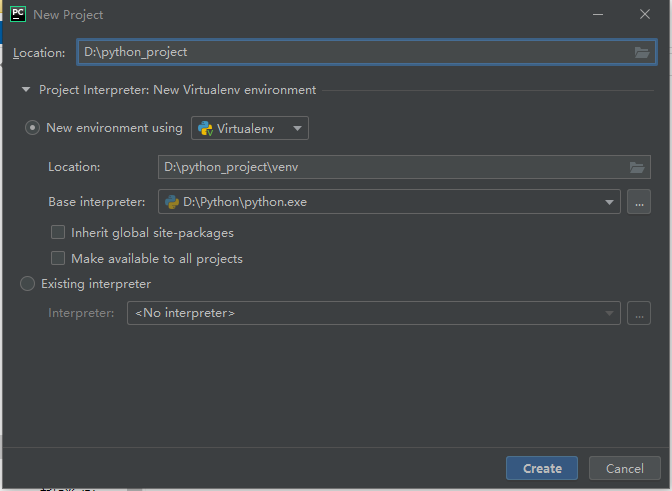


图2.12 PyCharm配置Python解析器

## 2.3 语音转文字模块实现

完成开发环境的配置后，接下来我们正式进入项目开发阶段。

本模块主要完成将用户的语言输入转换为文字表示。本项目采用百度语音服务将智能设备采集到的用户输入语音转换为对话系统可以处理的字符表示。

首先需要注册百度账号并登录，登录后进入开发者认证页面，填写相关信息完成开发者认证。进入百度AI云平台智能语音服务。

<https://ai.baidu.com/tech/speech/asr>



点击创建应用，填写相应信息，点击立即创建。





在创建完毕应用后，平台将会分配给此应用的相关凭证，主要为AppID、API Key、Secret Key。以上三个信息是我们实际开发应用中的主要凭证。

获取到项目中所需要的信息后，我们通过以下代码将获取到的用户输入语音转换为文字表示，具体代码如下：

**代码2.1 将语音转为文字**

#!/usr/bin/env python  
# \_\*\_ coding:utf-8 \_\*\_  
import sys  
import json  
import base64  
import time  
from urllib.request import urlopen  
from urllib.request import Request  
from urllib.error import URLError  
from urllib.parse import urlencode  
  
timer = time.perf\_counter  
  
# 从应用中获取的信息  
API\_KEY = ' Your API\_KEY '  
SECRET\_KEY = ' Your SECRET\_KEY '  
  
# 有此scope表示有asr能力，没有请在网页里勾选，非常旧的应用可能没有  
SCOPE = 'audio\_voice\_assistant\_get'  
  
  
class DemoError(Exception):  
 pass  
  
  
""" TOKEN start """  
TOKEN\_URL = 'http://openapi.baidu.com/oauth/2.0/token'  
  
  
def fetch\_token():  
 *'''  
 获取token* ***:return****:  
 '''* # 设置获取token的参数  
 params = {'grant\_type': 'client\_credentials',  
 'client\_id': API\_KEY,  
 'client\_secret': SECRET\_KEY}  
 # 通过post方式传递参数  
 post\_data = urlencode(params)  
 post\_data = post\_data.encode('utf-8')  
 req = Request(TOKEN\_URL, post\_data)  
 try:  
 f = urlopen(req)  
 result\_str = f.read()  
 except URLError as err:  
 print('token http response http code : ' + str(err.code))  
 result\_str = err.read()  
 result\_str = result\_str.decode()  
  
 # 获取token结果  
 result = json.loads(result\_str)  
 # 校验token结果是否正确  
 if ('access\_token' in result.keys() and 'scope' in result.keys()):  
 print(SCOPE)  
 if SCOPE and (not SCOPE in result['scope'].split(' ')): # SCOPE = False 忽略检查  
 raise DemoError('scope is not correct')  
 print('SUCCESS WITH TOKEN: %s EXPIRES IN SECONDS: %s' % (result['access\_token'], result['expires\_in']))  
 return result['access\_token']  
 else:  
 raise DemoError('MAYBE API\_KEY or SECRET\_KEY not correct: access\_token or scope not found in token response')  
  
  
""" TOKEN end """  
  
  
def asr(AUDIO\_FILE):  
 # 以下为参数设置  
 # 文件格式  
 FORMAT = AUDIO\_FILE[-3:]  
  
 CUID = '123456PYTHON'  
  
 # 采样率  
 RATE = 16000 # 固定值  
  
 # 1537 表示识别普通话，使用输入法模型。根据文档填写PID，选择语言及识别模型  
 DEV\_PID = 1537  
  
 # asr服务地址信息  
 ASR\_URL = 'http://vop.baidu.com/server\_api'  
  
 # 获取token  
 token = fetch\_token()  
 # 获取要识别的音频文件  
 speech\_data = []  
 with open(AUDIO\_FILE, 'rb') as speech\_file:  
 speech\_data = speech\_file.read()  
 # 若文件内容为空则抛出异常  
 length = len(speech\_data)  
 if length == 0:  
 raise DemoError('file %s length read 0 bytes' % AUDIO\_FILE)  
 # 使用base64加密编码  
 speech = base64.b64encode(speech\_data)  
 speech = str(speech, 'utf-8')  
 # 设置参数  
 params = {'dev\_pid': DEV\_PID,  
 'format': FORMAT,  
 'rate': RATE,  
 'token': token,  
 'cuid': CUID,  
 'channel': 1,  
 'speech': speech,  
 'len': length  
 }  
 # 设置请求格式  
 post\_data = json.dumps(params, sort\_keys=False)  
 req = Request(ASR\_URL, post\_data.encode('utf-8'))  
 req.add\_header('Content-Type', 'application/json')  
 try:  
 begin = timer()  
 f = urlopen(req)  
 result\_json = f.read()  
 # 计算服务响应实践  
 print("Request time cost %f" % (timer() - begin))  
 except URLError as err:  
 print('asr http response http code : ' + str(err.code))  
 result\_json = err.read()  
 # 获取语音识别结果  
 result\_str = str(result\_json, 'utf-8')  
 print(result\_str)  
 # 保存语音识别结果  
 with open("result.txt", "w") as of:  
 of.write(result\_str)  
 result\_data = json.loads(result\_json)['result']  
 return result\_data  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 # 待识别的音频文件  
 AUDIO\_FILE = './audio/16k.pcm' # 支持 pcm/wav/amr 格式  
 res = asr(AUDIO\_FILE)  
 print(res)

通过以上代码我们将./audio/16k.pcm音频文件中的语音内容进行识别返回结果如下：

{"corpus\_no":"6851191777346904654","err\_msg":"success.","err\_no":0,"result":["北京科技馆。"],"sn":"801724218701595167391"}

## 2.4 文字转语音模块

本项目采用百度语音服务将对话系统返回的文字内容转换为可以与用户直接交互的音频格式，通过智能设备的音频播放功能进行播放。

同上面的过程一样，我们新建一个语音合成的应用，获取系统生成的AppID、API Key、Secret Key信息。获取到项目中所需要的信息后，我们通过以下代码将文字内容转换为音频格式，具体代码如下：

**代码2.2 将文字转为语音**

#!/usr/bin/env python  
# \_\*\_ coding:utf-8 \_\*\_  
import sys  
import json  
from urllib.request import urlopen  
from urllib.request import Request  
from urllib.error import URLError  
from urllib.parse import urlencode  
from urllib.parse import quote\_plus  
  
  
# 从应用中获取的信息  
API\_KEY = ' Your API\_KEY '  
SECRET\_KEY = ' Your SECRET\_KEY '  
  
  
  
# 参数设置  
# 发音人选择, 基础音库：0为度小美，1为度小宇，3为度逍遥，4为度丫丫，  
# 精品音库：5为度小娇，103为度米朵，106为度博文，110为度小童，111为度小萌，默认为度小美  
PER = 0  
  
# 语速，取值0-15，默认为5中语速  
SPD = 5  
  
# 音调，取值0-15，默认为5中语调  
PIT = 5  
  
# 音量，取值0-9，默认为5中音量  
VOL = 5  
  
# 下载的文件格式, 3：mp3(default) 4： pcm-16k 5： pcm-8k 6. wav  
AUE = 3  
  
FORMATS = {3: "mp3", 4: "pcm", 5: "pcm", 6: "wav"}  
FORMAT = FORMATS[AUE]  
  
CUID = "123456PYTHON"  
  
TTS\_URL = 'http://tsn.baidu.com/text2audio'  
  
class DemoError(Exception):  
 print('error')  
  
  
""" TOKEN start """  
  
TOKEN\_URL = 'http://openapi.baidu.com/oauth/2.0/token'  
  
# 有此scope表示有tts能力，没有请在网页里勾选  
SCOPE = 'audio\_tts\_post'  
  
  
def fetch\_token():  
 *'''  
 获取token* ***:return****:  
 '''* print("fetch token begin")  
 # 设置token参数信息  
 params = {'grant\_type': 'client\_credentials',  
 'client\_id': API\_KEY,  
 'client\_secret': SECRET\_KEY}  
 post\_data = urlencode(params)  
 post\_data = post\_data.encode('utf-8')  
 # 发送请求获取token信息  
 req = Request(TOKEN\_URL, post\_data)  
 try:  
 f = urlopen(req, timeout=5)  
 result\_str = f.read()  
 except URLError as err:  
 print('token http response http code : ' + str(err.code))  
 result\_str = err.read()  
 result\_str = result\_str.decode()  
  
 print(result\_str)  
 result = json.loads(result\_str)  
 print(result)  
 if ('access\_token' in result.keys() and 'scope' in result.keys()):  
 if not SCOPE in result['scope'].split(' '):  
 raise DemoError('scope is not correct')  
 print('SUCCESS WITH TOKEN: %s ; EXPIRES IN SECONDS: %s' % (result['access\_token'], result['expires\_in']))  
 return result['access\_token']  
 else:  
 raise DemoError('MAYBE API\_KEY or SECRET\_KEY not correct: access\_token or scope not found in token response')  
  
  
""" TOKEN end """  
  
  
def tts(TEXT):  
 # 获取token  
 token = fetch\_token()  
 # 此处TEXT需要两次urlencode  
 tex = quote\_plus(TEXT)  
 print(tex)  
 # lan ctp 固定参数  
 params = {'tok': token, 'tex': tex, 'per': PER, 'spd': SPD, 'pit': PIT, 'vol': VOL, 'aue': AUE, 'cuid': CUID,  
 'lan': 'zh', 'ctp': 1}  
 # 对参数进行编码  
 data = urlencode(params)  
 print('test on Web Browser' + TTS\_URL + '?' + data)  
 # 获取请求返回结果  
 req = Request(TTS\_URL, data.encode('utf-8'))  
 has\_error = False  
 try:  
 f = urlopen(req)  
 result\_str = f.read()  
 # 获取返回结果的headers信息  
 headers = dict((name.lower(), value) for name, value in f.headers.items())  
 # 判定返回结果是否正确  
 has\_error = ('content-type' not in headers.keys() or headers['content-type'].find('audio/') < 0)  
 except URLError as err:  
 print('asr http response http code : ' + str(err.code))  
 result\_str = err.read()  
 has\_error = True  
 # 保存返回结果为音频格式  
 save\_file = "error.txt" if has\_error else 'result.' + FORMAT  
 with open(save\_file, 'wb') as of:  
 of.write(result\_str)  
  
 if has\_error:  
 result\_str = str(result\_str, 'utf-8')  
 print("tts api error:" + result\_str)  
  
 print("result saved as :" + save\_file)  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 # 待转换的文本信息  
 TEXT = "您好，有什么可以帮助您的吗？"  
 tts(TEXT)

## 2.5 对话模块

本项目中为了实现一个简单的语音助手，我们通过调用图灵机器人API来完成，可以帮助我们方便快捷的搭建一个与系统进行交互的对话模块。

首先需要注册登陆图灵官网账号。创建机器人获取apikey信息。

**代码2.3 对话模块**

#!/usr/bin/env python  
# \_\*\_ coding:utf-8 \_\*\_  
import json  
import requests  
  
API\_KEY = 'Your API\_KEY '  
  
  
def getTulingResponse(msg):  
 *'''  
 对话模块* ***:param*** *msg: 用户输入信息* ***:return****:  
 '''* # api 地址信息  
 api = 'http://openapi.tuling123.com/openapi/api/v2'  
 dat = {  
 "perception": {  
 "inputText": {  
 "text": msg  
 },  
 "inputImage": {  
 "url": "imageUrl"  
 },  
 "selfInfo": {  
 "location": {  
 "city": "北京",  
 "province": "北京",  
 "street": ""  
 }  
 }  
 },  
 "userInfo": {  
 "apiKey": API\_KEY,  
 "userId": '136772'  
 }  
 }  
 dat = json.dumps(dat)  
 # 发送对话请求  
 r = requests.post(api, data=dat).json()  
 # 对话返回信息  
 mesage = r['results'][0]['values']['text']  
 print('Sys: ', r['results'][0]['values']['text'])  
 return mesage  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
  
 flag = True  
 while flag:  
 # 获取用户输入  
 user\_input = input('User: ')  
 # 设置对话结束条件  
 if user\_input == 'bye':  
 flag = False  
 else:  
 # 对话系统回复信息  
 sys\_reply = getTulingResponse(user\_input)

效果展示如下，其中：

User: 你好

Sys: 我很好，你也要好好的

User: 今天天气怎么样

Sys: 不用问了，反正不是天晴就是下雨，不是下雨就是阴天，总之肯定不会下钱~

User: 你几岁了

Sys: 9岁了，我也是度过了无数个春秋的机器人了。

User: bye

## 2.6 语音助手实现

通过调用上面的三个模块，对于用户输入的语音信息，经过智能设备采集到的音频格式文件，通过语音转文字模块转换为文字表示，将用户输入传递给对话模块，获取到系统的回复，将回复的内容通过文字转语音模块转换为音频格式的文件。

具体代码如下：

#!/usr/bin/env python  
# \_\*\_ coding:utf-8 \_\*\_  
from asr.asr\_demo import asr  
from dialogue\_demo.dialogue\_demo import getTulingResponse  
from tts.tts\_demo import tts  
  
  
def main():  
 *'''  
 语音助手实现代码* ***:return****:  
 '''* # 获取用户输入的语音信息  
 user\_input = './asr/audio/16k.pcm'  
 # 将语音信息转为文字表示  
 user\_text = asr(user\_input)  
 print(user\_text)  
 # 获取用户输入信息的对话反馈信息  
 sys\_reply = getTulingResponse(user\_text)  
 print(sys\_reply)  
 # 将系统反馈信息转换为语音格式  
 tts(sys\_reply)  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()

系统实际效果如下：

用户输入语音信息为： 北京科技馆。

系统回复: 具体地址在哪里？