# 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。

在社区问答网站中，往往会出现越来越多的相似重复问题，文本相似度计算可以帮助检测这些重复问题，一方面降低数据冗余，减少数据存储维护的成本；另一方面减少重复问题可以帮助用户快速检索到想要获取的信息，提升用户体验。

文本相似度计算方法一般可以分为两类，一种是基于词频的方法，通常是一些较为传统的方法。另一种是基于语义的方法，通常是基于机器学习及深度学习的方法。

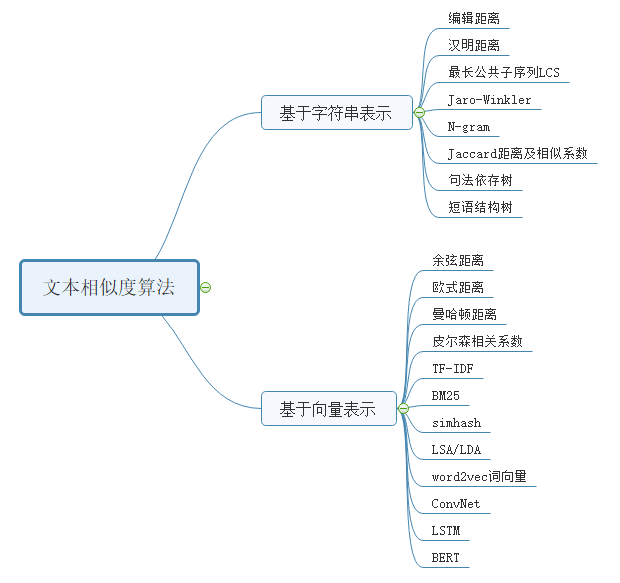


图3.1 文本相似度计算方法分类

## 3.2 基于词频的文本相似度计算

本节主要介绍基于词频的文本相似度计算方法及其优缺点。给定两个句子与，若两个句子中包含的字符种类及字符间的排列顺便均相同，那么可以认为两个句子相同，若两个句子中字符不同或者字符排列顺序不同，那么如何衡量两个句子之间的相似情况？

在机器学习出现之前，传统的文本相似度是基于句子中的字符信息及词频信息进行计算的。如基于字符变换的距离来计算相似度的编辑距离、汉明距离；基于信息检索中的TF-IDF、BM25、语言模型等方法，主要解决字面相似度问题。这些方法由于计算简单，适用范围广，到现在依旧是很多场景下的优秀基准模型。

### 3.2.1 编辑距离

编辑距离，又称为Levenshtein距离是指由原字符串转换为另外一个字符串所需要的最小编辑次数或最小代价，编辑即是对字符串指定位置的单个字符进行插入、删除和替换的操作。

考虑长度分别为m和n的两个字符串与，构造矩阵，利用动态规划方法获得该矩阵元素即

在该矩阵中，矩阵右下角元素LD(n,m)即为字符串与之间的编辑距离。

如将字符串“abc”转换为“ab”，只需要将字符串“abc”的“c”删除即可，因此这两个字符串之间的编辑距离为1；同理，将字符串“abd”转换为“abc”，只需要将字符“d”替换为“c”，故编辑距离为1；

**代码3.1 计算字符串间的编辑距离**

#!/usr/bin/env python  
# \_\*\_ coding:utf-8 \_\*\_  
import numpy as np  
  
  
def edit\_distance(str1, str2):  
 *'''  
 利用动态规划计算编辑距离* ***:param*** *str1:* ***:param*** *str2:* ***:return****:   
 '''* len1 = len(str1)  
 len2 = len(str2)  
 dp = np.zeros((len1 + 1, len2 + 1))  
 for i in range(len1 + 1):  
 dp[i][0] = i  
 for j in range(len2 + 1):  
 dp[0][j] = j  
 # 利用计算编辑距离的公式来计算  
 for i in range(1, len1 + 1):  
 for j in range(1, len2 + 1):  
 if str1[i - 1] == str2[j - 1]:  
 delta = 0  
 else:  
 delta = 1  
 dp[i][j] = min(dp[i - 1][j - 1] + delta, min(dp[i - 1][j] + 1, dp[i][j - 1] + 1))  
 return dp[len1][len2]  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 str1 = 'hello'  
 str2 = 'world'  
 # 计算str1与str2之间的编辑距离  
 ed = edit\_distance(str1, str2)  
 print(ed)

### 3.2.2 汉明距离

与编辑距离算法相近的是汉明距离，汉明距离主要应用于编码通讯领域，在误差检测和校正码处理上有很好的效果。对于两个等长字符串之间的汉明距离是两个字符串对应位置的不同字符的个数。

汉明距离的核心原理就是如何通过字符替换（最初应用在通讯中实际上是二进制的0-1替换），能将一个字符串替换成另外一个字符串。

如“karolin” 和 “kathrin” 的汉明距离为(字符2 3 4替换)

“karolin” 和 “kerstin” 的汉明距离为(字符1 3 4替换)

**代码3.2 计算字符串间的汉明距离**

#!/usr/bin/env python

# \_\*\_ coding:utf-8 \_\*\_

def hamming\_distance(str1, str2):

'''

Return the Hamming distance between equal-length sequences

:param str1:

:param str2:

:return:

'''

if len(str1) != len(str2):

raise ValueError("Undefined for sequences of unequal length")

return sum(el1 != el2 for el1, el2 in zip(str1, str2))

if \_\_name\_\_ == "\_\_main\_\_":

str1 = 'abc'

str2 = 'abd'

hd = hamming\_distance(str1, str2)

print(hd)

### 3.2.3 TF-IDF

TF-IDF（term frequency–inverse document frequency ）是一种用于资讯检索与文本挖掘的常用加权技术。TF-IDF是一种统计方法，用以评估一个字词对于一个文档集或一个语料库中的其中一份文件的重要程度。字词的重要性随着它在文件中出现的次数成正比增加，但同时会随着它在语料库中出现的频率成反比下降。

TF：在一份给定的文件里，词频（term frequency，TF）指的是某一个给定的词语在该文件中出现的次数。对于在某一特定文件里的词语ti来说，它的TF可表示为：

TF = 某个词在文档中的出现次数/文档中的总词数

​ IDF ：逆向文件频率（inverse document frequency，IDF）是一个词语普遍重要性的度量。某一特定词语的IDF，可以由总文件数目除以包含该词语之文件的数目，再将得到的商取对数得到：

IDF = log(语料库中的总文档数/语料库中出现该词的文档数)

最终，TF-IDF=TF \* IDF

### 3.2.4 BM25

BM25算法是一种应用广泛的对TF-IDF的改进算法，解决了TF-IDF偏向于长文档的问题。传统的TF值理论上是可以无限大的。而BM25与之不同，它在TF计算方法中增加了一个常量k，用来限制TF值的增长极限。

BM25还引入了平均文档长度的概念，单个文档长度对相关性的影响力与它和平均长度的比值有关系引入另外两个参数：L和b。L是文档长度与平均长度的比值。如果文档长度是平均长度的2倍，则L＝2。b是一个常数，它的作用是规定L对评分的影响有多大。加了L和b的TF计算公式变为:

TF = ((k + 1) \* tf) / (k \* (1.0 - b + b \* L) + tf)

IDF部分计算方法与TF-IDF中相同。 最终，BM25=TF \* IDF

### 3.2.5 BM25统计语言模型

统计语言模型用于计算给定一个问题，另一个问题由其生成的概率。通过引入马尔可夫假设，我们可以认为一句话中每个单词出现的概率只与它前面n个词有关，整句的概率就是各个词出现概率的乘积。该模型被称为ngram语言模型。

统计语言模型通常对语料库的大小有着较强的要求，通常来说，随着n-gram模型中n的增加，模型对于概率的估计会更加准确，但是需要的数据量也会成大大增加，所以，常用的统计语言模型通常为2-gram模型或者one-gram模型。

## 3.3 基于语义的文本相似度计算

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？