**北京信息科技大学**

**科研开发类项目实践(2)**

**结题报告**

**题目：基于知识图谱的语义搜索方法研究**

**学院：\_\_\_\_\_\_\_ \_\_\_计算机学院 \_\_\_ \_\_\_**

**专业：\_\_\_\_\_\_ \_\_计算机科学与技术\_\_\_\_ \_ \_\_**

**学生姓名：王亚可 班级/学号：计科1906/2019011411**

**学生姓名：王子健 班级/学号：计科1903/2019011415**

**学生姓名：薛冠文 班级/学号：计科1901/2019011416**

**指导老师：\_\_\_\_\_\_\_\_ \_\_王楠 \_\_\_\_\_\_ \_ \_\_\_\_\_**

**学年/学期：\_\_\_\_\_\_ 2021 - 2022（1）\_\_\_\_\_\_\_\_\_\_**

目录

[一、概述 2](#_Toc91512388)

[二、数据工程 2](#_Toc91512389)

[三、文本相似度 12](#_Toc91512390)

[四、知识工程 17](#_Toc91512391)

[五、文献搜索 22](#_Toc91512392)

摘要：

目前的文献搜索网站普遍使用字符匹配的方式进行匹配，没有针对文献的内容进行处理。对此，本文尝试对文本内容处理，针对抽取的知识进行匹配搜索。难点包括对科技文献信息的识别提取，摘要的知识抽取，数据库的使用等。本文基于科技文献对行业常用的语义搜索方法进行了一定的研究。实现了数据爬取、文献内容抽取、OCR识别等数据工程。使用Elasticsearch保存抽取的文献信息，使用arangoDB存取三元组信息。计算余弦相似度，并且将功能包装为API，部署在服务端可供用户调用。当然，本文对于文献的知识抽取的实现过程还不够成熟，数据库文献数量不够多，后续期望后续能够实现一个可用的搜索平台。

## 一、概述

**1、基本概述**

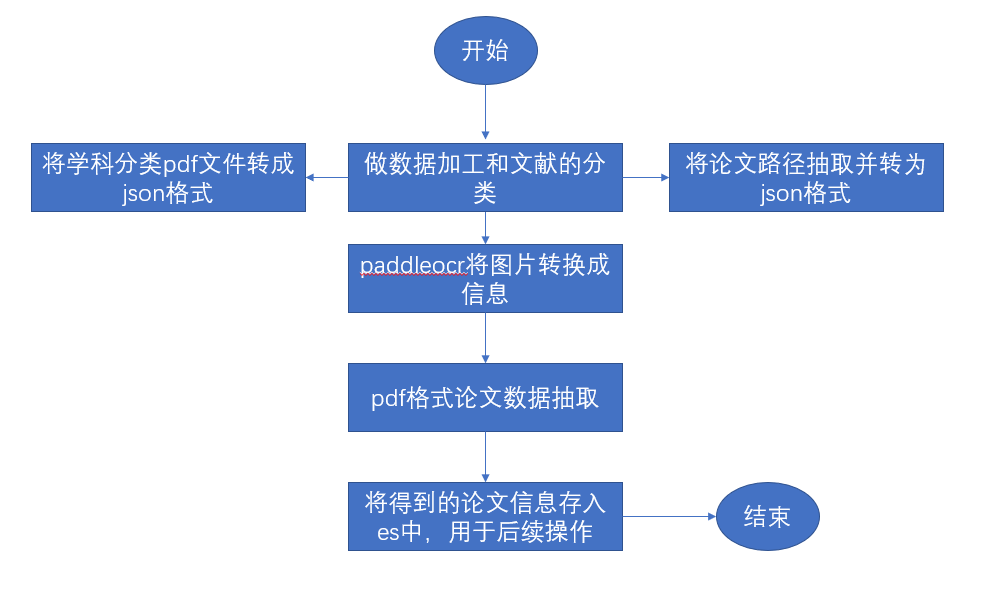
在大数据时代，数据和信息处理成为众多领域的共同关注点，大量的文本信息，如新闻，需求文档等，由相关人员使用自然语言撰写，从大量的数据中找出需要的数据的语义搜索技术是必须的。对科技文献的语义搜索可以让我们获取自己学习和研究所需要的文献；了解与自己的研究课题或研究方向相关的学术成果，掌握学术发展动态；分析文献信息，启发研究思路；获取必要的学术信息（如会议信息、专家信息、刊物信息等）。我们有很多事需要做：为了提高语义分析的准确性，需要搜集、创建大规模的语义知识资源，包括机器可处理的语义字典等；大多数期刊评价体系都是分学科进行评价，但是不同的评价体系在进行期刊分学科评价时，对于期刊分类归属的方法都不相同，所以需要对分学科方法进行详细解读，分析了按学科论文入手进行指标值统计，然后根据期刊所载论文和施引论文的分类进行综合计算给期刊自然归类的方法，并提出进一步研究的设想。在实践过程中，通过“面向专利情报研究的SAO语义结构分析方法述评“了解了sao语义结构分析方法；通过”融合HowNet和词林信息含量的词语相似度计算“了解到词林和知网的词语相似度算法经过融合后对相似度计算的有效提升与强力的计算能力，通过” 文本相似度计算方法研究综述“中的一些计算公式算法，有了明确的参考，这些文献让我们受益匪浅。

## 二、数据工程

**2.1、工程描述**

本文主要负责数据抽取方面的工作，为了实现将论文中的数据以json格式抽取出来并成功放入ElasticSearch数据库，做后续的精细化搜索或评估，我制定了四个任务工程。工程过程可分为：将学科分类表转换为json格式文件、将论文路径抽取并转换为json格式，这两步的目的是做数据加工和文献的分类，后续有助于提高检索精度和效率；之后将pdf论文抽取并转化为json格式文件；使用paddleocr轻量模型将图片信息提取成为文本信息。在得到需要的json文件后，将其传送到es中，以便后续同学进行操作。逻辑上就是学科分类是为文献分类做基础，然后进一步在做pdf分类（在pdf抽取文本基础上）。

工程流程图：



**2.2、pdf文件转化为json格式文件**

**2.2.1 json的背景**

JSON是 (JavaScript Object Notation, JS 对象标记)，它是一种数据交换格式。在JSON出现之前，大家一直用XML来传递数据。因为XML是一种纯文本格式，所以它适合在网络上交换数据。XML本身不算复杂，但是，加上DTD、XSD、XPath、XSLT等一大堆复杂的规范以后，任何正常的软件开发人员碰到XML都会感觉头大了，最后大家发现，即使你努力钻研几个月，也未必搞得清楚XML的规范。

终于，在2002年的一天，道格拉斯·克罗克福特（DouglasCrockford）同学为了拯救深陷水深火热同时又被某几个巨型软件企业长期愚弄的软件工程师，发明了JSON这种超轻量级的数据交换格式。

由于JSON非常简单，很快就风靡Web世界，并且成为ECMA标准。几乎所有编程语言都有解析JSON的库，而在JavaScript中，我们可以直接使用JSON，因为JavaScript内置了JSON的解析。把任何JavaScript对象变成JSON，就是把这个对象序列化成一个JSON格式的字符串，这样才能够通过网络传递给其他计算机。如果我们收到一个JSON格式的字符串，只需要把它反序列化成一个JavaScript对象，就可以在JavaScript中直接使用这个对象了。

**2.2.2 json的核心难点**

JSON对值的类型和格式有严格的规定

复合类型的值只能是数组或对象，不能是函数、正则表达式对象、日期对象。

简单类型的值只有四种：字符串、数值（必须以十进制表示）、布尔值和null（不能使用NaN, Infinity, -Infinity和undefined）。

字符串必须使用双引号表示，不能使用单引号。

对象的键名必须放在双引号里面。

数组或对象最后一个成员的后面，不能加逗号。

　　PS: 需要注意的是，空数组和空对象都是合格的JSON值，null本身也是一个合格的JSON值。

JSON 的语法规则

JSON 的语法规则十分简单，可称得上“优雅完美”，总结起来有：

数组（Array）用方括号(“[]”)表示。

对象（Object）用大括号（”{}”）表示。

名称/值对（name/value）组合成数组和对象。

名称（name）置于双引号中，值（value）有字符串、数值、布尔值、null、对象和数组。

并列的数据之间用逗号（“,”）分隔

实例

JSON 数据的书写格式是：名称/值对。

名称/值对组合中的名称写在前面（在双引号中），值对写在后面，中间用冒号隔开，

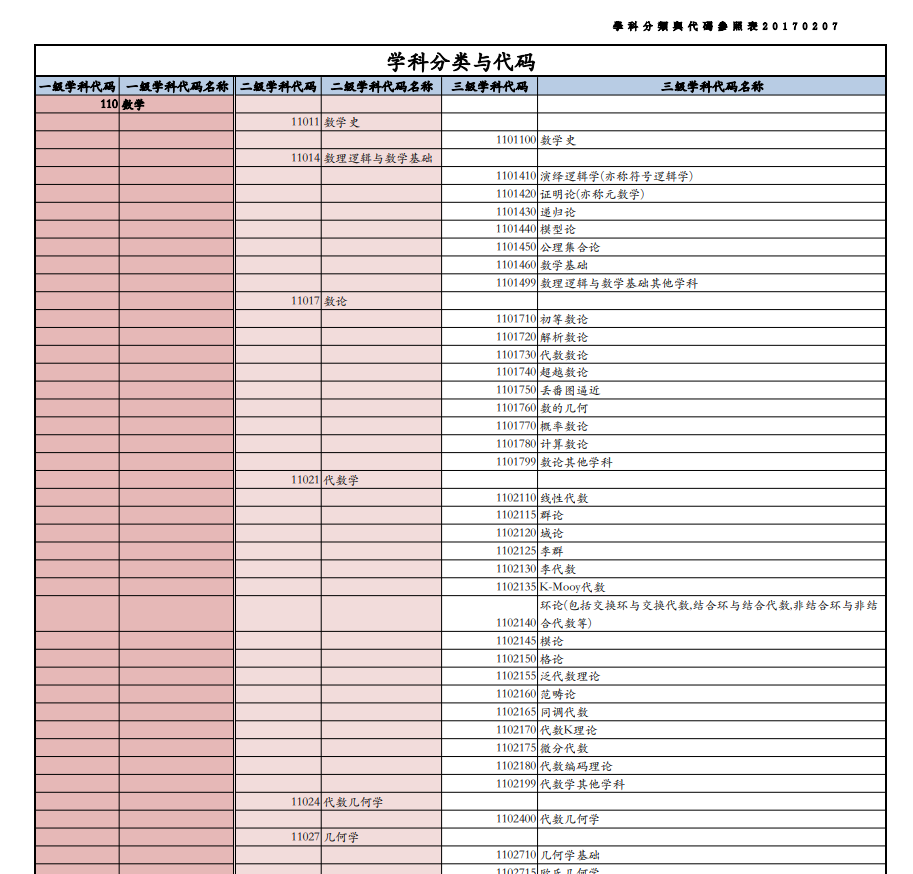
其中值可以是：数字（整数或浮点数）、字符串（在双引号中）、布尔值（true或false）、数组（在方括号中）、对象（在花括号中）、null

varjson= {"password":123456,"name":"myname","Booleans":true,"Array":[x,y,z],"object":{}}

或者是嵌套使用

**2.2.3 抽取过程**

1 找到要转换的pdf文件



左键双击打开查看，确认无误，

2 导入excel表格

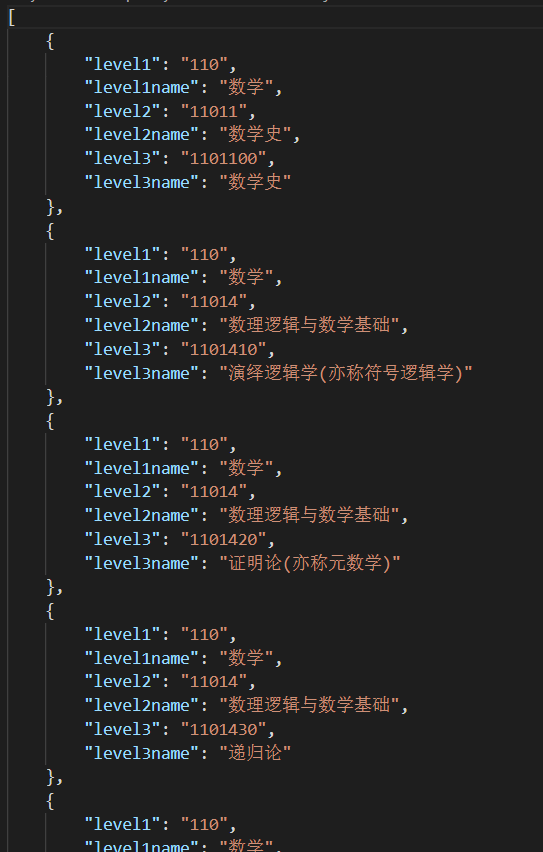
过程中对excel表格的格式要求很严格，需要人工校对



3 将表格中数据全选上传到转换网站**<https://uutool.cn/excel/>**，获取转换格式后的json文件



实验结果：获得了pdf转换为json格式的文件



**2.3、pdf格式文件抽取**

**2.3.1 使用工具**

Pdf格式论文抽取较为复杂，需要抽取论文的许多属性，例如作者、关键词、摘要、页数等，这个过程中我了解到 了许多关于pdf抽取的工具，不同代码实现效果不同。

目前Python用于处理PDF的模块，主要有3个：

PyPDF2：模块成熟，最后一次更新在2年前，适合页面级操作，文字提取效果较差。

PDFMiner：擅长文字抽取，目前主分支已停止维护，取而代之的是pdfminer.six

pdfplumber：基于pdfminer.six的文本内容抽取工具，使用门槛更低，如支持表格提取。

**2.3.2 PyPDF2**

功能简介：

提问文档信息（标题、作者）

逐页分隔文档

逐页合并文档

裁剪页面

将多个页面合并至单个页面

解密和加密PDF文档

读取pdf文档

import PyPDF2

pdf = PyPDF2.PdfFileReader('test.pdf')

获取文档信息

# 下面两个函数等价

print(pdf.getDocumentInfo())

print(pdf.documentInfo)

获取文档页码

print(pdf.getNumPages())

提取第m页的内容

print(pdf.getPage(m))

**2.3.3 PDFMiner**

读取pdf文件

#打开一个pdf文件

fp = open(u'C:/95530.pdf', 'rb')

#创建一个PDF文档解析器对象

parser = PDFParser(fp)

#创建一个PDF文档对象存储文档结构

#提供密码初始化，没有就不用传该参数

#document = PDFDocument(parser, password)

document = PDFDocument(parser)

#检查文件是否允许文本提取

if not document.is\_extractable:

raise PDFTextExtractionNotAllowed

#创建一个PDF资源管理器对象来存储共享资源

#caching = False不缓存

rsrcmgr = PDFResourceManager(caching = False)

# 创建一个PDF设备对象

laparams = LAParams()

# 创建一个PDF页面聚合对象

device = PDFPageAggregator(rsrcmgr, laparams=laparams)

#创建一个PDF解析器对象

interpreter = PDFPageInterpreter(rsrcmgr, device)

#处理文档当中的每个页面

# doc.get\_pages() 获取page列表

#for i, page in enumerate(document.get\_pages()):

#PDFPage.create\_pages(document) 获取page列表的另一种方式

replace=re.compile(r'\s+');

# 循环遍历列表，每次处理一个page的内容

for page in PDFPage.create\_pages(document):

interpreter.process\_page(page)

# 接受该页面的LTPage对象

layout=device.get\_result()

# 这里layout是一个LTPage对象 里面存放着 这个page解析出的各种对象

# 一般包括LTTextBox, LTFigure, LTImage, LTTextBoxHorizontal 等等

for x in layout:

#如果x是水平文本对象的话

if(isinstance(x,LTTextBoxHorizontal)):

text=re.sub(replace,'',x.get\_text())

if len(text)!=0:

print text

**2.3.4 pdfplumber**

.extract\_text() 用来提页面中的文本，将页面的所有字符对象整理为的那个字符串

.extract\_words() 返回的是所有的单词及其相关信息

.extract\_tables() 提取页面的表格

.to\_image() 用于可视化调试时，返回PageImage类的一个实例

用法展示：

import pdfplumber

import pandas as pd

with pdfplumber.open("95535.pdf") as pdf:

page = pdf.pages[1] # 第一页的信息

text = page.extract\_text()

print(text)

table = page.extract\_tables()

for t in table:

# 得到的table是嵌套list类型，转化成DataFrame更加方便查看和分析

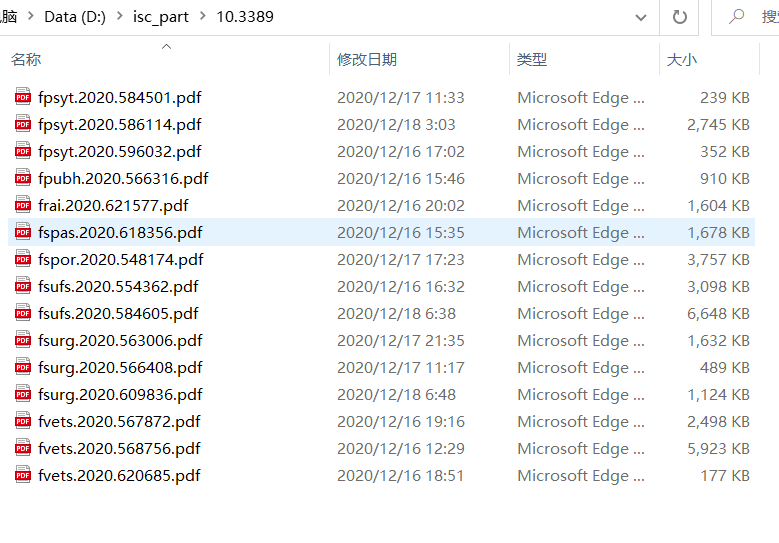
​ df = pd.DataFrame(t[1:], columns=t[0])

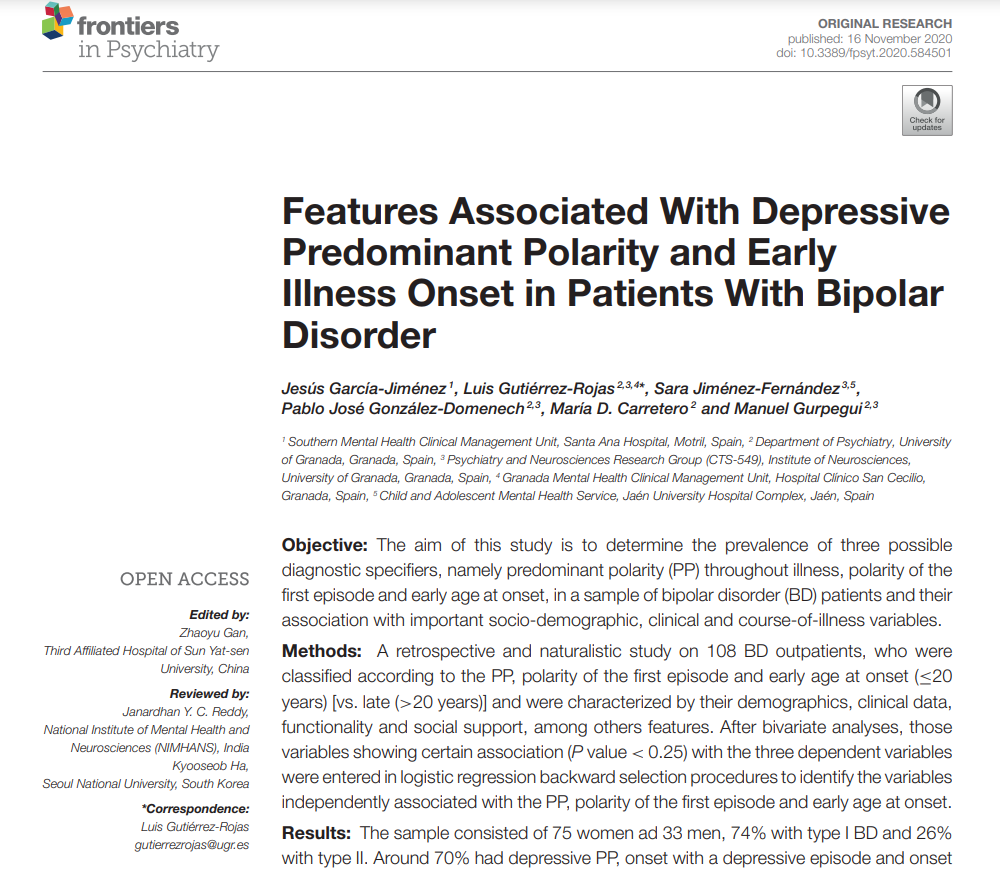
​ print(df)

**2.3.5 实验过程**

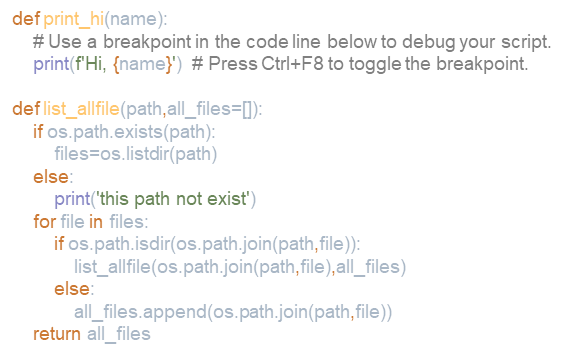
1爬取所有文件夹下的路径

为了便于批量查找所有论文的路径，在网上查阅相关资料后，应用os.listdir() （用于返回指定的文件夹包含的文件或文件夹的名字的列表）和os.path.isdir()（判断某一路径是否为目录）等方法来实现抽取出大量的路径，之后整理格式为json。

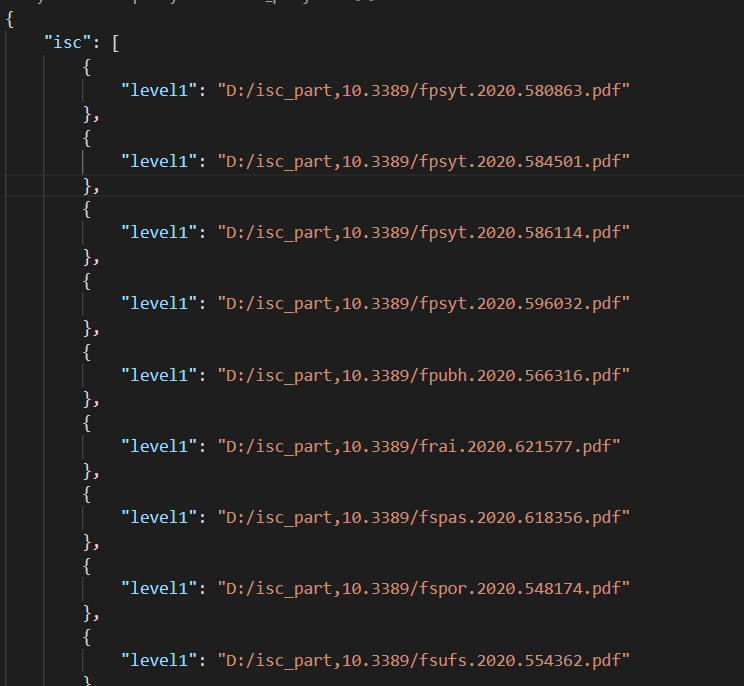




使用了如下代码进行路径的抽取：



最终结果：



2使用pypdf2库抽取

使用第一步中爬取到的目录，调用库中的方法可以得到论文的属性，但是得到的数据可能会有缺失也就是脏数据

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

all\_files=list\_allfile(r'D:\isc\_part')

# file\_handle = open('newtext.txt', mode='w')

for path in all\_files:

pdf\_title = PdfFileReader(path).getDocumentInfo().title # 标题

pdf\_author = PdfFileReader(path).getDocumentInfo().author

pdf\_subject = PdfFileReader(path).getDocumentInfo().subject

pdf\_creator= PdfFileReader(path).getDocumentInfo().creator

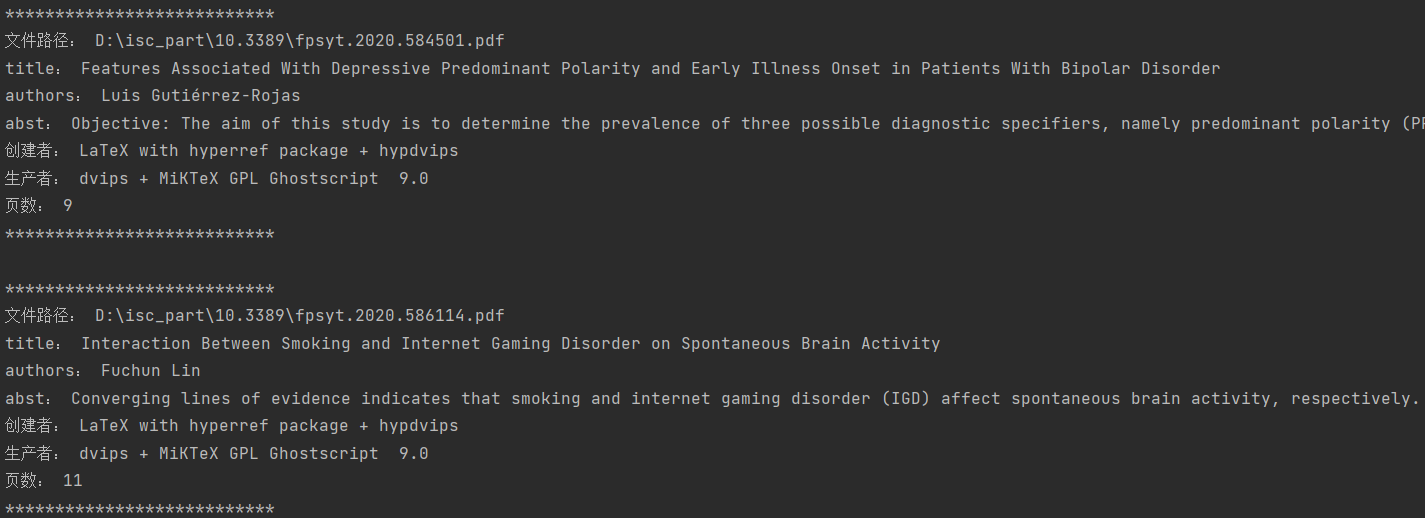
pdf\_producer= PdfFileReader(path).getDocumentInfo().producer

reader = PdfFileReader(path)

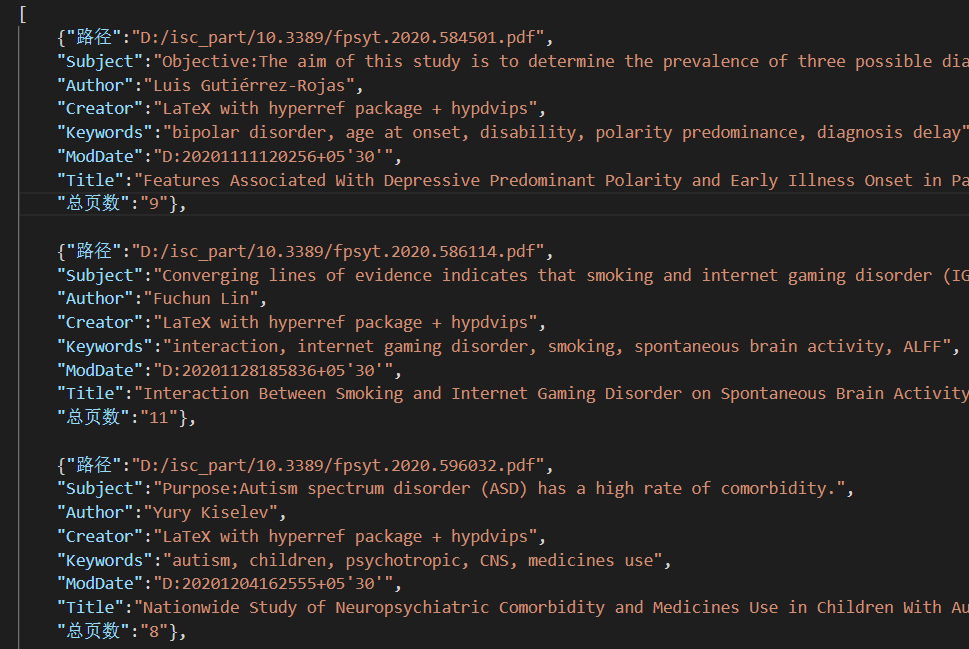
page = reader.getNumPages()

# data = open("C:/Users/rhy20/Desktop/newtext.txt", 'w+')

print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n文件路径：",path,"\ntitle：",pdf\_title,"\nauthors：",pdf\_author,'\nabst：', pdf\_subject,'\n创建者：',pdf\_creator,'\n生产者：',pdf\_producer,'\n页数：',page,"\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n")

将得到的结果整理格式转换成json

结果为：



**2.3、paddleocr将图片转换成信息**

**2.3.1 paddleocr介绍**

PaddleOCR发布的超轻量模型，主要由4.1M的检测模型和4.5M的识别模型组成。其中，检测模型的Base模型采用DB算法，文本模型的Base模型采用经典的CRNN算法。鉴于MobileNetV3在端侧系列模型中的优越表现，两个模型均选择使用MobileNetV3作为骨干网络，可将模型大小初步减少**90%以上**。

**2.3.2 安装使用PaddleOCR**

1.github下载：https://github.com/PaddlePaddle/PaddleOCR，解压。接下来需要配置PaddleOCR所需的环境

首先，推荐新建一个环境：

conda create -n paddle38 python=3.8

2.安装PaddlePaddle

pip install paddlepaddle -i https://mirror.baidu.com/pypi/simple

3.安装shapely，但是需要先从https://www.lfd.uci.edu/~gohlke/pythonlibs/下载shapely安装包Shapely‑1.7.1‑cp38‑cp38‑win\_amd64.whl，将其复制到F:\Anaconda3\envs\paddle38\libs文件夹下（根据自己新建的环境的位置修改），然后在anaconda Prompt中，cd到该目录下，执行：

pip install Shapely‑1.7.1‑cp38‑cp38‑win\_amd64.whl

4.cd到刚才解压的PaddleOCR文件夹中，执行：

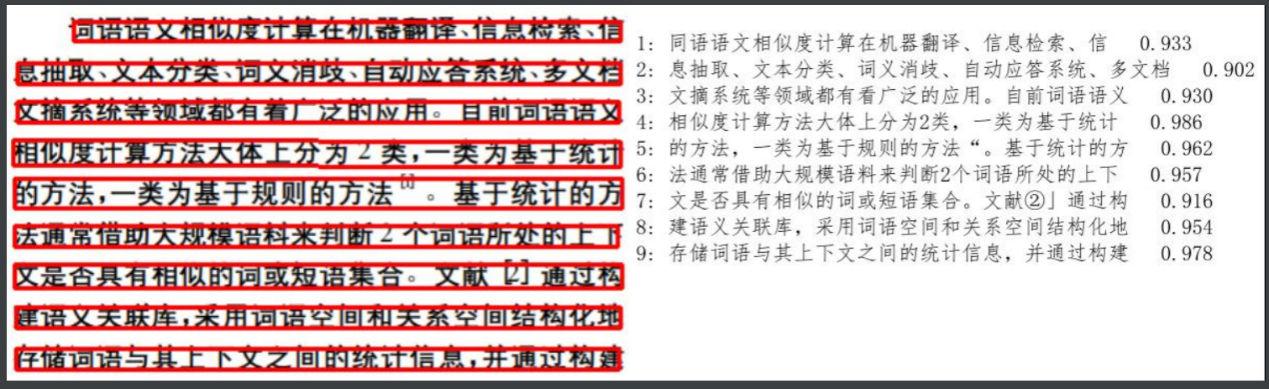
pip install -r requirements.txt

完成安装



使用框架后得到了图片转换的结果：



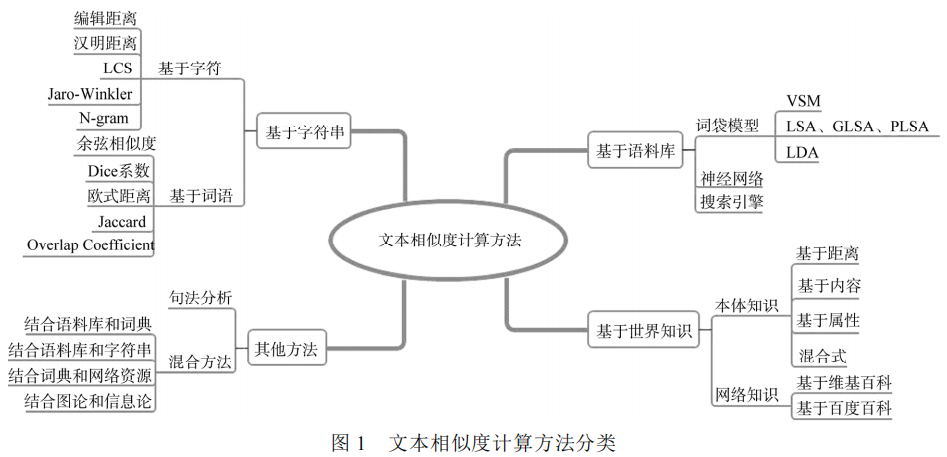


## 三、相似度匹配算法

**3.1、文本相似度匹配概述**

文本相似度表明了相似度与文本之间的共性和差异有关, 共性越大、差异越小, 则相似度越高; 共性越小、差异越大, 则相似度越低; 相似度最大的情况是文本完全相同。通过文本相似度算法我们可以得出两个文本或者两个向量是否存在一些联系，通过返回相似度的值得知两篇文章是否相似。当完成算法之后可以通过使用这个算法实现输入一段话后，调用Elasticsearch数据库，能够找到与这句话最为相似的文章，返回这篇文章的题目，摘要，路径等。

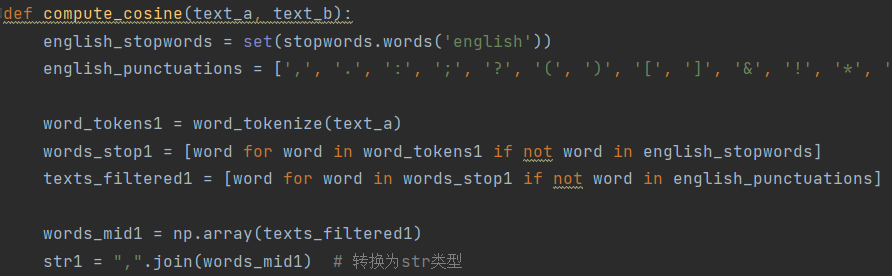
下图是文本相似度计算的各类方法，可以分为四大类分别是基于字符串，基于语料库，基于世界知识和其他算法。



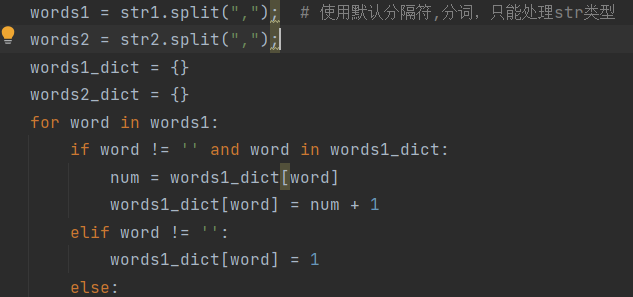
**3.2、算法的实现**

实现方法为基于字符串进行实现用余弦相似度算法进行实现。

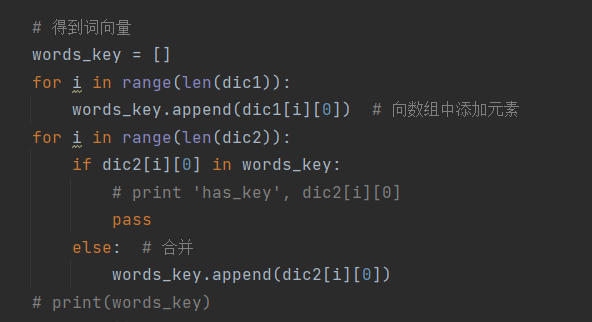
(1)将两句话进行初始化去除停用词,标点等无关内容



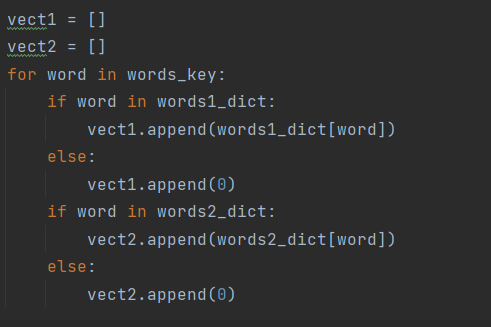
(2)将词语进行分词,创建一个字典记录每一个单词出现的次数



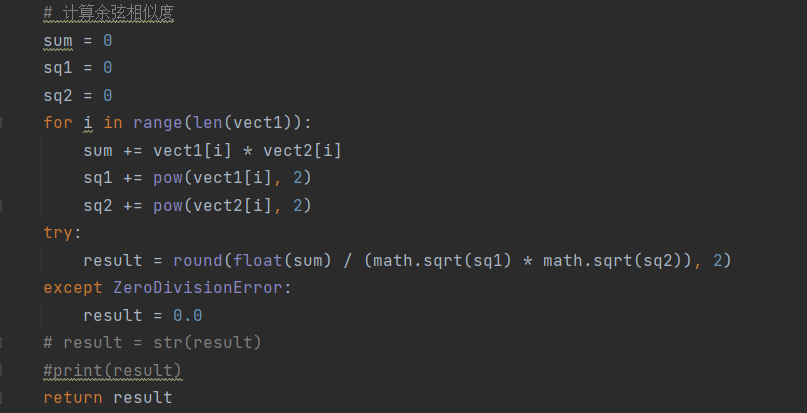
(3)将所有出现的词放到一个列表中



(4)计算词向量

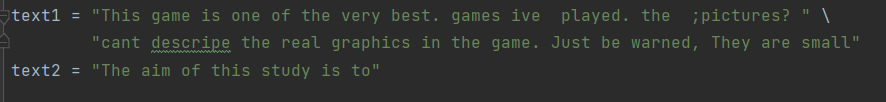


(5)计算两个语句向量形成的余弦值

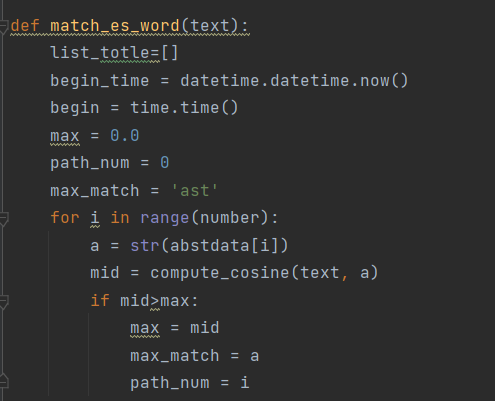


**3.3、算法的部署与测试**

(1)输入一段英文文字例如test1或者text2

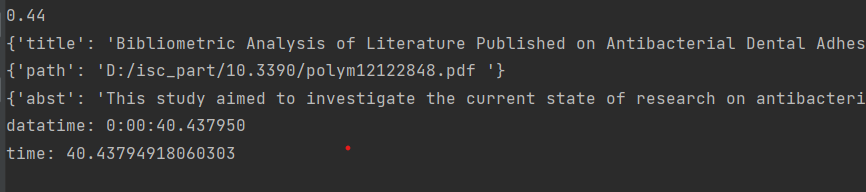


(2)通过函数让句子与存储在es数据库中文本匹配





(3)得出结论匹配相似度,标题,路径与摘要



1. 也可以使用flask模型将函数变为API接口，通过访问接口输入句子实现匹配，详细的请参照第五章



**3.4、对于后续的知识工程和语义搜索提供支持**

通过调用文本匹配算法实现两个文本的相似度计算,对知识工程部分实现三元组的匹配,返回匹配度或者相关数据.对于搜索部分能够实现输入句子在数据库寻找并且返回匹配程度最高的结果.

## 四、知识工程

**4.1 es的介绍与实现**

**4.1.1** **Elasticsearch基本概念**

Elasticsearch是一个基于Lucene的搜索服务器。它提供了一个分布式多用户能力的全文搜索引擎，基于RESTful web接口。Elasticsearch是用Java语言开发的，并作为Apache许可条款下的开放源码发布，是一种流行的企业级搜索引擎。Elasticsearch用于云计算中，能够达到实时搜索，稳定，可靠，快速，方便.

Elasticsearch可以进行以下工作:

1、分布式实时文件存储，并将每一个字段都编入索引，使其可以被搜索。

2、实时分析的分布式搜索引擎。

3、可以扩展到上百台服务器，处理PB级别的结构化或非结构化数据。

索引库

Lucene交互

应用程序

**4.1.2、存储方式**

Elasticsearch的文件存储，Elasticsearch是面向文档型数据库，一条数据在这里就是一个文档，用JSON作为文档序列化的格式



**4.1.3、Elasticsearch使用**

一是通过pycharm使用python脚本调用链接es之后进行增删改查

例如:

es = Elasticsearch(

['http://180.76.242.135:9200/'],

http\_auth=('eswzj', '2019011415'),

port=9200,

use\_ssl=False

)

使用es一些基本代码

(1)#查询数据

es.get(index='shiyan', doc\_type='\_doc', id='3')

(2)# 查询name

body = {

"query":{

"match":{

"name":"ll2"

}

}

}

re = es.search(index="shiyan",doc\_type="\_doc",body=body)

(3)# multi\_match: 在name和addr和age里匹配包含beijing关键字的数据

# 查询name,addr,age包含"beijing"关键字的数据

body = {

"query":{

"multi\_match":{

"query":"beijing",

"fields":["name", "addr","age"]

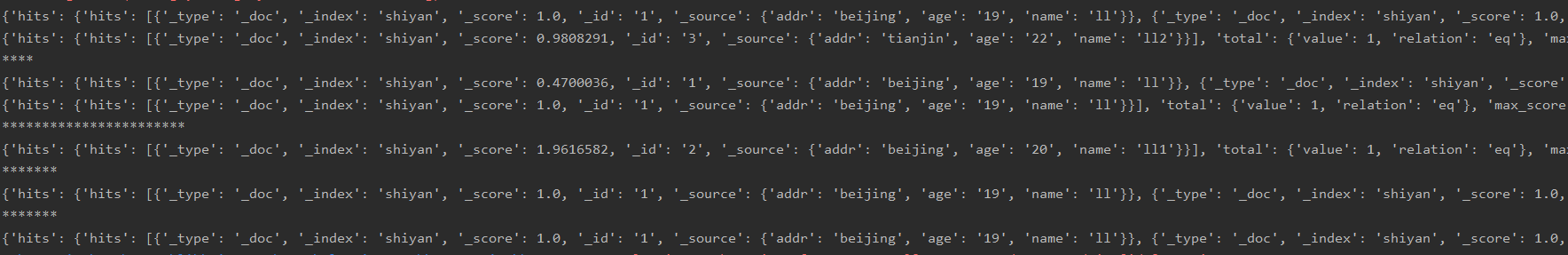
}

}

}

re = es.search(index="shiyan",doc\_type="\_doc",body=body)

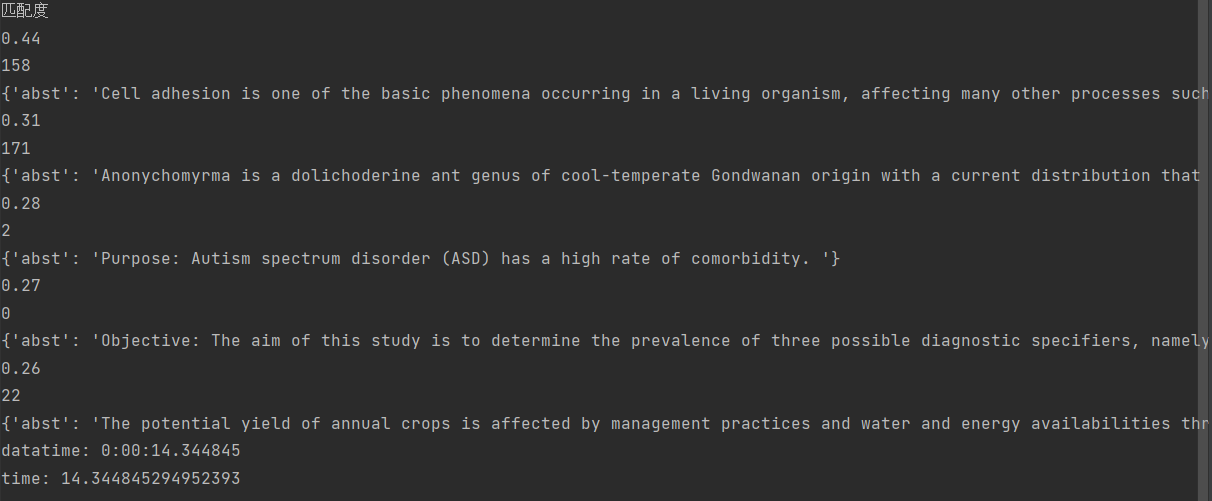
基本操作结果显示



二是使用Kibana进行基本增删改查等



(4)可以通过文本匹配算法，输入一段文本与es中的数据进行匹配，将匹配的相似度按照大小进行排序,显示出相似度比较高的几个,显示出相似度,数据库中的索引和摘要



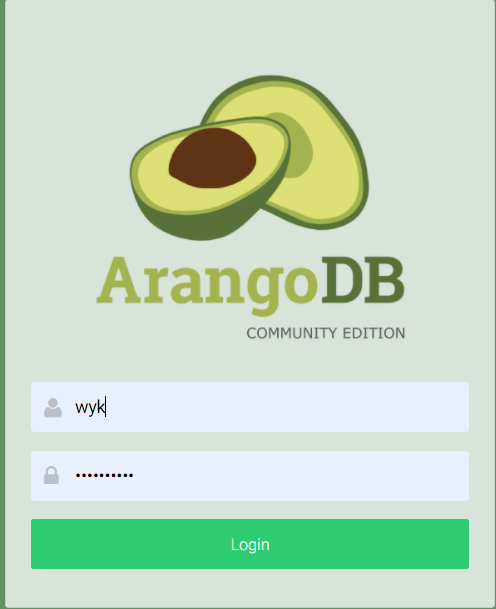
**4.2 图数据库与操作**

**4.2.1 图数据库**

知识图谱的构建，离不开图数据库。图数据库源起欧拉和图理论，也可称为面向/基于图的数据库，对应的英文是Graph Database。图数据库的基本含义是以“图”这种数据结构存储和查询数据，而不是存储图片的数据库。它的数据模型主要是以节点和关系（边）来体现，也可处理键值对。它的优点是快速解决复杂的关系问题。

图可以说是顶点和边的集合，或者说更简单一点儿，图就是一些节点和关联这些节点的联系（relationship）的集合。图将实体表现为节点，实体与其他实体连接的方式表现为联系。我们可以用这个通用的、富有表现力的结构来建模各种场景，从宇宙火箭的建造到道路系统，从食物的供应链及原产地追踪到人们的病历，甚至更多其他的场景。

常见的图数据库应用有Neo4j、Titan、Dgraph、ArangoDB、OrientDB。本文主要学习ArangoDB。



三元组生成，导师给了一个词库，因为小组没有语义匹配的经验，我们初步尝试使用字符匹配的方式，初步尝试的关系只有两种[“sub class of”，“is related to”]，比如“控制”与“远程控制”就是子类的关系（sub class of），“液压”与“液压缸”就是仅仅是相关的关系(is related to) 。

数据源是mergeWorld.txt，共有909773各单词，生成三元组超过22752万条，我们选择部分数据，在arangoDB数据库中进行尝试，构建图结构。

**4.2.2 图数据库操作**

分析官方数据样例，形成自己的数据的点集与对应的边集。

进入ArangoDB的目录，像这样输入文件的路径与文件名，



这样就导入了点集，接着输入

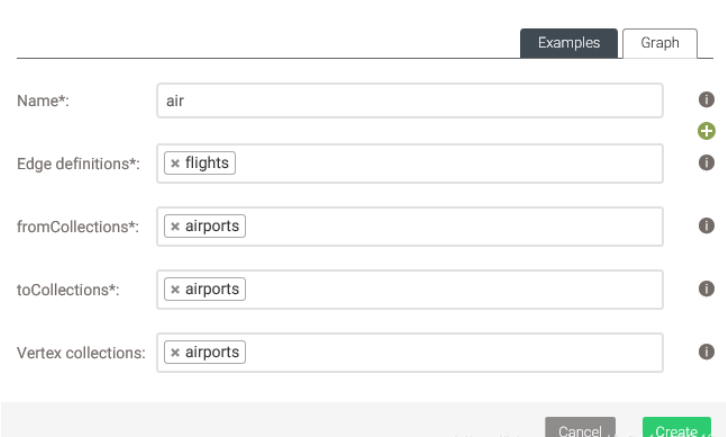


然后导入对应的边集，

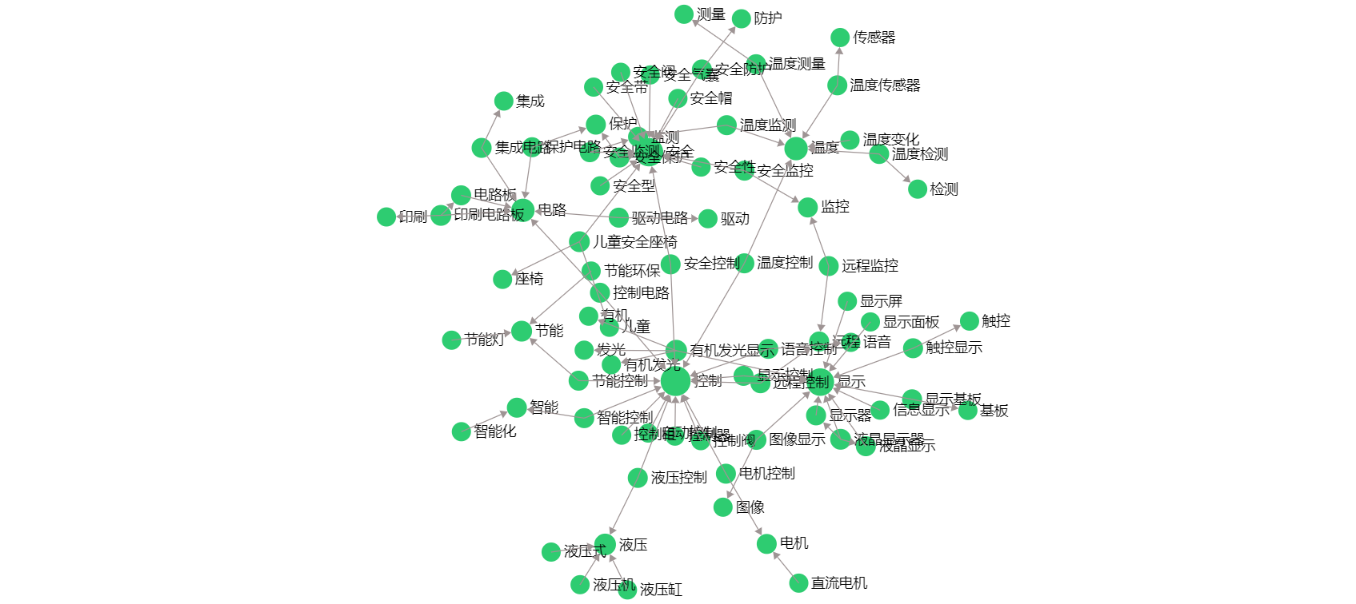
启动ArangoDB，

登陆账号

创建graph，指定节点与边的来源，创建图结构；



可以查看生成的图结构：



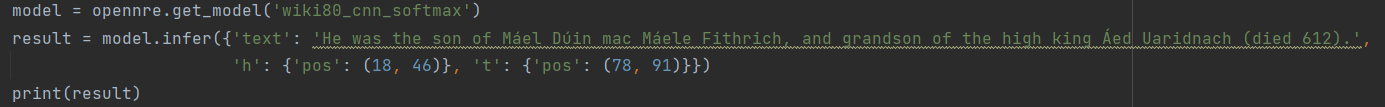
**4.3 关系抽取**

**上一节讲的基于字符的三元组抽取过于朴素，关系种类少，可用性弱。目前主流的关系抽取是基于深度学习的关系抽取，有效地改善了特征提取误差传播的问题，是近些年的研究热点。**主要的方法有卷积神经网络 （Convolutional Neural Network，CNN）和循环神经 网络（Recurrent Neural Network，RNN）两大类。。一般采用 CNN，RNN及其改进的模型进行关系抽取，由 于它们不需要手动构造特征并且预测结果的精度很高，因而得到了广泛的关注。

小组学习了deepdive-由斯坦福大学InfoLab实验室开发的一个开源知识抽取系统。还学习了deepke、openNRE等优秀的开源关系抽取项目。

以openNRE为例，先学习如何调用官方训练好的模型。

调用wiki80\_cnn\_softmax关系抽取模型，输入带抽取的例句：



运行模型，给出预测：



**当然，我们还学习了openNRE的数据集，尝试训练出自己的模型。**

**训练数据集：nyt10m，来源于纽约时报New York Times所标注的语料，命名实体是通过 Stanford NER 工具并结合 Freebase 知识库进行标注的。命名实体对之间的关系是链接和参考外部的Freebase知识库中的关系，结合远监督方法所得到的。**

**单条数据格式：**

**{"text": "A court in Rome acquitted five people accused of conspiring to murder Roberto Calvi , who was president of Banco Ambrosiano , one of Italy 's largest private banks , and a financial adviser to the Vatican when he was found hanged under Blackfriars Bridge in London in 1982 , his pockets stuffed with rocks , bricks and cash .", "h": {"pos": [70, 83], "id": "m.0fqrk", "name": "Roberto Calvi"}, "t": {"pos": [133, 138], "id": "m.03rjj", "name": "Italy"}, "relation": "/people/person/place\_lived"}**

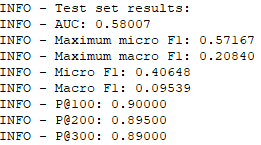
**有text、h、t、relation四个部分。**

**关系种类：共有25种关系**



**训练模型：基于CNN的包级别关系抽取模型。**

**训练结果：**



**4.4 知识驱动的匹配算法部署**

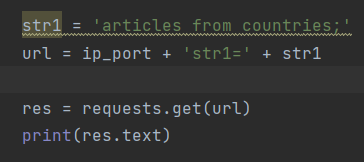
**实现关系抽取一是可以以图的形式直观地展示文献格式，可以帮助用户更快的对文献内容做一个了解，二是可以优化匹配逻辑。**

**例如，针对这篇文献的摘要 ：**

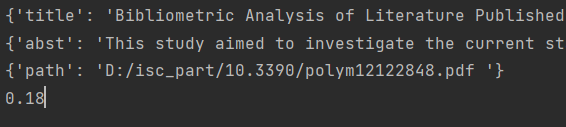
**This study aimed to investigate the current state of research on antibacterial dental adhesives. The interest in this field can be drawn from an increasing number of scholarly works in this area. However, there is still a lack of quantitative measurement of this topic. The main aim of this study was to consolidate the research published on the antibacterial adhesive from 1996 to 2020 in Web of Science indexed journals. The bibliometric method, a quantitative study of investigating publishing trends and patterns, was used for this study. The result has shown that a gradual increase in research was found, whereby a substantial increase was observed from 2013. A total of 248 documents were published in 84 journals with total citations of 5107. The highly cited articles were published mainly in Q1 category journals. Most of the published articles were from the USA, China, and other developed countries; however, some developing countries contributed as well. The authorship pattern showed an interdisciplinary and collaborative approach among researchers. The thematic evaluation of keywords along with a three-factor analysis showed that ‘antibacterial adhesives’ and ‘quaternary ammonium’ have been used commonly. This bibliometric analysis can provide direction not only to researchers but also to funding organizations and policymakers.**

可以找出存在一条三元组：【articles-from-countries】

那么可以使用这一条三元组对文献进行搜索：



使用的是python代码的request方法，访问对应的url，打印返回结果



即使只有3个单词，依然找出了对应的文献。

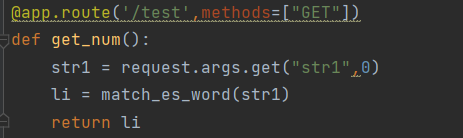
## 五、文献搜索

**5.1 工程实践**

基于第二章的抽取出的科技文献信息以及第三章的相似度匹配算法，小组尝试将实现文献搜索功能，并将功能包装为API，部署在服务器上，可以实现本地调用。

**搜索过程**：

将第三章相似度匹配算法抽象为函数，使用flask框架将功能部署对应的API路径，返回处理结果，返回语句与ES数据库中文献摘要匹配度最高的一条数据。



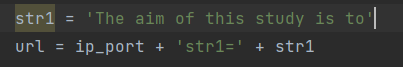
开启服务器程序：



客户端指定对应的url：



输入待匹配文本：



Request访问：



等待返回结果，并打印。

也可以直接在浏览器搜索的形式进行访问，

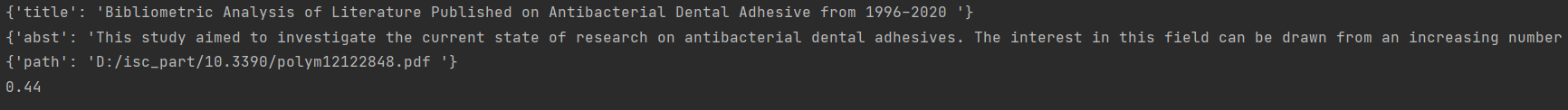


**回车，等待结果。**

**5.2 结果**

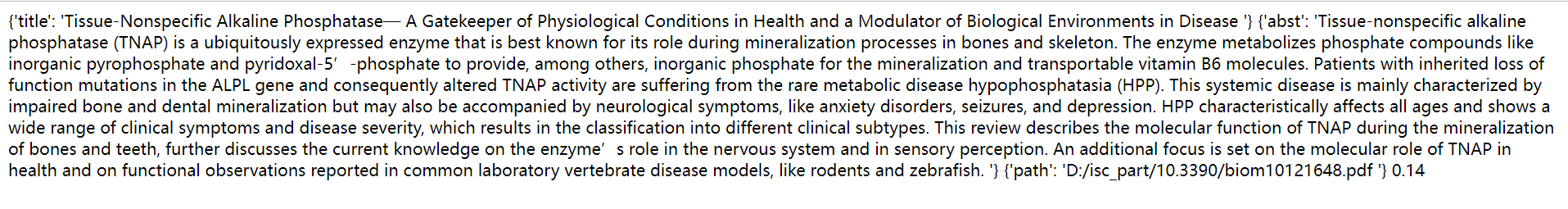
测试的待匹配语句：“The aim of this study is to”

Request方式访问并打印结果：



结果包括数据库中论文摘要与用户输入最匹配的论文的**title**、**abstract**、**path**、匹配相似度。

直接在浏览器访问



**得到的处理结果相同。**

**致谢：**

经过了这一学期的科研实践,我们非常感谢王楠老师悉心的教导与耐心的培养,由于这是我们第一次接触科研实践项目,也是第一次接触深度学习这类的知识,虽然过程并不是一帆风顺,中间难免会遇到困难与障碍,但由于王楠老师要求每周进行科研进度的汇报,坚持每周进行总结,汇总一周的进度与困难,每周在展示成果的时候会对我们提出新的要求与指出我们的不足督促我们进步,让我们最终都收获满满.感谢老师!

参考文献

[1] 秦春秀, 赵捧未, 刘怀亮. 词语相似度计算研究[J]. 情报 理论与实践, 2007, 30(1): 105-108. (Qin Chunxiu, Zhao Pengwei, Liu Huailiang. Research on Word Similarity Measurement [J]. Information Studies: Theory & Application, 2007, 30(1): 105-108.)

[2] 李慧. 词语相似度算法研究综述 [J]. 现代情报, 2015, 35(4): 172-177. (Li Hui. A Review on the Research of Word Similarity Algorithms[J]. Journal of Modern Information, 2015, 35(4): 172-177. )

[3]Lin D. An Information-theoretic Definition of Similarity [C]// Proceedings of the 15th International Conference on Machine Learning.1998

[4]徐蕙,及洪泉,姚晓明,李香龙,陆斯悦.智能电网中基于知识图谱的语义搜索算法[J].实验室研究与探索,2021,40(04):71-74+86.DOI:10.19927/j.cnki.syyt.2021.04.017.

[5]张吉祥,张祥森,武长旭,赵增顺.知识图谱构建技术综述[J/OL].计算机工程:1-16[2021-12-27].DOI:10.19678/j.issn.1000-3428.0061803.

[6]张俊娥.核心期刊评价的分学科方法解析——以《中文核心期刊要目总览》为例[J].大学图书馆学报,2021,39(03):67-75.DOI:10.16603/j.issn1002-1027.2021.03.016.

[7]逯万辉,谭宗颖.基于深度学习的期刊分群与科学知识结构测度方法研究[J].情报学报,2020,39(01):38-46.

[8]马芳,黄翠玉.中文科技期刊论文多标签分类研究[J].图书情报导刊,2019,4(02):26-32.

[9]陈靖元,周刚,卢记仓.融合HowNet和词林信息含量的词语相似度计算[J/OL].小型微型计算机系统:1-7[2021-12-27].http://kns.cnki.net/kcms/detail/21.1106.TP.20210521.1625.024.html.

[10]陈二静,姜恩波.文本相似度计算方法研究综述[J].数据分析与知识发现,2017,1(06):1-11.