北京郵電大學

艺术类中文期刊文章信息检索与抽取系统

——《信息与知识获取》作业三

班级	学号	姓名	分工
2018211305	2018211290	崔思颖	系统模块设计、数据收集与处理、信息检索模块的设计与实现、UI 设计与实现、作业 2 报告的撰写
2018211305	2018211250	岑明洲	信息抽取模块的实现、作业 3 报告的撰写
2018211305	2018211291	吴川宇	倒排索引文件的生成与查找、作业 1 报告的撰写

2021 年 6 月 8 日

目录

实验概述

根据老师在课上介绍的关于信息检索和信息抽取的算法，参考软件工程所描述的软件系统开发方法，利用 Python 语言，开发简单的搜索引擎系统，主要实现信息检索、信息抽取和图形化界面等功能。

实验目的

自己动手设计实现一个信息抽取实验系统，中、英文皆可，可以在作业 2 信息检索系统的基础上实现，也可以单独实现。特定领域语料根据自己的兴趣选定，规模不低于 100 篇文档，进行本地存储。对自己感兴趣的特定信息点进行抽取，并将结果展示出来。其中，特定信息点的个数不低于 5 个。可以调用开源的中英文自然语言处理基本模块，如分句、分词、命名实体识别、句法分析。信息抽取算法可以根据自己的兴趣选择，至少实现正则表达式匹配算法的特定信息点抽取。最好能对抽取结果的准确率进行人工评价。界面不作强制要求，可以是命令行，也可以是可操作的界面。

实验环境

操作系统：Windows 10

集成开发环境（IDE）：PyCharm、QtDesigner

程序设计语言：Python

系统模块划分

本系统主要划分为一下五个功能模块：用户输入模、信息检索模块、相关度计算模块、信息抽取模块、输出模块，其中，相关度计算模块包含在信息检索模块内。模块图如下所示：

用户输入

模块编号	N-01-01
模块名称	用户输入处理模块
模块输入	用户输入的自然语言字串
模块输出	0.txt（用户输入文档文件）
模块功能	将用户输入作为文档以进行后续计算
模块实现	对用户输入进行去空格、分词、去停用词、筛选中文后写入 0.txt 中，作为目标文档。

信息检索

模块编号	N-02-01
模块名称	索引建立模块
模块输入	所有文档
模块输出	倒排索引文件
模块功能	使用文档生成用于查询的倒排索引文件，包含其在文中的词频、位置信息
模块实现	对于每个关键词，找出其在每篇文档中出现的频率、出现的位置，依次写入 index.txt 文件。

模块编号	N-02-02
模块名称	矩阵生成模块
模块输入	包含查询词的集合
模块输出	矩阵 M、Pos。M[i][j]表示包含单词 j 在文档 i 中的 TF*IDF 值，M 的最后一行表示用户查询内容的向量；Pos[i][j]为单词 j 在文档 i 中的位置列表。
模块功能	将相关结果以矩阵的形式加入程序中进行计算
模块实现	调用 sklearn 库，将分词后文档中的词语转化为词频矩阵，并将词频矩阵统计成 TF-IDF 值。

相关度计算

模块编号	N-03-01
模块名称	相关度计算模块
模块输入	矩阵 M
模块输出	以相关度从大到小排序的（文档序号，相关度）列表 Sim
模块功能	计算每篇文档与查询内容的相关度并排序输出
模块实现	调用 sklearn 库中的 cosine_similarity 函数，计算目标文档与其他文档的余弦相似度，按相关度值从大到小排列

信息抽取

模块编号	N-04-01
模块名称	实体识别模块
模块输入	文档编号
模块输出	实体识别结果
模块功能	将检索结果中的文档进行实体识别，进行特定信息点抽取后，输出
模块实现	调用 pkuseg 工具包的 cut 方法进行词性划分和实体识别，以及改写 cocoNLP 工具包中的 extractor 模块的源码来实现信息抽取

输出

模块编号	N-05-01
模块名称	信息检索输出模块
模块输入	列表 Sim，Pos 矩阵，全部文档
模块输出	用图形界面打印按相关度从大到小排序的查询结果，包括相关度、题目、日期、版号、作者、主要匹配内容等信息
模块功能	将信息检索的结果进行输出
模块实现	在 UI 界面中新建文本框与按钮，并对布局进行设计，每个文本框中首先输出包含关键词的文档头部，再通过倒排索引文件查找到关键词的位置，输出主要匹配内容。

系统实现

数据集

我们选择了现有的复旦大学中文语料库作为数据源，它由复旦大学李荣陆提供。我们选取了其中 500 篇艺术领域的文本作为检索对象，文本示例如下：

【文献号】3-385
【原文出处】文艺理论研究
【原刊地名】沪
【原刊期号】200101
【原刊页号】13～20
【分 类 号】J2
【分 类 名】中国古代、近代文学研究
【复印期号】200107
【标 题】中国古代的文艺本体论
【作 者】蔡锺翔
【作者简介】中国人民大学中文系
【正 文】
在进入本论题之前，有必要对“本体”这个概念作一点辨析。
记得数年前在文艺报刊上曾讨论过“本体”滥用的问题。如《文艺报》1995年42期刊登的黄力之的《“本体”的滥用及其文化意义》，指出了这种情况。滥用的始作俑者可能是王蒙同志，他在《读评论文章偶记》（载《文学评论》1985年6期）一文中有这样一段话：
我以为，我们更应该重视对文学的本体论的研究。对文学的本体的提法的科学性我并没有把握，我请求读者和专家原谅我知识的不足和用语的大胆。但我以为文学的本体是存在的，它就是文学所反映所追求所赖以发生的宇宙、自然、世界、人生、社会、生活、人类的精神世界，它也就是古往今来古今中外的文学作品、文学宝库本身。王蒙同志的看法是有意义的，但他解释的文学本体，确实不符合哲学上本体范畴的涵义，其实他所指的文学中所反映的内容、文学的本质特征以及文学作品本身等等，是不能用“本体”这个概念来概括的。确如上述那篇文章说的，这样的所谓“本体”就成了“大而无当的东西”了。后来《文学评论》1996年第6期又发表了朱立元教授的论文《当代文学、美学研究中对“本体论”的误释》。他举出了五种误释：一、把“本体”误作“本身”。二、把“本体”误释为世界万物的“本源”或“本性”。三、把“本体论”与“宇宙论”混淆起来。四、把“本体性”与过程性、体验性、自足性、根本性等意义相混淆。五、把哲学本体论与西方存在主义哲学的联系割断。他还指摘了中国哲学史学者汤用彤、张岱、熊十力的误释。朱教授的意见应该说是中肯的，但他是严格的以西律中，则未免有些绝对化。我认为，西方文化与中国文化是异质文化，西方的概念、范畴引进或移植到中国来是难免会走样的，往往要同中国原有的概念、范畴相比附，使之本土化，这种走样或“误释”有些是可以容许的，当然我们并不主张滥用。张岱年先生已经认识到西方的“本体”范畴在中国哲学中是没有的，他在《中国哲学大纲》一书中指出：
印度哲学及西洋哲学讲本体，更有真实义，以为现象是假是幻，本体是真是实。本体者何？即是唯一的究竟实在。这种观念，在中国本来的哲学中，实在没有。中国哲人讲本根与事物的区别，不在于实幻之不同，而在于本末、原流、根支之不同。万有众象同属实在，不惟本根为实而已。以本体为唯一实在的理论，中国哲人实不主持之。（注：《中国哲学大纲》（北京：中国社会科学出版社，1982年），页9。）因此张先生就采用了“本根”这一传统概念，以示与西方的区别。但汤用彤先生等也没有错，因为otology译为“本体”也是借用了传统概念，中国哲学中就有“本末体用”之辨，那末也可以有中国的“本体”范畴和“本体论”。《中国大百科全书·中国哲学卷》“本体论”一条将中西本体论分别疏解，不失为一种通达的处理。本文所论即是就中国哲学的本体论而言，但不把它解为“大而无当的东西”。
“本体”是什么？是指宇宙万物的本原或最高的抽象存在。“本体论”则是指探究宇宙万物存在的终极根据的理论。中国古代的哲学本体论，主要有三种观点，即以道为本的道本论，以气为本的气本论，和以心为本的心本论（以理为本的理本论，可以归属于道本论）。这几种本体论，决定了文艺学中的不同的文艺本体论；文艺本体论可以说是从哲学本体论推衍出来的，而文艺本体论则对文艺理论产生深刻的影响。不了解古代的文艺本体论，就很难理解和把握古代文艺理论的基本精神。以下分别论述三种文艺本体论及其在文艺理论中的影响。

具体实现

本次实现的信息抽取系统是对作业二中信息检索系统的扩展，即给作业二的信息检索系统增加一个信息抽取功能，对作业二中检索出来的内容进行信息抽取。该信息抽取系统选择了 5 个信息　点对文本进行抽取，分别是艺术分类、作品名、时间、地点、人物。我们使用了一些工具包来实现，比如 pkuseg 和 cocoNLP，具体实现分别如下：

抽取艺术分类

话不多说，先上核心代码：

```
import pkuseg
seg = pkuseg.pkuseg(postag=True)  # 加载模型，给定用户词典

def find_art(string: str, d: dict):
    to_find_list = {'美术', '音乐', '舞蹈', '文学', '戏剧', '影视', '摄影', '曲艺', '杂技', '建筑', '园林',
                    '电影', '电视', '绘画', '雕塑', '书法', '戏曲', '诗歌', '散文', '小说', '陶艺'}
    pos_list = seg.cut(string)
    arts = []
    for w in pos_list:
        if w[0] in to_find_list:
            arts.append(w[0])
    arts = list(set(arts))
    if len(arts) == 0:
        arts.append('无')
    d['艺术类别'] = arts
```

这里我们用到了北大的一个中文分词工具包 pkuseg，利用其提供的 cut(string)方法进行分词，若分出来的词语列表中包含艺术类别中的某个关键词，则抽取出一个艺术种类。

抽取作品名

核心代码如下：

```python
import re

def find_book(string: str, d: dict):
    pattern = re.compile(r'《(.*?)》')
    books = pattern.findall(string)
    books = list(set(books))
    for i in range(len(books)):
        books[i] = '《' + books[i] + '》'
    if len(books) == 0:
        books.append('无')
    d['作品'] = books
```

这里是用正则表达式匹配来进行抽取书名信息，由于文本中书名都被"《"和"》"包围，故可用正则表达式《(.*?)》来匹配书名号中的内容，然后在书名首位加上书名号还原即可。

抽取时间

经过我们的多次对比实验，我们发现对于时间实体的识别，北大的自然语言处理工具包 pkuseg 对时间的识别效果（效率、准确度）更好（相对于斯坦福的 StanfordCoreNLP，清华的 thulac，jieba，pyltp 和 cocoNLP），因此我们使用 pkuseg 进行时间实体的抽取。

核心代码如下：

```python
import pkuseg
seg = pkuseg.pkuseg(postag=True)  # 加载模型，给定用户词典

def find_time(string: str, d: dict):
    times = []
    pos_list = seg.cut(string)
    for w in pos_list:
        if w[1] == 't':
            times.append(w[0])
    times = list(set(times))
    if len(times) == 0:
        times.append('无')
    d['时间'] = times
```

这里调用了 pkuseg 的 cut(string)方法对文本进行分词和词性识别处理，对分词后识别词性为 t 的单词进行抽取，即把时间实体信息抽取出来。

抽取地点

核心代码如下：

```
from cocoNLP.extractor import extractor

def find_location(string: str, d: dict):
    locations = ex.extract_locations(string)
    locations = list(set(locations))
    if len(locations) == 0:
        locations.append('无')
    d['地点'] = locations
```

这里我们使用了网络中的一个工具包 cocoNLP 进行地点信息的抽取。一开始使用该工具包，发现对地点的识别抽取效果不是很理想，会有很多重复现象，（比如，'陕西省安康市汉滨区'会被抽取成['陕西省安康市汉滨区','安康市汉滨区','汉滨区']），因此我们决定修改工具包的源代码以达到理想效果，修改后的抽取效果只留下了['陕西省安康市汉滨区']这一项，消除了冗余项。

修改前的源代码如下：

```
def get_location(self, word_pos_list):
    location_list = []
    if word_pos_list == []:
        return []
    for i,t in enumerate(word_pos_list):
        word = t[0]
        nature = t[1]
        if nature == 'ns':
            loc_tmp = word
            count = i + 1
            while count < len(word_pos_list):
                next_word_pos = word_pos_list[count]
                next_pos = next_word_pos[1]
                next_word = next_word_pos[0]
                if next_pos=='ns' or 'n' == next_pos[0]:
                    loc_tmp += next_word
                else:
                    i = count
                    break
                count += 1
            location_list.append(loc_tmp)
    return location_list  # max(location_list)

def extract_locations(self, text):
    if text == '':
        return []
    seg_list = [(str(t.word), str(t.nature)) for t in HanLP.segment(text)]
    location_list = self.get_location(seg_list)
    return location_list
```

修改后的源代码如下：

```
def get_location(self, word_pos_list):
    location_list = []
    if word_pos_list == []:
        return []
    i = 0
    while i < len(word_pos_list):
        element = word_pos_list[i]
        word = element[0]
```

```
        nature = element[1]
        if nature == 'ns':
            loc_tmp = word
            count = i + 1
            while count < len(word_pos_list):
                next_word_pos = word_pos_list[count]
                next_pos = next_word_pos[1]
                next_word = next_word_pos[0]
                if next_pos == 'ns':
                    loc_tmp += next_word
                else:
                    i = count
                    break
                count += 1
            location_list.append(loc_tmp)
            i -= 1
        i += 1
    return location_list  # max(location_list)

def extract_locations(self, text):
    if text == '':
        return []
    seg_list = [(str(t.word), str(t.nature)) for t in HanLP.segment(text)]
    location_list = self.get_location(seg_list)
    return location_list
```

具体实现为对实体词性识别为 ns 的词组进行抽取，并合并相邻项。

抽取人物

核心代码如下：

```
from cocoNLP.extractor import extractor

def find_name(string: str, d: dict):
    names = ex.extract_name(string)
    names = list(set(names))
    if len(names) == 0:
        names.append('无')
    d['人物'] = names
```

这里我们仍然使用效果较好的 cocoNLP 工具包进行人名的识别抽取，同样的，我们也对源代码进行了修改，修改前对人名的抽取默认会选择频次最高的一个，（比如：'袁隆平','屠呦呦'只会被抽取

出其中出现频次最高的一个，修改后可以实现都抽取出来）。修改后的源代码如下：

```
def extract_name(self, text):
    if text == '':
        return []
    seg_list = [(str(t.word), str(t.nature)) for t in HanLP.segment(text)]
    names = []
    for ele_tup in seg_list:
        if 'nr' in ele_tup[1]:
            names.append(ele_tup[0])
    return list(set(names))
```

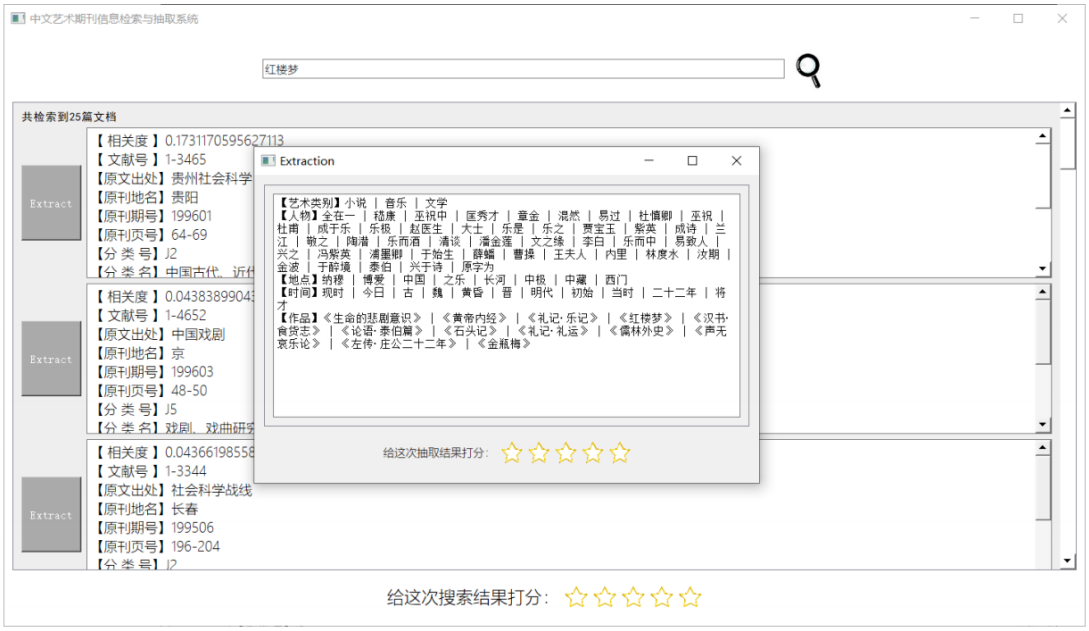
结果展示

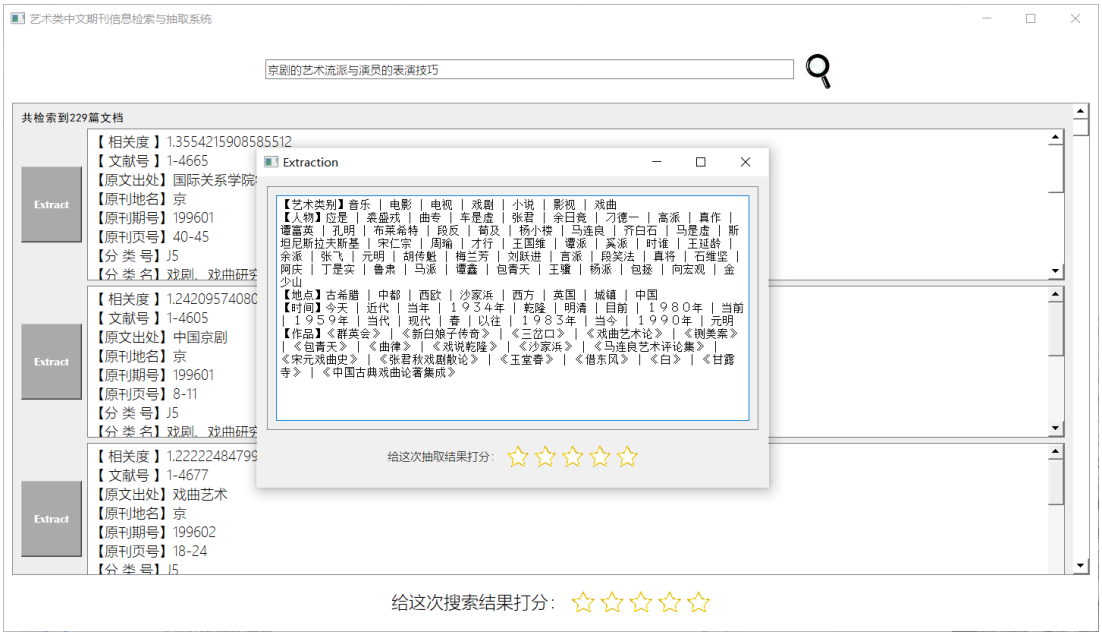
```
if __name__ == '__main__':
    txt_list = [20, 88]
    result = info_extract(txt_list)
    for i in result:
        print(i)
```

我们随便选择两个文档进行信息抽取，抽取结果如下：

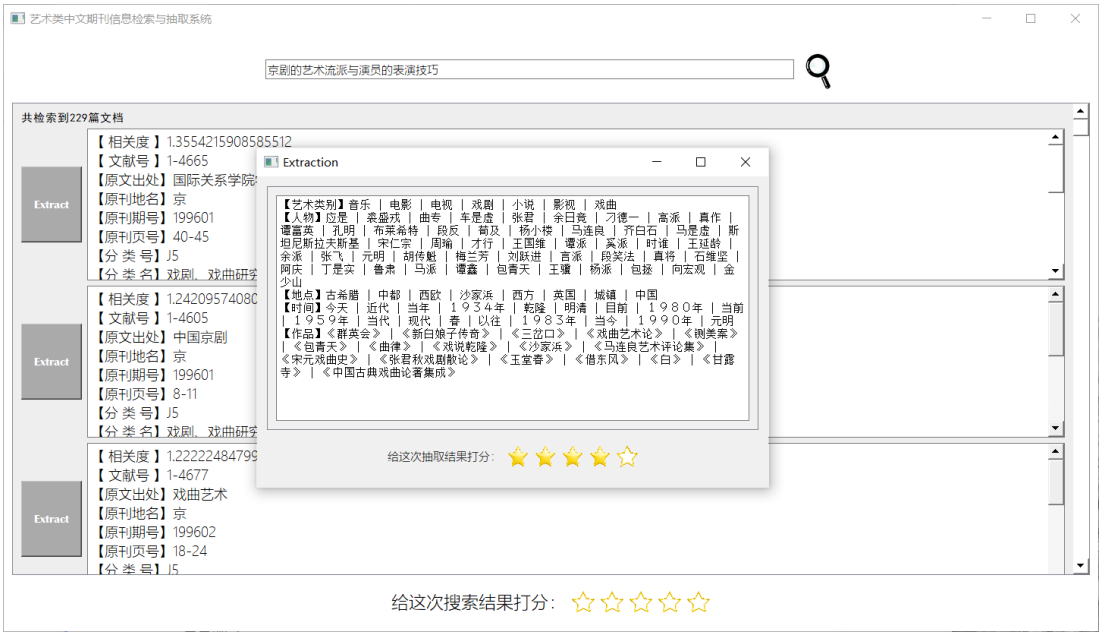
```
D:\python_Project\Extract\venv\Scripts\python.exe D:/python_Project/Extract/extract.py
艺术类别 ： 舞蹈 | 音乐 | 电影 | 戏曲
人物 ： 聂耳 | 格龙 | 龙格龙 | 郎上 | 卡衣 | 都能 | 洗星海 | 段以
地点 ： 欧洲 | 山西 | 黄河 | 日本 | 太行山 | 中国 | 苏联
时间 ： 过去 | 当时 | 前夜 | 九一八 | 1939年 | 现代 | 夜半
作品 ： 《九一八大合唱》 | 《黄河颂》 | 《只怕不抵抗》 | 《竹枝词》 | 《酷相思》 | 《天净沙》 | 《河边对口曲》 | 《花花的西伯利亚》 | 《黄河谣》 | 《保卫黄河》 | 《怒吼吧，黄河》 | 《黄河怨》 | 《在太行山上》

艺术类别 ： 电影 | 电视
人物 ： 卡桑 | 霍立
地点 ： 香港 | 北京 | 日本 | 英国 | 中国
时间 ： 年前 | 年初 | 1月 | 当代 | 12日 | 去年 | 1995年 | 年底 | 1996年 | 今年
作品 ： 《恐怖地带》 | 《卡桑得拉大桥》 | 《纽约大劫案》 | 《亡命天涯》 | 《生死时速》 | 《绝地战警》 | 《霹雳火》 | 《红番区》 | 《追捕》 | 《东京都新干线》 | 《真实的谎言》 | 《乱作一团》
```

与作业二的信息检索系统结合，进行信息抽取，点击搜索结果文档项旁侧的 Extract 按钮，即可实现对该文档的正文进行信息抽取，图形化界面结果如下：





点击 Extraction 窗口下方打分的五角星按钮，可以对抽取结果进行人工评价。点击过后按钮失效，避免重复评分。



评价的结果会写入项目文件夹中的“extraction_rating.csv”文件中，第一列为文档编号，第二列为评分，便于以后对于不同的文档抽取结果的准确度进行统计评价，从而改进系统。

A	B
71	5
111	5
147	5
64	4

工程文件说明

文件/文件夹	说明
docs_utf8	原始文档文件夹，所有文档已改为 UTF-8 编码并从 1 到 500 标号。
clean_data	包含所有文档进行分词、去停用词、去英文后的纯净文本
cocoNLP、stanfordcorenlp、stanford-corenlp-	进行信息抽取需要用到的开源代码库，我们对其中的源码进行了修改

4.2.2	
img	图形化界面使用的所有图片文件夹
extract_widget.ui	信息抽取小窗口的用户界面
GUI.ui	程序主界面
index.txt	倒排索引文件
source.qrc	图形化界面的资源文件
stop_words.txt	停用词表
word_split.py	对文档和用户输入进行分词、去停用词等处理
build_index.py	读文档，构造空间向量，生成倒排索引文件，相关度计算
extract.py	信息抽取
output.py	输出索引结果中的文档头和主要匹配内容
search_GUI.py	含 UI 界面的窗口类以及所有槽函数，窗口 UI 的构建，程序的主要逻辑
source_rc.py	.qrc 资源文件经工具处理后转化的.py 文件