文档编号：

河北省信息技术手册检测报告

详细设计说明书

石家庄铁道大学软件工程

2019年01月

**变更历史**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 版本号 | 版本日期 | 修订人 | 核准人 | 备注 |
| v1.0 | 2019.4.07 | 王莉 | 王莉 |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

目录

[1 引言 2](#_Toc535440466)

[1.1 编写目的 2](#_Toc535440467)

[1.2 适用对象及范围 2](#_Toc535440468)

[1.3 名词解释 2](#_Toc535440469)

[2 总体设计 3](#_Toc535440470)

[2.1 需求规定 3](#_Toc535440471)

[2.1.1 对功能的规定 3](#_Toc535440472)

[2.1.2 输入/输出要求 3](#_Toc535440473)

[2.1.3 故障处理要求 3](#_Toc535440474)

[2.2 基础服务 4](#_Toc535440475)

[3 系统功能设计 5](#_Toc535440476)

[3.1 热词查看功能 5](#_Toc535440477)

[3.1.1 实现设计 5](#_Toc535440478)

[3.1.2 信息搜索模块 5](#_Toc535440479)

[3.1.3 信息查看模块 12](#_Toc535440480)

[3.1.4 信息分类模块 20](#_Toc535440481)

[3.1.5 信息下载模块 25](#_Toc535440482)

[4.数据库](#_Toc535440483)[设计 29](#_Toc535440483)

# 引言

## 编写目的

编写详细设计的目的是：在概要设计的基础上，完成系统划分模块及明确模块之间的关系，同时给出相应的文档界面设计。

## 适用对象及范围

本设计说明书适用于参加本项目的所有管理人员、开发人员和维护人员。包括：项目经理、系统分析员、高级程序员、代码程序员、数据库程序员、测试人员、培训人员、实施人员、技术支持人员、项目监理人员、系统管理用户和最终使用用户。

## 名词解释

**Pdf文档：**[PDF](https://baike.baidu.com/item/PDF" \t "https://baike.baidu.com/item/PDF%E6%96%87%E6%A1%A3/_blank)全称Portable Document Format，是便携文档格式的简称，同时也是该格式的扩展名。DF拥有绝对空前超强的跨平台功能（适用与MAC/WINXX/UNIX/LINUX/OS2等所有平台），不依赖任何的系统的语言、字体和显示模式；和HTML一样拥有[超文本链接](https://baike.baidu.com/item/%E8%B6%85%E6%96%87%E6%9C%AC%E9%93%BE%E6%8E%A5" \t "https://baike.baidu.com/item/PDF%E6%96%87%E6%A1%A3/_blank)，可导航阅读；印刷[排版](https://baike.baidu.com/item/%E6%8E%92%E7%89%88" \t "https://baike.baidu.com/item/PDF%E6%96%87%E6%A1%A3/_blank)功能，可支持电子出版的各种要求；并得到大量第三方软件公司的支持，拥有多种浏览操作方式；并且其他传统的文档格式体积更小，更方便在Internet上传输。

**论文重合度：**论文重合度是[学术论文](https://baike.baidu.com/item/%E5%AD%A6%E6%9C%AF%E8%AE%BA%E6%96%87/4023535" \t "https://baike.baidu.com/item/%E8%AE%BA%E6%96%87%E9%87%8D%E5%90%88%E5%BA%A6/_blank)在内容上的相似或重合程度，通常用来检测[论文](https://baike.baidu.com/item/%E8%AE%BA%E6%96%87/149478" \t "https://baike.baidu.com/item/%E8%AE%BA%E6%96%87%E9%87%8D%E5%90%88%E5%BA%A6/_blank)是否[抄袭](https://baike.baidu.com/item/%E6%8A%84%E8%A2%AD/4655968" \t "https://baike.baidu.com/item/%E8%AE%BA%E6%96%87%E9%87%8D%E5%90%88%E5%BA%A6/_blank)，一般高校将重合度30%以上定为抄袭的文章，即论文审核不通过。

**关键词：**关键词源于英文“[keywords](https://baike.baidu.com/item/keywords/8284218" \t "https://baike.baidu.com/item/%E5%85%B3%E9%94%AE%E8%AF%8D/_blank)”，特指单个媒体在制作使用[索引](https://baike.baidu.com/item/%E7%B4%A2%E5%BC%95/5716853" \t "https://baike.baidu.com/item/%E5%85%B3%E9%94%AE%E8%AF%8D/_blank)时，所用到的词汇。是图书馆学中的[词汇](https://baike.baidu.com/item/%E8%AF%8D%E6%B1%87/1890163" \t "https://baike.baidu.com/item/%E5%85%B3%E9%94%AE%E8%AF%8D/_blank)。[关键词](https://baike.baidu.com/item/%E5%85%B3%E9%94%AE%E8%AF%8D/254446" \t "https://baike.baidu.com/item/%E5%85%B3%E9%94%AE%E8%AF%8D/_blank)搜索是[网络搜索](https://baike.baidu.com/item/%E7%BD%91%E7%BB%9C%E6%90%9C%E7%B4%A2/7161324" \t "https://baike.baidu.com/item/%E5%85%B3%E9%94%AE%E8%AF%8D/_blank)索引主要方法之一，就是访问者希望了解的产品、服务和公司等的具体名称用语。

**摘要：**摘要又称概要、内容提要。摘要是以提供文献内容梗概为目的，不加评论和补充解释，简明、确切地记述文献重要内容的短文。其基本要素包括研究目的、方法、结果和结论。具体地讲就是研究工作的主要对象和范围，采用的手段和方法，得出的结果和重要的结论，有时也包括具有情报价值的其它重要的信息。

# 总体设计

## 需求规定

### 对功能的规定

#### 自动分类功能

从数据库中读取内容部分，能够自动根据内容确定该类别的内容，并且写入到数据库，在文档中展示出每条信息的分类。

#### 自动提取关键词和摘要功能

从数据库中读取内容部分，自动提取该内容的关键词和摘要，并且写入到数据库，在文档中展示出每条信息的关键词和摘要。

#### 自动查重功能

从数据库中读取内容部分，根据关键词查取百度百科的解释意思，对比查出抄袭的部分。饭后将抄袭的部分写入到数据库。在文档中没有抄袭的部分用黑色字体析出，抄袭的部分用红色字体写出。计算出信息手册所有信息的相似比和相似片段数在文档中展示出来。

### 输入/输出要求

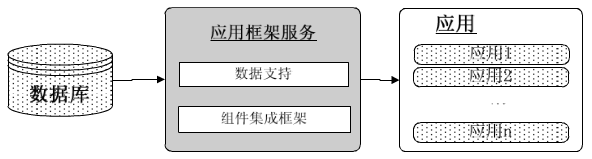
1. 输入为河北省信息技术手册，后缀名为.txt。里面内容包括信息技术的标题和内容。
2. 输出为河北省信息技术手册检测报告，后缀名为.pdf。里面包括信息技术的标题，内容，类别，关键词，摘要。所有信息技术的相似比和抄袭片段个数。

### 故障处理要求

1. 软件支撑环境运行错误。
2. IIS运行环境错误，此时平台无法正确运行，管理员参考相关产品手册。
3. 数据库运行错误：系统无法使用，平台将提供错误提示页面，并记录日志。
4. 应用系统错误：由本应用系统提供错误提示页面，程序运行错误记录在平台日志中，数据运行错误记录在应用系统日志中。

## 基础服务

1. 关系数据库



1. **数据库应用**

在本平台之上的应用不是直接访问数据库，而是通过应用框架服务层提供的数据库访问组件来访问数据库。

2.应用服务器

应用服务器为应用的业务逻辑提供了一个运行环境。它可以提供企业级的组件支持环境，支持网络环境下应用软件的快速开发和部署。应用组件在应用服务器的服务上运行，并利用网络基础架构提供的目录和安全服务。这些组件还可以利用数据库、事务处理和群集等设施。

# 系统功能设计

## 自动分类功能

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **序号** | **系统模块** | **功能** | **功能描述** | **操作** |
| 1 | 自动分类 | 自动分类功能 | 从数据库中读取内容部分，能够自动根据内容确定该类别的内容，并且写入到数据库，在文档中展示出每条信息的分类。 | 将河北省信息技术手册读入到数据库。 |
| 2 | 自动提取关键词和摘要 | 自动提取关键词和摘要功能 | 从数据库中读取内容部分，自动提取该内容的关键词和摘要，并且写入到数据库，在文档中展示出每条信息的关键词和摘要。 | 将河北省信息技术手册读入到数据库。 |
| 3 | 自动查重 | 自动查自动查重功能 | 从数据库中读取内容部分，根据关键词查取百度百科的解释意思，对比查出抄袭的部分。饭后将抄袭的部分写入到数据库。在文档中没有抄袭的部分用黑色字体析出，抄袭的部分用红色字体写出。计算出信息手册所有信息的相似比和相似片段数在文档中展示出来。 | 将河北省信息技术手册读入到数据库。 |

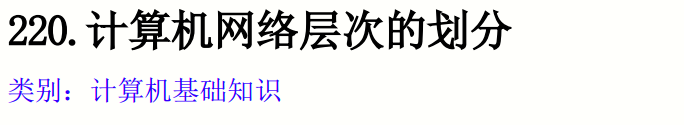
### 实现设计

通过对业务功能实现时序的分析，所有业务功能的实现处理过程分为3大模块：自动分类，自动提取关键词和摘要，自动查重

### 自动分类模块

#### 功能设计

从数据库中读取内容部分，能够自动根据内容确定该类别的内容，并且写入到数据库，在文档中展示出每条信息的分类。文档展示如图所示：



#### 类设计

|  |  |
| --- | --- |
| 类名 | 描述 |
| citiao | 信息技术，包含热词的名字、含义、类型。 |
| Copy | 抄袭片段，包括抄袭的片段，抄袭的网址 |

#### 代码设计

* 代码设计

*#!/usr/bin/env python  
# coding=utf-8***import** sys  
**import** jieba  
**from** sklearn.pipeline **import** Pipeline  
**from** sklearn.feature\_extraction.text **import** CountVectorizer, TfidfTransformer  
**from** sklearn.svm **import** LinearSVC  
**from** sklearn.multiclass **import** OneVsRestClassifier  
**from** sklearn.preprocessing **import** MultiLabelBinarizer  
**import** pymysql  
**import** pandas **as** pd  
**import** re  
**import** numpy **as** np  
  
  
**def** jieba\_tokenizer(x): **return** jieba.cut(x, cut\_all=**True**)  
  
  
**def** partition(x): **return** x  
  
  
**def** filter\_html(s):  
 d = re.compile(**r'<[^>]+>'**, re.S)  
 s = d.sub(**''**, s)  
 **return** s  
  
  
**def** gbk\_utf8(s):  
 s = s.decode(**'gbk'**, **"ignore"**).encode(**'utf8'**)  
 **return** s  
**def** write\_sql(id,classs):  
 db = pymysql.Connection(host=**"localhost"**, port=3307, user=**"root"**, password=**"root"**, database=**"jiansuo"**,  
 charset=**"utf8"**)  
 cursor = db.cursor()  
 sql = **"update yuan set classs='"** + classs + **"' where id="** + str(id)  
 **try**:  
 cursor.execute(sql)  
 db.commit()  
 **except**:  
 db.commit()  
 print(**"出错了！"**)  
 db.close()  
  
  
*# 链接mysql数据库*conn = pymysql.Connection(host=**"localhost"**,port=3307,user=**"root"**, password=**"root"**,database=**"jiansuo"**,charset=**"utf8"**)  
cursor = conn.cursor()  
cursor=conn.cursor()  
  
*# 训练数据样本*data\_ret = pd.DataFrame()  
  
sql = **"SELECT id, name,yuan\_copy.class,yuan\_copy.explain FROM yuan\_copy "***# print sql*cursor.execute(sql)  
  
txt\_ret = []  
*#class\_ret = [["信息化"],["大数据"],["云计算"],["区块链"],["智慧城市"],["工业互联网"],["信息安全"],["操作系统"],["计算机"],["法律法规"],["信息化战略"]]*class\_ret=[]  
id\_ret = []  
**for** row **in** cursor.fetchall():  
 content = filter\_html(row[3])  
 txt\_ret.append(content)  
 class\_s = row[2]  
 class\_l = class\_s.split(**" "**)  
 class\_ret.append(class\_l)  
 id\_ret.append(row[0])  
  
txt\_ret = txt\_ret  
  
X\_train = txt\_ret  
print(class\_ret)  
Y\_train = class\_ret  
  
classifier = Pipeline([  
 (**'counter'**, CountVectorizer(tokenizer=jieba\_tokenizer)),  
 (**'tfidf'**, TfidfTransformer()),  
 (**'clf'**, OneVsRestClassifier(LinearSVC())),  
])  
mlb = MultiLabelBinarizer()  
Y\_train = mlb.fit\_transform(Y\_train)  
classifier.fit(X\_train, Y\_train)  
print(classifier.score(X\_train,Y\_train))  
*# 测试数据*test\_txt\_set = []  
sql = **"SELECT id, name,keywords,yuan.explain FROM yuan "**cursor.execute(sql)  
test\_id\_ret = []  
  
**for** row **in** cursor.fetchall():  
 test\_txt\_set.append(filter\_html(row[3]))  
 test\_id\_ret.append(row[0])  
X\_test = test\_txt\_set  
  
prediction = classifier.predict(X\_test)  
  
result = mlb.inverse\_transform(prediction)  
*# 展示结果***for** i, label1 **in** enumerate(result):  
 classstr = **''  
 for** j, label2 **in** enumerate(label1):  
 classstr += str(label2) + **""** print(**"ID:"** + str(test\_id\_ret[i]) + **" =>class:"** + classstr)  
 write\_sql(test\_id\_ret[i],classstr)

* Citiao信息技术标题、内容、类型、关键词、摘要、id

**public** **class** citiao {

**private** **int** id;

**private** String name;

**private** String explain;

**private** String keywords;

**private** String abstracts;

**private** String classs;

**public** String getClasss() {

**return** classs;

}

**public** **void** setClasss(String classs) {

**this**.classs = classs;

}

**public** String getKeywords() {

**return** keywords;

}

**public** **void** setKeywords(String keywords) {

**this**.keywords = keywords;

}

**public** String getAbstracts() {

**return** abstracts;

}

**public** **void** setAbstracts(String abstracts) {

**this**.abstracts = abstracts;

}

**public** **int** getId() {

**return** id;

}

**public** **void** setId(**int** id) {

**this**.id = id;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** String getExplain() {

**return** explain;

}

**public** **void** setExplain(String explain) {

**this**.explain = explain;

}

}

* copy抄袭片段、起始位置、抄袭网址

**public** **class** copy {

**private** **int** id;

**private** **int** yuan\_id;

**private** String copy\_field;

**private** **int** start\_position;//字符串开始位置

**private** **int** last\_position;//字符串结尾位置

**private** String url;

**public** **int** getStart\_position() {

**return** start\_position;

}

**public** **void** setStart\_position(**int** start\_position) {

**this**.start\_position = start\_position;

}

**public** **int** getLast\_position() {

**return** last\_position;

}

**public** **void** setLast\_position(**int** last\_position) {

**this**.last\_position = last\_position;

}

**public** **int** getId() {

**return** id;

}

**public** **void** setId(**int** id) {

**this**.id = id;

}

**public** **int** getYuan\_id() {

**return** yuan\_id;

}

**public** **void** setYuan\_id(**int** yuan\_id) {

**this**.yuan\_id = yuan\_id;

}

**public** String getCopy\_field() {

**return** copy\_field;

}

**public** **void** setCopy\_field(String copy\_field) {

**this**.copy\_field = copy\_field;

}

**public** String getUrl() {

**return** url;

}

**public** **void** setUrl(String url) {

**this**.url = url;

}

}

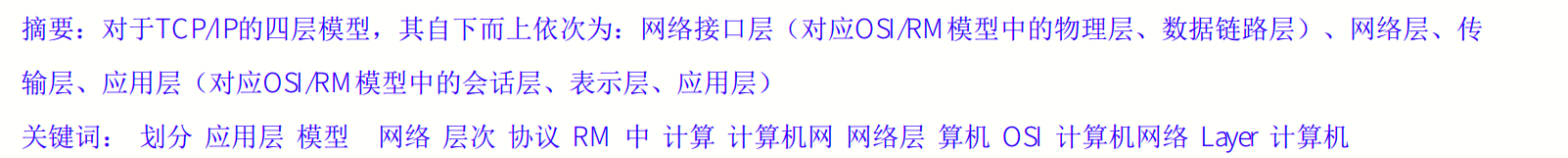
* 读写数据库

**import** pymysql  
id=[]  
name=[]  
explain=[]  
db=pymysql.Connection(host=**"localhost"**,port=3307,user=**"root"**, password=**"root"**,database=**"jiansuo"**,charset=**"utf8"**)  
cursor=db.cursor()  
sql=**"select** *\** **from yuan"  
try**:  
 cursor.execute(sql)  
 results=cursor.fetchall()  
 **for** row **in** results:  
 id.append(row[0])  
 name.append(row[1])  
 explain.append(row[2])  
**except**:  
 print(**"出错了！"**)  
db.close()

**import** pymysql  
**def** write(id,keywords,abstract):  
 db=pymysql.Connection(host=**"localhost"**,port=3307,user=**"root"**, password=**"root"**,database=**"jiansuo"**,charset=**"utf8"**)  
 cursor=db.cursor()  
 sql=**"update yuan set keywords='"**+keywords+**"',abstract='"**+abstract+**"' where id="**+str(id)  
 **try**:  
 cursor.execute(sql)  
 db.commit()  
 **except**:  
 db.commit()  
 print(**"出错了！"**)  
 db.close()

### 自动提取关键词和摘要模块

#### 功能设计

根据内容自动提取关键词和摘要，展示在文档中。界面设计如图所示：

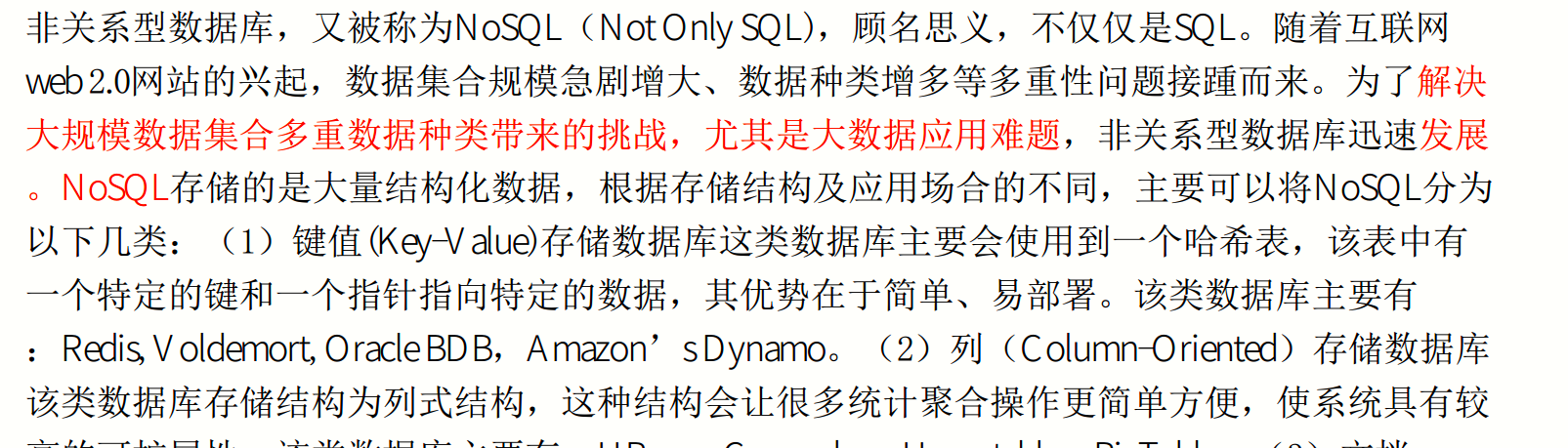
#### 代码设计

*# -\*- coding: utf-8 -\*-***import** jieba,copy,re,codecs  
**from** collections **import** Counter  
**from** sql\_read **import** id  
**from** sql\_read **import** name  
**from** sql\_read **import** explain  
**from** sql\_write **import** write  
  
*# summary = pyhanlp.HanLP.extractSummary(text, 3)  
# print(summary)***class** Summary():  
 *#\*\*\*\* 切分句子 \*\*\*\*\*\*\*\*\*\*\*\** **def** cutSentence(self,text):  
 sents = []  
 text = re.sub(**r'\n+'**,**'。'**,text) *# 换行改成句号（标题段无句号的情况）* text = text.replace(**'。。'**,**'。'**) *# 删除多余的句号* text = text.replace(**'？。'**,**'。'**) *#* text = text.replace(**'！。'**,**'。'**) *# 删除多余的句号* sentences = re.split(**r'。|！|？|】|；'**,text) *# 分句* biaoji=sentences  
 sentences = sentences[:-1] *# 删除最后一个句号后面的空句* **for** sent **in** sentences:  
 len\_sent = len(sent)  
 **if** len\_sent < 4: *# 删除换行符、一个字符等* **continue** sent = sent.strip(**'　 '**)  
 sent = sent.lstrip(**'【'**)  
 sents.append(sent)  
 **if** len(sents)==0:  
 sents=biaoji  
 **return** sents  
 *#\*\*\*\* 提取特征词 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\** **def** getKeywords(self,title,sentences,n=10):  
 words = []  
 *#\*\*\*\* 分词，获取词汇列表 \*\*\*\*\*  
 # split\_result = pseg.cut(text)* **for** sentence **in** sentences:  
 split\_result = jieba.cut(sentence)  
 **for** i **in** split\_result:  
 words.append(i)  
 *#\*\*\*\* 统计词频TF \*\*\*\*\** c = Counter(words) *# 词典  
 #\*\*\*\* 去除停用词(为了提高效率，该步骤放到统计词频之后)* self.delStopwords(c)  
 *#\*\*\*\* 标题中提取特征 \*\*\*\*\*\*\*\*\** words\_title = [word **for** word **in** jieba.cut(title,cut\_all=**True**)]  
 self.delStopwords(words\_title)  
 *#\*\*\*\* 获取topN \*\*\*\*\*\*\*\*\*\*\*\** topN = c.most\_common(n)  
  
 words\_topN = [i[0] **for** i **in** topN **if** i[1]>1] *#在topN中排除出现次数少于2次的词* words\_topN = list(set(words\_topN)|set(words\_title)) *# 正文关键词与标题关键词取并集* **return** words\_topN  
 *#\*\*\*\* 去除停用词 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\** **def** delStopwords(self,dict):  
 sw\_file = codecs.open(**'stopwords.txt'**,encoding=**'utf-8'**)  
 stop\_words = []  
 **for** line **in** sw\_file.readlines():  
 stop\_words.append(line.strip())  
 *#\*\*\*\*\* 输入参数为list \*\*\*\*\*\*\*\*\*\*\*\*\*  
 # if type(dict) is types.ListType:* **if** type(dict) **is** list:  
 words = dict  
 **for** word **in** words:  
 **if** word **in** stop\_words:  
 words.remove(word)  
 *#\*\*\*\*\* 输入参数type为 <class 'collections.Counter'> \*\*\*\*\** **else**:  
 words = copy.deepcopy(list(dict.keys()))  
 **for** word **in** words:  
 **if** word **in** stop\_words:  
 **del** dict[word]  
 **return** words  
 *#\*\*\*\* 提取topN句子 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\** **def** getTopNSentences(self,sentences,keywords,n=3):  
 sents\_score = {}  
 len\_sentences = len(sentences)  
 *#\*\*\*\* 初始化句子重要性得分，并计算句子平均长度* len\_avg = 0  
 len\_min = len(sentences[0])  
 len\_max = len(sentences[0])  
 **for** sent **in** sentences:  
 sents\_score[sent] = 0  
 l = len(sent)  
 len\_avg += l  
 **if** len\_min > l:  
 len\_min = l  
 **if** len\_max < l:  
 len\_max = l  
 len\_avg = len\_avg / len\_sentences  
 *# print(len\_min,len\_avg,len\_max)  
 #\*\*\*\* 计算句子权重得分 \*\*\*\*\*\*\*\*\*\** **for** sent **in** sentences:  
 *#\*\*\*\* 不考虑句长在指定范围外的句子 \*\*\*\*\*\** l = len(sent)  
 **if** l < (len\_min + len\_avg) / 2 **or** l > (3 \* len\_max - 2 \* len\_avg) / 4:  
 **continue** words = []  
 sent\_words = jieba.cut(sent) *# <generator object cut at 0x11B38120>* **for** i **in** sent\_words:  
 words.append(i)  
 keywords\_cnt = 0  
 len\_sent = len(words)  
 **if** len\_sent == 0:  
 **continue  
 for** word **in** words:  
 **if** word **in** keywords:  
 keywords\_cnt += 1  
 score = keywords\_cnt \* keywords\_cnt \* 1.0 / len\_sent  
 sents\_score[sent] = score  
 **if** sentences.index(sent) == 0:*# 提高首句权重* sents\_score[sent] = 2 \* score  
 *#\*\*\*\* 排序 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\** dict\_list = sorted(sents\_score.items(),key=**lambda** x:x[1],reverse=**True**)  
 *# print(dict\_list)  
 #\*\*\*\* 返回topN \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\** sents\_topN = []  
 **for** i **in** dict\_list[:n]:  
 sents\_topN.append(i[0])  
 *# print i[0],i[1]* sents\_topN = list(set(sents\_topN))  
 *#\*\*\*\* 按比例提取 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\** **if** len\_sentences <= 5:  
 sents\_topN = sents\_topN[:1]  
 **elif** len\_sentences < 9:  
 sents\_topN = sents\_topN[:2]  
 **return** sents\_topN  
 *#\*\*\*\* 恢复topN句子在文中的相对顺序 \*\*\*\*\*\*\*\*\** **def** sents\_sort(self,sents\_topN,sentences):  
 keysents = []  
 **for** sent **in** sentences:  
 **if** sent **in** sents\_topN **and** sent **not in** keysents:  
 keysents.append(sent)  
 keysents = self.post\_processing(keysents)  
 **return** keysents  
 **def** post\_processing(self,keysents):  
 *#\*\*\*\* 删除不完整句子中的详细部分 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\** detail\_tags = [**'，一是'**,**'：一是'**,**'，第一，'**,**'：第一，'**,**'，首先，'**,**'；首先，'**]  
 **for** i **in** keysents:  
 **for** tag **in** detail\_tags:  
 index = i.find(tag)  
 **if** index != -1:  
 keysents[keysents.index(i)] = i[:index]  
 *#\*\*\*\* 删除编号 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\** **for** i **in** keysents:  
 *# print(i)* regex = re.compile(**r'^一、|^二、|^三、|^三、|^四、|^五、|^六、|^七、|^八、|^九、|^十、|^\d{1,2}、|^\d{1,2} '**)  
 result = re.findall(regex,i)  
 **if** result:  
 keysents[keysents.index(i)] = re.sub(regex,**''**,i)  
 *#\*\*\*\* 删除备注性质的句子 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\** **for** i **in** keysents:  
 regex = re.compile(**r'^注\d\*：'**)  
 result = re.findall(regex,i)  
 **if** result:  
 keysents.remove(i)  
 *#\*\*\*\* 删除句首括号中的内容 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\** **for** i **in** keysents:  
 regex = re.compile(**r'^.∗.∗'**)  
 result = re.findall(regex,i)  
 **if** result:  
 keysents[keysents.index(i)] = re.sub(regex,**''**,i)  
 *#\*\*\*\* 删除来源(空格前的部分) \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\** **for** i **in** keysents:  
 regex = re.compile(**r'^.{1,20} '**)  
 result = re.findall(regex,i)  
 **if** result:  
 keysents[keysents.index(i)] = re.sub(regex,**''**,i)  
 *#\*\*\*\* 删除引号部分（如：银行间债市小幅下跌，见下图：） \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\** **for** i **in** keysents:  
 regex = re.compile(**r'，[^，]+：$'**)  
 result = re.findall(regex,i)  
 **if** result:  
 keysents[keysents.index(i)] = re.sub(regex,**''**,i)  
 **return** keysents  
 **def** main(self,title,text,i):  
 sentences = self.cutSentence(text)  
 keywords = self.getKeywords(title, sentences, n=8)  
 sents\_topN = self.getTopNSentences(sentences, keywords, n=3)  
 keysents = self.sents\_sort(sents\_topN, sentences)  
 keys=**' '**.join(keywords)  
 print(keys)  
 print(keysents[0])  
 write(i,keys,keysents[0])  
  
 **return** keysents  
  
**if** \_\_name\_\_==**'\_\_main\_\_'**:  
 j=0  
 **for** i **in** id:  
 print(i)  
 summary = Summary()  
 summary.main(name[j],explain[j],i)  
 j=j+1

### 自动查重模块

#### 功能设计

对比内容和关键词的百度百科解释对比查重。界面设计如图：



1. **抄袭片段界面**



1. **检测结论界面**

#### 代码设计

* 关键词爬虫以及内容对比

import java.util.ArrayList;

import java.util.Date;

import java.util.List;

import org.assertj.core.util.Collections;

import org.jsoup.Jsoup;

import org.jsoup.nodes.Document;

import us.codecraft.webmagic.Page;

import us.codecraft.webmagic.Site;

import us.codecraft.webmagic.Spider;

import us.codecraft.webmagic.processor.PageProcessor;

public class chachong implements PageProcessor {

static String chaxun = null;

static String url = null;

public static void main(String[] args) {

List<citiao> cs = DBUtil.load\_all();

for (citiao c : cs) {

String explain=c.getExplain();

String[] kk =c.getKeywords().split(" ");

List<String> ks=new ArrayList<String>();

for (int i=0; i<kk.length; i++) {

if (!kk[i].equals("")) {

ks.add(kk[i]);

}

}

for (String k:ks) {//将百度百科爬取到的意思和原文解释进行对比

shuru(k);

if(chaxun==null) {

continue;

}

List<String> chongfus = duibi(explain, chaxun);

List<String> chaoxi = new ArrayList<String>();

int count = 0;// 抄袭的总字数

if (chongfus != null) {// 有重复的字符

for (String chongfu : chongfus) {

if (chongfu.length() > 7) {// 知网规定重复超过7个算是抄袭

chaoxi.add(chongfu);

count = count + chongfu.length();

}

}

}

if (count > 0) {

System.out.println("抄袭句子");

for (String cx : chaoxi) {

copy cop=new copy();

cop.setYuan\_id(c.getId());

cop.setCopy\_field(cx);

cop.setUrl(url);

System.out.println(cx);

DBUtil.add\_copy(cop);

// 写入到数据库

}

}

}

}

}

public static List<String> duibi(String explain, String chaxun) {

List<String> chongfus = new ArrayList<String>();

String chongfu = "";

boolean lianxu = false;

temp p=null;

for (int i = 0; i < explain.length(); i++) {

for (int j = 0; j < chaxun.length(); j++) {

if (i >= explain.length() || j >= chaxun.length()) {

break;

}

if (explain.charAt(i) == chaxun.charAt(j)) {//如果有重复的字符

if ((lianxu == false) && (!chongfu.equals(""))) {// 上一句重复的话

chongfus.add(chongfu);

chongfu = "";

}

if(p==null) {//没有存储进去字符,开始存储di'yi'g

p=new temp();

p.setStart\_position(i);

}

p.setLast\_position(i);

chongfu = chongfu + explain.charAt(i);

lianxu = true;

i++;

} else {

lianxu = false;

}

}

}

chongfus.add(chongfu);

return chongfus;

}

@Override

public Site getSite() {

return Site.me().setSleepTime(1000).setRetryTimes(10);

}

private static String selectDocumentText(String htmlText) {

Document doc = Jsoup.parse(htmlText);

String select = doc.text();

return select;

}

@Override

public void process(Page page) {

// 定义如何抽取页面信息

List<String> htmls = page.getHtml().xpath("//div[@class='lemma-summary']/html()").all();

for (String html : htmls) {

chaxun = selectDocumentText(html);

}

}

public static void shuru(String word) {

long startTime, endTime;

startTime = new Date().getTime();

Spider create = Spider.create(new chachong());

String strr = "https://baike.baidu.com/item/" + word;

url = strr;

create.addUrl(strr).thread(1).run();

endTime = new Date().getTime();

System.out.println("用时为：" + (endTime - startTime) / 1000 + "s");

}

}

* 查找抄袭片段在原文中中的位置

**import** java.util.List;

**public** **class** findPosition {

**public** **static** **void** main(String[] args) {

List<copy> cops = DBUtil.*load\_all2*();

**for** (copy cop : cops) {

String explain = DBUtil.*load\_expain*(cop.getYuan\_id());

**int** start=*strStr*(explain,cop.getCopy\_field());

System.***out***.println(cop.getId()+":"+explain.charAt(start)+"---"+explain.charAt(start+cop.getCopy\_field().length()-1));

//System.out.println(cop.getId()+":"+start);

DBUtil.*update\_copy*(start,start+cop.getCopy\_field().length()-1 , cop.getId());

}

}

**public** **static** Integer strStr(String str, String target) {

str=str+"#";

**int** length = str.length();

**int** targetLength = target.length();

**if** (targetLength > length) {

**return** -1;

}

**if** (targetLength == 0) {

**return** 0;

}

**int** num = length - targetLength;

**for** (**int** i = 0; i < num; i++) {

**if** (target.charAt(0) != str.charAt(i)) {

**while** (++i <= num && str.charAt(i) != target.charAt(0))

;

}

**if** (i < num) {

**int** j = i + 1;

**int** end = i + targetLength;

**for** (**int** k = 1; j < end && str.charAt(j) == target.charAt(k); j++, k++)

;

**if** (j == end) {

**return** i;

}

}

}

**return** -1;

}

}

* 对数据库进行操作

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.util.ArrayList;

import java.util.List;

public class DBUtil {

public static Connection getConnection() {

try {

Class.forName("com.mysql.jdbc.Driver").newInstance();

} catch (InstantiationException e1) {

// TODO Auto-generated catch block

e1.printStackTrace();

} catch (IllegalAccessException e1) {

// TODO Auto-generated catch block

e1.printStackTrace();

} catch (ClassNotFoundException e1) {

// TODO Auto-generated catch block

e1.printStackTrace();

}

String user = "root";

String password = "root";

String url = "jdbc:mysql://localhost:3307/jiansuo";

Connection connection = null;

try {

connection = DriverManager.getConnection(url,user,password);

} catch (SQLException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

return connection;

}

public static void update\_copy(int start,int last,int id) {

Connection connection = DBUtil.getConnection();

String sql = "update copy set start\_position="+start+",last\_position="+last+" where id="+id;

PreparedStatement preparedStatement = null;

try {

preparedStatement = connection.prepareStatement(sql);

preparedStatement.executeUpdate();

} catch (SQLException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}finally {

DBUtil.close(preparedStatement);

DBUtil.close(connection);

}

}

public static void add(citiao c) {//增

//获得链接对象

Connection connection = DBUtil.getConnection();

//准备sql语句

//创建语句传输对象

PreparedStatement preparedStatement = null;

ResultSet resultSet = null;

try {

String sql = "insert into yuan value ("+c.getId()+",'"+c.getName()+"','"+c.getExplain()+"',null,null)";

preparedStatement = connection.prepareStatement(sql);

preparedStatement.executeUpdate();

} catch (SQLException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}finally {

//关闭资源

DBUtil.close(resultSet);

DBUtil.close(preparedStatement);

DBUtil.close(connection);

}

}

public static void add\_copy(copy c) {//增

//获得链接对象

Connection connection = DBUtil.getConnection();

//准备sql语句

//创建语句传输对象

PreparedStatement preparedStatement = null;

ResultSet resultSet = null;

try {

String sql = "insert into copy value (null,"+c.getYuan\_id()+",'"+c.getCopy\_field()+"','"+c.getUrl()+"')";

preparedStatement = connection.prepareStatement(sql);

preparedStatement.executeUpdate();

} catch (SQLException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}finally {

//关闭资源

DBUtil.close(resultSet);

DBUtil.close(preparedStatement);

DBUtil.close(connection);

}

}

public static List<citiao> load\_all() {//查

Connection connection = DBUtil.getConnection();

//准备sql语句

String sql = "select \* from yuan";

//创建语句传输对象

PreparedStatement preparedStatement = null;

ResultSet resultSet = null;

List<citiao> cs =new ArrayList<citiao>();

try {

preparedStatement = connection.prepareStatement(sql);

resultSet = preparedStatement.executeQuery();

while(resultSet.next()) {

citiao c = new citiao();

c.setId(resultSet.getInt("id"));

c.setName(resultSet.getString("name"));

c.setExplain(resultSet.getString("explain"));

c.setKeywords(resultSet.getString("keywords"));

c.setAbstracts(resultSet.getString("abstract"));

c.setClasss(resultSet.getString("classs"));

cs.add(c);

}

} catch (SQLException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}finally {

DBUtil.close(resultSet);

DBUtil.close(preparedStatement);

DBUtil.close(connection);

}

return cs;

}

public static String load\_expain(int id) {//查

Connection connection = DBUtil.getConnection();

//准备sql语句

String sql = "select \* from yuan where id="+id;

//创建语句传输对象

PreparedStatement preparedStatement = null;

ResultSet resultSet = null;

String expain="";

try {

preparedStatement = connection.prepareStatement(sql);

resultSet = preparedStatement.executeQuery();

while(resultSet.next()) {

expain=resultSet.getString("explain");

}

} catch (SQLException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}finally {

DBUtil.close(resultSet);

DBUtil.close(preparedStatement);

DBUtil.close(connection);

}

return expain;

}

public static List<copy> load\_all2() {//查

Connection connection = DBUtil.getConnection();

//准备sql语句

String sql = "select \* from copy";

//创建语句传输对象

PreparedStatement preparedStatement = null;

ResultSet resultSet = null;

List<copy> cops =new ArrayList<copy>();

try {

preparedStatement = connection.prepareStatement(sql);

resultSet = preparedStatement.executeQuery();

while(resultSet.next()) {

copy c = new copy();

c.setId(resultSet.getInt("id"));

c.setYuan\_id(resultSet.getInt("yuan\_id"));

c.setCopy\_field(resultSet.getString("copy\_field"));

cops.add(c);

}

} catch (SQLException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}finally {

DBUtil.close(resultSet);

DBUtil.close(preparedStatement);

DBUtil.close(connection);

}

return cops;

}

public static List<copy> load\_id(int yuan\_id) {//查

Connection connection = DBUtil.getConnection();

//准备sql语句

String sql = "select distinct start\_position,last\_position from copy where yuan\_id="+yuan\_id+" ORDER BY start\_position";

//创建语句传输对象

PreparedStatement preparedStatement = null;

ResultSet resultSet = null;

List<copy> cops =new ArrayList<copy>();

try {

preparedStatement = connection.prepareStatement(sql);

resultSet = preparedStatement.executeQuery();

while(resultSet.next()) {

copy c = new copy();

c.setStart\_position(resultSet.getInt("start\_position"));

c.setLast\_position(resultSet.getInt("last\_position"));

cops.add(c);

}

} catch (SQLException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}finally {

DBUtil.close(resultSet);

DBUtil.close(preparedStatement);

DBUtil.close(connection);

}

return cops;

}

public static void close(Connection connection ) {

try {

if (connection != null) {

connection.close();

}

} catch (SQLException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

public static void close(PreparedStatement preparedStatement ) {

try {

if (preparedStatement != null) {

preparedStatement.close();

}

} catch (SQLException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

public static void close(ResultSet resultSet ) {

try {

if (resultSet != null) {

resultSet.close();

}

} catch (SQLException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

}

* 写入文档

import java.awt.Color;

import java.io.FileOutputStream;

import java.text.DecimalFormat;

import java.util.List;

import com.lowagie.text.Chapter;

import com.lowagie.text.Chunk;

import com.lowagie.text.Document;

import com.lowagie.text.Font;

import com.lowagie.text.PageSize;

import com.lowagie.text.Paragraph;

import com.lowagie.text.Phrase;

import com.lowagie.text.pdf.BaseFont;

import com.lowagie.text.pdf.PdfWriter;

public class writeFile {

private static int normal = Font.NORMAL; // 正常字体

private static Color black = new Color(0, 0, 0); // 黑色

private static Color red = new Color(255, 0, 0); // 红色

private int bold = Font.BOLD; // 粗体

private Color blue = new Color(0, 0, 255); // 蓝色

private float setting = 20; // 首行缩进参数

static int count=0;//用红色标注的字体个数

static int cop\_count=0;//用红色标注的片段个数

static int sum=0;//总字数

public static void main(String[] args) throws Exception {

writeFile pdfDemo = new writeFile();

pdfDemo.writePdf("F:\\大数据\\大数据技术应用大作业-1\\信息技术手册检测报告.pdf");

System.out.println("写入完成！");

}

public static Document createDoc(String filename) throws Exception {

Document document = new Document(PageSize.A4, 50, 50, 50, 50);

PdfWriter.getInstance(document, new FileOutputStream(filename));

return document;

}

public void writePdf(String filename) throws Exception {

Document document = createDoc(filename); // 打开文档

document.open(); // 文档里写入

Chunk chunk1 = new Chunk(convertChunkToChinese("信息技术手册检测报告", 20, bold, black));

document.add(chunk1);

document.add(new Paragraph("\n"));

Paragraph pp = new Paragraph(convertParToChinese("检测范围：◎百度词条", 9, normal, black));

document.add(pp);

Paragraph pp1 = new Paragraph(convertParToChinese("颜色标注说", 9, normal, black));

document.add(pp1);

Paragraph pp2 = new Paragraph();

pp2.add(new Phrase(convertParToChinese("口", 9, normal, red)));

pp2.add(new Phrase(convertParToChinese("：复写片段(相似或疑似重)", 9, normal, black)));

document.add(pp2);

List<citiao> cs = DBUtil.load\_all();

for (citiao c : cs) {

Paragraph chapterTitle = new Paragraph("\n\n");

Paragraph p0 = new Paragraph(convertParToChinese(c.getId() + "." + c.getName(), 14, bold, black));

chapterTitle.add(p0);

Paragraph p3 = new Paragraph(convertParToChinese("类别：" + c.getClasss(), 9, normal, blue));

chapterTitle.add(p3);

Paragraph p = new Paragraph(convertParToChinese("摘要：" + c.getAbstracts(), 9, normal, blue));

chapterTitle.add(p);

Paragraph p1 = new Paragraph(convertParToChinese("关键词：" + c.getKeywords(), 9, normal, blue));

chapterTitle.add(p1);

Paragraph p2 = returnExpain(c.getId(), c.getExplain());// 解释

chapterTitle.add(p2);

document.add(chapterTitle);

}

document.add(new Paragraph("\n\n"));

document.add(new Paragraph(convertParToChinese("检测结论：",22, normal, black)));

Paragraph a1=new Paragraph();

a1.add(new Phrase(convertParToChinese("全文相似度：", 9, normal, black)));

sum();

DecimalFormat decimalFormat=new DecimalFormat(".00");//构造方法的字符格式这里如果小数不足2位,会以0补足.

String p=decimalFormat.format((float)100\*count/sum);

System.out.println(p);

a1.add(new Phrase(convertParToChinese(p+"%", 9, normal, red)));

document.add(a1);

document.add(new Paragraph(convertParToChinese("总相似片段：" +cop\_count, 9, normal, black)));

document.close();

}

public static Paragraph returnExpain(int yuan\_id, String explain) throws Exception {

List<copy> copys = DBUtil.load\_id(yuan\_id);

Paragraph a = new Paragraph();

String word = "";

int j = 0;

if (copys.size() != 0) {

copy c = copys.get(j);

j++;

for (int i = 0; i < explain.length(); i++) {

if (i < c.getStart\_position()) {

word = word + explain.charAt(i);

} else if (i == c.getStart\_position()) {

if (word != "") {// 抄袭语句前面的语句

a.add(new Phrase(convertParToChinese2(word, 12, normal, black)));

word = "";

}

word = word + explain.charAt(i);

} else if (i == c.getLast\_position()) {

word = word + explain.charAt(i);

if (word != "") {// 抄袭语句前面的语句

cop\_count++;

count=count+word.length();

a.add(new Phrase(convertParToChinese2(word, 12, normal, red)));

word = "";

if (j < copys.size()) {

c = copys.get(j);

j++;

}

}

}else if (i == explain.length()-1) {

word = word + explain.charAt(i);

a.add(new Phrase(convertParToChinese2(word, 12, normal, black)));

}else if(i>explain.length()){

word = word + explain.charAt(i);

}else if(i>c.getStart\_position()&&i<c.getLast\_position()) {

word = word + explain.charAt(i);

}else if(i>c.getLast\_position()) {

word = word + explain.charAt(i);

}

}

} else {

a = new Paragraph(convertParToChinese(explain, 12, normal, black));

}

return a;

}

public static void sum() {

List<citiao> citiaos=DBUtil.load\_all();

for(citiao c:citiaos) {

sum=sum+c.getExplain().length();

}

}

public static Paragraph convertParToChinese(String text, int fontsize, int fontStyle, Color color)

throws Exception {

BaseFont baseFontChinese = BaseFont.createFont("STSong-Light", "UniGB-UCS2-H", BaseFont.NOT\_EMBEDDED);

Font fontChinese = new Font(baseFontChinese, fontsize, fontStyle, color);

Paragraph graph = new Paragraph(text, fontChinese);

return graph;

}

public static Phrase convertParToChinese2(String text, int fontsize, int fontStyle, Color color) throws Exception {

BaseFont baseFontChinese = BaseFont.createFont("STSong-Light", "UniGB-UCS2-H", BaseFont.NOT\_EMBEDDED);

Font fontChinese = new Font(baseFontChinese, fontsize, fontStyle, color);

Phrase graph = new Phrase(text, fontChinese);

return graph;

}

public static Chunk convertChunkToChinese(String text, int fontsize, int fontStyle, Color color) throws Exception {

BaseFont baseFontChinese = BaseFont.createFont("STSong-Light", "UniGB-UCS2-H", BaseFont.NOT\_EMBEDDED);

Font fontChinese = new Font(baseFontChinese, fontsize, fontStyle, color);

Chunk graph = new Chunk(text, fontChinese);

return graph;

}

}

# 4.数据库设计

1. yuan表记录信息技术的标题，内容，分类，关键词及摘要。如表4-1所示。



表4-1 信息技术表(yuan)

2.抄袭表记录内容的编号，抄袭的片段，抄袭的网址，抄袭片段在原来内容中的起始位置。如表4-2所示。

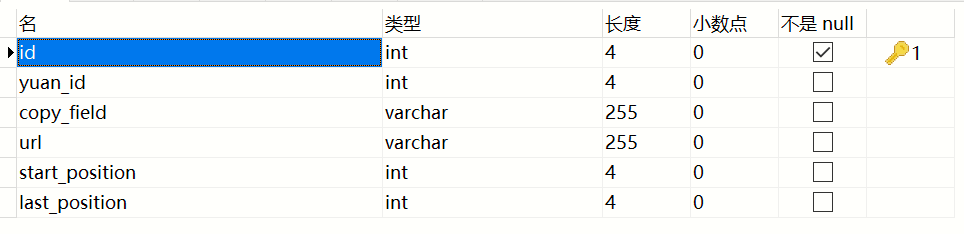


表4-2 抄袭表(copy)