DBWorld 搜索引擎

吴紫薇 PB16110763

实验内容

为DBWorld（https://research.cs.wisc.edu/dbworld/browse.html）设计并开发一个搜索引擎。

实验环境

Linux Ubuntu 16.04

Python + Flask web framework

Libs: selenium, whoosh, nltk

实验步骤及方法

1. 抓取DBWorld信息（mycrawler.py class DBworldCrawler）

   使用selenium加载firefox webdriver,初始化类，根据日期时间新建一个文件夹以保存爬取的文件。

```
def __init__(self, menu_url="https://research.cs.wisc.edu/dbworld/browse.html",
output_dir="messages"):
        self.menu_url = menu_url
        self.output_dir = output_dir
        # load browser driver
        opt = webdriver.FirefoxOptions()
        # no display mode
        opt.add_argument('--headless')
        self.browser = webdriver.Firefox(firefox_options=opt)
        if DEBUG_INIT:   print "Driver loaded."

        # make a new dir to store jsons
        self.new_dir_path = "{}/{}".format(self.output_dir,
datetime.now().strftime("%Y%m%d%H%M%S"))
        os.mkdir(self.new_dir_path)
```

   抓取全部消息，通过分析网页代码发现消息由“TBDY”这一tag标识，每个信息由“TD”标识，从而抓取到包括sent日期，type类型，author作者，subject主题，href详情链接，deadline截止日期，webpage网页链接；我们把这些信息以json文件的形式保存下来。

```
def crawl_menu_url(self):
        self.browser.get(self.menu_url)
        msgs = self.browser.find_elements_by_tag_name("TBODY")
        # process all messages, table contents
        p = progressbar.ProgressBar()
        p.start()
        print "Crawling messages from {}".format(self.menu_url)
        total_num = len(msgs)
```

```
        for idx, msg in enumerate(msgs):
            TDs = msg.find_elements_by_tag_name("TD")
            Sent = TDs[0].text
            Type = TDs[1].text
            Author = TDs[2].text
            Subject = TDs[3].text
            tmp = TDs[3].find_element_by_tag_name("A")
            Detail_href = tmp.get_attribute("HREF")
            Deadline = TDs[4].text
            try:
                WebPage_href =
TDs[5].find_element_by_tag_name("A").get_attribute("HREF")
            except Exception as e:
                WebPage_href = ""
            if DEBUG_MENU : print "
{}\t{}\t{}\t{}\t{}\t{}\t{}\n".format(Sent,Type,Author,Subject,Detail_href,Deadline,
WebPage_href)

            # save as json
            out_path = "{}/{}.json".format(self.new_dir_path, idx)
            out_dict = {
                "sent": Sent,
                "type": Type,
                "author": Author,
                "subject": Subject,
                "href": Detail_href,
                "deadline": Deadline,
                "webpage": WebPage_href
            }

            with open(out_path,"w") as out_file:
                out_file.write(json.dumps(out_dict))

            p.update(idx*100 / total_num)

        p.finish()
```

完成消息爬取后，进行详情信息爬取；先从之前保存好的json文件中读出href信息，再爬取该网页内容，通过分析网页代码发现，content内容由“PRE”这一tag标识；将详情内容保存成txt文件。

```
def crawl_detail_url(self):
        jsondir=self.new_dir_path
        total_num = len(os.listdir(jsondir))
        p = progressbar.ProgressBar()
        p.start()
        print "Crawling detail contents of messages."
        for idx in range(total_num):
            # load json
            jsonpath = "{}/{}.json".format(jsondir, idx)
            with open(jsonpath, "r") as in_file:
                buf = in_file.read()
                dic_data = json.loads(buf)
                url = dic_data["href"]
```

```
                try:
                    self.browser.get(url)
                    content = self.browser.find_element_by_tag_name("PRE").text
                except Exception:
                    content = ""

            with open("{}/{}.txt".format(self.new_dir_path, idx), "w") as out_file:
                out_file.write(content.encode("utf-8"))

            p.update(idx*100 / total_num)
        p.finish()
```

2. 建立索引文件及检索程序 （myindexer.py class DEworldIndexer ）

   首先建立索引文件的schema

```
schema = Schema(
            author=TEXT(stored=True),
            sent=DATETIME(stored=True, sortable=True),
            deadline=DATETIME(stored=True, sortable=True),
            subject=TEXT(stored=True),
            content=TEXT(stored=True),
            doctype=TEXT(stored=True),
            href=TEXT(stored=True),
            webpage=TEXT(stored=True)
        )
```

   读取保存好的json文件和txt文件，抽取出包括时间、地点、会议名称等关键信息，构建索引。

```
for idx in range(self.docnum):
            jsonpath = "{}/{}.json".format(self.msgdir, idx)
            txtpath = "{}/{}.txt".format(self.msgdir, idx)

            in_json = open(jsonpath, "r")
            buf = in_json.read()
            dic_data = json.loads(buf)
            #print(dic_data)

            in_txt = open(txtpath, "r")
            txt_data = in_txt.read()
            #print(txt_data)

            sent_tmp = dic_data["sent"]
            if len(sent_tmp):
                sent_tmp = sent_tmp.split("-")
                sent_field = "{}-{}-{}".format(sent_tmp[2], mon2num[sent_tmp[1]],
sent_tmp[0])


            deadline_tmp = dic_data["deadline"]
            if len(deadline_tmp):
                deadline_tmp = deadline_tmp.split("-")
```

```
                deadline_field = "{}-{}-{}".format(deadline_tmp[2],
mon2num[deadline_tmp[1]], deadline_tmp[0])
            else:
                deadline_field = None

            #print(sent_field)
            #print(deadline_field)
            if deadline_field:
                writer.add_document(
                    author=dic_data["author"],
                    sent=sent_field,
                    deadline=deadline_field,
                    subject=dic_data["subject"],
                    content=txt_data,
                    doctype=dic_data["type"],
                    href=dic_data["href"],
                    webpage=dic_data["webpage"]
                )
            else:
                writer.add_document(
                    author=dic_data["author"],
                    sent=sent_field,
                    subject=dic_data["subject"],
                    content=txt_data,
                    doctype=dic_data["type"],
                    href=dic_data["href"],
                    webpage=dic_data["webpage"]
                )

            print("{} added".format(txtpath)
        # commit adding process
        writer.commit()
```

基于whoosh的检索程序，根据前端选择的不同检索域对不同的信息进行检索。

```
class DBworldSearcher:

    def __init__(self, indexdir, fieldlist=["subject", "content"]):
        self.indexdir = indexdir
        ix = open_dir(indexdir)

        #self.parser = QueryParser("subject", self.ix.schema)
        self.parser = MultifieldParser(fieldlist, ix.schema)
        self.parser.add_plugin(DateParserPlugin())
        self.searcher = ix.searcher()

    def search(self, querytext, limit):
        myquery = self.parser.parse(querytext)
        results = self.searcher.search(myquery, limit=limit)
        return results
```

3. 服务器和网页开发

主要包括主页（mainpage.html）和搜索详情页（results.html）

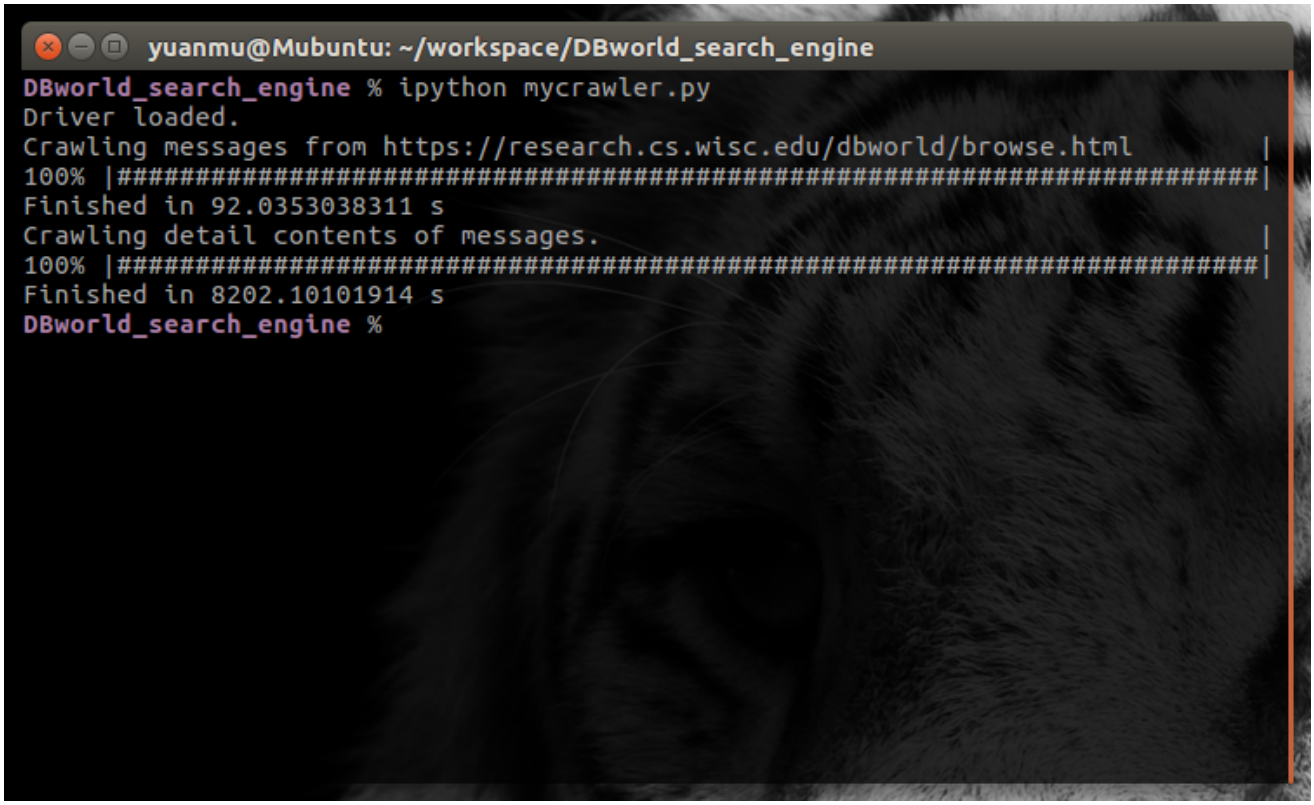
通过一个form传递query文本和目标检索域，以request args的形式传递给search函数。

```
@app.route('/', methods=["GET", "POST"])
def mainpage():
    # POST and query not empty
    if request.method == "POST" and len(request.form["query"]):
        query = request.form["query"]
        filedid = request.form["field"]
        #print(filedid)
        return redirect(url_for('search', q=query, p=1, f=filedid))
    # GET
    return render_template("mainpage.html")

@app.route('/search', methods=["GET","POST"])
def search():
    # POST a new query
    if request.method == "POST":
        query = request.form["query"]
        filedid = request.form["field"]
        return redirect(url_for('search', q=query, p=1, f=filedid))

    # Search query
    query = request.args["q"]
    page = int(request.args["p"])
    filedid = request.args["f"]

    if filedid == "0":
        # search subject & content
        dbworld_searcher = sub_con_searcher
        tmp = dbworld_searcher.search(querytext=query, limit=page*10)
        time_cost = round(tmp.runtime, 3)
        results = [(x["sent"], x["author"], x["subject"],
            x["deadline"], x.highlights("content"), x["href"], x["webpage"],
x["doctype"]) for x in tmp]
        return render_template("results.html",
            msg=[len(tmp), time_cost], query=query, page=page, results=results)

    elif filedid == "1":
        # search author
        dbworld_searcher = auth_searcher
        tmp = dbworld_searcher.search(querytext=query, limit=page*10)
        time_cost = round(tmp.runtime, 3)
        results = [(x["sent"], x["author"], x["subject"],
            x["deadline"], x["content"][:600], x["href"], x["webpage"],
x["doctype"]) for x in tmp]
        return render_template("results.html",
            msg=[len(tmp), time_cost], query=query, page=page, results=results)

    elif filedid == "2":
        # search conference
        dbworld_searcher = conf_searcher
        tmp = dbworld_searcher.search(querytext=query, limit=page*10)
```

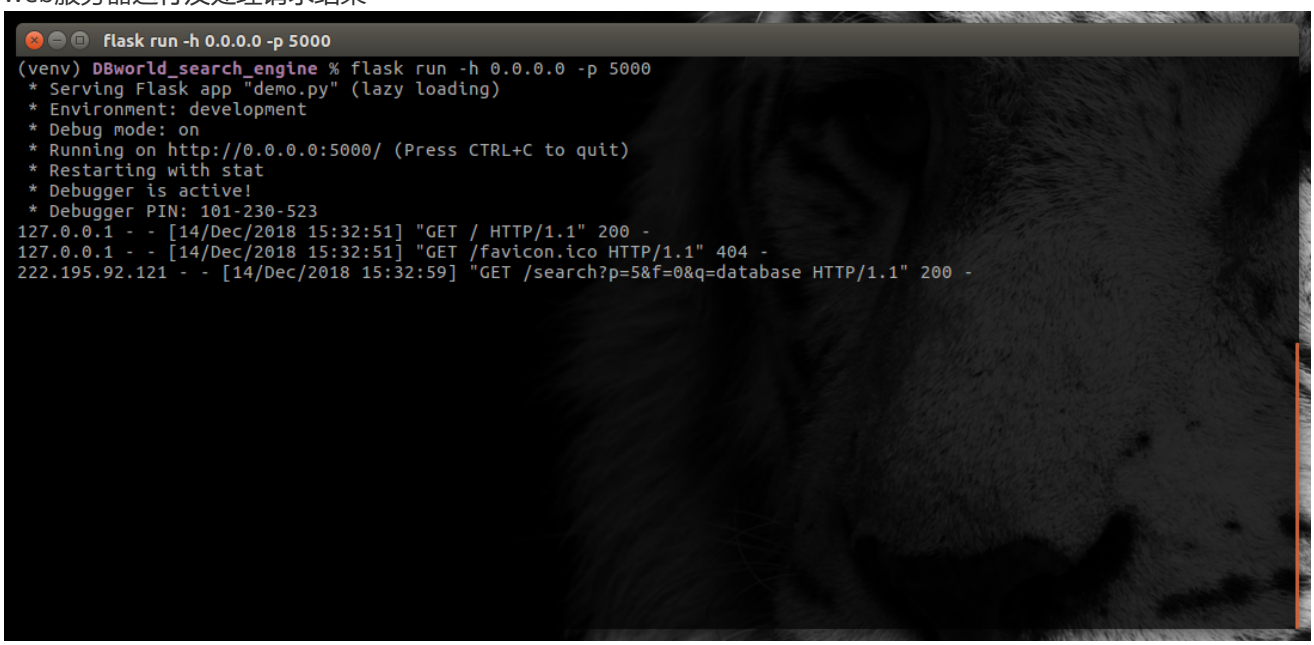
```
        time_cost = round(tmp.runtime, 3)
        results = [(x["sent"], x["author"], x.highlights("subject"),
            x["deadline"], x["content"][:600], x["href"], x["webpage"],
x["doctype"]) for x in tmp]
        return render_template("results.html",
            msg=[len(tmp), time_cost], query=query, page=page, results=results)

    elif filedid == "3":
        # search sent date
        dbworld_searcher = sent_searcher
        tmp = dbworld_searcher.search(querytext=query, limit=page*10)
        time_cost = round(tmp.runtime, 3)
        results = [(x["sent"], x["author"], x["subject"],
            x["deadline"], x["content"][:600], x["href"], x["webpage"],
x["doctype"]) for x in tmp]
        return render_template("results.html",
            msg=[len(tmp), time_cost], query=query, page=page, results=results)

    elif filedid == "4":
        # search ddl date
        dbworld_searcher = ddl_searcher
        tmp = dbworld_searcher.search(querytext=query, limit=page*10)
        time_cost = round(tmp.runtime, 3)
        results = [(x["sent"], x["author"], x["subject"],
            x["deadline"], x["content"][:600], x["href"], x["webpage"],
x["doctype"]) for x in tmp]
        return render_template("results.html",
            msg=[len(tmp), time_cost], query=query, page=page, results=results)
```

实验结果说明及演示

爬虫运行结果

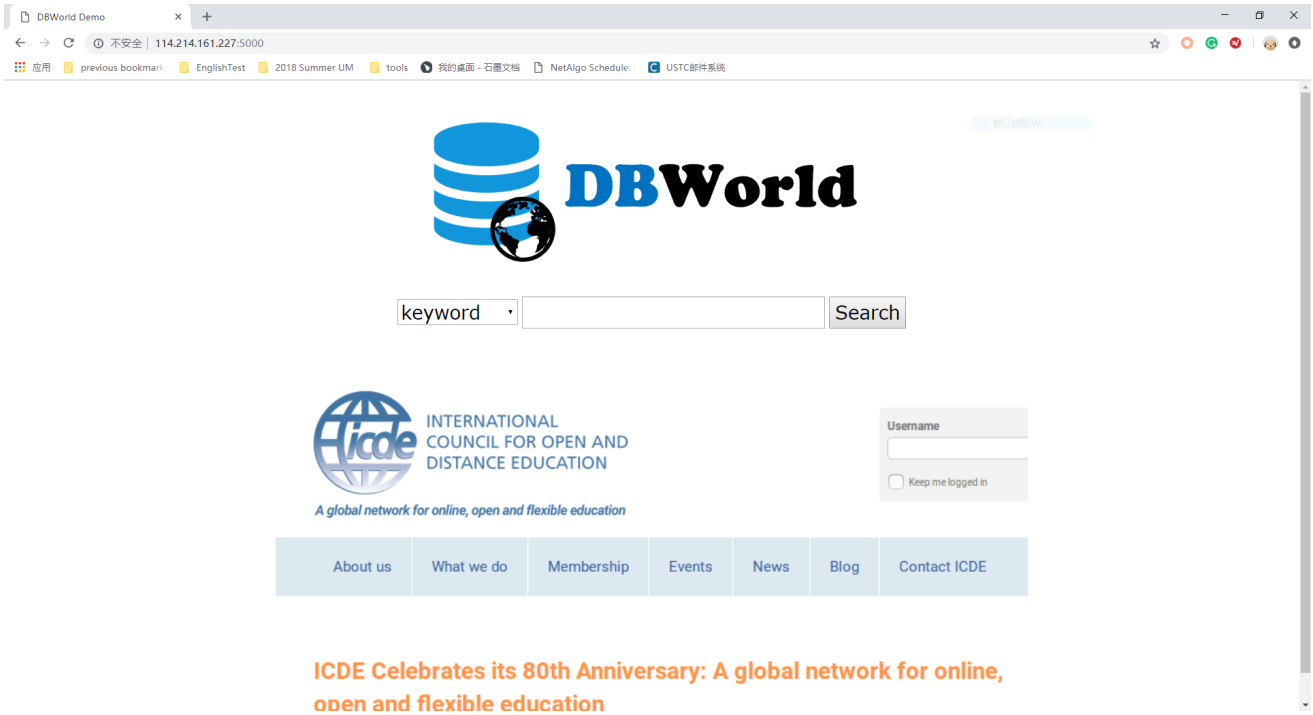


web服务器运行及处理请求结果

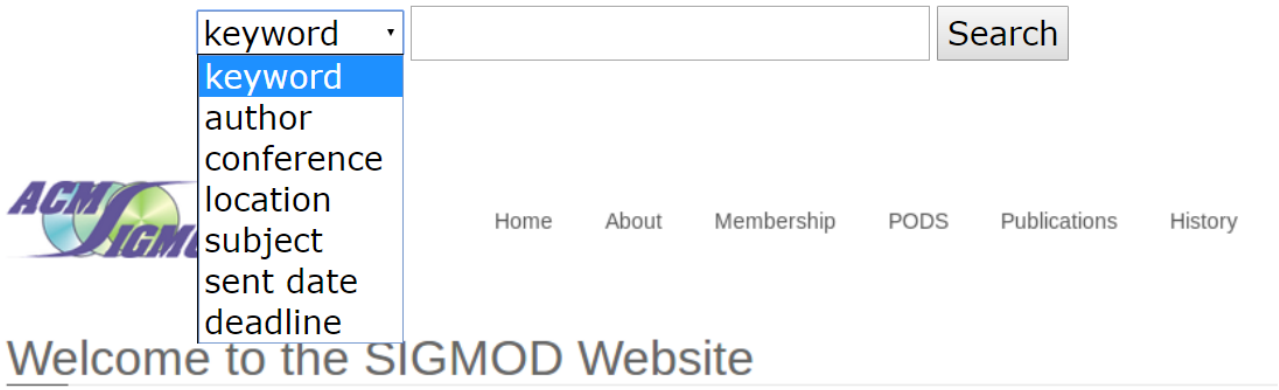


demo网页 http://114.214.161.227:5000/ 使用校园网访问

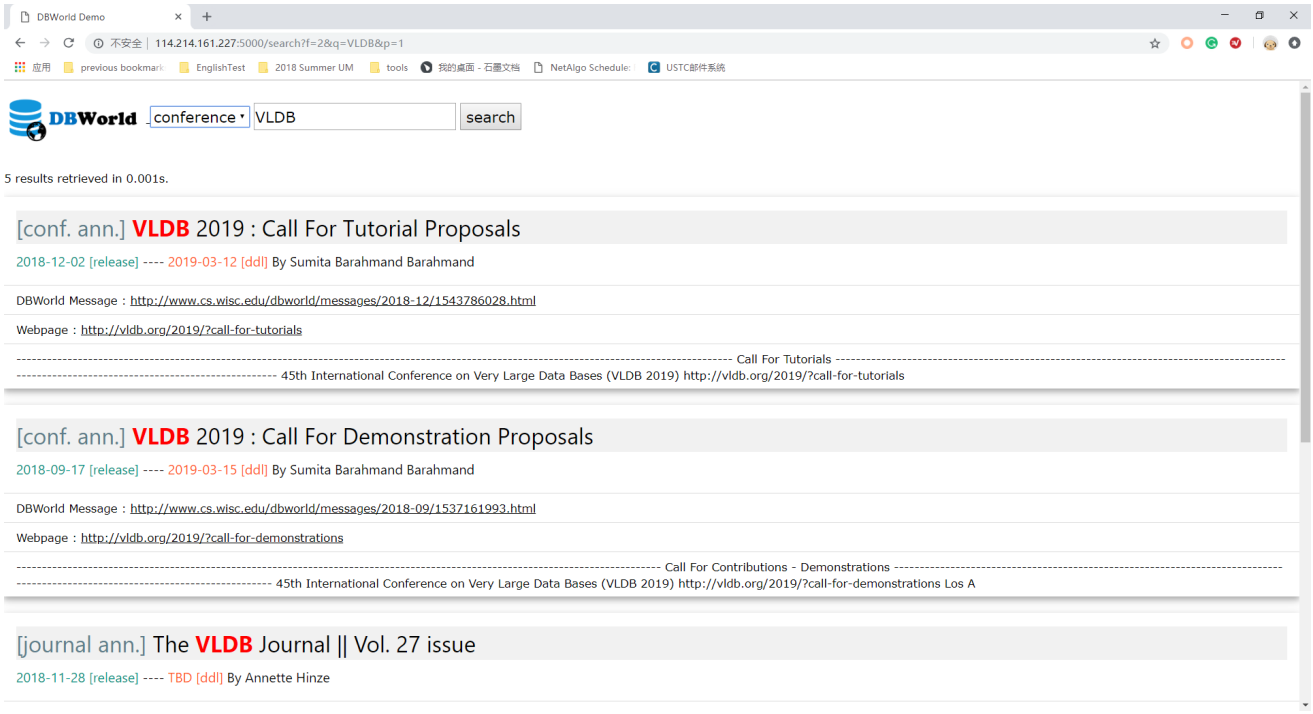
主页:



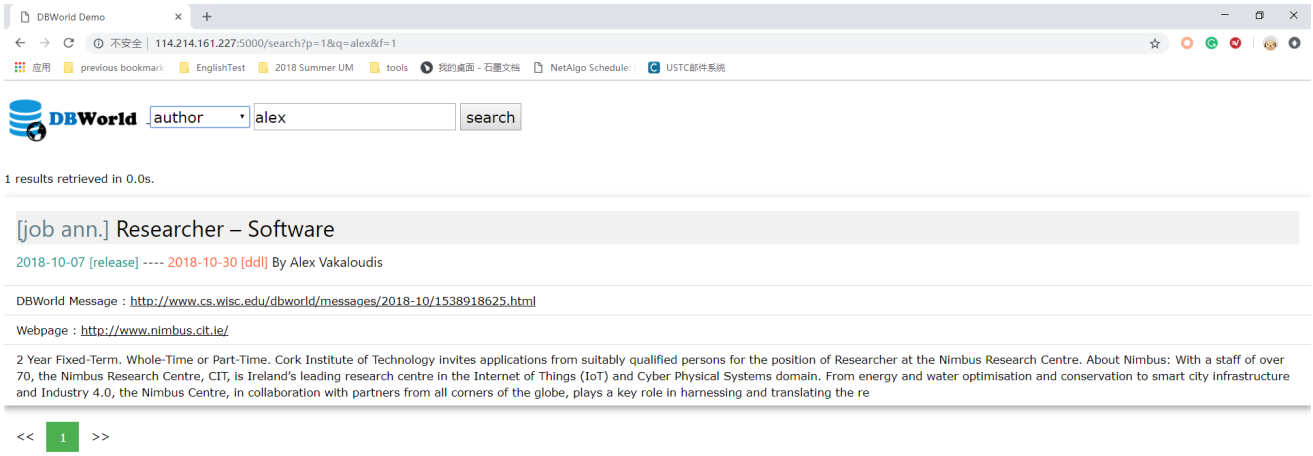
多种检索域选择：



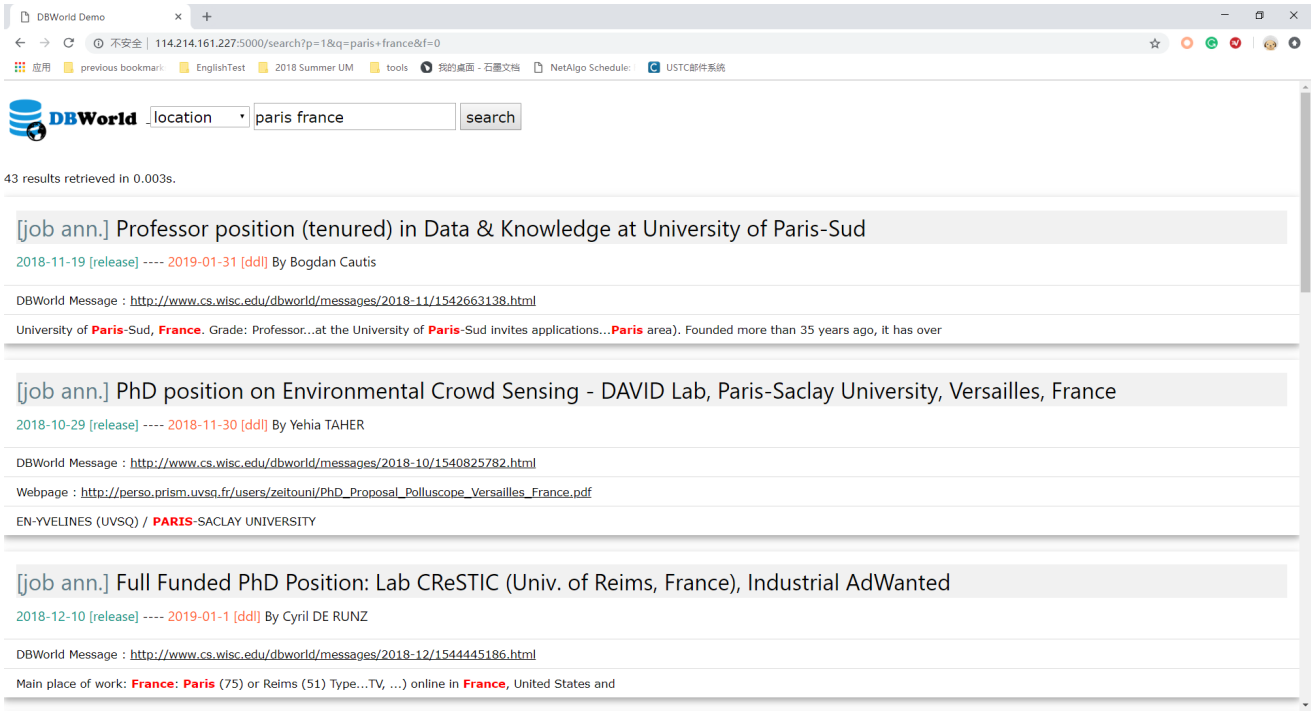
检索会议&期刊名 conference：VLDB



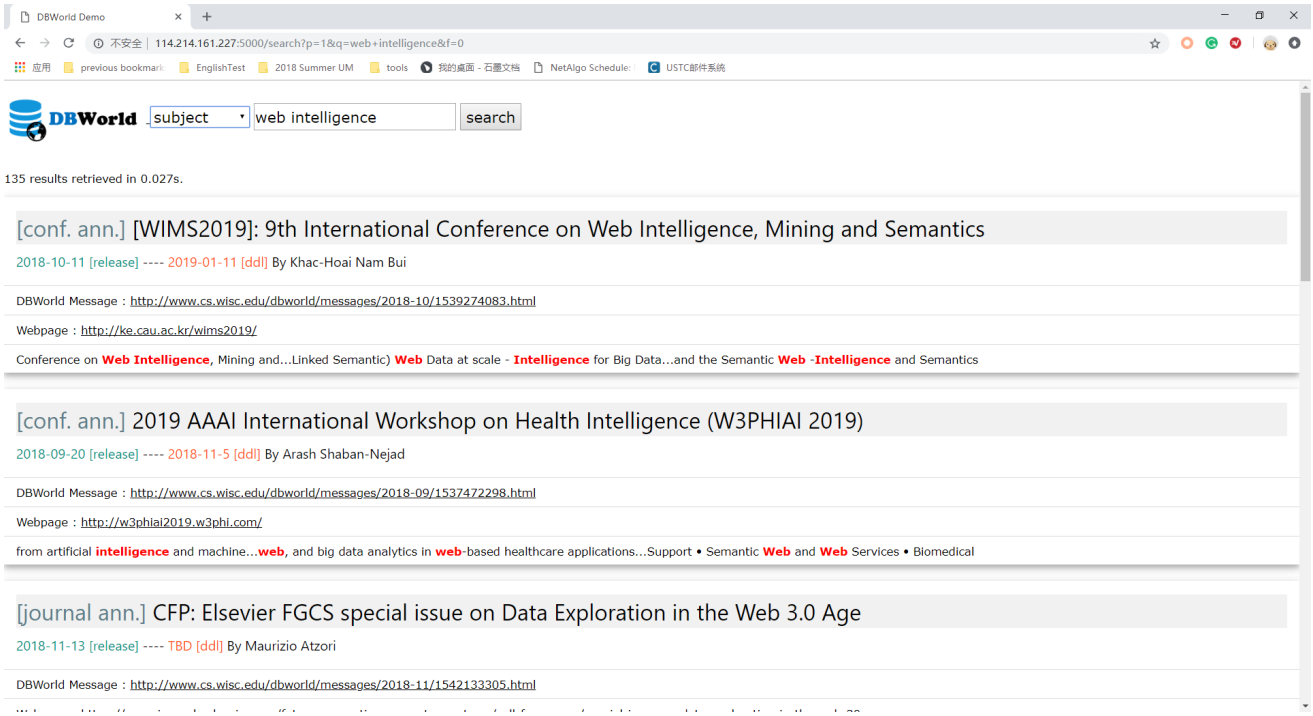
检索作者author：alex



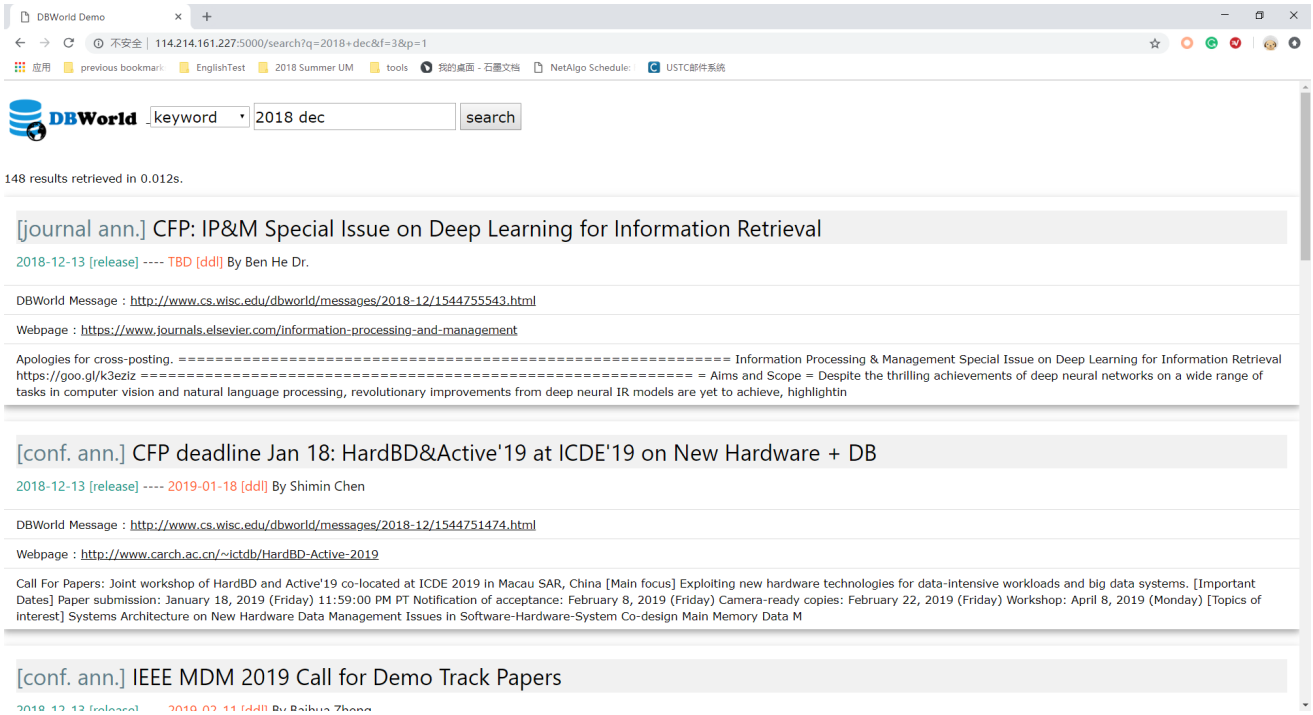
检索会议地点 location：paris france



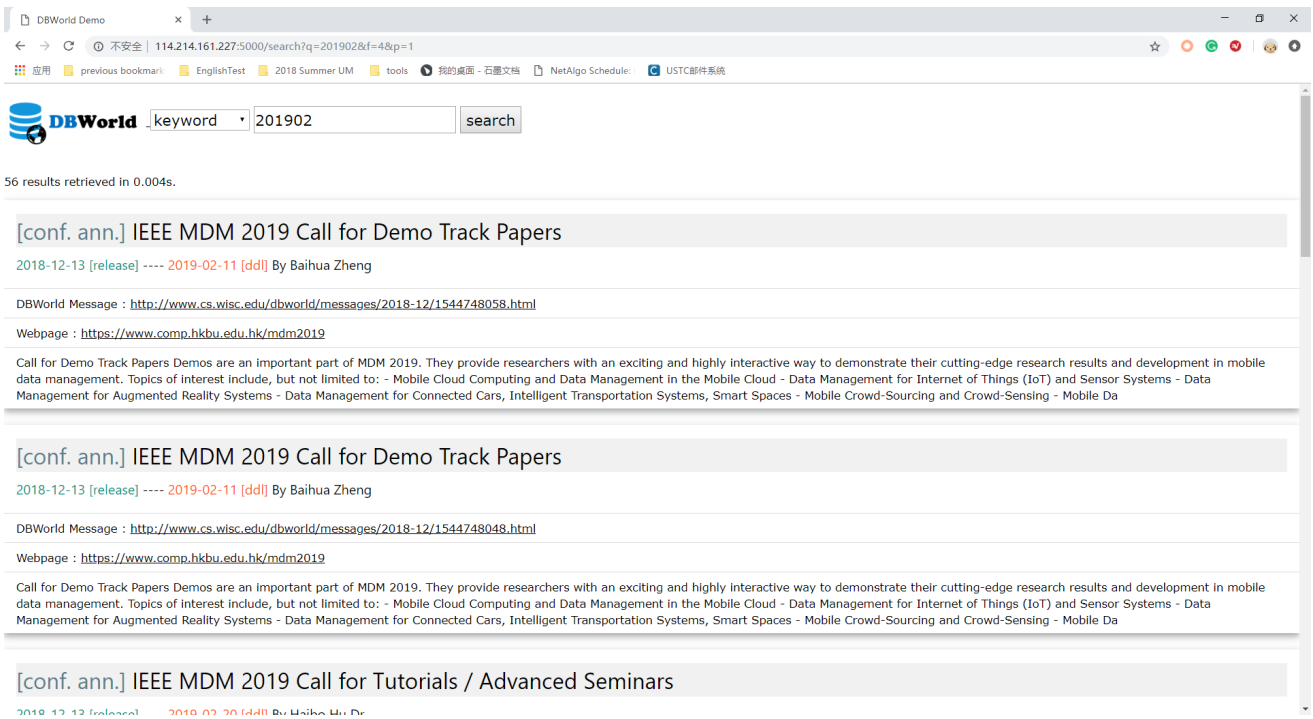
检索会议主题 subject：web intelligence



检索发送时间sent：2018 dec



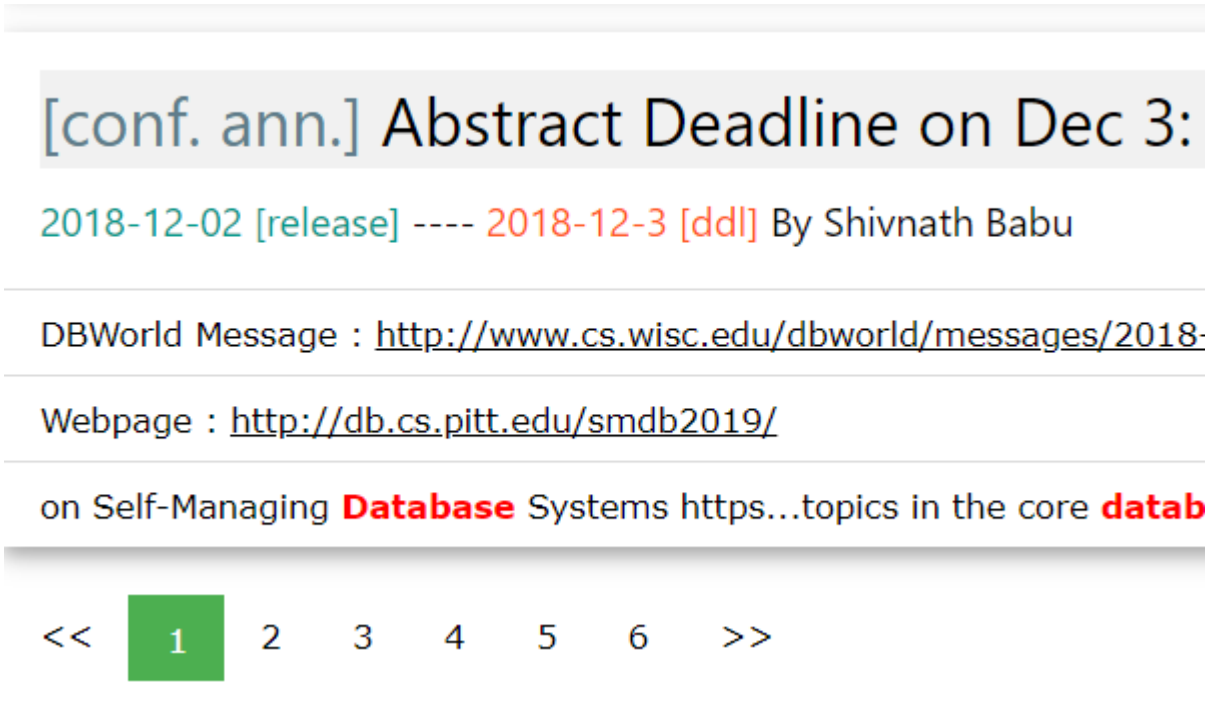
检索截止时间deadline：201902

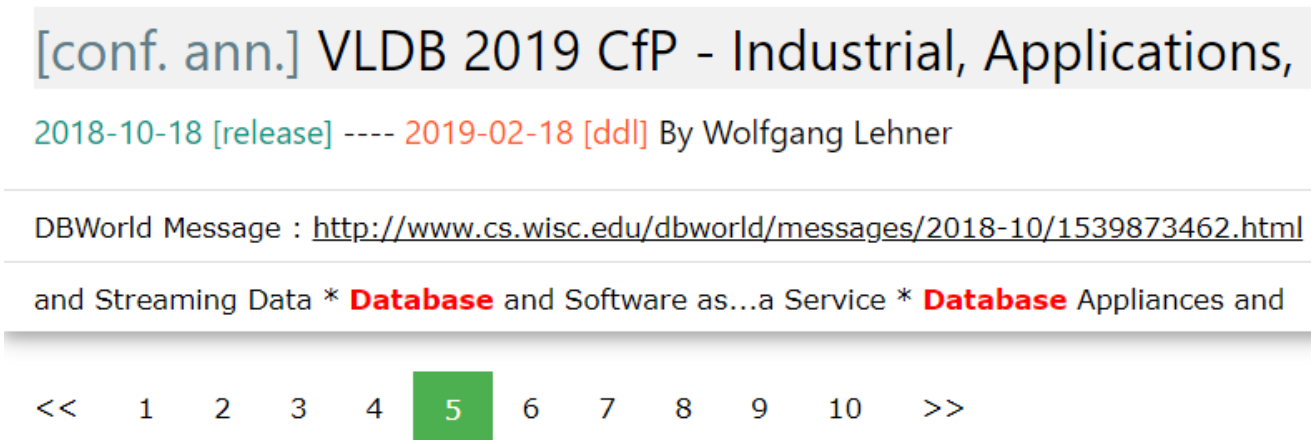


实验总结

亮点：

1. 满足了本实验的要求，实现了多种常用的检索目标的搜索
2. 信息准确，搜索反应速度快，结果展示页面清晰美观（相关词高亮，结果分页）





不足:

1. 爬取详细信息速度较慢（主要原因应该在于访问网站速度慢，打开详情网页大概需要3-4s）