厦門大學

《社交网络技术与应用》

数据爬取与问答系统

厦门大学
信息学院

目录

1 实验介绍

问答系统是 NLP 的一个重要方向，本实验重点试验文本问答。而针对中文文本数据，往往需要进行预处理和中文分词，中文分词不准确，将造成偏差过高。因此在对社交媒体文本问题进行理解需要数据预处理与中文分词。本实验只实现精准匹配。

本章实验难度为初级。

- 初级实验：基于检索的文本问答
- 初级实验：基于 URL 的数据爬取

实验目的

- 了解 Python3 基础编程语法
- 了解如何安装 Python3 的第三方库
- 掌握如何用 Python 实现基于 URL 的数据爬取
- 实现自己修改实验课模板代码实现所需数据爬取
- 掌握问答系统的原理与使用
- 掌握前端页面开发

实验清单

实验	简述	难度	软件环境	开发环境
文本问答	基于社交媒体问答数据，使用基于检索的方法建立问答对话系统	初级	Python3	本地PC
爬虫	从包含问答的页面中提取问题标题、提问内容以及回答内容，并将结果保存为 CSV 文件	初级	Python3	本地PC

2 文本问答

2.1 实验介绍

本次实验我们将实现基于检索的文本问答。问答系统所需要的数据已经提供，对于每一个问题都可以找得到相应的答案，所以可以理解为每一个样本数据是<问题、答案>。 那系统的核心是当用户输入一个问题的时候，首先要找到跟这个问题匹配的已经存储在库里的问题，然后直接返回相应的答案即可。

并且我们将实现一个爬虫，从包含问答的页面中提取问题标题、提问内容以及回答内容，并将结果保存为 CSV 文件，然后将 CSV 文件导入数据库中。

2.2 实验目的

掌握 Django 框架

掌握基于检索的问答系统的基本原理

掌握问答系统的基本过程

掌握如何用 Python 实现基于 URL 的数据爬取

实现自己修改实验课模板代码实现所需数据爬取

2.3 实验步骤

本次实验使用的数据集来源于网上公开的问答数据集（baoxianzhidao_filter）以及用爬虫爬取的问答数据。

实验任务介绍：本次实验我们将设计一个问答系统，进行文本问答的教学。

2.3.1 文本数据准备

步骤 1 新建问答数据表

直接在 app 的 models.py 文件中添加，代码：

```python
# 创建问答知识库表
class QuestionAnswerKB(models.Model):
    question = models.TextField()
    answer = models.TextField()
    def __str__(self):
        return self.question
```

创建后的表格如果想要在 Django 后台管理系统进行管理，就需要在 admin.py 里头注册，注意要在 admin.py 中导入所需的包（例如，QuestionAnswerKB）：

```
from django.contrib import admin
from import_export import resources
from movierecommendation.models import DoubanMovie, QuestionAnswerKB
from import_export.admin import ImportExportModelAdmin
```

```python
class QuestionAnswerResource(resources.ModelResource):
    class Meta:
        model = QuestionAnswerKB
        export_order = ('question','answer')

@admin.register(QuestionAnswerKB)
class MQuestionAnswerAdmin(ImportExportModelAdmin):
    list_display = ('question','answer')
    search_field = ('question')
    resource_class = QuestionAnswerResource
```

在命令行输入“python manage.py makemigrations”创建迁移文件

输出：

```
C:\Users\emmal\doubanrecommendation>python manage.py makemigrations
Migrations for 'movierecommendation':
  movierecommendation\migrations\0003_questionanswerkb.py
    - Create model QuestionAnswerKB

C:\Users\emmal\doubanrecommendation>
```

在命令行输入“python manage.py migrate”完成迁移文件的导入

输出：

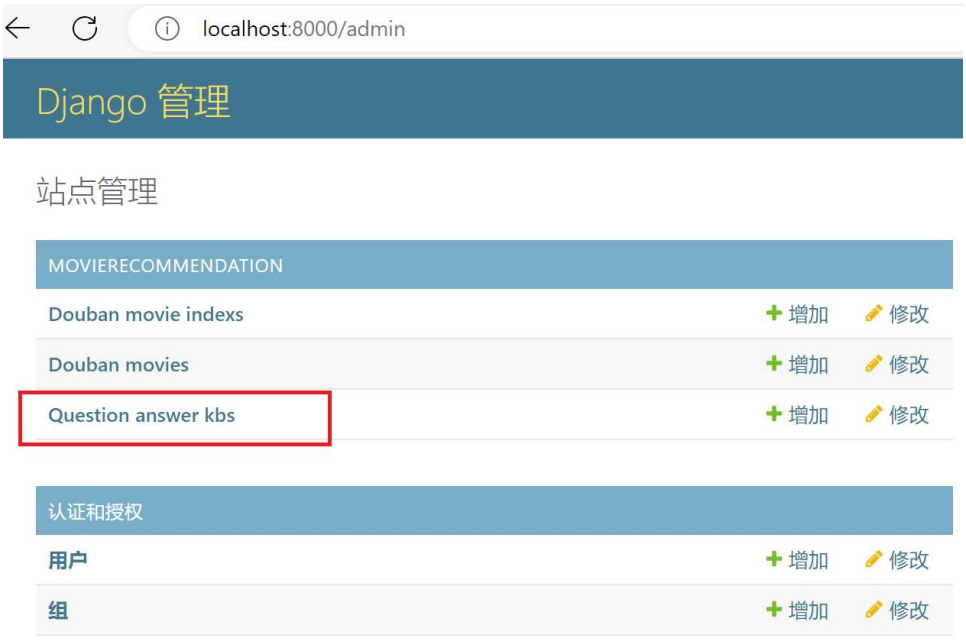
```
C:\Users\emmal\doubanrecommendation>python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, movierecommendation, sessions
Running migrations:
  Applying movierecommendation.0003_questionanswerkb... OK

C:\Users\emmal\doubanrecommendation>
```

启动服务器（python manage.py runserver），并登录后台管理系统（http://localhost:8000/admin）

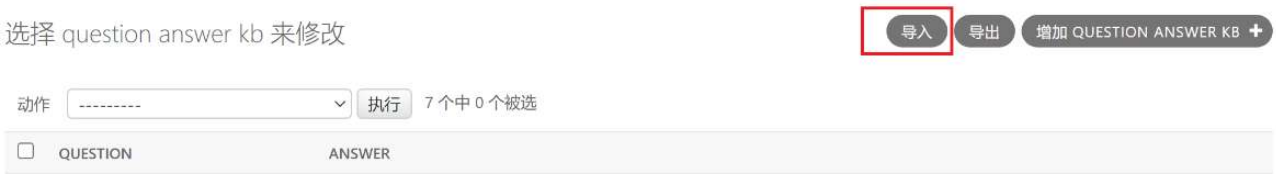
如果忘记管理员账号密码可以使用如下步骤进行重置：

```
PS E:\##study\助教\lab2\doubanrecommendation> python manage.py shell
Python 3.9.9 (tags/v3.9.9:ccb0e6a, Nov 15 2021, 18:08:50) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
(InteractiveConsole)
>>> from django.contrib.auth.models import User
>>> user=User.objects.get(pk=1)
>>> user
<User: admin>
>>> user.set_password('admin')
>>> user.save()
>>> exit()
```

登录后台管理系统可以看到数据库的所有表格：



点击表格名称进入管理页面，你可以点击“导入”按钮

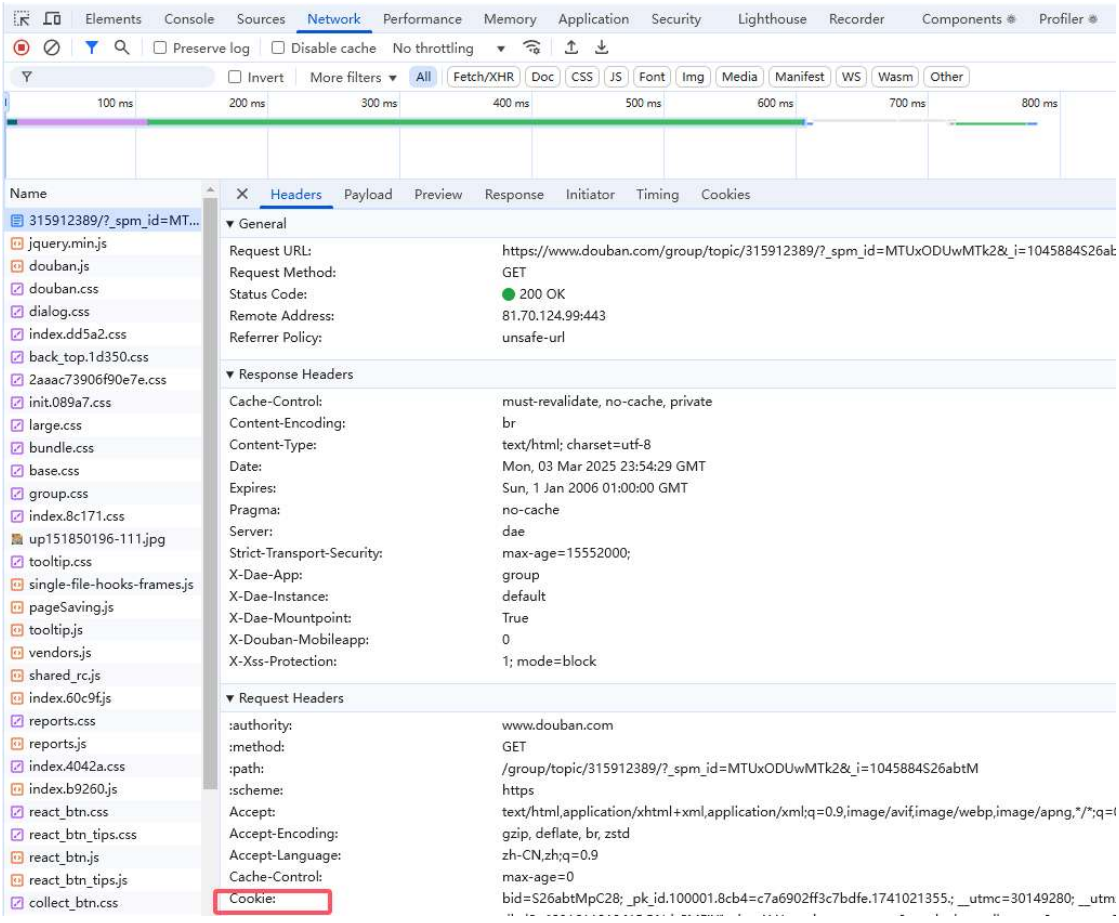


通过选择数据 CSV 文件来导入问答数据（示例数据文件请见“baoxianzhidao_filter.csv”）：

导入后的数据如下：

接下来我们将使用 Python3 实现基于 URL 和用户模拟登录的数据爬取实验，首先请在豆瓣小组中找一个你感兴趣的话题。如下示例：

登录豆瓣，点击 F12，然后按 F5 刷新页面，在左侧目录中寻找右侧 headers 里包含 cookie 的选项，然后记录下你的 cookie：



新建一个 python 项目，创建一个 python 文件，文件名任意，在文件中编写 python 爬虫代码。

```
import requests
import pandas as pd
from bs4 import BeautifulSoup
import re

# ====== 配置区 ======
TOPIC_URL = "https://www.douban.com/group/topic/89252805/"  # 替换为目标问答页面的 URL
COOKIE = "" #替换为你的 cookie

# ====== 请求头配置 ======
headers = {
    'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/91.0.4472.124 Safari/537.36',
    'Referer': 'https://www.douban.com/',
    'Cookie': COOKIE,
    'Accept':
'text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8',
    'Accept-Language': 'zh-CN,zh;q=0.9,en;q=0.8',
    'Connection': 'keep-alive'
}

# ====== 核心函数 ======
def fetch_douban_topic():
    try:
        # 发送请求
        response = requests.get(TOPIC_URL, headers=headers, timeout=15)
        response.raise_for_status()

        # 解析 HTML
        soup = BeautifulSoup(response.text, 'html.parser')

        # 提取问题标题
        title = soup.find('h1').text.strip()
        title = filter_gbk_unsupported(title)

        # 提取问题内容
        question_content = extract_clean_question_content(soup)
        # print(soup)
        clean_text(question_content)
        question_content = filter_gbk_unsupported(question_content)

        # 提取回答内容
        answers = []
        reply_items = soup.find_all('li', class_='reply-item')
        # print(reply_items)
        for reply in reply_items:
            answer_content = reply.find('div', class_='reply-content').find('div',
class_='markdown').text.strip()
            answers.append(filter_gbk_unsupported(answer_content))

        # 存储数据
        data = {
            # '序号': 1,  # 主键（序号）
            'question': title + ":" + question_content,
```

```
            # 'answer': "\n".join(answers), # 将回答内容拼接为字符串
            'answer': [f" 回答{i+1}: {answer};" for i, answer in enumerate(answers)],
        }

        # 保存为 CSV 文件
        df = pd.DataFrame([data])
        df.to_csv('douban_topic.csv', encoding='gbk', index=False)
        print("数据已保存到 douban_topic.csv")

    except Exception as e:
        print(f"抓取失败：{str(e)}")

# ====== 辅助函数：提取干净的提问内容 ======
def extract_clean_question_content(soup):
    # 找到提问内容的容器
    topic_content = soup.find('div', class_='topic-content').find('div', class_='rich-content')
#.text.strip()
    if not topic_content:
        return ""

    # 移除不需要的元数据标签
    for tag in topic_content.find_all(class_=['sharing-button', 'action-collect', 'action-
react', 'collect', 'share']):  # 替换为实际的类名
        tag.decompose()  # 移除标签

    # 提取干净的文本内容
    clean_content = topic_content.get_text(separator='\n').strip()
    return clean_content

# ====== 辅助函数：清理文本中的多余换行符 ======
def clean_text(text):
    # 使用正则表达式替换连续的换行符为单个换行符
    text = re.sub(r'\n+', '\n', text)
    # 移除文本开头和结尾的换行符
    text = text.strip()
    return text

def filter_gbk_unsupported(text):
    filtered_text = []
    for char in text:
        try:
            # 尝试将字符编码为 gbk
            char.encode('gbk')
            filtered_text.append(char)
        except UnicodeEncodeError:
            # 如果无法编码，跳过该字符
            continue
    return ''.join(filtered_text)

# ====== 执行入口 ======
if __name__ == "__main__":
    fetch_douban_topic()
```

运行代码会得到一个 csv 文件，然后在后台管理系统导入问答数据：

你也可以尝试爬取其他网站的问答数据，但是注意修改配置区和请求头配置，不同的网站可能会有不同的反爬虫机制。

步骤 2 添加问答页面

定义路由:

在 movierecommendation/url.py 中定义路由：

```
#添加路由
url(r'^questionAnswering', views.questionAnswering, name='questionAnswering'),
```

在 doubanrecommendation/url.py 中定义路由：

```
#添加路由
url(r'^questionAnswering', views.questionAnswering),
```

定义页面响应函数，在 movierecommendation/views.py 中定义相应页面：

代码：

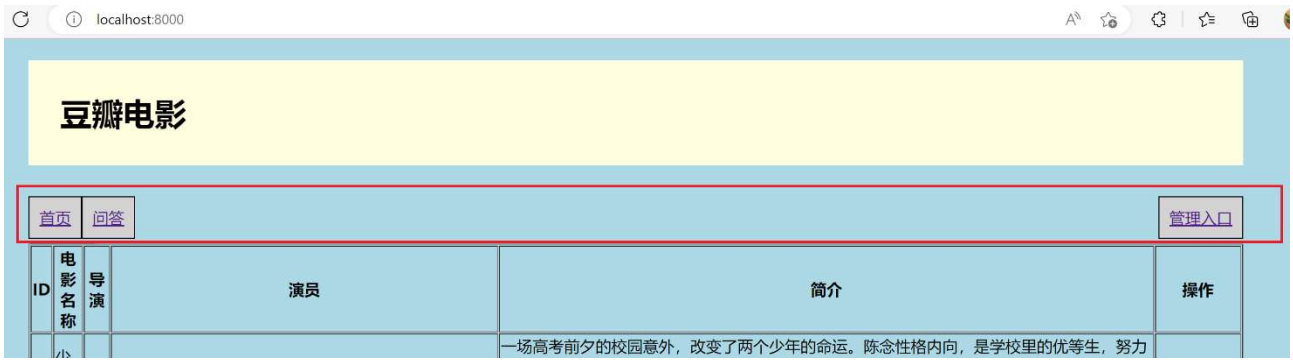
```
    # 定义问答页面.
    def questionAnswering(request):
        return render(request, 'questionAnswering.html')
```

在页面模板“doubanRecommendation.html”中加入导航栏，添加“首页”、“问答”、“管理入口”页面导航

```
<!DOCTYPE html>
<html language="zh-cn">
  <head>
     {% load static %}
    <link href="{% static 'css/style.css' %}" rel="stylesheet" type="text/css" />
    <script> var questionAnsweringUrl = "{% url 'questionAnswering' %}";</script>
    <script src="https://code.jquery.com/jquery-3.6.4.min.js" type="text/javascript"></script>
    <script src="{% static 'js/mysearch.js' %}"></script>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
    <title>豆瓣电影数据列表</title>
  </head>
  <body>
    <h1>豆瓣电影</h1>
    <div class="navigation">
        <ul class="nav">
            <li><a href="http://127.0.0.1:8000/">首页</a></li>
            <li class="last"><a href="http://127.0.0.1:8000/questionAnswering/">问答</a></li>
        </ul>
        <ul class="nav-admin">
            <li><a href="http://127.0.0.1:8000/admin">管理入口</a></li>
        </ul>
    </div>
    <div class="clear"></div>
    <div>
        <table border="1" cellspacing="1" cellpadding="1" >
            <tr>
                <th>电影名称</th>
                <th>导演</th>
                <th>演员</th>
```

定义 CSS 样式：static/css 下面修改“style.css”，添加导航栏样式，样式可以根据喜好设计

```
/*导航栏*/
ul.nav{
    list-style: none;
    margin: 0px;
    padding: 0px;
    width: auto;
}
ul.nav li{
    float: left;
    margin-top: 10px;
    padding: 10px;
    border-style: solid;
    border: 1px solid;
    background-color: lightgray;
}
ul.nav-admin{
    list-style: none;
    margin: 0px;
    padding: 0px;
    width: auto;
}
ul.nav-admin li{
    float: right;
    margin-top: 10px;
    padding: 10px;
    border-style: solid;
    border: 1px solid;
    background-color: lightgray;
}
.clear {
    clear: both;
}
```



定义“问答”页面模板：templates 下面添加模板“questionAnswering.html”，显示问答页面。

```
<!DOCTYPE html>
<html lang="zh-cn">
<head>
    {% load static %}
    <link href="{% static 'css/style.css' %}" rel="stylesheet" type="text/css" />
    <script src="https://code.jquery.com/jquery-3.6.4.min.js" type="text/javascript"></script>
    <script src="{% static 'js/mysearch.js' %}"></script>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
    <title>王晓黎豆瓣项目</title>
</head>
<body>
    <h1>问答系统</h1>
    <div class="navigation">
        <ul class="nav">
            <li><a href="http://127.0.0.1:8000/">首页</a></li>
            <li class="last"><a href="http://127.0.0.1:8000/questionAnswering/">问答</a></li>
        </ul>
        <ul class="nav-admin">
            <li><a href="http://127.0.0.1:8000/admin">管理入口</a></li>
        </ul>
    </div>
    <div class="clear"></div>
    <div class="container">
        <div class="content">
            <div class="item item-left">
                <div class="avatar avatar-bot"></div>
                <div class="bubble bubble-left">您好！我是litBot，很高兴为您服务。</div>
            </div>
        </div>
        <div class="input-area">
            <textarea name="text" id="chattextarea"></textarea>
            <div class="button-area">
                <button id="chatbotsendbtn">发送</button>
            </div>
        </div>
    </div>
</body>
</html>
```

定义 CSS 样式：static/css 下面修改“style.css”，添加问答窗口的样式，样式可以根据喜好自己设计。

注意：在 avatar-bot 和 avatar-user 中需要指定图片，你可以自行下载图片并修改对应的目录，否则将无法正常显示

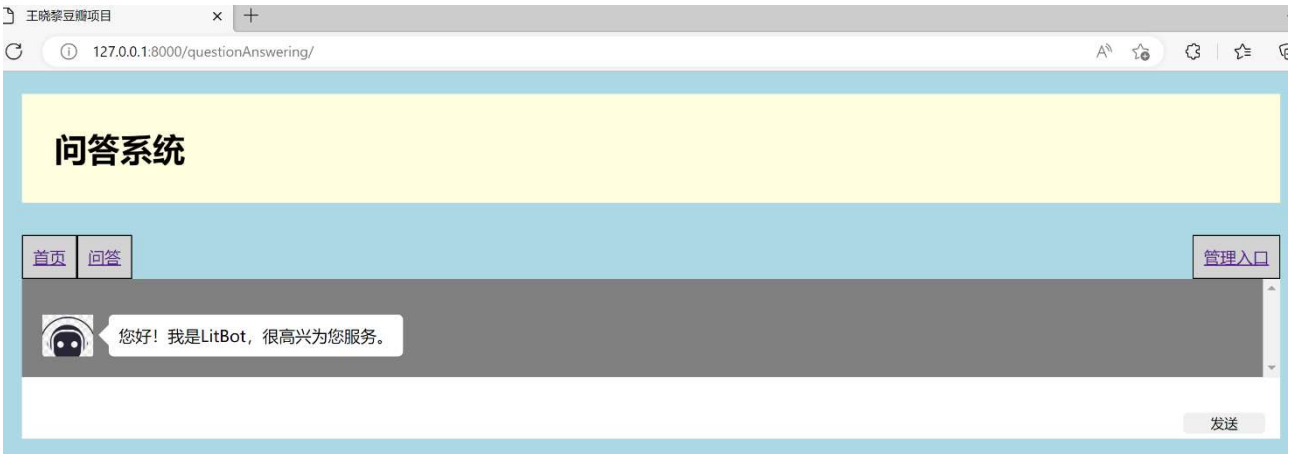
```
/*问答页面*/
.content{
    width: calc(100% - 40px);
    padding: 20px;
    overflow-y: scroll;
    flex: 1;
    background-color: gray;
}
.content:hover::-webkit-scrollbar-thumb{
    background:rgba(0,0,0,0.1);
}
.bubble{
```

```
    max-width: 400px;
    padding: 10px;
    border-radius: 5px;
    position: relative;
    color: #000;
    word-wrap:break-word;
    word-break:normal;
}
.item-left .bubble{
    margin-left: 15px;
    background-color: #fff;
}
.item-left .bubble:before{
    content: "";
    position: absolute;
    width: 0;
    height: 0;
    border-left: 10px solid transparent;
    border-top: 10px solid transparent;
    border-right: 10px solid #fff;
    border-bottom: 10px solid transparent;
    left: -20px;
}
.item-right .bubble{
    margin-right: 15px;
    background-color: #9eea6a;
}
.item-right .bubble:before{
    content: "";
    position: absolute;
    width: 0;
    height: 0;
    border-left: 10px solid #9eea6a;
    border-top: 10px solid transparent;
    border-right: 10px solid transparent;
    border-bottom: 10px solid transparent;
    right: -20px;
}
.item{
    margin-top: 15px;
    display: flex;
    width: 100%;
}
.item.item-right{
    justify-content: flex-end;
}
.item.item-center{
    justify-content: center;
```

```
}
.item.item-center span{
    font-size: 12px;
    padding: 2px 4px;
    color: #fff;
    background-color: #dadada;
    border-radius: 3px;
    -moz-user-select:none; /*火狐*/
    -webkit-user-select:none; /*webkit 浏览器*/
    -ms-user-select:none; /*IE10*/
    -khtml-user-select:none; /*早期浏览器*/
    user-select:none;
}
.input-area{
    border-top:0.5px solid #e0e0e0;
    height: 60px;
    display: flex;
    flex-flow: column;
    background-color: #fff;
}
textarea{
    flex: 1;
    padding: 5px;
    font-size: 14px;
    border: none;
    cursor: pointer;
    overflow-y: auto;
    overflow-x: hidden;
    outline: none;
    resize: none;
}
.button-area{
    display: flex;
    height: 20px;
    margin-right: 10px;
    line-height: 40px;
    padding: 5px;
    justify-content: flex-end;
}
.button-area button{
    width: 80px;
    border: none;
    outline: none;
    border-radius: 4px;
    float: right;
    cursor: pointer;
}
.avatar-bot{
```

```
    background-image: url("../images/bot.jfif");
    width: 50px;
    background-size: 50px;
}
.avatar-user{
    background-image: url("../images/cat.png");
    width: 50px;
    background-size: 50px;
}
```

打开前端页面，可以看到：



修改 JS 文件：在 static/js 下面 添加 JS 文件"mysearch.js"，添加问答提交相应事件。

代码：

```
    $(document).ready(function(){
        // 点击检索按钮，发请求
        $("#chatbotsendbtn").on("click", function () {
            var searchtext = $.trim($('#chattextarea').val());
            if (searchtext == "") {
               alert("请输入您的问题");
               return;
            }

            // 将问题添加到聊天窗口的末尾
            var question_html = '<div class="item item-right">'
                        + '<div class="bubble bubble-right">' + searchtext + '</div>'
                        + '<div class="avatar avatar-user"></div>'
                        + '</div>';
            $('.content').append(question_html);
            // 清空问题文本框
            $('#chattextarea').val('');
            $('#chattextarea').focus();
            // 滚动条置底
            var height = $('.content').scrollTop();
            $(".content").scrollTop(height);
```

```
            $.ajax({
                type: "get",
                url: "/searchanswer",
                data: {
                    "id": $("#chatbotsendbtn").attr("id"),
                    "text": searchtext
                },
                dataType: "json",
                beforeSend: function() {
                    // 设置 disabled 阻止用户继续点击
                    $("#chatbotsendbtn").attr("disabled", "disabled");
                },
                complete: function () {
                    // 请求完成移除 disabled 属性
                    $("#chatbotsendbtn").removeAttr("disabled");
                },
                success: function(result){
                    if(result.status == 200){
                        // 将答案添加到聊天窗口的末尾
                        var answer_html = '<div class="item item-left">'
                            + '<div class="avatar avatar-bot"></div>'
                            + '<div class="bubble bubble-left">' + result.answer + '</div>'
                            + '</div>';
                        $('.content').append(answer_html);
                        // 滚动条置底
                        var height = $('.content').scrollTop();
                        $(".content").scrollTop(height);
                        console.log("检索答案成功");
                    }
                    else{
                        // 将答案添加到聊天窗口的末尾
                        var answer_html = '<div class="item item-left">'
                            + '<div class="avatar avatar-bot"></div>'
                            + '<div class="bubble bubble-left">对不起！我不明白您的问题，可以换种问法吗？</div>'
                            + '</div>';
                        $('.content').append(answer_html);
                        // 滚动条置底
                        var height = $('.content').scrollTop();
                        $(".content").scrollTop(height);
                        console.log("检索不到答案");
                    }
                },
                error: function (jqXHR, textStatus, e) {
                    alert("提交异常："+e);
                }
            });
```

```
        });
    });
```

步骤 3 定义后面 Js 路由

在 movierecommendation/url.py 中定义路由：

```
#添加路由
    url(r'^questionAnswering/$', views.questionAnswering, name='questionAnswering'),
```

在 doubanrecommendation/url.py 中定义路由：

```
#添加路由
url(r'^searchanswer', views.searchanswer)
```

步骤 4 定义页面响应函数

在 movierecommendation/views.py 中定义响应函数（注意添加相应的包）：

代码：

```
    # 定义问答检索请求链接.
    @csrf_exempt
    def searchanswer(request):
        res = {
            'status': 404,
            'text': 'Unknown request!'
        }
        if request.method == 'GET':
            name = request.GET['id']
            if name == 'chatbotsendbtn':
                try:
                    # 获取前端的问题文本
                    text = request.GET['text']
                    # 检索问题，匹配答案
                    # 精确检索太苛刻了，如何实现近似问题检索？
                    qa_rec = QuestionAnswerKB.objects.get(question=text)
                    if qa_rec:
                        res = {
                            'status': 200,
                            'answer': qa_rec.answer
                        }
                    else:
                        res = {
                            'status': 201,
                            'answer': 'No answer!'
                        }
                except ObjectDoesNotExist:
                    res = {
```

```
                    'status': 201,
                    'answer': 'No answer!'
                }
    return HttpResponse(json.dumps(res), content_type='application/json')
```

问答效果如下：

实验小结

本实验我们设计并实现了问答系统和网络爬虫，主要介绍了前端页面、CSS 样式、JS 响应事件的实现和如何使用爬虫，以及后端基于检索的问答方法的实现。

思考题

接下来，请大家根据上面的学习内容，完成作业一的相关实验内容。