毕业设计（论文）

基于协同过滤的博客推荐系统

学　　院：数理科学学院

专业班级：信科 11901 班

学生姓名：吕亮亮

指导教师：刘杰

摘　要

随着互联网的快速发展，越来越多的人加入了博客创作和阅读的行列，使得博客平台上的内容呈现出爆炸式增长的趋势。然而，这也带来了一个新的挑战，即如何帮助用户在众多博客文章中找到他们感兴趣的内容。在这样的背景下，本文提出了一种基于协同过滤的博客推荐系统，以解决上述问题。协同过滤是一种常用的推荐算法，它利用用户行为数据和博客内容信息来进行个性化推荐。该系统首先收集用户的历史阅读记录和行为数据，如浏览记录、点赞和评论等，以了解用户的兴趣和偏好。然后，通过分析博客文章的内容特征，如主题、关键词等，建立起博客之间的相似性关系。基于这些信息，系统可以为每个用户生成个性化的推荐列表。我们使用 Python 语言进行开发，前端采用交互式的框架 Vue，后端采用功能完备的 Django 框架进行开发，并最终使用 Nginx 和 uwsgi 进行部署运行；在系统的实现过程中，为了提高推荐的准确性和多样性，我们还采用了一些改进策略。首先，引入了时间衰减因子，让新发布的文章也有充足的展示空间，其次，对用户的历史行为进行加权，以更好地反映用户的当前兴趣。在完成本项目后，最终实验结果表明，该基于协同过滤的博客推荐系统在提供个性化推荐方面具有良好的性能。与传统的热门文章推荐方法相比，该系统能够更准确地捕捉用户的兴趣和偏好，并为其推荐具有高度相关性和多样性的博客文章。

关键词：协同过滤算法； 推荐系统； 博客； 个性化推荐； Django； nginx； uwsgi

ABSTRACT

With the rapid development of the Internet, more and more people join the ranks of blog creation and reading, which makes the content on the blog platform show an explosive growth trend. However, this presents a new challenge in helping users find what they are interested in among the many blog posts. In this context, this paper proposes a blog recommendation system based on collaborative filtering to solve the above problems. Collaborative filtering is a common recommendation algorithm which makes use of user behavior data and blog content information to make personalized recommendation. The system first collects historical reading records and behavioral data, such as browsing history, likes and comments, to understand users' interests and preferences. Then, by analyzing the content features of blog articles, such as themes, keywords, etc., the similarity relationship between blogs is established. Based on this information, the system can generate a personalized recommendation list for each user. We use Python language for development. The front-end uses the interactive framework Vue, the back-end uses the fully-functional Django framework for development, and finally uses Nginx and uwsgi for deployment and operation. During the implementation of the system, in order to improve the accuracy and diversity of recommendations, we also adopted some improvement strategies. First, a time decay factor was introduced to allow ample room for new posts, and second, historical user behavior was weighted to better reflect users' current interests. After the completion of this project, the final experimental results show that the blog recommendation system based on collaborative filtering has good performance in providing personalized recommendation. Compared with traditional popular article recommendation methods, the system can more accurately capture users' interests and preferences and recommend highly relevant and diverse blog articles for them.

Keywords： collaborative filtering algorithm; Recommendation system; Blog; Personalized recommendation; Django; nginx; uwsgi

目　　录

一、引言

1.1 研究背景和意义

随着信息技术的不断进步，各式各样的互联网应用不断涌现，网站模式也从以文章为中心变成以用户为中心，用户逐渐喜欢了在网上创作和分享等，使得信息量呈指数级的爆炸增长，面对如此庞大的信息量，用户很难找到自己感兴趣的东西。

在这样的背景下，通过关键字搜索查找自己感兴趣内容的搜素引擎应运而生，在大多数情况下，用户往往不明白自己的需求是什么，或者在搜索时使用的关键字“词不达意”，从而无法得到自己想要的内容，同时，搜索引擎用户量巨大，往往考虑的不是某一个人，而是某一群人，但在实际生活中，用户 A 和用户 B 他们使用相同的关键词去搜索内容，他们得到的内容很大程度上是一样的，但是用户 A 和用户 B 他们原本想要表达的意思可能大相径庭，但是搜索引擎为了考虑大多数人群，从而导致无法满足个性化的需求[1]。

在这样情形下推荐系统应运而生。自 20 世纪 90 年代以来，以国外亚马逊、Ebay，国内淘宝、京东为代表的电子商务蓬勃 兴起，电子支付日益成熟，信息安全也不断完善，使推荐系统有个更大的发展空间 。在巨大市场需求的带动下，推荐系统在工业界和学术界获得了广泛使用，更多的国内外学者开始研究推荐系统，ACM、CCIR、SIGIR、RecSys 等众多国内外会议期刊收录的推荐系统相关 的文章越来越多。我国也比较重视推荐系统的研究和应用，国家自然科学基金支持很多推荐系统相关课题，中国推荐系统大会每年举办一次，汇集了国内一些走在前沿的推荐系统专家[2]。

博客是一种常见的网络媒体，它为用户提供了一个自由、开放、多元的信息交流平台。随着博客的数量不断增加，用户需要花费大量时间才能找到自己感兴趣的内容。因此，设计一个基于协同过滤算法的博客推荐系统，对于提高用户的使用体验和博客平台的活跃度具有重要意义。

二、相关技术和理论

2.1 协同过滤推荐算法

协同过滤是一种经典的推荐算法，它基于用户的历史行为和兴趣，寻找相似的用户或项目，并通过这些相似性来预测用户可能感兴趣的内容。本文中，我们将使用基于用户的协同过滤算法，其中我们将计算用户之间的相似度，并推荐给用户类似于他们历史上阅读过的博客。

协同过滤算法的核心思想是根据用户的行为历史（例如，点击、购买、评论等）来推荐他们可能感兴趣的物品。在博客推荐系统中，我们可以根据用户对博客的浏览历史和收藏历史等信息，计算用户之间的相似度，并将这些相似用户的浏览历史作为推荐给目标用户的博客[3]。

2.2 Python 语言介绍

Python 是一种高级、解释性、面向对象的编程语言，被广泛应用于科学计算、数据分析、人工智能、Web 开发等领域。Python 具有简单易学、代码可读性高、语法简洁、跨平台等特点，是一种非常流行的编程语言。

Python 的语法简洁清晰，使用缩进来表示代码块，避免了繁琐的语法规则。同时，Python 支持面向对象编程，具有封装、继承和多态等特性，可以方便地组织代码，使得程序具有更好的可维护性和可扩展性。

Python 有丰富的 Web 开发框架和库，如 Django、Flask 等，可帮助开发者快速搭建 Web 应用程序，同时提供高级功能和工具，如 ORM、数据库访问、模板引擎等，开发 Web 应用更加简单快捷。Python 的语法简洁、库和框架丰富，可提高开发效率，是 Web 开发的优选语言之一。

2.3 Vue 框架介绍

Vue.js 是一款流行的前端 JavaScript 框架，可以用于构建交互式的 Web 界面。Vue.js 的核心思想是通过组件化来构建应用，每个组件都具有独立的状态

和行为，可以相互嵌套和组合，从而构建出复杂的应用程序。Vue.js 还提供了响应式数据绑定、虚拟 DOM、模板语法、生命周期钩子、指令等功能，使得开发人员能够更加高效地构建 Web 应用。

Vue.js 的响应式数据绑定能力是其最大的特点之一。当应用状态改变时，Vue.js 会自动更新页面上对应的 DOM 元素，使得应用更加动态、高效。同时，Vue.js 还支持虚拟 DOM 技术，可以减少 DOM 操作的次数，提升应用性能。Vue.js 还提供了一套简单易用的模板语法，可以方便地将组件的数据和视图进行绑定，使得数据绑定更加方便。

2.4 Django 框架介绍

Django 是一个用于构建 Web 应用程序的高级 Python Web 框架。它提供了一种快速而高效的方式来构建 Web 应用程序，并且可以轻松地与 Python 的各种科学计算和机器学习工具进行集成。在本文中，我们将使用 Django 框架来实现我们的博客推荐系统。

Django 框架可以大大简化 Web 应用程序的开发过程，它提供了许多强大的功能，如用户认证、ORM、模板引擎等，使得开发者可以专注于业务逻辑而不必过多考虑底层细节。此外，Django 还支持各种数据库和缓存后端，可以灵活地适应各种场景。

2.5 nginx 介绍

Nginx 是一种高性能的 HTTP 和反向代理服务器，也可用于邮件代理服务器和通用的 TCP/UDP 代理服务器。Nginx 在高并发量的情况下表现出色，可同时处理数百万请求，并提供高可用性和可伸缩性。

Nginx 具有简单的配置和灵活的扩展性，能够轻松处理虚拟主机、负载均衡、SSL 协议和缓存等功能。Nginx 还能够运行在多种操作系统上，并且使用非常简单，可以在短时间内安装、配置和使用。

Nginx 的反向代理功能可以将客户端的请求转发到后端的 Web 服务器上，实现负载均衡和提高性能。同时，Nginx 还支持 FastCGI、uWSGI、SCGI 等协议，

可以将 Web 服务器与应用程序分离，使得 Web 服务器更加灵活和可扩展[4]。

2.6 uwsgi 介绍

uWSGI 是一种 Web 服务器和应用程序容器，它支持多种协议和语言，如 WSGI、HTTP、FastCGI、uWSGI 等，可以与多种 Web 服务器、应用服务器、应用程序框架和语言库配合使用。

uWSGI 可以提高 Web 应用程序的性能和可靠性，它的运行模式可以根据需要选择多进程、多线程、协程等模式，支持内存共享和多进程管理，还能够实现负载均衡、快速部署和容器化等功能。

三、博客推荐系统设计和实现

3.1 需求分析

3.1.1 功能需求

用户注册/登录功能：用户可以注册一个账号，进行登录。

个人信息管理功能：用户可以修改个人信息，如昵称、头像、个人简介等。

博客发布功能：用户可以发布博客，包括标题、正文、标签等信息，并选择是否公开。

博客编辑功能：用户可以对自己发布的博客进行编辑，包括标题、正文、标签等信息。

博客删除功能：用户可以删除自己发布的博客。

博客点赞功能：用户可以对其他用户发布的博客进行点赞。

博客评论功能：用户可以对其他用户发布的博客进行评论。

博客收藏功能：用户可以收藏其他用户发布的博客。

博客推荐功能：系统根据用户的历史行为，为用户推荐可能感兴趣的博客。

3.1.2 非功能需求

安全性：系统需要对用户信息进行保护，如密码加密存储、防止 SQL 注入等。

可靠性：系统需要具备较高的稳定性和可靠性，确保用户数据不会因系统故障而丢失。

可用性：系统需要易于使用，用户体验友好。

性能：系统需要具备较高的性能，如快速响应用户请求、较短的加载时间等。

扩展性：系统需要具备良好的扩展性，能够满足用户需求的增长。

3.1.3 技术需求

语言：系统采用 Python 语言开发。

框架：系统前端采用 Vue 框架，后端采用 Django 框架，使用 MVC 的开发模式。

数据库：系统采用 MySQL 数据库存储用户数据和博客数据，采用 Redis 做数据缓存。

推荐算法：系统采用协同过滤算法作为博客推荐算法。

3.2 概要设计

3.2.1 系统整体框架

用户在请求页面后,请求首先会被代理服务 nginx 捕获,在事先定义好 nginx 和 uwsgi 之间的网关协议后，如果是请求的是当前的博客系统，nginx 会将请求转发给 uwsgi，uwsgi 再去请求的具体的应用，也就是当前的博客系统，在博客系统中，首先会将这个请求封装成一个请求对象，进入系统后首先会来到中间件层，如果在中间件中没有对请求执行响应操作，之后就会流入到业务层（model，view 和 templates），进入业务层，首先会根据请求地址由 view 层负责进行路由分发，将请求分发到具体的视图函数中进行处理，再由视图函数和数据层进行交互，在处理完业务逻辑后，通过 templates 层对返回结果渲染，打包成一个响应对象，最后再进行返回。需要注意的是，在 model，templates 和 view 处理整

个业务流程时，中间件是一直贯穿其中的。系统应用层整体采用 MTV 框架模式，即模型 Model，视图 View 和模版 Templates。同时使用 redis 数据库用作缓存，前后端数据交互使用 form 表单以及异步传输数据的 ajax 进行[5]。系统的整体架构图如下所示:

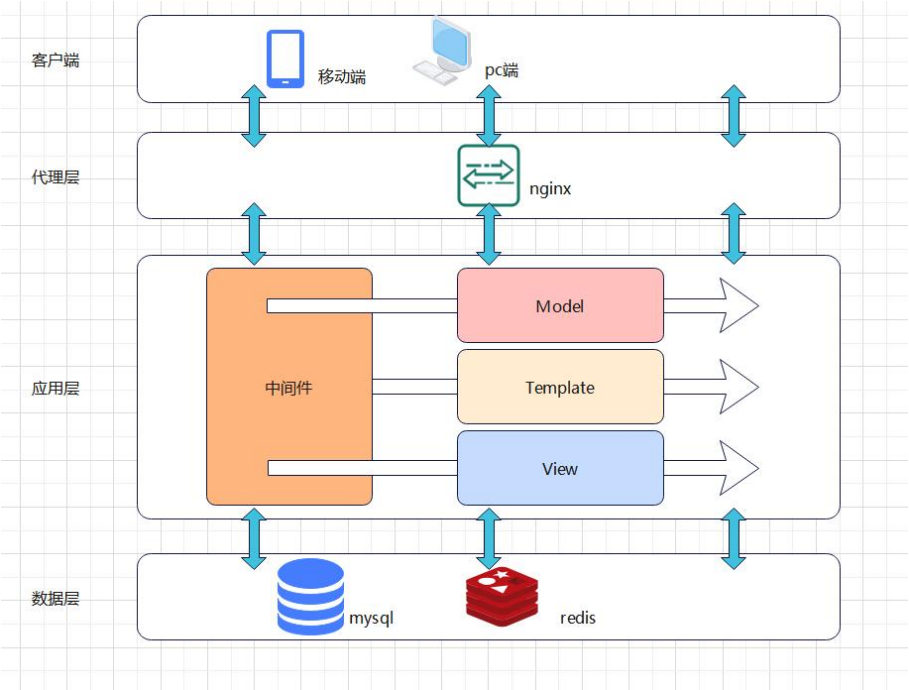

图 1. 系统架构图

3.2.2 功能模块划分

根据功能的不同，可以系统整体分为如下 7 大模块，分别是用户模块，博客模块，交互模块，筛选模块，统计模块，管理模块，以及推荐模块。

用户模块：用户模块包含的主要功能是注册，登录和修改密码，其核心功能就是身份验证，将需要登录和不需要登录才能使用的功能区分，同时，用户模块在推荐模块中也起着一定的作用，对于登录后的用户能够唯一标识，精确记录操作历史，从而实现更好的推荐效果。

博客模块：博客模块中，主要涉及的功能就是对博客的增删改查，除了显示博客的功能，其他功能都依赖用户模块，例如：写博客，更新博客和删除博客都需要登录后才能进行；

交互模块：交互模块包含了点赞，收藏，留言和评论功能，是用户和博主进

行交流的功能模块。

筛选模块：为了方便用户查找想要的内容，给用户提供检索功能，分为关键词检索，分类检索和标签检索。

统计模块：统计模块包含的主要功能有排行榜统计和访问量统计，在排行榜中统计了天/周/月热度最高的博客，同事对于整站的访问量后台以及用户的行为也会进行统计分析。

管理模块：在管理模块中，主要包含的功能是用户管理，博客管理和权限管理，是面向管理员的，操作能够对整站产生影响。在用户管理模块中，可以进行新增，删除，修改用户等操作，同时能够对用户的权限进行具体的划分，比如：对某一个模型的查询，删除，更新等功能都能明确划分；在管理模块中，能够对整站的博客进行管理；在权限管理中，能够划分权限组，为用户授权或添加到特定的权限组等功能。

推荐模块：根据统计模块中统计到的用户行为数据，应用协同过滤算法进行计算得出推荐列表。功能模块划分图如下：

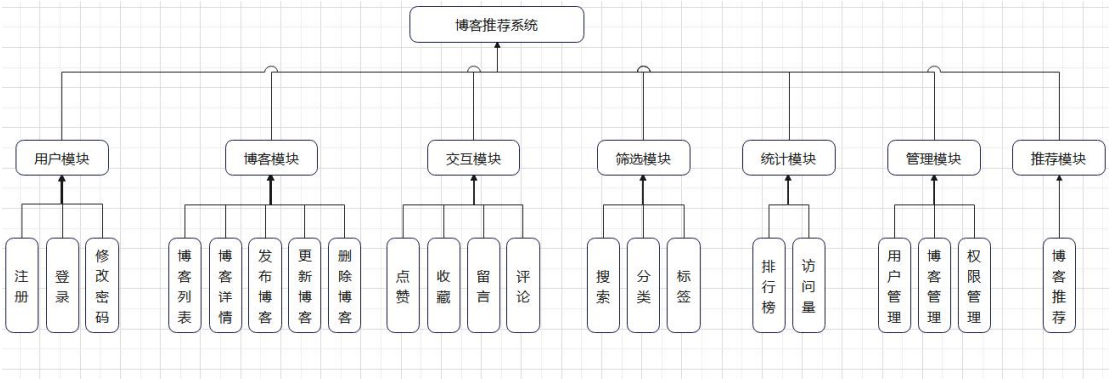

图 2.功能模块图

3.2.3 模型设计

根据需求分析，可以设计以下数据库表：

用户表(user)，博客表(blog)，博客分类表(category)，留言（message），评论（comment），收藏（collection），点赞（like），搜索记录表（search），推荐表（recommend），友链（friendlink），文件表（file），请求记录表（request record），用户操作记录表（action），错误记录表（error），它

们之间的关系如下图所示：

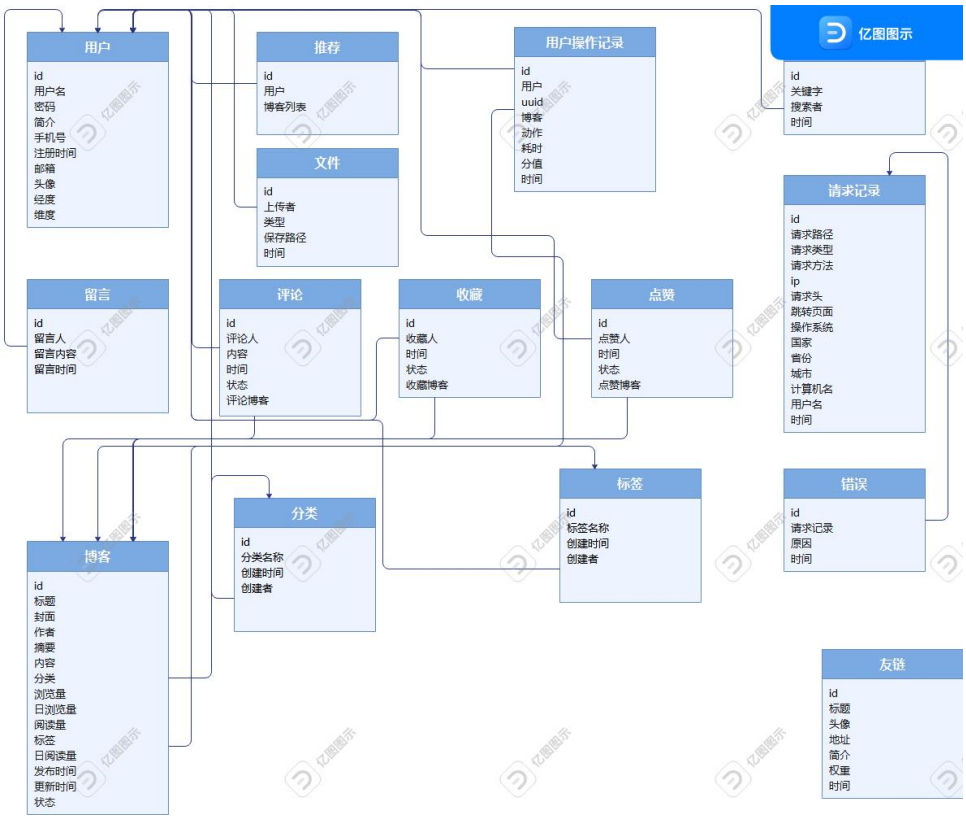

图 3. 模型的 ER 图

3.3 使用 Django 框架实现博客系统

3.3.1 用户相关

3.3.1.1 注册

用户可以通过手机号注册一个账号。注册后可以使用登录功能登录网站，在登录成功后，能够使用发布博客，点赞，评论，留言等功能，同时对于登录的用户，可以更精确地对用户行为进行记录，从而为用户提供更好的个性化服务。

实现过程

在访问注册页面后（此时使用 get 方法），首先会获取一个表单，用户通过表单输入相关信息，点击“注册”按钮，前端会进行输入合法性的校验，在校验通过后，会通过 Ajax 将表单信息以 post 的方法异步传输给后端；来到后端后，

首先通过 request 对象获取传输的信息，之后还会做一系列的检查（包含：手机号格式检查，手机号重复检查，验证码过期检查，验证码正确性检查等），虽然其中的一些检查在前端已经校验过，但是在实际生活中危险是无处不在的，除了一般的用户会使用浏览器正常进行访问，一些技术相关的人员可能会通过特定的工具进行访问，所以为了防止接口的恶意攻击，提高服务的安全性是非常有必要的。

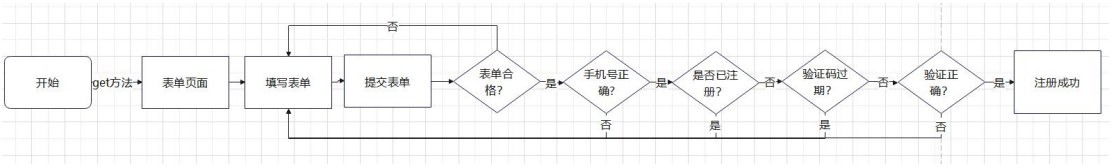

图 4.注册流程

3.3.1.2 登录

用户可以使用注册时提供的用户名和密码登录网站。登录后可以享受更多的功能，例如评论、点赞、收藏、博客编辑等。

实现过程

在访问登录页面后（同样使用的是 get 方法），用户会获取到表单页面，输入相应的表单信息，点击“登录”即可，请求来到后端后，会检查用户名是否存在以及密码的正确性，如果检查不通过会响应相应的提示给前端，检查通过后会使用 django 内置的 login 方法实现用户的登录。使用 django 内置的登录功能的主要原因有两点：

1. 便利性

使用 django 内置的 login，不需要自己额外去保存相关的信息，比如缓存，登录状态等，并且可以直接使用。

2. 兼容性

由于用户表是直接继承自 django 内置的 User 模型，所以用户相关的操作也必须使用 django 内置的，否则相关功能不会生效；

3.1.1.3 更新密码

用户可以通过提供注册时使用手机号码来重置密码。这是当用户忘记密码时，能够重新设置密码的方法（注意：密码重置功能每天仅限 3 次）。

实现过程

继承（Inheritance）是面向对象编程中的一个重要概念，它指的是一个对象（称为子类或派生类）从另一个对象（称为父类或基类）那里继承了一部分属性和方法，从而可以直接使用这些属性和方法，而不必重新定义。继承的关系可以理解为“is-a”的关系，即子类是父类的一种特殊情况。除了 python 语言自带的“继承”这个概念，在 django 的模板中，也同样有继承这个概念，对比“注册”（参考下图左）和“更新密码”（参考下图右）所需要的参数信息，可以发现基本上是一样的，但是为了区别“注册”功能，在“更新密码”功能中额外加入一个标识参数进行标识即可；因此“更新密码”的模板直接继承自“更新密码”注册的模板即可。

图 5. 注册（左）

图 6. 更新密码（右）

后端的业务逻辑也同样使用“注册”的接口，只不过需要根据标识参数的值来执行不同的业务逻辑。

3.3.2 博客相关

3.3.2.1 博客列表

用户可以查看所有的博客文章列表，列表会根据用户的操作历史记录，博客的热度和发布日期等因素计算出的得分进行排序，从而为用户推荐感兴趣，质量高，且新鲜的博客。

实现过程

博客列表的实现可以大致分为三部分：1.获取所有博客对象，2.计算推荐度，3.分页。

其中步骤 2 就是应用协同过滤推荐算法到博客系统的具体步骤，因此，在应用之前，一个简单的博客系统只需要步骤 1 和步骤 3 即可，由于协同过滤算法会在下面进行详细的讲解，所以这里就对步骤 1 和步骤 3 进行讲解。

步骤 1 的实现非常简单，直接通过“Blog 模型“实例对象调用相应的接口即可获取查询集，我这里用到的是 only 接口，映射到 sql 语句时，就是只查询需要的字段，而不是通过“*”匹配所有的字段，这样在一定程度上提高了查询效率。

图 7.博客列表的 only 查询语句

步骤 3 的实现则使用到了 django 内置的 Paginator 对象，将步骤 1 中得到的所有博客对象以及每页的数量作为参数传给 Paginator 对象，就会生成相应的页面对象，从而达到分页的效果。

3.3.2.2 博客详情

用户可以查看单篇博客的详情内容，并进行点赞、收藏、评论等操作。这个功能可以让用户更深入地了解某个主题或文章。

实现过程

在博客详情页面中，以内容的展示为核心功能。通过博客的 id 可以找到指定的博客对象，因此每一篇博客都唯一对应一个页面。用户在请求一个博客的详情页面后，后端会首先判断用户的登录状态，从而获取“收藏”，“点赞”的状态，同时判断对当前博客的编辑权限；之后会统计当前的博客的访问量（包含 pv，uv，dpv，duv），以及记录当前用户的访问记录，最后将内容返回给前端。

来到前端页面后，页面的大体布局如下图所示：顶部为导航栏，左下为博客内容，右下为功能页。由于在开发过程中一直使用的是 pc 端进行开发，没有考虑到移动端（显示宽度较小的设备），因此在使用移动设备进行查看时，页面内容会显示异常。

图 8.《关于》博客详情页面-PC 端展示

为了解决移动端和 PC 端的适配问题，后来用到了 bootstrap 的栅格系统和 JavaScript 中的事件监听方法。

栅格系统（Grid System）：Bootstrap 的栅格系统是一种响应式的布局方式，将页面分成 12 个等宽的列，可以根据需要将页面划分为不同数量的列，实现不同的布局效果。栅格系统提供了多种列宽度的选项，可以通过添加 CSS 类名来实现。

JavaScript 中的事件监听（Event Listening）指的是在页面元素上注册事件处理程序（Event Handler），当该页面元素上发生特定的事件时，事件处理程序就会被调用，从而实现对事件的响应。事件监听是实现 JavaScript 交互性和动态效果的重要方式。在 JavaScript 中，可以使用 addEventListener()方法或者 DOM 元素属性来添加事件监听器[6]。

有了这两个方法后，就可以根据访问设备的显示宽度进行分开显示，当显示宽度大于 1000px 时，页面内容进行“3-1”的划分，博客内容占用 9 列，功能页占用 3 列；当页面宽度小于 1000px 时，页面只显示博客内容，将功能页进行隐藏，然后使用 JavaScript 的事件监听，将监听窗口的加载（load）和重置大小（resize)方法注册到 window 对象上，将上面的逻辑添加到 EventHandler 即可，如下：移动端正常显示博客内容。

图 9.《关于》博客详情页面-移动端展示

3.3.2.3 博客编辑

博客作者可以通过编辑页面对已有的博客进行修改或删除，也可以新增博客。这个功能可以让博客作者更方便地管理和更新自己的博客。

实现过程

博客编辑的功能实现可以分为两部分：1.请求编辑页面，2.保存博客数据。

当用户使用的是 GET 请求时，返回博客编辑页面，需要注意的是，因为博客需要关联到作者，因此必须要求用户登录后才能使用此功能，为了减小开发和维护的成本，使用装饰器实现了登录监测功能。

装饰器（Decorator）是一种用于修改类、函数、方法或属性的语法特性。装饰器可以在不修改类或函数源代码的情况下，为其添加新的功能或修改原有功

能。在多数情况下，一段代码可能关联很多模块，但是为了开发新的功能需要对其进行修改，此时如果直接修改可能需要很大的时间和人力成本，那么装饰器就是很好的选择，这是选择装饰器的原因之一；当很多接口都有相同的逻辑，如果每一个接口都去重复编写这段逻辑，不仅编写起来耗时，维护起来也会更加的耗时，因此如果将这些相同的逻辑封装并打包成一个装饰器，最后装饰在每一个接口上，那么可以大大减少开发成本和维护成本，这是选择装饰器的原因之二，也是我在本次博客系统中选择装饰器实现登录检测功能的原因，考虑到多个接口：编辑博客，点赞，收藏，评论等接口几乎都需要进行登录状态的检查，因此将其封装成一个装饰器之后装饰这些接口很大程度上减少了我开发和维护的成本，具体代码逻辑如下：

```
def require_login(require_self=False):
    def outer(func):
        def inner(*args, **kwargs):
            # 根据第一个参数是否是视图函数的实例来取 self 和 request 的值
            if isinstance(args[0], View):
                self = args[0]
                request = args[1]
            else:
                request = args[0]
            if not request.user.is_authenticated:
                return JsonResponse({
                    'code': '400',
                    'msg': '请先登录!'
                })
            if require_self:
                tmp_user_id = request.data.get('user') or request.POST.get('user') or reque
st.GET.get('user')
                if str(request.user.id) != tmp_user_id:
                    return JsonResponse({
                        'code': '400',
                        'msg': '非本人操作!'
                    })
            try:
                res = func(*args, **kwargs)
            except Exception as e:
                print(e)
                return None
            return res
        return inner
```

在这个函数中，首先定义一个 outer 函数作为装饰器的外层函数，该函数接收一个参数 func，表示要被装饰的函数。在 outer 函数中定义一个 inner 函数作为装饰器的内层函数，该函数接收任意数量的位置参数 args 和关键字参数 kwargs，这些参数将被传递给被装饰的函数。在 inner 函数中首先根据第一个参数的类型来取得 self 和 request 的值。如果第一个参数是视图函数的实例，则 self 为第一个参数，request 为第二个参数；否则，self 为 None，request 为第一个参数，在这里做 self 和 request 的检测的原因主要是为了兼容基于函数的视图函数和以及 restframework 中基于类的视图方法。之后调用 User 实例的 is_authenticated 方法检测用户的登录状态，如果未登录，则返回一个 JsonResponse，提示用户先登录。如果 require_self 为 True，则检查请求中的用户 id 与登录用户的 id 是否一致，如果不一致，则返回一个 JsonResponse，提示用户非本人操作。如果用户已登录，并且 require_self 为 False 或者 require_self 为 True 且操作为本人操作，则调用被装饰的函数，并将其返回值返回。如果调用被装饰的函数发生异常，则打印异常信息，并返回 None。

Vditor 是一款浏览器端的 Markdown 编辑器，支持所见即所得、即时渲染（类似 Typora）和分屏预览模式。它使用 TypeScript 实现，支持原生 JavaScript 以及 Vue、React、Angular 和 Svelte 等框架。考虑到不同的用户有不同的博客编辑习惯，而市面上常用的所见即所得，即时渲染和分屏预览这几种模式它都能够支持，同时能够任意切换，并且也支持我前端使用的 Vue 框架，因此我使用了它作为了博客内容编辑的工具，博客编辑页面如下：

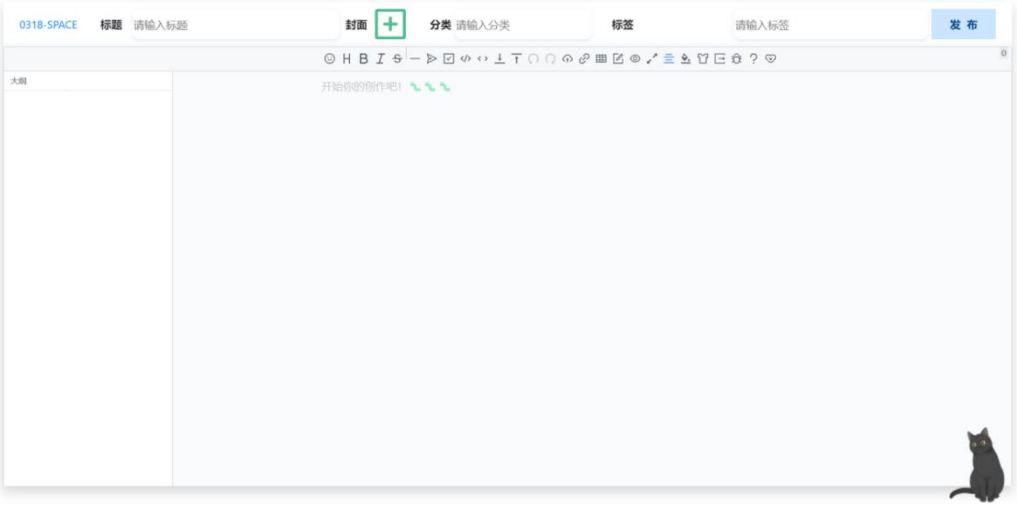

图 10. 博客编辑页面

当用户使用的是 POST 请求时，说明用户返回的是编辑后的博客数据，此时就对博客内容进行保存。

3.3.3 交互相关

在交互相关的功能中，默认都需要用户登录后才能使用，因此在下面的讲解过程中，默认都是用户已登录的状态（登录状态的检测已在上述进行讲解）。

3.3.3.1 点赞

用户可以对某篇博客进行点赞，点赞数会在博客详情页和博客列表页展示。这个功能可以让用户表达对某篇博客的喜爱程度。

实现过程

用户在点击“点赞”的图标后，根据当前点赞的状态，后端会做相反的操作：点赞->取消点赞，未点赞->点赞，每一个用户对于每一篇博客都只会产生一条点赞记录，点赞的状态会根据这条记录中的“status”属性来标记；对于点赞量的计算这统计这些点赞记录即可获得。

收藏

用户可以将某篇博客加入自己的收藏夹中，以便以后查看。这个功能可以让用户更方便地找到自己感兴趣的博客。

实现过程

收藏功能本质上和点赞功能是一样的，因此实现过程同点赞功能。

3.3.3.2 评论

用户可以对某篇博客进行评论，并查看其他用户的评论。这个功能可以让用户表达对博客的看法或提出问题，也可以让用户更深入地了解某个主题或文章。

实现过程

评论功能中包含了获取评论列表，新增评论和删除评论的功能。在实现这些功能的过程中主要使用到了 django-rest-framework 框架。

restful 是一种网络应用程序的设计风格和开发方式，基于 http，可以使用 xml 格式定义或 json 格式定义。restful 适用于移动互联网厂商作为业务接口的场景，实现第三方 ott 调用移动网络资源的功能，动作类型为新增、变更、删除所调用资源。django-rest-framework 则是便于开发者开发快速开发 restful 接口的一个工具，简单来说，django-rest-framework 的作用等同于 django 中的 view+form，我们既可以基于 model 来直接生成接口，也可以自定义 serializers（跟 form 的用法很像）的字段来生成接口。在这里我使用了自定义 serializers 的方法来生成接口，然后在视图类中继承自 django-rest-framework 中的 ModelViewSet 类，覆写其中的 list，destroy 和 create 方法来实现评论的获取，删除和新增功能。

图 11. 博客评论

3.3.4 筛选相关

3.3.4.1 搜索

用户可以通过关键字搜索博客文章，如果查找到相关博客，后台也会根据用

户的操作历史等数据对博客进行排序后展示给用户。

实现过程

用户输入关键词，点击“Search”按钮后，数据来到后端时，首先会将关键词进行一个分词操作，之后根据关键词生成相应的 sql 语句在数据库中进行查找，将查找到的结果传给 Paginator 对象，进行分页之后再返回，如果没有查到则会提示相应的信息。

对关键词的分词使用了 jieba 库中的 lcut 方法，它的分词原理主要基于基于“前缀词典”、“后缀词典”、“中间词典”和“未登录词典”的分词方法，以及基于概率的统计模型，对分词后的结果再进行一次“去停用词”操作后得到关键词列表 keyword_list。

在查询时由于关键词可能有多个，使用 Django 自带的 ORM 功能无法很好的表达出这个并列关系，因此使用了自定义的原生 SQL 语句来实现，在 SQL 语句中，使用了 like 模糊查询，在标题和博客内容中进行模糊查询，并使用 order by 语法将结果按照 pv 值进行排序，相关逻辑代码如下：

```
# 搜索
@classmethod
def search(cls, keyword_list: List[str]):
    sql = f'SELECT * FROM 博客 where 文章是否已删除="否" and 标题 like "%{keyword_list[0]}%" or 文章内容 like "%{keyword_list[0]}%"'
    for keyword in keyword_list[1:]:
        sql += f' or 标题 like "%{keyword}%" or 文章内容 like "%{keyword}%"'
    # 按照浏览量排序
    sql += ' order by pv'
    return Blog.objects.raw(sql)
```

图 12.“搜索”功能的代码逻辑

3.3.4.2 分类

博客文章可以被归为不同的分类。用户可以通过分类来浏览相关的博客文章。这个功能可以让用户更方便地找到自己感兴趣的博客。

实现过程

“分类”的逻辑和“搜索”的逻辑相似，但是区别在于：“分类”在进行数据查找时，使用的是 Django 的 ORM 功能进行查找，筛选条件从关键词查询，变为了类别查询，并且是精确查询。

3.3.4.3 标签

用户可以通过标签来筛选相关的博客文章。这个功能可以让用户更方便地找到与某个主题相关的博客。

实现过程

“标签”的逻辑和“分类”的相同，区别在于查询条件从分类改为了标签。

3.3.5 统计相关

3.3.5.1 排行榜

用户可以查看博客文章的排行榜，排行榜是根据全站用户对博客的点赞、评论、浏览等进行统计后排序的结果，排行榜会分为日排行榜，周排行榜和月排行榜。这个功能可以让用户了解哪些博客文章比较受欢迎。

实现过程

一次统计的过程为：获取最近一个月内所有博客页面的请求，统计这一个月中每篇博客的访问量；再从这一个月中获取最近一周所有博客页面的请求，统计这一周中每篇博客的访问量；最后再从这一周中获取最近一天所有博客页面的请求，统计这一天中每篇博客的访问量；对统计后的结果按照访问量进行排序，就获得了每月/周/日的 topk 博客列表。

由于一次统计耗时较长，因此为了不影响用户的体验感，这里使用了周期任务（使用 APScheduler 库实现的）和 Redis 缓存。将上上面的一次统计过程看做一个任务，使用 APScheduler 库提供的功能来周期（目前周期为 1 小时）执行，每次执行完毕后将结果保存在 Redis 中，使用时，直接从 Redis 中取即可。

3.3.6 文件相关

3.3.6.1 上传文件

用户可以上传文件，例如图片、音频、视频等，用于在博客文章或者用户头像等功能中使用。这个功能可以让博客作者更方便地插入多媒体素材。

实现过程

“文件上传”功能就是对上传的文件进行保存然后返回相应的地址，以方便用户的使用。但是考虑到服务器的承载压力，比如：文件过大，在上传和下载时都会给服务器造成很大的网络压力；以及文件的安全性，比如：同一时间，两个用户同时上传了两个文件，它们内容不同，但是文件名是相同的，那么后上传完成的文件可能会覆盖先上传完成的文件，从而导致文件错误。所以做了如下措施：

1.根据文件的类型限制文件的大小

表 1.文件类型-大小关系表

文件类型	Max size
Image	2M
Audio	50M
Video	200M
Text	20M
Other	200M

2.基于时间戳为每次上传的文件生成唯一的文件名

如下图所示：首先使用内置库 uuid 中的 uuid1 方法（uuid 是一种由 128 位二进制数表示的唯一标识符，它可以用来识别具有不同标识符的物体或信息。在 Python 中，uuid 模块提供了生成 uuid 的功能，其中 uuid.uuid1 生成的 uuid 是基于时间戳和主机 MAC 地址生成的。）基于时间戳生成一段唯一标识，然后再将这段唯一标识对应的 16 进制的字符串，将其和原文件名得到新的文件名，最后再对文件进行保存。

图 13.“生成新的文件名”

3.3.7 管理相关

3.3.7.1 管理台

管理员可以通过管理台来管理网站的博客文章、用户账号等信息，例如审核博客文章、禁言用户等。这个功能可以让管理员更方便地管理网站。

实现过程

由于 Django 已经实现了功能完备的后台管理功能，我们在基于 Django 自带管理台的基础上上，使用第三方库 simpleui 对其功能做了进一步完善。

3.3.8 总结

至此，博客系统的用户模块，博客模块，交互模块，筛选模块，统计模块，文件模块和管理模块共 7 大模块的功能基本开发完成，用户模块负责用户的注册、登录、更新密码等操作，为博客系统提供了安全的身份验证机制；博客模块负责博客的创建、编辑、删除等操作，是博客系统的核心功能之一；交互模块负责博客的评论、点赞等操作，增加了用户之间的互动；筛选模块提供了按标签、按分类等多种方式对博客进行筛选的功能；统计模块负责统计博客的访问量等数据，为博客系统的运营提供了依据；文件模块负责用户上传的图片、音频等文件的管理；管理模块为管理员提供了对用户、博客、评论等内容的管理权限，保证了博客系统的安全性和稳定性。每个模块都有其各自的核心功能，通过模块之间的相互调用就形成了整个博客系统的功能。

3.4 收集用户数据

根据操作性质的不同，用户数据的收集采用了前端和后端同时进行。所有的用户行为包括：点击，阅读，点赞，取消点赞，收藏，取消收藏，评论，删除评论，打赏（预留行为，当前版本未开发）。其中由于“阅读”行为是持续性的，而前后端交互使用的是 http 协议，每次请求都是一次性的，所以无法在一次请求后获取到完整的结果，因此必须由前端监听该行为的状态和结果的返回；其他行为则是瞬时性的，直接在后端接口中添加相应的行为记录即可。

用户执行不同的行为说明用户对当前博客的喜爱程度是不一样的。比如：当用户在阅读某一篇博客时，发现里面的内容和自己所学的内容完全一样，并且在自己所学的内容上还做了进一步拓展，于是用户就激动地对这篇文章进行点赞和收藏，但是过了一段时间后，用户再次阅读这篇文章时，由于用户自己知识的增长，觉得这篇文章写的很浅，于是取消了收藏。为了表示用户不同行为表达对博客的喜爱程度，我们对用户的行为进行了打分，打分越高，则表示该用户对博客的喜爱程度越高。相关“行为-打分”表如下：

表 2.用户行为-打分表

行为	分数	耗时相关
点击	1	否
阅读	5	是
点赞	3	否
取消点赞	-2	否
收藏	4	否
取消收藏	-3	否
评论	2	否
删除评论	-1	否
打赏	5	否

注：其中“阅读”行为是持续性行为，因此，它的打分和耗时长短有关。计算方法为：每篇博客会根据内容计算出预计阅读时长 read_time，假设当前阅读

时长为 time1，则当前阅读后的打分 score=min(time1/read_time*5，5)。

使用 uuid 标识每一位用户。用户在登录的情况下，可以直接使用登录的这个身份来记录他所有的行为数据；但是如果用户没有登录，或者说账号都没有注册，那么要保存并保证用户的行为记录和用户他本身唯一对应，就必须通过手段为用户生成一个唯一标识来表示这个用户。这里我采用了 uuid 来生成用户的唯一 id，并通过 cookie 保存在客户端。因此在保存行为数据时，既保存了用户的登录信息，也保存了此时的 uuid，部分行为数据如下：

ID	用户	UUID	博客	行为	耗时	分值	时间
894	-	690e6609-a44d-4bdd-934e-3677b4425019	python后端面试笔记，祝愿秋招拿到满意的offer。	阅读	45.5	0	2023年4月14日 00:45
893	-	690e6609-a44d-4bdd-934e-3677b4425019	python后端面试笔记，祝愿秋招拿到满意的offer。	阅读	8.8	0	2023年4月14日 00:44
892	-	690e6609-a44d-4bdd-934e-3677b4425019	python后端面试笔记，祝愿秋招拿到满意的offer。	点击	0.0	1	2023年4月14日 00:44
891	-	690e6609-a44d-4bdd-934e-3677b4425019	python后端面试笔记，祝愿秋招拿到满意的offer。	点击	0.0	1	2023年4月14日 00:44
890	-	-	python后端面试笔记，祝愿秋招拿到满意的offer。	点击	0.0	1	2023年4月14日 00:44
889	-	-	python后端面试笔记，祝愿秋招拿到满意的offer。	阅读	1.3	0	2023年4月14日 00:44
888	-	9adb4b05-1b3b-4ef0-becf-eae0153ab393	python后端面试笔记，祝愿秋招拿到满意的offer。	阅读	8.7	0	2023年4月14日 00:43
887	-	9adb4b05-1b3b-4ef0-becf-eae0153ab393	python后端面试笔记，祝愿秋招拿到满意的offer。	点击	0.0	1	2023年4月14日 00:43
886	-	9adb4b05-1b3b-4ef0-becf-eae0153ab393	python后端面试笔记，祝愿秋招拿到满意的offer。	点击	0.0	1	2023年4月14日 00:43
885	-	9adb4b05-1b3b-4ef0-becf-eae0153ab393	python后端面试笔记，祝愿秋招拿到满意的offer。	点击	0.0	1	2023年4月14日 00:43
884	-	9adb4b05-1b3b-4ef0-becf-eae0153ab393	golang学习笔记系列	阅读	3.4	0	2023年4月14日 00:43
883	-	9adb4b05-1b3b-4ef0-becf-eae0153ab393	golang学习笔记系列	点击	0.0	1	2023年4月14日 00:43
882	-	9adb4b05-1b3b-4ef0-becf-eae0153ab393	golang学习笔记系列	点击	0.0	1	2023年4月14日 00:43
881	-	9adb4b05-1b3b-4ef0-becf-eae0153ab393	golang学习笔记系列	点击	0.0	1	2023年4月14日 00:43
880	-	95508b86-7951-498c-899a-de7cc97563dd	mysql创建新用户	阅读	0.0	0	2023年4月10日 23:40

图 14. 部分用户行为数据

3.5 使用 Python 实现协同过滤算法

协同过滤，从字面上理解，包括协同和过滤两个操作。所谓协同就是利用群体的行为来做决策(推荐)，生物上有协同进化的说法，通过协同的作用，让群体逐步进化到更佳的状态。对于推荐系统来说，通过用户的持续协同作用，最终给用户的推荐会越来越准。而过滤，就是从可行的决策(推荐)方案(标的物)中将用户喜欢的方案(标的物)找(过滤)出来[7]。

协同过滤分为基于用户的协同过滤和基于标的物(物品)的协同过滤两类算法。它们的核心思想是很朴素的”物以类聚、人以群分“的思想。所谓物以类聚，就是计算出每个标的物最相似的标的物列表，我们就可以为用户推荐用户喜欢的标的物相似的标的物，这就是基于物品(标的物)的协同过滤。所谓人以群分，就是我们可以将与该用户相似的用户喜欢过的标的物的标的物推荐给该用户(而该

用户未曾操作过)，这就是基于用户的协同过滤。具体思想可以参考图 15。

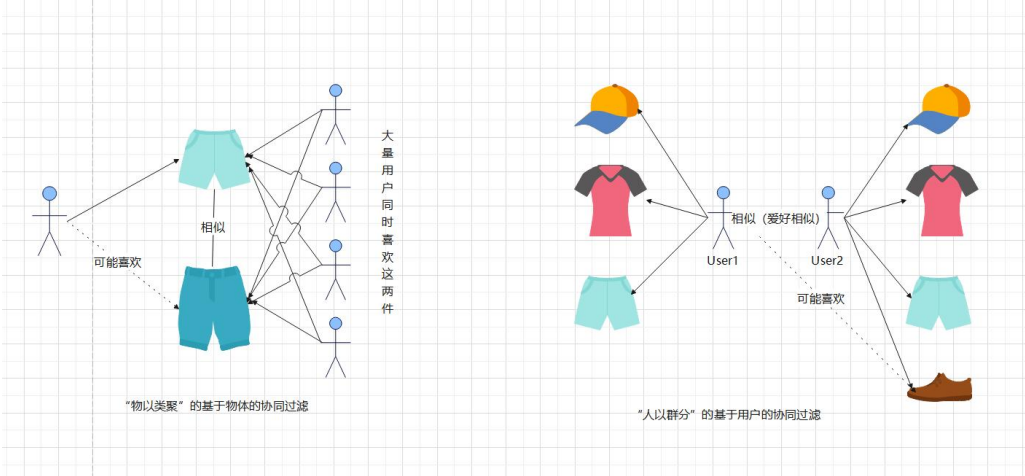

图 15. 协同过滤原理图解

协同过滤的核心就是计算相似度。对于基于人的协同过滤算法，核心是计算人与人之间的相似度；对于基于物的协同过滤算法，核心是计算物与物之间的相似度[8]。

我们可以采用非常朴素的思想来计算相似度。我们将用户对标的物的评分构建如下用户行为矩阵，矩阵的某个元素代表某个用户对某个标的物的评分，如果某个用户对某个标的物未产生行为，值为 0。其中行向量代表某个用户对所有标的物的评分向量，列向量代表所有用户对某个标的物的评分向量。有了行向量和列向量，我们就可以计算用户与用户之间、标的物与标的物之间的相似度了。具体来说，行向量之间的相似度就是用户之间的相似度，列向量之间的相似度就是标的物之间的相似度。

$$\begin{bmatrix} R11 & R12 & R13 & \dots & R1m \\ R21 & R22 & R23 & \dots & R2m \\ \dots & \dots & \dots & \dots & \dots \\ Rn1 & Rn2 & Rn3 & \dots & Rnm \end{bmatrix}$$

图 16. 用户行为矩阵

虽然基于用户的协同和基于物品的协同是两种不同的算法，但是它们的本质

是一样的，都是通过计算相似度来进行推荐的，因此，在实际生活中往往选其一即可，这里我以基于用户的协同过滤为例进行讲解[9]。

3.5.1 计算用户相似度

由上述可知，在用户的行为矩阵中，每一个元素代表了某一个用户对某一个物品的打分，因此矩阵中的每一行代表了某一个用户对所有物品的打分，通过计算任意两行向量之间的相似度值就可以得出这两行所对应的用户之间的相似度。常见的相似度计算的方法有很多，这里挑选了余弦相似度，和皮尔孙相关系数来进行实现和比较。[10]

余弦相似度

余弦相似度是一种衡量两个向量之间的相似度的方法。它用来计算两个向量在 n 维空间中的夹角余弦值，可以用来比较这两个向量的方向是否相同，即是否具有相似的指向，而与它们的大小无关（取值范围为：[0,1]，值越大说明越相关）。

余弦相似度的计算公式为：

$$cosine\_sim(A,B) = A \cdot B / \| A \| \| B \|$$

其中，A • B 表示 A 和 B 的向量内积，||A|| 和 ||B|| 分别表示 A 和 B 的向量长度（模长）[8]。

Python 代码实现：

```
# 计算用户的余弦相似度
def get_cos_similar(v1: list, v2: list):
    num = float(np.dot(v1, v2))  # 向量点乘
    denom = np.linalg.norm(v1) * np.linalg.norm(v2)  # 求模长的乘积
    return (num / denom) if denom != 0 else 0
```

皮尔孙相关系数

皮尔孙相关系数（Pearson correlation coefficient）是衡量两个变量之间线性关系强度的一种方法，常用于数据分析和数据挖掘中（取值范围为：[-1,1]，值越大说明越相关）。

假设我们有两个变 X 和 Y，它们的样本数据分别为$x_1, x_2, \ldots, x_n$和$y_1, y_2, \ldots, y_n$，则它们的样本均值分别为：

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i, \bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$$

它们的样本标准差分别为：

$$s_x = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n} (x_i - \bar{x})^2}, s_y = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n} (y_i - \bar{y})^2}$$

它们的样本协方差为：

$$s_{xy} = \frac{1}{n-1}\sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})$$

则两个变量的皮尔逊相关系数为：

$$r_{xy} = \frac{s_{xy}}{s_x s_y}$$

其中，r_{xy}的取值范围在[-1, 1]之间，当r_{xy}=1 时表示两个变量呈完全正相关，当 r_{xy}=-1 时表示两个变量呈完全负相关，当r_{xy}=0 时表示两个变量之间没有线性关系[11]。

Python 代码实现：

```
#计算用户的皮尔孙相似度
def get_pearsonr_similar(v1: list, v2: list):
    return scipy.stats.pearsonr(v1, v2)
```

示例

假设有如下用户的行为矩阵，代表了他们对 8 篇博客各自的打分（打分区间为：[1,10]）。

其中 Bob1 对 B1，B2 和 B3 的打分和 Alice 相似，其他的打分差异很大；Bob2 在 Bob1 的基础上，修改了对 B4 和 B5 的打分和 Alice 一样；Bob3 在 Bob2 的基础上，进一步修改了对 B6，B7 和 B8 的打分和 Alice 一样，因此原则上，Bob1 到 Bob3 他们与 Alice 的相似度会越来越高。

表 3.用户行为矩阵表

用户	B	B	B	B	B	B	B	B

	1	2	3	4	5	6	7	8
Alice	8	7	4	8	10	9	1	2
Bob1	8	6	2	2	1	2	10	9
Bob2	8	6	2	8	10	0	9	8
Bob3	8	6	2	8	10	9	1	2

分别计算 Alice 与 Bob1，Bob2 和 Bob3 的余弦相似度和皮尔孙相似度，得到结果如下：

表 4.用户相似度

	余弦相似度	皮尔孙相似度
Bob1	0.557	-0.685
Bob2	0.766	-0.113
Bob3	0.994	0.979

根据上面的计算结果,可以看出 Alice 与 Bob1 的余弦相似度为 0.557,Alice 与 Bob2 的余弦相似度为 0.766，Alice 与 Bob3 的余弦相似度为 0.994。由此可知，Alice 与 Bob3 的相似度最高，说明他们对博客的评分比较相似，而 Alice 与 Bob1 的相似度最低，说明他们对博客的评分比较不相似。

同时，根据皮尔孙相似度的计算结果，可以看出 Alice 与 Bob1 的皮尔孙相似度为-0.685，表示他们之间的相关性很低，Alice 与 Bob2 的皮尔孙相似度为-0.113，也表示他们之间的相关性较低，而 Alice 与 Bob3 的皮尔孙相似度为 0.979，则表示他们之间的相关性很高，更能反映他们之间博客评分的相似性。因此，可以得出结论，对于这组数据而言，余弦相似度和皮尔孙相似度都认为 Alice 与 Bob3 之间的博客评分相似度最高。

通过分析计算结果可知,余弦相似度和皮尔孙相似度的计算结果和我们假设

的条件一致。

3.5.2 预测用户对物品的评分

预测用户对物品的评分是协同过滤算法的核心问题之一。基于用户的协同过滤常用的预测方法有两种：

1. 加权平均法（Weighted Mean）：该方法计算目标用户对某一物品的加权平均分。以和目标用户有相似评分行为的用户他们对目标物品的评分为基数，以他们和目标用户之间的相似度为权重进行加权平均[12]。

2. 基于最近邻接的方法（Nearest Neighbor）：该方法选取与目标用户最相似的一组用户，计算这组用户对该物品的平均评分，将其作为目标用户对该物品的评分[13]。

上面的两种计算方法都各有优劣，基于最近邻接的方法简单，计算速度快，加权平均能够利用所有与目标用户有相似行为的信息，可以平滑地处理用户评分中的噪声和偏差，能够在一定程度上提高预测的准确性。

为了达到更好的推荐效果，我们将这两种算法结合起来，在基于最近临近算法的基础上我们再去使用加权平均，使用最近临近算法我们可以找出和目标用户最相似的一组用户，基于这组用户对目标物品的评分为基数，以他们和目标用户的相似度为权重，进行加权平均，将这两组算法结合起来，其一，通过最近临近，我们缩小了计算范围，得到了和目标用户最相似的一组用户，其二，即使是最相似的一组用户，他们之间的评分也会有各自的差异，通过加权平均，能够在一定程度上消除这种差异，进一步提高预测的准确性[14]。

如下表所示，假设通过最近临近算法计算得到 Bob1，Bob2 和 Bob3 是和 Alice 相似度最高的三个用户，采用余弦相似度方法进行计算，他们与 Alice 的相似度分别为：

S(Bob1)=0.995，S(Bob2)=0.989，S(Bob3)=0.896

通过加权平均计算 Alice 对 B5 的评分：

S=(S(Bob1)*7+S(Bob2)*9+S(Bob3)*5)/(S(Bob1)+S(Bob2)+S(Bob3))=7.065

表 5.用户打分表

用户	B1	B2	B3	B4	B5
Alice	5	2	8	4	?
Bob1	5	1	8	4	7
Bob2	4	1	8	3	9
Bob3	9	2	5	5	5

因此最终预测 Alice 对 B5 的评分为 7.065。

3.5.3 为用户推荐

在上一步中，我们预测出用户对某一个物品的评分后，就可以直接以该评分为推荐度，并按照推荐度进行降序排序，向用户推荐评分最高的前 k 个物品。

3.6 应用协同过滤算法至博客系统

在完成对基于用户的协同过滤算法的研究后，我们使用 python 语言实现并进行封装，最终将其封装成 3 个基本函数，分别是 process_user，get_xxx_similar（xxx 可以选，因为相似度的计算可以是余弦相似度，也可以是皮尔孙相似度）和 cf_user。其中 process_user 是对用户数据的一些预处理，比如：对于两个用户之间的差集，我们会在各自缺的地方进行补零操作，get_xxx_similar 则是计算用户之间相似度的函数，cf_user 则负责基于上面的两个方法计算出用户的最终评分[15]。

对于博客系统，我们将该算法应用在“博客列表”的接口中，在将数据响给前端前，我们调用协同过滤算法的接口，对整个博客列表中每篇博客进行推荐度的计算，并按照推荐度进行降序排序，在前端用户拿到数据后，整个博客列表的数据就是从推荐度高到推荐度底的展现形式。

3.7 使用 nginx 和 uwsgi 部署

使用 Django 自带的 runserver 命令启动 Django 应用，只适用于开发环境或简单的测试环境，这是因为 runserver 命令的设计目的是为了方便开发人员进行

调试，而不是为了处理高并发，负载均衡等生产环境下需要考虑的问题。

因此，对于实际的生产环境，考虑到服务的性能，安全和稳定性等问题，我们选择使用 nginx 和 uwsgi 来部署，相关部署流程如下[16]：

注意：在部署过程中，除了以下流程外，还需要注意环境和相关工具等条件已具备，这里只介绍大致流程，就不做过多的赘述。

3.7.1 编写 uwsgi 配置文件

```
[uwsgi]

#使用 nginx 连接时使用
socket = 127.0.0.1:6666

#项目目录
chdir = /home/admin/blog-plus/MyBlog

#项目中 wsgi.py 文件的目录(相对'项目目录'的路径或绝对路径)
wsgi-file = MyBlog/wsgi.py

#指定项目的 application
module = MyBlog.wsgi

#虚拟环境
virtualenv = ../venv

#python 解释器的路径
pythonpath = ../venv/bin/python3

#指定启动的工作进程数
processes = 4
master = true

# 指定工作进程中的线程数
threads = 8
enable-threads = true

# 保存启动之后主进程的 pid
pidfile = uwsgi.pid

# 日志的输出路径
daemonize = uwsgi.log
```

```
34.
35. #指定静态文件路径（使用 nginx 后就不需要了）
36. #static-map = /static=/MyBlog/static
37.
38. #py 文件修改,自动重新加载(推荐开发中使用)
39. py-autoreload = 1
40.
41. #设置 socket 的监听队列的大小(默认是 100)
42. listen = 150
```

3.7.2 编写 nginx 的配置文件

```
1.  server
2.  {
3.      listen 80;
4.      server_name www.lll.plus;
5.      charset utf-8;
6.
7.      include    mime.types;
8.      default_type application/octet-stream;
9.
10. ##################################################################################
11.    # 个人博客系统
12.     location / {
13.       include uwsgi_params;
14.       uwsgi_connect_timeout  300; #设置 uwsgi 超时时间
15.       uwsgi_pass 127.0.0.1:6666;  #端口要和 uwsgi 里配置的一样
16.     }
17.
18.     location /static/ {
19.         alias /home/admin/blog/MyBlog/static/;  #静态资源路径
20.     }
21.
22.    location /media/ {
23.        alias /home/admin/blog/MyBlog/media/;
24. }
25. }
```

3.7.3 模型迁移和收集静态文件

为了将 Django 中的模型映射到数据库的表中，需要进行模型迁移，在不使用 runserver 命令启动后，需要将静态文件收集到指定目录，并使用 nginx 进行代理，这样页面才能进行正常显示。相关命令如下：

```
#模型迁移
python3 manage.py makemigrations
python3 manage.py migrate

#收集静态文件
python3 manage.py collectstatic
```

3.7.4 启动服务

在编写好配置文件并将它们移动到相应的位置后，先使用 uwsgi 启动 Django 项目，再使用 nginx 启动代理服务，整个项目在没有意外的情况下就能正常启动了[17]。相关启动命令如下：

```
#使用 uwsgi 启动 Django
uwsgi uwsgi.ini

#启动 nginx 服务
systemctl start nginx
```

四、总结与展望

4.1 总结

从项目的设计到具体实施，再到论文的总结，每一个过程中都有不一样的体验和收获。在最初进行项目设计时，我按照标准的项目开发流程进行，从提出需求，需求分析，系统设计，架构设计等等开始，与以往不一样的是，在具体实施开发前，我做了充足的准备，虽然这也消耗了我大量的时间和精力，但是以长远的眼光来看，这却给我后来的工作铺平了道路，在提前设计好的方案引导下，后

面的开发比以往都要顺利很多，因为系统中任何一个部分出了问题我都可以参考设计的方案很快发现问题并解决，这也让我明白了提前规划的重要性，俗话说：磨刀不误砍柴工，在做好充足的准备后，后面的工作也许会更加顺利；在着手具体开发的过程中，既巩固了对 Django，numpy 等旧知识的理解，也学习到了一些新的东西，比如前端开发框架 Vue，学习完它后，让我对前端有了更深的了解，同时也提高了我的开发效率，当然，更重要的是学习和应用了协同过滤算法，在研究协同过滤的过程中参考了很多文献和文章，通过对比，分析，得到自己的理解，并最终实现和应用，当然，我很高兴最终实现和应用了它，但是更让我兴奋的是探究它的这个过程，特别是遇到问题，并将其解决的那一刹那！俗话说：授人以鱼不如授人以渔，只有当你掌握了方法，在遇到下一个“协同过滤”的“鱼”时，也能将其轻易“捕获”；最后在进行论文总结时，在这个过程中，让我有机会重新去审查我的设计方案以及项目的代码，对于有错误或设计不好的地方，我能够加以改正，同时通过反复查看，偶尔也能让我产生出更好的解决（设计）方案，特别是代码逻辑，经过不断的更新迭代，总能让我感到很满意！最后，不管是哪个阶段，收货和成长才是最重要的！

4.2 展望

虽然最终完成了整个项目的开发，也成功将协同过滤算法应用在系统中，但是其中也有很多设计不好或者有问题的地方。现阶段，系统中遗留的还有如下一些问题：

1. 改进推荐算法：目前的计算已经很耗时了，在利用缓存的条件下能有效解决这个问题，但是由于博客数量以及用户操作数量的日益增加，后面计算的耗时会越来越长，用户的体验感会随之下降。

2. 功能需要进一步完善：虽然一个博客系统基本的功能实现了，但是任有一些不完善的地方，比如用户的社交系统，用户的个人中心系统，这些目前都还不具备。

3. 结构优化：目前在 Django 中都是基于函数开发的视图函数，实现方式比较简单，后续希望能够实现基于类的视图函数来代替，提高抽象能力。

后续希望能够将这些问题一一解决或优化，同时也希望能够发现更多的问题

并将它们逐一解决，希望在对项目的每一次更新迭代中都能学习和掌握新的东西，也希望能够一直坚持对这个项目的更新迭代，让它在每一次更新迭代中茁壮成长。

参考文献

[1] 苏庆,陈佳欣,黄鸿林,黄佃宽,何楚明.基于深度知识追踪的个性化推荐编程实训系统建设与教学实践[J].实验技术与管理,2023.03.034:6-8.

[2] 赵智,姚爽,韩丹.数字科技馆展品个性化推荐系统设计[J].互联网刊,2023(07):33-35.

[3] 白源,马浚,刘松华,李泽鹏.基于用户评分一致性的协同过滤个性化推荐算法[J].广州大学学报(自然科学版),2023,22(01):9-16.

[4] 吴枰.个性化推荐系统在高职计算机通识课教学中的应用探究[J].电脑知识与技术,2022,18(36):157-159.

[5] Lin Jing,He Mingkai,Pan Weike,Ming Zhong. Collaborative filtering with sequential implicit feedback via learning users' preferences over item-sets[J]. Information Sciences,2023,621:6-9.

[6] Wang Jiahao,Mei Hongyan,Li Kai,Zhang Xing,Chen Xin. Collaborative Filtering Model of Graph Neural Network Based on Random Walk[J]. Applied Sciences,2023,13(3):12-17.

[7] 郭飞雁,罗校清.基于用户画像的在线学习资源个性化推荐服务研究[J].中小学电教,2022(12):60-63.

[8] 覃琼花.基于协同过滤算法的个性化推荐系统研究[J].科技资讯,2022,20(10):4-6.

[9] 赵家贝.基于数据挖掘的职位信息可视化分析与个性化推荐[D].华东师范大学,2022.DOI:10.27149/d.cnki.ghdsu.2022.001181:4-8.

[10] 王默涵.一种协同过滤的科学数据精准推荐系统设计[J].中国科技信息,2023(06):102-104.

[11] 李臣龙,强俊.基于用户特征分析的协同过滤算法优化[J].黑河学院学报,2023,14(02):180-182.

[12] 刘晓蒙.基于协同过滤的学习资源推荐算法[J].信息与电脑(理论版),2023,35(01):63-65.

[13] Behera Gopal,Nain Neeta. Collaborative Filtering with Temporal Features for Movie Recommendation System[J]. Procedia Computer Science,2023,218:3-9.

[14] Li Shengwen,Chen Renyao,Sun Chenpeng,Yao Hong,Cheng Xuyang,Li Zhuoru,Li Tailong,Kang Xiaojun. Region-aware neural graph collaborative filtering for personalized

recommendation[J]. International Journal of Digital Earth,2022,15(1):6-8.

[15] Jena Kalyan Kumar,Bhoi Sourav Kumar,Malik Tushar Kanta,Sahoo Kshira Sagar,Jhanjhi N Z,Bhatia Sajal,Amsaad Fathi. E-Learning Course Recommender System Using Collaborative Filtering Models[J].Electronics,2022,12(1).:8-9.

[16] Ofem Ajah Ofem,Agana Adah Moses,Felix Oromena Elemue. Collaborative Filtering Recommender System for Timely Arrival Problem in Road Transport Networks Using Viterbi and the Hidden Markov Algorithms[J]. International Journal of Software Innovation (IJSI),2022,11(1):3-8.

[17] 赵峰涛. 基于协同过滤算法的高校图书书目推荐系统设计[J]. 微型电脑应用, 2022, 38(12):67-69.

致谢

在这篇论文即将结束的时刻，我深切地感受到自己在完成这篇论文过程中得到的帮助和支持。在此，我要向所有在我的研究中给予帮助和支持的人们表达我们最诚挚的谢意。

首先，我要感谢我的导师，他为我提供了无私的指导和悉心的教导。导师不仅给予我们学术上的指导，还在生活上给予了我们大量的帮助和关怀。他教会我们如何做学问，如何做人，成为了我人生中不可或缺的一部分。

我还要感谢所有参与我们调查、实验和讨论的被试和志愿者，他们的付出是本研究取得成功的重要保障。同时，我要感谢所有给予我们支持和帮助的机构和组织，没有他们的支持，我们无法顺利完成研究。

最后，我还要感谢家人和朋友们的支持和鼓励，他们一直在我们身边，为我加油打气，鼓励我坚持下去。我感激他们在我们追求梦想的路上的陪伴和支持。

再次感谢所有给予我们帮助和支持的人们，您们的支持和鼓励是我们完成这篇论文的最大动力。