

南京信息工程大学

本科生毕业论文(设计)



题 目 个性化新闻推荐系统设计与实现

学生姓名 万宇轩

学 号 201733050050

系 别 应用技术学院

专 业 计算机科学与技术

指导教师 黄群

二〇二一年五月十五日

声 明

本人郑重声明：

- 1、持以“求实、创新”的科学精神从事科学研究工作。
- 2、本论文中除引用外，所有测试本论文中除引文外，所有测试、数据和相关材料均为真实有效的。
- 3、本论文是我个人在指导教师的指导下进行的研究工作和取得的研究成果，请勿用于非法用途。
- 4、本论文中除引文和致谢的内容外，并未抄袭其他人或其他机构已经发表或撰写过的研究成果。
- 5、关于其他同志对本研究所做的贡献均已在论文中作了声明并表示了谢意。

作者签名： 万宇轩

日期： 2021年5月15日

目 录

1. 引言	1
1.1. 研究背景和意义	1
1.2. 国内外研究现状	1
1.3. 论文主要研究内容	1
1.4. 结构安排	1
2. 相关理论介绍	2
2.1. 个性化推荐系统	2
2.2. Embedding 技术	2
2.3. LSTM 自动编码器	3
2.4. Wide&Deep 模型	3
3. 新闻推荐系统需求分析	4
3.1. 需求概述	4
3.2. 功能需求	4
3.2.1. 新闻更新和存储功能	4
3.2.2. 个性化新闻推荐功能	4
3.2.3. 用户行为收集功能	4
3.3. 非功能性需求	5
3.4. 本章小结	5
4. 系统设计	5
4.1. 系统架构设计	5
4.2. 总体技术架构路线	6
4.3. 系统功能模块设计	6
4.3.1. 爬虫模块设计	7
4.3.2. 推荐模块设计	7
4.3.3. 业务系统模块设计	8
4.4. 推荐算法设计	8
4.4.1. 新闻文本建模	9
4.4.2. 候选集生成	9
4.4.3. 候选集排序	10
4.4.4. 新闻热度值	10

4.5. 系统数据库设计	12
4.5.1. 概念模型设计	12
4.5.2. 关键数据库表结构设计	14
4.6. 本章小结	16
5. 系统详细设计与实现	16
5.1. 爬虫模块	17
5.1.1. 新闻数据爬取实现	17
5.1.2. 爬虫日志分析实现	18
5.2. 推荐模块	18
5.2.1. 新闻文本建模算法实现	18
5.2.2. 候选集生成算法实现	19
5.2.3. 候选集排序算法实现	21
5.3. 业务系统模块	22
5.3.1. 前端部分实现	22
5.3.2. 后端部分实现	26
5.4. 系统效果展示	28
5.4.1. 用户新闻浏览	28
5.4.2. 用户新闻评分	28
5.4.3. 新闻推荐效果比对	29
5.5. 本章小结	31
6. 结束语	32
参考文献	32
致谢	34

个性化新闻推荐系统设计与实现

万宇轩

南京信息工程大学应用技术学院，江苏 南京 210044

摘要：近年来，新闻推荐成为一个热门的研究领域。一个设计良好的新闻推荐系统可以为各大信息门户网站吸引大量的用户流量。在此背景下，本文设计并实现了一个 B/S 结构的新闻推荐系统。首先，系统业务层使用 Vue.js 作为前端框架来实现页面的显示；采用 Spring Boot 和 Flask 来分别作为 Java 和 Python 的后端框架，从而对数据库进行操作；采用 Tensorflow 对新闻数据进行处理，并用来构建 LSTM 自编码器和 Wide&Deep 推荐模型；采用 Scrapy 作为爬虫服务的数据抓取框架；系统数据库采用 MongoDB 和 Redis。最后，本系统将深度学习与推荐系统相结合，实现了一个深度学习新闻推荐系统。因为有了深度学习，本系统可以更好地捕捉用户的兴趣，进而为用户提供高质量的推荐服务。

关键词：推荐系统；深度学习；LSTM 自编码器；Wide&Deep 模型；

Design and Implementation of Personalized News Recommendation System

Wan Yuxuan

College of Applied Technology, NUIST, Nanjing 210044, China

Abstract: In recent years, news recommendation has become a hot research field. A well-designed news recommendation system can attract a large amount of user traffic to major information portals. In this context, this paper designs and implements a news recommendation system with B/S structure. First, the system business layer uses Vue.js as the front-end framework to display the page; uses Spring Boot and Flask as the back-end frameworks of Java and Python to operate the database; uses Tensorflow to process news data and use it Construct LSTM autoencoder and Wide&Deep recommendation model; adopt Scrapy as the data capture framework of crawler service; adopt MongoDB and Redis as the system database. Finally, this system combines deep learning and recommendation system to realize a deep learning news recommendation system. Because of deep learning, the system can better capture users' interests, and then provide users with high-quality recommendation services.

Key words: Recommendation system; deep learning; LSTM autoencoder; Wide&Deep;

1. 引言

1.1. 研究背景和意义

近年来，随着互联网的飞速发展，互联网上的数据量已经达到了天文数字，人类进入了大数据时代，获取信息的渠道和获取信息的数量都变得更加丰富，但与此同时，也逐渐迷失在海量的数据中，无法快速准确地找到自己需要的信息。

在此背景下，推荐系统逐渐进入人们的视野。推荐系统是一个集数据挖掘、特征提取和信息检索于一体的复杂工程系统。推荐系统的出现有效地解决了信息过载的问题。目前，推荐系统已成为业界和学术界的研究热点。

本课题的研究意义在于使用推荐系统来帮助用户发掘感兴趣的新闻，主要通过挖掘用户的历史记录数据、用户的个人爱好信息以及用户的高评分新闻等数据，从而分析出该用户可能感兴趣的新闻，进而推荐给用户。

1.2. 国内外研究现状

早期，推荐系统主要采用协同过滤^[1]、关键词抽取^[2]等方法，但这些方法无法发现推荐项目与用户深层特征之间的关系，限制了推荐的效果。

因为上述原因的限制，基于深度学习的推荐方法，近年来开始逐渐被人们所接纳并广泛推广开来。

2015 年，澳大利亚国立大学首次将神经网络的概念引入推荐系统领域，提出了 AutoRec^[3]模型。随后，微软在 2016 年发布了 Deep Crossing^[4]模型，该模型的发布为推荐领域的深度学习奠定了基础；紧接着，在同一年，由谷歌发布的 Wide&Deep^[5]模型完全引爆了深度学习在推荐系统上的应用。2017 年，华为和哈尔滨工业大学对谷歌的 Wide&Deep 模型进行了改进，得到了 DeepFM^[6]模型，进一步提高了特征融合的能力。

1.3. 论文主要研究内容

传统的新闻特征提取主要是用词频统计的方式，即 TF-IDF^[7]方法，该方法有几个明显的缺点，一是这种词频统计方式会导致最后的统计信息缺乏词的位置信息，二是无法体现词语之间的上下文结构等。基于以上原因，首先，本文通过引入深度学习中的 Embedding^[8]技术，来实现新闻标题词的向量化，接着通过引入自编码器，来构建新闻标题的文本模型，然后，使用多路召回策略来生成候选集，最后，使用 Wide&Deep 模型计算候选集的用户点击率，并对其进行排序，得到用户的新闻推荐列表。

1.4. 结构安排

本文主要由六章组成，每一章的主要内容如下：

1. 引言。首先介绍了本课题的研究背景和意义，然后介绍了国内外新闻推荐系统的相关研究现状，最后说明了本文的主要内容和结构。

2. 相关理论介绍。主要介绍本文的理论基础，如推荐系统、嵌入技术、自编码技术等。

3. 新闻推荐系统的需求分析。本文主要描述了系统功能需求和非功能需求的定义。

4. 系统设计。主要描述了系统的总体结构设计，详细描述了系统的总体技术路线，并对

系统的核心部分——推荐算法进行了详细描述。

5. 系统的详细设计与实现。主要对系统的实现进行了详细的描述和介绍，同时展示了最终的实现效果。

6. 结束语。主要对全文进行了总结，并对今后可能的工作方向作了简要介绍。

2. 相关理论介绍

本章节主要对系统的理论知识部分做具体的介绍说明，主要包括个性化推荐系统的概述、Embedding 技术、LSTM^[9]自编码器以及 Wide&Deep 模型的介绍。

2.1. 个性化推荐系统

推荐系统^[10]深受各大互联网公司的认可，因为能够快速而精准的为用户提供其感兴趣的信息，并且推荐系统能够很好的处理“人”与“信息”之间的关系。

一般来说，推荐系统主要分为两部分，具体的划分如下图 2-1 所示。

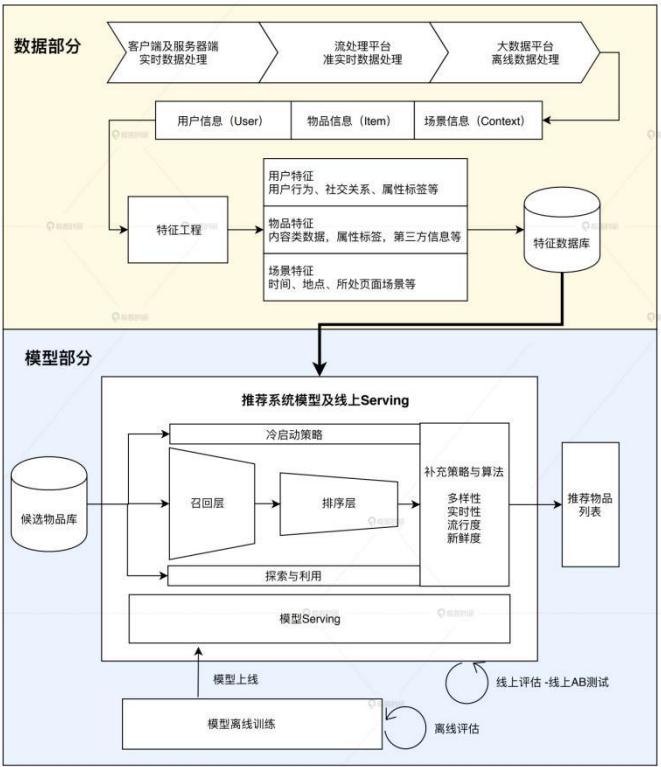


图 2-1 推荐系统技术架构图

如上图所示，数据部分负责诸如用户、物品以及场景等信息数据的采集和处理。模型部分也就是推荐模型，这一部分是推荐系统的核心，主要由“召回层”和“排序层”组成。召回层主要使用一些有效的召回策略或规则，从而能够快速而有效地从海量数据集中找到用户可能感兴趣的数据。排序层使用算法模型对上一步生成的候选集进行进一步排序，同时，结合一些辅助策略，对生成的推荐列表进行调整，最终生成用户的推荐列表。

2.2. Embedding 技术

Embedding 技术，它的中文翻译是“嵌入技术”，但一般常用“向量”、“向量化”或者“向量映射”来代替。

嵌入技术是深度学习领域的“基本核心操作”。嵌入技术主要是用一个低维的稠密向量来表示对象，对象可以是任何实体，也可以是非实体，例如单词、新闻文本、苹果等。由此可以看出，Embedding 向量能够表达相应物品的特征信息，而且，不同 Embedding 之间可以进行向量运算操作，从而得到不同物品之间的距离，也就是相似度。

2.3. LSTM 自动编码器

自动编码器是一种神经网络模型。首先，编码器压缩输入数据以减小数据维度大小，然后，将压缩后的数据还原恢复为原始的输入数据，具体的模型结构如图 2-2 所示。

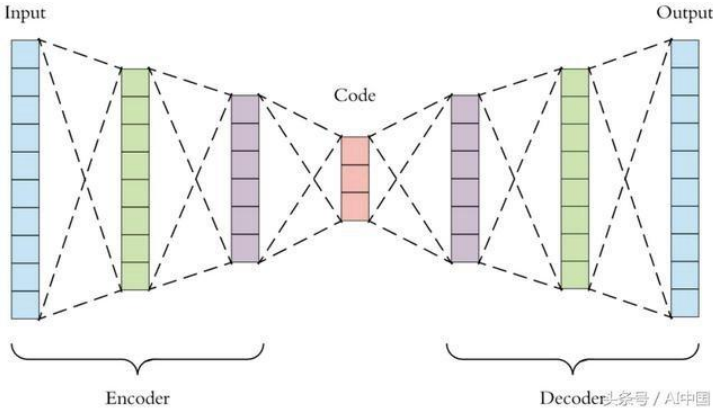


图 2-2 自动编码器模型结构图

LSTM 自编码器是自动编码器的一种特例，因为 LSTM 神经网络能够用于处理具有时序信息的序列化数据，例如文本数据、语音数据等，所以 LSTM 自编码器也就可以用来处理此类的数据了。

在本文中，主要使用 LSTM 自编码器来实现新闻标题到新闻 Embedding 的转换。

2.4. Wide&Deep 模型

该模型是 Google 在 2016 年提出的模型结构，是由单层 Wide 和多层 Deep 组成的混合模型。如图 2-3 所示。

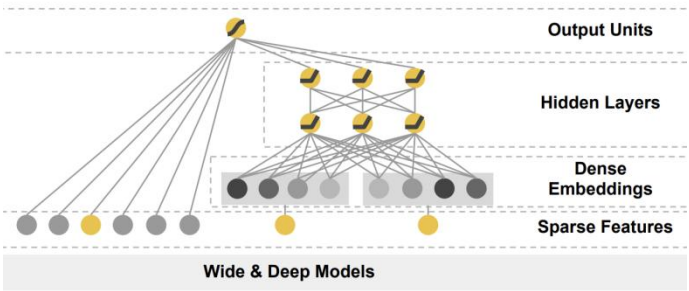


图 2-3 Wide&Deep 模型结构图

如图所示，模型左侧的 Wide 部分是一个较为稀疏的单层神经网络，作用是让模型拥有一定的记忆能力，而模型右侧的 Deep 部分是一个稠密的多层神经网络，主要作用是让模型拥有泛化能力。

在本文中，通过将用户的标签数据作为模型 Wide 部分的输入，将用户的历史新闻以及高评分新闻作为模型 Deep 部分的输入，从而得到用户的特征，然后再与候选新闻进行点积

操作，从而得到新闻的预估点击率，最后对新闻的预估点击率进行排序，得到用户推荐列表。

3. 新闻推荐系统需求分析

本章主要对个性化新闻推荐系统进行功能需求分析和非功能需求定义。

3.1. 需求概述

现今国内外的一些新闻网站，诸如网易新闻，微软新闻等，这些网站的业务流程都较为简单，主要表现为：按照用户的历史行为，来给用户推荐其感兴趣的新闻，从而提高网站流量，进而通过用户流量来变现，变现的手段主要是内部相关的广告服务。

因此，基于以上分析，整个系统分为四个部分。首先，最重要的是用户的个性化新闻推荐部分，该部分主要涉及到一些推荐算法的设计和实现；其次，就是新闻的获取以及存储；接着就是用户的行为收集，这是为新闻的个性化推荐提供可靠地数据支持；最后是对该系统的一些功能需求，包括注册登录，新闻浏览，个人标签管理，新闻评分等。

3.2. 功能需求

本小节会对上文所提到的一些基本的功能性需求概述进行进一步的详细说明。

3.2.1. 新闻更新和存储功能

(1) 新闻更新。市场上主流新闻推荐系统上的新闻大多是由企业新闻工作者编辑转载的，这样，不管是新闻的质量还是新闻来源的稳定性，都有一定的保证，但本系统由于自身条件限制，只能依靠爬虫技术来解决这一问题，通过爬取新浪新闻网站的新闻，来保证系统自身新闻的持续更新。

(2) 新闻数据的预处理和存储。从新闻网站抓取的新闻不能直接用于推荐，因为这些数据会存在一些错误，诸如创建时间，编撰人，所处分类等，这些都有可能因为一些特殊原因，而没有成功抓取到，或者抓取的数据与本系统所需的数据有出入，需要做一些预处理之后才能被推荐系统使用，因此，爬虫爬取的新闻数据应与推荐系统用的新闻数据分开存放，然后由中间组件对爬虫数据进行预处理后，再转储到推荐系统的数据库中。

3.2.2. 个性化新闻推荐功能

新闻推荐功能是系统的核心功能，主要包括以下要求：

(1) 系统需要及时响应用户的请求，为不同的用户返回不同的新闻推荐结果，每次返回的新闻应该是不同的。推荐的新闻需要符合用户的兴趣，同时也必须是用户没有浏览过的新闻。

(2) 系统应可以根据用户当前所浏览的新闻，来推荐与当前新闻类似的其它新闻。

(3) 因为新闻的时效性等缘故，系统所推送的新闻应该是近期所发生的，不可以是几周前乃至几个月前的新闻。

3.2.3. 用户行为收集功能

用户行为收集功能是用于收集用户在本系统中的一些操作，包括用户是否点击了新闻，用户对新闻的评价等等，这主要是为了丰富用户画像，方便更深层次的挖掘用户的阅读兴趣。

3.3. 非功能性需求

个性化新闻推荐系统是一个复杂的工程项目，需要多个子项目协同运行，这样才能使得推荐结果有一个较好的预期，因此为了保证项目的良好运行，在非功能性方面有以下几个需求：

(1) 运行效率。新闻推荐系统需要在几百毫秒内检索海量的数据，并从这些数据中准确找到符合用户兴趣的新闻。

(2) 可扩展性。系统在开发的过程中，需要保证系统整体的可扩展能力，即必须使用模块化设计的思路，尽量减少在添加新模块或修改已有模块后对其他模块的影响。

(3) 可维护性。系统在运行过程中，必定会产生一些错误，为此需要提供相关的日志文件供维护人员查看并定位错误的来源。

3.4. 本章小结

本章对个性化新闻推荐系统进行了详细的需求分析。首先总结了新闻推荐系统的一般需求，然后分析了系统的功能需求和非功能需求。功能需求分析主要介绍新闻爬取、预处理和存储、个性化新闻推荐、用户行为收集以及用户的一些基本功能的需求；非功能性需求主要定义系统的执行效率、可扩展性和可维护性。

4. 系统设计

4.1. 系统架构设计

系统总体架构设计如图 4-1 所示。整个系统自下而上分为数据层、服务层和显示层。

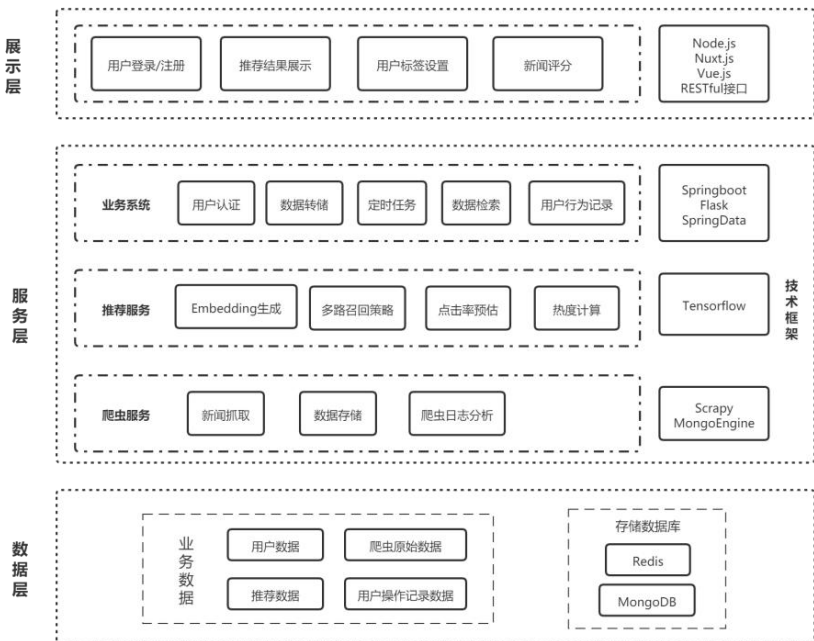


图 4-1 个性化新闻推荐系统架构图

数据层是新闻推荐系统最基础的部分，用于存放系统的各类数据，包括但不限于爬虫爬取的原始新闻数据，用户数据，推荐数据等。其中大部分数据存放在 MongoDB 中，少部分数

据存储在 Redis 中。

服务层是新闻推荐系统的核心层，主要包含三大服务，爬虫服务，推荐服务以及业务系统服务。

1. 爬虫服务主要负责原始新闻的爬取，以及爬取后新闻数据的存储。
2. 推荐服务负责新闻 Embedding 的生成，使用多路召回策略来召回用户新闻集，使用 Wide&Deep 模型来计算用户的新闻点击率，根据点击率排序，从而生成用户推荐列表。
3. 业务系统服务用于对展示层的一些请求进行响应，主要负责用户的登录认证，对爬虫的调度，数据的检索以及调用推荐服务的 API 接口来提供推荐服务。

最后，展示层是用户与系统交互的地方，用户可以注册并登录系统，浏览推送的新闻等。

4.2. 总体技术架构路线

根据上一节所介绍的系统总体架构设计，本节具体阐述一下系统的总体技术架构路线，具体的技术架构路线如下图 4-2 所示。

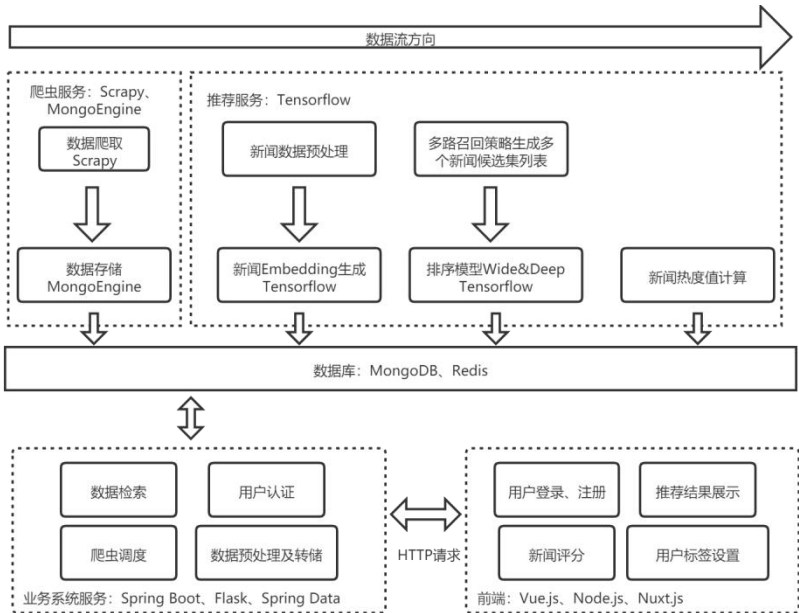


图 4-2 个性化新闻推荐系统架构技术路线图

首先，该系统采用 Scrapy 框架来抓取新闻数据，并使用 MongoEngine 将数据存储在数据库中，然后采用 Tensorflow 对数据进行处理，包括新闻嵌入的生成，采用多通道召回策略生成新闻候选集，使用 Wide&Deep 模型获得预估点击率等，从而得到新闻的推荐列表，并将其存储到 MongoDB 与 Redis 中。最后，由于本新闻推荐系统是一个前后端分离的项目，所以前端使用 Vue.js 等技术来实现数据的加载和显示，后端使用 Spring Boot 和 Flask 来对外提供 API 接口的调用，使用 Spring Data 来实现数据库的相关操作，前后端使用 HTTP 进行通信。

4.3. 系统功能模块设计

根据上述三个不同层次的系统架构设计和需求分析中“高内聚、低耦合”的原则，对新闻推荐系统的功能模块进行了划分，如图 4-2 所示。

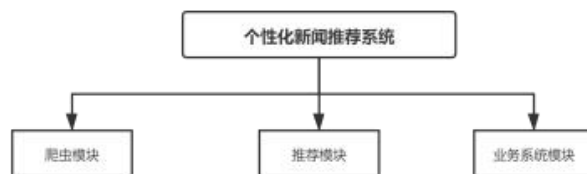


图 4-3 系统功能模块划分

4.3.1. 爬虫模块设计

爬虫模块是推荐系统的基础，是所推荐新闻的主要数据来源，主要功能如图 4-3 所示。

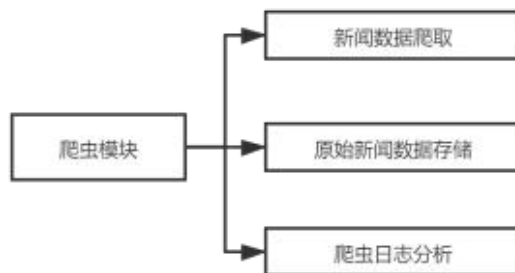


图 4-4 爬虫模块功能

爬虫模块包含三个子功能，分别为新闻数据抓取，原始新闻数据存储以及爬虫日志分析。当业务系统调度爬虫模块的爬虫时，爬虫模块都会自动产生一个子线程用于该爬虫实时的日志分析，每次分析的结果会保存在 Redis 数据库中，用于业务系统来判断爬虫的运行状态。爬虫每爬取一个新闻后，都能及时将其保存在 MongoDB 数据库中，以供推荐模块的使用。

4.3.2. 推荐模块设计

推荐模块是系统的核心，主要通过一些算法模型来生成用户可能感兴趣的新闻推荐列表，其功能如图 4-5。

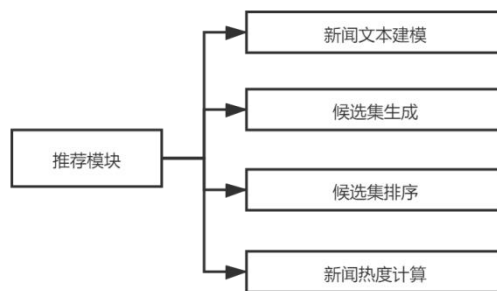


图 4-5 推荐模块功能

具体的推荐算法会在后续有所说明，此处对各子功能的作用做一个简要的介绍。首先，新闻文本建模负责将新闻的标题转化为一个低维的向量，该向量包含一个新闻所有的隐含特征；候选集生成子模块是推荐模块的主要部分，该部分可以通过多路召回策略来从海量的新闻数据中，筛选出合适的新闻候选集；候选集排序子模块用于对候选集生成子模块生成的新闻候选集进行排序，通过排序模型来从候选集中挑选出用户最感兴趣的几条数据，并生成用户的新闻推荐列表；最后是新闻热度计算子模块，该模块负责实时计算数据库中新闻当前的

热度值为多少。

4.3.3. 业务系统模块设计

业务系统模块的功能较多，如下图 4-6 所示，主要分为两部分，一个是 Web 展示子模块，另一个是业务逻辑子模块。

Web 展示子模块也就是前端，主要包括用户的登录、注册，新闻的浏览，新闻推荐列表的展示以及设置用户的兴趣标签等；业务逻辑子模块也就是后端，负责数据库的读写等操作，包括新闻数据的获取，用户信息的认证，用户行为数据的记录以及一些定时任务，例如原始新闻数据的预处理，爬虫的定时运行，定时调用推荐模块的相关服务来生成用户的推荐列表和计算新闻的热度值等。

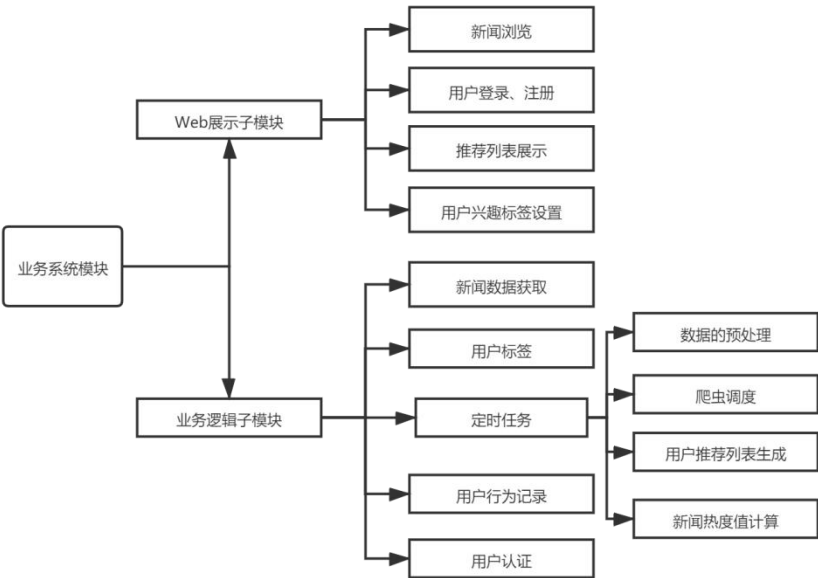


图 4-6 业务模块功能

4.4. 推荐算法设计

本新闻推荐系统的算法流程主要分为三部分，新闻文本建模、新闻候选集生成以及候选集排序，如下图 4-7 所示。

新闻文本建模包含新闻标题分词、新闻词向量序列生成以及新闻 Embedding 生成这三部分，这样最后可以为每一个新闻标题生成一个唯一的低维向量表征。

新闻候选集主要使用多路召回的策略来生成，主要的召回策略包括使用用户的兴趣标签、用户的新闻历史记录、用户的高评分新闻以及当前的热点新闻等，选取每一路召回的前 Top K 个记录，去重后来生成总的新闻候选集。

在生成总的新闻候选集后，通过使用 Wide&Deep 模型来融合用户的兴趣标签向量、用户的新闻历史记录序列以及用户的高评分新闻序列，从而对候选集中的新闻进行排序，最后生成用户的新闻推荐列表。

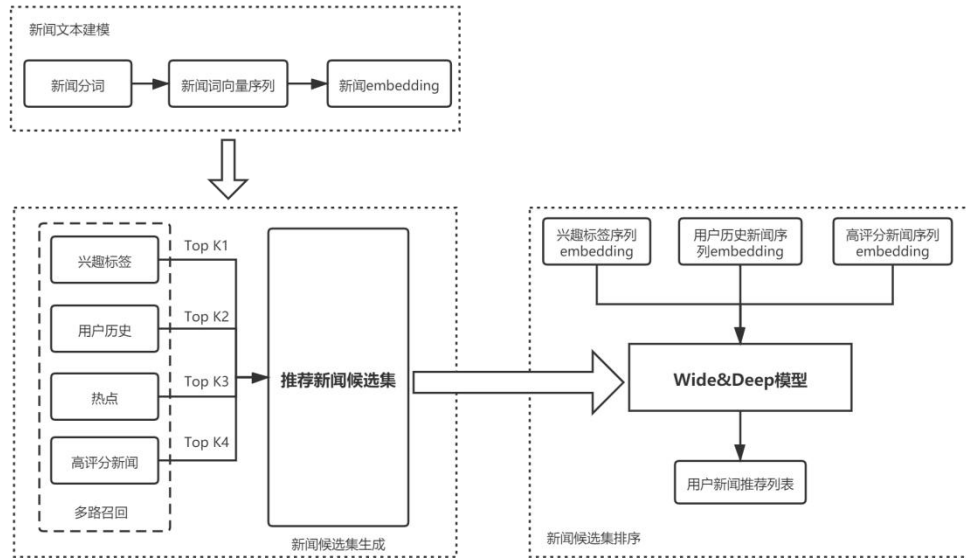


图 4-7 推荐算法设计

4.4.1. 新闻文本建模

新闻建模主要有分词处理、停用词过滤、标题词向量化以及新闻 Embedding 生成这四个流程，具体流程见图 4-8 所示。



图 4-8 新闻文本建模

首先对新闻标题进行分词处理，得到词序列 $S_1=[word_1, word_2 \dots word_n]$ ，其中 $word_i(1 \leq i \leq n)$ 为新闻标题第 i 个词，接着进行停用词过滤，得到词序列 S_2 ，然后将词序列 S_2 进行向量化，得到词向量化后的词序列 S_v ，接着将词向量化后的词序列 S_v 输入到 LSTM 自编码器中的 Encoder 部分，从而得到低维稠密的新闻 Embedding。

4.4.2. 候选集生成

候选集生成也称为召回阶段，可以说召回的新闻质量决定着最终的推荐结果。本文为了能够有效解决推荐系统的冷启动问题，依靠着多种不同的召回策略，从新闻池中选择用户可能感兴趣的新闻候选集。

本系统的召回都是基于 Embedding 来进行的，主要的召回策略有用户兴趣召回，用户新闻浏览记录召回，用户高分新闻召回以及热点新闻召回。

用户兴趣召回主要将用户的每一个兴趣标签转为 Embedding 向量，然后将所有的 Embedding 求和取平均，从而得到一个表征用户兴趣的 Embedding，接着使用该 Embedding 与近期的新闻 Embedding 求余弦相似度（余弦相似度计算见图 4-9），取 Top K 个作为用户感兴趣的新闻候选集。

用户新闻浏览召回以及用户高分新闻召回与用户兴趣召回类似，都是将用户最近浏览

的 20 篇新闻的 Embedding 或者高评分新闻的 Embedding 求和取平均值，然后计算余弦相似度，取前 Top K 个作为该召回的候选集。

$$\cos(\theta) = \frac{\sum_{i=1}^n (x_i \times r_i)}{\sqrt{\sum_{i=1}^n (x_i)^2} \times \sqrt{\sum_{i=1}^n (r_i)^2}}$$

图 4-9 余弦相似度计算公式

最后是热点新闻召回，该召回策略主要是为了解决冷启动问题，防止新注册或者未登录的用户因为没有个人数据而无法进行推荐操作。该召回策略就是通过计算新闻的热度值，取前 Top K 个作为该策略的候选集，具体热度值的计算可见后文的相关介绍。

4.4.3. 候选集排序

候选集排序是新闻推荐列表生成前的最后一步，在经过多路召回生成总的新闻候选集后，将在本阶段计算得出最后的新闻推荐列表顺序。

在本文中，主要是用 Wide&Deep 模型来计算用户对新闻的点击率预估，然后对其进行排序，具体的算法设计如图 4-10。

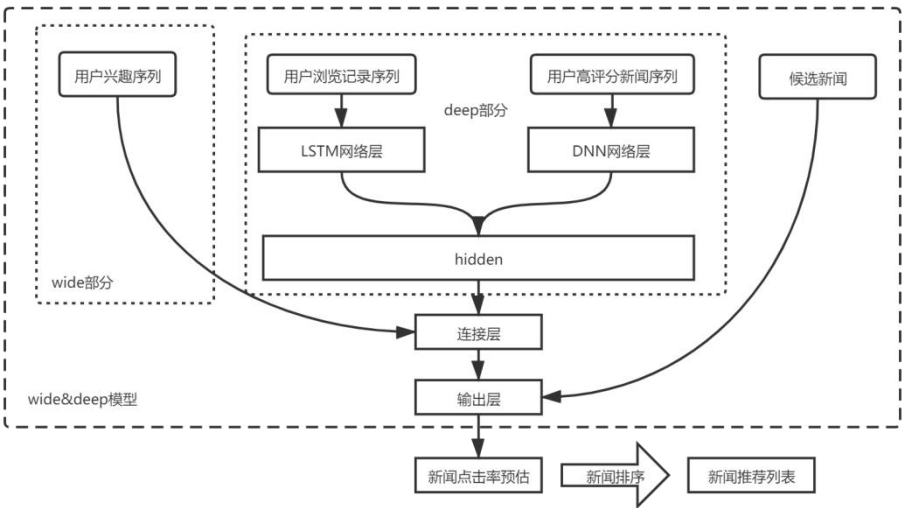


图 4-10 候选集排序

如图所示，本系统将用户的兴趣作为模型 Wide 部分的输入，将用户的新闻浏览序列以及用户高评分新闻序列作为模型 Deep 部分的输入，其中用户浏览记录序列输入到 LSTM 网络中，用来捕获用户对新闻的兴趣变迁，将用户的高评分新闻序列输入到 DNN 网络中，从而捕获用户喜爱的新闻特征，最后将 Wide 与 Deep 进行拼接，得到用户的总体特征，最后与候选新闻的 Embedding 进行内积运算，再经过一个 sigmoid 层，从而得到输出，即该候选新闻点击率预估值。在得到所有候选新闻的点击率预估值后，对其进行排序，则生成了用户新闻的推荐列表。

4.4.4. 新闻热度值

众所周知，新闻具有较强的时效性，因此，一般而言，新闻的热度应该遵循以下几点：

1. 已发布的新闻应该有一个初始的热度值，且热度值可以通过用户与之进行的互动而增加，包括浏览新闻，给新闻评分等。
 2. 新闻的热度值应随着时间的流逝而衰减，且衰减的速度应逐渐增加
 3. 如果一条新闻的热度值随着时间的推移，持续处于一个较高的水平，那么，必须要有越来越多的用户与之进行互动，保证热度值提升的速度与衰减的速度相当。
 4. 无论多么热门的新闻，必须要保证其热度值在 24 小时内能够衰减到一个较低的水平。
- 根据以上的几点，可以得出一个大致的新闻热度值计算函数，如下图所示。

$$Score = \frac{S_0(News) + S_1(User)}{T(Time)}$$

图 4-11 新闻热度值函数

图中的 Score 即为最终的热度值，其中， S_0 代表新闻的初始热度值， S_1 代表用户与新闻互动的得分， T 是一个关于新闻已发布时长的指数函数。

在系统的具体实现中，本文使用所有新闻的总数除以该新闻类别的新闻总数，取对数后再乘 100，来代表新闻的初始值；使用该新闻的浏览次数与用户对新闻的总评分，来作为用户与新闻的互动得分；最后，使用一个以 e 为底，以发布时长为指数的幂指函数作为函数 T 。具体的新闻热度公式如图所示。

$$\frac{100 * \ln(N_1/N_2) + C * 2 + S}{e^{(0.2 * H)}}$$

图 4-12 新闻热度值计算公式

在上述公式中，假设 X 为当前待计算热度值的新闻，其中 N_1 代表近 5 天所有的新闻个数， N_2 代表近 5 天中，新闻 X 所处频道的新闻个数，因此， $100 * \ln(N_1/N_2)$ 就代表了新闻的初始热度值； C 代表新闻 X 被浏览的次数， S 代表所有用户对新闻 X 评分的总和，则 $C * 2 + S$ 就是用户与新闻互动的得分； H 代表新闻 X 已发布的小时数，则 $e^{(0.2 * H)}$ 就是一个关于已发布时间的幂指函数 T 。

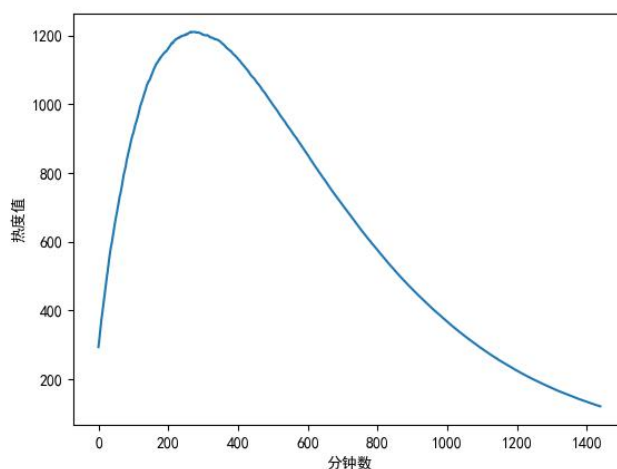


图 4-13 新闻热度值随时间变化图

上图 4-13 是一个使用该热度计算公式，通过计算而得到的一个热度衰减案例，该案例中各参数的初始值见表 4-1。

表 4-1 热度计算公式参数初始值

参数名	初始数值	意义
N_1	20000	总的新闻数
N_2	1200	新闻 X 所属频道的新闻数
C	0	新闻被浏览的次数
S	0	用户对该新闻的总评分

其中参数 C 与参数 S 是一个随时间变化的值，它的数值每分钟随机增加 Q，其中 Q 是一个随机变量，服从数学期望为 5，标准差为 1 的正态分布。

由图 4-13 可以看出，随着时间的增长，基本会在 300 到 400 分钟，也就是距离新闻发布 5~6 小时这个时间段，新闻热度达到顶峰，之后热度便会持续衰减，且随着时间的延长，衰减速度也会增加，最终在距离新闻发布 24 小时的时候，热度值几乎衰减至 0。很显然，该新闻热度值的计算公式是符合上述 4 条对新闻热度值的定义，因此可以很好的衡量新闻的热度值。

4.5. 系统数据库设计

本节主要介绍数据库方面的相关设计，包含概念模型设计以及表结构的设计等。

4.5.1. 概念模型设计

系统的主要实体包括用户、新闻、新闻频道等实体。总的 ER 图如 4-14 所示。

- (1) 用户可以点击多条新闻，每条新闻也可以被多个用户点击；
- (2) 一个用户可以生成多条浏览记录，而一条浏览记录只能由一个用户生成；
- (3) 一个用户可以有多个用户标签，一个用户标签也可以由多个用户拥有；
- (4) 一个用户可以为多条新闻打分；同时，一条新闻也可以由多个用户打分；
- (5) 一个用户只能有一个新闻推荐列表，一个新闻推荐列表只能由一个用户拥有；
- (6) 一条新闻属于一个新闻频道，一个新闻频道可以有多条新闻；
- (7) 一条新闻只能有一条嵌入，一条新闻嵌入只能属于一条新闻；

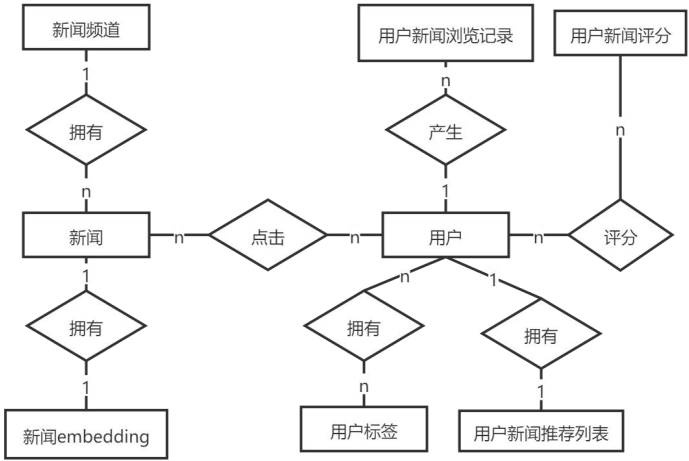


图 4-14 总 ER 图

用户实体用于表示用户数据，其属性如图 4-15 所示。

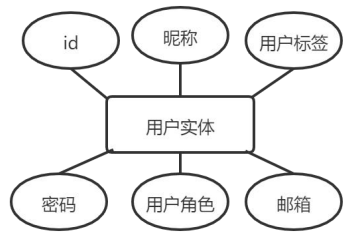


图 4-15 用户实体属性

新闻实体用于表示新闻数据，其属性如图 4-16 所示。

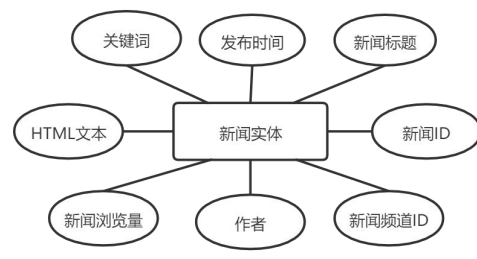


图 4-16 新闻实体属性

用户标签实体用户用于表示用户标签，其属性如图 4-17 所示。

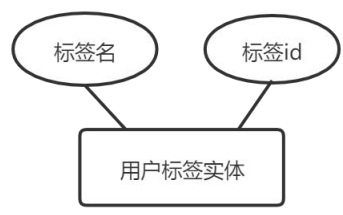


图 4-17 用户标签实体属性

用户新闻推荐列表实体用于表示用户的推荐列表，其属性如图 4-18 所示。

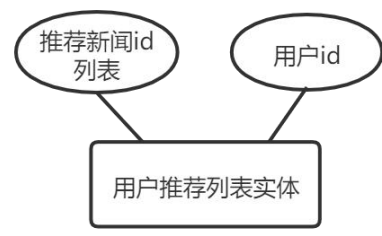


图 4-18 用户推荐列表实体属性

用户浏览记录实体用于表示用户浏览记录，其属性如图 4-19 所示。

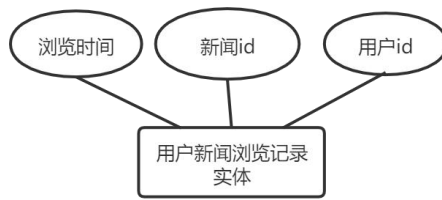


图 4-19 用户新闻浏览记录实体属性

用户评分记录实体用于表示用户评分记录，其属性如图 4-20 所示。



图 4-20 用户新闻评分实体属性

新闻频道实体用于表示频道信息，其属性如图 4-21 所示。



图 4-21 新闻频道实体属性

新闻 Embedding 实体，其属性如图 4-22 所示。

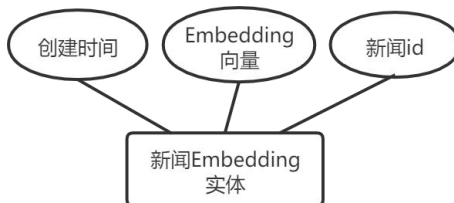


图 4-22 新闻 Embedding 实体属性

4.5.2. 关键数据库表结构设计

根据上述的实体模型与实体关系模型可知，本系统一共有 8 张数据表。

1. 用户信息表

该表用于记录用户的个人信息，包括昵称、密码、邮箱等数据。

表 4-2 用户信息表(user)

名称	字段名	数据类型	备注
用户 id	user_id	varchar (128)	主键
密码	password	varchar (20)	
昵称	nickname	varchar (10)	
邮箱	email	varchar (50)	
用户角色	role	varchar (10)	
用户标签 id 列表	labels	varchar (255)	外键

2. 新闻信息表

此表用于存储新闻数据，主要包括新闻标题、作者、内容和关键字。

表 4-3 新闻信息表(news)

名称	字段名	数据类型	备注
新闻 id	id	varchar (128)	主键
作者名	author	varchar (50)	
标题	title	varchar (20)	
创建时间	create_date	bigint	
频道 id	channel_id	varchar (128)	外键
关键词	keywords	varchar (255)	
浏览量	view_count	int	
html 文本	html_content	varchar (10000)	

3. 用户标签表

此表用于存储兴趣标签数据。

表 4-4 用户标签表(labels)

名称	字段名	数据类型	备注
标签 id	id	varchar (128)	主键
标签名	label	varchar (10)	

4. 用户新闻推荐列表

此表用于存储每个用户的新闻推荐列表数据

表 4-5 用户新闻推荐列表(user_rescommend)

名称	字段名	数据类型	备注
用户 id	uid	varchar (128)	主键、外键
新闻 id 列表	recommender_list	varchar (255)	

5. 用户新闻浏览记录表

此表记录了用户浏览新闻的行为数据，包括浏览新闻和浏览时间。

表 4-6 用户新闻浏览记录表(user_history)

名称	字段名	数据类型	备注
记录 id	id	varchar (128)	主键
用户 id	uid	varchar (128)	外键
新闻 id	news_id	varchar (128)	外键
浏览时间	browser_time	int	

6. 用户新闻评分表

此表用于保存用户的新闻打分数据。

表 4-7 用户新闻评分表(user_news_score)

名称	字段名	数据类型	备注
评分 id	id	varchar (128)	主键
用户 id	uid	varchar (128)	外键
新闻 id	news_id	varchar (128)	外键
评分	score	double	

7. 新闻频道表

此表包含所有新闻频道数据。

表 4-8 新闻频道表(news_channel)

名称	字段名	数据类型	备注
频道 id	id	varchar (128)	主键
频道名	channel_name	varchar (50)	
频道新闻个数	count	int	

8. 新闻 Embedding 表

此表用于存储矢量化后的新闻数据。

表 4-9 新闻 Embedding 表(news_embedding)

名称	字段名	数据类型	注释
新闻 id	news_id	varchar (128)	主键
Embedding 数据	embedding	text	

4.6. 本章小结

本章主要介绍了系统整体架构的相关设计，划分了系统的各个模块，详细描述了各个模块的具体功能设计，并阐述了系统的核心，即推荐算法的设计。最后，详细介绍了系统的数据库设计。

5. 系统详细设计与实现

本章主要介绍了新闻推荐系统的具体实现，包括爬虫、推荐和业务系统模块。

5.1. 爬虫模块

爬虫模块主要由新闻数据爬虫、新闻数据存储和爬虫日志分析三个功能组成。因为存储新闻数据的功能相对简单，所以这里省略。

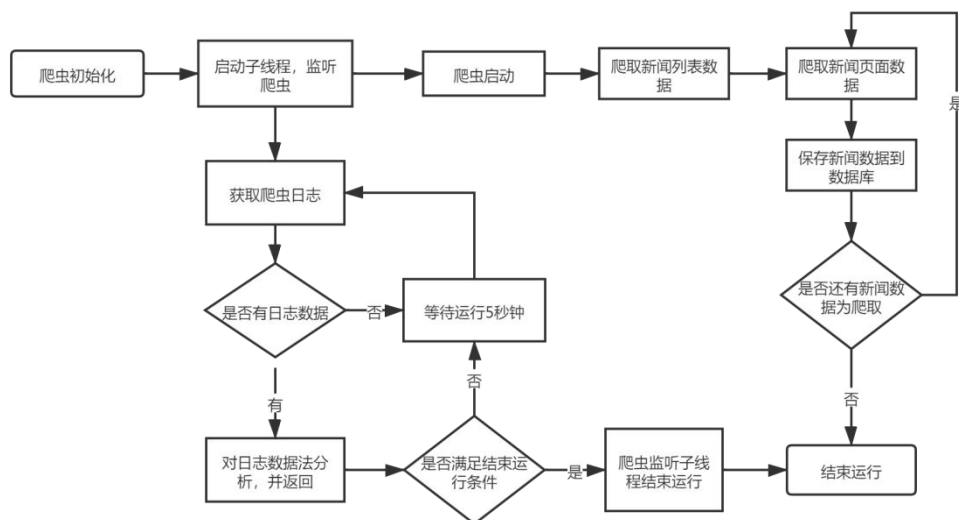


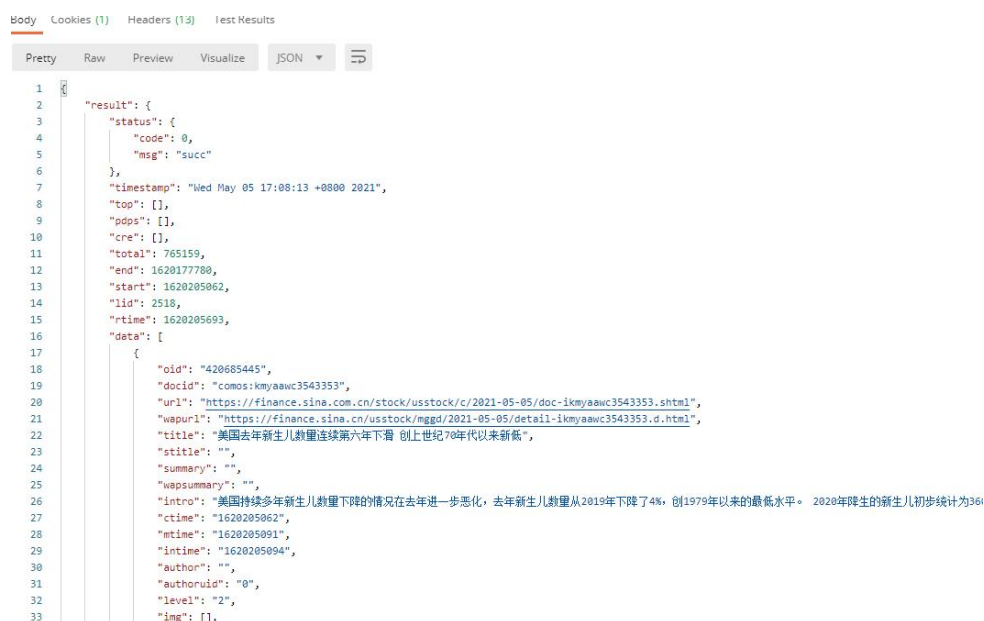
图 5-1 爬虫模块总体运行流程

如上图所示是爬虫模块具体的流程。爬虫模块首先会进行爬虫运行的初始化操作，主要有日志文件数据的初始化，Redis 数据库中，爬虫运行状态数据的初始化等。接着会启动子线程用于监听爬虫运行状态，同时也会启动爬虫。接着爬虫与爬虫监听程序会同步运行，爬虫在抓取新闻数据的时候，会把关键信息输出到日志文件中，爬虫监听程序不断读取日志文件中的数据，从而得到爬虫运行状态信息，并保存到 Redis 数据库中。最后，当满足运行结束条件后，爬虫以及爬虫监听程序则会结束运行。

5.1.1. 新闻数据爬取实现

本系统主要爬取新浪新闻上的数据，在经过抓包分析之后发现，新浪新闻的新闻 API 为“<https://feed.mix.sina.com.cn/api/roll/get?pageid=153>”，这个请求有四个参数，其中有用的参数为 num、lid 以及 page，其中 num 就是一次请求返回的新闻个数，lid 就是新闻的频道 id，page 就是请求的页码，请求结果如图 5-2 所示。

在获得请求后，开始对请求的结果数据进行解析。数据的解析主要从返回的结果中得到新闻的列表信息，这个列表信息包含每一篇新闻的 id，标题，新闻 HTML 页面的 URL 地址以及新闻的创建时间等基础数据信息。



```
1  {
2    "result": {
3      "status": {
4        "code": 0,
5        "msg": "succ"
6      },
7      "timestamp": "Wed May 05 17:08:13 +0800 2021",
8      "top": [],
9      "pds": [],
10     "cre": [],
11     "total": 765159,
12     "end": 1620177780,
13     "start": 1620205062,
14     "lid": 2518,
15     "rtime": 1620205693,
16     "data": [
17       {
18         "oid": "420685445",
19         "docid": "comos:kmyaawc3543353",
20         "url": "https://finance.sina.com.cn/stock/usstock/c/2021-05-05/doc-ikmyaawc3543353.shtml",
21         "wapurl": "https://finance.sina.cn/stock/mgqd/2021-05-05/detail-ikmyaawc3543353.d.html",
22         "title": "美国去年新生儿数量连续第六年下降 创上世纪70年代以来新低",
23         "stitle": "",
24         "summary": "",
25         "wapsummary": "",
26         "intro": "美国持续多年新生儿数量下降的情况在去年进一步恶化，去年新生儿数量从2019年下降了4%，创1979年以来的最低水平。 2020年诞生的新生儿初步统计为361",
27         "ctime": "1620205062",
28         "mtime": "1620205091",
29         "intime": "1620205094",
30         "author": "",
31         "authoruid": "0",
32         "level": "2",
33         "img": [],
```

图 5-2 新闻 API 请求返回结果

在获得每一篇新闻的基本数据后，发送请求给新闻 HTML 页面的 URL 地址，从而获得新闻 HTML 网页数据，此处主要使用 XPath 来对 HTML 页面数据进行解析，进而得到新闻的文本内容、关键词、作者、所属频道、浏览量等信息。

5.1.2. 爬虫日志分析实现

在本系统中，启动爬虫的时候，会自动启动一个子线程用于监听爬虫的运行状态信息。该子线程通过分析爬虫日志数据，从而得到爬虫当前的运行状况。其中，运行状态数据主要有：爬虫启动时间、爬虫终止时间、已爬取的新闻个数、重复爬取的新闻个数、保存到数据库的新闻个数等信息，包含的流程如上文图中图 5-1 所示。

在图中，我们可以看到爬虫监听程序内部是一个循环结构，每隔 5 秒就会运行一次。在运行的过程中，首先会从日志文件中读取数据，然后对日志数据进行分析，得到当前的运行状态信息。该爬虫监听程序结束的终止条件有两个，一个是爬虫爬取的数据个数超过了当前需要的个数，此时就会发送命令，强制关闭爬虫；另一个就是爬虫正常爬取完所有数据后，就会自动停止当前程序。其中，爬虫的运行数据信息会实时保存到 Redis 数据库中，以供其它程序进行读取。

5.2. 推荐模块

推荐模块是本系统的核心模块，主要包含新闻文本的建模、候选集生成，候选集排序以及新闻热度计算等功能，本章节主要介绍新闻文本的建模、候选集生成以及候选集排序的算法实现。

5.2.1. 新闻文本建模算法实现

对新闻文本的建模，主要包括分词处理、停用词过滤、标题词向量化以及新闻 Embedding 生成这四个步骤，这些步骤在上文推荐算法设计中，已经有了较为详细的描述，此处仅对具体的算法实现进行一个补充说明。

该算法的代码如下所示，其中，参数 news_titles 就是要进行 Embedding 化的新闻标题列

表。

```
def generate_title_embedding(self, news_titles):
    assert self.news_model is not None
    encoder = self.news_model.get_layer("encoder") # 获取模型编码层
    cut_titles_seq = self.cut_titles(news_titles) # 对标题进行分词、停用词过滤处理
    title_seq = self.get_title_seq(cut_titles_seq) # 将词转化为词序列
    news_embedding = []
    index = 0
    for news_title in title_seq:
        print("编码标题: " + news_titles[index])
        index += 1
        _, encoder_state = encoder(tf.constant([news_title])) # 将新闻词序列转为新闻
```

Embedding

```
        news_embedding.append(np.array(encoder_state[0]).reshape(200, ).tolist())
    return news_embedding
```

上述代码可以大致分为两部分，一部分是对新闻标题的预处理，一部分是使用 LSTM 自编码器进行标题 Embedding 的生成。

在代码中，首先会获取 LSTM 自编码器的 Encoder 部分，接着对新闻标题进行分词、停用词过滤以及标题词向量化。最后，将所有预处理好的新闻标题送入 Encoder 进行新闻的 Embedding 生成，具体生成的新闻 Embedding 是一个 200 维的稠密向量。生成完成后，会保存在数据库中，具体的数据如下图所示。

news_embedding		embedding []		
_id	ObjectId	0 Double	1 Double	2 Double
1	"comos-kmxzfmk6823880"	0.9999714493751526	0.4035078287124634	0.1587841659784317
2	"comos-kmxzfmk6825420"	0.9999521970748901	-0.9964526891708374	-0.08510897308588028
3	"comos-kmyaawa9645295"	0.999939501285553	-0.976283609867096	-0.062027595937252045
4	"comos-kmxzfmk6778285"	0.9999598860740662	-0.9266424775123596	0.03736549988389015
5	"comos-kmxzfmk6765879"	0.9999276995658875	-0.9901629686355591	-0.029127372428774834
6	"comos-kmxzfmk6730531"	0.9999791383743286	-0.9559918642044067	0.03213917464017868
7	"comos-kmxzfmk6732842"	0.9999794363975525	-0.3695846199989319	-0.058037545531988144
8	"comos-kmxzfmk6720471"	0.9999477863311768	-0.8544952869415283	0.03704822063446045
9	"comos-kmyaawa9575905"	-0.18298396468162537	-0.9891718626022339	-0.06788446754217148
10	"comos-kmxzfmk6717431"	0.9999603629112244	-0.7987977862358093	-0.021523524075746536
11	"comos-kmyaawa9559016"	0.9999656081199646	-0.972977340221405	-0.0027140993624925613
12	"comos-kmxzfmk6676830"	0.999682216835022	-0.995180070400238	-0.06748891621828079
13	"comos-kmyaawa9513067"	0.9997971653938293	-0.996161699295044	0.02546902559697628

图 5-3 新闻 Embedding

5.2.2. 候选集生成算法实现

本系统的候选集生成算法主要包含四种召回策略，分别是兴趣召回、用户历史新闻召回、用户高评分新闻召回以及热点新闻召回。其中兴趣召回、用户历史召回以及用户高评分新闻

召回的实现思路较为类似，所以此处主要介绍一下兴趣召回的实现。

1. 兴趣召回算法实现

基于用户兴趣的召回算法流程如图 5-4 所示。首先，读取用户的兴趣标签列表，然后将用户的每一个兴趣标签转化为兴趣标签词向量，并保存到一个兴趣标签词向量矩阵中，然后对矩阵的行向量进行平均，得到用户兴趣嵌入，并计算用户兴趣与候选新闻的余弦相似度，最后，将所有求得的相似度值进行排序，取前几个放入新闻候选集中。



图 5-4 兴趣召回算法流程图

具体的算法代码如下所示。

```
def recall_by_interesting_tags(interesting_tags, recent_news_embedding, recall_rate=0.6):  
    # 基于用户标签来召回新闻  
    global _embedding_matrix  
    if len(interesting_tags) == 0:  
        # 如果该用户没有兴趣标签，则返回零向量  
        return np.zeros((200, ))  
    user_interest_embedding_matrix = [] # 用户兴趣向量矩阵  
    interesting_tags_cut = cut_titles(interesting_tags)  
    for interesting_tag_cut_one in interesting_tags_cut:  
        for interesting_tag in interesting_tag_cut_one:  
            # 获得兴趣标签对应的词向量 id  
            idx = _word2idx.get(interesting_tag, -1)  
            if idx != -1: # 如果词向量存在  
                user_interest_embedding_matrix.append(_embedding_matrix[idx]) # 将词向量添加到兴趣矩阵中  
    # 对兴趣矩阵求平均值，得到用户兴趣 embedding  
    user_interest_embedding = np.mean(user_interest_embedding_matrix, axis=-2).tolist()  
    # 计算余弦相似度  
    news_score = cosine_similar(recent_news_embedding, user_interest_embedding)  
    # 对余弦相似度进行索引排序  
    news_index = np.argsort(news_score)  
    return news_index[:int(len(news_score) * recall_rate)] # 返回指定召回率的新闻候选集列表
```

表

5.2.3. 候选集排序算法实现

在得到新闻候选列表后，需要使用排序算法对新闻候选集进行进一步的处理，该部分主要对新闻候选集进行排序工作，从而生成最后的用户新闻推荐列表。本文所采用的排序模型是 Wide&Deep 结构，该模型是一个点击率预估的深度学习模型。该模型会根据用户的融合特征来对候选新闻进行重新评分，并且在这个再排序的过程中，不会受到候选生成阶段的影响。

排序算法使用的模型结构具体如下图所示。

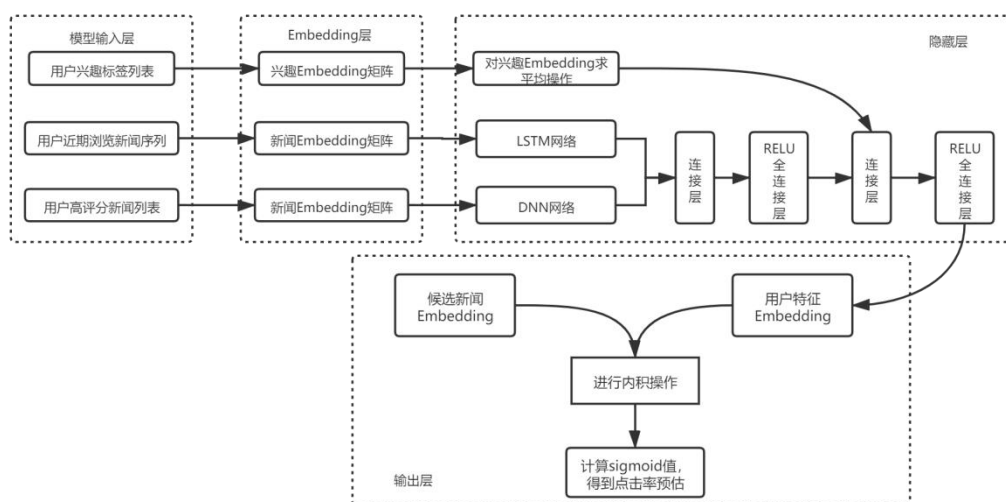


图 5-5 排序模型结构图

排序模型分为四个部分：输入层、嵌入层、隐藏层和输出层。首先，用户的相关信息会进入到模型的输入层，然后经过 Embedding 层，将各信息序列转为相应的 Embedding 序列，接着这些 Embedding 序列会经过隐藏层部分，其中用户兴趣 Embedding 序列会进行求平均操作，得到一个特征提取后的用户兴趣 Embedding，用户近期浏览新闻 Embedding 序列以及用户高分新闻 Embedding 序列会分别经过 LSTM 以及 DNN 网络，然后将这两个网络的输出进行拼接，得到一个维度更高且提取出了用户新闻特征的 Embedding，然后将这个 Embedding 经过全连接层，从而对用户新闻特征进行融合，进而得到用户新闻的融合特征，最后将这个经过特征交叉融合过后的 Embedding 与用户兴趣 Embedding 进行拼接，再次经过一个全连接层进行特征融合，此时就得到了一个可以用来表征用户特征的 Embedding。有了用户特征 Embedding，就可以与候选新闻进行内积运算，运算完后，再使用激活函数为 sigmoid 的神经元，计算得到候选新闻的点击率预估值，对所有候选新闻计算完对应的点击率预估值之后，排序即可得到用户新闻推荐列表。

该算法部分的核心代码如下所示：

```
def call(self, history_inputs, interesting_inputs, high_score_inputs):  
    # 获得用户历史新闻 embedding 序列  
    history_emb = self.news_embedding(history_inputs)  
    history_bn = self.bn(history_emb) # 正则化处理  
    history_out = self.history_layer(history_bn) # 提取用户历史新闻特征
```

```

# 获得用户历史新闻 embedding 序列
high_score_news_emb = self.news_embedding(high_score_inputs)
high_score_news_bn = self.bn(high_score_news_emb) # 正则化处理
high_score_out = self.high_score_layer(high_score_news_bn) # 提取用户高评分新闻特征
connect_out = self.connect([history_out, high_score_out]) # 新闻特征拼接
merge_out = self.merge_news_layer(connect_out) # 新闻特征融合
# 获取用户兴趣标签 embedding 序列
interesting_emb = self.label_embedding(interesting_inputs)
interesting_bn = self.bn(interesting_emb) # 正则化处理
interesting_mean_out = self.mean_layer(interesting_bn) # 计算兴趣均值
connect_out2 = self.connect([merge_out, interesting_mean_out]) # 新闻融合特征与用户
兴趣特征拼接

user_feature_out = self.user_feature_layer(connect_out2) # 融合得到用户特征
return user_feature_out

```

5.3. 业务系统模块

业务系统模块由前端和后端组成。前端主要负责页面信息的显示,后端主要负责 MongoDB 等数据库的读写操作,具体的前后端交互逻辑,如图 5-6 所示。

如图所示,Node.js 服务器位于前端。当用户向 Node.js 服务器发送页面请求时,节点服务器首先向后端服务器请求相应的 API 接口,获取页面所需的数据。在后端服务器处理完所有数据后,将返回到 Node 服务器。然后,Node 服务器根据现有的页面模板将请求的数据呈现给页面,最后将页面数据发送给用户浏览器。

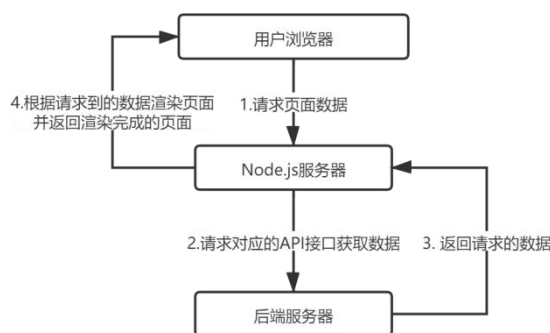


图 5-6 业务系统前后端交互逻辑图

5.3.1. 前端部分实现

业务系统的前端主要用来展示用户的推荐结果、用户新闻的点击和浏览,用户标签的设置、相似新闻推荐、新闻评分以及新闻热榜等功能。

(1) 用户新闻推荐列表展示

给用户推荐的新闻以列表的形式进行展示。当用户访问新闻列表或者要浏览新闻时,前端就通过后端的新闻 API 接口来获取新闻数据,后端查询数据库之后,返回所需要的新闻

数据给前端，具体的新闻推荐列表展示，如下图所示。



图 5-7 新闻推荐列表

在新闻推荐列表中，用户不仅可以看到新闻的标题信息，还可以看到新闻的作者、新闻观点、新闻发布日期、新闻所属的新闻频道以及新闻的当前热值。

(2) 修改用户标签后，新闻推荐结果不同

用户登录后，用户可以进行兴趣标签选择，每次修改完兴趣标签后，前端将修改后的用户标签数据发送给后端，后端根据用户修改后的标签数据，调用推荐模块的相关服务，重新生成用户的推荐列表，具体的标签选择框，如下图所示。



图 5-8 用户兴趣标签选择框

在兴趣标签选择框中，用户不仅可以选择已有的兴趣标签，还可以自定义兴趣标签，但每一个用户至多只能拥有 5 个兴趣标签。下图是用户已经选择的兴趣标签，以及用户的一些基本信息等。



图 5-9 用户已选兴趣标签

(3) 新闻价值评分

用户在登陆之后，可以对看过的新闻进行评分，评分的分值从 1 分到 5 分不等，高于 3 分的属于高评分新闻数据，同时，用户的新闻评分数据可以用于推荐算法部分，进而生成更为符合用户兴趣的新闻推荐列表，具体的新闻评分部分见如下图所示。

评分: ★★★★★ ☆

新闻价值: 4

图 5-10 新闻评分

(4) 新闻相似推荐

浏览完一条新闻后，新闻下面会有一个类似新闻的推荐列表。通过计算新闻嵌入与最近新闻嵌入之间的余弦相似度并进行排序，从而得到新闻相似列表，在得到与当前新闻最接近的 30 篇新闻后，将其缓存到 Redis 数据库中，设置缓存过期时间为 3 小时，这样可以显著减少后续服务器的计算量，然后随机从这 30 篇新闻中抽取 5 篇新闻，作为相似推荐结果返回给前端。相似推荐列表见下图所示。



图 5-11 相似推荐列表

(5) 新闻浏览页面

用户在点击新闻推荐列表中的新闻后，后端服务器便会检索该新闻的相关数据信息，并返回给前端显示，结果如下图所示。



图 5-12 新闻浏览页面图

用户的每一次新闻点击记录都会保存在数据库中，以供推荐服务计算用户新闻列表推荐时使用。

(6) 新闻频道导航栏

在首页的左侧，有一个新闻频道导航栏。用户可以点击自己想看的新闻频道，其中除了第一个推荐频道的列表数据是通过推荐算法计算得出的，其余频道的新闻列表均是按照新闻发布时间进行排序的，这样的安排可以使用户既能够看自己喜欢看的新闻，也可以浏览各领域最新发布的新闻，具体的新闻频道导航栏见下图所示。

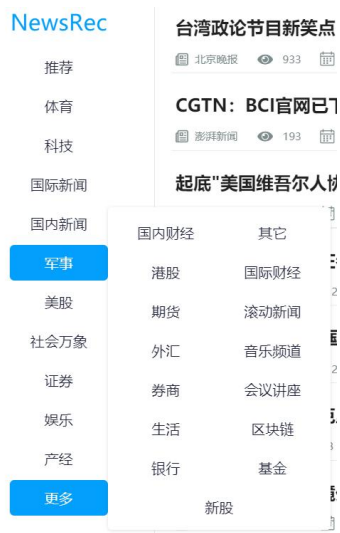


图 5-13 新闻频道导航栏

(7) 新闻热榜

新闻热榜用于展示每一个新闻频道当前热度值最高的新闻，该热度值的计算公式见上文候选集生成算法实现的热点新闻召回部分。

新闻热榜会根据当前所属新闻频道的不同而产生相应的变化，如果当前所属的新闻频道是推荐频道，则新闻热榜显示的是所有新闻中热度值最高的新闻。

新闻热榜	
1. 美国暂停使用强生疫苗，白宫称百日...	🔥 3780
2. 山东墨龙续跌超10% 预计首季亏损9...	🔥 3707
3. 日本核废水还有其他出路吗 五种备...	🔥 3492
4. 机构观点：IMF上调亚洲经济预估，...	🔥 3487
5. 新规发布！化妆品功效宣传不准自卖...	🔥 3480
6. 强生疫苗问题扰乱接种进程 欧盟称...	🔥 3471
7. 中国信通院完成国内首家TAG数字广...	🔥 3450
8. 4月15日，华夏、中融、广发、融通...	🔥 3449

图 5-14 所有新闻热榜

如果当前所处的新闻频道不是推荐频道，则会显示当前频道中热度值最高的新闻，例如当前处于军事板块，则会显示军事板块中热度值最高的几篇新闻，如下图所示。

军事热榜	
1. 日本决定核废水排海 外国网友:这是...	🔥 3244
2. 突发！日媒称日本“番茄酱之王”停...	🔥 3238
3. 外交部:日本行为极不负责任 保留作...	🔥 3188
4. 仅剩68位！南京大屠杀幸存者陈文英...	🔥 3009
5. 一个华裔美国人的扎心困惑：中国...	🔥 2912
6. 起底“美国维吾尔人协会”：接受NED...	🔥 2878
7. 胡锡进：现实些 台防卫能力比1949...	🔥 2826
8. 我国多地渔民捞起外军海上间谍设备...	🔥 2813

图 5-15 军事新闻频道热榜

5.3.2. 后端部分实现

业务系统的后端主要负责调用推荐模块的服务生成用户新闻推荐列表，新闻数据的检索以及一些定时任务等，此处由于篇幅限制，仅介绍一下新闻推荐列表的生成策略以及新闻热榜的生成。

(1) 新闻推荐列表的生成

每一个用户的新闻推荐列表都是不一样的，该列表是根据用户的兴趣标签，用户的浏览记录以及用户高评分新闻等信息生成出来的，业务系统后端主要负责将这些数据检索出来后，通过 HTTP 请求发送给推荐服务模块进行相关的生成操作，具体的流程见图 5-16。

由图 5-16 可知，当用户请求新闻推荐列表时，后端服务器首先会判断该用户是否已经登录，如果用户没有登录，则直接返回按新闻热度值排序的新闻列表；如果用户登录了，则会从当前请求线程中读取用户的相关信息，然后从数据库中检索该用户是否有推荐列表，如果存在推荐列表，则对该列表进行分页处理，并返回请求的页面数据。

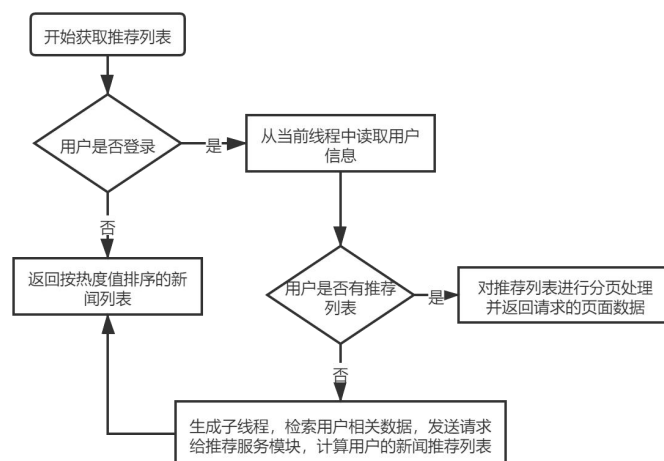


图 5-16 推荐流程图

一般来说，用户的推荐列表每隔 5 分钟，后端服务器便会进行自动更新，但如果该用户是在更新间隔时期注册的，则该用户因为是刚注册的，可能会没有推荐列表，此时后端服务器会生成一个子线程，该子线程负责将用户的相关数据发送给推荐服务模块，让其计算该用户的推荐列表，但因为该计算较为耗时，所以为了能够尽快返回数据，该次请求会返回按热度排序的新闻列表，待下一次请求时，便可返回已经计算好的新闻推荐列表。

(2) 新闻热榜的生成

系统的新闻热榜生成是一个定时任务，每隔 10 分钟就会自动更新一次，每一次更新都会生成各频道的新闻热榜数据，并将其保存到 Redis 数据库中以供访问，具体的更新流程如图所示。

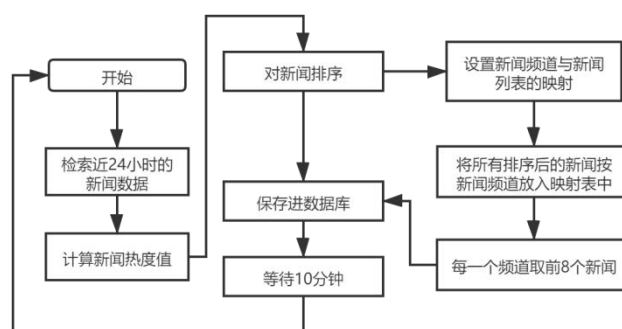


图 5-17 新闻热榜生成

如图所示，后端服务器每隔 10 分钟便会进行一次新闻热榜的计算生成，首先会检索最近 24 小时发布的新闻数据（由图 4-13 可知，超过了 24 小时，新闻的热度基本为 0），然后对这些新闻数据计算其热度值，计算完成后，对其进行排序操作并保存到数据库中。该排序完的新闻热度列表，可用于未登录用户的推荐。

在得到新闻热度值排序的列表后，设置一个映射表，该映射表的 Key 是新闻频道，Value 是一个优先级队列，接着对新闻热榜进行遍历，将新闻添加到各个不同新闻频道的优先级队列中，这样最终就生成了各新闻频道的热榜，取每一个热榜的前 8 个新闻作为展示数据，将

其保存到 Redis 数据库中。

5.4. 系统效果展示

本部分介绍了系统的核心功能，包括新闻推荐列表的生成、同类新闻的推荐和新闻评分。

5.4.1. 用户新闻浏览

点击新闻后，用户可以进入新闻浏览页面，浏览新闻内容。



图 5-18 新闻浏览

5.4.2. 用户新闻评分

用户在登录后可以对当前新闻进行评分操作，但如果没有登录，则无法进行评分。



图 5-19 已登录用户可进行评分

如下图所示，如果当前用户没有登录，则无法进行评分操作。



图 5-20 未登录用户无法评分

5.4.3. 新闻推荐效果比对

当用户未登录时，因为系统缺乏用户的个人数据，也就无法进行新闻个性化推荐，所以返回的推荐列表都是按照新闻热度值进行排序。



图 5-21 未登录的新闻首页（新闻热度值）

下表是系统未登录时返回的推荐列表，这个列表是按照新闻热度值进行排序的。

表 5-1 未登录时新闻推荐列表

新闻标题名	新闻类别
下周榆林煤价或又将全面起飞!榆林横山区煤价率先提涨 10%-20%	期货
中国信通院完成国内首家 TAG 数字广告流量反欺诈国际认证	国内财经
小摩:美国税率调整将对苹果盈利构成不利影响	其它
恒指下跌 0.1%阿里巴巴-SW 低开 1.6%领跌蓝筹	其它
亚信科技:精细化运营服务为智慧高速提速	证券

在用户登录后，系统会根据用户的个人数据进行个性化推荐，该用户的新闻推荐列表如下图所示。



图 5-22 用户新闻推荐列表（浏览新闻前）

如图所示，该用户的兴趣标签为生活、军事、娱乐以及科技，由于用户是新创建的账户，还没有任何浏览历史以及评分数据，此时系统主要根据兴趣标签进行推荐，具体前几条推荐数据如下表所示。

表 5-2 新用户新闻推荐列表（浏览新闻前）

新闻标题名	新闻类别	用户标签
大陆军机超低空飞入台所谓“防空识别区”意味着什么	军事	生活 军事 娱乐 科技
特斯拉回应德州致命事故：事故发生时有人在驾驶	科技	
人民日报：中超版权有收益球迷也受益	体育	
realme 配件全家桶爆料：将于 4 月 30 日在海外发布	科技	
泰国总理预计将于周五宣布实施更严格的防控措施	国际	

从上述表格中的数据可以得出，有了用户的一些个人数据后，推荐效果更为明显，并且给该用户推荐的新闻与其兴趣标签吻合，推荐的结果较为符合预期。

表 5-3 用户新闻近期浏览记录

新闻标题名	新闻类别
蔚来：未参与推动任何第三方参与某品牌维权行为	科技
知情人士：360 将继续投资智能汽车领域暂不涉及硬件层面	科技
委内瑞拉军队在边境地区和哥伦比亚武装激战 72 小时	军事
张超烈士牺牲 5 周年 中国海军已进入双航母时代	军事
合买团 11+2 复式票擒双色球 568 万：刚合作整一个月	体育

当用户与系统进行了一些交互后，例如浏览了上表 5-3 所示的新闻，并对下表 5-4 所示的新闻进行打分操作后，用户的新闻推荐列表产生了变化，如图 5-23 所示。

表 5-4 用户新闻打分记录

新闻标题名	新闻类别	打分
张超烈士牺牲 5 周年 中国海军已进入双航母时代	军事	5
俄海军升级型无畏级编入太平洋舰队常备战力序列	军事	4.5
委内瑞拉军队在边境地区和哥伦比亚武装激战 72 小时	国际	2.5
上海车展车顶维权车主回应通行证来源：实名申请 合理合法	科技	4.5
知情人士：360 将继续投资智能汽车领域 暂不涉及硬件层面	科技	5

下图是用户进行了浏览和打分操作后，重新生成的用户新闻推荐列表。

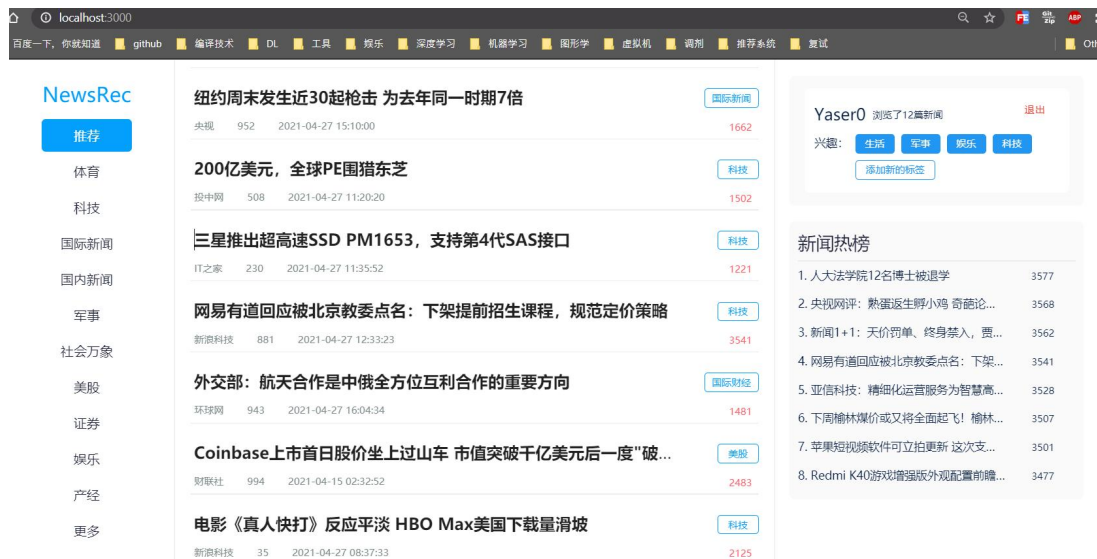


图 5-23 用户新闻推荐列表（浏览新闻后）

前几条数据如下表 5-5 所示，该表与表 5-2 对比可以看出，用户在浏览了新闻并且进行评分操作后，新闻推荐列表产生了明显的变化。

表 5-5 新用户新闻推荐列表（浏览新闻后）

新闻标题名	新闻类别	用户标签
纽约周末发生近 30 起枪击 为去年同一时期 7 倍	国际	生活
200 亿美元，全球 PE 围猎东芝	科技	军事
三星推出超高速 SSD PM1653, 支持第 4 代 SAS 接口	科技	娱乐
网易有道回应被北京教委点名：下架提前招生课程，规范定价策略	科技	科技
外交部：航天合作是中俄全方位互利合作的重要方向	国际	

5.5. 本章小结

本章节主要对个性化新闻推荐系统的实现做了一个详尽的论述，大致包括以下几个方面的实现：

1. 爬虫模块的新闻爬取以及爬虫日志分析的实现。
2. 推荐模块中候选集生成算法以及候选集排序算法的具体实现。
3. 业务系统中前后端的具体实现。

本章节还对系统实现后的效果进行了相关的展示。从效果图可以看出，该新闻推荐系统的推荐效果十分明显，可以有效的为不同的用户推荐不同的新闻，且推荐的新闻都较为符合用户的兴趣。

6. 结束语

本文主要讨论了一个结合深度学习的新闻推荐系统的设计与实现。将深度学习与推荐系统相结合，实现了一个深度学习推荐系统。同时，在深度学习的加持下，可以更好地捕捉用户的兴趣特征，进而为用户提供更好的推荐服务。

首先，本文因为缺乏足够的新闻数据来进行推荐，使用了爬虫技术，从新浪新闻抓取了大量的新闻数据以供推荐使用；然后，根据推荐算法的需要，将新闻原本的文本数据，通过一系列处理，包括分词、停用词过滤、标题词向量化以及新闻 Embedding 生成这四步，从而生成了可供推荐算法使用的输入数据；接着，使用多路召回策略，包括兴趣召回、历史召回、高评分召回以及热点召回，来对抓取的海量新闻数据进行召回操作，从而生成新闻候选集；最后，将这些候选新闻通过 Wide&Deep 模型计算得到用户对这些候选新闻的点击率预估结果，排序后得到用户的个性化新闻推荐列表。

本新闻推荐系统的推荐效果与预期相符合，对于不同的用户，可以为其推荐感兴趣的新闻。本系统界面美观，易于用户的使用，且本系统鉴于良好的模块化设计，各模块之间相互协作良好，保证了系统运行的稳定性。

最后，作为一个新闻推荐系统，系统功能还可以进一步进行扩展，同时，现阶段系统的用户行为数据采集还不够完善，缺乏对用户新闻浏览所用的设备、所处环境以及新闻浏览的时长等数据的采集与分析，如果有了这些数据，相信推荐算法可以更有效地挖掘用户的特征，进而提高推荐的质量。另外，当前推荐算法的文本建模部分还有进一步优化的空间，可以尝试将 LSTM 自编码器更换为 Bert 模型，该模型是谷歌新发布的语言模型，有着更好的语言建模效果。

参考文献

- [1] 姚婷. 基于协同过滤算法的个性化推荐研究[D].北京理工大学,2015.
- [2] 周会祥,盛武平. 一种基于 TF-IDF 方法优化的新闻关键词提取方法及系统[P]. 江西省: CN112256843A,2021-01-22.
- [3] Suvash Sedhain,Aditya Krishna Menon,Scott Sanner,Lexing Xie. AutoRec[P]. World Wide Web,2015.
- [4] Shan, Ying, et al. "Deep crossing: Web-scale modeling without manually crafted combinatorial features." Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining. 2016.
- [5] Cheng, Heng-Tze, et al. "Wide & deep learning for recommender systems." Proceedings of the 1st workshop on deep learning for recommender systems. 2016.

- [6] Guo, Huifeng, et al. "DeepFM: a factorization-machine based neural network for CTR prediction." arXiv preprint arXiv:1703.04247 (2017).
- [7] Ramos, Juan. "Using tf-idf to determine word relevance in document queries." Proceedings of the first instructional conference on machine learning. Vol. 242. No. 1. 2003.
- [8] He, Xiaofei, et al. "Neighborhood preserving embedding." Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1. Vol. 2. IEEE, 2005.
- [9] Gers, Felix A., Jürgen Schmidhuber, and Fred Cummins. "Learning to forget: Continual prediction with LSTM." (1999): 850-855.
- [10] 王喆.深度学习推荐系统[M].电子工业出版社:北京,2020-3:1.
- [11] 甄玉敏. 基于知识图谱的个性化新闻推荐系统的设计与实现[D].北京交通大学,2020.
- [12] 郭小勇. 基于深度学习的新闻推荐算法研究[D].重庆邮电大学,2020.
- [13] 李博. 基于项目特征和排序学习的新闻推荐系统设计及实现[D].北京邮电大学,2019.
- [14] 李鲁君. 基于词嵌入的个性化新闻推荐算法研究[D].上海师范大学,2019.
- [15] 王明亚. 基于词向量的文本分类算法研究与改进[D].华东师范大学,2016.
- [16] 宋润青. 基于词向量和文本特征的事件提取[D].西安电子科技大学,2015.
- [17] 张鹏华. 基于移动互联网的个性化新闻推荐系统研究与实现[D].吉林大学,2012.
- [18] 吴多坚. 基于 word2vec 的中文文本相似度研究与实现[D].西安电子科技大学,2016.
- [19] 杨飞. 基于 LSTM 的文本相似度识别方法研究[D].吉林大学,2018.
- [20] 余钊. 基于 Attention 机制的短文本相似度计算[D].哈尔滨工业大学,2020.
- [21] 魏图南. 个性化新闻推荐系统的设计与实现[D].河北科技大学,2019.
- [22] 王松.Spring Boot+Vue 全栈开发实战[M].清华大学出版社:北京,2019:13-36
- [23] 田萱,丁琪,廖子慧,孙国栋. 基于深度学习的新闻推荐算法研究综述[J]. 计算机科学与探索,,:1-31.
- [24] 基于用户行为的新闻推荐算法的研究[J]. 李增,刘羽,李诚诚. 计算机工程与科学. 2020(03)
- [25] 王豪. 个性化新闻推荐系统的设计与实现[D].北京邮电大学,2019.
- [26] 孙俊. 基于深度神经网络的推荐系统模型研究[D].浙江师范大学,2020.
- [27] 王喆.深度学习推荐系统实战[EB/OL].<https://time.geekbang.org/column/intro/349>,2020-09-21.
- [28] Shumpei Okura,Yukihiro Tagami,Shingo Ono,Akira Tajima. Embedding-based News Recommendation for Millions of Users[P]. Knowledge Discovery and Data Mining,2017.
- [29] Wu, Chuhan, et al. "Neural news recommendation with attentive multi-view learning." arXiv preprint arXiv:1907.05576 (2019).
- [30] Zheng, Guanjie, et al. "DRN: A deep reinforcement learning framework for news recommendation." Proceedings of the 2018 World Wide Web Conference. 2018.
- [31] Liu, Jiahui, Peter Dolan, and Elin Rønby Pedersen. "Personalized news recommendation based on click behavior." Proceedings of the 15th international conference on Intelligent user interfaces. 2010.

致谢

本文是在黄群老师的指导，并通过跟其他同学的学习交流下完成的。从论文的立题到最终完成，他们都给予了关怀和帮助。特别感谢黄群老师对毕业设计的教导与监督，如果没有老师的帮助我不可能那么顺利地完成毕业设计论文。然后，感谢我的父母，拥有他们的支持、鼓励和信任，我才能坚定不移去克服学习、生活等方面遇到的困难。

最后，对百忙之中抽出时间对我的论文进行评审的老师表示感谢。