襄阳职业技术学院（毕业）论文

基于 Spark 的电商推荐系统的设计与实现

专业班级：大数据技术与应用 2002

学　　生：刘星明

学　　号：202002331

指导教师：史润

教学单位：信息技术学院

毕 业 届：2023 届

毕 业 设 计（论 文）课 题 任 务 书

信息技术学院 系（院） 大数据技术 专业 2002 班 学生 刘星明

毕业设计(论文)课题 基于Spark的电商推荐系统的设计与实现

二、毕业设计(论文)工作自 2023 年 01 月 10 日起至 2023 年 05 月 30 日止

三、毕业设计(论文)进行地点 襄阳职业技术学院学院

四、毕业设计(论文)的内容要求

本课题采用Spark全栈技术实现商品的个性化推荐，借助大数据项目常用的一些工具，使JavaEE系统部署业务服务，将个性化推荐作为电商网络营销的一种新的手段，能为电商公司带来巨大的利润。国内最大的电商平台——阿里巴巴（淘宝）自从使用了电商推荐服务以后，淘宝的商品关联支付转化率提升了近100%，商品的客单价提升了29%，可见推荐系统在电子商务实现精准营销[2]技术研究中的重要意义。

电商推荐系统的目标是根据用户浏览商品以及对商品评分等行为，以此推测用户对其他相似商品的喜好程度，智能预测符合用户口味偏好的商品，并将推荐结果展示给用户[3]。可想而知，系统仅仅可通过用户近期浏览过的商品和对商品的评分情况就能知道用户喜欢什么样的口味、价位、质量等因素的商品，可以减少用户浏览商品的时间，而这样的推荐系统，一定会受到电商平台和电商用户的青睐。因此，有了电商个性化推荐系统和准确度较高的推荐算法，能为电商企业和消费者提升粘合度，达到商品的“精准营销”的目的，不仅能给电商平台提升市场竞争力，而且给用户提升网上购物的消费体验，实现电商交易的“双赢”。

五、参考文献

[1]伍之昂,曹杰. 电子商务推荐系统导论[M]. 科学出版社

[2]陆俊尧,李玲娟.基于Spark的协同过滤算法并行化研究[J].计算机技术与发展

[3]周恒新. 基于Hadoop架构的商业推荐引擎协同过滤算法设计与实现[D]

[4]尚硅谷. 机器学习和推荐系统项目实战

指导教师 史润

学 生 刘星明

目录

目录 2
摘要 3
关键词 3
1、前言 4
2、项目体系架构设计 4
2.1 项目系统架构 4
2.1.1 数据存储部分 4
2.1.2 离线推荐部分 5
2.1.3 实时推荐部分 5
2.2 项目数据流程 5
2.2.1 初始化部分 5
2.2.2 离线推荐部分 5
2.2.3 实时推荐部分 5
2.3 数据模型 6
2.3.1 Product —— 商品数据表 6
2.3.2 Rating —— 用户评分表 6
3、项目环境搭建 6
3.1 基于 DOCKER 容器构建所需镜像 6
3.1.1 Centos7 部署 docker 服务 6
3.1.2 编写 Dockerfile，构建所需镜像 7
3.2 IDEA 中创建项目，初始化数据到 MONGODB 7
4、离线推荐服务 7
4.1 基于离线数据的统计推荐 8
4.1.1 历史热门商品统计 8
4.1.2 最近热门商品统计 8
4.1.3 商品平均的分统计 8
4.2 基于隐语义模型的协同过滤推荐 8
4.2.1 训练 ALS 模型 8
4.2.2 使用 ALS 模型预测 9
4.2.3 商品相似度矩阵 9
4.3 算法实现 10
5、实时推荐系统 10
5.1 实时推荐算法流程流程 10
5.2 实时推荐模型算法设计 10
5.3 算法实现 12
6、冷启动问题处理 12
7、结束语 13
参考文献 13

基于 Spark 的电商推荐系统的设计与实现

学生：刘星明

指导教师：史润

摘要

现如今，电商、大数据逐渐成为当今时代信息技术发展的主题，基于大数据应用技术实现的电商推荐系统逐渐成为各大电商平台研究推荐系统服务的重点。阿里巴巴集团创始人马云也多次对外界强调阿里巴巴是一家大数据公司而不是电商企业，这足以说明数据是电商行业的命脉，可见大数据在电商行业中的重要性。然而，如何处理海量的用户行为数据，从用户行为中挖掘出用户的商品购买偏好是所有电商平台的重点难题，至今没有一家公司能做到推荐服务 100%准确。在新的大数据时代里，一款名为 Spark 的大数据计算框架如同它的名字一样让星星之火照亮整个大数据世界，Spark 以高效的海量数据处理能力等诸多优点，现已成为众多电商平台实现推荐服务的首选技术框架。

关键词

电商，大数据分析，Spark，推荐系统，数据挖掘

1. 前言

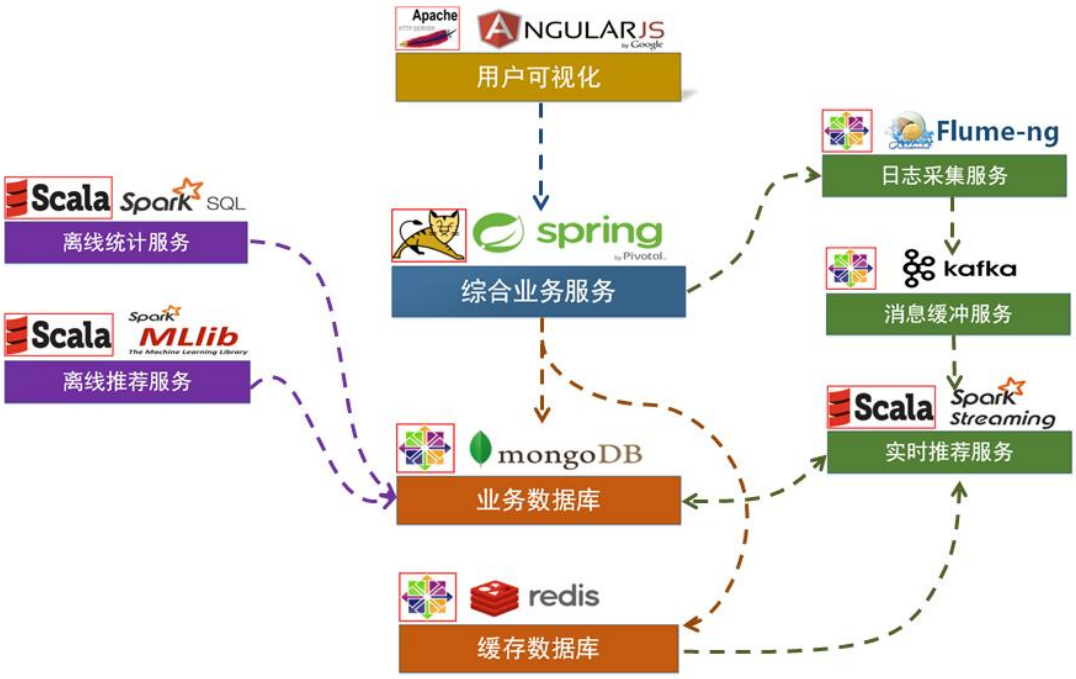
随着信息技术的不断发展，如今互联网已经成为了人们日常生活中密不可分的一部分。人们每天都会在互联网上进行各种各样的活动例如看电影、购物、阅读新闻时事等，但随着互联网上信息的越来越多，人们发现越来越难以从互联网上的海量信息中找出最适合自己的那些，例如当你登录 Netflix 想要看一部电影放松时却不知道哪一部符合自己的口味。推荐系统的出现正是为了解决这种“信息过载”的问题，它会预测用户的需求并推荐给用户其最可能喜欢的内容，缓解了人们从海量信息中做出选择的烦恼。

项目以推荐系统建设领域知名的经过修改过的中文亚马逊电商数据集作为依托，以某电商网站真实业务数据架构为基础，实现了离线推荐与实时推荐体系，综合利用了协同过滤算法以及基于内容的推荐方法来提供混合推荐。

2. 项目体系架构设计

2.1 项目系统架构

本文着重实现下图中的离线和实时推荐模块，用户可视化与综合业务服务部分采用模拟方式。



2.1.1 数据存储部分

业务数据库：项目采用广泛应用的文档数据库 MongDB 作为主数据库，主要负责平台业务逻辑数据的存储。

缓存数据库：项目采用 Redis 作为缓存数据库，主要用来支撑实时推荐系统部分对于数据的高速获取需求。

2.1.2 离线推荐部分

离线统计服务：批处理统计性业务采用 Spark Core + Spark SQL 进行实现，实现对指标类数据的统计任务。

离线推荐服务：离线推荐业务采用 Spark Core + Spark MLlib 进行实现，采用 ALS 算法进行实现。

2.1.3 实时推荐部分

日志采集服务：通过利用 Flume-ng 对业务平台中用户对于商品的一次评分行为进行采集，实时发送到 Kafka 集群。

消息缓冲服务：项目采用 Kafka 作为流式数据的缓存组件，接受来自 Flume 的数据采集请求。并将数据推送到项目的实时推荐系统部分。

实时推荐服务：项目采用 Spark Streaming 作为实时推荐系统，通过接收 Kafka 中缓存的数据，通过设计的推荐算法实现对实时推荐的数据处理，并将结构合并更新到 MongoDB 数据库。

2.2 项目数据流程

2.2.1 初始化部分

通过 Spark SQL 将系统初始化数据加载到 MongoDB 中。

2.2.2 离线推荐部分

离线统计服务从 MongoDB 中加载数据，将商品平均评分统计、商品评分个数统计、最近商品评分个数统计三个统计算法进行运行实现，并将计算结果回写到 MongoDB 中；离线推荐服务从 MongoDB 中加载数据，通过 ALS 算法分别将用户推荐结果矩阵、相似度矩阵回写到 MongoDB 中。

2.2.3 实时推荐部分

Flume 从综合业务服务的运行日志中读取日志更新，并将更新的日志实时推送到 Kafk

a 中；Kafka 在收到这些日志之后，通过 kafkaStream 程序对获取的日志信息进行过滤处理，获取用户评分数据流 UID|MID|SCORE|TIMESTAMP，并发送到另外一个 Kafka 队列；Spark Streaming 监听 Kafka 队列，实时获取 Kafka 过滤出来的用户评分数据流，融合存储在 Redis 中的用户最近评分队列数据，提交给实时推荐算法，完成对用户新的推荐结果计算；计算完成之后，将新的推荐结构和 MongDB 数据库中的推荐结果进行合并。

2.3 数据模型

2.3.1 Product —— 商品数据表

字段名	字段类型	字段描述	字段备注
productId	Int	商品的 ID	
name	String	商品的名称	
categories	String	商品所属类别	每一项用“|”分割
imageUrl	String	商品图片的 URL	
tags	String	商品的 UGC 标签	每一项用“|”分割

2.3.2 Rating —— 用户评分表

字段名	字段类型	字段描述	字段备注
userId	Int	用户的 ID	
productId	Int	商品的 ID	
score	Double	商品的分值	
timestamp	Long	评分的时间	

3. 项目环境搭建

3.1 基于 docker 容器构建所需镜像

3.1.1 Centos7 部署 docker 服务

```
# 安装依赖，yum-util提供yum-config-manager功能，另外两个是devicemapper驱动依赖的
yum install -y yum-utils device-mapper-persistent-data lvm2

# 配置yum源（阿里云）
yum-config-manager --add-repo http://mirrors.aliyun.com/docker-ce/linux/centos/docker-ce.repo

# 查看所有仓库所有docker版本
yum-config-manager --add-repo http://mirrors.aliyun.com/docker-ce/linux/centos/docker-ce.repo

# 安装所需版本
yum install docker-ce-18.03.1.ce

# 启动docker，加入开机启动
systemctl start docker;systemctl enable  docker
```

3.1.2 编写 Dockerfile，构建所需镜像

文件内容过多，因此将其放入笔者的 Github 的仓库中，graduation/基于 docker 构建环境.md at master · liuming13/graduation (github.com)。

3.2 idea 中创建项目，初始化数据到 MongoDB

利用 Spark SQL 分别读取 products.csv 和 ratings.csv 数据，将其分别写入 MongoDB 的 Product 和 Pating 中，用于模拟生产环境。

数据位于：ECommerceRecommendSystem/recommender/DataLoader/src/main/resources at master · liuming13/ECommerceRecommendSystem (github.com)

Spark SQL 实现：ECommerceRecommendSystem/DataLoader.scala at master · liuming13/ECommerceRecommendSystem (github.com)

4. 离线推荐服务

离线推荐服务是综合用户所有的历史数据，利用设定的离线统计算法和离线推荐算法周期性的进行结果统计与保存，计算的结果在一定时间周期内是固定不变的，变更的频率取决于算法调度的频率。

离线推荐服务主要计算一些可以预先进行统计和计算的指标，为实时计算和前端业务

相应提供数据支撑。

离线推荐服务主要分为统计推荐、基于隐语义模型的协同过滤推荐以及基于内容和基于 Item-CF 的相似推荐。

4.1 基于离线数据的统计推荐

离线统计服务，主要是根据已有的数据进行维度分析。最后会得出一个普遍性的结果。维度是可以由我们控制的，因此实现三个比较具有代表性的。数据来源为我们在 3.2 中初始化到 MongoDB 中的两张表。

4.1.1 历史热门商品统计

实现思路：统计历史热门商品，也就是说以商品 id 为维度，用它进行分组，最后对评分的的人数进行计数。

4.1.2 最近热门商品统计

实现思路：统计最近热们的商品，与历史热门商品相比，需要先过滤掉不是最近的用户评分记录，如何界定最近这个条件需要根据业务情况。

4.1.3 商品平均的分统计

实现思路：根据商品 id 分组，求出用户评分的平均数。

4.2 基于隐语义模型的协同过滤推荐

该模块是离线推荐的核心，因为有了它我们才可以根据每个用户的行为和特征，向其推荐他可能感兴趣商品，使用的技术是 SparkMLlib 实现，它的本质就是先对所有的物品进行分类，再根据用户的兴趣分类给用户推荐该分类中的物品，其算法已经 Spark 所实现。

4.2.1 训练 ALS 模型

具体思路：将 MongeDB 中 Product 和 Ratring 数据拿到，根据 productId 做 join，然后使用 ALS 利用用户 id，商品 id，用户对商品的评分，训练出隐语义模型。

模型的预测与评估：隐语义模型主要由 rank、iterations、lambda 三个参数控制。那么什么参数最优我们是无法事先知道的，但是我们可以对模型进行评估，在多种参数的设置下取误差最小的那三个参数。通常的做法是计算均方根误差（RMSE），考察预测评分与实际评分之间的关系。

$$\mathrm{RMSE} = \sqrt{\frac{1}{N}\sum_{t=1}^{N}(observed_t - predicted_t)^2}$$

4.2.2 使用 ALS 模型预测

具体思路：将用户 id 与商品 id 做笛卡尔积，去掉用户购买过此商品的记录，即可得到每个用户对其未购买过的商品的评分，最后根据用户 id 分组，根据评分列逆序，取出我们需要的前 n 个水平 id，最后将(用户 id, 推荐列表)格式的数据写入 MongoDB 的 UserRecs 表中。

4.2.3 商品相似度矩阵

通过 ALS 计算商品相似度矩阵，该矩阵用于查询当前商品的相似商品并为实时推荐系统服务。

离线计算的 ALS 算法，算法最终会为用户、商品分别生成最终的特征矩阵，分别是表示用户特征矩阵的 U(m x k)矩阵，每个用户由 k 个特征描述；表示物品特征矩阵的 V(n x k)矩阵，每个物品也由 k 个特征描述。

V(n x k)表示物品特征矩阵，每一行是一个 k 维向量，虽然我们并不知道每一个维度的特征意义是什么，但是 k 个维度的数学向量表示了该行对应商品的特征。

有了特征向量，我们就可以计算相似度了，这个时候计算向量与向量之间的相似度算法就成了我们所考量的对象，比较常见的由皮尔逊相关系数、欧氏距离、Spearman 秩相关系数、曼哈顿距离，笔者选用最为常见的余弦相似度算法，余弦相似度用向量空间中两个向量夹角的余弦值作为衡量两个个体间差异的大小。相比距离度量，余弦相似度更加注重两个向量在方向上的差异，而非距离或长度上。

$$T(x,y) = \frac{x \bullet y}{\|x\|^2 \times \|y\|^2} = \frac{\sum x_i y_i}{\sqrt{\sum x_i^2}\sqrt{\sum y_i^2}}$$

实现思路：通过前面训练出来的模型对象的 productFeatures 方法，拿到商品的特征向量，最后让其自己和自己做笛卡尔积，特征向量与特征向量之间做余弦相似度计算，最后根据相似度逆序，取与当前商品最相似的前 N 个水平，以（商品 id，最相似的前 N 个商品列表）的格式写入 MongoDB 的 PRODUCT_RECS 表中。

4.3 算法实现

https://github.com/liuming13/ECommerceRecommendSystem/blob/master/recommender/StatisticsRecommender/src/main/scala/com/ming/statistics/StatisticsRecommender.scala

https://github.com/liuming13/ECommerceRecommendSystem/blob/master/recommender/OfflineRecommender/src/main/scala/com/ming/offline/OfflineRecommender.scala

https://github.com/liuming13/ECommerceRecommendSystem/blob/master/recommender/OfflineRecommender/src/main/scala/com/ming/offline/ALSTrainer.scala

5. 实时推荐系统

实时推荐系统与离线推荐系统最大的不同就在于，实时推荐系统可以根据用户最近的行为和特征进行个性化推荐，而离线推荐系统根据用户第一次的行为到用户最后一次的行为特征进行推荐的。

另外，在实时推荐中由于时间性能上要满足实时或者准实时的要求，所以算法的计算量不能太大，避免复杂、过多的计算造成用户体验的下降。鉴于此，推荐精度往往不会很高。实时推荐系统更关心推荐结果的动态变化能力，只要更新推荐结果的理由合理即可，至于推荐的精度要求则可以适当放宽。

5.1 实时推荐算法流程流程

用户 u 对商品 p 进行了评分，触发了实时推荐的一次计算；

选出商品 p 最相似的 K 个商品作为集合 S；

获取用户 u 最近时间内的 K 条评分，包含本次评分，作为集合 RK；

计算商品的推荐优先级，产生<qID,>集合 updated_S；

将 updated_S 与上次对用户 u 的推荐结果 Rec 进行合并，产生新的推荐结果 NewRec；作为最终输出。

5.2 实时推荐模型算法设计

当用户 u 对商品 p 进行了评分，将触发一次对 u 的推荐结果的更新。由于用户 u 对商品 p 评分，对于用户 u 来说，他与 p 最相似的商品们之间的推荐强度将发生变化，所以选取与商品 p 最相似的 K 个商品作为候选商品。

每个候选商品按照“推荐优先级”这一权重作为衡量这个商品被推荐给用户 u 的优先

级。

这些商品将根据用户 u 最近的若干评分计算出各自对用户 u 的推荐优先级，然后与上次对用户 u 的实时推荐结果的进行基于推荐优先级的合并、替换得到更新后的推荐结果。

具体来说：

首先，获取用户 u 按时间顺序最近的 K 个评分，记为 RK；获取商品 p 的最相似的 K 个商品集合，记为 S；

然后，对于每个商品 q∈S ，计算其推荐优先级E_{uq}，计算公式如下：

$$E_{\mathrm{uq}}=\frac{\sum_{r\in RK} sim(q,r)\times R_r}{sim\_sum}+\lg\max\{incount,1\}-\lg\max\{recount,1\}$$

其中：

R_r表示用 u 对商品 r 的评分；

sim(q,r)表示商品 q 与商品 r 的相似度，设定最小相似度为 0.6，当商品 q 和商品 r 相似度低于 0.6 的阈值，则视为两者不相关并忽略；

sim_sum 表示 q 与 RK 中商品相似度大于最小阈值的个数；

incount 表示 RK 中与商品 q 相似的、且本身评分较高（>=3）的商品个数；

recount 表示 RK 中与商品 q 相似的、且本身评分较低（<3）的商品个数；

公式的意义如下：

首先对于每个候选商品 q，从 u 最近的 K 个评分中，找出与 q 相似度较高（>=0.6）的 u 已评分商品们，对于这些商品们中的每个商品 r，将 r 与 q 的相似度乘以用户 u 对 r 的评分，将这些乘积计算平均数，作为用户 u 对商品 q 的评分预测即

$$\frac{\sum_{r\in RK} sim(q,r)\times R_r}{sim\_sum}$$

然后，将 u 最近的 K 个评分中与商品 q 相似的、且本身评分较高（>=3）的商品个数记为 incount，计算 lgmax{incount,1}作为商品 q 的“增强因子”，意义在于商品 q 与 u 的最近 K 个评分中的 n 个高评分(>=3)商品相似，则商品 q 的优先级被增加 lgmax{incount,1}。如果商品 q 与 u 的最近 K 个评分中相似的高评分商品越多，也就是说 n 越大，则商品 q 更应该被推荐，所以推荐优先级被增强的幅度较大；如果商品 q 与 u 的

最近 K 个评分中相似的高评分商品越少，也就是 n 越小，则推荐优先级被增强的幅度较小；

而后，将 u 最近的 K 个评分中与商品 q 相似的、且本身评分较低（<3）的商品个数记为 recount，计算 lgmax{recount,1}作为商品 q 的“削弱因子”，意义在于商品 q 与 u 的最近 K 个评分中的 n 个低评分(<3)商品相似，则商品 q 的优先级被削减 lgmax{incount,1}。如果商品 q 与 u 的最近 K 个评分中相似的低评分商品越多，也就是说 n 越大，则商品 q 更不应该被推荐，所以推荐优先级被减弱的幅度较大；如果商品 q 与 u 的最近 K 个评分中相似的低评分商品越少，也就是 n 越小，则推荐优先级被减弱的幅度较小；

最后，将增强因子增加到上述的预测评分中，并减去削弱因子，得到最终的 q 商品对于 u 的推荐优先级。在计算完每个候选商品 q 的 后，将生成一组<商品 q 的 ID, q 的推荐优先级>的列表 updatedList：

$$\text{updatedList} = \bigcup_{q \in S} \{qID, E_{uq}\}$$

而在本次为用户 u 实时推荐之前的上一次实时推荐结果 Rec 也是一组<商品 m,m 的推荐优先级>的列表，其大小也为 K：

$$\text{Rec} = \bigcup_{m \in \text{Rec}} \{mID, E_{um}\}, \quad len(\text{Rec}) = K$$

接下来，将 updated_S 与本次为 u 实时推荐之前的上一次实时推荐结果 Rec 进行基于合并、替换形成新的推荐结果 NewRec：

$$New\ \text{Rec} = topK(i \in Rec \cup updatedList, cmp = E_{ui})$$

其中，i 表示 updated_S 与 Rec 的商品集合中的每个商品，topK 是一个函数，表示从 Rec updated _ S 中选择出最大的 K 个商品，cmp = E_{ui}表示 topK 函数将推荐优先级E_{ui}值最大的 K 个商品选出来。最终，NewRec 即为经过用户 u 对商品 p 评分后触发的实时推荐得到的最新推荐结果。

5.3 算法实现

https://github.com/liuming13/ECommerceRecommendSystem/blob/master/recommender/OnlineRecommender/src/main/scala/com/ming/online/OnlineRecommender.scala

6. 冷启动问题处理

整个推荐系统更多的是依赖于用于的偏好信息进行商品的推荐，那么就会存在一个问题，对于新注册的用户是没有任何偏好信息记录的，那这个时候推荐就会出现问题，导致没有任何推荐的项目出现。

处理这个问题一般是通过当用户首次登陆时，为用户提供交互式的窗口来获取用户对于物品的偏好，让用户勾选预设的兴趣标签。

当获取用户的偏好之后，就可以直接给出相应类型商品的推荐。

7. 结束语

近四个月的毕业设计已经结束，在这段时间里得到老师和同学的很大的帮助，首先我要感谢我的导师史润老师。无论从论文的写作与修改，还是相关操作技术的指导，史润老师都给予了我极大的帮助。这将对我以后的学习生活产生很大的帮助，让我不断前进、进步。谨以此表达我对史润老师最崇高的敬意和深深的感谢，她将对我的学习及工作产生很大的影响。

此外，作为一个“技术的使用者、代码的搬运工”，我一直对全球各大开源技术源码和资料的提供者保持尊敬、感谢的态度。本课题项目从头到尾使用的组件包括 Apache Spark、Apache Zookeeper、Scala、Java、Redis、MongoDB、CentOS-Linux、Maven 等等软件无一例外全部都是永久免费的。在此，我衷心地对 Apache 等开源贡献组织以及全球的开源项目贡献开发者们表示感谢！是他们推动了大数据的发展，推动了时代的进步。

最后，我还要感谢我的学校，在这里，我实现从一个懵懵懂懂的高中生到大学生的蜕变。我相信，我在这三年里学到的知识，将会是我一生最大的财富。

参考文献

[1]伍之昂,曹杰. 电子商务推荐系统导论[M]. 科学出版社

[2]陆俊尧,李玲娟.基于 Spark 的协同过滤算法并行化研究[J].计算机技术与发展

[3]周恒新. 基于 Hadoop 架构的商业推荐引擎协同过滤算法设计与实现[D]

[4]尚硅谷. 机器学习和推荐系统项目实战