华南理工大学

South China University of Technology

硕士学位论文

基于 storm 新闻推荐系统的研究与实现

作 者 姓 名	张慧
学 科 专 业	计算机应用技术
指 导 教 师	苏锦钿
所 在 学 院	计算机科学与工程学院
论文提交日期	2015 年 5 月

The Research and Implementation on Personalized News Recommendation Based on Storm

A Dissertation Submitted for the Degree of Master

Candidate：zhang hui

Supervisor：A.P. SuJinDian

South China University of Technology

Guangzhou, China

分类号：TP391 **学校代号：10561**

学　号：201220112805

华南理工大学硕士学位论文

基于 storm 新闻推荐系统的研究与实现

作者姓名：张慧　　指导教师姓名、职称：苏锦副教授

申请学位级别：工学硕士　　学科专业名称：计算机应用技术

研究方向：个性化推荐

论文提交日期：2015 年 6 月 1 日　　论文答辩日期：2015 年 6 月 5 日

学位授予单位：华南理工大学　　学位授予日期：　　年　　月　　日

答辩委员会成员：

主席：　许勇

委员：　闵华清、余志文、苏锦钿、马千里

华南理工大学

学位论文原创性声明

本人郑重声明：所呈交的论文是本人在导师的指导下独立进行研究所取得的研究成果。除了文中特别加以标注引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写的成果作品。对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式标明。本人完全意识到本声明的法律后果由本人承担。

作者签名：张慧　　日期：2015年 6月 1 日

学位论文版权使用授权书

本学位论文作者完全了解学校有关保留、使用学位论文的规定，即：研究生在校攻读学位期间论文工作的知识产权单位属华南理工大学。学校有权保存并向国家有关部门或机构送交论文的复印件和电子版，允许学位论文被查阅（除在保密期内的保密论文外）；学校可以公布学位论文的全部或部分内容，可以允许采用影印、缩印或其它复制手段保存、汇编学位论文。本人电子文档的内容和纸质论文的内容相一致。

本学位论文属于：

□保密，在_____年解密后适用本授权书。

☑不保密，同意在校园网上发布，供校内师生和与学校有共享协议的单位浏览；同意将本人学位论文提交中国学术期刊（光盘版）电子杂志社全文出版和编入CNKI《中国知识资源总库》，传播学位论文的全部或部分内容。

（请在以上相应方框内打“√”）

作者签名：张慧　　日期：2015.6.1

指导教师签名：[illegible]　　日期　2015.6.10

作者联系电话：　　电子邮箱：

联系地址（含邮编）：

摘要

目前随着互联网的进一步发展，以及移动设备近几年的崛起，2015 年全球网民有望突破 30 亿，已经将互联网推进大数据时代，同时互联网上的各种信息也极其膨胀和冗余，个性化推荐在大数据时代显得尤其重要。通过大数据的计算，个性化推荐能够智能地为用户推荐用户所感兴趣的内容，让人们从海量数据的迷茫中解脱出来。

目前新闻推荐系统推荐算法大多基于协同过滤、基于内容的推荐算法。但是又面临着协同过滤的缺点，数据的稀疏性问题和冷启动的缺陷。基于新闻的属性，本文在利用协同过滤算法的同时，结合 TF-IDF（词频-逆文档频率）算法在内容处理上优势，以及权重的设置，提出了一个新闻个性化推荐系统的设计思路与解决方案。

本文的主要工作包括：

1) 基于协同过滤推荐

除了将用户访问列表、新闻条目的访问列表作为相似度的度量外，还加入了新闻关键字，新闻类别作为相似度的计算之中，同时根据新闻发表时间，适当调整推荐项目推荐值，使推荐结果更加符合用户的浏览行为。

2) 基于内容推荐

新闻内容包括新闻类别、新闻分词后的关键字。在基于新闻类别的推荐中，用户浏览的各个类别的新闻数量与用户的兴趣是正相关的。在统计一个用户足够多的浏览记录之后，可以得出用户在每个新闻类别上浏览的新闻数量，这个数量可以认为是用户对每个新闻类别的兴趣值。

在基于新闻关键字的推荐中，通过统计用户浏览历史中的新闻中关键字，将这些新闻关键字聚合起来作为用户的关键字。在推荐的后台数据库，根据 TF-IDF 算法利用这些关键字为每条新闻评分，推荐集合按照评分的高低降序排列。

实验结果证明：将基于协同过滤算法的新闻推荐值与基于内容推荐算法的新闻推荐值相加，并作为每条新闻的最终推荐值的混合推荐方式能够有效提高推荐效果。为了加快推荐算法的速度，整个推荐过程是以 storm 为核心，将新闻库的信息、MapReduce 离线计算结果都保存在内存数据库中，storm 根据内存中的用户数据、新闻数据做实时推荐。

关键字：storm；TF-IDF；新闻推荐；个性化推荐；协同过滤

Abstract

With the further development of the Internet, and mobile devices are expected to exceed 3 billion global Internet users in 2015. Internet has been to promote the era of big data, while a variety of information on the Internet is extremely expansion and redundancy, personalized recommended is particularly important in the era of big data. By calculating big data, recommender system can send users content that user will be interested in intelligently, so that people free from the confused mass data, and improve efficiency

Current news recommendation system based on collaborative filtering and recommended algorithm based content. But it faces the shortcomings of collaborative filtering, the sparsity of data and cold start defects.

The main work of this paper is:

1) Based collaborative filtering recommendation

In addition to the list which user access, the access list of news items as an external measure of similarity, we also joined the news keyword and news category as the calculation of similarity. Besides, according to the news published in time, we optimize parameters to the recommended values of recommended projects, with the recommended results are more in line with the user's browsing behavior.

2) Based content recommendation

News content includes keywords and news category.

In the recommendation based on the news category, the number of news category, that user browses on, is positively related with how users interested in news of that category. After system analyzes each user's browsing history, and count the number of each news category user browses, this number can be considered the value of user interest for each news category.

In the recommendation based on the keywords, system calculates keyword vector for each user from history. Algorithm use the keyword vector as points that user is interested in. And system uses TF-IDF to recommend news set for users based on keyword vector. TF-IDF will calculates score for each item in recommendation set, and set news item in descending order according to score.

Experimental results show: the final score of news is the sum of the score calculated by CF and the score calculated by algorithm based content, this way to mixed two algorithm can improve the recommendation set effectively.

In order to spend up the computational process, system treats storm as core, and pushes

content of each news and result calculated by Mapreduce to redis database, storm just recommend the result saved in redis.

Keywords: storm; TF-IDF; news recommendation; Personalized Recommendation; collaborative filtering

目录

第一章 诸论

1.1 研究背景

在目前的互联网大数据时代，信息纷繁复杂，海量数据让人容易陷入选择困难，迷失在信息海洋中，在当下信息过载的背景下，个性化显得尤其重要，它能够帮助人们从大量无用信息中找到自己真正感兴趣的资讯。

个性化推荐是基于大数据的背景下，通过分析用户的海量数据，从中归纳出用户的行为模型、行为特征、个人喜好，机器学习人工智能方法动态跟踪用户的需求，及时提供最符合用户的推荐结果，著名的尿布和啤酒的关系就是机器学习算法的一个经典案例，它能给用户带来良好的用户体验，增加用户粘性，从而推动商业成功。

在大数据的背景下，用户需求和商业利益推动着技术发展，Google 发布关于 GFS[1]、MapReduce[2]、BigTable[3]的三篇论文后，Apache 随后根据这三篇论文开发了开源软件 Hadoop、HBase、HDFS。同时其它大数据工具 storm、spark、Nginx 是应势而生，成为大数据时代利器，人工智能、机器学习算法又重新崛起，并得到新的发展，行业的实际需求在技术研究和商业应用中架起一座桥梁，行业需求向技术研究抛出了新的问题，技术研究成果又推动行业进步，所以在过去几年大数据技术蓬勃发展，关于机器学习、人工智能的会议也是异常火热。

像在淘宝、亚马逊等大型电商而言，他们是推荐系统是重要需求者，也是推进者。普通大众不可能去查看每个商品再决定去哪些，或者单纯依靠用户关键字搜索去找商品，这都不能充分挖掘用户的消费需求。只有借助推荐系统，将用户可能感兴趣的商品主动推荐给用户，有针对性地刺激用户，才能挖掘用户的消费需求，亚马逊图书类的个性化推荐能为系统贡献 30%左右的销售额，个性化推荐的重要性可见一斑。

同样地在新闻系统方面，个性化也是极其重要，门户网站每天发布的娱乐、财经、军事、体育、科技等新闻数以万计，而每个人浏览新闻的时间有限，传统的关键字搜索只能提供与关键字直接相关的一些搜索结果，对于用户来说，它是一种被动式的，不够智能的，没有提供与关键字间接关联的结果。而个性化推荐能够利用用户的历史数据，挖掘用户的喜好和潜在的兴趣，主动向用户推荐挖掘结果，有效提高大家的时间利用率，节省个人的时间，还能够将用户感兴趣的内容主动推送过来，改善用户体验，增加用户粘性。

个性化推荐算法著名的有协同过滤、基于内容、信任模型、基于用户兴趣传播等算

法。推荐系统一般不会只用一个推荐算法，由于各个算法都有自身的优缺点和适合的场景，一个好的推荐系统应用利用各个算法去扬长避短，混合使用，以达到最佳的推荐效果。

1.2 国内外研究现状

1.2.1 个性化推荐系统的发展历史

推荐系统可以视为其它学科的一个延伸性学科，信息检索从广义上来说也是一种推荐，只不过通过关键字匹配的一种比较被动的推荐方式[4]。

1994 年，以明尼苏达大学推出的 GroupLens 系统为标志[5]，推荐系统才作为一个相对独立的研究领域。GroupLens 系统第一次提出协同过滤的推荐思想，同时为推荐系统建立了形式化的模型。这个系统为此后几十年的推荐系统提供一个重要方向——协同过滤。

此后陆续有学者提出了基于用户的协同过滤（也称基于邻居的协同过滤），2001 年的 Sarwr 教授提出了基于 item 的协同过滤[6]，2003 年 Amazon 发表论文讲解了基于 item 的协同过滤在 Amazon 的商业应用也取得了巨大的成功[7]，Amazon 发表的数据表明基于 item 的协同过滤算法为系统贡献了近 30%的营业额，推荐系统巨大的商业前景进一步推动其发展。

2000 年 Fu、Budizk、Hammond 等人提出了基于关联规则的推荐系统[8]。

后来还出现了基于内容的推荐、基于用户统计信息的推荐、基于效用的推荐、基于知识的推荐等一系统推荐技术。

在新闻个性化推荐方面，Google News 和 Yahoo! News 作为新闻聚合行业的两大巨头，她们的技术代表了行业最先进，在文章的后面将详细分析她们目前的个性化推荐技术。在 2007 年时，Google News 发表了一篇论文，基于协同过滤算法的推荐系统[9]。在 2010 年时，Google News 又发表了一篇关于个性化推荐的混合推荐系统，结合基于内容的推荐和基于协同过滤的推荐算法[10]。

1.2.2 研究简介

目前个性化推荐算法中协同过滤算法最受重视，算法在电子商务上的实践效果特别成功，算法细节会在后面章节讲到。Amazon 作为协同过滤的先驱，用成功的商业应用证明了协同过滤的有效性。协同过滤有两种算法，一种是基于 user 的协同过滤，另一种是基于 item 的协同过滤。在日常生活中，用户很容易接受身边朋友的推荐，在这种推荐

延伸到电子商务领域，用户也倾向于相似用户对自己的推荐，基于 user 的协同过滤推荐，首先计算出相似用户，之后将相似用户浏览或者购买的 item 推荐给当前用户。在电子商务领域，由于新闻的增加非常频繁，而商品本身的增加会较少，如果利用基于 user 的协同过滤推荐，会增加计算的复杂度，而基于 item 的协同过滤推荐能够很好地适用电子领域，它是计算 item 之间的相似性，统计每个 item 用户，这些用户可以组成一个向量，只要计算两个 item 之间用户向量的夹角，就可以判断它们是否是相似 item 了，最后将当前 item 的相似 item 作为推荐项。

但是协同过滤也有自身的优缺点。算法的优势是，随着用户数据积累的越来越多，它推荐效果越好，而且不用专业的领域知识辅助；劣势是，它存在冷启动、矩阵稀疏性缺点，目前业界对于协同过滤的研究也主要集中在弥补它的这两个缺陷，项目评估和基于降维预测的推荐算法可以克服和缓解这些缺陷，关于这方面的问题，文章后续也会做分析和讲解。

著名的啤酒和尿布的关系案例是基于关联规则推荐的经典案例，关联规则的一个通俗解释就是：通过用户的历史购买记录中，可以得出一些结论。比如，如果一个用户购买了 *item-A*，那么他/她很有可能还会购买 *item-B*。可能说 *item-A* 和 *item-B* 之间存在一定的关联。如果在一个实际场景中，一个新用户购买了 *item-A*，此时系统向他/她推荐 *item-B*，他/她很有可能会购买 *item-B*。

关联规则容易挖掘用户的新的兴趣点，并且不需要任何领域的专业知识。关联规则的缺点就是随着数据量的日益增加，关联规则的挖掘的复杂度也相应增大，而且关联规则会出现爆炸式增长[11]。

在新闻推荐方面，另一种重要算法就是基于内容的推荐算法，最经典的当属 TF-IDF 算法，它的大致思想是：如果一个单词出现的频率越高，且在其它文档中出现的频率越低，那么这个单词就越重要。算法细节后面章节会有简介。近些年，门户网站开始出现类似“猜你喜欢”模块，这些就是个性化的推荐。

由于两种推荐算法都有各自的缺点，协同过滤存在冷启动、数据稀疏性问题；基于内容的过滤算法存在不能发现新的兴趣点的缺陷。所以就出现了混合型的推荐算法，扬长补短，在没有用户历史数据时，先根据 profile 利用内容的推荐算法推荐数据，等有足够的数据时再发挥协同过滤的优势。这也是被广泛采纳的方案。

1.2.3 新闻推荐系统实例介绍

业界有两个新闻聚合网站巨头，一个是 Yahoo! News，另一个是 Google News。它们的推荐系统代表着新闻推荐方面的最高技术。

图 1-1 Google News

图 1-2 Yahoo News

Google News 是 Google 推荐一项新闻聚合服务，2002 年发布第一个 beta 版本，2006 年正式推出了官方正式版，刚开始是由 Krishna Bharat 推动的。它在世界上有超过 4500 个新闻源，60 个以上的国家和地区、28 种语言版本；它提供的新闻包括过去 30 天内的各个新闻源的新闻信息，它的用户已经超过一亿，Google News 发表的推荐方面的论文具有权威性[12]。

它发表两篇了关于新闻推荐算法的论文。第一篇是基于协同过滤算法而做的推荐，第二篇是基于内容和基于协同过滤的混合推荐算法。目前的新闻都是混合型的，因为每种算法都有自身的一些局限性，将它们合理地混合，能够扬长避短，提高推荐效果。

1.2.3.1 协同过滤的推荐推荐算法

2007 年，Google News 团队发表了《Google News Personalization：scalable online collaborative filtering》[9]，这篇论文主要讲解 Google News 在 2007 年以前使用协同过滤算法作为主要推荐算法，为 Google News 用户做个性化推荐。推荐的算法是混合了基于内存和基于模型的协同过滤推荐算法。

文中介绍了两种聚类技术：PLSI 和 MinHash 算法[13]。两个算法根据用户浏览的新闻各自为每条新闻得到一个分数。预测用户u_a对新闻条目n_k的评分：

$$r_{u_a,n_k} \propto \sum_{c_i:u_a \in c_i} w(u_a, c_i) \sum_{u_i:u_i \in c_i} I_{(u_j,n_k)} \tag{1-1}$$

$w_{(u_a,c_i)}$是用户u_a在用户群c_i的成员因子。

下面简单介绍下 MinHash 算法。

1.2.3.2 MinHash 算法

MinHash[13]是 Min-wise Independent Permutation Hashing 的缩写。普通计算用户相似性时，需要针对每一对用户做计算，这种计算的复杂度是$O(n^2)$。这对于大规模的用

户量，这种计算复杂度是不可接受的，为了提高效率，出现了一些改进的算法，MinHash就是其中之一。

在讲解 MinHash 算法之间，先介绍下 Jaccard 相似度[14]

$$Jaccard(X,Y) = \frac{|X \cap Y|}{|X \cup Y|} \tag{1-2}$$

Jaccard 相似度的值落在[0,1]的区间里。它的基本思想是随机交换用户浏览过的新闻列表中的第一项，再计算第一个条目的 hash 值，如果相等则认为这两个用户相似。这种做法是有效性可以从 Jaccard 相似度的计算公式中可以看出。如果两个用户的 Jaccard 相似性越高，那么他们的条目随机排列后，第一个元素相同的可能性就越高。用公式表示就是：

$$p\big(MinHash(s_1) = MinHash(s_2)\big) = Jaccard(s_1, s_2) \tag{1-3}$$

1.2.3.3 基于用户点击行为的个性化新闻推荐系统

2010 年，Google News 团队发表了《Personalized News Recommendation Based on Click Behavior》[10]，针对用户点击行为，分析用户的兴趣变化，结合协同过滤算法，提出了一种基于协同过滤和基于内容的混合推荐算法。

文章是基于内容的推荐算法是通过分析用户的点击行为，为用户生成个人 profile，针对每个用户的 profile 文件做个性化推荐。Profile 文件的准确性决定了一个推荐系统是否成功。

当用户浏览一个新闻页面时，用户实际上是在寻找新的信息，因此检测用户的兴趣变化对于预测用户的行为至关重要。通过分析知道用户的兴趣会随着时间而变化，但是这种变化会新闻重要事件的出现相一致，Google News 设计了一个贝叶斯模型预测每个用户的兴趣变化情况。

文章的主要有三方面的贡献

1) 针对大规模用户日志，分析出用户的新闻兴趣点。

2) 利用用户的点击行为，提出一种推荐方法，既考虑了用户的个性化兴趣，也考虑了当前的新闻热点。

3) 提出一种基于内容和基于协同过滤的混合推荐算法，

用户的新闻兴趣可以概括为两类：一类是长期的，另一类是短期的[15]。短期的新闻兴趣点一种是与当前的新闻热点有关，一般同一区域的用户的短期兴趣点是一致的，新闻这种短期的兴趣点转移也非常快；长期的新闻兴趣点能够反映一个用户真实的个人兴

趣，可能会与用户的性别、年龄、专业相关，这种兴趣是相对稳定的，不会轻易发生改变。

用户短期兴趣的预测是通过统计同一区域用户的浏览趋势而得到的，而长期兴趣的预测是利用贝叶斯模型[16]计算而来的。

论文提出的贝叶斯模型如下：

用户过去一段时间 t 内的点击记录，可以计算出用户在时间 t 内对每种新闻类别的点击概率：

$$\begin{aligned} p^t(click \vee category = c_i) &= interest^t(click|category = c_i) \\ &= p^t(click|category = c_i) \\ &= \frac{p^t(category = c_i|click) * p^t(click)}{p^t(category = c_i)} \end{aligned} \tag{1-4}$$

其中$p^t(category = c_i)$是用户浏览过新闻是c_i类别的先验概率；$p^t(click)$是无论新闻类型是什么，用户点击文章的先验概率。

用户对在时间 t 内对c_i类别新闻的权重值

$$\begin{aligned} interest(category = c_i) &= \frac{\sum_t \left(N^t * interest^t(category = c_i)\right)}{\sum_t N^t} \\ &= \frac{\sum_t \left(\frac{N^t * p^t(category = c_i|click) * p^t(click)}{p^t(category = c_i)}\right)}{\sum_t N^t} \end{aligned} \tag{1-5}$$

N^t是在时间 t 内，用户点击新闻的总量。通过这种计算方法，可以预测用户对每种新闻类别的喜好程度。

为了生成最后的推荐列表，基于内容的推荐分数用 *CR(article)*表示，*CR(article)*是基于新闻类别的一个分数，基于协同过滤的分数用 *CF(article)*表示，最后的混合的分数 *Rec(article)=CR(article) * CF(article)*。

在 Google 论文的实验部分，它是用 CTR 为评测参数，CTR 是 click-through rates 的缩写，意思是用户的点击率。如果个性化推荐区域，用户的点击率高于其它普通模块，则表明个性化的推荐是有效的，用户的点击率越高，则表示推荐效果越好。

Google News 作为新闻聚合的业界巨头，它拥有庞大的用户群、海量的用户数据，它的算法很有代表性，是新闻推荐的技术的先进代表。正是基于对 Google News 的技术的了解。本文提出的系统，融入了关键字，以期能从新闻内容入手，结合协同过滤，提出一种新闻推荐系统。

1.3 本文的研究内容

1.3.1 初级

本文研究的内容主要是组合型的推荐算法。个性化推荐结果包括两个方面：一方面是协同过滤的推荐结果；另一方面是基于关键字和 TF-IDF 算法，新闻类别的结果。

1）协同过滤推荐。

基于用户的协同过滤，在用户历史记录中，通过计算相似度，可以找出一个用户的最邻近用户。之后在给用户做推荐时，将邻近用户的 Top K 浏览项的作为推荐结果。

基于条目的协同过滤，在新闻库中，实时计算任意两条新闻之间的相似度，从而找出每条新闻的邻近新闻，当一个用户查看了 *news-A*，系统则将 *news-A* 的邻近新闻推荐给该用户。

2）基于内容的推荐

在系统初始时，可以通过引导用户填写 profile，系统从 profile 中提取特征，就可以根据这些特征作基于内容的个性化推荐。之后随着用户历史数据的积累，当有足够的历史数据时，系统可以从收集的历史记录中提取用户关键字。

新闻数据包括两部分：一部分是用户的历史记录，另一部分是当前的新闻库（过去十天内的新闻汇总）。

在用户的历史记录中，利用分词器将新闻标题分词，去掉形容词、动词、停用词，将其中的名词当作用户关键字 key。每个关键字 key 的频率就是关键字 key 的权重。可以得到一个向量

$$w = \{w_1, w_2, \dots w_j, \dots w_{n-1}, w_n\} \tag{1-6}$$

其中w_j表示第 j 个关键字的权重。

为新闻库中的新闻分词，记录分词结果，并建立倒排索引。将新闻库当作一个搜索引擎，再把上一步中的关键字 *key* 作为搜索引擎的关键字搜索新闻库，每个新闻会有一个权值；每个 *key* 搜索之后的结果会组成一个 *N*M* 的二维矩阵 *NW*。（假设 *M* 是新闻的数量，*N* 是关键字的数量）

将向量 *KW* 与矩阵 *NW* 做乘法，最后的结果就是一个 *M* 维的向量，向量的每个值代表每个新闻的推荐值。

1.3.2 改进

上面提到的是实际是简单的 TF-IDF 的权重算法，此外还有些改进的地方。

新闻作为一种特殊的文档，具有自身的特性——时效性。用户的兴趣会变化，一年前的热点已经还是当前的热点；同时会不断出现新的新闻热点，用户也会有新的兴趣点，一个好的推荐系统应该能跟上用户的需求，不断发现、挖掘、迎合用户的新的需求。

一般而言，用户的兴趣随着时间而改变，用户历史数据中的热点，会有衰老的问题。对应地来说，越老的数据，它的权重应该越小，为此本文的关键字权重加入了一个衰老因子。

通过对数据集进行分析，发现一篇新闻的活跃期是其发布后三天，之后还有 7 天左右的平淡其，十天之后的用户会大减少。基于此，在上面的混合推荐结果中，根据新闻的发表时间，适当增减新闻的推荐值。

1.4 本文的结构安排

第一章绪论主要提出研究背景，基于互联网的大数据时代，信息冗余和信息过载问题，传统的互联网方式遇到了瓶颈，个性化推荐作为利器极大地提高人们的信息获取效率，机器学习和人工智能近几年又一次兴起，并取得了一定的成果，特别是在电子商务领域。

第二章个性化推荐算法主要介绍目前主流的推荐算法，包括协同过滤算法、基于内容的推荐算、混合型推荐算法；并简述了各个算法的优点各缺点，以及目前的解决方案。

第三章主要讲述了基于 storm 新闻推荐系统中的算法，以及这些算法如何设计与实现的。

第四章简述了利用第三章的算法，如何搭建一个新闻推荐系统，推荐系统中的算法实现过程与细节。

第五章是实验验证及分析，主要是利用上述提到的算法，在公开的数据集上进行测试，并对测试结果做说明。

第六章是本文的总结与展望，对本文进行总结，并分析算法的一些不足之处，以及提出后期的一些改进思路。

第二章 新闻推荐的相关技术

个性化推荐在大数据背景下是如此的重要，推荐算法过去十年左右的作为一个重大研究方向，个性化推荐系统初期提出的部分算法已经不合时宜了，在目前的大数据的环境下，一些推荐的相关算法暴露自身的局限性。例如基于规则推荐算法随着数据的增加，会出现组合爆炸的问题；K-Means 在聚类时，需要多次计算距离，导致在分布式环境下效率低下，之前关于 K-Means 的并行化也是个研究热点；协同过滤算法，在电子商务平台上，由于商品和用户的数量越来越多，协同过滤算法中的矩阵的稀疏性更加明显。

本章着重介绍主流的推荐算法，包括两种协同过滤算法、基于内容推荐、基于规则的推荐、基于信任模型的推荐、二部图推荐以及混合型推荐，并分析它们的优缺点，介绍目前的一些改进方案。

2.1 开源工具简介

1) Apache Storm[17]：是一个开源的分布式的实时计算系统，它的目的是帮助人们处理数据流，提供实时的批处理。Storm 适用于实时分析、在线机器学习、连续计算、分布式 RPC、ETL。它具有可扩展性、容错性、保证数据会被处理、配置简单、操作容易。目前阿里巴巴、百度、雅虎、twitter 等一大批知名公司都在用 storm。

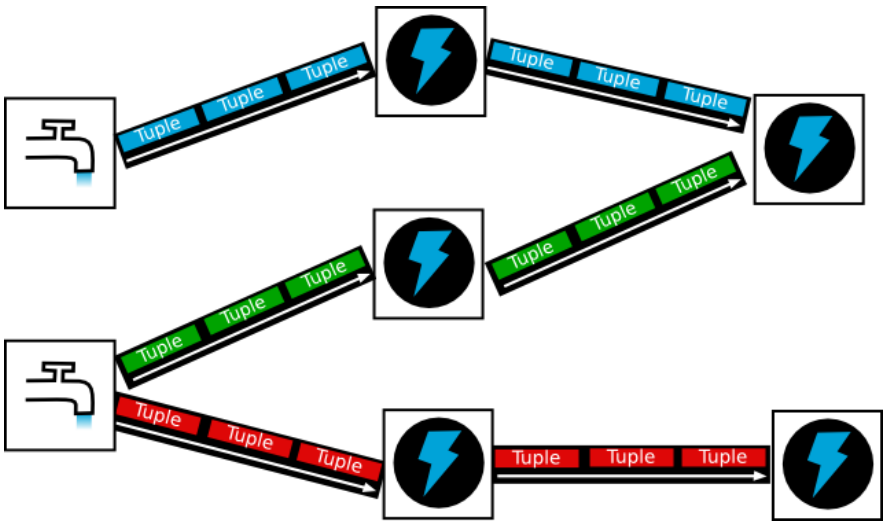

图 2-1 storm 数据处理流程

图 2-1 是 storm 的逻辑图，storm 上的运行单元是 topology，每个 topology 包括两个重要组成部分：Spout 和 Bolt。图中左边的两个水龙头是 Spout，代表数据流入，所有的数据通过 Spout 流入 Storm 集群，在 Spout 负责将数据解析，之后利用 nextTuple 函数将数据封装成一个个 tuple 交给下一个 Bolt 做处理。Spout 会声明 tuple 中的字段；图中

闪电标识的节点代表 Bolt，是数据的逻辑处理单元。

为了提高推荐系统的响应速度，论文设计的系统以 storm 为枢纽，连接各个模块，以将推荐的延时降到最低，实时计算每个新闻条目的相似度，统计四大模块的推荐值，赋予权重，从而得出最终的推荐列表。

2) Hadoop[18]：作为目前业界最重要的大数据处理利器，众所周知，它是 2004 年 Google 发表的 MapReduce 论文的开源实现，2006 年 Apache 基金会正式成立 Hadoop 项目，现被广泛使用，已经形成一个丰富繁荣的 Hadoop 技术生态圈。它的优势是利用大量廉价机器组成一个强大的分布式系统，每个 job 分为 Map 阶段和 Reduce 阶段，充分利用每台机器的资源，组成一个高可靠性、高效、可扩展的分布式系统。Hadoop 也有自身的文件系统 HDFS，作为分布式的存储系统，可以将充分利用分布式的优点，大大增加本身的存储大小。

论文的系统利用 hadoop 的 MapReduce 做离线计算，统计用户浏览新闻的关键字，为后续的新闻推荐的关键字模块打基础；统计用户浏览的新闻类别，计算用户的兴趣分布权重；计算相似用户，为基于用户的协同过滤推荐做准备。

3) HBase[19]也是 Apache 下的顶级开源项目，是 Google 的 BigTable 论文的开源实现，它和 Google BigTable[3]相对应，BigTable 以 GFS 为底层文件存储系统，HBase 以 HDFS(Hadoop File System)为底层存储系统。作为 NoSQL 数据库，（1）具有线性扩展，数据是放在一个个 Region 中，当 Region 达到一定的容量时，会自动分割；（2）对持续高速读写有较好的性能，通过合适的设计，可以将读写的数据分配到不同机器的 Region，这样可以大幅提高读写速度；（3）由于底层文件系统是 HDFS，其自身有自动备份和容错功能，另外还有 HDFS 的 log，进一步保证数据的安全性。

在本文设计的系统中，HBase 作为新闻数据的存储容器，所有抓取的新闻数据都放在 HBase 数据表中，包括 URL、标题、新闻正文、新闻关键字、新闻描述、新闻的分词结果。在用户关键字统计模块中，根据用户浏览过的新闻，结合新闻分词结果，以及新闻页面关键字等，利用 MapReduce 程序完成每个用户关键字的统计。

4) Redis[20]是一种开源的、内存型数据库。之前 Redis 由 VMware 所支持，从 2013 年开始，Pivotal Software 开始赞助其发展。根据 DB-Engines.com 的月度排名，Redis 是最受欢迎的 key-value 数据库。Redis 同时也支持众多的编程语言：C、C++、C#、Java、Objective-C。Redis 与其它结构存储系统数据库的关键不同就是：它不仅支持 string（字符串），也支持抽象的数据结构，包括 List of String、Sets of Strings、哈希表。

Redis 是将数据放在内存之中，它还具有持久性，因为它的数据内容会同步到硬盘；它还支持 master-slave 的复制方式，Redis 主服务器的数据可以复制给任意数据的从服务器；Redis 将内存作为数据容器，它的读写速度远远高于利用磁盘的数据库；同时 Redis 支持集群部署，可以打破一台机器内存的限制，进一步扩大数据规模。

在论文所设计的系统中，Redis 存储了推荐过程的中间结果值，包括用户的关键字列表和关键字权重、新闻类别权重、最新的新闻库信息、新闻库分词结果、新闻内容中的关键字倒排索引等，这些数据量不大，且计算过程中会经常用到，是 Redis 数据库的典型应用场景，这些数据作为 Storm 数据流，可以更大程度发挥 storm 的数据处理速度。

2.2 文档特征向量处理

2.2.1 TF-IDF 算法

1957 年，Hans Peter Luhn 第一次提出“词频体现了一个词的重要性”[21]；1972 年，Karen Spärck Jones 第一次提出“逆文档频的概念”[22]；TF-IDF 算法正是这两种思想的有机结合。之后 TF-IDF 算法[23]在文档分析、搜索引擎等领域广泛应用。

tf(*t*,*d*)表示词 *t* 在文档 *d* 中的 *tf* 值，*f* (*t*,*d*)表示词 *t* 在文档 *d* 中出现的次数。*tf* 根据情况不同，有几种计算方式

1) *tf* (*t*,*d*)取值为 0、1 的，0 表示没有在文档 *d* 中出现，1 表示出现了。
2) *tf* (*t*,*d*)=1 + *log*(*f*(*t*,*d*))，如果 *f*(*t*,*d*)是 0，则 *tf* 也为 0。
3) $tf(t,d) = 0.5 + \frac{0.5*f(t,d)}{max\{f(w,d):w\in d\}}$ 第二种计算方式会偏向于长文档，通过除以最大词频可以减少长文档的在算法当中的优势。

TF-IDF 算法在信息检索方面的应用十分成功，各个搜索引擎都是基于此算法而拓展的。

但是对于本文的对象“新闻“而言，原始的 TF-IDF 算法就有一定的缺陷。新闻推荐有两个重要特性，与传统搜索引擎区别开来。

- 新闻具有时效性。系统仅根据关键字推荐一篇几年前的新闻，这是不合理的。
- 用户的关注点会随时间而衰退。新闻有热度，几年前的热点，现在可能已经过时了，用户也不再关注了。

基于此，在做新闻推荐时，需要考虑时效性问题，最近的新闻会有高的优先级，而旧的新闻相应的降低它的优先级。

经过对数据集的分析，发现一篇新闻发表之后有三天的活跃期，三天之后，浏览量

会保持在一个相似较低的水平，十天之后 TF-IDF 算法中，每个关键字的权重是词频与逆文档的乘积。在一个文本中，如果有一个新出现的词语，它的 IDF 会很高，这会导致即使这个词不重要，它的 TF-IDF 值也会偏高。

2.2.2 信息熵

而信息熵[24]可以解决这种问题，熵本是信息论中一个术语，是香农提出的一个词。它是指在一条文本中包括的消息的平均数量，熵的值代表一个信息载体（关键字）的不不确定性。如果一个信息载体的越低，则表示它能表达的信息越少；如果一个信息载体的信息熵值越高，则代表它能表达的信息越多。在实际文档分析过程中，如果一个 word 的信息熵很高，则越能说明其可能是一个重要的关键字。如果 *WX* 这两个 word 经常一起出现，则它们可能是一个由 *XW* 两个 word 合成的关键字。

它的计算很简单，公式如下：

$$H(w) = -\sum p * \log(p) \tag{2-1}$$

其中 w 是指某个关键字 w，p 是指关键字 w 左右出现的不同词的概率。

举个例子，在一个文档中，w 的出现有两种情况： 两次是 *XWY*，三次是 *AWB*，所以 w 左侧的信息熵等于右侧的信息熵，其值为

$$\mathrm{H}(w) = \frac{-3}{5} * log\left(\frac{3}{5}\right) - \frac{2}{5} * log\left(\frac{2}{5}\right)$$

信息熵有如下几个特点：

1) Maximum(最大化)

由上面的计算公式可知，如果每个 p 的值都相等，则信息熵的值会达到最大值。

2) Additivity(加法)

无论文本被分为多少个模块，它的信息熵的值是可以直接相加的。

3) Symmetry(对称性)

信息熵的值与词的位置无关，无论词序如何。对于一段文本，不管如何排列，最终信息熵的值量一样的。

4) Continuity(连续性)

信息熵的值是连续的。如果概率值发生轻微变化，信息熵的值变动也会很少。

2.2.3 小结

本小节的内容主要是在介绍文本处理的相关算法：TF-IDF、信息熵。

这两个算法在新闻推荐中发挥着重要作用，关键字是基于关键字内容推荐算法的基础，关键字的分析好坏会大大影响后续的推荐效果。

信息熵与 TD-IDF 算法相比，有其优势，它能够避免新词的出现对 TF-IDF 算法的干扰，适合新闻文档关键字的提取。

2.3 协同过滤

协同过滤是一种预测用户对某个 item 的喜好程度的手段，所谓“协同”是指借助大家的帮助来实现过滤，从而找出你可能会喜欢的 item。

协同过滤算法的历史得追溯到 1992 年，施乐公司的一个研究院的员工每天会接收到许多的电子邮件，导致员工的邮件处理效率很低，于是就设计了第一个协同过滤系统，每个员工设置关键字，系统就从邮件中选出三封最相关的邮件推荐给员工。

协同过滤里程碑事件是 GroupLens 的新闻筛选系统，每个看完新闻的人给新闻一个评分，系统记录每个人的评分结果，系统找出与评分高的新闻相似的新闻推荐给用户，这就是基于 user 协同过滤的雏形。

真正将协同过滤推广的是亚马逊公司，应用于图书推荐，并取得了巨大的商业成功，为亚马逊贡献了 30%左右的销售额。目前协同过滤分为基于 user 的协同推荐和基于 item 的协同推荐，一般来说，数据量越大，协同过滤算法的准确度就越高，所以当下大数据的舞台，正是协同过滤算法的用武之地。

2.3.1 基于模型的协同过滤推荐

与基于内存的协同过滤推荐相对应，基于模型的协同过滤利用用户历史评分数据建立模型，用模型预测用户对未知新闻的评分，最早应用模型协同过滤的方法是概率模型：聚类模型、贝叶斯模型。后来也融入了各种机器学习方法、关联规则。对于模型推荐算法，大多数做法是捕获用户的兴趣，将用户分为多个用户群。Google News 团队发表第二篇论文就是利用贝叶斯模型预测用户的兴趣，从而做推荐。

1) 聚类模型推荐[25]

在原始的协同过滤推荐时，是针对每个用户作推荐，复杂度是$O(n^2)$，n 是用户数量。随着系统用户数量的增加，会大大增加系统的计算代价。

而聚类模型的推荐是将用户划分为若干个组，组内用户的相似度都达到一定的阈值，推荐时是以用户组为单位做推荐的。这种推荐方式虽然会损失一定的准确率，但是它大大减少了算法的复杂度。而算法复杂度可以通过其它方式弥补。

2) 贝叶斯模型推荐

在贝叶斯模型中，用户的兴趣分为几个类，通过统计用户过去的浏览记录，提出其中的特征，再基于特征分析用户对这些分类的先验概率。在做推荐时，根据先验概率，预测用户的喜好程度。

3) 关联规则模型推荐[26]

这个比较好理解，从用户的大量浏览记录中，分析那些新闻之间有关联，只是将“啤酒”与“尿布”的例子[27]应用于新闻上来而已，现在的一些门户网站会将关联的新闻人为地作为一个专题呈现给用户，而关联规则的推荐就是希望用系统自动达到这种效果。

2.3.2 基于内存的协同过滤推荐

基于内存的协同过滤推荐算法，根据系统用户的历史评分记录，预测用户对 item 的评分，一般，这种预测是通过相似用户评分加权计算得到的，而相似用户用皮尔森系数或者 cosine 计算来的，用户 a 对新闻 k 的评分预测公式

$$r_{u_a, n_k} = \sum_{i \neq a} I_{(u_a,\ n_i)} * w(u_a, u_i) \tag{2-2}$$

$w(u_a, u_i)$是指用户之间的相似性，如果用户 a 点击了新闻 i，则$I_{(u_a,\ n_i)}$等于 1，否则为 0。r_{u_a, n_k}也可以是一个二元值，如果大于某个阈值其值为 1，否则为 0。

基于内存的应用得越来越广泛，因为它思想简单，效果直接，目前它的最大的问题是如何将其应用大规模数据计算上，正如前面提到的那样，随着用户数据 的增加，计算的复杂度会急剧增加，因为好需要在每两两用户之间计算相似性。

基于内存的协同过滤算法分为两类：

1) 基于 user 的协同过滤，在用户的历史记录中，先将用户划分为 N 个群，然后利用 KNN（K 近邻）算法[28]找出 K 个相似用户。在推荐时，将相似用户浏览的新闻作为推荐项。
2) 基于 item 的协同过滤，在新闻库中，计算出相似新闻，当用户浏览了一条新闻，则将这条新闻的相似条目作为推荐项。

2.3.2.1 基于 user 的协同推荐—user CF

前提是相似用户的对同一个 item 的评分应该也是相似的。

举个简单例子：假设现在要为 user-Z 预测对 item-K 的评分，user-A、user-B、user-C 和 user-Z 是最相似的用户，那么只要将 user-A、user-B、user-C 对 item-K 的评分的平均值作为 user-Z 对 item-K 的评分。

从例子中，可以看出算法的两个关键步骤：

1) 找出 user-Z 的相似用户

2) 根据相似用户计算 user-Z 对 item 的评分。

基于 user 的协同过滤也被称为基于邻居的协同过滤，它的推荐都是基于相似用户得到的。相似用户的计算方法，目前大致有三种

- 欧几里德距离

  在欧几里德空间里，两个向量的距离[29]定义为：

$$distance(X,Y)=\sqrt{(x_1-y_1)^2+\cdots\ldots+(x_n-y_n)^2} \tag{2-3}$$

- Cosine 相似度[30]

  几何学的 cosine，计算两个向量的夹角，如果夹角越小，则意味两个向量越相似。设向量 X，向量 Y 的长度分别是 $len(X)$、$len(Y)$；Cosine 计算公式：

$$cosine<X,Y>=\frac{X*Y}{len(X)*len(Y)} \tag{2-4}$$

  其中 $X*Y$ 为向量 X 与向量 Y 的内积。

- 皮尔森系数[31]

  皮尔森系数的值是两个变量的协方差和标准差的相除的结果。

$$\rho_{X,Y}=\frac{cov(X,Y)}{\sigma_x*\ \sigma_y} \tag{2-5}$$

- Jaccard 系数[14]

$$Jaccard(X,Y)=\frac{X\cap Y}{X\cup Y} \tag{2-6}$$

本文中用户相似度正是通过这种方式计算的，它适用于用户评分二元值的情况。

找出与目标用户最相似的 Top K 用户，最经典的一个算法就是 K-NN（K Nearest Neighborhood），它的思想很简单明了。如图 2-2 所示，

为了找出中间绿色圆点的最邻近的 3 个邻居，只需要以自身为圆心画一个圆，使这个圆恰好圈出 3 个点即可。这个距离计算方法根据场景而定，基于用户的协同过滤算法的计算就是 cosine 相似度。

找出相似用户后，就可以根据相似用户的评分预估用户的评分。最简单的方法就是 top3 相似用户评分的平均值；也可以将相似度作为权重，乘以相似用户的评分，得出用户的评分估值。

由于基于 user 的协同过滤算法原理，决定了它自身的缺陷，为了找出相似用户，需

要两两计算用户之前的相似度，然后再找出最相似的几个用户，它的算法复杂度随着用户数量的增加而增加，同时也随着 item 的增加而增加，它同时存在数据从而决定了 user 协同过滤不同适用于大规模数据。

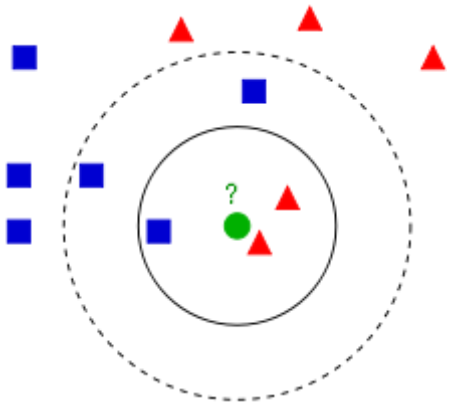

图 2-2　KNN 算法图例

2.3.2.2 基于 item 的协同推荐—item CF

基于 user 的协同过滤算法[32]存在着自身的缺点，随着数据规模增大，相似用户的计算复杂度也会线性增加，2001 年 Sarwr 教授提出了基于 item 的协同过滤算法，2003 年亚马逊发布的著名的论文——Amazon.com recommendations: item-to-item collaborative filtering，意味着基于 item 的协同过滤算法在电子商务领域的实践取得了成功。

基于 item 协同过滤算法，是通过计算两个 item 之间的相似性，从而向用户推荐相似 item 即可。

item CF 的向量是购买过 item 的 user 对 item 的评分；user CF 是 user 对所有购买过 item 的评分。两个向量的不同，导致两个算法的复杂度相差甚远。user CF 需要在所有用户中两两进行计算；item CF 只需要针对 user 已经购买过条目，找出它们相似的条目。对比可知，item CF 的计算量远小于 user CF。

item CF 的相似性计算与 user CF 一样，一般都是选择 cosine 相似性计算方法。

2.3.3 协同过滤概述

协同过滤算法随着用户数据的积累，算法的准确度将会越来越高，它的效果在实际应用中也得到验证，算法也不需要专家介入和领域知识，对新的 item 也能及时响应。

但是它也有自身的局限性，存在稀疏性问题、冷启动问题。目前协同过滤的研究也主要集中这两个问题。

稀疏性问题是指在一个系统中，一个用户购买或者评分的 item 是非常少，大部分的 item 用户是没有评价信息的，这样造成用户评价矩阵非常稀疏。冷启动是指在系统初期，

没有任何评分数据或者评分数据太少，不能保证算法的准确性。针对协同过滤算法的缺点，目前的方法主要有：

- 简单填值法：可以使用评分$(Max + Min) / 2$，也可以是用户所有评分的均值。
- 相似性传递：假设 user-A 和 user-B 相似，user-B 和 user-C 相似。如果 user-A 的评分不能通过 user-B 得到，那么就通过 user-C 取得。
- 增加用户-系统的交互：在用户第一次进入系统时，提供一些交互页面，让用户手动输入部分个人评分。
- 基于聚类的方法：聚类算法将用户分为一个个小群，小群的均值作为用户的值。
- 降维：将 user-item 的评分矩阵降维，主流的降维的方法有矩阵分解、主成分分析。

2.4 基于内容的推荐算法

另一种重要的推荐算法就是基于内容推荐[33]，协同过滤算法只是单纯地利用用户的评分信息，没有考虑 item 本身的一些属性，以电影推荐为例，一个人对电影的评分很大程度是由导演、主演、编剧、发行公司等因素决定的，协同过滤没有考虑这些，它没有充分挖掘条目本身的特征。基于内容的推荐正是基于此，充分利用并挖掘条目的性质和特征，以获得更好的推荐效果。

基于内容的推荐大致包括三个部分，1)所有的 item 的特征用一组向量表示；2)将用户的 profile（个人喜好）也转化为特征；3)根据 profile 的特征与 item 的特征，利用推荐算法给出推荐结果。

2.4.1 item 特征提取

特征提取的目的就是将 item 规范化、量化，便于计算。领域不同，特征提取的方式也不尽相同。

在文档的推荐系统中，最经典的就是 TF-IDF 算法，它将一个一个关键字当作文档的特征，每个词语的权重通过 TF-IDF 算法计算而来。Syskill、Weber 两个系统都是用 128 个关键字表示一个文档的特征；Fab 网页推荐系统采用 100 个关键字表示一个网页内容。

在电影推荐系统当中，item 就是一部电影，它的特征一般就是电影名、导演、男女主演、发行公司、编剧等。

在电子商务领域，item 特征可以是浏览、收藏、加入购物车该 item 的用户，以及

用户对 item 的评分。

2.4.2 Profile 特征提取

Profile 包括用户过去评价过的 item，用户手动输入的喜好信息。根据这些信息生成一个 model，结合机器学习算法就可以实现推荐。

在电影推荐系统中，假设在 user-A 的 profile 中，可能提取出 user-A 喜欢的特征有：演员周星驰、导演冯小刚、动作片、喜剧片。系统就可以利用算法去电影库中找出符合这些特征的电影，将它们作为推荐结果。

在文档推荐中，特征一般就是文档的词语，在一个文档中，最能体现文章内容的就是文档的名词，包括人名、地名、事物名、事件名。一些突发事件中，一个名词就大致代表文章的类别，例如在之前有“马航 370”事件中，分词后的名词“马航 370”就能代表文章的关键点。一个关键字的权重一般是通过 TF-IDF 算法计算而来，TF 是指关键字出现的次数，IDF 是关键字在整个文档集出现次数的倒数，意思是如果某个关键字在其他文档中出现的次数越多，则表示这个关键字的区分度越小。

2.4.3 基于内容过滤算法的优缺点

与协同过滤相比，优点

- 不需要历史记录
- 没有稀疏性问题：这个优点是相对协同过滤算法而言，在用户没有大量评价数据时，也能保证推荐效果。
- 不需要领域知识[34]

不需要特定的领域知识，只要针对特征即可。

当然，内容过滤算法也有自身的缺点，它的关键是特征提取、特征匹配，它的缺点也正在于此。

1)特征的提取决定了算法的质量，在文本方面表现出色，但是在音频和视频方面的特征提取有难度。

2)不能及时反应用户新的喜好，由于它的特征是从历史记录中获取的，很久之前的一些特征可能已经不重要，但是算法没有区分，本篇论文正是针对这个缺点，做了改进；

3)在系统出现新用户时，与协同过滤算法一样，会遇到冷启动问题，只能通过引导用户手输入。

2.5 混合推荐

上面提到的算法都有各自的优点各缺点，在实际系统中，一般不会只用一种推荐算法，而是将两个算法做一个混合。总的来说不外乎两种情况

- 在协同过滤中融入基于内容的推荐：针对协同过滤的缺点，当用户已经有 profile 的情况下，利用基于内容的推荐来缓解协同过滤的冷启动问题和稀疏性问题。
- 在基于内容中融入协同过滤的推荐：在一些不能很好地提取特征的 item 项，利用协同过滤来弥补基于内容推荐的缺陷。

当然，除了上面提到的推荐算法外，目前还有基于模型的协同过滤、基于网络结构（二分图）等推荐技术。在介绍推荐后，另一个重要主题就是推荐系统的评价标准。

以 Google News 的为例，它在初期是以协同过滤推荐为主。后来加入了基于内容的推荐方式，它算法中的内容是指新闻类别。

2.6 推荐系统的评价标准

推荐技术业界已经有好几种，只有通过评价标准来衡量一个推荐技术的好坏，评价标准里有几个重要概念：召回率(Recall)、准确率(Precision)、MAE，F 值。

召回率和准确率是信息检索系统中的重要评价指标。

召回率(Recall) = 系统找出的相关项 / 数据集中所有相关项。

准确率(Precision) = 系统找出的相关项 / 系统找出的项目集合。

但是召回率各准确率之前又存在着矛盾，一个好的检索系统应该同时兼顾召回率各准确率。即使一个系统的准确率很低，但是如果它返回的结果项足够多时，它也能达到百分之百的召回率；同理，即使一个系统的召回率很低，但是如果将它的返回项设为很少，它的准确率可能会很高。

在一些预测评分的推荐系统中，另一种业界普遍采用的评价方法是 MAE(Mean Absolute Error)，实际值和预测值的平均绝对误差。计算公式

$$\mathrm{MAE} = \sum_{i=1}^{N} |P_i - C_i| / N \tag{2-7}$$

其中 P 代表预测值，C 代表实际值，N 代表数据总量。

除此之外，在统计分析中，还有一种重要的评价标准，就是 F 值（F-Measure）,它既考虑了准确率，又考虑了召回率。它的计算公式如下：

$$F = \frac{1}{(\frac{1}{Recall} + \frac{1}{Precision})} = 2 * \frac{Precision * Recall}{Precision + Recall} \quad (2\text{-}8)$$

上式中，F 值的计算中，召回率和准确率的权重是一样的，一种更通用的计算是对召回率和准确率之间加上一个权重设置。

$$F = (1 + \beta^2) * \frac{precision * recall}{\beta^2 * precision + recall} \quad (2\text{-}9)$$

2.7 本章小结

本章节简介了新闻推荐系统中的相关算法，从目前推荐系统中主流的开源工具，再到新闻文本关键字处理，之后对介绍了 Google News 的新闻推荐系统。

开源工具是新闻推荐系统中的重要组成部分，利用这些数据存储、数据分析、实时推荐等开源工具，可以大大减少系统开发难度和开发周期。

TF-IDF 算法、信息熵两个关键字相关算法，是关键字推荐的基石。

基于协同过滤推荐和基于内容推荐是新闻推荐系统的核心，最终的推荐列表是混合了这两种推荐算法的结果。

第三章 基于 storm 新闻推荐系统的算法

本章主要介绍系统中的推荐算法，系统的推荐算法是以 storm 为核心，根据用户浏览新闻的行为特征，定制推荐算法。整个系统的推荐算法分为三大模块：基于协同过滤推荐算法、基于内容推荐算法、两种算法结果的混合。

3.1 新闻推荐系统算法概述

一个好的推荐系统应该充分利用推荐内容本身的特征，对算法做相应的调整，以达到更好的推荐效果。通过对开源数据集的测试和重要论文的结论，可以总结出用户浏览用户的几个重要特征：

1) 每个用户的新闻兴趣点会随着时间的改变而改变。

每天在世界各个角度发生的事情数不胜数，每天都会有不同的社会热点、突发事件出现，新的社会现象、社会事务出现，被大家关注、被大众淡忘、然后遗忘。

在一个用户连续的浏览记录中，很容易得出用户的新闻兴趣点会随着时间而变化。一般而言，越旧的新闻，用户现在越少可能会再关注。所以在计算新闻关键字，算法会对其权重乘以一个时间衰减因子。

2) 用户关注的新闻兴趣点分为两类：长期兴趣和短期兴趣。

短期新闻一般会是热点新闻事件，它们会经常变动，用户的关注期相对较短。例如“马航失事”、“奥运会”等，这种新闻会一个非常活跃期，之后慢慢淡下去，之后就很难再引起用户的关注。

长期新闻，与短期新闻相对，它是一种相对稳定的兴趣点，是用户真实兴趣的反应。科技达人会长期持续地关注科技事件，金融人士可能会重点关注各种经济相关新闻。

短期新闻一般通过协同过滤推荐算法向用户推荐，也可以通过统计新闻点击量、点击率等新闻特征检测社会热点事件、突发事件。

长期新闻一般通过基于内容的推荐算法向用户推荐，可以通过统计用户在各类新闻上的浏览量作预测，根据历史记录中的新闻关键字可以统计出用户的长期新闻兴趣点。当出现这类新闻时，将它推荐给对应的用户。

3) 公众用户的点击量反应了新闻的趋势，新闻热点。

新闻热点新闻一般点击量会在短时间内飙升，它的点击速率会高于其它新闻。可以通过计算用户的点击速率的计算，当点击速率达到某个阈值，就认为它是热点新闻。

4) 不同的地方，新闻热点不一样。

不同地域关注的新闻热点不一样，这可以是语言限制，也有一些新闻事件本身是地域相关的。例如国外很少会关注中文的新闻，也很少会关注国内一些体育赛事。

5) 同一地区内，用户浏览的新闻热点是大致相同的。

6) 每新闻发布后，有三天的浏览活跃期，七八天的浏览平淡期，十天之后的浏览量会急剧衰减。

通过对数据集的测试，发现一条新闻前三天的浏览量占比达到 70%，之后的七天内浏览量占比 20%左右。这给新闻推荐提出提供了重要指导作用。

在新闻推荐集中，增加 3 天内发布的新闻，适当配置前 4~9 天内的新闻，减少十天之前发布的新闻的权重。

3.2 协同过滤推荐设计

协同过滤作为系统的重要组成部分，思想与传统协同过滤算法思想一致，它的优势随着数据的积累愈加明显。基于内容的推荐算法只能根据已有的关键字做推荐，而协同过滤它能够帮助用户发现新的可以感兴趣的新闻，及时将热点新闻推荐给用户。

协同过滤算法在系统的两个部分发挥重要作用：

1) 根据用户浏览记录分析出相似用户。

每个用户的浏览记录都会保存在 HBase 数据库中，可以利用 MapReduce 计算用户之间的相似度（计算方法是 Jaccard 系数），将计算结果也保存 Redis 之中。

2) 基于用户的协同过滤推荐。

Mahout 以分布式模式做基于用户的协同过滤推荐，这些推荐结果保存在 Redis 之中，待 storm 做实时推荐时，将 Mahout 的推荐结果融入到最终的推荐列表中。

3) 针对十天的新闻库，对其中的新闻做基于 item 的协同过滤算法。

基于新闻条目的协同过滤算法的推荐结果保存到 Redis 之中，每条新闻的推荐结果、推荐值在 storm 做推荐融合时都会用到。

假如 user-A 看了 news-1，则可以将 news-1 的相似新闻作为推荐项。

3.2.1 基于 user 的协同过滤

基于用户的 CF，主要有以下几个步骤

1) 计算当前用户的相似用户。

这部分工作是离线计算的，每六小时启动 Hadoop 的 MapReduce 程序，计算每对用

户之间的相似性，需要计算两种相似性：

第一种，基于谷本系数的相似用户计算。谷本系数英文名 Tanimoto Coefficient-based Similarity，它的计算公式就是 Jaccard 系数。与 cosine、皮尔森等计算不同，它主要适用于没有偏好程度的计算方式，cosine 适用于用户可以评分，比如给电影评分，这个分数可以是 1~10，是一个范围的，而谷本系数适用于没有喜好值，二元值的情况下，本文的数据源恰好就是这种数据，如果一个用户查看了某条新闻，则认为用户喜欢这种新闻，否则就是不喜欢，这正时谷本系数的适用场景。

$$Jaccard(u_i, u_i) = \frac{U_i \cap U_j}{U_j \cup U_i} \tag{3-1}$$

第二种，根据用户浏览记录，系统会从中提出每个用户的关键字和对应的关键字权重。关键字的权重，可以组成一组向量，通过计算 cosine 相似性，度量第对用户之间的相似性。

根据上面计算出来的相似用户、相似用户为每条新闻预测当前用户的评分，可以预测当前用户对某条新闻的评分。

$$r(u_k, n_j)^* = \sum_{u_i \in C_k} I(u_k, u_i) * s(u_i, n_j) / r\left(||C_k||, n_j\right) \tag{3-2}$$

$r(u_k, n_j)$是系统预测用户u_k对新闻条目n_j的评分。

C_k是用户u_k相似用户的集合，$||C_k||$是用户u_k相似用户的集合的大小。

$r\left(||C_k||, n_j\right)$是$C_k$集合中所有浏览过新闻$n_j$的用户个数。

$I(u_k, u_i)$是用户u_k和u_i的相似度，$s(u_i, n_j)$是用户u_i对新闻条目n_j的评分，这个评分是通过浏览时间等因素评估出来的一个评分。

计算出$r(u_k, n_j)^*$之后，乘以一个与时间相关的因子，得出最终的预测评分。

$$r(u_k, n_j) = \alpha * r(u_k, n_j)^* \tag{3-3}$$

这里的α系数是，通过统计新闻发表时间与用户浏览新闻的时间差计算而来的，具体的值会在论文的实验部分阐述。

根据上面的公式，可以得出算法复杂度：$O(n^2 * m^2)$，其中 n 是系统用户个数，m 是平均用户访问新闻队列长度。

Mahout[35]内容实现了计算相似用户、基于内容推荐的接口，过程如图 4-7 所示。在本文设计的系统中，用户日志作为数据流源源不断地注入 web 服务接口，之后再流入 Redis 之中，再流入给 storm 的 spout 接口，最后会写入 Redis 和 HBase 之中。

Mahout 以分布式的方式周期性地为用户做基于用户的推荐的，这些推荐结果保存在 Redis 之中，推荐结果就是一个键值对，并且有每条推荐新闻的推荐值，如果值越高就代表系统预测它越可能被当前用户所喜欢。

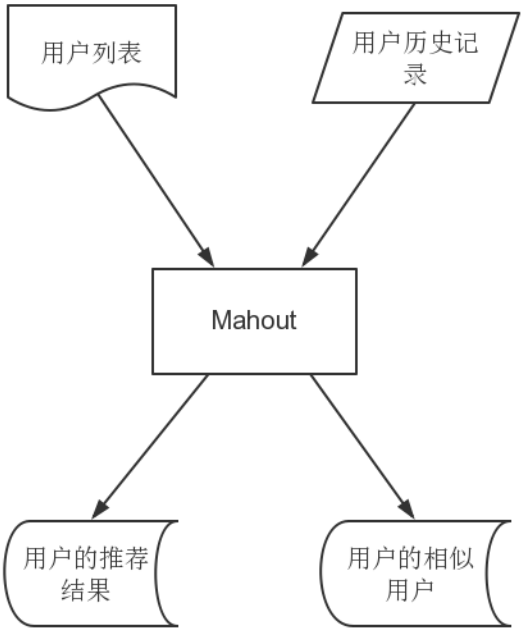

图 2-1 Mahout 协同过滤推荐过程

3.2.2 基于 item 的协同过滤

基于条目的协同过滤算法，是从新闻条目的角度出发，通过一些计算方式找出某个新闻条目的相似新闻，从而推荐给用户。

每一对新闻条目之间的相似性有以下几种度量方式。

- 新闻浏览的用户的相似性

对于新闻库中的第一条新闻，系统会记录所有的浏览者，这些浏览的用户可以组成一个集合，同样可以计算两个集合的 Jaccard 系数。

$$SU(n_i, n_j) = \frac{User_{n_i} \cap User_{n_j}}{User_{n_i} \cup User_{n_j}} \tag{3-4}$$

- 新闻关键字的相似性

前面已经提过了，新闻的关键字有四个来源，从其中可以为每条新闻得出一组关键字和关键字的权重，关键字可以简单概述一篇新闻的大意，因此将关键字的相似性作为两篇新闻的相似性一个重要考量指标。

基于新闻关键字的相似性度量的计算方式是 Jaccard 系数，它的计算公式前面已经

提到，运用到计算关键字的 Jaccard 系数，它的公式是

$$SK(n_i, n_j) = \frac{Key_{n_i} \cap Key_{n_j}}{Key_{n_i} \cup Key_{n_j}} \tag{3-5}$$

Key_{n_i}是新闻条目n_i的关键字，Key_{n_j}是新闻条目n_j的关键字。

基于以上两种新闻相似性度量标准，每一对新闻的相似度结果是

$$Similarity(n_i, n_j) = \mathrm{SU}(n_i, n_j) + \alpha * SK(n_i, n_j) \tag{3-6}$$

本文所设计系统中新闻条目的相似性是基于这两者的混合的结果。其中α是因子，用来调节新闻访问用户队列的相似度和基于关键字相似度之间的权重。

在基于条目的推荐中，一般是通过计算两个条目的浏览用户的向量来计算它们之间的相似性的。在本文的系统中，因为没有涉及到用户对新闻的评分，只是认为如果用户看了一篇新闻，则认为他对这条新闻的评分为 1，否则为 0，所以选择 Jaccard 系数。

基于上面的计算过程，可以得出基于 item 的算法复杂度：$O(p^2 * q^2)$。其中 p 是新闻条目个数，q 是新闻的访问队列长度。

3.2.3 小结

本节中，介绍了两种基于协同过滤算法：用户 CF、新闻条目 CF。与电子商务的协同过滤相比。

1）本文的推荐系统中，没有评分数据，而电子商务系统中，用户可以给商品评分。

2）在电子商务中，商品的发布时间，与用户的购买时间，这两者之间没有什么明显的关系。而新闻推荐系统不同，一条新闻的活跃期只有那么几天时间，而传统的协同过滤算法没有时间的因素，所以在新闻的推荐系统中。推荐库中新闻需要过滤，只留下近期的新闻，然后从中做推荐，这种方式可以提高推荐结果的准确率。

3.3 基于内容的推荐设计

基于内容的推荐是本文的重心，由于新闻的时效性，与传统内容推荐不同，本文的设计加入了关键字衰退因素。基于内容的推荐也包括几个部分

1) 在系统初始化时，引导用户选择自己喜欢的新闻类型，系统将用户的选择结果形成最初始的 profile，根据这个 profile 就可以实现简单的基于内容的推荐，这种方式可以避免系统冷启动问题。

2) 在积累足够多的历史记录时，就可以根据用户的浏览记录生成用户的 profile，profile 是由关键字和关键字的权重组成的。每个用户的 profile 可以视为一个向量。

3) 统计用户浏览新闻的类别，计算用户兴趣点的分布情况，对各类新闻的浏览权重，可以预估用户对新闻类别的喜好程度，在做推荐时，加入这方面的考虑因素。

4) 统计每条新闻的热度，将热点用户也推荐新闻。

3.3.1 基于用户关键字的推荐

3.3.1.1 新闻关键字

新闻关键字也是来自新闻的四个域：新闻标题、新闻正文、新闻页面的 keywords 标签、description 标签。

前面已经提到，新闻的这些数据会保存到 HBase 和 Redis 中，HBase 中存放历史的新闻数据，以及分词后关键字的列表、权重。

对于新闻标题、keywords 标签、description 标签，系统是直接将分词结果中的名词作为关键字，忽略其它词性的分词结果。

TF-IDF 是 NLP 中的一个重要算法，它在信息检索方面应用得十分广泛，但是它也有个问题，一个关键字的权重是 TF（关键字频率）与 IDF（逆文档词频）的乘积，在 IDF 的计算可能会出现问题，例如一个新词出现了（这个新词不重要），它在其它文档中没有出现，所以它的 IDF 会很高，这样会导致一些关键字的权重计算不够准确。

基于信息熵在文档处理方面的优势，所以对新闻正文的权重不是用户 TF-IDF 的计算方式，而是采用信息熵的计算结果作为权重。

做根据信息熵做推荐时，需要将一个新闻文档的关键字处理，保证一篇新闻的所有关键字的信息熵之和是 1，这样做是为在统计用户的关键字权重时，保证每篇新闻的权重值是一样的。

3.3.1.2 用户关键字

用户的 Profile 关键字来自于用户浏览的新闻，系统会一个新闻中的四个域：新闻标题、新闻正文内容、新闻 HTML 的 keyword 标签、新闻 HTML 的 description 标签内容。

所有关键字是利用 NLPIR 汉语分词系统分词[36]得出的关键字，NLPIR 分词系统是由张华平博士开发的，目前已经更新到 2015 版本，并支持多种编码，全球的用户数量超过 20 万。

下面着重讲解用户关键字 profile 是怎么生成的，首先 profile 的关键字是名词，系统会对用户浏览的新闻标题、正文、页面 keyword 标签、页面 description 标签四个域进行分词，并标注词性，关键字的权重。

$$w_{i,key} = \sum \alpha * e \tag{3-7}$$

e 是关键字的信息熵，信息熵会先被归一化；α是一个与时间相关的衰退因子，用户 i的关键字key的权重用$w_{i,key}$表示，它的值是用户 i浏览的每个标题的$\alpha * e$值的叠加。通过这种方式计算出用户 i 的每个关键字权重。

每个用户的关键字权重可以组成一个向量：

$$Weight = \{w_1, w_2, \dots w_j \dots w_n\} \tag{3-8}$$

w_j 表示第 j 个关键字的权重，n 是关键字的个数。

接下来，将新闻库的新闻当作搜索引擎背后的数据，针对 profile 的每个关键字，每个新闻会有一个相关度的值。

$$A = \begin{matrix} a_{1,1} & \dots\dots & a_{1,n} \\ \dots\dots & a_{i,j} & \dots\dots \\ a_{m,1} & \dots\dots & a_{m,n} \end{matrix} \tag{3-9}$$

对于上面的矩阵 A，n 是关键字的个数，m 是新闻条目的数量，$a_{i,j}$表示关键字 j 在新闻条目 i 的权重值，这个权重值就是 TF-IDF 算法计算出来的；矩阵的第 j 列表示，关键字 j 在每个新闻条目的权重。每个新闻对于用户的权重的结果计算方式：

$$NewsWeight = A * Weight^T \tag{3-10}$$

计算过程的变数就是式(1)的衰减因子α,可以有

- ✧ Log 衰减，$\alpha = log(t_{now} - t_{view})$。$t_{now}$是当前时间戳，$t_{view}$是浏览新闻时的时间戳。
- ✧ 正态分布$\alpha = \frac{1}{\sqrt{2\pi\delta}} exp\left(\frac{-(x-\mu)^2}{2\delta^2}\right)$

  μ是当前的时间戳，δ是个可变的参数。

本文设计的推荐系统与其它系统相区别的一个重要特点就是我们的数据包含用户新闻的浏览时长，一般而言，用户对新闻所花的时间越长，则可以认为其对这条新闻的关注度越高，其中的关键字所占的权重就应该更高，将这个作为考量因素，有利于提高系统的推荐质量。

为了加快实时计算速度，所有用户的关键字权重列表存放在 Redis 的 list 中。

计算用户关键字的伪代码

```
统计用户集合
for (user_id in set<User_ID>)
{
```

```
        根据用户编号，找出用户的历史浏览记录
        set<News_ID> = getNewsIDsInLogs(user_id)
        for (news_id in set<News_ID>)
        {
            找出每条新闻标题的关键字和信息熵
            map_title<keyword, entropy> = getMapEntropy_title(news_id)
            找出每条新闻正文的关键字和信息熵
            map_content<keyword, entropy> = getEntropy_content(news_id)
            找出每条新闻 keywords 标签的关键字和信息熵
    map_keywords<keyword,entropy> = getEntropy_keywords(news_id)
                找出每条新闻 descirption 标签的关键字和信息熵
     map_description<keyword,entropy>= getEntropy_description(news_id)

            为每个域中的关键字设置一个权重值，然后将关键字、信息熵合并成一个 map
            mapUserID = a1 * map_title + a2 * map_content + a3 * map_description + a4 * map_keywords
        }
    }
```

3.3.1.3 新闻关键字倒排索引

系统只会对新闻库中的新闻建立倒排索引，在做基于用户关键字的推荐时，storm 会从 Redis 中取出用户的关键字列表和每个关键字的权重，再根据这些关键字及其权重值，查询关键字倒排索引，为每个新闻给出一个评分。

Storm 根据用户关键字对每条新闻的预测评分会做一个归一化处理，最后这些评分流入混合推荐模块。

归一化的处理很简单，只是每条新闻的评分都除以评分中最高的数。

3.3.2 用户近期浏览处理

用户的浏览记录流入 Storm 中，处理过程如下：

用户的每一条浏览记录中，包含几个个字段：新闻编号、用户编号、用户浏览时间。会存放 HBase 之中，后经过 storm 处理后，有三个计算结果：

1) 将当前用户加入对应的 URL 新闻的浏览用户列表中，后面计算 URL 新闻的相似性是根据这个浏览用户列表做计算的。

2) 将 URL 新闻加入当前用户的新闻浏览列表中，这些新闻在根据相似用户的做协同过滤推荐时需要用到。

3) 更新用户的关键字权重。由于新闻标题、正文事先已经被分词存入 HBase 之中。为找出当前用户关键字权重。首先从 HBase 中找出浏览的新闻 URL 对应的关键字信息，结合浏览时长，为这些关键字赋上一定的权重值，之后再加入当前用户的关键字列表中。

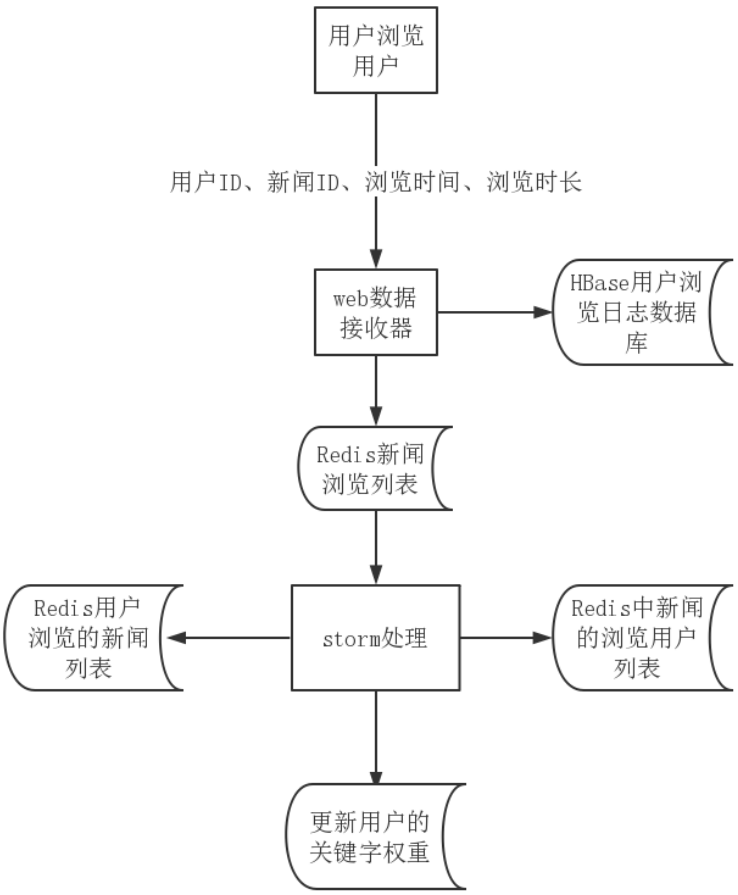

图 3-2 用户日志处理模块

Storm 关键字处理

上面已经提过，storm 的任务是以 topology 单位的，每个 storm 有若干个 pout 做数据流入接口，同时也可以有，这个 spout 是从 Redis 中获取数据，将 Redis 作为数据流入，可以有 N 个 bolt 做逻辑处理。

Redis 没有传统关系数据库中数据库表的概念，可以 HMset 实现类似 Table 的数据

结构。用户关键字的处理涉及三方面的数据。

1) 新闻库中每条新闻关键字列表、关键字的权重。

2) 用户关键字、关键字的权重。

3) 新闻库中每个关键字的倒排索引。

用户本身会有一个关键字列表和权重，定义为

$$kw = \{k_1 = w_1, k_2 = w_2, \dots \dots k_m = w_m, \} \tag{3-11}$$

m 是用户关键字个数，k_i是指用户的关键字，w_i是关键字k_1的权重值。

用户的每个关键字k_i在第一条新闻中会有一个评分，这个评分，可以通过 TF-IDF 算法得到。假设新闻库中有 n 条新闻，当前用户有 m 个关键字，n 条新闻对 m 个关键字都会有一个评分，这样会生成一个 $n * m$ 的二维矩阵。

$$matrix = \begin{matrix} s_{1,1} & \dots & s_{1,m} \\ \dots & s_{i,j} & \dots \\ \dots & \dots & s_{n,m} \end{matrix} \tag{3-12}$$

其中$s_{i,j}$是第 j 个关键字在第 i 条新闻中的评分。最后预测用户对每条新闻的喜好程度：

$$NewsWeight = matrix * kw \tag{3-13}$$

会得到一个 n*1 的向量，第 i 条值就是根据关键字预测用户对第 i 条新闻的喜好程度。

3.3.3 用户新闻类别兴趣权重

Google News 中的贝叶斯预测方法[16]。根据历史记录，计算每个用户对当前新闻类别的点击率的先验概率，再预测出每个用户对某个新闻条目的评分。

在本文设计的推荐系统中，与 Google News 的推荐算法不同，不是利用贝叶斯计算用户对每种新闻的喜好程度。而是通过计算用户对每种新闻的点击量来作为标准。在数据采集的 Android 程序中，会为每个页面建立一个标签页，用户自主选择自己感兴趣的新闻进行查看，用户对每个新闻类别的点击量能反应用户对每种新闻的喜好程序，所以系统没有采集贝叶斯预测法，最后统计用户对每个新闻类别的点击量，将点击量作为用户的兴趣权重值。

在做推荐时，会为每个新闻条目根据它的类别给定一个初始权重值。

用户所有的浏览记录会保存在 HBase 之中，系统会以 15 天为统计周期，计算用户对新闻类别的兴趣权重，根据上面所提到的，用户的兴趣可能会随着时间的推移而改变，所以周期性的计算方式也方便后续进行考虑衰减因子。

用户对每个新闻类别的兴趣权重也保存在 Redis 中，方便 storm 的实时计算。

3.3.4 热点新闻的计算与推荐

在前面章节已经提及，“同一地区内，用户浏览的新闻热点是大致相同的”，所以系统会统计每个地区的新闻热点，然后将热点新闻推荐给在这个地区的用户。

热点新闻计算有三个条件

1) 新闻的浏览量达到一个阈值。

既然是热点新闻，就代表这个用户已经被很多人浏览过，所以它的浏览量必须达到一个阈值，系统才认为它可能是一个热点新闻。

2) 新闻的浏览速率达到一个阈值。

点击速率 = 浏览量 / 时间区间。

浏览量就是目前这个新闻被多少个用户点击查看，时间区间是新闻发布时间与当前时间的时间差。

3) 新闻的展示与用户浏览比达到一个阈值。

如果一个新闻是一个热点新闻，则认为只要它一旦展示给用户，用户就很大概率会点击进入浏览它的内容。所以也将新闻展示与用户浏览比作为一个参考指标。

第四章 基于 storm 新闻推荐系统的实现

本文的思想在前面的篇幅已经简略提及，本文所设计的新闻推荐系统推荐结果由两部分组成，一方面是协同过滤推荐，另一方面是基于内容的推荐结果。

协同过滤的推荐包括基于用户的 CF 和基于新闻条目的 CF，而基于内容的推荐包括利用新闻类别、新闻关键字、新闻发表时间这三个内容而做的推荐。

4.1 新闻推荐系统实现概述

Google 于 2007 年发表的论文中就提出了基于内容、基于协同过滤的混合推荐系统。它的内容是从新闻浏览的新闻类别入手，利用贝叶斯模型[37]预测用户的兴趣；再结合协同过滤算法，最终的推荐结果是两个算法的混合。由于 Google 的隐私政策，Google 的推荐系统中没有利用用户的浏览时长。

本文设计的推荐系统，实现了一整套解决方案，包括生成数据的安卓客户端程序、数据接收的 web 服务模块、日志收集模块、新闻推荐模块。

用户的浏览历史保存在 HBase 之中，MapReduce 用于从历史记录中推荐相似用户，storm 负责实时推荐。由于新闻的时效性，本文推荐系统所推荐的新闻是过去十天内的，推荐库是十天的新闻库。

用户浏览的新闻有如下几个特点，对于一个用户而言，用户感兴趣的新闻分为两类，一类是长期的兴趣点；另一类是短期的兴趣点。短期的兴趣点一般是指社会热点新闻、各种突发事件，这些新闻会经常改变，不固定。而长期的兴趣点常常反应着个人喜好，相对稳定。

基于用户浏览新闻的四个特点，针对短期兴趣点新闻和长期兴趣点的新闻，需要有针对性地做推荐。长期兴趣点可以通过分析用户的历史记录，从中挖掘出来，偏向于基于内容的推荐方式。而这种短期兴趣的突发性新闻，可以通过相似用户的浏览行为做推荐，也可以将同一地区的用户的浏览热点推荐给用户。

经过对数据集测试，发现了用户浏览时间与新闻发布时间的差值关系。一篇新闻在发布后的前三天用户浏览量占比超过 70%，而第四天到第八天，用户浏览量占比 20%左右。

利用用户浏览时间与样子发布时间的差值关系，三天以前的新闻推荐完全可以通过离线计算完成；最近三天的新闻推荐结果可以利用近实时计算得到；最后可以利用实时

计算，得出今天的推荐结果，同时综合离线计算和近实时计算的结果，为用户生成一个最终失推荐结果。

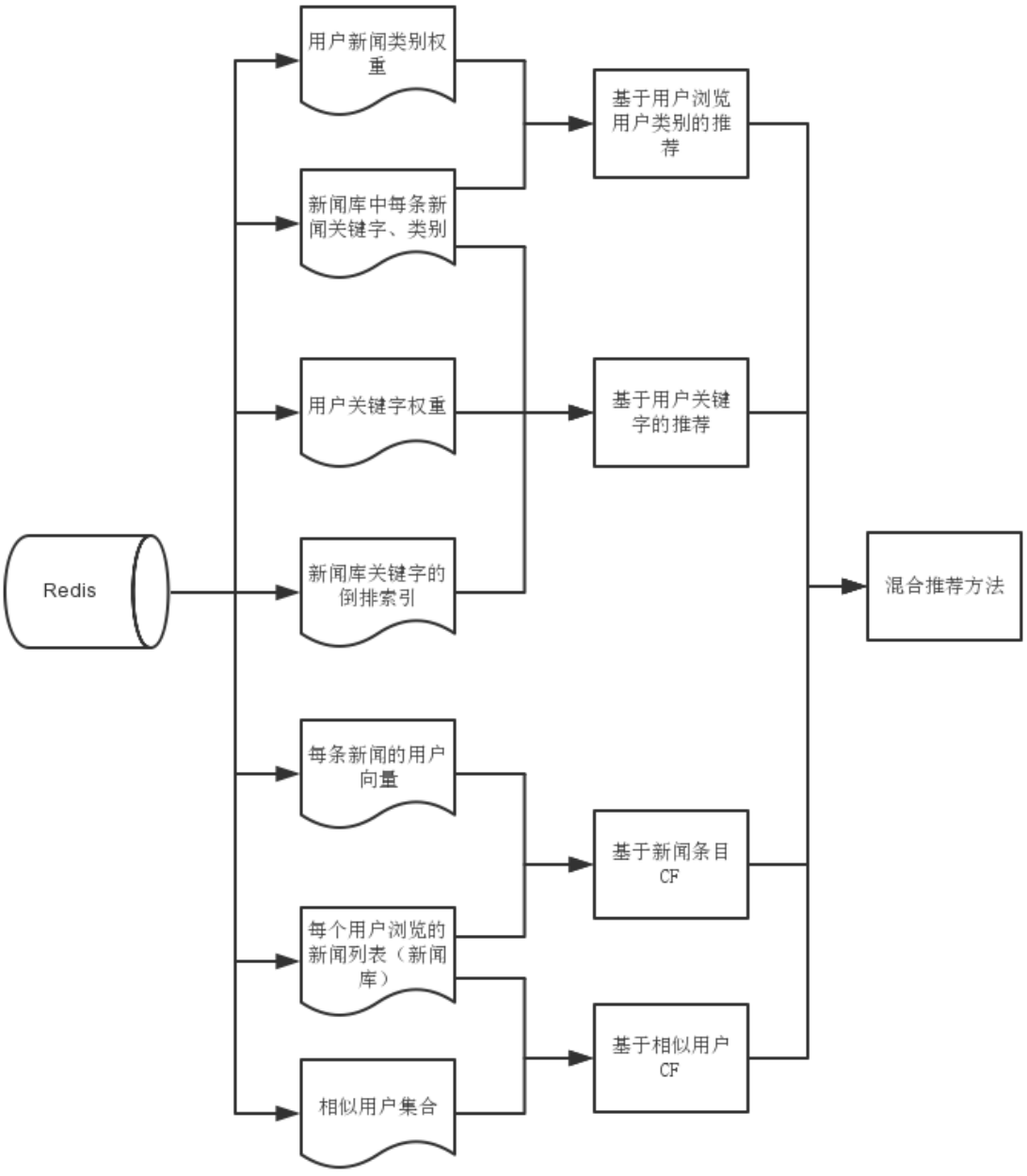

图 4-1 storm 推荐模块

本章节主要介绍推荐系统的设计与实现，涉及的一个推荐系统的方方面面。从用户浏览数据采集、接收、分析、中间存储；新闻数据的采集、存储，新闻文本的分析，关键字统计。

为了提高推荐速度，整个推荐过程，包括离线计算、近实时计算、实时计算。

1) 离线计算。

离线计算是由 Hadoop MapReduce job 完成，它会以 6 小时为单位，周期性启动统计用户的关键字，计算相似用户。在基于现有新闻库中，为用户计算出一个基于用户、基于新闻条目的协同过滤推荐结果。

2) 近实时计算

近实时计算是为了减少“实时计算”的压力，以一种近实时计算的方式做一些增量计算。更新新闻条目的相似性、更新相似用户的相似性。

3) 实时计算

基于离线计算的结果和近实时推荐的结果，为用户提供一个实时的推荐列表。还有更新用户浏览记录。

本文在核心在在个性化的推荐部分，它是基于内容推荐、基于协同过滤算法推荐的融合。在基于协同过滤推荐方面，与传统协同过滤算法相相似。在基于内容推荐方面，有两个方面，一方面是根据用户的浏览记录，分析用户对各个新闻类别的喜好程度；另一方面就是根据用户浏览新闻的关键字，提取用户关注的关键字权重。最后的推荐结果有四个方面的推荐。

1) 基于用户的协同过滤推荐。

这是传统的协同过滤算法。

第一步，计算相似用户，根据用户历史记录，找出每个用户的相似用户集合，同时给出当前用户与相似用户的相似度数值。

第二步，根据相似用户，以及之间的相似度，预测用户的推荐列表。

系统测试的数据集中，有 100,000 条浏览记录，10,000 个用户，平均每个用户只有十条的浏览记录。这样的数据规模给基于用户的协同过滤推荐带来了一定的稀疏性问题。在测试环节中，发现通过这种最原始的基于用户的协同过滤推荐，它的推荐效果很不好，大量用户的召回率为 0。所以后来考虑利用相似用户的推荐集扩充推荐集，这样提高召回率。

所有需要记录过去一段时间内，每个用户的相似用户集，心及它们之间的相似度数值。计算相似用户后，将这些信息存放到 Redis 中，在 storm 实时处理过程，会有一个模块，统计当前用户的相似用户所浏览过和新闻，根据浏览量、浏览时间计算出一个推荐列表。

2) 基于新闻条目的协同过滤推荐。

与传统基于 item 推荐算法相似，用户所有的浏览记录会流入 storm 中，storm 再将

用户放入新闻的浏览列表中，浏览的用户可以组成一个向量，利用 cosine 相似性，计算出新闻条目之间的相似性。此为用户做推荐时，此算法模块会根据新闻之间的相似性，将用户浏览过的新闻的相似新闻推荐出来。

3) 基于用户关键字的推荐

用户所有的浏览记录会保存到 HBase 中，每天凌晨零点，系统会统计用户昨天浏览的所有新闻 URL，根据这些 URL 找出用户浏览的新闻关键字，这些关键字不会被赋予权重。这些信息会以 list 形式保存在 Redis 之中。

同时，最近的新闻库里的新闻的正文、标题等分词结果会保存在 Redis 之中，也会为关键字建立倒排索引，方便根据用户历史统计的关键字找出这些关键字相关的新闻。

这个模块的推荐，就是基于最近的亲库中每条新闻的关键字、用户浏览历史

的关键字，将两者之间做匹配，找出其中最匹配的 top N 新闻作为推荐项。

4) 基于用户新闻类别权重分布的推荐

这部分的推荐方法，参考了 Google 的贝叶斯预测方法，分析用户浏览历史中，每种新闻类别会有一个兴趣权重，这个兴趣可以通过贝叶斯计算出来，

简单来说，根据这些新闻兴趣权重，每一个新闻条目会有兴趣值，这个值可以应用于推荐之中。

4.1.1 系统中的数据

系统中的数据是来源于 CCF 比赛数据，大约有 100,000 条用户浏览记录，用户数据有 10,000 个，用户是通过随机抽取的方式确定的，用户浏览记录的数据时间段是 2014 年 3 月份。

原始数据中的每一行包括六个字段：用户编号、新闻编号、浏览时间、新闻标题、新闻正文内容、新闻发布时间。

为了提高推荐质量，本文所设计的系统中，为每条新闻增加了几个字段：新闻 URL，新闻页面中 keywords，description 标签。

由于 CCF 的比赛数据来源于国内某门户网站，所有通过新闻标题和域名过滤，可以在搜索引擎中获取新闻的 URL，根据新闻 URL 中，可以得出用户的类别、新闻 keywords 标签、新闻 description 标签。

这些标签内容为基于内容推荐提供数据基础。

新闻类别包括：经济、金融、公司、政经、世界、文化、图片、视频。

新闻关键字是通过 NLPIR 汉语分词系统分词而来的，每个关键字信息熵值作为权

重值。

整个系统中，会涉及以下几种数据

1) 新闻库。

包括新闻 ID、新闻正文、新闻类别、HTML 中的 keyword, description 标签、新闻标题分词结果、新闻正文分词、keywords 分词、description 分词结果。历史数据存放在 HBase 之中，近期的数据存放在 Redis 之中。

2) 用户的浏览记录。

包括用户编号、用户浏览某条新闻的新闻编号、浏览时间，这些数据放在 HBase 之中。近期新闻库中新闻的浏览记录同时也会保存到 Redis 之中。

3) 相似用户的集合。

相似用户列表、相似度在基于用户的协同过滤推荐过程需要用到，存放在 Redis 之中。

4) 用户的关键字，以及关键字的权重。

这些数据在基于内容的推荐过程会被频繁使用，它们存放在 Redis 之中。

5) 近期新闻库中关键字信息，以及每个关键字的倒排索引，存放于 Redis 之中。

6) 近期新闻库中，每条新闻的浏览用户信息，这些数据在基于新闻条目的协同过滤推荐过程需要用到，它们也是放置在 Redis 之中。

4.1.2 新闻客户端

为了接收用户的浏览记录，开发一款 Android app。其界面如图 4-2 所示。

程序的左侧有一个抽屉式的菜单，可以直接切换新闻类别，也可以手势左右滑动来切换新闻类别，在同一个新闻类别列表中，可以手势上下滑动可以实现列表滚动的效果。

新闻客户端主要有两个功能：

1) 向服务器端请求新闻数据。

在客户端请求数据之前，系统会将新闻库放到 Redis 数据库。客户端程序只需要通过 web services 向 Redis 请求新闻数据即可。在请求数据时，也会发送用户 ID、客户端当前最新的新闻 ID。Web services 接收请求后，会将最新的新闻、推荐给用户的新闻以 json 格式返回给用户，json 格式中的数据项有用户标题、新闻 URL、新闻发布时间的。

2) 用户浏览数据的产生器。

它会记录用户请求的新闻列表、用户浏览了新闻、浏览的时间，用户的每次点击事件都会发送给服务器，服务器接收后将这些数据，流入到 storm，storm 会将这些信息立

即更新到相似的数据项当中。例如将用户信息加入到新闻的访问队列中，在用户的访问队列中加上当前新闻编号，将当前新闻的关键字加到当前用户的关键字列表之中。

图 4-2 android 界面

4.1.3 数据接收

在一般数据规模不大的情况下，只需要写一个简单的 web service，就可以接

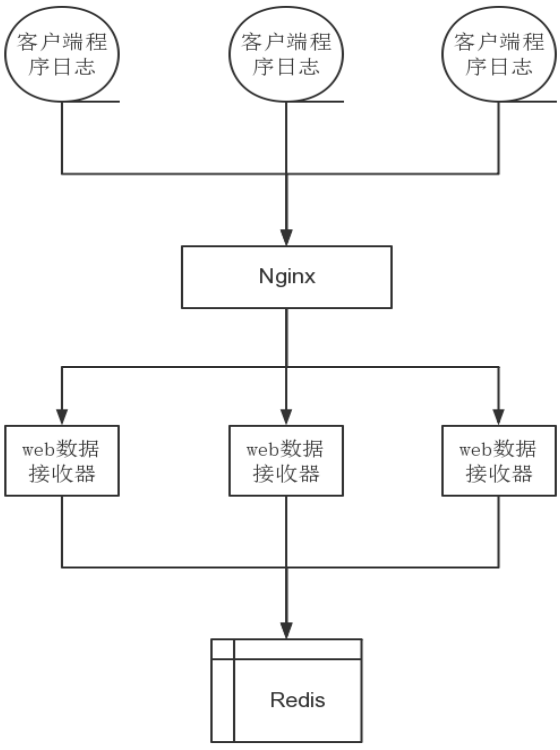

图 4-3 日志收集过程

收客户的数据。但是对于一个大规模的用户群系统而言，这种数据接收机制是不适应的，为了实现对能够接收大规模的用户数据，可以使用 Nginx[38]做负载均衡，做用户的请求尽量均匀地分配给后台的 web 数据接收服务器。为了加快用户浏览日志的保存，再将这些日志保存到内存之中，当数据量达到一定阈值之后，再批量传入 Redis 中，之后经由 storm 处理。系统的日志采集过程如图 4-5 所示。

4.1.4 数据库表设计

数据库作为数据容器，在整个系统当中发挥着重要作用。系统用到了 HBase 数据库和 Redis 数据库。HBase 适合存储海量数据，且不会经常被更改，在系统模块中，主要用来存储新闻数据、用户浏览历史、用户关键字表。

新闻数据表

新闻数据表是 HBase Table 作为新闻的存储容器，它保存着新闻的相关数据，以及新闻标题、正文、keywords、description 标签四个域的分词结果。新闻编号是唯一标识，作为 *rowkey*

表 4-1 新闻数据表

rowkey	新闻编号
Content	新闻正文
Content_word	新闻正文分词后，关键字列表、关键字信息熵
Description	新闻 HTML 中 description 标签
Description_word	Description 标签分词后的关键字和信息熵
Keywords	新闻 HTML 中的 keywords 标签
Keywords_word	Keywords 标签分词后的关键字和信息熵。
Publish_time	新闻发布时间。
Title	新闻标题。
Title_word	新闻标题分词后的结果。
URL	新闻 URL，从 URL 中可以知道用户的类别。

用户日志表

用户浏览日志信息经过 storm，最终会保存在 HBase 的用户日志表中，这个表的数据项比较少，数据项有如下：

表 4-2 用户日志表

Rowkey	用户编号#新闻编号
News_id	新闻编号
User_id	用户编号
View_time	用户浏览时间

用户关键字表

用户关键字表是通过 MapReduc 结合“用户日志表”和“新闻数据表”计算出来的，由于新闻的关键字来源于四个域，所以用户的关键字也有对应的四个域。

Rowkey 是用户编号加入时间区间。因为用户关键字是周期性统计的，之后会根据每个周期的关键字数据，将每个用户的关键字再做聚合。

表 4-3 用户关键字

Rowkey	用户编号#时间区间
Keyvalue_title	新闻标题中关键字列表及权重
Keyvalue_content	新闻正文中关键字列表及权重
Keyvalue_keywords	新闻页面中 keyowrds 标签中关键字列表及权重
Keyvalue_description	新闻页面中 description 标签中关键字列表及权重

4.1.5 小结

本节，从系统角度出发，提纲挈领地简述了本文设计的系统的各个模块。数据采集、数据接收、数据存储、数据处理流程。各个流程的算是结果是什么，存放在哪是，以及后续怎么利用。

4.2 离线计算

系统的计算包括两种，一种是离线计算，另一种是实时计算。离线计算是由 MapReduce 完成，而实时计算是由 storm 完成。

离线计算任务会每六小时启动一次，根据目前已有的浏览日志、新闻分词数据等，利用基于用户的 CF、基于新闻条目的 CF、用户关键字、新闻类别等信息为每个用户生成一个推荐列表。同时它会记录已有用户的相似度、更新用户的关键字、统计用户的浏览新闻类型。

4.2.1 计算相似用户

相似用户的计算是以七天为单位进行计算的。相似用户作为基于用户协同过滤的数量基础，在基于用户的 CF 过程中，是通过根据相似用户的浏览历史与相似度，为用户生成一个推荐列表。

相似用户是通过 Mapreduce 计算出来的，每对用户的相似度是利用 Jaccard 系数作度量的。

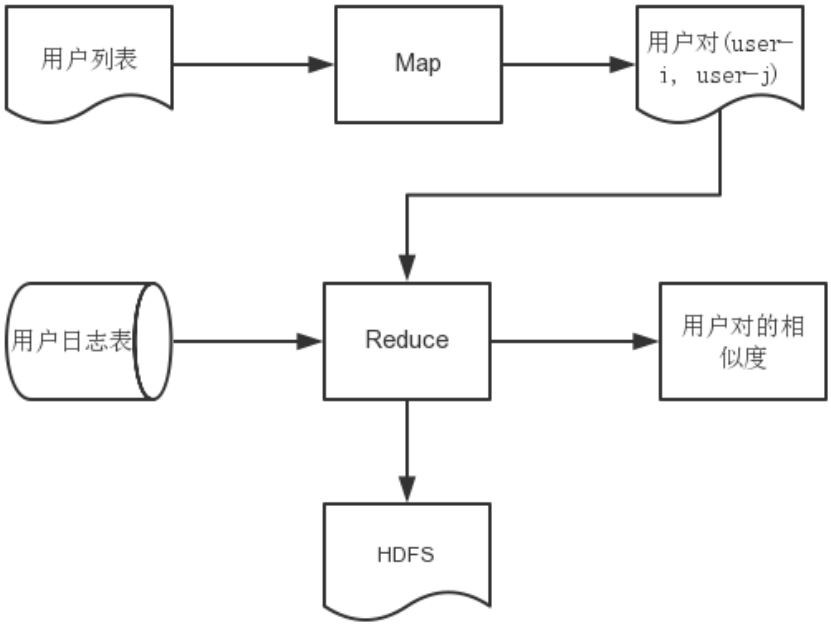

图 4-4 MapReduc 计算相似用户

在计算用户相似度上，算法复杂度还是挺高的，是$O(n^2)$的复杂度。

算法伪代码

Map 过程
Long userID = value.toString(); *For* (*id in user_list*)*{* // 通过这个 if 判断可以保证每个 Map 的用户对不会重复

```
    If (id > userID) {
        Context.write(userID+"#"+id, 1)
    }
}
```

Reduce 过程

```
Long userID = getUser1();
Long id = getUser2();
Set<Long> set1 = getNewsID(userID);
Set<Long> set2 = getNewsID(id);
Set<Long> union = set1 + set2;
Set<Long> intersection = set1 ^ set2;
Context.write(userID+"#"id , intersection.size() / union.size()))
```

在 Map 阶段，产生一系列的用户对，将这些用户对，输出到 Reduce 阶段，Reduce 针对获取的所有有用户对，分别得出他们的访问列表，将两个访问列表交集的长度 除以 两个访问列表的并集长度。将他们的商作为相似度的值，结果存储在 HDFS 当中。

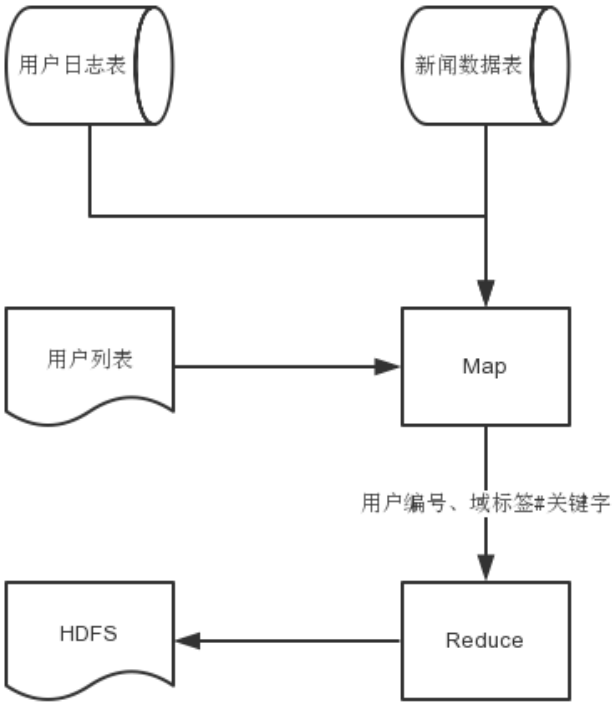

图 4-5 统计用户关键字

4.2.2 统计用户关键字

关键字作为基于内容推荐中的基础，对关键字字处理会直接影响系统后续的推荐结果，在本文论述的系统中，关键字的提取是通过 NLPIR 分词系统。

系统对新闻正文、新闻标题、HTML 页面中的 keywords，description 标签进行分词，而且对于分词结果只取其中的名词性质的分词单元。

对于新闻正文而言，会出现很多不重要的词语，所以系统不是将关键字出现的次数作为衡量的权重，而是利用信息熵，信息熵代表了信息的价值，这种方法可以减少新闻正文分词的数据大小，又可以更加准确地描述关键字的重要性。

信息熵的值，与新闻内容无关，它只是表示关键字的随机性，无论新闻正文的内容是什么，只要关键字是相同的概率分布，它们的信息熵就会相等。用户的关键字统计也是采用 MapReduce 做计算得来的，如图 4-5 所示，每天凌晨会自动启动关键字统计的 MapReduce job。

Map 阶段，首先获取每个用户的浏览新闻编号，根据新闻编号获取新闻对应新闻的四个域的关键字，最后将用户编号、关键字、域标签输出给 Reduce。

Reduce 阶段，读取某个用户的所有关键字，之后根据域标签统计关键字以及关键字权重，将结果写入 HDFS 之中。

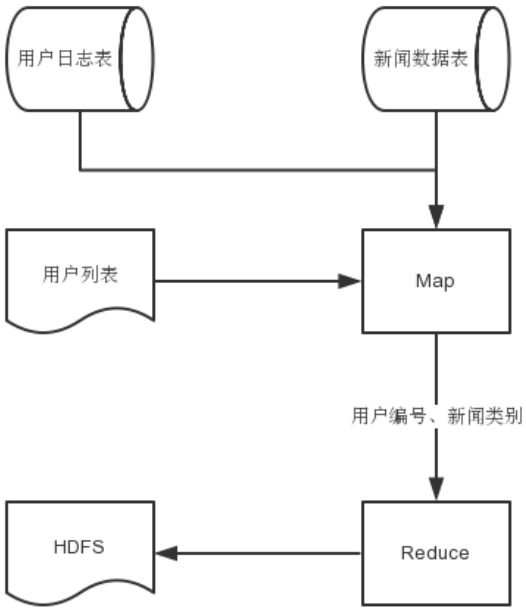

图 4-6 统计用户浏览新闻类别

4.2.3 统计用户浏览的新闻类别

用户浏览的新闻类别的统计，与用户关键字的统计过程相似，在关键字的统计是以关键字为 key，而新闻类别是以新闻类别为 key。计算过程如图 4-6 所示。

Map 阶段：

- 从用户日志表中，找出用户浏览的所有新闻编号。
- 根据新闻编号从新闻数据表中读取 URL。
- 根据新闻 URL 判断新闻的类别
- 后将用户编号、新闻类别输出给 Reduce。

Reduce 阶段：遍历 Map 阶段的所有键值对，再计算每个新闻类别的数量，最后写入 HDFS 当中。

4.3 近实时计算

近实时的计算工作主要是为了减少实时计算的压力。它的运行周期是二十分钟，每隔二十分钟启动，根据过去二十分钟内用户浏览信息，更新用户相似度、新闻条目相似度、用户关键字信息。

与离线计算不同的是，为了提高近实时计算的速度，系统会对用户做聚类。通过聚类，可以大大减少计算复杂度，以提高近实时计算的效率。

与离线计算相似的是，近实时计算结果也包括以下几个方面，与离线计算不同的是，它的输入数据是在 Redis 之中，同时它的计算结果又会与离线计算结果、前期的近实时计算结果合并。

- 利用 K-means 算法，对用户进行聚类。
- 与离线计算的用户相似度进行合并

  利用 Redis 记录的用户浏览日志，同样是用 Jaccard 系数作为相似性度量。
- 与离线计算的新闻条目相似度进行合并
- 利用 TD-IDF 算法，结合用户的关键字，与当前新闻库中关键字信息生成一个基于关键字的推荐列表。
- 与离线计算的新闻推荐列表合并，生成新闻的推荐列表。

4.4 实时计算

实时计算是由 storm 完成的，storm 的运行单位是 Topology（拓扑），spout 作为 Topology 的数据源头，将数据源源不断地流入 storm，之后 bolt 会作为数据处理的单位，

进行各种过滤、聚合操作。

在 storm 实时处理过程，Redis 作为 storm 数据源，也作为数据容器，既保存了用户浏览记录、新闻信息，也保存了倒排索引、用户的浏览新闻列表、新闻的访问用户列表。

4.4.1 统计热点新闻

新闻四大特点之一就是“公众用户的点击量反应了新闻的趋势，新闻热点”。一个用户很大程度上会关注同一地区用户关注的热点新闻。

因此需要统计同一地区用户的新闻点击情况。首先将用户划分为不同区域用户；根据当前用户浏览记录，更新对应新闻的浏览量。

4.4.2 更新新闻库

最近的新闻库会保存在 Redis 之中，storm 会对新闻做处理，建立并更新倒排索引，同时将新闻的各个字段插入到 HBase 之中。

对于每条新闻，Redis 会保留新闻编号、新闻标题、新闻 URL、新闻发布时间、新闻关键字以及关键字的信息熵。

4.4.3 用户浏览日志的处理

在用户数据收集的章节中，已经提及。用户浏览日志收集后会流入 Redis。Storm 会从 Redis 数据库中获取记录，对它们做相应处理，如图 4-7 所示。

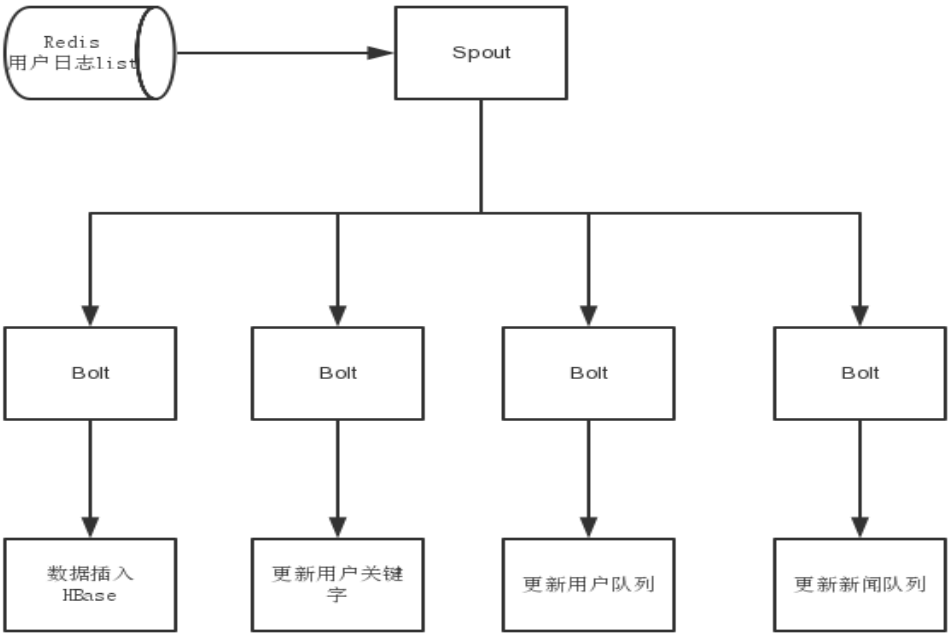

图 4-7 用户浏览日志处理

1) 将 Redis 中每一条用户浏览记录插入到“用户日志表”中。

用户的浏览数据最终会保存到 HBase 之中，Redis 只是用于实时推荐，加快系统日志收集速度，作为中间存储。

2) 新闻用户关键字。

上面章节已经介绍如何计算用户关键字中，在系统运行过程中，新闻关键字会保存在 Redis 数据库中。当一条新闻的浏览日志进入时，storm 需要实时更新对应用户的关键字权重。

3) 将当前用户添加到浏览新闻的访问队列之中。

在基于新闻条目的协同过滤推荐模块中，需要计算新闻条目之间的相似性。相似性的值就是通过计算每两个新闻条目的访问者集合的 Jaccard 系数。

所以需要为新闻库中每条新闻维护一个浏览用户的队列。每收集一条用户浏览日志就会更新新闻访问者队列。

4) 将浏览的新闻编号添加到用户的浏览队列中。

在基于用户的协同过滤推荐模块中，是根据相似用户的浏览记录，预测当前用户对每条新闻的喜好程序。算法的前提是得到相似用户集合。

与相似条目的计算方法类似，相似用户的计算也是通过 Jaccard 系数。所以需要为每个用户维护一条浏览的新闻队列。

与新闻条目的相似性不同，因为新闻时效性的特点，新闻条目的相似值是局限于目前新闻库中新闻的相似性。而相似用户还会根据历史记录中相似性，也就是说用户的相似性是历史记录中相似性和新闻库中新闻浏览记录的相似性的总和。

4.4.4 推荐集合

正如上面提到的，离线计算是以六小时为周期，近实时计算是以二十分钟为单位。实时计算正是利用离线计算和近实时计算的结果，

上面已经介绍了基于协同过滤推荐、基于内容推荐的细节。最后需要将各个模块的推荐结果融合，以产生最后的推荐集合。

U_i代表在基于用户的协同过滤中，对新闻条目 i 的推荐值。

I_i 代表在基于新闻条目的协同过滤中，对新闻条目 i 的推荐值。

K_i代表在基于内容推荐中，对新闻条目 i 的推荐值。

C_i 代表根据用户的历史记录，根据新闻条目 i 的新闻类别，对新闻条目 i 的推荐值。

最后对新闻条目 i 的推荐值计算公式如下：

$$Recommend_i = U_i + I_i + K_i + C_i \quad (4\text{-}1)$$

4.5 本章小结

本章详细介绍系统算法细节，以及系统的各个模块。

1) 介绍系统总结框架，数据的存放、数据分析、推荐模块等。
2) 细化了基于协同过滤推荐算法的实现过程，如何计算用户相似性，如何计算新闻条目相似性，如何生成基于协同过滤的推荐列表。
3) 详述了基于内容的推荐算法的实现过程，如何统计用户关键字，如何计算关键字权重，如何根据用户关键字生成用户的推荐列表。
4) 分析了离线计算、近实时计算、实时计算过程。
5) 讲解 storm 如何融合两种推荐算法的结果，为用户生成一个最终的推荐结果。

第五章 实验验证及分析

5.1 测试数据

在前面中已经提及过，本文中用到的数据是基于 2014 年 CCF 竞赛的数据[39]，用户浏览记录有 100,000 条，用户 10,000 个。浏览记录的时间跨度是 30 天，相当于平均一个用户一个月内只看了 10 条数据，这种数据太过稀疏了。为了保证保证测试结果的有效性，测试中用的是浏览记录超过 30 条的用户，这样的用户数有 456 个。

为的提高推荐的准确性，人为地扩充了数据内容，从搜索引擎上爬取每条新闻的 URL 中，从 URL 对应的页面中，可以获取新闻类别、新闻 keywords 标签、新闻 description 标签。

这些信息为基于内容的推荐提供了一定的数据基础，原始数据中的新闻数据项只有三个：新闻标题、新闻正文、新闻发表时间。

5.1.1 基于用户协同过滤推荐结果

为了测试推荐算法的有效性，系统进行了多组测试。基于用户的协同过滤涉及两个参数：邻居数量、用户浏览数据。

测试时，邻居数是从 5 个开始递增，10 个、15 个、20 个，一直到 50 个邻居。用户的浏览数据从前 20 天开始，前 21 天、22 天，一直增加到 29 天。

邻居数据有十种情况，用户浏览数据有 10 种情况，所以在测试时，每个用户会有 100 个基于用户协同过滤推荐结果。

5.1.2 基于新闻条目协同过滤推荐结果

与基于用户 CF 推荐相似，测试的数据集从前 20 天，一直增加到前 29 天。

由于基于新闻条目的 CF 只与用户浏览数据相关，所以在基于新闻条目推荐的结果里，每个用户只有十种情况。

5.1.3 基于用户关键字的推荐结果

上面提及了用户关键字的统计方法，关键字来源于四个域，这四个域的权重各不相同，keywords 标签、新闻标题这两者的权重应该高一些，而 description 标签、新闻正文这两者的权重相对低些。

基于关键字的推荐结果，需要为新闻建立倒排索引，根据用户历史数据中的关键字，计算出每条新闻相对当前用户的一个权重值。当前权重越高，则表示预测当前用户越有

可能喜欢这条新闻。

5.2 测试环境

操作系统：CentOS release 6.5 (Final)

网络环境：100M 局域网

Hadoop 集群硬件配置：3 台联想电脑，内存 8G，CPU Corei5-3470 3.2GHZ，磁盘是 7200 转 机械硬盘。

Storm 集群硬件配置：在同一台机器上，由 VirtualBox 虚拟出三台机器，CPU 单核，内存 2G。

Hadoop 版本：hadoop 2.2.0

HBase 版本：hbase 0.98.3

Redis 版本：Redis 3.0

Storm 版本：storm 0.9.4

5.3 评价标准

对于结果的评价标准就是 F 值计算方法。

$$\mathrm{F} = \frac{1}{\frac{1}{preceison} + \frac{1}{recall}} \tag{5-1}$$

其中 *precision* 是指准确率，*recall* 是指召回率。
准确率的计算公式如下：

$$precision = \frac{\sum_{u_i \in U} hit(u_i)}{\sum_{u_i \in U} L(u_i)} \tag{5-2}$$

公式中的 U 是指所有用户的集合，在测试过程中，U 的大小就是一万个用户，$L(u_i)$表示系统推荐给用户u_i的推荐列表的长度，$hit(u_i)$是指在推荐列表中，用户u_i实际浏览的新闻数量。

召回率的计算公式如下：

$$recall = \frac{\sum_{u_i \in U} hit(u_i)}{\sum_{u_i \in U} T(u_i)} \tag{5-3}$$

U 是所有用户集合，$T(u_i)$ 是用户实际浏览新闻的数量。与准确率计算公式中的一样，$hit(u_i)$是指实际浏览的新闻中出现在新闻推荐列表中的数量。

5.4 测试结果

5.4.1 系统计算效率的实验

1) MapReduce 计算时间

利用 MapReduce 分别计算不同规模的数据集，分析其计算时间与数据集大小之间的关系，分别测试了 10,000 到 110,000 条数据集规模的等级上，MapReduce 与 Mahout 推荐算法融合的。每种数据测试 10 次，最后的计算时长平均值作为测试结果。

Hadoop 环境是两台机器是 DataNode，一台机器是 NameNode。最终的计算时间变化如图 5-1 所示，横坐标是数据集的大小，竖坐标是计算时长。

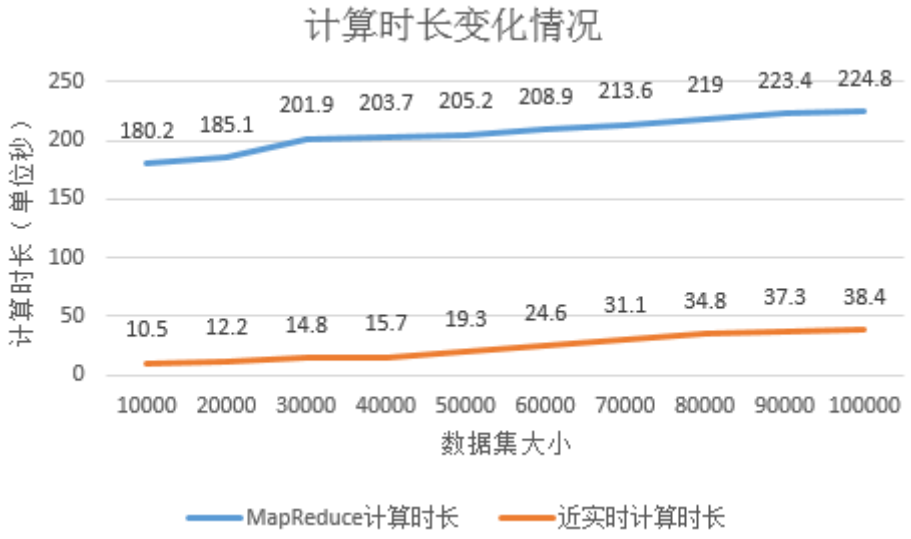

图 5-1 数据规模与计算时长之间的关系图

除了上图的数据集，还测试了 500,000 数据集，计算时间是 275 秒，1,000,000 数据集，计算时间是 386 秒。

从 MapReduce 的实验数据可知，在数据规模在百万以内，随着数据规模的增长，计算时长几乎是线性增加。鉴于只有三台机器组成的 Hadoop 集群就有这样的计算能力，从而推测 通过增加集群规模，可以有效地应对数据集的增长。

2) 近实时计算时间

为了与上面 MapReduce 离线计算方式相对比，再做了近实时计算的计算时长与数据规模的关系。

近实时计算包括根据过去一段用户浏览记录，对用户做聚类；之后再根据聚类做协同过滤算法，得到推荐结果；同时与离线计算结果、以前的近实时计算相融合。

近实时计算的数据规模与计算时长之间的关系与 K-means 聚类算法的 K 值相关。

以 K 值 1000 为例。近实时计算过程包括聚类算法、协同过滤推荐。

数据规模与计算时长之前的关系如图 5-1 所示，它与 MapReduce 相比，它的计算时长明显更短。

3) 系统响应速度

推荐系统的响应速度，是推荐系统中重要考量指标。在本文设计的系统中，当服务器得到一个用户请求，就从 Redis 中读取 Redis 中的推荐结果，返回给请求。

Redis 作为内存型 NoSQL 数据库，它的响应速度很快，能够同时上万的访问请求。具体的响应速度，与发起的请求有关，如果客户端的请求越多，那么实时响应速度相应会慢点。测试数据如下图 5-2 所示，横坐标客户端的请求量，竖坐标系统响应时间。

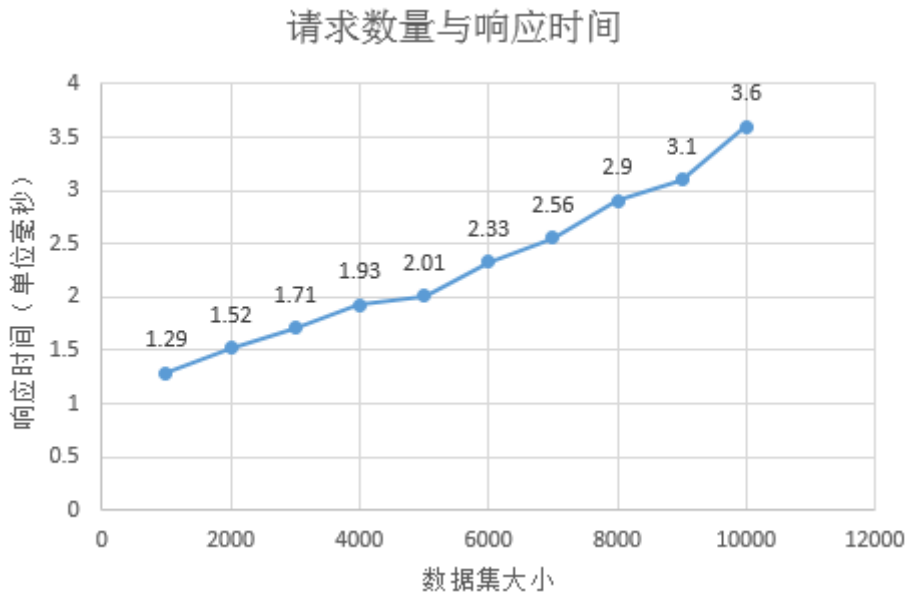

图 5-2 客户端请求数量与系统响应时间的关系

从图上可以看出，通过 redis 作为数据库响应请求，其响应速度能够达到毫秒级别。经过测试，发现即使当用户请求达到 20,000 时，响应速度还能保持在 4ms 左右。Storm 结合 Redis 数据库可以很好地适合实时推荐场景。

5.4.2 推荐算法的相关实验

1) 新闻发表时间与新闻浏览时间的关系

根据 CCF 竞赛数据集的浏览记录，统计新闻发布时间与用户浏览新闻时间的分布情况，CCF 数据集中，用户的浏览记录的时间区间大致落在是 2014 年 3 月 1 日~3 月 31 日的时间区间内。本文统计了在 2014 年 3 月 16 前发表的新闻，计算这些新闻在半个月内的用户浏览时间情况，统计的浏览记录有 67671 条，计算的结果是：在一天之内新闻

被浏览的记录有 32868 条，两天之内的有 11271，三天之内的有 5567，之后的数据依次是 2916，2712，2636，2709，2334，1054。第九天之后的浏览是就很少了，之后还会持续减少。根据这些数据画出的直方图如下所示：

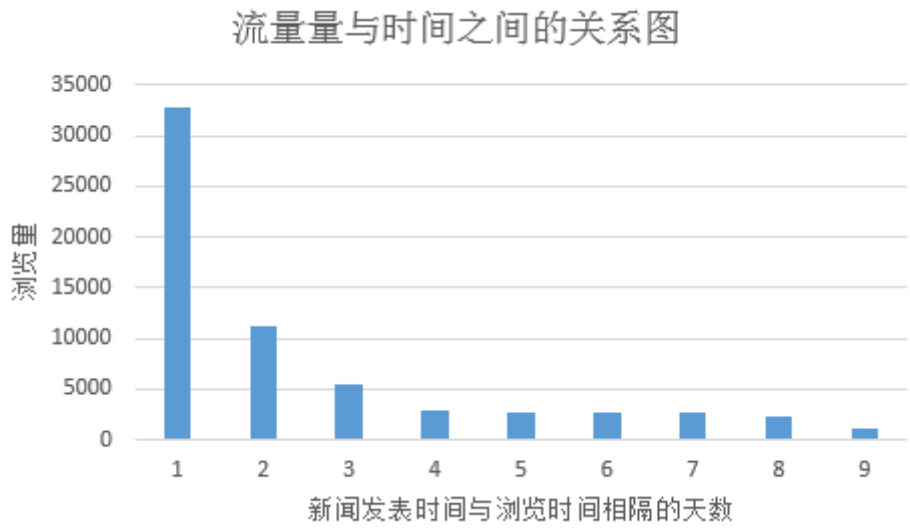

图 5- 3 新闻发布时间与浏览时间的关系

通过上图，可以得出一个结论：一条新闻有三天的活跃度，之后它的浏览量会持续衰减，在 10 天之后，它的浏览量就很少了。

基于这个结论，可以大大提高推荐的准确率。在给用户做推荐时，它的推荐集中，增加三天内的新闻权重，适当提高 10 天内新闻的权重，减少 10 天之前的新闻集。

2) 相似用户对推荐的贡献值

系统随机统计了 6 组用户浏览日志数据，每组中的浏览日志有 1000 条。相似用户预测当前用户浏览闲聊的贡献值的计算公式是：

$$r(u_i, n_k) = \sum_{u_j \in S_{u_i}} similar(u_i, u_j) * I(u_j, n_k) \tag{5-4}$$

式中$r(u_i, n_k)$预测用户u_i对新闻条目n_k的预测值，

S_{u_i}是当前用户u_i的相似用户的集合，$similar(u_i, u_i)$是用户u_i和用户u_j的相似度。如果用户u_i浏览了新闻n_k则$I(u_j, n_k)$的值为 1，否则为 0。

$r(u_i, n_k)$最终的值代表相似对预测当前用户浏览新闻的贡献值，根据数据最终的统计结果如图 5-2 所示。

其中相似度贡献值落在 0~5 这个区间的数量有 2606 条，达到 43.43%；而落在 5~10 这个区间的值是 2102 条，达到 35%；

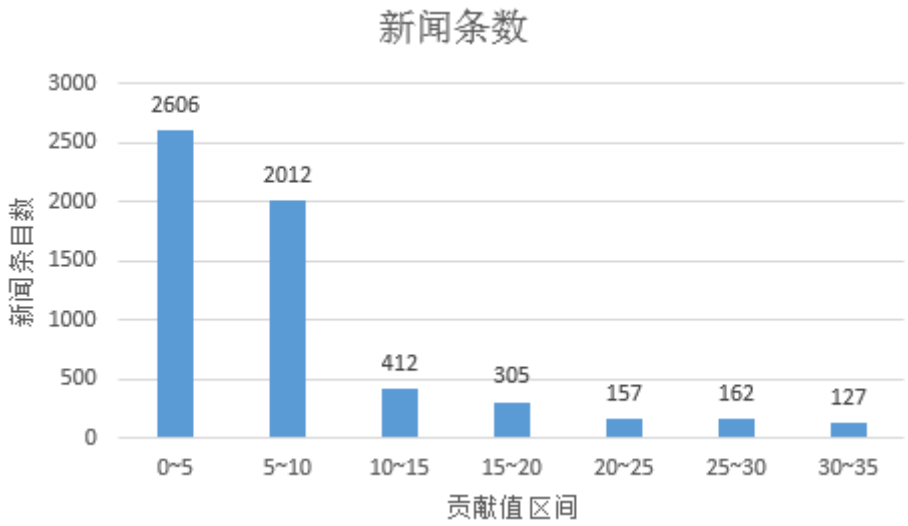

图 5-4 相似用户的贡献值

3) 基于协同过滤推荐算法的实验

基于协同过滤推荐算法的实验，包括基于用户的协同过滤推荐和基于新闻条目的协同过滤推荐结果。

Goto : /zhbs go

Go to parent directory

Name	Type	Size	Replication	Block Size	Modification Time	Permission	Owner	Group
SimilarUser	dir				2015-04-27 04:49	rwxr-xr-x	zhang	supergroup
itembased	dir				2015-04-26 15:58	rwxr-xr-x	zhang	supergroup
userbased	dir				2015-04-26 13:30	rwxr-xr-x	wu	supergroup

图 5-5 HDFS 文件目录

为了加快实验速度，对数据都做预处理，事先利用 mahout 为所有用户做推荐，而推荐结果会保存在 HDFS 文件上。HDFS 上计算结果目录如图 5-5 所示。

similarUser 文件夹下保存的是相似用户数据，它的目录中会为每个用户创建一个文件，文件名就是用户编号，文件内容是该用户的所有相似用户，以及他们之间的相似度数值。

itembased 目录下存放的是基于新闻条目的推荐结果。根据前面 30 天的数据预测用户之后浏览的新闻。

userbased 目录下存放就是基于用户协同过滤推荐的结果。用户编号就是文件名，文件内容是一个列表，存放推荐集合，以及推荐的权重值。

✧ 最初级的协同过滤算法，得出的 F 值只有 0.881%。

✧ 从协同过滤推荐项中，过滤掉推荐值小于 0.7 的推荐项后，得出的准确率是

0.9803%，召回率是 0.985%，F 值是 0.9826%。

✧ 将推荐值与新闻发表时间相关联。

协同过滤推荐项的推荐值乘以一个与时间相关的系数，这个系数是从根据图“新闻发布时间与浏览时间的关系”中的统计结果而来，但是它不是以天以单位，而是以小时为单位，这样的控制粒度更精确。最终的推荐集合中，会过滤掉推荐值小于 0.3 的推荐项。

最终它的推荐准确率是 1.189%，召回率是 1.137%，F 值是 1.312%。

表 5-1 推荐算法结果值

算法	准确率	召回率	F 值
初级协同过滤算法	0.8751%	0.8836%	0.87932%
协同过滤算法（新闻发表时间因子）	1.189%	1.137%	1.312%
基于内容推荐（新闻关键字）	1.1794%	1.2013%.	1.19025%
基于内容推荐（新闻关键字、新闻类别）	1.21%	1.235%	1.2224%
混合基于内容，协同过滤算法	2.0407%	2.1%	2.0699%
混合基于内容，协同过滤算法(加入热点新闻)	2.167%	2.181%	2.1739776%

4) 基于内容的推荐算法实验

推荐内容的推荐算法是从几个方面：新闻类别、新闻关键字。

新闻关键字作为新闻内容一个最重要的特征。关键字能够很好地体现和概括新闻的内容。

本文对 CCF 数据集的属性做了扩充，通过爬虫获取了新闻页面对应的 URL，从中得到了新闻类别，title 标签，keywords 标签，description 标签。从而大大地扩充了新闻

关键字内容。

每条的新闻的关键字的来源有四个：新闻正文、title 标签、keywords 标签、description 标签。

同时，新闻类别作为新闻内容的一个重要特征，通过分析用户历史记录，从中统计出用户在每种新闻类别上的浏览量，将各个新闻类别浏览量的比例作为用户对每种新闻类别的权重偏好。

具体的推荐将是如表 5-1 所示，基于内容的推荐效果远没有基于协同过滤的效果好，原因分析是数据集的数据规模不够大，Google News 论文中的基于内容的推荐是利用用户过去一年的浏览记录做分析，从这种大规模的数据集得出的关键字才能真正反映用户的关键字喜好和新闻类别倾向。

5) 混合推荐算法的实验

最后的实验是是混合的基于内容推荐、基于协同过滤的推荐结果。推荐效果如表 5-1 所示。

在混合推荐时，还加入了热点新闻，由于数据太过稀疏。同一地区的用户浏览的热点大致相同，所以加入了热点新闻，通过实验发现。加入热点新闻确实能在一定程度上提高推荐效果。

6) 基于内容和主题特征的个性化新闻推荐算法[40]

文献[40]中所提出的基于内容和主题特征的个性化推荐算法，是利用关键字和主题为用户建立兴趣模型，然后将待推荐新闻库中与用兴趣模型进行匹配，计算得出用户的推荐列表。

该算法是有监督的学习算法，前期需要将每篇新闻解析出多个主题，每篇新闻用主题向量进行表示。

7) 基于协同过滤的个性化新闻推荐算法[41]

上面测试了基于内容的推荐算法，再次测试基于协同过滤的推荐算法，这个算法利用了隐式的用户行为数据，再将隐式数据转化为显式数据，为用户计算相似度；同时结合了新闻类别、新闻内容信息，提出了行为-内容相似计算方法，在一定程度上解决了数据稀疏性问题。

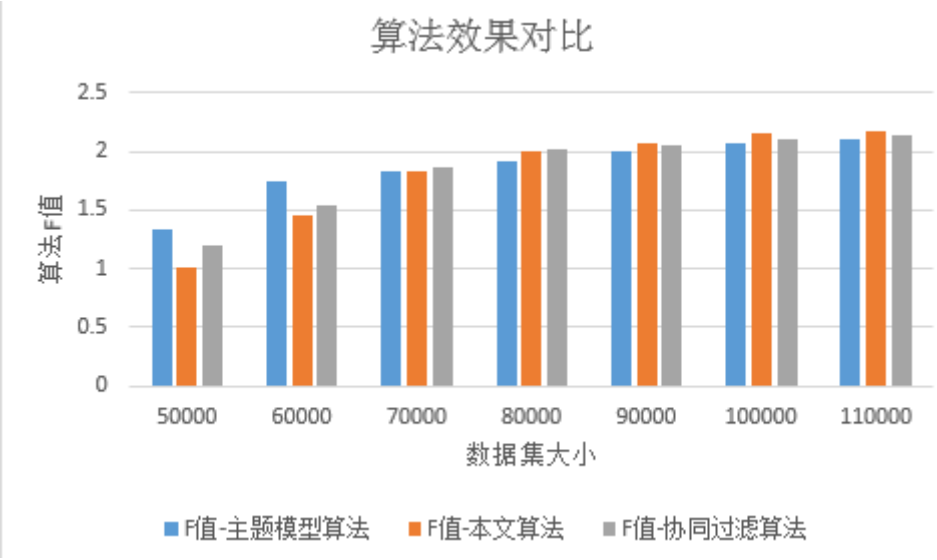

图 5-6 文献[40]和[41]算法与本文算法计算结果对比

从图 5-6 可知，主题模型算法[40]在数据集上的测试结果表明：

✧ 随着数据集的增加，算法的 F 值也会随之分增长。

✧ 在数据集规模在 7000 以内，“主题特征算法”的结果优于本文设计的算法，但是当数据集规模大于 7000 之后，本文设计的算法更有优势。

基于协同过滤的个性化新闻推荐算法[41]在数据集上的测试结果表明：

✧ 在系统前期它的效果是高于本文设计的算法，同时低于基于主题模型的推荐算法。

✧ 随着库数据的增加，算法发挥了基于协同过滤算法的优势，推荐效果越来越好，与本文设计的算法十分接近。

总的来说，与文献[40]和[41]中算法相比，本文的算法在系统冷启动时期，算法的效果一般，但是随着用户数据的积累，算法的效果会越来越好。

5.4.3 实验结果分析

1) 系统计算效率

作为大数据的新闻推荐系统，系统的计算效率是一项重要的性能指标。借助于当前的 MapReduce 大数据处理工具配合 storm 的实时计算能力，以及 Redis 内存型数据库，能够缓解计算压力，为用户及时提供推荐结果。

MapReduce 能够在 380 秒左右的时间内处理百万级别的数据集，计算每个用户的推荐结果。Redis 数据库能够在 4 毫秒响应 20,000 个用户请求。这种处理能力的响应速度能够应对大数据系统中计算需求。

2) 推荐算法

除了上面的计算效率，推荐算法是推荐系统的核心，算法的准确率和召回率作为算法评价标准。

本文设计的算法还有自身局限，采用的是基于协同过滤和基于内容的混合推荐算法，它有几个不足之处：

a) 没有解决用户冷启动问题：

在新用户进入系统时；或者系统初期，没有大量的用户浏览记录。这两种情况下，都不能为用户提供推荐集合。只能引导用户手动输入自己感兴趣的关键字，系统再针对这些关键字提供推荐结果。

b) 数据稀疏性问题

在协同过滤推荐过程当中，用户的浏览的新闻条目对于整个新闻库来说，是非常稀疏的，数据太过稀疏会影响相似用户的计算，以及后续的推荐结果。

同时，与其它推荐算法相似，本文提出的混合推荐算法也有自身的优点：

a) 协同过滤算法会针对系统中用户浏览行为做参数调整，使算法能够更好地适用用户浏览行为模式。

b) 基于内容的推荐算法是利用新闻内容，也就是新闻中分词结果。这个推荐方式，能够从新闻内容出挖掘出用户感兴趣的关键字，同时，它的算法复杂度低，与用户数据成线性增长。

c) 算法与时间相关，能够动态跟踪用户的兴趣变化。越久的浏览记录，它的权重会衰减，而近期浏览记录权重会相应增加。

d) 随着用户浏览数据的积累，相似用户的计算会更加准确，协同过滤算法的效果也更好；同时根据用户浏览记录得到的关键字也会更多，关键字的权重更加能反应用户真实的喜好。

第六章 总结与展望

6.1 本文总结

互联网的飞速发展，以及庞大的网民数量，海量的信息资源。整个互联网自然地进入了现在的大数据时代。同时，随着计算机硬件也是高速发展，摩尔定律见证了 50 年的 CPU 的发展。

在这样的背景下，Google 发表一篇用 MapReduce 方式，利用廉价机器组成一个庞大的集群，以达到对大数据的处理能力，此后各种关于大数据和开源项目如雨后春笋般涌现。

本文主要介绍了一种基于 storm 的推荐系统，依托目前的大数据开源工具，利用几台普通机器搭建了一个基于 storm 的新闻推荐系统。

本文的主要贡献在于针对新闻时效性的特点，提出了一种实时推荐系统的设计方案。在系统设计过程中利用一些现有的开源项目，大大降低了系统开发难度，借鉴了 Google News 的新闻个性化推荐系统的，不仅在实现了基于用户的协同过滤、基于新闻条目的协同过滤，还利用爬虫扩充了数据源，得到了新闻类别、新闻 keywords 标签、新闻 description 标签。通过对这些字段的充分挖掘，可以得出用户更可靠的关键字。从而将传统的协同过滤和的基于内容（关键字）的推荐相结合，从而得出一个更加有效的推荐结果。

6.2 研究展望

虽然本文中介绍了如何实现一个新闻的个性化推荐系统，但是还有一些地方可以改进的。

1) 关键字的调优。

关键字作为基于内容推荐中的数据单元，它好坏直接决定了基于用户关键字的效果如何。例如在测试的数据集，部分新闻报导了 2014 年 3 月份的马航失事事件，“马航”在 ICTCLAS 分词中并不是一个关键字。所以需要人为地为分词系统导入一些关键字。由于互联网的发展，以及社交网络的崛起，网上会不断出一些新的词汇，为了跟随用户的趋势，关键字模块需要长期维护关键字。

2) 利用新闻的浏览时长

开源的数据之中，没有这一个元素，在正常的浏览行为，如果一个用户在一篇文章上花费的阅读时间越长，则认为该用户对这篇新闻更加感兴趣，那么这条

新闻的关键字权重也就相应地增加。

3) 突发热点事件新闻的推荐

在浏览新闻门户网站时，遇到突发事件时，网站都是通过系统中的强制提醒的方式告知用户的，这种方式优势是够快，够及时。在传统的推荐算法中，对于热点新闻的反应速度不够理想，不能对突发事件及时响应。这方面的工作也会一个个性化新闻推荐系统的突破点。

参考文献

[1] Ghemawat, Sanjay, Howard Gobioff, and Shun-Tak Leung. The Google file system[J]. ACM SIGOPS operating systems review, 2003, 37(5):29-43.

[2] Dean, Jeffrey, and Sanjay Ghemawat. MapReduce: a flexible data processing tool[J]. Communications of the ACM, 2010, 53(1):72-77.

[3] Chang F, Dean J, Ghemawat S, et al. Bigtable: A distributed storage system for structured data[C]. Proceedings of Usenix Symposium on Operating Systems Design & Implementation. 2006:205--218.

[4] 王继成, 萧嵘, 孙正兴, 等. Web 信息检索研究进展[J]. 计算机研究与发展, 2001, 38(2): 187-193.

[5] Resnick P, Iacovou N, Suchak M, et al. GroupLens: an open architecture for collaborative filtering of netnews[C]. Proceedings of the 1994 ACM conference on Computer supported cooperative work. ACM, 1994:175-186.

[6] Sarwar B, Karypis G, Konstan J, et al. Item-based collaborative filtering recommendation algorithms[C]. Proceedings of the 10th international conference on World Wide Web. ACM, 2001:285-295.

[7] Linden, Greg, Brent Smith, and Jeremy York. Amazon.com recommendations: Item-to-item collaborative filtering []. Internet Computing, IEEE 7.1 2003: 76-80.

[8] Fu, Xiaobin, Jay Budzik, Kristian J. Hammond. Mining navigation history for recommendation[C]. Proceedings of the 5th international conference on Intelligent user interfaces. ACM, 2000.

[9] Das A S, Datar M, Garg A, et al. Google news personalization: scalable online collaborative filtering[C]. Proceedings of the 16th international conference on World Wide Web. ACM, 2007:271-280.

[10] Liu J, Dolan P, Pedersen E R. Personalized news recommendation based on click behavior[C]. Proceedings of the 15th international conference on Intelligent user interfaces. ACM, 2010:31-40.

[11] 李杰, 徐勇, 王云峰, 朱昭贤. 面向个性化推荐的强关联规则挖掘[J]. 系统工程理论与实践, 2009, (8):144-152.

[12] Wikipedia.org. Google News - Wikipedia, the free encyclopedia. http://en.wikipedia.org/wiki/Google_News

[13] Cohen E. Size-estimation framework with applications to transitive closure and reachability[J]. Journal of Computer and System Sciences, 1997, 55 (3): 441-453.

[14] Wikipedia.org. Jaccard index- Wikipedia, the free encyclopedia. http: // en. Wiki pedia.org/wiki/Jaccard_index

[15] Billsus D, Pazzani M J. A Hybrid User Model for News Story Classificat ion[J]. Cism International Centre for Mechanical Sciences, 1999:99--108.

[16] Jensen F V, Nielsen T D. Bayesian networks and decision graphs[J]. 199 2.

[17] Apache.Storm, distributed and fault-tolerant realtime computation. https://sto rm.apache.org/

[18] Hadoop, Apache. "Hadoop." 2009-03-06]. http://hadoop. apache. org (2009).

[19] HBase, Apache. "Apache HBase." (2013).

[20] Sanfilippo, Salvatore, and Pieter Noordhuis. "Redis." 2011-03-19. http://redi s.io (2010).

[21] Luhn, Hans Peter. Selective dissemination of new scientific information wi th the aid of electronic processing equipment. American Documentation 12. 2 (1961):1 31-138.

[22] Spärck Jones, Karen. IDF term weighting and IR research lessons[J]. Journ al of Documentation 60.5. 2004: 521-523.

[23] Robertson, Stephen. Understanding inverse document frequency: on theoreti cal arguments for IDF[J]. Journal of documentation, 2004, 60(5):503-520.

[24] Białynicki-Birula I, Mycielski J. Uncertainty relations for information entro py in wave mechanics[J]. Communications in Mathematical Physics, 1975, 44(2): 129-132.

[25] Wang F H, Shao H M. Effective personalized recommendation based on t ime-framed navigation clustering and association mining[J]. Expert Systems with Applications, 2004, 27(3): 365-377.

[26] Lin W, Alvarez S A, Ruiz C. Collaborative recommendation via adaptive association rule mining[J]. Data Mining and Knowledge Discovery, 2000, 6: 83-105.

[27] 百度百科. 啤酒与尿布_百度百科, http://baike.baidu.com/view/ 19782 39.h tm

[28] Fukunaga K, Narendra P M. A branch and bound algorithm for computin g k-nearest neighbors[J]. Computers, IEEE Transactions on, 1975, 100(7): 750-753.

[29] wikepedia.org. Euclidean distance - Wikipedia, the free encyclopedia http:// en.wikipedia.org/wiki/Euclidean_distance

[30] Wikipedia.org, Cosine similarity - Wikipedia, the free encyclopedia. http://e n.wikipedia.org/wiki/Cosine_similarity

[31] Wikipedia.org. Pearson product-moment correlation coefficient - Wikipedia, the free encyclopedia http://en.wikipedia.org/wiki/Pearson_product-moment_ correlation_coefficient

[32] Sarwar B, Karypis G, Konstan J, et al. Item-based collaborative filtering r ecom- mendation algorithms[C]. Proc International Conference on the Worl d Wide Web. ACM, 2001:285--295.

[33] Pazzani M J, Billsus D. Content-Based Recommendation Systems[J]. Lectu re Notes in Computer Science, 2007:325--341.

[34] 王国霞, 刘贺平. 个性化推荐系统综述[J]. 计算机工程与应用, 2012, 48(7): 66-76. DOI:10.3778/j.issn.1002-8331.2012.07.018.

[35] Mahout, Apache. Scalable machine learning and data mining. 2011-11-6(20 12).

[36] 张华平, 刘群. ICTCLAS 汉语分词系统 [EB/OL]. Diss. 2008.

[37] Rish I. An empirical study of the naive Bayes classifier[C]. Proc Icpr, 20 01, 1(2):17-20.

[38] Reese, Will. "Nginx: the high-performance web server and reverse proxy." Linux Journal 2008.173 (2008): 2.

[39] CCF, CCF 大数据竞赛, 2014-09-01. http://115.28.182.124/c/00000000050 (20 14).

[40] 刘金亮. 基于主题模型的个性化新闻推荐系统的研究与实现[D]. 北京邮电大学, 2013.

[41] 曹一鸣. 基于协同过滤的个性化新闻推荐系统的研究与实现[D]. 北京邮电大学, 2013.

攻读硕士学位期间取得的研究成果

一、已发表（包括已接受待发表）的论文，以及已投稿、或已成文打算投稿、或拟成文投稿的论文情况（只填写与学位论文内容相关的部分）：

序号	作者（全体作者，按顺序排列）	题目	发表或投稿刊物名称、级别	发表的卷期、年月、页码	相当于学位论文的哪一部分（章、节）	索引收录情况

注：在“发表的卷期、年月、页码”栏：

1 如果论文已发表，请填写发表的卷期、年月、页码；

2 如果论文已被接受，填写将要发表的卷期、年月；

3 以上都不是，请据实填写“已投稿”，“拟投稿”。

不够请另加页。

二、与学位内容相关的其它成果（包括专利、著作、获奖项目等）

致谢

在研究生阶段进入尾声之时，回想起硕士三年期间的点滴，思绪万千。在此衷心感谢我的家人、导师、实验室的师兄弟们。

是家人的关怀无时无刻不在温暖着我，也是我前进的最大动力。无论悲喜，都有家人的温暖的港湾。

导师作为师长，教我知识，教我做人。给过我长辈般的指导，兄长般的关爱。其严谨的治学态度、认真负责的为人作风、渊博的知识，都给我留下了深刻的印象，让我受益匪浅，感受良多，甚至受益终身。

其次是感激实验室的师兄弟，与他们交流扩展我的知识范围，开拓我的视野。在实验室和我一起解决项目、学术问题。和他们一起参加业余活动也丰富了我的生活，让我的身体得到锻炼，也在其中获取了欢声笑语，以及大家的友谊。

最后感谢我的三位舍友，和他们的生活也是乐在其中，从他们身上学会与人相处。与他们的交流中，也了 解更多的知识，包括专业的、非专业的，拓展了我的视野。

Ⅳ－2答辩委员会对论文的评定意见

张慧同学的硕士论文“基于storm新闻推荐系统的研究与实现”有较好的理论意义和应用价值。

论文针对新闻时效性的特点，推出了一种实时推荐系统的设计方案，实现了基于用户过滤，基于新闻条目的协同过滤，利用了爬虫扩充了数据源，得到了新闻类别、新闻keywords标签，新闻description标签。

作者论点正确，结论合理。反映作者具有一定的理论基础、较好的科研和工程能力。

论文结构清晰，图文表达规范，达到硕士学位论文水平。答辩过程讲述较清楚，回答问题基本正确。经答辩委员会无记名投票，同意该同学通过硕士论文答辩，同意授予硕士学位。

论文答辩日期：2015 年 6 月 5 日

答辩委员会委员共 5 人，到会委员 5 人

表决票数：优秀（ ）票；良好（ 3 ）票；及格（ 2 ）票；不及格（ ）票

表决结果（打“√”）：优秀（ ）；良好（√）；及格（ ）；不及格（ ）

决议：同意授予硕士学位（√）　　不同意授予硕士学位（ ）

答辩委员会成员签名

（主席）