计算机研究与发展
Journal of Computer Research and Development
DOI:10.7544/issn1000-1239.2017.20170128
54(8):1631-1643,2017

播存网络环境下 UCL 推荐多样性优化算法

顾 梁 杨 鹏 董永强
(东南大学计算机科学与工程学院 南京 211189)
(计算机网络和信息集成教育部重点实验室(东南大学) 南京 211189)
(guliang@seu.edu.cn)

A Diversified Recommendation Method for UCL in Broadcast-Storage Network

Gu Liang, Yang Peng, and Dong Yongqiang
(*School of Computer Science and Engineering, Southeast University, Nanjing* 211189)
(*Key Laboratory of Computer Network and Information Integration (Southeast University), Ministry of Education, Nanjing* 211189)

Abstract By introducing broadcast distribution into TCP/IP, Broadcast-Storage network has clear advantages in reducing the redundant traffic in the Internet and remitting information overload problem. Uniform content label (UCL) is used to express the needs of users and help users obtain the information resources in Broadcast-Storage network. In the process of UCL recommendation, one key problem that needs to be solved is that how to improve the diversity of recommendation based on the features of Broadcast-Storage network, e.g., rich semantic information and high novelty. To solve this problem, this paper proposes a diversification method UDSCT for UCL recommendation based on semantic cover tree. UDSCT consists of two components. The first one is constructing the semantic cover tree for UCLs, which obeys some proposed invariants and considers the semantic information of UCL and the ratings from users. Besides that, new UCLs are given priority to improve the novelty of the whole UCL list. The second component is the query of diversified UCL list, which uses simple tree query and heuristic list supplement operation to obtain the diversified UCL list fast and returns specified UCL sets rapidly according to users' need. Theoretical analysis and a series of experiments results show that, UDSCT outperforms some benchmark algorithms and is suitable for Broadcast-Storage network.

Key words Broadcast-Storage network; uniform content label; recommendation; diversity; novelty

摘 要 播存网络将广播分发模式引入现有互联网体系结构,极大地降低网络共享过程中产生的冗余流量,可有效缓解信息过载问题.播存网络采用统一内容标签(uniform content label, UCL)适配用户兴趣和推荐信息资源,在 UCL 个性化推荐过程中,如何结合播存网络的富语义、高时效特征,有效地提高

收稿日期:2017-03-10;**修回日期**:2017-05-15
基金项目:国家自然科学基金项目(61472080,61672155);中国工程院咨询研究项目(2015-XY-04);国家"八六三"高技术研究发展计划基金项目(2013AA013503);软件新技术与产业化协同创新中心项目
This work was supported by the National Natural Science Foundation of China (61472080, 61672155), the Consulting Project of Chinese Academy of Engineering (2015-XY-04), the National High Technology Research and Development Program (863 Program) of China (2013AA013503), and the Project Funded by the Collaborative Innovation Center of Novel Software Technology and Industrialization.
通信作者:杨鹏(pengyang@seu.edu.cn)

UCL 推荐列表的多样性,成为播存网络中一个亟需解决的关键问题.针对播存网络环境的需求,提出了一种基于语义覆盖树的 UCL 推荐多样性优化算法 UDSCT,将该问题分为 UCL 语义覆盖树构建和多样化 UCL 列表查询 2 个步骤.在 UCL 语义覆盖树构建阶段,基于语义覆盖树的若干约束条件,充分考虑 UCL 语义信息及非语义用户评分信息,同时,较新的 UCL 具有较高的优先权,以保证列表的时效性;在多样化 UCL 列表查询阶段,采用简单树查询及启发式列表补充操作,可快速高效地获得多样性优化后的 UCL 推荐列表,并可进一步根据用户请求快速返回指定的 UCL 集合.通过理论分析及一系列仿真实验验证,结果证明:UDSCT 算法相对于基准算法能够获得更好的多样性优化效果及效率,可有效满足播存网络环境的需求.

关键词 播存网络;统一内容标签;推荐;多样性;时效性

中图法分类号 TP393;TP18

互联网在给人们带来丰富信息资源的同时,自身流量也与日俱增.造成互联网协议流量激增的主要原因之一是互联网的内容访问日益呈现出幂律分布的特性,即大量用户对热门资源的重复性访问耗费了大量的数据传输带宽.现有互联网架构和内容传输技术体系越来越难以为高效信息共享提供有效的支持.

为改变这种局面,播存网络(Broadcast-Storage network, BSN)[1-2]把信息资源映射为互联网中可拆可聚的活版基元,将广播分发模式引入现有互联网体系结构,采用以"广播+存储"为特征的信息共享方式,通过物理广播将热门信息资源直接辐射分发到用户终端附近的边缘服务器供用户访问,从而利用物理广播的天然优势,突破传统信息共享方式所面临的宽带资源制约,实现"共享不限人数"的新型高效信息共享途径,极大地降低网络中共享过程中产生的冗余流量,从而有效缓解"信息过载"问题.

播存网络模型如图 1 所示[3].其中,统一内容标签(uniform content label, UCL)[4]与信息资源一一对应,包含信息资源的摘要信息,用户通过其接收到的 UCL 判断是否需要访问信息资源全文.同时,由

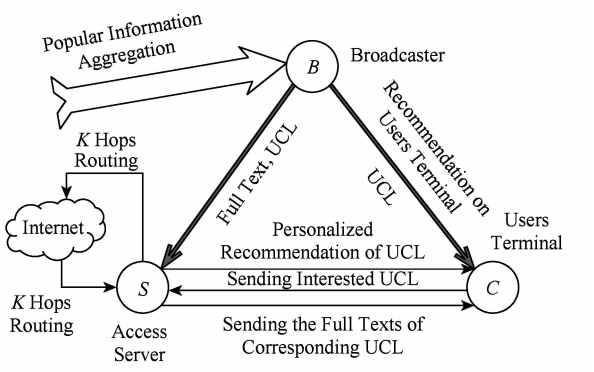


Fig. 1 The model of Broadcast-Storage network[3]

图 1 播存网络模型[3]

于 UCL 数量巨大而用户接收能力有限,播存网络根据用户特点建立用户兴趣模型,预测用户对所有 UCL 的感兴趣程度(量化为预测评分),并采用主动推荐的方式将感兴趣程度较高的 UCL 推送给用户,帮助用户高效精确地寻找及访问其偏好的信息资源.而由于 UCL 种类繁杂、数量巨大,在推荐过程中,若仅依据用户对 UCL 的预测评分,容易造成所推荐的 UCL 相似度过高、种类单一,难以全面地挖掘播存网络中用户的兴趣.因此,如何在 UCL 个性化推荐过程中,在保证精度的基础上,尽可能地提高所推荐 UCL 的类别多样性,成为播存网络中亟需解决的关键问题.

本质上而言,该问题为信息获取领域的结果集合多样性优化问题,并已被证明为 NP 难问题.目前已有的解决方法多为启发式算法,可大致分为 2 条思路:置换启发式(swap heuristics, SH)[5]和贪婪启发式(greedy heuristics, GH)[6].假设最终的多样化集合包含 k 个元素,前者首先从所有待选元素中随机选出 k 个元素,然后遍历所有剩余的待选元素,逐一与已选出的元素进行替换比较,从而优化最终的集合多样性;后者采用 k 步筛选,每个筛选过程增加一个元素至集合,并保证当前的集合多样性最高,直至集合中包含 k 个元素.

然而,在播存网络环境中,传统结果集合多样性优化具有自身的一些局限性,具体表现在 3 个方面:

1) 难以有效确定加权参数.传统多样性优化方法往往需要设定加权参数以在精度及多样性之间取得平衡.通常,数据集的特征对加权参数的确定有重要影响.而在播存网络环境中,由于 UCL 分发较为频繁,数据集不断变化,很难确定一个恰当的加权参数能够持续保证 UCL 推荐的多样性及精度.

2) 缺乏 UCL 时效性处理机制.播存网络环境中

UCL 包含信息资源的摘要信息,与信息资源一一对应,更新速度快,具有很强的时效性.传统多样性优化方法很少考虑优化对象的时效性,容易导致优化后的集合中包含比较陈旧的 UCL,降低用户体验.

3) 无法快速产生优化结果. UCL 数目巨大,在播存网络持续动态运行时,如何基于大量的 UCL 数据快速及时地为用户提供多样性优化结果尤为重要.传统多样性优化方法通常需要大量的迭代运算,收敛较为缓慢,无法高效满足播存网络的需求.

针对上述 3 个问题,本文基于语义覆盖树提出了一种适用于播存网络环境的 UCL 推荐多样性优化算法(UCL diversification based on semantic cover tree, UDSCT),能够克服传统多样性优化方法在播存网络环境中存在的不足,并可充分利用 UCL 语义信息及非语义用户评分信息,在保证 UCL 推荐精度的前提下,优化 UCL 推荐结果的多样性.

1　相关工作

目前,播存网络的相关研究主要围绕互联网访问特征、播存网络的基本架构、UCL 的格式设计与标引及播存网络环境中的 UCL 个性化推荐机制等几个方面展开.

文献[7]提出用户对 Web 网页的访问是由用户需求行为确定的一个随时间演化的复杂双模式二分网络,并通过实证研究分析发现,其入度分布呈现出典型的无标度特征和集聚现象. 文献[8]对互联网中用户访问 Web 网页时所产生的路由跳数进行了研究与测量,得到每条访问记录所需的平均路由跳数为 13.11. 文献[7-8]为播存网络模型提供了理论基础与数据支撑.

文献[4]对播存网络的原理与基本模型进行了系统详细的阐述. 文献[2]进一步对播存网络模型进行了研究,给出了播存网络的普适模型及其数学描述,并重点研究了 4 种实现模式,为播存网络提供了严格的定义及规范.播存网络中,用户通过 UCL 判断是否需要某条信息资源. 文献[3]针对播存网络的特点,提出一种播存网络环境下的 UCL 推荐方法,可根据用户的历史访问记录建立用户特征模型,精确预测用户可能感兴趣的 UCL. 然而,文献[3]未考虑 UCL 推荐列表的多样性,本文在文献[3]的基础上重点研究 UCL 推荐的多样性优化问题.

结果集合多样性优化问题已在信息获取领域得到广泛关注,研究人员已经证明该问题为 NP 难问题,并提出了多种启发式解决方法. Adomavicius 等人在文献[9]中指出,采用启发式算法改变初始推荐列表中项目的排序可使得推荐性能,尤其是多样性得到很大提高,并且启发式优化算法在逻辑上独立于预测算法,算法可部署性较好. 根据算法设计思路,现有基于启发式思想的推荐列表多样性优化研究主要可以分为 2 条思路:置换启发式(swap heuristics, SH)和贪婪启发式(greedy heuristics, GH).

GH 方法根据选定的效用函数依次从总集合中选出当前最佳的项目加入子集合,其中效用函数可以采用相关性函数或者距离函数,不同的效用函数会带来差异性的多样性优化效果. Carbonell 等人[10]提出采用最大边缘相关(maximal marginal relevance, MMR)作为效用函数的贪婪优化方法,是该领域的先驱性研究之一.此后,研究人员进一步针对效用函数进行研究. 文献[11]提出多样性效用函数所需满足的一些公理,并指出目前没有任何一种方法可以同时满足这些公理. Ziegler 等人[12]提出了一种基于推荐列表内部相似度最小化思想的多样性优化效用函数,在每次进行贪婪选择时,最小化当前推荐列表的内部综合相似度. Ashkan 等人[6]提出一种 DUM 方法,将推荐列表多样性优化问题转化为子模函数约束条件下的模函数最大化问题,该方法的特色在于仅需要最大化模函数,降低了问题复杂度.

与 GH 方法不同的是,SH 方法首先初始化一个待推荐项目子集合,然后逐步遍历总集合中的其他元素,并与子集合中的某个元素交换,以提高子集合的多样化. Erkut 等人[13]分析了 SH 方法的特点,并将该方法划分为 2 种:1)选出第 1 个能够提高当前子集合多样性的项目并进行置换;2)选出能够最大幅度提高当前子集合多样性的项目并进行置换. Erkut 等人认为通常情况下前者的效果更好. Vieira 等人[5]进一步研究 SH 方法发现,在置换过程中加入随机性会取得更好的效果,针对这种现象,他们通过分析发现,在优化过程中,一些对当前集合多样性没有提高作用的项目在后续过程中可能会产生有益影响. Liu 等人[14]提出一种多置换(multi-swap)策略,在一次置换过程中,考虑同时置换多个项目,加快了算法收敛速度.

除了 SH 和 GH 启发式方法,也有研究人员尝试从其他思路解决该问题. 文献[15]利用多臂 bandits 模型学习优化推荐集合多样性,该方法基于用户的点击行为,可及时捕捉用户兴趣变化,及时调整优化效果.但该方法忽略了优化对象的语义信息,

抑制了集合多样性的提高. 文献[16]引入概率模型解决推荐系统中多样性与相关性之间相对矛盾的问题,并通过控制参数调整二者的重要性. 虽然这种混合模型在很多场景下效果不错,但一般情况下很难获得一个通用的算法控制参数. Qin 等人[17]提出利用规则熵提高多样性,并证明了方法的有效性,但仍然需要提前设定算法控制参数. Drosou 等人[18]提出一种基于覆盖树的方法优化集合多样性,取得了不错的优化效果,但该方法仅基于用户的评分模式,同样缺乏利用优化对象语义信息的机制.

上述研究成果涉及到播存网络及结果集合多样性优化 2 个方面,为本文研究提供了指引与启发. 但是,综合分析以上研究可知,已有多样性优化方法在播存网络环境下存在不少问题:一方面,播存网络环境下 UCL 数量巨大,更新频繁,而已有方法大多需要基于当前已有数据通过预先运算寻找合适的控制参数,以提高结果集合的多样性,无法高效满足播存网络环境要求;另一方面,播存网络环境下 UCL 对应于信息资源,具有明显的时效性,已有方法主要针对多样性与相关性,缺乏对时效性的考虑,容易造成优化后的结果集合"过时",降低播存网络环境中的用户体验;此外,由于 UCL 的数量巨大,播存网络环境下用户对于 UCL 的反馈速度要求更为迫切,已有方法大多需要大量迭代运算,效率有待提高完善.

基于以上分析,本文提出一种基于语义覆盖树的 UCL 推荐多样性优化算法 UDSCT,克服已有方法在播存网络环境中存在的不足. 首先,本文在普通覆盖树基础上,面向 UCL 设计了一种时间敏感的语义覆盖树,在构造该树的过程中同时考虑了用户评分模式及 UCL 语义信息,且时效性较强的 UCL 优先插入树的顶端;其次,本文基于该树提出了具体的多样性优化算法,包括 1 种子集合初步查询算法及 2 种启发式调整算法,本文还设计了用户请求响应机制,可快速反馈与用户指定 UCL 高度相关的结果集合;最后,本文证明与分析了上述多样性优化算法及用户请求响应机制的正确性及有效性,仿真实验结果表明,该算法具有良好的优化效果,可有效满足播存网络环境下 UCL 推荐多样性优化的需求.

2 UDSCT 算法详细描述

本节首先描述基于语义覆盖树的推荐列表多样性优化算法的总体流程,然后给出算法相关的形式化定义,最后详细描述具体的 UCL 语义覆盖树构造算法及 UCL 推荐多样性优化算法.

基于语义覆盖树的 UCL 推荐列表多样性优化算法的总体流程如图 2 所示. 该算法首先根据时间顺序构建时间敏感的 UCL 语义覆盖树,然后基于该树通过查询操作及补充算法获得多样化的 UCL 列表,最后将该 UCL 推荐给用户并根据用户请求响应聚焦请求.

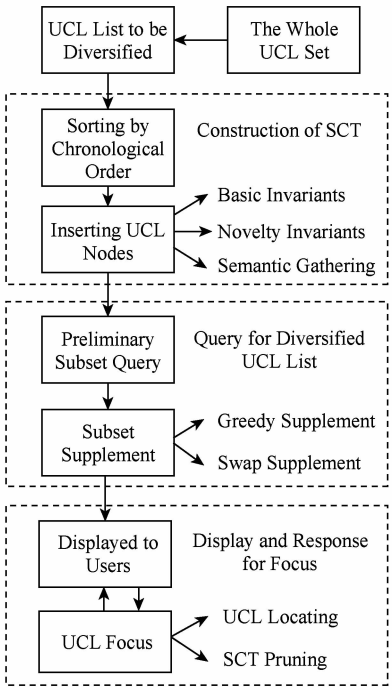


Fig. 2 The flow chart of UDSCT

图 2 UDSCT 算法流程图

2.1 UDSCT 算法相关定义

为更清楚地描述该问题及其解决算法,本节给出了相关定义.

定义 1. k-UCL 列表多样性优化. 令 $I=\{i_1,i_2,\cdots,i_m\}$ 为 UCL 列表,U 为多样性度量函数,k 为正整数($k\leqslant|I|$),k-UCL 列表多样性优化目标为寻找 I 的最优化子列表 S^*,满足条件:

$$S^*=\arg\max_{S\subseteq I}U(I,S),S^*\subseteq I\wedge|S^*|=k. \quad (1)$$

在获得当前最优 UCL 子列表后 S^*,系统将其呈现给用户,用户可能对其中的某一个 UCL 感兴趣,并希望得到一个与该 UCL 相关度更高的子列表. 本文将这个过程定义为聚焦,形式化描述如下:

定义 2. UCL 聚焦. 在定义 1 的基础上,聚焦操作根据用户的反馈查询当前最优 UCL 列表 S^* 中更符合用户兴趣的子列表 S',使得:

$$S'=\arg\max_{S\subseteq S^*}U(I,S). \quad (2)$$

以上定义均基于语义覆盖树实现,语义覆盖树由覆盖树发展演化而来,覆盖树最初由 Beygelzimer 等人在文献[19]中提出,用以快速查找最近邻.覆盖树为分层的树数据结构,树中的每一层均覆盖其下面所有的层,树中的层数用整数 l 标示,且 l 从上往下逐步减小.对于给定的一个项目(如 UCL)列表 S,S 中的每个项目均对应于覆盖树中的至少一个节点.

定义 3. 覆盖树.令 C_l 表示第 l 层的节点列表,$d(p,q)$表示节点 p 和 q 之间的距离,覆盖树必须同时满足 3 个约束条件:

1) 嵌套性.$C_{l-1}\subset C_l$,即如果 p 出现在第 l 层($p\in C_l$),则它必出现在第 l 层之下的所有层($p\in C_j, j>l$).

2) 覆盖性.对于每一个 $p\in C_l$,有且仅有一个 $q\in C_{l-1}$作为 p 的父节点,且 q 满足 $d(p,q)\leqslant 1/2^{l-1}$.

3) 分离性.所有的 $p,q\in C_l$,均满足 $d(p,q)>1/2^l$.

基于覆盖树的以上特性,可以总结出其在 UCL 推荐多样性优化问题方面所具有的独特优势.一方面,覆盖树由多层节点构成,每一层都包含了不同数目的节点并且同层节点之间的距离均存在最低阈值,该阈值随着树高度的降低而逐步减小,即上层中的节点之间最小距离较大,而下层中的节点之间最小距离较小,这种特性保证了每一层以及整个覆盖树的节点多样性.另一方面,覆盖树可以根据用户的请求在覆盖树中的合适高度选取节点列表返回给用户,查询过程简单、运算量小,可以保证 UCL 推荐多样性优化算法的高效性.

但是,覆盖树在解决播存网络环境下 UCL 推荐多样性优化问题时仍然存在着 2 个较为明显的缺陷.1)覆盖树并不考虑节点的内容语义信息,无法有效利用 UCL 所包含的丰富语义信息;2)在 UCL 推荐多样性优化过程中,UCL 的时效性对用户体验有着重要影响,覆盖树缺乏针对时效性的处理机制.

为此,本文对覆盖树进行相应扩展,设计了一种时间敏感的 UCL 语义覆盖树.

定义 4. UCL 语义覆盖树.语义覆盖树为覆盖树的一种特例,在覆盖树的基础上增加了时效性及语义信息等限制条件,UCL 语义覆盖树需同时满足 3 个约束条件:

1) 基本条件.覆盖树所满足的 3 个约束条件:嵌套性、覆盖性和分离性.

2) 时效性.对于所有的 UCL $p\in C_i$,令 UCL s 为 p 在 $l-1$ 层的父节点,则有 $t_s\leqslant t_p$,其中 t_s,t_p 分别表示 UCL s 和 p 的时间.

3) 语义聚集性.对于所有的 UCL $p\in C_i$,令 s 为 p 在 $l-1$ 层的父节点,则有,$s=\arg\max_{h\in c_{i-1}} sem(s,h)$,其中 $sem(s,h)$为语义度量函数.

上述条件中,时效性保证了每个父节点均比其子节点更加新颖,从而使得从上往下遍历树寻找匹配用户兴趣时能够优先选择时效性更高的 UCL.而语义聚集性则确保了在构建树的过程中,父节点与子节点之间的语义相似度更高,间接地促进了 UCL 列表的多样性效果.

2.2　UDSCT 算法描述

在介绍完相关基本概念后,本节将继续介绍如何完成 UCL 语义覆盖树的构造、如何基于该树完成 UCL 推荐多样性优化以及如何进一步快速响应用户的聚焦请求.

2.2.1　UCL 语义覆盖树构造

UCL 语义覆盖树采用自顶向下、逐个插入 UCL 的方式构造,首先选择最新的 UCL 作为树的根,然后遵循"新颖优先"的原则插入其他 UCL.需要说明的是,在插入过程中,必须严格保证 3 个约束条件不被违背.构造的详细过程如算法 1 所示.

算法 1. UCL 语义覆盖树的构造.

输入:待优化 UCL 列表 I;

输出:列表 I 对应的 UCL 语义覆盖树.

/* 按照时间顺序对 I 中的 UCL 进行排序,依次调用函数 $Insert()$ */

① $I=reversedChronological(I)$;
② Foreach i in I
③　If $indexOf(i)==0$
④　　$T_{root}=i$;
⑤　Else
⑥　　$Insert(i,Q_0,0)$;
⑦　EndIf
⑧ EndForeach

/* $Insert$(item p,Q_l,level l):将 UCL p 插入到 T 的第 l 层 */

⑨ $C=\{$children of each node in $Q_l\}$;
⑩ If $\min_{c\in C} dis(c,i)>1/2^l$
⑪　return true;
⑫ Else
⑬　$l=l+1$;
⑭　$flag=Insert(i,Q_l,l)$;
⑮　If $flag==$true && $\min_{c\in C} dis(c,i)\leqslant 1/2^{l-1}$
⑯　　$PC=\{c\,|\,dis(c,i)\leqslant 1/2^{l-1},\ t.c>t.i\}$;

⑰ $p=\arg\max_{c\in PC} sem(V_c,V_i)$;
⑱ make p parent of i;
⑲ return false;
⑳ Else
㉑ return $flag$;
㉒ EndIf
㉓ EndIf

算法 1 中 $Insert()$是基于文献[19]中的插入算法,但由于该算法仅可用于构建基本覆盖树,故需要对其进行适当改进,在构建过程中除了需要遵循定义 3 中的约束条件,还需兼顾定义 4 中的条件.具体而言,算法 1 首先根据时间顺序将已有 UCL 列表进行排序,然后依次调用插入函数 $Insert$(item p,Q_l,level l)(行①～⑧),该操作保证了构造过程满足了时效性约束条件,最先插入的最新颖的节点作为树的根.插入函数以待插入的 UCL、待插入的层数以及该层已含有的 UCL 作为输入参数,然后逐层递归地检查直到发现满足分离性约束条件的层数(行⑨～⑭).在第一次找到该层之后,该函数将上一层中满足覆盖性条件的 UCL 节点识别出放入待插入节点的父节点候选节点集合(行⑮⑯).最终,该算法依次计算待插入 UCL 的特征向量与父节点候选节点集合中的节点的特征向量的语义关联程度,选出关联度最高的 UCL 节点作为待插入节点的最终父节点(行⑰⑱).

在构造 UCL 语义覆盖树的过程中,语义聚集条件使得 UCL 节点总是选择与其语义最为相关的 UCL 节点作为父节点,这进一步加强了同层节点之间的多样性,并为之后的 UCL 聚焦操作提供了重要基础.

2.2.2 多样化 UCL 结果集合查询

UCL 语义覆盖树中的每个节点都对应了 UCL 推荐列表中的一个 UCL,当用户确定想要获取的 UCL 数目后,只需逐层遍历树的节点,获得合适的 UCL 列表.具体而言,层数越低,该层的 UCL 之间距离越大,该层的多样性便越高.下降搜索过程如算法 2 所示.

算法 2. 多样化列表初步查询算法.

输入:UCL 列表 I 对应的语义覆盖树 T、结果子列表中 UCL 数目 k;

输出:包含 k 个 UCL 多样化的子列表 S^*.

① $l=0$;
② While ($l<T.height$) do
③ If $|Q_l|\geqslant k$
④ Break;
⑤ EndIf
⑥ $l=l+1$;
⑦ EndWhile
⑧ $S_b=Q_{l-1}$;
⑨ $C=Q_l-Q_{l-1}$;
⑩ $S^*=supplement(S_b,C,k)$;
⑪ return S^*.

对于 UCL 列表 I 对应的 UCL 语义覆盖树以及用户指定的子列表大小,算法 2 首先定位相邻的两层 l 和 $l-1$,使得$|Q_{l-1}|<k<|Q_l|$,其中$|Q_l|$和$|Q_{l-1}|$分别表示两层的 UCL 节点数量(行②～⑦).然后,算法将 $l-1$ 层中的所有 UCL 节点放入子列表中并调用补充算法 $supplement(S_b,C,k)$,从出现在 l 层但没有出现在 $l-1$ 层的 UCL 节点中(Q_l-Q_{l-1})挑选剩余的 $k-|Q_{l-1}|$个 UCL 节点(行⑧～⑩).

本文设计了 2 种补充算法以计算出完整的 UCL 子列表,分别为:贪婪补充算法(见算法 3)与置换补充算法(见算法 4).在贪婪补充算法中,依次从 Q_l-Q_{l-1}中挑选出与当前子列表中所有 UCL 语义关联度最小的 UCL,即 $p=\arg\min_{c\in C} rel(V_c,V_{S^*})$,加入到子列表,然后不断重复挑选操作,直至确认所有的 k 个 UCL 节点.

算法 3. 贪婪补充算法.

输入:基础子列表 S_b、候选集合 C、结果子集合中 UCL 数目 k;

输出:包含 k 个 UCL 多样化的子集合 S^*.

① $S^*=S_b$;
② While ($|S^*|<k$) do
③ $p=\arg\min_{c\in C} rel(V_c,V_{S^*})$;
④ $S^*=S^*\bigcup\{p\}$;
⑤ EndWhile
⑥ return S^*.

算法 4. 置换补充算法.

输入:基础子列表 S_b、候选集合 C、结果子集合中 UCL 数目 k;

输出:包含 k 个 UCL 多样化的子集合 S^*.

① $S^*=S_b$,$R=Random_{k-|S^*|}(C)$;
② $S^*=S_b\bigcup R$;
③ $\theta=$MAX;
④ While ($\theta>\delta_0$) do
⑤ $r_a=Random_1(C-R)$,$r_b=Random_1(S^*)$;

⑥　$\theta=div(S^*\cup\{r_a\}-\{r_b\})-div(S^*)$;
⑦　If $\theta>\delta_0$
⑧　　$S^*=S^*\cup\{r_a\}-\{r_b\}$;
⑨　Else
⑩　　Continue;
⑪　EndIf
⑫ EndWhile
⑬ return S^*.

而在置换补充算法中,首先从候选集合中随机选出 $k-|S^*|$ 个 UCL 节点并加入到初始子集合 S^* 中,$Random_m(S)$ 函数负责随机从 S 中选择 m 个 UCL.本质上而言,S^* 由 2 部分组成:1)来自初始子集合 S_b 的已确定部分;2)来自候选集的待优化部分.之后,分别随机从剩余的候选集合 $C-R$ 及待优化部分 R 中取出 2 个 UCL 节点 r_a 和 r_b,然后计算出是否 r_a 比 r_b 对已选出子集合的多样性提高更大.该多样性的提高被量化为变量 θ,为了保证算法的效率,算法 4 设定了置换阈值 δ_0,当且仅当 $\theta>\delta_0$ 时,置换才会进行,否则将忽略该对节点,继续选出下一对进行比较.置换补充算法不断重复随机选节点、多样性比较等操作,直至子集合的多样性几乎保持不变,即 $\theta<\delta_0$.

2.2.3　UCL 聚焦响应算法

在将初步多样性优化后的 UCL 列表呈现给用户后,用户会对其进行浏览并会根据自身兴趣及偏好查找自己感兴趣的 UCL.假设该用户对某个 UCL 感兴趣,并想进一步了解与该 UCL 更加相关的一些 UCL,此时,此时将调用 UCL 聚焦响应算法,算法的处理流程如算法 5 所示.

算法 5. UCL 聚焦响应算法.

输入:UCL 列表 I 对应的语义覆盖树 T、用户指定的 UCL q、子列表中 UCL 数目 k;

输出:包含 k 个 UCL 的子列表 S'.

① $l=0$;
② While ($l<T.height$) do
③　If $q\in Q_{\text{length}}$ && ($\exists p\in q.children$ && $p\neq q$)
④　　Break;
⑤　EndIf
⑥　$l=l+1$;
⑦ EndWhile
⑧ $T'=new(T)$ && $q=T'.root$;
⑨ $S'=Query(T',k)$;
⑩ return S'.

算法 5 首先根据用户请求,从已构造的 UCL 语义覆盖树中逐层查询用户感兴趣的 UCL q(行①~⑦).需要注意的是,根据覆盖树的特性,相同的 UCL 可能会在树中出现多次,为了找到恰当的节点,算法需要找到第 1 个带有非自身子节点的 q(行③).一旦找到该节点,算法并对该覆盖树进行剪枝处理,仅保留以 q 为根节点的子树(行⑧).最终,基于该子树,便可以调用子集合查询算法(算法 2)获取与该 UCL 更加相关的 UCL 列表.

3　算法分析

本节将分析 UDSCT 算法复杂度并总结、证明其具有的相关性质.

3.1　算法复杂度

本文所提 UDSCT 算法主要由构造语义覆盖树及查询 2 部分组成,其中查询部分仅需定位层数及少量列表补充操作,所占时间、空间极少,因此本文将重点分析构造语义覆盖树所需的时空复杂度.语义覆盖树是普通覆盖树的特例,普通覆盖树构造的时间复杂度为 $O(n\log n)$,空间复杂度为 $O(n)$[19].对于语义覆盖树,其构造的主要区别在于需要对 UCL 节点进行排序并在构建过程中进行语义关联度计算,前者(快速排序)的时间复杂度为 $O(n\log n)$,空间复杂度为 $O(\log n)$,而后者时空复杂度均为 $O(1)$,因此,语义覆盖树构造的时间复杂度为:$O(n\log n)+O(n\log n)+O(1)=O(n\log n)$,空间复杂度为 $O(n)+O(\log n)+O(1)=O(n)$,由此可知,UDSCT 算法复杂度较低.

3.2　算法性质

基于语义覆盖树具有 3 条重要性质.首先,对于待插入树的 UCL i 而言,一旦其在树中的所属层数 l 确定,则在 $l-1$ 层中必定有且仅有一个 UCL 可以当作 UCL i 的父节点.

定理 1. 给定待插入 UCL i,函数 $Insert$(item i,Q_l,level l)能且仅能在 $l-1$ 层中找到 i 的一个父节点 UCL,且在 l 层中,所有的节点满足分离性条件 $\min\limits_{c\in C}dis(c,i)>1/2^l$.

证明.在插入函数中,UCL i 的所属层 l 通过自上而下的遍历方式进行寻找,其所属层必须满足分离性不等式 $\min\limits_{c\in C}dis(c,i)>1/2^l$,也就是说,第 l 层中的任意节点与 i 之间的距离均必须大于 $1/2^l$,这也表明,在第 $l-1$ 层中必定存在至少一个 UCL

p 到 i 的距离不大于 $1/2^{l-1}$，即 $dis(i,p)<1/2^{l-1}$，而该不等式恰好为寻找 UCL i 候选父节点的条件. 此时，可以通过语义约束条件从这不少于一个 UCL 的候选集内选出唯一一个 UCL 作为 i 的父节点，因此可知，UCL i 的父节点有且仅有一个. 证毕.

下面证明 UCL 语义覆盖树构造算法的正确性.

定理 2. UCL 语义覆盖树构造算法满足定义 4 中的所有约束条件.

证明. UCL 语义覆盖树共包含 5 个限制性约束条件，其中，嵌套性、覆盖性和分离性为覆盖树的基本约束条件，由于 UCL 语义覆盖树为覆盖树的优化数据结构，本文不再赘述这 3 个约束条件的证明. 在构造过程中，算法 1 首先将待插入 UCL 节点按照时间顺序排序，然后在插入 UCL 节点时检查父节点是否比其子节点时效性更高($t.i>t.p$)，因此，算法的时效性得以保证. 而对于语义聚集性，算法在计算待插入 UCL 节点的父节点时总会从候选集合中选出与其语义相似度最高的节点作为其父节点，如定理 1 所述，因此，语义聚集性得证. 综上所述，最终构造出来的 UCL 语义覆盖树可同时满足 5 个约束条件. 证毕.

最后，本文将继续证明，采用聚焦操作得到的 UCL 覆盖树同样满足 UCL 语义覆盖树的约束条件.

定理 3. 基于 UCL 语义覆盖树进行聚焦操作，获得的子树同样满足 5 个限制性约束条件.

证明. 假设 q 为 UCL 语义覆盖树 T 中第 m 层的 UCL 节点，且聚焦操作返回的新语义覆盖树为 T'，即，q 为 T' 的根节点. 对于 T' 中的任意 UCL 节点，令 L 为其在树 T 中的层数，那么，在树 T' 中，其所在层数为：$L'=L-m$. 显然 T' 保留了嵌套性、时效性和语义聚集性. 为了证明本定理，本文将论证 T' 同样满足分离性和覆盖性. 对于树 T 中的任意层 n，令 m_dis 为其中任意 2 个 UCL 节点的最小距离，则其必定满足：

$$m\_dis>1/2^n=(1/2^m)\times(1/2^{n-m}).$$

与此同时，在树 T' 中，其新的层数为 $L'=n-m$，因此可得：

$$m\_dis>1/2^n=(1/2^m)\times 1/2^{L'}.$$

也就是说剪枝后的子树保留了分离性条件，其距离下界为 $(1/2^m)\times 1/2^{L'}$，易知，该下界取决于新的层数 L' 及其根节点在原始树 T 中的层数. 同理可证，算法生成的新树 L' 同样满足覆盖性条件. 证毕.

4 性能评估

4.1 数据集及实验过程

为了展示基于 UDSCT 算法的性能优势并与其它算法进行公平对比，本文在公开数据集 MovieLens① 上进行实验，重点测试该算法所生成的最终推荐列表在给定指标上的性能. 该数据集包含了 943 个用户对 1682 部电影的评分，评分区间为 1～5 分，每个用户至少评价 20 部电影，每部电影还平均带有 17 个语义标签，可用以代表 UCL 所包含的语义信息. 此外，本文采用随机打时间戳的方法标记每个电影的时间(播存网络中，当 UCL 数量较大时，其时间戳可视为随机方式生成)，以确定构造 UCL 语义覆盖树时的插入顺序及进行时效性对比.

实验过程如下：实验随机选择 100 名用户进行测试，首先通过预测算法预测用户的初步推荐列表，然后采用不同的推荐列表多样性优化算法优化列表，最后统计测试不同算法所取得的相关性能指标并进行对比，针对对比结果给出相应分析. 为保证实验结果的稳定性，上述过程均重复 10 组，本文所述结果为 10 组结果的平均值. 另外，在实验过程中，本文采用 PCC 值度量基本距离，而对于语义距离，本文基于用户对电影的语义标签信息进行计算.

4.2 评估标准

为全面测试 UDSCT 算法的性能，本文将围绕 UCL 列表平均精度、列表多样性(包括列表内部元素差异度及列表类别多样性)、列表时效性、算法运行时间等指标进行测试.

UCL 推荐列表平均精度(average list precision, ALP)计算公式为

$$ALP=\frac{1}{n}\sum_{j=1}^{n}r_j, \tag{3}$$

其中，n 表示列表中的 UCL 个数，r_j 表示第 j 个 UCL 的评分.

列表内部元素的差异度(list internal difference, LID)计算公式为

$$LID=\frac{1}{n(n-1)}\sum_{i=1}^{n}\sum_{j=1,j\neq i}^{n}(1-sim(u_i,u_j)), \tag{4}$$

① http://www.cs.umn.edu/Research/GroupLens/

其中,u_i 表示第 i 个 UCL,sim 表示相似度函数,LID 值越高,说明列表的多样性越好.

UCL 列表类别多样性(list category diversity, LCD)的计算公式为

$$LCD = \sum_{j=1}^{n} \frac{1}{m_j^2}, \tag{5}$$

其中,$\frac{1}{m_j^2}$表示第 j 个 UCL 对类别多样性的贡献度,m 为该类别在前 $j-1$ 个条目中已出现的次数,若该类别为第 1 次出现,则 $m=1$,该类别已出现次数越多,当前条目对推荐列表的类别多样性贡献越低.

UCL 列表时效性(list novelty, LN)的计算公式为

$$LN = \frac{1}{n}\sum_{i=1}^{n} \frac{|t_i - t_{\text{newest}}|}{|t_{\text{oldest}} - t_{\text{newest}}|}, \tag{6}$$

其中,t_i 为列表中第 i 个 UCL 的时间,t_{newest} 为最新 UCL 的时间,t_{oldest} 为最旧 UCL 的时间,LN 值越小,UCL 列表的时效性便越高.

4.3　实验结果

本实验部分将根据第 4.2 节中所述评估标准,对比分析如下 5 个算法:基于贪婪策略的 UDSCT 算法(GUDSCT)、基于置换策略的 UDSCT 算法(SUDSCT)、基于普通覆盖树的 UCL 推荐多样性优化算法(UDCT)[18]、基于预测评分排序的无多样性优化初始排序算法(NODIV)、基于多置换策略的多样性优化算法(MSDIV)[14]. 其中,NODIV 算法不包括任何多样性优化操作,用以作为其他方法的对比基准;UDCT 算法用以验证本文算法是否可利用 UCL 语义及时效有效地提高相应的多样性及时效性指标;MSDIV 算法用以验证本文算法是否可以快速取得多样性优化结果.

在实验过程中,本文首先对每个目标用户进行评分预测,并从高到低排序,取前 n 个(n 取值为 15～50)UCL 作为初始列表,然后采用以上算法对列表进行多样性优化,取前 k 个($k \leqslant n$,本文取 $k=10$ 或 15)UCL 作为最终推荐列表,并计算各项指标,在不同算法之间进行对比. 另外,需要说明的是,本文将 SUDSCT 算法与 MSDIV 算法中的置换阈值 $\delta_0=0.01$. 该参数对算法的多样性及运行速度会有一定影响,本文通过实验发现,当 $\delta_0=0.01$ 时,实验效果较好. 而每次置换的元素个数会对 MSDIV 算法的运行时间有一定影响,为保证 MSDIV 算法能够取得较少的运行时间,本文将其设置为 5.

4.3.1　UCL 推荐多样性

本部分将根据 UCL 推荐列表内部元素差异度 LID 及列表类别多样性 LCD 两个指标测试不同算法的多样性优化效果.

图 3 和图 4 分别展示了 $k=10$ 和 $k=15$ 两种情况下不同算法之间 LID 值的对比结果. 需要指出的是,LID 值介于 0,1 之间.

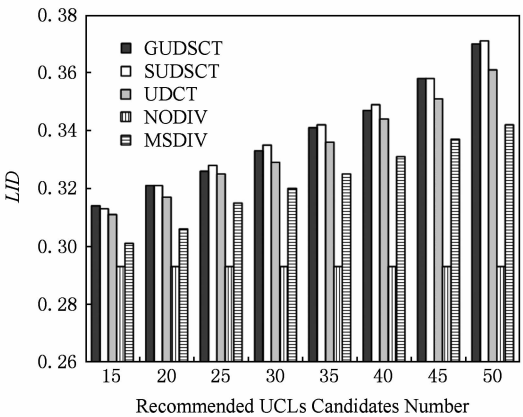


Fig. 3　LID comparison of different methods when $k=10$

图 3　$k=10$ 时不同方法 LID 对比

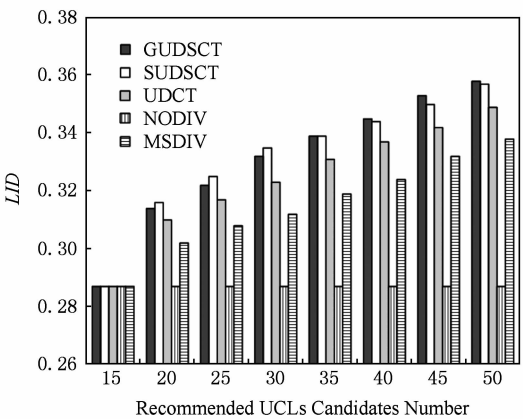


Fig. 4　LID comparison of different methods when $k=15$

图 4　$k=15$ 时不同方法 LID 对比

不难发现,在所有算法中 NODIV 算法的 LID 值最低,无论候选集合包含多少个 UCL(从 15～50),其 LID 值始终维持在 0.3 以下. 从图中还可以发现 GUDSCT 与 SUDSCT 算法的 LID 值一直较为接近,均高于其他 3 种算法,也就是说基于 UCL 语义覆盖树的 UDSCT 算法具有更好的多样性优化效果. 而 UDCT 算法的曲线介于 UDSCT 算法与 MSDIV 之间,这表明基于覆盖树的算法所具有的多样性优化效果比典型的启发式算法更好,但弱于基于 UCL 语义覆盖树的算法. 本文认为,UDCT 算法与 UDSCT 算法之间存在的这种差异正是由于 UDCT 算法缺乏对 UCL 语义信息的考虑. 此外,随着候选集合的增大,所有多样性优化算法的 LID 值

均逐渐升高,同时,GUDSCT 与 SUDSCT 算法增幅较大.

图 5 和图 6 分别展示了最终推荐列表集合大小 $k=10$ 和 $k=15$ 两种情况下不同算法之间 *LCD* 值的对比结果.

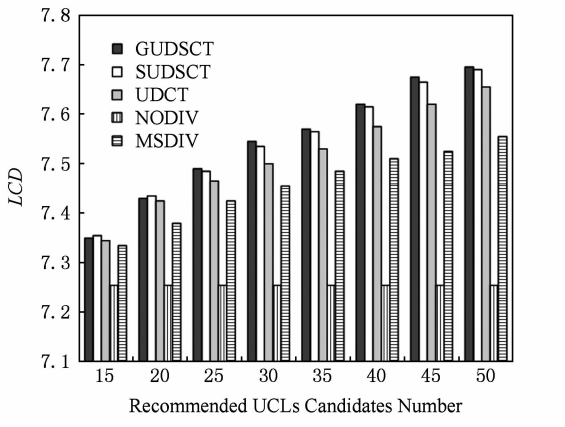


Fig. 5 *LCD* comparison of different methods when $k=10$

图 5 $k=10$ 时不同方法 *LCD* 对比

$k=10$ 时,*LCD* 的取值区间为[1.998,10]. 当所有 UCL 属于同一个类别时,*LCD* 取值最小,根据式(5)可知,$LCD=1+1/2+\cdots+1/2^9\approx1.998$;当所有 UCL 均属于不同类别时,*LCD* 取值最大,*LCD*=10. $k=15$ 时,计算过程同上.

整体上而言,图 5 和图 6 中数据的变化趋势与图 3 和图 4 较为一致,与其他 3 种算法相比,基于 UCL 语义覆盖树的 GUDSCT 与 SUDSCT 算法仍然展示出明显的性能优势. 在 $k=15$,$n=50$ 时,与 MSDIV 方法相比,这 2 种方法的 *LCD* 值提高了将近 10%. 此外,需要说明的是,当候选集合与最终推荐结果集合含有相同的 UCL 数目时(即 $k=n$),所有算法的多样性相同,因为此时无需进行多样性优化.

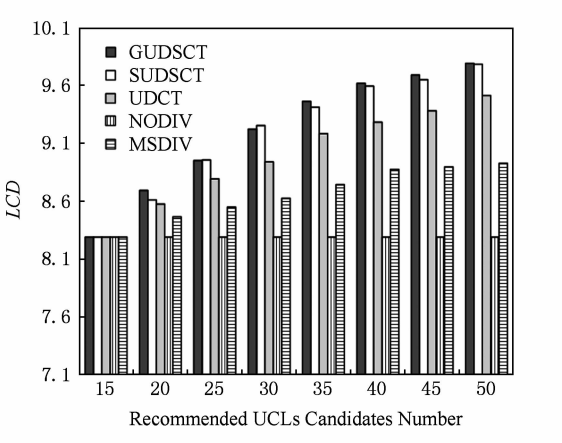


Fig. 6 *LCD* comparison of different methods when $k=15$

图 6 $k=15$ 时不同方法 *LCD* 对比

4.3.2 UCL 推荐精度

表 1 描述了不同算法对于 UCL 推荐精度 *ALP* 的影响. 从表 1 可以观察到,NODIV 算法的 *ALP* 值保持不变,其他算法的 *ALP* 值随着候选集合的变大而逐步降低. 这是由于 NODIV 算法仅根据预测分数对 UCL 列表排序,而候选集合的增大并不会对前 k 个 UCL 的预测分数及排序产生影响,但在其他算法中,一些预测评分较低但会提高列表多样性的 UCL 可能出现在前 k 个位置,从而使得 *ALP* 值下降.

Table 1 *ALP* Comparison of Different Methods

表 1 不同方法之间的 *ALP* 对比

n	$k=10$					$k=15$				
	GUDSCT	SUDSCT	UDCT	NODIV	MSDIV	GUDSCT	SUDSCT	UDCT	NODIV	MSDIV
15	4.309	4.307	4.304	4.315	4.291	4.303	4.303	4.303	4.303	4.303
20	4.298	4.298	4.275	4.315	4.261	4.295	4.293	4.271	4.303	4.248
25	4.294	4.294	4.259	4.315	4.224	4.286	4.287	4.244	4.303	4.209
30	4.288	4.289	4.214	4.315	4.197	4.281	4.283	4.209	4.303	4.181
35	4.284	4.285	4.196	4.315	4.164	4.275	4.277	4.187	4.303	4.155
40	4.279	4.279	4.188	4.315	4.143	4.268	4.27	4.173	4.303	4.139
45	4.271	4.272	4.161	4.315	4.129	4.265	4.265	4.158	4.303	4.112
50	4.262	4.261	4.149	4.315	4.113	4.262	4.261	4.143	4.303	4.105

从表 1 还可以发现,基于 UCL 语义覆盖树的 2 种算法在推荐精度方面均优于 UDCT 和 MSDIV 算法,且候选集合愈大,优势愈明显. 此外总体上而言,所有优化算法在 $k=15$ 时取得的 *ALP* 值均低于 $k=10$ 时,这是因为推荐结果集合较大时,若高预测评分数量有限,一些低预测评分 UCL 将出现在

推荐集合中,而当播存网络环境持续长时间运行后,高预测评分 UCL 数量会逐步增加,这种现象会得到有效缓解.

4.3.3　UCL 推荐时效性

不同算法之间的 UCL 推荐时效性比较结果如表 2 所示. 如 4.2 节所述,LN 值越低,UCL 推荐列表的时效性越强. 从表 2 可以发现,GUDSCT 与 SUDSCT 算法具有较强的时效性,在 $n=50$,$k=15$ 时,二者的 LN 值分别达到了 0.356 和 0.359. 与之形成鲜明对比的是,UDCT,NODIV,MSDIV 等算法的 LN 值始终在 0.57 左右浮动,无明显变化. 这种现象表明,在这些算法中,UCL 在时间上是随机分布的,显然,本文所提基于 UCL 语义覆盖树的算法可显著提高 UCL 推荐列表的时效性.

Table 2　*LN* Comparison of Different Methods

表 2　不同方法之间的 *LN* 对比

n	$k=10$					$k=15$				
	GUDSCT	SUDSCT	UDCT	NODIV	MSDIV	GUDSCT	SUDSCT	UDCT	NODIV	MSDIV
15	0.426	0.422	0.572	0.572	0.574	0.576	0.576	0.576	0.576	0.576
20	0.418	0.411	0.576	0.572	0.566	0.426	0.419	0.578	0.576	0.566
25	0.408	0.403	0.568	0.572	0.568	0.421	0.417	0.567	0.576	0.57
30	0.392	0.392	0.561	0.572	0.572	0.399	0.402	0.565	0.576	0.572
35	0.387	0.386	0.572	0.572	0.571	0.382	0.384	0.578	0.576	0.574
40	0.375	0.373	0.575	0.572	0.577	0.371	0.375	0.579	0.576	0.579
45	0.369	0.365	0.563	0.572	0.576	0.359	0.359	0.569	0.576	0.581
50	0.362	0.358	0.573	0.572	0.572	0.356	0.359	0.575	0.576	0.575

4.3.4　算法运行时间

UDSCT 算法基于语义覆盖树优化 UCL 推荐列表的多样性,在树构造完毕后,每次优化仅需一次语义覆盖树查询操作及少量的 UCL 列表调整操作,本实验部分将比较 UDSCT 算法与其他典型算法的运行时间情况,实验结果分别如图 7、图 8 所示:

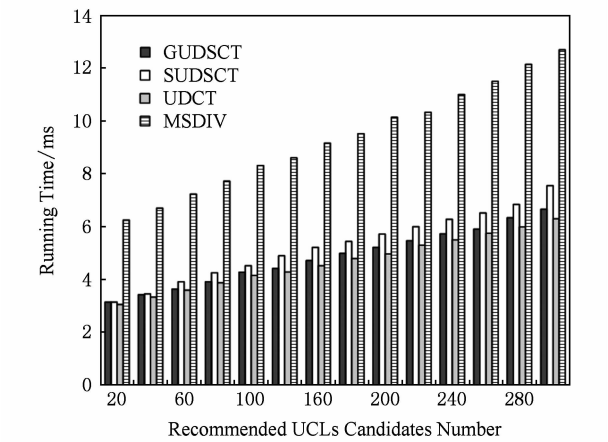


Fig. 7　Running time comparison of different methods when $k=10$

图 7　$k=10$ 时不同算法之间的运行时间对比

需要指出的是,由于 NODIV 算法中无多样性优化操作,故不参加本部分实验.

从图 7、图 8 中首先可以发现,MSDIV 算法比

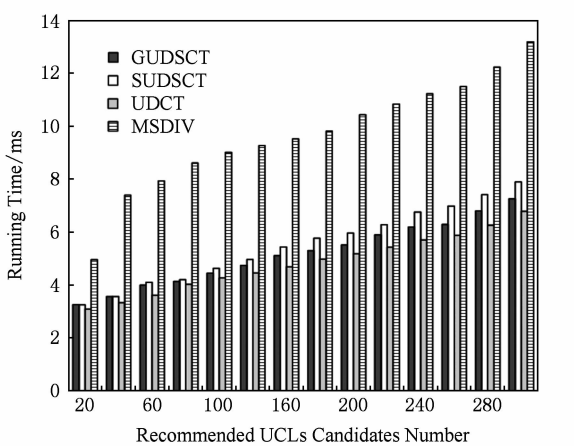


Fig. 8　Running time comparison of different methods when $k=15$

图 8　$k=15$ 时不同算法之间的运行时间对比

基于覆盖树的 3 种算法(GUDSCT,SUDSCT,UDCT)耗时更久. 此外随着候选集合的增大,MSDIV 算法的耗时持续攀升,而基于覆盖树的算法则缓慢上升,后者在 $n=200$ 时的耗时仍少于前者在 $n=20$ 的耗时,这表明基于覆盖树的算法具有较好的可扩展性.

从图 7、图 8 还可以发现,基于普通覆盖树的 UDCT 算法比本文所提的 UDSCT 算法(包括 GUDSCT,SUDSCT)耗时更少,深入分析可知,这是因为 UDCT 算法在构建树的过程中无需排序操作

及语义关联运算,节省了部分时间.同时 SUDSCT 算法耗时比 GUDSCT 稍久一点,这也不难理解,因为前者在查询后的少量调整过程中仍需要较多的迭代置换尝试.

4.3.5 UCL 聚焦操作

在验证 UCL 聚焦操作性能时,由于该操作主要基于树的裁剪操作,其多样性、精度、时效性等指标取决于当前已构建的 UCL 语义覆盖树,相关性能已在上述 4 个实验部分测试,本部分实验主要关注 UCL 聚焦操作的响应时间,实验结果如图 9 所示:

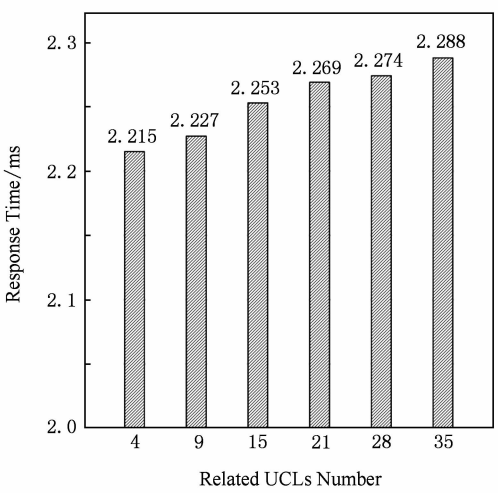


Fig. 9 Response time of UCL Focus

图 9 UCL 聚焦响应时间

图 9 主要包括了 6 组实验结果,横坐标为响应时间,纵坐标为待聚焦的 UCL 所包含的相关 UCL 数目,每组实验重复 5 次,图 9 所展示的为 5 次的平均结果.观察图 9 可以发现,无论相关 UCL 数目多少,UCL 聚焦操作的响应时间总能保持在 2.3 ms 以内,UCL 数目对响应时间仅有轻微影响.分析可知,这是因为在不同的条件下,算法需要进行的定位、剪枝操作基本一致,耗时也差异不大.图 9 结果表明 UDSCT 中基于树查询的 UCL 聚焦操作可快速响应用户的操作请求.

5 结 论

本文针对播存网络环境的现实需求,基于语义覆盖树提出一种 UCL 推荐多样性优化算法 UDSCT,可充分利用 UCL 语义信息及非语义用户评分信息,在保证 UCL 推荐精度的前提下,优化 UCL 推荐结果的多样性.首先,在多样性优化过程中,无需设定或训练任何加权参数,可基于 UCL 语义信息及非语义用户评分信息,取得稳定可靠的多样性优化效果;其次,在优化排序过程中,UCL 越新,其处理优先权就越高,从而有效提高 UCL 推荐列表的时效性;最后,UDSCT 算法收敛速度较快,一旦该 UCL 语义覆盖树构建完成,仅需要一次查询操作及少量推荐列表调整操作,便可得到多样化的 UCL 列表.实验结果证明,与基准算法相比,UDSCT 算法能够获得更好的效果,可有效满足播存网络环境的需求.

参 考 文 献

[1] Yang Peng, Li Youping. General architecture model of broadcast-storage network and its realization patterns [J]. Acta Electronica Sinica, 2015, 43(5): 974-979 (in Chinese)
(杨鹏,李幼平. 播存网络体系结构普适模型及实现模式[J]. 电子学报, 2015, 43(5): 974-979)

[2] Li Youping, Yang Peng. New mechanism for sharing cultural big data [J]. Communications of the CCF, 2013, 5(5): 36-40 (in Chinese)
(李幼平,杨鹏. 共享文化大数据的新机制[J]. 中国计算机学会通讯, 2013, 5(5): 36-40)

[3] Gu Liang, Yang Peng, Luo Junzhou. A collaborative filtering recommendation method for UCL in broadcast-storage network [J]. Journal of Computer Research and Development, 2015, 52(2): 475-486 (in Chinese)
(顾梁,杨鹏,罗军舟. 一种播存网络环境下的 UCL 协同过滤推荐方法[J]. 计算机研究与发展, 2015, 52(2): 475-486)

[4] Xing Ling, Ma Jianguo, Ma Weidong. Information Sharing Theory and Network Architecture [M]. Beijing: Science Press, 2011: 151-168 (in Chinese)
(邢玲,马建国,马卫东. 信息共享理论与网络体系结构[M]. 北京:科学出版社, 2011: 151-168)

[5] Vieira M R, Razente H L, Barioni M C, et al. On query result diversification [C] //Proc of the 27th IEEE Int Conf on Data Engineering (ICDE 2011). Piscataway, NJ: IEEE, 2011: 1163-1174

[6] Ashkan A, Kveton B, Berkovsky S, et al. Optimal greedy diversity for recommendation [C] //Proc of the 24th Int Conf on Artificial Intelligence. Menlo Park: AAAI, 2015: 1742-1748

[7] Ma Weidong, Li Youping, Ma Jianguo, et al. Empirical study of region user behaviors for Web pages [J]. Chinese Journal of Computers, 2008, 31(6): 960-967 (in Chinese)
(马卫东,李幼平,马建国,等. 面向 Web 网页的区域用户行为实证研究[J]. 计算机学报, 2008, 31(6): 960-967)

[8] Begtaševiü F, Van Mieghem P. Measurements of the hopcount in Internet [C] //Proc of the 2nd Int Conf on Passive and Active Measurement. Berlin: Springer, 2001: 23-24

[9] Adomavicius G, Kwon Y. Improving aggregate recommendation diversity using ranking-based techniques [J]. IEEE Trans on Knowledge and Data Engineering, 2012, 24(5): 896-911

[10] Carbonell J, Goldstein J. The use of MMR, diversity-based reranking for reordering documents and producing summaries [C] //Proc of the 21st Annual Int ACM SIGIR Conf on Research and Development in Information Retrieval. New York: ACM, 1998: 335-336

[11] Gollapudi S, Sharma A. An axiomatic approach for result diversification [C] //Proc of the 18th Int Conf on World Wide Web. New York: ACM, 2009: 381-390

[12] Ziegler C N, McNee S M, Konstan J A, et al. Improving recommendation lists through topic diversification [C] //Proc of the 14th Int Conf on World Wide Web. New York: ACM, 2005: 22-32

[13] Erkut E, Ulküsal Y, Yeniçerioğlu O. A comparison of p-dispersion heuristics [J]. Location Science, 1996, 21(10): 1103-1113

[14] Liu Ziyang, Sun Peng, Chen Yi. Structured search result differentiation [J]. Proceedings of the VLDB Endowment, 2009, 2(1): 313-324

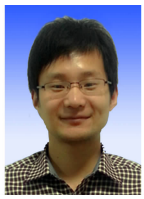
[15] Radlinski F, Kleinberg R, Joachims T. Learning diverse rankings with multi-armed bandits [C] //Proc of the 25th Int Conf on Machine learning. New York: ACM, 2008: 784-791

[16] Javari A, Jalili M. A probabilistic model to resolve diversity-accuracy challenge of recommendation systems [J]. Knowledge and Information Systems, 2015, 44(3): 609-627

[17] Qin Lijing, Zhu Xiaoyang. Promoting diversity in recommendation by entropy regularizer [C] //Proc of the 23rd Int Joint Conf on Artificial Intelligence. Menlo Park: AAAI, 2013: 2698-2704

[18] Drosou M, Pitoura E. Dynamic diversification of continuous data [C] //Proc of the 15th Int Conf on Extending Database Technology. New York: ACM, 2012: 216-227

[19] Beygelzimer A, Kakade S, Langford J. Cover trees for nearest neighbor [C] //Proc of the 23rd Int Conf on Machine learning. New York: ACM, 2006: 97-104



Gu Liang, born in 1989. PhD candidate. His main research interests include recommendation systems, machine learning, and future network architecture.

Yang Peng, born in 1975. Associate professor at the School of Computer Science and Engineering, Southeast University, China. His main research interests include future network architecture, information-centric networking, distributed computing, formal theories and techniques, etc.

Dong Yongqiang, born in 1973. Associate professor at the School of Computer Science and Engineering, Southeast University, China. His main research interests include future network architecture, information-centric networking, wireless ad hoc networks, and delay-tolerant opportunistic networking (dongyq@seu.edu.cn).