Hans Journal of Data Mining 数据挖掘, 2022, 12(4), 320-333
Published Online October 2022 in Hans. http://www.hanspub.org/journal/hjdm
https://doi.org/10.12677/hjdm.2022.124031

Hans汉斯

基于排名机制的领域Web网页发现

王安涛，李征宇，李　贵

沈阳建筑大学，信息与控制工程学院，辽宁 沈阳

收稿日期：2022年8月22日；录用日期：2022年9月22日；发布日期：2022年9月30日

摘　要

对很多Web数据集成应用来说，领域Web发现能力至关重要。从目前来看，现有的主题爬取策略依然有效，并随之产生了不少依据这些策略的主题爬虫，然而配置主题爬虫困难且费时，因此提出基于排名机制的领域Web网页发现算法，该算法在现有的主题爬取策略之上，利用给定的样本网页集，使用基于排名的方法，系统地结合多种Web网页发现策略，迭代发现并提取领域Web新网页。实验表明，该方法具有较高的网页准确率，验证了方法的有效性。

关键词

主题爬取，网页排名，领域Web网页发现

Domain Web Pages Discovery Based on Ranking Mechanism

Antao Wang, Zhengyu Li, Gui Li

School of Information & Control Engineering, Shenyang Jianzhu University, Shenyang Liaoning

Received: Aug. 22nd, 2022; accepted: Sep. 22nd, 2022; published: Sep. 30th, 2022

Abstract

Domain Web discovery capabilities are critical to many Web data integration applications. From the current point of view, the existing focused crawling strategies are still effective, and many focused crawlers based on these strategies have been created. However, configuring focused crawlers is difficult and time-consuming. Therefore, a domain Web page discovery algorithm based on ranking mechanism is proposed. Based on the existing focused crawling strategies, the algorithm uses a given set of sample web pages, uses a ranking-based method, and systematically combines various web page discovery strategies to iteratively discover and extract new web pages in the

文章引用: 王安涛, 李征宇, 李贵. 基于排名机制的领域 Web 网页发现[J]. 数据挖掘, 2022, 12(4): 320-333.
DOI: 10.12677/hjdm.2022.124031

domain. Experiments show that the method has high web page accuracy, which verifies the effectiveness of the method.

Keywords

Focused Crawling, Page Rank, Domain Web Pages Discovery

1. 引言

互联网高速发展的今天，对于网页信息的获取，一种很自然的想法是使用通用的商业搜索引擎，如：谷歌(Google)、百度(Baidu)、必应(Bing)。我们主要使用基于关键字的搜索以及类似站点的相关搜索，这就要求我们需要阅读大量搜索引擎返回给我们的检索信息，来提取结果。显而易见，对于大规模的数据收集任务，这种信息获取的方式是复杂且低效的。并且对于通常具有特定检索目的、特定领域、特定背景的用户来说，通用搜索引擎返回的结果可能包含大量无用的网页信息[1]。

为解决上述搜索引擎所出现的问题，近些年来也提出了不同的方法来解决这些问题：

1) 领域发现工具(Domain Discovery Tool, DDT) [2]：DDT 旨在简化对于给定领域构造分类器的过程，以帮助用户发现相关网页。它提供了一个易于使用的界面，这个界面总结了搜索结果并且帮助用户创建查询计划。

2) 前后爬取(Forward and Backward Crawling) [3]以及 DEXTER [4]：它们均用于自动发现 Web 网页内容。由于它们依赖域分类器，所以它们需要适应不同域。

为了解决这些问题：提出了基于排名机制的领域 Web 网页发现算法框架 DWDBRM (Domain Web pages Discovery Based on Ranking Mechanism)，DWDBRM 不依赖精确域分类器，该算法通过给出小部分有代表性的样本网页，就可以自动发现额外相关网页，这些网页不仅可以用于构造域分类器，而且可以充当主题爬取的种子网页集。它的主要贡献如下：

1) 自助领域发现需要精确域分类器，而该方法仅需要一组小样本有代表性的网页就可以自动发现相关 Web 网页，不需要精确域分类器；

2) 由于现有的各种独立网页排名方法适用于不同领域，并且存在效率和精度的差异，因此本文提出一种组合网页排名方法，组合多种独立排名函数，对网页进行排名；

3) 目前所做的发现工作均不支持多种搜索技术，因此本文在此框架中结合多种不同的搜索技术，主要做法是：在每轮迭代中使用 multi-armed bandits-based [5]策略来选择最佳搜索操作。实验表明，通过使用 multi-armed bandits-based 策略，我们的方法能够获得较高网页收获率(即相关网页或相关网站与检索的网页数量的比值)。

2. 相关工作

针对领域 Web 网页发现，已经提出过多种技术，这些技术大致可以分为两类：

1) 基本搜索发现技术(Search-based discovery techniques)：该技术依赖于搜索引擎 API (例如：Google、Bing、Baidu)来查找与给定关键字类似的 Web 网页。

2) 基本爬取发现技术(Crawling-based Discovery techniques)：该技术通过自动下载和递归的方式利用已有的 Web 网页链接来爬取新网页。

2.1. 基本搜索发现技术

关键字搜索和相关搜索这两种基本搜索发现操作是通过搜索引擎 API 来实现的，然而大多数现存的搜索技术使用关键字搜索。

文献[6] [7]使用网页内容与 Web 链接结构来计算 Web 网页之间的相关性。文献[8]提出了一个使用相关反馈的系统来收集种子网页以自助主题爬取：它提交关键字给 Bing 搜索引擎，从结果页中提取关键字，按域的相关性分类，并且使用这些关键字来构建新的搜索计划。这些研究表明：提交多种短查询以及检索小部分结果页与提交长查询检索更多的结果页相比，短查询及检索小部分结果页这种方式可以获取更多相关网页。由于它们的系统都是基于关键字搜索的，因此它们都不支持其他搜索技术。文献[4]提出了一个发现管道，它使用关键字搜索来查找包含产品规格的网页(比如：相机和计算机)，然而它们也使用反向(backward)搜索，不能应用于其他领域。

2.2. 基本爬取发现技术

向前爬取、向后爬取是两种基本的爬取操作：向前爬取的链接是从已发现的 Web 网页中提取。文献[9] [10] [11] [12]提出的方法都是在向前爬取的基础上来发现领域 Web 网页和网站。文献[9] [10] [13]最突出的方法是利用在线学习策略的主题爬取方法，使用两种分类器来集中搜索：1) critic 分类器：将网页分为相关和不相关的两个域，但是该分类器需要大量的正实例和负实例来训练；2) apprentice 分类器：是自动学习的，以在线学习方式，查找与域相关的网页最相关的链接。

向后爬取(或逆向爬取)通过反向链接来发现新网页，由于 Web 是单向的，它必须通过搜索引擎 API 来搜索之前的大规模的爬取结果。文献[3]提出了一种结合向前和向后爬取技术来查找双语网站的爬取策略，它使用了两个链接分类器，来训练访问过的网页内容以及它们的相关目标域，通过结合这两种技术来确定访问发现链接的次序。

综上所述，虽然这些技术可以用于查找特定领域的 Web 网页，但之前所做的工作均不支持多个不同的发现技术，并且大多数技术需要一个精确的域分类器，并且当分类器不准确时，会变得无效，而我们的方法不需要精确的域分类器，只需要小部分相关网页即可。

3. 问题及解决方案概述

3.1. 问题定义

定义 1. 网页发现问题。D 表示兴趣域，w 表示一个网页，如果 w 与兴趣域相关，则称 $w \in D$。给定网页种子集 S，使得 $S \subset D$，本文目标就是利用已发现网页 w_+且 $w_+ \in D$，来扩充网页种子集 S。

定义 2. 网页重要性排名问题。R 表示 Web 网页的排名值，指定 u 表示单个 Web 网页；令 F_u 为 u 所指向的页面集，B_u 为指向 u 的页面集；$N_u = |F_u|$ 为 u 所指向的链接数量，c 为标准化系数，基本定义如下：

$$R(u) = c\sum_{v \in B_u} \frac{R(v)}{N_v} \tag{1}$$

定义 3. 网页相关性排名问题。在缺少识别 D 中网页模型的情况下，通过指定网页种子集 S 作为近似的排名任务问题：给出 S，目标是发现与 S 类似的新网页。即，给定一组已发现的网页集 D_+，利用网

页相关性排名方法，计算已发现网页 w_+与 S 的相似度分数来对 D_+中的网页 w_+进行排名，排名位置越靠前，则 w_+与集合 S 中的网页越相似， $w_+ \in D$ 的可能性就越大。

3.2. 框架概述

DWDBRM 框架采用领域 Web 网页发现、领域 Web 网页排名这两种策略来执行迭代，从提交查询给搜索引擎开始，基于领域知识来对搜索结果进行评分，制定新的查询计划。DWDBRM 框架如图 1 所示：

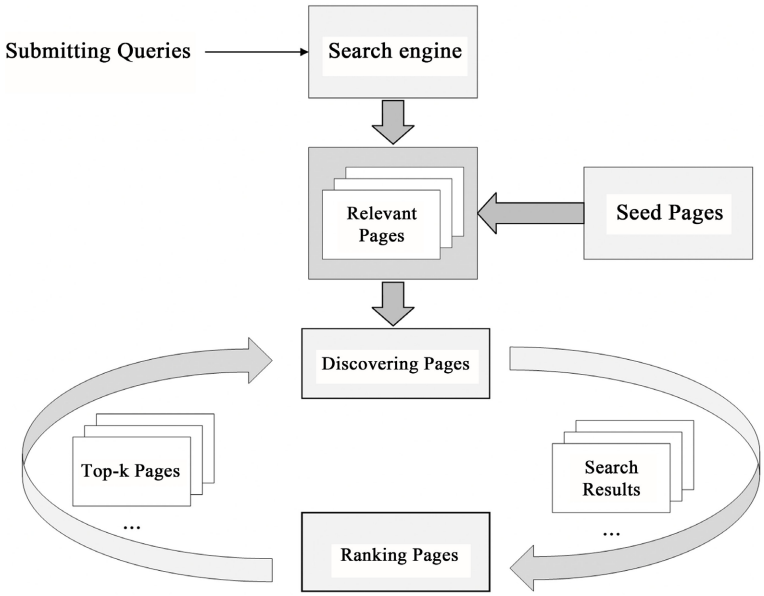

Figure 1. Architecture of DWDBRM
图 1. DWDBRM 结构

1) 该框架使用搜索引擎 API 来实现 Web 网页搜索操作。

2) 该框架由网页排名和网页发现两个主要部分组成。同时为了整合搜索引擎 API 以及排名方法，需要标准化输入输出这两个部分：网页发现部分最开始将具有代表性的相关网页种子集作为输入，执行搜索操作，执行完最开始的一轮迭代后，选取符合条件的新的上一轮迭代排名后的网页结果列表作为新的网页种子集输入，继续执行搜索操作，输出新发现的网页列表集；网页排名部分将上步网页发现部分发现的网页列表作为输入，输出对应的网页排名列表。

3) 该框架统一了多个搜索操作，并且可以公平地比较搜索操作在某些领域的有效性。并且本框架可以减少人工审查网页实例和标记网页实例来构建域分类器的工作量。

DWDBRM 从种子网页集开始迭代搜索、排名以及选择最佳结果，基本算法框架与流程如算法 1 所示(其中 KS 为 KeywordSearch，RS 为 RelatedSearch，FC 为 ForwardCrawling，BC 为 BackwardCrawling)：

算法 1：Page Crawling
输入：PageSeeds (相关网页种子集)
输出：RankedResults (排名后的结果网页列表)

1) **procedure** Crawling (PageSeeds)
2) results = ∅

Continued

```
3)      RankedResults = ∅
4)      topk = PageSeeds
5)      preRankedResults = ∅
6)      operators = {(KS, score), (RS, score), (FC, score), (BC, score)}
7)      op = operators.random_op
8)      while results.length < pageNumberCondition
9)          TempResults = start_crawl(op, topk)
10)         results = results ∪ TempResults
11)         RankedResults = CombinedFunctionRank(results, α)
12)         topk = get_top_k(RankedResults)
13)         op = select_op(operators, op, preRankedResults, RankedResults)
14)         preRankedResults = RankedResults
15)     return RankedResults
```

- 用一个参数 PageSeeds 调用该算法。我们称 PageSeeds 为事先准备的种子网页集。
- 算法开始阶段分别初始化爬取结果网页列表 results、排名的结果网页列表 RankedResults、先前排名的结果网页列表 preRankedResults 为空，初始化 op 操作为发现操作集 operators 中的任意一个操作，指定每次迭代的爬取网页数量条件(如：爬取 5000 个网页后，停止算法执行，返回最终的排名网页集 RankedResults) (行 2~行 8)。
- 开始本轮网页爬取，爬取后得到临时搜索结果列表 TempResults (行 9)，合并(不去重)结果列表 results 与 TempResults (行 10)。
- 传入评分系数 α 并结合网页重要性与网页相关性排名算法(见算法 2)对上步 results 进行排名，得到每个网页的最终排名分数，返回最终网页排名结果列表 RankedResults (行 11)。
- 利用 top-k 查询机制，获取网页排名结果列表 RankedResults 中最好的具有代表性的前 k 个网页，作为下一轮迭代爬取的网页种子集(行 12)。
- 接着 select_op()函数(见算法 3)对目前现有的发现操作集 operators 中的四种操作进行评分，更新操作评分，选择当前迭代中的得分最高者操作，并赋值给变量 op，用于执行下一轮爬取，更新 preRankedResults，接着跳回到行 8，判断循环条件，执行下一轮迭代，否则执行(行 15)。
- 迭代获取的网页数量达到设定值，终止迭代，返回最终排名网页列表 RankedResults (行 15)。

4. 基于排名与发现操作的网页爬取

本节分别给出基于组合排名函数策略的排名算法与基于 multi-armed bandits-based 策略的发现操作选择算法。

4.1. 网页排名

网页排名问题可以看作是信息检索中的常规排名问题，其中 S 代表网页种子集，网页相关性排名的目标是利用 D_+中新发现的网页 w_+与 S 的相似度来对 D_+中的网页 w_+进行排名；网页重要性排名的目标是将有价值的、重要的网页赋予高分进行排名。

4.1.1. 基于回归排名

该方法使用回归排名(Regression-Based Ranking)来学习排名函数，并计算每个网页 w_+在 D_+中的分数。

因此需要使用正实例和负实例。选取负实例的具体做法是：从公共的 Web 网页库中随机选择网页来获得负实例。由于 Web 网页库相对于样本页面的数量来说比较大，因此减少了选择与该领域相关的 Web 页面的概率；训练分类器的具体做法是：利用网页种子集 S、负实例拟合训练集，真实爬取的网页作为测试集，以此来预测分类概率，来计算排名分数。本文使用逻辑回归(logistic regression, LR)来进行回归排名，我们将相关类归为 0 类，不相关类归为 1 类，相关类计算公式定义如下：

$$Score(w)=\frac{1}{1+e^{z}} \tag{2}$$

其中 $z=w*w_{LR}^{T}+b$ 为线性预测器，参数 w_{LR} 与参数 b 为逻辑回归模型的参数。

另外一种非常规的技术就是新颖性检测(Novelty Detection)：该技术不需要正实例和负实例就可以对所属的类别进行预测，例如：考虑一个由 p 个特征描述的相同分布的 n 个观测值组成的数据集，通常来说，它将学习一个粗略地、紧密地边界来构造初始观测分布的轮廓，绘制并嵌入进 p 维空间中；如果观测值位于边界划分的子空间内，则这组观测值与初始观测值相同，可以划为相同类。否则，观测值是异常的，划为异常类。

本文分别使用 sklearn 提供的机器学习核心算法库 linear_model、SVM 中的 LogisticRegression、OneClassSVM 来进行网页的排名。

4.1.2. 基于相似度排名

基于相似度排名受 k-最近邻算法启发，根据网页种子集 S 中的所有网页与新发现的网页的相似度的平均值计算出一个网页的排名分数。在网页 w_i 与 w_j 之间给出一个相似度函数 $Sim(w_i,w_j)$，则 $w\in D_+$ 的网页分数计算如下：

$$Score(w)=\frac{1}{|S|}\sum_{w_s\in S}Sim(w,w_s) \tag{3}$$

考虑使用 cos 和 Jaccard 作为相似度函数，这些方法可以用于找出输入网页种子集 S 的不同排名优势，由于它们使用网页的不同表示，它们可以在排名任务中暴露出不同的排名特征。所以，我们为了计算 cos 和 Jaccard 相似度，cos 使用网页关键字词频向量空间模型表示网页，Jaccard 使用关键字是否存在关键字这种二元属性组成的二元向量表示网页。cos 和 Jaccard 相似度函数可以作为 $Sim()$函数，来分别计算上述 Score 分数，cos 和 Jaccard 相似度函数计算公式分别如下：

$$\text{Jaccard}(x,y)=\frac{x\cap y}{x\cup y} \tag{4}$$

$$\text{Cosine}(x,y)=\frac{\sum_{i=1}^{n}x_iy_i}{\sqrt{\sum_{i=1}^{n}x_i^2}\sqrt{\sum_{i=1}^{n}y_i^2}} \tag{5}$$

其中 x,y 来分别代表网页 w_x,w_y。

4.1.3. 基于贝叶斯排名

给出一系列属于类 C 的相关网页集，通过评估属于类 C 的项目的标准化概率来为新发现的网页打分。通过这种定义，把相关项目看成网页种子集 S，类 C 看做兴趣域 D，这样贝叶斯的分类策略可以直接应用到我们的排名问题中来。对应地，通过预测得到的概率，可以作为网页 w_+与网页种子集 S 的相似的分数，评分计算公式如下：

$$Score(w)=\frac{P(w|S)P(S)}{P(w)} \tag{6}$$

4.1.4. PageRank 排名

PageRank，又称网页排名、PR，是 Google 公司所使用的对其搜索引擎搜索结果中的网页进行排名的一种算法。该算法与前几个算法思想不同，它是根据网页的重要性进行排名，即被越多网页所指向的网页，该网页的排名就靠前。PageRank 算法计算公式如下：

$$R(w)=c\sum_{v\in B_w}\frac{R(v)}{N_v}+cE(w) \tag{7}$$

其中 $E(w)$ 为 Web 网页对应的源排名向量。实验表明，对于刚开始的网页的爬取，由于爬取的网页数量较少，很难形成有效的 PageRank 分数，随着爬取的网页数量增加，排名效果会得到改善。

4.1.5. 基于组合函数策略的网页排名

有很多可供选择的排名函数，但不同的排名函数对同一个领域的网页排名结果各有差异，排名结果有好有坏，也就是说，使用不同的排名函数对同一个领域排名，所得出的排名结果不同。因此，为了更好的得出网页排名结果，采用组合排名函数策略，结合多种排名函数得出的网页排名结果列表进行网页排名的合并，实验表明，这种策略可以增强排名的鲁棒性、减少排名的方差。组合排名函数评分计算公式如下：

$$Score(w)=(1-\alpha)\frac{\sum_{f_i\in F}f_i(w)}{|F|}+\alpha R(w) \tag{8}$$

其中，F 为以上小节种涉及到的回归排名、相似度排名、贝叶斯这三类排名函数列表，$f_i(w)$ 指示 w 所属于的第 i 个排名函数所得出的该网页在排名列表的位置。R 为 PageRank 排名得到网页排名的位置，α 为评分系数，分数越小说明排名越靠前，越相关。实验发现，均值函数应用在本文房产领域表现好、速度快。网页排名算法框架与流程如算法 2 所示(本文评分系数 α 取 0.2)：

算法 2：Rank Pages (网页排名算法)
输入：results (某轮迭代发现的网页列表)，α (评分系数)
输出：RankedResults (排名后的结果网页列表)

```
1)  procedure CombinedFunctionRank (results, α)
2)      combinedF_results = ∅, RankedResults = ∅
3)      n_results = remove_duplicated_values(results)
4)      The n_results are calculated respectively and the
5)              results BS_rank_reuslts, LR_rank_reuslts
6)              OneClass_reuslts, cos_results, jaccard_results
7)              are obtained respectively.
8)      pagerank_results = PageRank(n_results)
9)      foreach n_results
10)         sum_posi = calculate the sum_pos of n_results[i] in
11)                                 every ranked_reuslts
12)         pr_posi = find pos of n_result[i] in pagerank_results
13)         combinedF_results.add((1 − α)*(sum_posi)/5 + α*pr_posi)
14)     RankedResults = sort(combinedF_results)
15)     return RankedResults
```

- 首先初始化 combinedF_results，RankedResults 为空，并对 results 进行去重处理得到 n_results (行 2~行 3)；
- 使用一系列的排名函数对爬取的结果网页列表 n_results 分别进行排名，得到各排名结果列表 BS_rank_reuslts，LR_rank_reuslts，OneClass_reuslts，cos_results，jaccard_results (行 4~行 7)，而后进行 PageRank 排名(行 8)；
- 遍历 n_results 中的每个网页，找到每个网页在各排名结果网页列表中的位置，并计算各网页的排名分数，根据评分系数 α 赋予网页重要性与相关性的分数权重，并将计算得到的某网页的排名分数加入到结果列表 combinedF_results 中(行 9~行 13)；
- n_results 遍历结束，根据排名分数对 combinedF_results 进行排序并赋值给 RankedResults (行 14)；
- 返回 RankedResults (行 15)。

4.2. 网页发现操作

网页发现操作的目标就是发现新网页，因此可以利用现存的网页发现操作来爬取目标网页。实验测试了四种不同的发现操作，为了在算法框架中整合这些发现操作，本文输入部分采用标准化的网页种子集 PageSeeds，输出部分采用标准化的利用 top-k 查询策略获取到的排名后的最相关的网页列表。

1) 向前爬取(Forward Crawling)：该操作是通过输入网页的 HTML 文本内容提取的链接来发现新网页。一个需要注意的问题是：从输入网页中的正文文本提取的关键字有好坏之分，所以为了能够获取特定领域的关键字来生成更相关的网页的方法是：从元数据标签中提取关键字(即描述(description)、关键字(keyword))，这比从正文中提取的关键字噪声要小得多。在搜索相关网页时，对于每一个网页，利用描述、关键字标签(tag)提取、标记元数据，选择最频繁使用的标记作为搜索关键字候选。本文使用每个网页头部标签<title></title>来判断是否为相关网页的一个特征。

2) 向后爬取(Backward Crawling)：向后爬取采用反向链接搜索、向前爬取这两个操作来发现新的网页。反向链接搜索操作通过提交链接(link)给搜索引擎，并调用搜索引擎 API 来获取输入网页的反向链接。由于输入与特定域相关，发现的反向链接可能充当指向相关网页的中心，因此可以从中心网页提取的链接和向前爬取操作来发现新的额外地相关网页。

3) 关键字搜索(Keyword Search)：关键字搜索使用搜索引擎 API 来搜索与关键字相关的网页。需要注意的问题是：关键字与域相关，但不能包含足够的信息来检索相关网页，例如关键字：产品(products)、物业(properties)、沈阳(Shenyang)、售卖(sale)都与房地产相关，但是从上下文来看，它需要构建发现额外相关网页的查询。为了解决这个问题，考虑将关键字与域的描述结合起来，来检索更多相关网页，这不需要增加额外的负担。比如：在房地产领域，给出房地产关键字作为种子关键字，对应的一些关键字条件是房地产产品、房地产物业、房地产沈阳、房地产售卖。

4) 相关搜索(Related Search)：相关搜索操作以网页 URL 作为输入，并返回相关的网页列表，Google 搜索 API、BingAPI 支持这个操作。根据这些搜索平台具有的数据性质以及它们的底层算法，结果可能会有很大不同。然而，这种搜索方式很适合发现相关网页。

发现操作选择

考虑在算法 1 的每次迭代中选择一个发现操作来最大限度地发现相关网页。从某些方面来说，使用不同的发现操作，可以多样化发现过程，最终可以提高相关网页的收获率。

为了解决这个问题，提出策略 multi-armed bandits (MAB)，特别是：UCB1 算法[5]，一个在不同网页发现操作中的权衡策略，MAB 被认为是解决开发勘探问题的有效手段，简单地说就是，选择有高积累的

奖赏的强盗武器，惩罚那些过去过度使用的那些强盗武器(即，选择具有高收获率的网页操作，抛弃掉那些过度使用的高收获率的网页操作，在网页发现操作的选择中进行权衡)。为了实现该目标：我们需要定义两个变量 op,μ_{op}，分别代表强盗手臂以及对应操作的奖赏(op 代表发现操作，μ_{op} 代表本次迭代选择的发现操作发现新的相关网页的评分，发现的新相关网页越多，μ_{op} 值越大)。如果使用某种发现操作 op 返回了 k 个网页，那么对于此操作的奖赏 μ_{op} 定义如下：

$$\mu_{op}=\frac{\sum_{i=1}^{k}\left(1-\frac{pos_i}{len}\right)*I_i}{k} \tag{9}$$

其中：

① I_i 采用二进制值：

$$I_i=\begin{cases}0 & \text{网页}i\text{已经被发现}\\ 1 & \text{否则}\end{cases} \tag{10}$$

② pos_i 指示网页 i 在排名列表中的位置，i 的标准奖赏是 $1-\frac{pos_i}{len}$，len 为排名列表的长度。

需要注意的是：虽然我们不知道新发现的网页相关性，但我们可以通过网页排名函数计算出的排名分数来近似它们。通过这种定义，如果 i 在网页列表的底部或者顶部，那么 I_i 将被分别赋值 0 或 1。由于目标是最大化新的相关网页的数量，我们给先前迭代已经发现的网页赋值为 0，而不管它所处的位置如何。因此，i 的奖赏被定义为 $\left(1-\frac{pos_i}{len}\right)*I_i$。

最后，我们把每一个发现操作模拟为一个强盗手臂。假定在每个时间 t，当我们选择了四个操作中的一个操作，在 op,t 下发现了 $n_{op,t}$ 个网页，根据算法 UCB1，op 在时间 t 的分数被定义如下：

$$Score_{op,t}=\overline{\mu}_{op}+\sqrt{\frac{2\ln(n)}{n_{op,t}}} \tag{11}$$

其中：$\overline{\mu}_{op}$ 为某个操作 op 在历次迭代中，计算出的平均值。

在这个等式中，n 为所有操作的检索网页的总数量。在每次迭代中，算法选择得分最高的操作，来执行。

我们使用这些公式来计算 $Score_{op,t}$。在每一轮搜索后，重新计算分数并且选择高分发现操作进行下一轮迭代。采取这种策略，可以明显提高网页收获率。选择操作算法框架与流程算法 3 所示：

算法 3：Select Operator (UCB1 算法实现)

输入：operators，op，preRankedResults，RankedResults

输出：maxScore_op (最高分的某个操作)

```
1) procedure select_op (operators, op, preRankedResults, RankedResults)
2)     I_i = 0
3)     μ_op = 0.0, len = RankedResults.length
4)     for each every link in RankedResults
5)         if link.search(preRankedResults)
6)             or link.index(RankedResults) = = len
7)                 I_i = 0, μ_op+ = 0.0
8)         else if link.index(RankedResults) = = 0
```

Continued

```
9)          I_i = 1, μ_op+ = Ii
10)     else
11)         I_i = 1
12)         μ_op+ = (1 - link.index(RankedResults)/len) * I_i
13)   μ_op = μ_op/len
14)   update(operators, op, len, len-preRankedResults.length, μ_op)
15)   maxScore_op = maxScore(operators)
16)   return maxScore_op
```

- 初始化 I_i 为 0、 μ_{op} 为 0.0，当前迭代排名结果列表 *len* 为 RankedResults.length (行 2~行 3)；
- 循环遍历当前排名结果列表 RankedResults 中的每个 link，如果在先前迭代产生的排名结果列表 preRankedResults 中的网页在新产生的排名结果列表 RankedResults 中再次出现，或者该网页在当前排名结果列表 RankedResults 的尾部，则 $I_i = 0$，并计算 μ_{op} (行 5~行 7)，否则继续执行第 9 行语句；
- 如果该网页处于排名结果列表 RankedResults 的头部，则 $I_i = 1$，并计算 μ_{op} (行 8~行 9)，否则执行第 11 行语句；
- 如果 RankedResults 中的网页不在先前的排名结果列表 preRankedResults 中，还有既不在 preRankedResults 的列表头部，也不在 preRankedResults 列表的尾部，则 $I_i = 1$，并计算 μ_{op} (行 12)；
- RankedResults 遍历结束，计算最终的 μ_{op} 分数(行 13)；
- 传入相关参数(其中 *len* 为本轮迭代排名结果列表的长度(即 *n*)，len-preRankedResults.length 为本轮迭代新发现的网站数量(即 $n_{op,t}$))，然后将本轮迭代得到的当前操作评分进行更新(即计算 $Score_{op,t}$) (行 14)；
- 找到目前为止评分最高的发现操作赋值给 maxScore_op (行 15)；
- 返回评分最高操作 maxScore_op (行 16)。

5. 实验

为了验证本文提出的基于排名机制的领域 Web 网页发现算法框架的可行性和有效性，本文使用 python 语言实现算法框架设计。

实验数据选取具有代表性的商业地产网页：房天下、安居客、链家网等 20 多个种子网页作为种子网页集，使用 Bing 搜索引擎来完成部分网页发现工作。具体流程是：每轮迭代后，使用 top-k 查询机制生成新的种子网页列表集，然后利用种子网页列表集爬取新的相关网页，之后将爬取的网页列表与评分系数 α 作为相关排名函数的参数，进行网页排名，选择高分发现操作进行下一轮迭代，反复进行这个迭代过程，直到达到网页数量爬取条件，在达到网页数量爬取条件后，我们将得到的结果网页列表用于发现与排名的评估。

5.1. 评估方法

网页发现评估指标使用：网页/网站收获率(Harvest Rate, HR)对网页/网站发现方法进行评估，HR 计算公式如下：

$$\mathrm{HR} = \frac{\text{relevant\_webs}}{\text{retrieved\_webs}} \tag{12}$$

网页排名评估指标使用准确率(P@N)、平均数排名(Mean Rank)以及中位数排名(Median Rank)来对网页排名进行评估：

① P@N 为在前 N 个排名结果网页列表中的相关网页数与前 N 个排名结果网页列表的比值，其计算如下公式：

$$\mathrm{P@N} = \frac{\mathrm{relevant\_webs} \cap \mathrm{get\_ranked\_resuls}(N)}{\mathrm{get\_ranked\_resuls}(N)} \tag{13}$$

② Mean Rank 为排名结果相关网页列表中的相关网站的排名位置的平均值，其计算公式如下：

$$\mathrm{MR} = \frac{1}{n} \sum_{w_i \in D_+^*} \mathrm{position}(w_i) \tag{14}$$

其中 D_+^* 为属于兴趣域 D 的发现的网站列表，n 为相关网页列表长度。

③ Median Rank 为排名结果网页列表中，在前 N 个相关网页中的第 $\frac{N}{2}$ 个相关网页在结果排名列表中的位置，其计算公式如下：

$$\mathrm{MidR} = \mathrm{mid\_relevant\_web\_position}\left(\mathrm{get\_n\_relevant\_webs}(N)\right) \tag{15}$$

5.2. 实验结果与分析

5.2.1. 网页发现评估

本文使用自适应主题爬取工具 ACHE 作为对照试验，验证本文发现操作算法(算法 3)的有效性。ACHE 的爬取策略主要是利用在线学习爬取策略来优先获取未被访问的链接，它可以用于主题爬取。由于 ACHE 是尽可能地爬取相关网页，因此 ACHE 爬取相关网站的数量会很少，我们施加限制条件：每个网站最多爬取 20 个网页；但这种限制，会导致某些相关网页丢失并且不会被爬取工具 ACHE 发现，甚至 ACHE 永远不会爬取它，这还会产生较低地网页收获率、链接用完提前停止运行的弊端。

本实验我们使用真实房地产数据来评估网页发现操作算法，为了使实验比较结果更加清晰，对于网站发现收获率的计算判断标准是：只有属于房产类，且网址以顶级域名结尾，才算是相关网站。另外，对于每种发现方法，爬取网页数量的目标是：results 在检索 10,000 个网页停止(为便于计算，将爬取到的 10,000 个网页五等分，等分的每部分网页进行去重，并且去除第 $i+1$ 部分中 i 中已包含的旧网页，计算并保存每种爬取方法的网页/网站的收获率)；采用关键字搜索、相关搜索、正向爬取、向后爬取来获得网页，网页收获率与网站收获率(为便于观察，纵轴值与横轴值的比值即为收获率)实验结果图分别如图 2、图 3 所示：

实验开始时，在 BANDIT 方法中，我们赋予四种发现操作同样的高分分数，每种操作分别进行一轮迭代并分别评分，之后根据评分选择得分高者操作用于下一轮迭代。

从图 2 可以看出，在区间[0, 4000]向后爬取收获率优于其他方法，但整体上 BANDIT 收获率仅次于关键字搜索，除了关键字搜索，BANDIT 性能要优于其他搜索方法。

从图 3 可以看出，关键字搜索相关网站在前期收获率较高，因为搜索引擎在前期会检索最相关的网页，随着爬取的网站数量越多，它很快会达到平稳状态；BANDIT 充分利用了四种发现、爬取操作的优势，保持一种网页发现的稳定状态，发现新网页，而且性能要比各种搜索发现、爬取发现技术稳定。

综上所述 BANDIT 的网页发现、网站发现性能稳定，网页发现操作选择算法可行有效。

5.2.2. 网页排名评估

本实验组合排名函数的评分系数 α 取 0.2，需要注意的是：由于每轮迭代爬取的网页数量较少，因此

很难形成有效的 PageRank 分数，随着每轮迭代积累的爬取网页数量的增加，PageRank 排名可以筛选出部分重要的网页并将其排在前列。

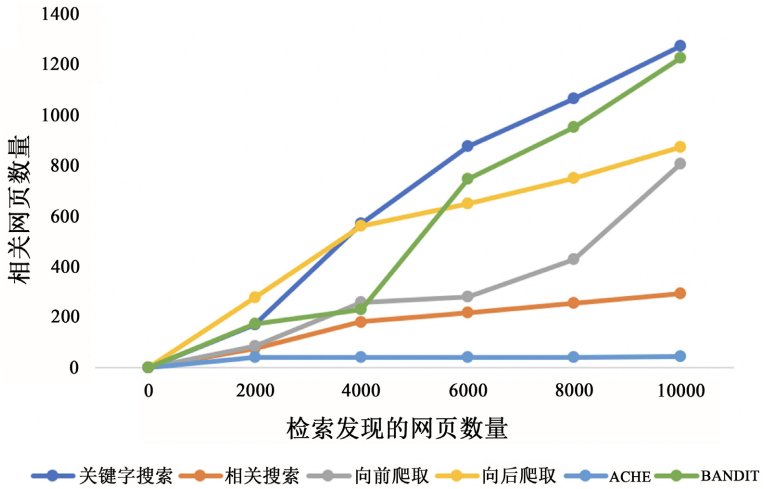

Figure 2. Web Pages harvest rate
图 2. 网页收获率

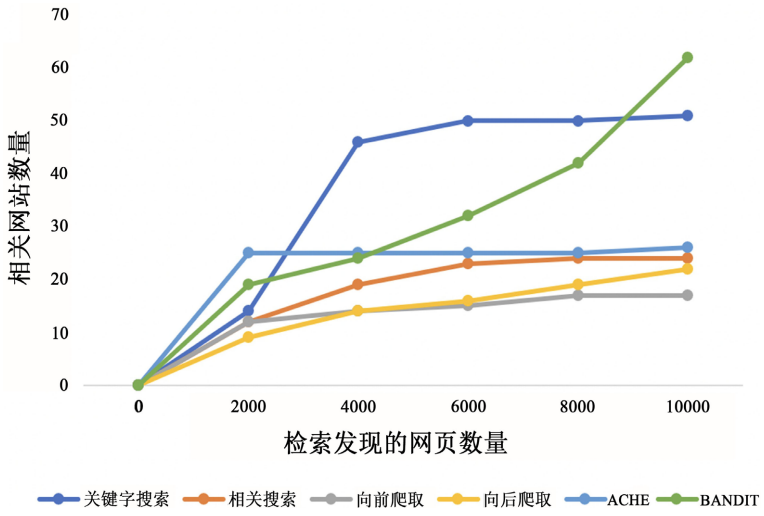

Figure 3. WebSites harvest rate
图 3. 网站收获率

本文 Naive Bayesian (以下简称 NB)使用 url 长度、url 域名长度、关键字计数等属性作为分类属性、OneClassSVM (以下简称 OneClass)使用关键字词频、是否存在关键字这种二元属性等属性作为分类属性；logistic regression (以下简称 LR)使用是否存在关键字这种二元属性作为分类属性。

五种相似性排名函数以及我们提出的 CombinedFuntionRank 函数(以下简称 CombinedF)的排名评估指标 P@N 准确率、Mean Rank 分数、Mean Rank 分数，分别如下表 1、表 2 所示(以下评估为对爬取后的 10,000 个结果网页列表 results 去重后得到的网页列表长度为 4242 的评估结果)：

P@N 值越大，表示相关网页的准确率越高。从表 1 可知：在 $N = 17$、$N = 35$ 的情况下，我们提出的排名函数 CombinedF 具有较高的准确率，优于其他方法。

Table 1. P@N evaluation
表 1. P@N 评估

评价方法 P@N	NB	LR	OneClass	Cosine	Jaccard	CombinedF
***N* = 17**	0.882	0.706	0.765	0.412	0.647	0.941
***N* = 35**	0.800	0.629	0.857	0.457	0.686	0.886

Table 2. Mean Rank and Median Rank
表 2. Mean Rank 与 Median Rank

评价方法	NB	LR	OneClass	Cosine	Jaccard	CombinedF
Mean Rank	89.96	92.14	105.93	77.29	77.14	80.95
Median Rank	9	12	12	18	12	9

Mean Rank、Median Rank 的值越小表示相关网页的位置越靠前，其中 Median Rank 的结果为从索引为 0 的网站开始算起，相关网页数 $n = 17$ 的中间位置的结果；

从表 2 可以得知：我们提出的排名函数 CombinedF 在 Mean Rank 的排名位置平均值要差于 Cosine、Jaccard 方法，但优于其他方法，Cosine 与 Jaccard 分数高的原因是：我们最初迭代选取的种子网页为一系列相关网站；在 MedianRank 中，我们提出的排名函数 CombinedF 在相关网页的中间位置与 NB 方法持平，除此以外，要比其他方法的位置要更靠前。

综上所述，我们提出的基于排名机制的 Web 网页发现算法框架 DWDBRM 是可行有效的。

6. 结论

本文提出了基于排名机制的 Web 网页发现算法框架 DWDBRM，该算法系统地结合多种 Web 网页发现策略，迭代发现提取领域 Web 新网页。结合网页相关性以及网页重要性对网页排名，提高了爬取相关网页的准确率，最后通过实验，验证了本文所提算法框架的有效性可行性。

参考文献

[1] 汤羽, 林迪, 范爱华, 吴薇薇. 大数据分析与计算[M]. 北京: 清华大学出版社, 2018.

[2] Krishnamurthy, Y., Pham, K., Santos, A., and Freire, J. (2016) Interactive Exploration for Domain Discovery on the Web. *ACM KDD Workshop on Interactive Data Exploration and Analytics* (*IDEA*), 64-71. https://nyuscholars.nyu.edu/en/publications/interactive-exploration-for-domain-discovery-on-the-web

[3] Barbosa, L., Bangalore, S., and Sridhar, V.K.R. (2011) Crawling Back and Forth: Using Back and Out Links to Locate Bilingual Sites. *Proceedings of* 5*th International Joint Conference on Natural Language Processing*, Chiang Mai, 8-13 November 2011, 429-437. https://aclanthology.org/I11-1048

[4] Qiu, D.S., Barbosa, L., Dong, X.L., Shen, Y.Y., and Srivastava, D, (2015) Dexter: Large-Scale Discovery and Extraction of Product Specifications on the Web. Proc. *Proceedings of the VLDB Endowment*, **8**, 2194-2205.

[5] Auer, P., Cesa-Bianchi, N., and Fischer, P. (2002) Finite-Time Analysis of the Multiarmed Bandit Problem. *Machine Learning*, **47**, 235-256. https://doi.org/10.14778/2831360.2831372

[6] Dean, J. and Henzinger, M.R. (1999) Finding Related Pages in the World Wide Web. *Computer Networks* 31, **11**, 1467-1479.

[7] Murata, T. (2001) Finding Related Web Pages Based on Connectivity Information from a Search Engine. *Poster Proceedings of* 10*th International Conference on World Wide Web* (*WWW*), Hong Kong, 1-5 May 2001, 18-19. http://www10.org/cdrom/posters/frame.html

[8] Vieira, K., Barbosa, L., Silva, A.S., Freire, J., and Moura, E. (2016) Finding Seeds to Bootstrap Focused Crawlers. *World Wide Web*, **19**, 449-474. https://doi.org/10.1007/s11280-015-0331-7

[9] Barbosa, L. and Freire, J. (2007) An Adaptive Crawler for Locating Hidden-Web Entry Points. In *Proceedings of the*

16*th International Conference on World Wide Web* (*WWW*), New York, 8 May 2007, 441-450. https://doi.org/10.1145/1242572.1242632

[10] Chakrabarti, S., Punera, K., and Subramanyam, M. (2002) Accelerated Focused Crawling through Online Relevance Feedback. In *Proceedings of the* 11*th International Conference on World Wide Web* (*WWW*), New York, 7 May 2002, 148-159. https://doi.org/10.1145/511446.511466

[11] Chakrabarti, S., van den Berg, M., and Dom, B. (1999) Focused Crawling: A New Approach to Topic-Specific Web Resource Discovery. *Computer Networks*, **31**, 1623-1640. https://doi.org/10.1016/S1389-1286(99)00052-3

[12] Ester, M., Kriegel, H.-P., and Schubert, M. (2004) Accurate and Efficient Crawling for Relevant Websites. In *Proceedings of the Thirtieth International Conference on very Large Data Bases* (*VLDB*), Toronto, 31 August-3 September 2004, 396-407. https://doi.org/10.1016/B978-012088469-8.50037-1

[13] Meusel, R., Mika, P., and Blanco, R. (2014) Focused Crawling for Structured Data. In *Proceedings of the* 23*rd ACM International Conference on Conference on Information and Knowledge Management* (*CIKM*), New York, 3 November 2014, 1039-1048. https://doi.org/10.1145/2661829.2661902