Web Structure Management기법을 이용한 Spamming page filtering algorithm

신광섭[0], 이우기[*], 강석호
서울대학교 산업공학과[0], 성결대학교 컴퓨터공학부[*]
whatever@ara.snu.ac.kr[0], wook@sungkyul.edu[*], shkang@cybernet.snu.ac.kr

Spamming page filtering algorithm using Web structure management management

Kwangsup Shin,[0] Lee Wookey, Sukho Kang
Dept. of Industrial Engineering Seoul National University, Dept. of Computer Science Sungkyul University[*]

Abstract

정보 통신 기술의 발달로 엄청난 양의 정보가 World Wide Web을 통해 저장되고 공유된다. 특히, 사용자가 WWW을 이용하여 필요한 정보를 얻고자할 때, 가장 많이 사용되는 것이 Web search engine이다. 그러나 Web search engine의 algorithm 자체의 부정확성과 악의적으로 작성된 Web page로 인해 search engine 결과가 사용자의 요구와 일치하지 못하는 문제가 발생한다. 본 논문에서는 여러 Web search algorithm 중에서 Web structure management 기법을 중심으로 문제점을 분석하고 이를 해결할 수 있는 수정된 algorithm을 제시한다. 마지막으로 제시된 algorithm이 spamming page를 filtering하는 과정을 예시하여 논증한다.

1. Introduction

정보 통신 기술의 발달로 인터넷을 이용해 수많은 사용자들은 자신이 가지고 있는 정보를 Web page의 형태로 저장하거나, 새로운 정보를 얻기 위해서 Web page를 검색할 수 있게 되었다. 이러한 과정을 Information access process라하며 사용자가 Web search engine에 자신이 얻고자하는 정보를 표현하는 질의어를 입력하는 것으로 시작된다[1]. 대부분의 Web search engine이 가지고 있는 가장 큰 문제점은 질의어에 대한 engine의 결과가 사용자의 요구와 일치하기 어렵다는 점이다[2]. 이런 문제의 원인은 검색 알고리즘 자체의 부정확함과 악의적으로 작성된 web page에서 찾을 수 있다.

Web search engine에서 사용하는 algorithm은 크게 Web contents management, Web structure management 그리고 Web usage management으로 구분할 수 있다[3]. 초기 Web search engine에서 사용하는 방법은 page 자신이 가진 contents를 중심으로 하는 Web contents management이었으나 algorithm 자체의 약점을 이용한 Web page로 인해 최근에는 Google과 같이 Web의 link structure를 이용하는 Web structure management이 널리 사용된다. 그러나 Web structure management 역시 악의적으로 이용될 가능성은 존재한다. 1991년 최초로 발생하였으나, 최근 다시 발생한 Google Bombing[4]과 같은 사건이 여기에 해당된다.

본 논문에서는 Web structure management이 가지는 문제점을 인식하고 악의적으로 작성된 Web page를 자동으로 여과할 수 있도록 Web structure management 알고리즘을 수정하여 사용자가 요구하는 정확한 정보를 Web search engine이 제공할 수 있도록 하고자 한다.

2. Related Works

Web structure management은 Web page 사이의 hyperlink structure를 사용하여 Web page를 분류하거나 page의 중요도를 계산한다. 즉, Web page내에 저장된 content를 이용하는 것이 아니라 다른 Web page가 해당 Web page를 어떻게 설명하고 있는가에 중심을 두는 기법이다[5]. 대표적인 algorithm으로 PageRank와 HITS가 있다. PageRank algorithm은 각 Web page는 중요도를 가지고 있으며, link를 따라서 중요도가 동일한 크기로 source page로부터 target page로 전달되므로 많은 수의 in-link를 가지는 page가 그렇지 못한 page에 비해 높은 중요도를 나타내게 된다. 그러나 link structure만을 이용하기 때문에 Web page의 context를 고려하지 못하는 한계점이 존재한다. HITS algorithm은 Web page를 특정 query에 따라 authority page와 hub page score를 계산하여 subgraph를 형성하고 이를 query와 높은 연관성을 가지는 page를 찾는데 사용하는 방법이다[3]. 현재 Web search engine 중 세계적으로 가장 유명한 Google은 PageRank

algorithm을 사용하고 있다.

3. Preliminaries

3.1. Definition of problem

Spamming page는 특정 Web site 혹은 Web page에 인위적 조작을 가해 Web Search engine에 특정 질의어가 입력되었을 경우, 가장 높은 우선순위를 가질 수 있도록 한 page로 정의한다. 이런 spamming page를 작성하는 방법은 대상 Web search engine의 algorithm에 따라 다르게 나타나는데, 크게 page내의 contents를 조작한 방법과 link 정보를 조작하는 방법으로 구분할 수 있다. 전자의 경우, Web structure management algorithm을 이용하면 filtering 효과가 뛰어나다[3]. 그러나 후자는 Google bombing과 spamming page는 앞서 언급한 PageRank algorithm의 한계점을 이용한 것으로, 특정 web page의 rank를 높이기 위한 목적으로 해당 page의 in-link 수를 증가시킨 경우이다.

본 논문에서는 Web structure management 기법 중 가장 유명한 PageRank algorithm을 수정하여 spamming page가 상위 rank에 나타나지 못하도록 Web search engine의 성능을 개선하고자 한다.

3.2. Web structure

World Wide Web 전체를 하나의 directed labeled graph로 인식될 수 있다[2]. Web page는 URL을 ID로 가진 하나의 node로, page 사이의 link를 edge로 그리고 link context를 label로 사용한다. Link context는 source node로부터 target node로 정보를 전달하는 역할을 하며, Robust Hyperlink에서 정의한 lexical signature[7]와 link string[1]을 함께 사용한다. 이제까지의 Web structure management algorithm에서는 link context는 일반적으로 무시되어 왔다[2]. 그러나 최근에는 link structure를 이용하여 web page를 clustering하거나[2], web page의 중요도를 계산하는 방법론[7]들이 많이 제시되어 왔다. 본 논문에서는 link context를 이용하여 target page의 중요도를 상대적으로 조절하고, 이를 이용하여 각 page의 rank를 계산한다.

4. Filtering Algorithm

Spamming page를 filtering하는 algorithm은 다음 [그림 1]과 같다. Link context C_{ij}와 target page $tURL_j$ 사이의 similarity는 다음 식을 이용해서 계산한다.

$$sim(C_{ij}, tURL_j) = \max(CO(S_l, C_{ij}))$$

S_l은 target page 내의 l번째 문장을 나타내며, 우변은 S_l과 C_{ij}내 term들 사이의 co-occurrence 값 중 최대값을 나타내며, 그 값은 다음 식을 이용하여 계산한다.

$$CO(S_l, C_{ij}) = \frac{|S_l \cap C_{ij}|}{|C_{ij}|}$$

각 문장별로 link context와 co-occurrence 값을 similarity로 사용할 경우, page 전체를 대상으로 similarity를 계산하는 것에 비해 page내의 context를 더욱 잘 반영할 수 있게 된다[8]. 그리고 최대값을 이용하는 이유는 관련성이 높은 page가 page의 길이 혹은 term의 수에 의해 similarity가 낮아지는 경우인 1종 오류를 방지하기 위한 것이다.

$sURL_i$: i_{th} *source page URL* $i \in [1,n]$
$tURL_j$: j_{th} *target page URL* $j \in [1,n]$
$Rank(i)$: i_{th} *page's Rank*
$\overrightarrow{Rank}$: *Page rank vector*
L_{ij} $(sURL_i, tURL_j, C_{ij}, w_{ij})$
: *link from* i_{th} *source page to* j_{th} *target page*
$C_{ij} = \{t_k\}$: *set of terms in the link context*
w_{ij} : *weight value of* L_{ij}
M : *transition matrix of weight value*

```
for i {
        for j {
            if (exist L_ij)     w_ij ← sim(C_ij, tURL_j)
            else                w_ij ← 0
        }
        w'_ij ← normalize(w_ij)
}

M ← [ w'_ij ]_{n×n}
Rank⃗ ← M^T Rank⃗
```

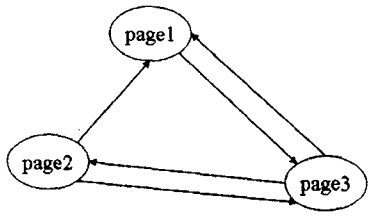
[그림 1] filtering algorithm

위 algorithm에서 w_{ij}를 normalize하는 이유는, transition matrix M의 행의 합이 1보다 작은 않은 경우가 발생할 수 있기 때문이다. 이 경우 Filtering algorithm이 반복해서 수행되면서 전체 page rank의 합이 초기값으로 유지되지 않고, 0으로 수렴하게 되고, 행의 합이 1을 초과할 경우 발산한다. 따라서 반드시 w_{ij}를 normalize시켜주어 행의 합이 1이 되도록 수정해야 한다.

1) link string

5. Example

다음 [그림 2]는 filtering algorithm을 설명하기 위한 간단한 예제이다. page 수는 3개, link는 5개로 이루어진 graph이고, 초기 page rank 값은 [100, 50, 70]으로 가정한다.



[그림 2] Example Graph

이 예제를 기존의 page rank algorithm을 이용하면 다음과 같은 결과를 얻을 수 있다.

$$\begin{pmatrix} 0 & 0 & 1 \\ 0.5 & 0 & 0.5 \\ 0.5 & 0.5 & 0 \end{pmatrix} \begin{pmatrix} 100 \\ 50 \\ 70 \end{pmatrix} = \begin{pmatrix} 60 \\ 35 \\ 125 \end{pmatrix}$$

다음 [표 1]은 각 link context와 target page와의 similarity를 계산한 값이다.

[표 1] Similarity

Link	L_{13}	L_{21}	L_{23}	L_{31}	L_{32}
w_{ij}	0.38	0.35	0.01	0.02	0.48

이 때, L_{23}와 L_{31}은 잘못된 정보를 담고 있는 link로, 낮은 similarity를 가지게 된다. 다음 [표 2]는 [표 1]을 source page별로 normalize 한 결과이다.

[표 2] Normalized similarity

Link	L_{13}	L_{21}	L_{23}	L_{31}	L_{32}
w_{ij}	1.0	0.972	0.028	0.04	0.96

위 표를 이용해서 transition matrix M을 작성하고, update된 page rank vector를 구한 결과는 다음과 같다.

$$\begin{pmatrix} 0 & 0 & 1 \\ 0.972 & 0 & 0.028 \\ 0.04 & 0.96 & 0 \end{pmatrix}^T \begin{pmatrix} 100 \\ 50 \\ 70 \end{pmatrix} = \begin{pmatrix} 51.4 \\ 67.2 \\ 101.4 \end{pmatrix}$$

L_{23}와 L_{31}은 잘못된 정보를 담고 있었으므로, 기존의 algorithm에 비해 page1과 page3의 rank가 상대적으로 감소되었고 page2의 rank가 증가되었음을 알 수 있다.

6. Conclusions and future works

우리는 앞의 예제에서 link context를 이용한 page rank algorithm이 기존의 algorithm의 성능을 개선해 주고 있음을 확인하였다. 본 논문에서 제시한 filtering algorithm을 사용할 경우, Google bombing과 같은 spamming page가 상위에 rank될 확률이 줄어들 뿐만 아니라, 기존의 link structure를 사용하는 algorithm에 page context를 고려할 수 있는 기능을 추가할 수 있게 된다.

앞으로 filtering algorithm을 개선하여 실제로 spamming page를 filtering할 수 있는 Web search engine을 구현하고, 실험을 통해 algorithm을 검증하고자 한다.

References

[1] Ricardo Baeza-Yates, Berthier Ribeiro-Neto, Modern Information Retrieval, ACM Press Books, 1999

[2] Maria Halkida, Benjamin Nguyen, Iraklis Varlamis, Michalis Vazirgiannis, THESUS: Organizing Web document collections based on link semantics, The VLDB Journal(2003) 12, 320-332, 2003

[3] Arvind Arasu, JungHoo cho, Hector Garcia-Molina, Adneras Paepcke, and Sriram Rachavan, Searching the Web, ACM Transactions on Internet Technology, Vol.1, No.1, pp.2-43, August 2001

[4] Ken "Caesar" Fisher, Google Bombing heating up, Ars Technica Newsdesk, January 22th 2004,

[5] Raymon Kosala, Hedrik Blockeel, Web mining Research :A Survey, ACM SIGKDD, Vol.2, Issue1, pp.1-15, July 2000

[6] Thomas A. Phelps, Robert Wilensky, Robust Hyperlinks Cost Just Five Words Each, UCB Computer Science Technical Report, 10 January, 2000.

[7] Taher H. Haveliwala, Topic-Sensitive PageRank:A Context-Sensitive Ranking Algorithm for Web Search, IEEE Transactions on Knowledge and Data Engineering, vol. 15, no. 4, 2003

[8] E.K. Park, S.I. Moon, D.Y. Ra, and M.G. Jang. Web Document Retrieval Using Sentence-query Similarity. In Proceedings of the 11th Text Retrieval Conference (TREC-11), notebook version, 2002