내용분석을 통한 향산된 링크기반 검색

이경희°,김민수,김민구
아주대학교 정보통신전문대학원
{lkyunghi@korea.com°,bijey@ajou.ac.kr, minkoo@ajou.ac.kr}

Improved Link-based Retrival with Content Analysis

kyunghee Lee°, Minsoo Kim, minkoo Kim
Ajou University

요 약

정보검색이 발달함에 따라 인터넷 환경에서의 정보 검색은 하이퍼링크 정보를 분석하여 이용하는 추세에 있다. 최근에는 주어진 주제어나 질의어에 대해 가장 적합한 검색 방법을 결정하기 위해 하이퍼텍스트 기반 링크 구조를 분석하는 알고리즘이 늘어나고 있는 실정이다. Bharat[2]은 HITS 알고리즘의 문제점을 지적하고, 이를 개선하기 위한 방법을 제안하였다. 본 논문에서는 Bharat이 제시한 확장 질의어를 만드는 방법에 대한 문제점 제기와 이 문제에 대한 개선안을 제시하고자 한다. .

1. 서 론

전통적인 정보검색의 주요 관점은 작고, static한 이종의 텍스트자료에서는 많은 비약적인 발전을 가져왔다. 그러나 인터넷이 발전하므로써, 웹의 환경에서 전통적인 검색방법으로는 좋은 결과의 검색을 할 수가 없다. 웹은 다이나믹하고, 거대한 정보를 가지고 있으며 하이퍼링크 환경에서 작동하기 때문에 전통적인 검색방법으로는 좋은 결과를 기대하지 못한다. 웹의 정보는 빠르게 변화하고 있다. Web 정보검색은 사용자가 질의어를 입력하였을때 관계있는 웹페이지를 보여주는 것이다. Web 정보검색에 대한 수많은 논문이 발표된 상태이다. 이들 논문 대부분은 링크기반 검색을 이용한 것이다.

웹 검색의 발전과정을 보면 1세대는 질의어 기반 검색이고, 2세대는 링크기반검색, 3세대는 링크기반검색에 기반을 두고 여러 가지 알고리즘을 혼영 한 것이다. 이중 2세대인 링크기반 검색에서는 HITS 알고리즘을 많이 이용하였다. HITS 알고리즘은 하이퍼링크 정보를 사용한 알고리즘이다. HITS 알고리즘은 웹 검색에 많은 공헌을 하였고, 웹 검색이 획기적인 발전을 할 수 있는 계기가 된 것이다. 이 HITS 방법은 웹 검색에서 좋은 결과를 가져왔지만 많은 문제점이 야기되었다. Bharat은 HITS 알고리즘의 문제점을 제기하고, 개선안을 논문에서 제시하였다. 본고에서는 Bharat의 문제점과 개선사항을 논할 것이다. 본 논문의 구성은 다음과 같다. 2장에서는 HITS 알고리즘과 Bharat's 검색 방법의 문제점, 3장에서는 그 문제의 해결방안에 대해 설명하고, 4장에서는 실험결과에 대해 논할 것이고 마지막 5장에서는 결론으로 마무리 짓고자 한다

본 논문은 과학기술부의 국가지정연구실 사업(과제명: 차세대 인터넷을 위한 지능형 온톨로지 자동생성 시스템 개발, 과제번호 : M10302000087-03J0000-04400) 지원으로 수행되었음

2. HITS 와 Bharat's 알고리즘의 문제점

HITS 가제시한 알고리즘은 아래 설명과 같다.

2.1 HITS 알고리즘

Kleinberg가 제안한 HITS 알고리즘은 웹문서의 링크 정보를 분석하여, authority와 hub로 나누고, authority는 질의어에 대해 많은 정보를 가지고 있는 페이지이고, hub는 많은 정보를 가리키는 페이지로 웹문서를 구분하는 알고리즘을 사용하여 웹 환경에서 좀 더 좋은 결과를 가져왔다. HITS 알고리즘을 간단히 설명하면, HITS 알고리즘은 웹페이지의 inlink 와 outlink 를 이용하여 웹페이지의 점수를 계산한 것이다. inlink의 합을 "authority"라 하였고, outlink의 합을 hub라 하여 페이지마다 질의어와의 관계성을 hub 값과 authority값을 이용하여 평가한 것이다 HITS 알고리즘은 검색엔진에서의 상위 200개의 문서를 root set이라하고, 이 root set의 inlink와 outlink를 이용하여 확장한 집합을 base set이라 한다[1].

2.2 Bharat 이 지적한 HITS 알고리즘의 문제점

Bahart이 HITS 알고리즘에서 지적한 3가지 문제점은 다음과 같다.

- 호스트간의 서로 상호적으로 관계가 강화된 상태 : 아래 그림[2-1]에서 보는바와 같이 호스트 A에 있는 문서 집합들이 호스트 B에 있는 한 문서를 link할 때 호스트 A 의 hub 값과 호스트 B 의 Autority값이 증가하게 된다.

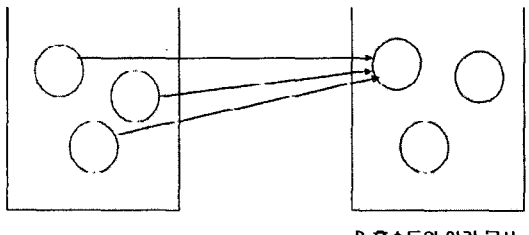

그림 1. 여러 문서에서 한 문서를 링크하는 경우

위의 그림 1처럼 한 A라는 호스트에 있는 여러 문서가 B라는 호스트의 한 문서를 링크하는 경우 올바른 authority와 hub값을 가질 수가 없다. 이에 Bharat은 위 그림과 같은 문제를 해결하기위해 autority값을 구할 때, 한 호스트에서 링크하는 문서들은 그 문서 개수만큼 나눠주었다. 공식은 아래와 같은 공식을 사용하였다 [2].

$$A[n]:=\sum_{(n',n)\in N} A[n']\times auth-wt(n,n')$$

hub값을 구하는 공식은 다음과 같다.

$$H[n]:=\sum_{(n,n')\in N} A[n']\times hub-wt(n,n')$$

- 자동적으로 생성된 링크인 경우 : 일반적으로 authoring tools에 의해 자동적으로 링크가 생성되는 문제이다. 이렇게 자동적으로 생긴 링크들은 authority를 부여하는데 아무런 관계가 없다.
- 관계없는 노드들 : 사용자가 입력한 질의어와 관계없는 문서들을 결과로 나타낸다.

Bharat 알고리즘은 HITS 의 3가지 문제를 해결하였다. HITS 알고리즘의 첫 번째 문제를 극복하기 위해, Bharat은 한 호스트에서의 문서에서 다른 문서로 링크할 때, 그 호스트에 문서 수만큼 aurority 의 값을 나누는 것이다. 또한 HITS 의 두 번째 문제를 해결하기위해 노드에 relevance weight 을 계산하여 일정한 threshold 값 아래에 있는 페이지들을 제거하였다. HITS 의 세 번째 문제인 질의어와 관계없는 문서에 대한 해결책으로 질의어를 확장하여 한번 검색을 하였다..

2.3 Bharat's 알고리즘

첫 번째 단계는 알타비스타 검색엔진을 사용하여 질의어를 주어 검색결과를 얻는다. 이 문서 결과 중에서 상위 200개의 문서를 root set 이라 한다. 두 번째 단계에서는 root set 에 있는 각문서의 위에서부터 1000개의 term을 broad query 로 만든다. 세 번째 단계에서는 broad query 와 base set 과의 유사도를 계산하여 threshold 값 아래에 있는 문서는 제거한다. Bharat's 의 알고리즘은 다음과 같다[2].

1. 알타비스타 검색엔진을 이용하여 쿼리를 주고 root set을 얻는다.
2. root set 으로부터 broad query를 만든다.
3. root set 으로부터 inlink와 oulink을 이용하여 base set을 만든다.
4. broad query와 base set 간의 유사도를 계산한다.
5. threshold 아래에 있는 관계없는 문서는 제거한다.
6. 각 문서의 authority 와 hub 값을 계산한다.
7. authority와 hub를 Normarlize 시킨다.

2.4 Bharat's알고리즘의 문제점

Bharat은 broad query를 결정하는데 root set의 각 문서에 1000개의 term을 선택하여 broad query를 만들었다. original query를 확장하여 broad query를 만드는데 broad query 에서의 term들은 orginal query 와 별로 relevant 하지 않다. 그 이유는 original query 의 결과로 나온 문서들의 상위 1000개의 term들이 original query 와 유사도가 없다는 것이다. original query를 확장하여 broad query를 만드는 이유는 사용자에게 좀더 original query에 가까운 결과를 돌려주기 위해서이다. broad query는 original query와의 relevant가 높아야 좋은 결과를 사용자에게 돌려준다. 그러나 Bharat이 만든 Broad query는 original query와 별관계가 없는 term들로 이루어져 있다.

3. Bharat의 문제를 해결한 제안 알고리즘

broad query를 만들고자 할때, 좀더 original query와 관계가 깊은 query를 만드는 방법은, 첫 번째는 Bharat과 비슷하게 알타비스타 검색엔진을 이용하여 결과를 돌려받고, 이 결과 중에서 상위 200개의 문서를 root set 이라고 가정한다. 이 root set의 문서들을 k-menas 기법을 사용하여 5개의 Clustering 분류하여 각 문서들의 유사도를 구하여 가장 비슷한 cluster 집합을 구한다. 200개의 root set이 5개의 cluster 로 분류되고, 그 중 유사도가 가장 높은 cluster 집합에 있는 모든 문서의 term들을 broad query 로 하여 검색을 하였다. 제안된 알고리즘은 다음과 같다.

1. 알타비스타 검색엔진을 이용하여 original query를 주어 검색 결과를 얻는다.
2. 그 검색 결과 중에서 상위 200개의 문서를 root set 이라 한다.
3. 그 root set 집합 중에서 임의의 문서를 선택하여 Okapi의 inquery 를 만드는 공식을 이용하여 seed 를 만든다.
4. 5개의 임의문서를 가지고 seed를 만들어 5개의 cluster 를 만든다
5. root set 의 모든 문서를 Okapi 공식에 적용시켜 5개의 cluster 에 해당하는 하나의 cluster 로 분류 시킨다.
6. 그 5개의 cluster 들의 유사도를 계산 한다.
7. 그중 유사도가 가장 높은 cluster 를 선택하여 그 cluster 에 있는 모든 term 들을 broad query 로 한다.

위의 알고리즘에서 root set 의 임의의 문서를 선택하여 Seed를 만들 때, 다음과 같은 Okapi 공식과 INQUERY's normalized idf score를 혼합하여 이용한다[3].

$$w_{i,j}=\frac{tf_{i,j}}{tf_{i,j}+0.5+1.5\frac{doclen_j}{avgdoclen}}\cdot\frac{\log(\frac{colsize+0.5}{docf_i})}{\log(colsize+1)}$$

위의 공식에서 $v_{i,j}$는 j번째 문서 안에서 i번째 term의 weight를 나타낸다. 이문서의 모든 term에 대한 v값을 더하여 그 문서의 5개의 cluster중에서 가장 값이 비슷한 cluster로 분류 한다. 위에서 설명하였듯이, 만들어진 broad query 와 base set 간에 유사도를 계산하여 threshold값 보다 낮은 문서는 제거한다. threshold를 만드는 방법은 다음과 같이 3가지 방법을 이용하였다.

1) Median weight(Imp) : 모든 relevance weight 의 median 값을 구하여 threshold로 지정하였다.

2) Start set Median Weight(Med) : root set에서의 relevance weight 의 median값을 threshold로 한다.

3) Fraction of Maximum Weight : 최대 값을 10으로 나눈 값을 threshold 로 한다.

위와 같은 세 가지 방법으로 threshold값을 구분한 후, threshold 값 이하 값을 가진 문서들은 제거한다.

4. 실험 환경및 결과

4.1 실험환경

알타비스타 검색엔진을 이용하였고, HillTop query 중에서, "keebler","pizza",등 2개의 query를 가지고 실험 하였다.

4.2 실험 결과

표1의 결과는 "keebler" 질의어 결과 authority 에 대한 정확도이다.

query	문서 갯수	Bharat 알고리즘			제안한 알고리즘		
		Imp	Med	Max by 10	Imp	Med	Max by 10
keebler	10	0.5	0.6	0.4	0.44	0.5	0.6
	20	0.35	0.45	0.5	0.41	0.47	0.59

표1.keebler 질의어에서의 authority문서 정확도

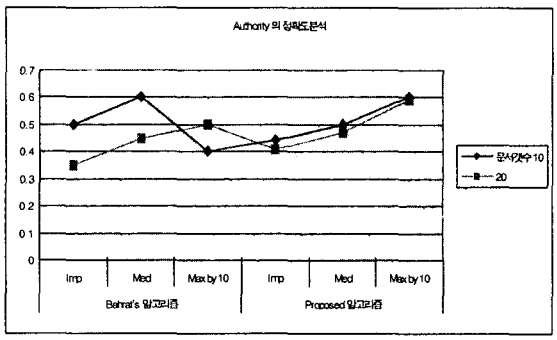

그림 2. keebler 질의어에서의 authority문서 정확도

query	문서 갯수	Bharat 알고리즘			제안한 알고리즘		
		Imp	Med	Max by 10	Imp	Med	M a x by 10
keebler	10	0.5	0.55	0.44	0.67	0.6	0.67
	20	0.42	0.37	0.39	0.44	0.42	0.44

표2. keebler 질의어에서의 hub문서 정확도

위의 표2 는 "keebler" 질의어를 주어서 hub 문서에 대한 정확도를 계산한것이다.

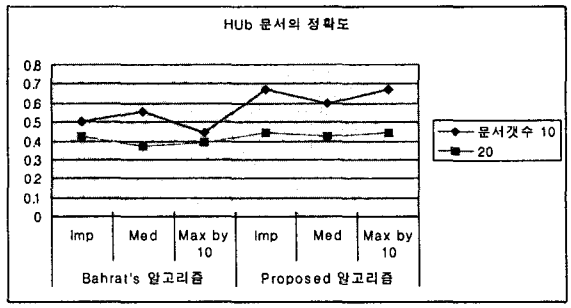

그림3. keebler 질의어에서의 hub문서 정확도

authority 정확도에서 문서의 갯수가 10개인 경우에는 Bharat 알고리즘보다 2%의 향상을 보였고, 20개의문서에서는 13%의 향상을 보였다. hub의 정확도에서는 10개의 문서인 경우 30%의 향상을 보였고, 20개의 문서인 경우 10%가 향상되었다. 또한 "pizzahut" 이라는 질의어를 주었을 경우도 비슷한 결과를 보였다.

5. 결론

본 논문에서는 Bharat이 만든 알고리즘 중에서 질의어를 확장하여 Broad query를 만드는데 있어서 original query 와 관계없는 broad query 에 문제점을 제기하였고, 이에 대한 해결 방안으로 root set 의 모든 문서들을 5개의 cluster 로 나누고 이 cluster 중에 가장 유사도가 높은 cluster 에 있는 term 을 broad query를 만드는 것을 제안하였다. 위와 같은 방법으로 하였더니 논문에서 제안한 질의어 확장방식에서 Bharat 알고리즘보다 authority 의 정확성이 7.5% 향상되었고 hub 정확도는 평균 20%가 향상되었다. 실험 데이터는 Hilltop 질의어 중에서 "keebler", "pizzahut" 두 개의 질의어를 사용하여 실험한 결과 전체 *authority* 와 *hub* 문서에 대한 정확도가 많이 향상되었다. authority 는 향상율이 별로 좋지않은 반면, hub 문서에 대한 정확도는 많이 향상되었다.

향후 과제에서는 authority 문서에 대한 정확도를 향상시키는 연구가 계속 되어야 할 것이다. 또한 실험 결과를 전문가가 정확도를 검증할 때, 웹의 특성인 다이나믹한 정보로 인하여 많은 문서들이 열리지 않았다. 이 문제에 대해서도 더 많은 연구가 진행되어야 할 것이다.

참고 문헌

[1] Jon M.Kleinberg " Authritative Sources in a Hyperlinked environment" (1998)

[2] Krishna Bharat " Improved Algorithms for topic Distillation in a Hyperlinked Environment" (1998)

[3] Anton Leuski " Evaluating Document Clustering for Interactive Information Retrieval"

[4] Junghoo Cho " The PageRank Citation Ranking : Brining order to Web" (1998)