http://dx.doi.org/10.5392/JKCA.2013.13.06.030

웹크롤러의 수집주기 최적화

Refresh Cycle Optimization for Web Crawlers

조완섭*, 이정은**, 최치환***
충북대학교 경영정보학과/대학원비즈니스데이터융합학과*,
충북대학교 비즈니스데이터융합학과**, 충북대학교 바이오정보기술학과***

Wan-Sup Cho(wscho@chungbuk.ac.kr)*, Jeong-Eun Lee(masjjong@naver.com)**,
Chi-Hwan Choi(chihwan.choi86@gmail.com)***

요약

웹 크롤러는 서버의 부담을 최소화하면서도 최신의 데이터를 웹사이트로부터 수집하고 유지해야 한다. 빅데이터 시대와 같이 데이터가 폭발적으로 증가하는 시대에 데이터 소스로부터 자주 모든 데이터를 추출하는 것은 서버에 심각한 부담을 주게 된다. 무선통신 기술과 다양한 스마트 기기들의 확산으로 정보가 급속도로 생성되고 있으며, 어디에서나 어느 시간이나 지속적으로 생성 및 변경되고 있다. 웹크롤러는 이러한 상황을 감안하여 최신의 정보를 적은 오버헤드로 유지해 나가는 것이 중요한 이슈로 부각되고 있다. 본 논문에서는 웹사이트의 변경사항을 체크할 수 있는 효과적인 방안과 웹사이트의 수집 주기를 동적으로 변경함으로써 적은 비용으로 최신성을 유지할 수 있는 방안을 제시한다. 핵심 아이디어는 과거 히스토리로부터 웹사이트 변경이 집중되는 시간을 파악하여 웹수집 주기를 결정하는데 반영한다는 점이다. 논문에서는 특정 웹사이트의 데이터를 추출하는 Java 크롤러를 개발하고, 제안된 방식과 기존 방식의 유용성을 비교하였다. 제안된 기법을 사용하면 정적인 방식보다 서버 오버헤드를 절반정도(46.2%)로 줄이면서도 최신성을 더욱 높게 보장할 수 있게 된다.

■ 중심어 : | 웹수집 주기 | 크롤러 | 웹사이트 | 동적수집 |

Abstract

Web crawler should maintain fresh data with minimum server overhead for large amount of data in the web sites. The overhead in the server increases rapidly as the amount of data is exploding as in the big data era. The amount of web information is increasing rapidly with advanced wireless networks and emergence of diverse smart devices. Furthermore, the information is continuously being produced and updated in anywhere and anytime by means of easy web platforms, and smart devices. Now, it is becoming a hot issue how frequently updated web data has to be refreshed in data collection and integration. In this paper, we propose dynamic web-data crawling methods, which include sensitive checking of web site changes, and dynamic retrieving of web pages from target web sites based on historical update patterns. Furthermore, we implemented a Java-based web crawling application and compared efficiency between conventional static approaches and our dynamic one. Our experiment results showed 46.2% overhead benefits with more fresh data compared to the static crawling methods.

■ keyword : | Web Crawling Cycle | Crawler | Web Sites | Dynamic Collection |

* 이 논문은 2011년도 충북대학교 학술연구지원사업의 연구비지원에 의하여 연구되었음.
접수번호 : #130523-001
접수일자 : 2013년 05월 23일
심사완료일 : 2013년 05월 31일
교신저자 : 조완섭, e-mail : wscho@chungbuk.ac.kr

1. 서 론

인터넷 상의 웹사이트 등에 산재된 데이터로부터 사용자가 관심이 있는 부분을 추출하여 관련있는 데이터를 통합하고, 가공함으로써 데이터의 가치를 높이는 활동이 다양한 영역에서 이루어지고 있다. 예를 들어, 트위터, 블로그, 온라인 쇼핑몰 등으로부터 데이터를 추출하여 고객의 성향을 분석함으로써 마케팅이나 제품 설계에 반영할 수 있으며, 웹서버의 웹로그 데이터로부터 웹사이트 최적 구성이나 성능 및 안전에 관한 정보를 분석해낼 수 있다. 또한 M2M 등 다양한 센서 장비로부터 스트림 데이터를 추출하여 분석함으로써 재난 혹은 안전관련 유용한 정보를 분석할 수 있다.

웹은 방대한 양의 다양한 자료들로 구성된 개방형 집합체이며 그 접근이 매우 용이하지만 웹에서 제공되는 자원은 전통적으로 도서관이나 온라인 데이터베이스에서 이용 가능했던 자원과 크게 다르다. 매우 이질적인 자료들이 비계층적으로 복합하게 형성되어 있으며, 자료의 생성과 소멸이 빈번히 일어나고 역동적이고, 물리적 경계를 갖지 않는 매우 유동적인 정보환경이 그 특징이다.

다양한 특성을 가진 웹사이트로부터 유용한 정보를 추출하여 가치를 창출하기 위한 첫단계는 데이터 추출과 정제 및 데이터베이스로의 통합단계(ETL: Extraction, Transformation, Loading)이다. 다양한 ETL 도구가 제품으로 시중에 나와 있으며[13][17], Open Source 도구도 제공되고 있다. 이들은 다양한 소스(데이터베이스나 ERP 시스템 등)로부터 데이터를 추출하는데 유용하다.

ETL 중에서 특정 웹사이트에 저장된 정보를 추출하는 도구로서 웹 크롤러(Web Crawler)가 사용되고 있다[1][2][7-9]. 웹 크롤러는 모든 웹문서를 수집 대상으로 하는 범용 크롤러[10], Focused Crawler[12], 특정한 주제의 문서만을 수집하는 Topical crawler[14], 래퍼기반 웹크롤러[16] 등으로 구분할 수 있다. 래퍼기반 웹크롤러는 해당 사이트의 구조를 미리 분석한 정보 (url-list)를 이용하여 웹크롤러가 해당 페이지만 접근하므로 쇼핑몰의 VOC (voice of customer) 정보 등을 수집하는데 적합하다.

크롤러의 경우 웹사이트 정보가 방대한 경우 웹서버에 부담을 주거나 크롤링 시간이 오래 걸리는 문제가 발생한다[2][3]. 실제로 오픈소스 크롤러 도구를 사용하여 특정 블로그 사이트로부터 관심있는 데이터(웹페이지)를 추출하기 위해서는 수일 이상의 장시간이 소요될 수 있으며, 대량의 불필요한 데이터가 함께 추출되기도 한다. 참고문헌[2]에 따르면 400만개의 웹문서 데이터를 수집하는 데는 약 20일 정도가 소요되었다. 이러한 경우 웹사이트로부터 정보를 수집하는 주기를 결정하는 것이 중요한 문제이다. 주기가 너무 짧으면 수집 비용과 시간이 많이 소요되고, 너무 길면 최신성을 보장하는데 한계가 있기 때문이다.

본 연구에서는 래퍼기반 웹크롤러에서 갱신주기(refresh cycle)를 동적으로 결정하는 방안을 제안한다. 일반적으로 웹사이트 내의 웹페이지들도 그 특성에 따라서 갱신주기나 빈도가 다양한 특성을 가진다. 따라서 웹 크롤러에 하나의 갱신주기를 적용하면 수집 비용에 비하여 최신성을 확보하는데 한계가 있다. 자주 변하는 페이지의 경우 갱신주기를 짧게 배정해야 하며, 나아가 시간대에 따라서 다른 갱신주기를 지정할 필요도 있다. 본 논문에서는 웹페이지의 URL들을 몇가지 유형으로 구분한 후, 각 유형에 특성을 반영하여 갱신주기가 이루어지도록 한다.

본 논문의 구성은 다음과 같다. 제 2 장에서는 기존의 웹크롤링 기법들에 관하여 설명한다. 제 3장에서는 동적 웹크롤링 방법을 제안한다. 제 4장에서는 제안된 방법을 사용한 크롤러의 제작에 관하여 설명한다. 마지막으로 제 5 장에서 결론을 맺는다.

2. 관련연구

검색엔진의 근간이 되는 웹크롤러(web crawler)는 인터넷상에 존재하는 웹문서들을 추적하여 필요한 정보를 수집하는 기술을 말하며, 야후와 같은 인터넷 검색시스템, 전자상거래 상품검색등 대부분의 인터넷 산업의 근간이 되는 핵심기술이다[5].

웹크롤러는 웹서버를 순환하면서 각 홈페이지에 있는 수많은 정보를 수집하는 프로그램으로 사람이 홈페이지의 각 링크를 따라가서 정보를 얻는 반복적인 작업

을 대신하여 프로그램이 자동으로 웹페이지의 내용을 분석한 후 URL을 하나씩 접속하여 정보를 수집한다. [그림 1]은 웹크롤링시 큐를 이용한 스케줄 및 멀티쓰레드 방식으로 고성능 웹크롤링 방법에 대해 보여준다. 그림에서 Frontier는 방문해야 하는 URL (혹은 URL 패턴)들의 리스트이며, 일반적으로 FIFO(first input first output) 큐로 구현된다. 하나의 URL에 대한 페이지를 가져오면 그 안에 또 다른 웹링크가 포함될 수 있으며, 웹크롤러는 이들 페이지도 함께 가져와야 한다. 필요한 경우 웹링크를 따라가는 깊이를 제한할 수도 있다.

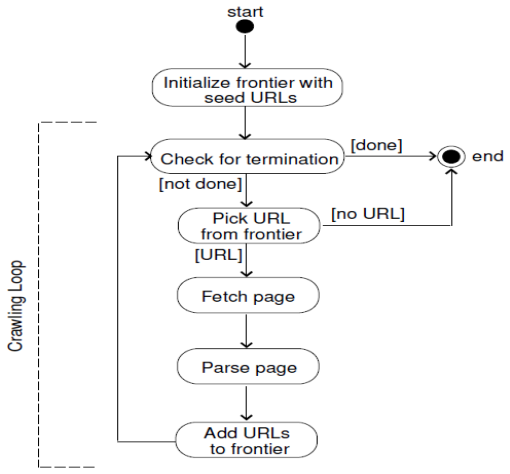


그림 1. 웹크롤링의 기본 구성[19]

웹 크롤링에서 중요한 사항은 수집될 웹페이지의 탐색범위를 정하는 문제와 수집주기 (refresh cycle)을 정하는 것이다. 탐색범위를 전체 웹사이트로 하여 광범위한 수집을 하는 경우도 있고(blind 혹은 exhaustive crawler)[19], 특정 주제를 중심으로 검색을 하거나 (topical 혹은 focused crawler)[19], URL의 패턴들을 입력받아 특정한 URL들만 검색하는 경우로 구분할 수 있다 (structure-based crawler) [1][4][7][8][19].

일반적인 크롤러는 모든 웹 문서를 수집 대상으로 하며, 포커스드 크롤러(focused crawler)와 토피컬 크롤러(topical crawler)는 특정 주제의 문서만을 수집하기 위해 일반적인 크롤러에 문서 분류기(document classifier)나 규칙(rules)을 포함하고 있다[10]. 이 때, 포커스드 크롤러나 토피컬 크롤러에서 분류기의 성능이 좋지 못하고 규칙의 양이 불충분할 경우 사용자가 원하는 데이터를 수집할 수 없다. 이러한 이유로 수집 대상 사이트의 구조를 분석하여 직접적으로 데이터를 수집하는 래퍼 관련 연구가 진행 되어왔다[1].

웹페이지 수집주기는 정적인 방식과 동적인 방식으로 구분할 수 있다. 일반적으로 정해진 시간에 웹페이지를 검색하여 원하는 자료를 가져오는 정적 수집주기 (static refresh cycle)를 많이 사용하지만 변경되지 않는 자료일지라도 정해진 시간에 수집해야 하므로 네트워크 오버헤드가 증가하여 효과적이지 못하다. 이에 비하여 동적 수집주기(dynamic refresh cycle) 방식을 사용하면 자주 변경되는 페이지는 자주 가져오고 그렇지 않은 페이지는 가끔씩 가져오게 되므로 서버나 클라이언트의 부담이 경감되는 이점이 있다[5][6].

동적 수집주기 방식은 연구결과가 많지 않다. 참고문헌 [6]에서는 어떤 페이지가 자주 변경되는지를 파악하여 이를 바탕으로 페이지별로 수집주기를 다르게 정하는 방식을 사용한다. 여기서는 변경이 자주 있는 페이지의 경우 수집주기를 일정한 값(예를 들어 0.5)을 곱하여 수집주기를 단축하고, 변경이 적은 페이지는 일정한 값을 곱하여 (예를 들어, 1.2) 수집주기를 늘려가는 방식을 사용한다.

참고문헌 [10]에서는 웹페이지의 freshness와 age 개념을 도입하여 소스 데이터에 비하여 웹크롤러가 가져온 로컬 데이터가 얼마나 오래되었는가를 측정하는 방안을 제시하였다. 이 문제는 복제 데이터의 유지기법과 유사하며, 얼마나 서버의 부하를 줄이면서 데이터 신선도를 높이는 가에 대한 tradeoff 로 볼 수 있다. 또한 이 연구에서는 웹페이지의 변경주기를 포아송(Poisson) 분포를 가정하고 예측하는 방법을 제안하고 있다. 반면 [20]에서 웹 문서의 변화를 표현하는 척도로 '다운로드 성공률', '변경률', '나이 변이 계수'를 제시하였으며, 이를 통해 다운로드 성공과 변경이 과거 기록과 밀접한 연관이 있음을 발견하였다. 또한 과거 기록을 통한 웹 문서의 다운로드 성공과 변경을 예측할 수 있는 모델을 제안하였다.

위의 [20]접근방식에서 웹수집을 고정적으로 이틀에 한번씩 총50회를 실시하였고, 수집결과 약 절반가량의 URL이 불규칙적으로 변하며 나머지 절반가량은 변하

지 않는 것으로 확인되었다. 하지만 분석된 수집주기결과를 적용한 웹문서 수집은 수행하지 않았으며, 이틀에 한번을 주기로 웹문서를 수집하기 때문에 수집주기보다 빈번하게 변경되는 문서의 변화는 고려되지 못하였다.

또한 VOC 수집 크롤러는 [1]과 같이 웹링크를 5단계를 거쳐야하므로 일반크롤러, 포커스드크롤러, 토피컬 크롤러 등으로 수집이 어려우며 래퍼기반의 크롤러가 가장 성능이 좋은 것으로 확인되었다. 본 논문의 접근방식도 래퍼크롤러의 요구사항을 반영하고 있으며, 사이트 구조분석 후 크롤러를 설계 및 구현하는 방식이다.

3. 웹사이트 수집주기 최적화 기법

본 장에서는 웹크롤링 수집주기의 동적 스케줄 방법을 제안하고 실제 웹사이트에 대하여 적용한 실험 결과를 설명한다.

수집주기를 동적으로 적용하기 위하여 웹 크롤러는 일정기간 동안에 웹페이지의 변경여부를 시험적으로 확인하고 이 정보를 축정한 후 분석하여 초기 수집주기를 설정한다. 초기 수집주기는 시간이 지남에 따라 실제 상황과 차이가 있을 수 있으므로 일정 기간이 지나면 다시 그동안 축적된 정보를 바탕으로 갱신해 나가는 방식으로 수집주기를 지속적으로 적응시켜 나간다. 이 장에서는 동적으로 수집주기를 결정하기 위한 데이터 수집 및 관리 방안을 제안하고, 이를 분석하여 수집주기 결정에 사용하는 기법을 소개한다.

3.1. 웹크롤링의 동적 스케줄의 방법 설계

제안된 방식에서는 검색 대상이 되는 웹사이트의 URL 별로 일정 기간 동안 변경관련 정보를 그림 2와 같은 데이터베이스 테이블에 축적해 나간다. 이 테이블에 유지되는 정보는 날짜와 시간별(현재 20분 간격으로 1개월간 URL의 변경여부를 조사함)로 각 URL에 해당하는 페이지의 변경여부와 변경된 사이즈 (변경크기로써 추정치임)를 보관한다. 특정 URL에 해당하는 페이지의 변경여부는 기본적으로 페이지 콘텐츠에 대한 해쉬값으로 판정할 수 있으며, 변경크기는 지난번 동일 페이지에 대한 페이지 크기를 이번에 가져온 크기와 비교하여 결정한다. 변경크기는 추정치로써 페이지의 일부가 동일한 크기의 새 콘텐츠로 대치(replace)된 경우에는 그 차이를 정확하게 알 수는 없다는 단점이 있다.

웹 크롤러는 일정 기간동인 URL의 변경정보를 [그림 2]의 테이블과 같이 저장한 후, 이를 분석함으로써 최적의 수집주기를 결정하게 된다. 테이블에서는 변경된 페이지의 일부만 보여주고 있으며 (hash = 1), 각 필드의 의미는 다음과 같다.

표 3.1 필드항목 정의

필드	의미
data_time	20 분 간격으로 구성된 변경여부 검침시간
url	검색대상이 되는 url 주소
get_size	해당 페이지에서 가져온 데이터 크기
update_size	지난번에 가져온 데이터와 비교하여 변경된 크기
hash	변경여부를 나타내는 hash 값 (1: 변경됨, 0: 변경안됨)

3.2 URL별 갱신정보의 분석

표 3.2 URL별 갱신정보 테이블

id	date_time	url	get_size	update_size	hash
1	2013-03-01 오전 9:20:00	u1	3500	10	1
2	2013-03-01 오전 9:40:00	u5	4600	100	1
3	2013-03-01 오전 9:40:00	u7	15000	200	1
4	2013-03-01 오전 9:40:00	u10	13000	20	1
5	2013-03-01 오전 9:40:00	u15	5000	35	1
6	2013-03-02 오전 9:40:00	u1	380	100	1
7	2013-03-02 오전 11:40:00	u2	4500	20	1
8	2013-03-02 오전 10:00:00	u8	8900	20	1
9	2013-03-02 오전 10:20:00	u9	6800	200	1
10	2013-03-02 오전 10:40:00	u10	17800	10	1
11	2013-03-02 오전 11:00:00	u11	3900	30	1
12	2013-03-02 오전 11:20:00	u13	500	40	1
13	2013-03-03 오후 1:00:00	u1	3800	100	1
14	2013-03-03 오후 2:20:00	u2	580	20	1
15	2013-03-03 오후 2:40:00	u3	400	20	1
16	2013-03-04 오후 3:20:00	u1	5000	20	1

[표 3.2]의 테이블로부터 [표 3.3]와 같이 주어진 학습기간 (여기서는 3주로 설정하였음) 동안에 “URL별 변경횟수”를 분석할 수 있다. 이는 [표 3.2]의 테이블에 대하여 SQL 질의를 사용하여 URL별로 변경횟수의 합과 변경크기를 계산한 것이다. 이 분석결과를 사용하여 변

경횟수합의 크기에 비례하여 해당 URL들의 수집주기를 짧게 한다. 즉, URL u1의 경우 3주 동안에 14회의 변경이 이루어졌으며, 주당 평균회수는 4.7회로 가장 변경이 많은 페이지로 나타났다. 따라서 수집주기를 가장 짧게 설정해야 함을 알 수 있다. 경우에 따라서 변경크기까지 고려하여 수집주기를 결정할 수도 있으나 여기서는 복잡도를 감안하여 (내용을 대체하는 경우 정확도에도 문제가 있음) 변경크기는 고려하지 않는다.

표 3.3 학습기간 동안 URL별 변경회수 합과 변경크기

URL	변경횟수합	주당평균 변경횟수	변경크기
u1	14	4.7	2450
u5	10	3.3	1570
u3	10	3.3	1850
u2	8	2.7	2270
u4	7	2.3	1580
u11	7	2.3	825
u10	6	2	450
u9	3	1	590
u8	3	1	121
u7	3	1	570
u6	3	1	950
u12	3	1	950
u15	2	0.7	135
u13	1	0.3	40

다음에는 변경횟수합을 기준으로 다음과 같이 변경주기를 결정한다. 구현의 편의성을 위해 주당평균변경횟수에 비례하여 변경주기를 16단계로 나누어 설정한다 (응용 분야별로 적극적으로 변경된 내용을 수집하는 경우와 그렇지 않은 경우로 구분하여 단계의 개수는 조정할 수 있음).

- G14: 주당 14회 수집그룹(매일 2회 수집)
- G13 : 주당 13회 수집그룹(매일 1회 수집)
- G12 : 주당 12회 수집그룹
- G2 : 주당 2회 수집그룹 [1.5~2.5)
- G1 : 주당 1회 수집그룹 [0.5~1.5)
- 2G1: 2주당 1회 수집 [0.25~0.5)
- 4G1: 4주당 1회 수집 [0~0.25)

G14~G1 그룹은 어떤 페이지의 주당평균변경회수와 가장 가까운 정수값(nearest integer function)이 i이면 Gi 그룹으로 배정하여 얻어진다. 예를 들어 url u1의 경우 주당 평균변경회수가 4.7 이므로 G5에 속한다. 나머지 2G1, 4G1 그룹은 가장 가까운 정수값을 취하면 모두 0이 되므로 (0.25~0.5) 사이의 값은 2G1 그룹으로, (0~0.25) 사이의 값은 4G1 그룹으로 각각 배정한다. 시스템 구현시에는 복잡도를 감안하여 16개의 그룹을 줄여서 단순화할 수 있다. 예를 들어 다음과 같이 6개의 그룹으로 묶을 수 있다.

- 4G1 : 4주마다 1회 수집
- 2G1 : 2주마다 1회 수집
- G1 : 주 1회 수집주기
- G3 (G2, G4 포함) : 주 3회 수집주기
- G7 (G5~G10) : 주 7회 수집주기 (매일 1회)
- G14 (G11~) : 주 14회 수집주기 (매일 2회)

이 기준에 따르면 url u1의 경우 G5에 속하지만 단순화한 후에는 G7에 속하게 되어 매일 한 번씩 수집하게 된다.

각 URL 별로 수집주기를 그룹화한 후에 남은 문제는 '어느 시기 (요일, 시간대)에 수집하는 것이 좋은가?' 라는 문제이다. 예를 들어, [표 3.3]에서 u1의 경우 G7에 속하게 되어 매일 수집하는 형태이지만 하루 중에서 어느 시간에 수집할 지를 결정하는 문제가 남아 있다. 그러나, u4의 경우 주당 변경회수가 2.3이므로 G3에 속하고, 이 경우 어느 요일에 수집할 것인가와 어느 시간대에 수집할 것인가를 모두 결정해야 한다. 이를 위하여 본 논문에서는 [표 3.4]과 같은 페이지 변경정보를 활용한다.

[표 3.4]는 url 별, 요일별, 시간대별로 변경횟수합을 보여주고 있으므로 이를 분석하면 어느 요일, 어느 시간대에 수집하는 것이 가장 유리한지를 알 수 있다. [표 3.4]에서 u1의 경우 모든 변경이 09시~15시 사이에 이루어졌으므로 15시 이후에 수집을 실행하는 것이 유리하다. 구현의 편의성과 단순화를 위해 야간 시간대(18:00~익일 09:00)에 변경이 집중되는 페이지의 경우 낮 12시, 주간 시간대(09:00~18:00)에 변경이 집중되는 사이트의 경

우 밤 12시에 수집을 시작할 수 있을 것이다.

이와 같은 과정을 거쳐 최종적으로는 [표 3.4]와 같은 URL의 수집주기 테이블을 얻는다. 이 테이블에는 시간이 지남에 따라 어느 url을 접근하여 데이터를 가져와야 할지가 기록되어 있다.

표 3.4 URL별로 변경회수 합과 변경크기

URL	요일	시간대	변경횟수합	변경크기합
u1	1	13	1	100
u1	2	11	1	100
u1	2	15	1	20
u1	3	11	1	390
u1	4	9	2	420
u1	4	10	1	400
u1	5	9	1	50
u1	5	10	1	130
u1	5	14	1	400
u1	6	9	2	310
u1	7	9	1	100
u1	7	10	1	30
u10	2	11	1	50

3.3 수집주기의 결정과 크롤러 동작과정

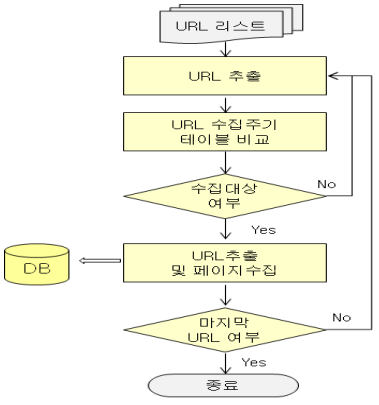


그림 2. 웹크롤링 동적스케쥴 구성도

이러한 수집주기 개념을 활용한 웹크롤러의 일반적인 동작과정은 [그림 2]와 같다. URL은 웹사이트의 상품 리스트가 담긴 페이지를 추출한 후, 리스트페이지에서 관심카테고리의 상품에 대한 URL을 다시 추출한다. 추출된 URL은 키와 값 쌍으로, 카테고리명은 키로, 추출된 URL은 값으로 URL리스트맵(URL값은 집합(set) 컬렉션으로 저장)에 저장되며, URL 리스트맵<key(=카테고리명),value(=URL집합)>으로부터 하나씩 URL을 가져와서 다음 주기에 자료를 가져와야 할 (갱신해야 할) URL이면 해당 페이지를 추출하여 데이터베이스에 저장한다. 그리고 그 페이지에 또 다른 웹페이지 링크들이 포함되어 있으면 이들도 따라가서 해당 페이지의 데이터를 추출하게 된다.

일반적인 웹크롤러에서 문제점으로 꼽히는 중복문제에 대해서 본 논문은 다음과 같은 방식으로 접근하였다. 수집될 URL의 중복수집방지를 위하여 카테고리별 상품 리스트 페이지를 먼저 수집하고, 수집된 리스트 페이지에서 추출할 URL을 URL리스트맵에 저장하는 방식을 사용한다. 이는 기존의 큐(Queue)를 활용할 경우에 데이터 중복에 대한 제약이 없어 야기되는 URL중복 문제를 맵(Map)과 집합(set) 컬렉션을 활용함으로써 해결하였으며, 리스트 페이지가 과도하게 많아 메모리의 용량을 초과하지 않도록 카테고리별 리스트페이지를 수집하고, 그 리스트 페이지에서 다시 URL을 추출 한 후 페이지를 수집하는 방식을 사용한다 (카테고리별로 상품의 품목이 다르기 때문에 다른 카테고리에서 동일한 URL이 나올 확률은 매우 적다고 볼 수 있다).

표 3.5 URL_수집주기 테이블

요일	수집시간	url-list
일	12:00	
	0:00	u1
월	12:00	
	0:00	u1, u2, u3, u4, u5, u10, u11
화	12:00	
	0:00	u1
수	12:00	
	0:00	u1, u2, u3, u4, u5, u10, u11
목	12:00	
	0:00	u1
금	12:00	
	0:00	u1, u2, u3, u4, u5, u10, u11
토	12:00	u6, u7, u8, u9, u12, u15
	0:00	u1
2nd 토	0:00	u13
3th 토	0:00	u14

여기서 중요한 가정은 페이지 리스트에 있는 각각의 URL에는 그 URL의 수집주기 정보가 포함되어 있다는 점이다. 이 정보는 앞에서 설명한 방식으로 결정되어 각 url 에 추가된다.

URL의 수집주기 테이블은 정기적으로 갱신함으로써 최신 상태를 반영하도록 한다. 초기에는 1개월간의 시험운영 결과를 가지고 URL별 갱신정보 테이블[표 3.2]을 작성하고, 이를 바탕으로 수집주기를 결정하였다.

그러나 운영과정에서 [표 3.2]의 변경이력정보가 축적되면 이를 분석하여 현재 상황을 감안한 수집주기를 재생성할 수 있다. 이 때 시간이 지남에 따른 변경정보를 시계열 분석하여 가까운 장래의 변경패턴을 예측하는 것도 가능하지만 본 논문에서는 이를 다루지는 않는다.

4. 시스템 구현과 분석

본 장에서는 제안된 방식으로 구현한 크롤러의 적합성을 평가하고, 이를 다른 데이터에 대한 방식과 비교한다.

먼저, 인터넷 쇼핑몰로부터 상품과 관련된 고객의 VOC (voice of customers) 정보를 추출하여 데이터베이스로 구축하고, 이를 분석하기 위한 크롤러를 제안된 방식으로 구축하여 시험한다. [그림 3]은 개발된 시스템의 초기 웹화면 일부를 보여주고 있다.

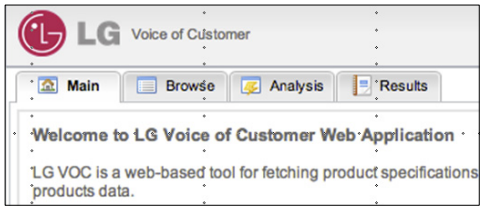


그림 3. VOC 시스템 초기화면

다음으로 이 결과를 기존의 타 데이터에 적용한 결과와 비교한다.

4.1 VOC 데이터 크롤링의 적용결과

시중에 널리 사용되고 있는 대표적인 A 쇼핑몰 사이트로부터 관심있는 정보를 추출하기 위하여 오픈소스 크롤러를 사용한 결과 대용량 데이터를 12일 이상 추출하였으며, 그 중에서 관심있는 데이터는 불과 1% 미만이었다. 한편, 성능문제를 해결하기 위하여 쓰래드를 증가시켜 병렬처리를 하였으며, 이 경우 쓰래드 증가에 따른 서버측의 부담이 급속히 증가하는 심각한 문제가 발생하였다. 이 문제를 해결하기 위하여 래퍼기반 크롤러를 자체적으로 개발하였으며, 그 결과 99% 이상의 정확한 데이터를 30시간 정도 걸려서 수집하였고, 원하는 데이터를 대부분 수집할 수 있게 되었다. [표 4.1]에서 오픈소스와 자체개발 크롤러의 정확도를 평가하였다.

$$\text{정확도} = \frac{\text{상품정보페이지수}}{\text{전체수집페이지수}} \times 100 \quad (1)$$

표 4.1 오픈소스와 자체개발 크롤러의 비교

항목	오픈소스	자체개발
수집기간	12일 + α	30 시간
수집데이터량	36GB	39GB
상품정보개수/전체수집개수	3012/450665	24312/24316
정확도	0.67%	99.98%

4.1.1 대상 URL 리스트

제안된 시스템에서는 관심있는 URL 리스트를 사용자가 직접 해당 사이트의 구조와 내용을 분석한 다음에 수집하여 ETL 시스템으로 입력하게 된다. 특정 페이지에서 연결된 페이지들은 정규식 형태의 패턴을 사용하여 표현할 수도 있다. 현재 A 쇼핑몰에서 제품과 연관된 VOC (voice of customer) 정보가 수록된 28개 url을 선정하였다. 이 url 중에는 정규식을 이용한 패턴을 포함하므로 실제 추출 대상이 되는 URL 개수는 1200개 정도이다.

4.1.2 갱신정보 테이블 구축과 URL 수집주기 결정

1단계에서 수집된 URL 리스트에 대하여 1개월간의 변경주기 정보를 수집한 후, 제안된 방식으로 이를 분석하여 각 URL에 대하여 수집주기를 결정한다. 본 연구에서 사용된 url-list에 대하여 수집주기를 결정한 결과는 다음과 같다.

- 하루 2회 수집하는 페이지들 : 240개 (20%)

- 하루 1회 수집하는 페이지들 : 144개 (12%)
- 주당 3회 : 216개 (18%)
- 주당 1회 : 480개 (40%)
- 2주당 1회 : 60개 (5%)
- 4주당 1회 : 60개 (5%)

4.1.3 평가

제안된 방법은 사용자가 작성한 목표 url 집합을 가지고 크롤링을 시작하므로 목표 url 리스트가 정확하다면 정확도(전체 검색된 것 중에서 관심있는 것의 비율)와 재현율(전체 관심있는 것 검색된 것의 비중)은 모두 1이 된다. 따라서 서버의 부하를 줄이는 정도와 최신성을 보장하는 정도를 분석하는 것이 더 의미가 있다.

- 제안된 방식은 매일 크롤링하는 것에 비하여 서버의 부담은 얼마나 줄이는가 ?

여기서는 다음 수식 (2)과 같이 하루수집 URL개수를 전체 수집URL개수로 나누어 부하율(S(x))를 정의한다.

$$S(x) = \left(\frac{\text{하루 수집 } URL\text{의 개수}}{\text{전체 수집 } URL\text{의 개수}} \right) \times 100 \qquad (2)$$

위의 수식을 기준으로 하루 수집해야 하는 요청을 서버의 부하율 100%로 가정할 경우 제안된 방식의 부하율은 다음과 같다(실제 서버의 부하율을 측정하는 것은 서버의 물리적인 성능과 미들웨어의 성능부분을 일반화하여 계산하기는 어려우므로 수집 페이지를 요청 개수를 서버의 부하율 100%로 가정하고 하루 URL 개수를 특정하여 서버부하율이 어느정도 줄어드는지를 계산).

A쇼핑몰에서 제품과 연관된 VOC 정보가 추록된 28개 url을 선정하였고 실제 하루에 주기적으로 수집하는 url개수는 1200개 정도이다. 그러나, 제안된 방식의 경우 [표 4.2]와 같이 하루에 수집되는 페이지 수가 644개로 계산되므로 기존 방식에 비하여 절반정도(644/1200=53.8%)로 줄어든 것을 알 수 있다.

다음으로 중요한 사항은 제안된 방식이 최신성을 얼마나 잘 반영하는가이다. 기존의 방식은 부하 때문에 매일 1회 이상 데이터를 수집하는 것은 어렵다. 그러나 제안된 방식은 웹페이지의 변동주기 등 특성을 고려하여 하루에도 몇 번씩 가져오는 그룹과 1주일 혹은 1개월에 한번씩 가져오는 그룹들로 구분하므로 최적에 가까운 시점을 판단하여 데이터를 크롤링한다. 따라서 기존 방식보다 뛰어난 최신성을 보장하게 된다.

표 4.2 제안된 방식의 하루평균 요청 페이지수

수집 주기	하루 평균 페이지 요청수
하루 2회 수집	480
하루 1회	144
주당 3회	7
주당 1회	6
2주당 1회	5
4주당 1회	2
계	644

4.2 타 분야 데이터와의 비교

참고문헌 [6]에서는 생물정보학연구센터커뮤니티 BIRC [21] 와 네이트 칼럼의 최신뉴스[22] 두 사이트를 대상으로 데이터 수집주기를 동적으로 적용하는 실험을 수행하였다. 이 방식에서는 웹 데이터 수집주기를 동적으로 변경함으로써 59%의 네트워크 부하 감소를 실현하였다.

제안된 방식은 참고문헌[6]의 방식을 한층 더 실용적으로 개선한 모델이다. 참고문헌[6]에서는 변경의 빈도가 적으면 수집주기를 일정한 값만큼 늘려가고, 반대로 변경의 빈도가 많으면 수집주기를 일정한 값만큼 줄여나가는 방식이지만(값의 크기는 임의로 정함) 크롤러의 수집주기 계산이 정교한 만큼 복잡해진다. 참고문헌[6]에서는 수집된 페이지들의 최신성에 관한 별도의 분석은 논의하지 않았다.

제안된 방식은 1개월 정도의 학습기간 동안에 각 페이지의 갱신빈도를 확인하고, 이에 따라서 적절한 수집주기를 결정하며 (제안된 방식을 주기적으로 적용하면 참고문헌[6]과 유사한 방식이 됨), 특히 변경되는 시간까지 고려하여 수집 시점을 결정함으로써 최신성을 더욱 높이는 방식이다. 또한 VOC 사이트를 대상으로 데이터를 효과적으로 추출하는데 적용함으로써 실용성을 검증하였다.

5. 결론

본 논문에서는 래퍼기반 웹 크롤링에서 url-list의 수집주기를 결정하는 방법을 제시하였다. 주어진 url-list에 대하여 1개월 동안의 변경기록 데이터를 분석하여 각 url의 수집주기를 적절하게 결정하고, 웹 크롤러에서 이를 바탕으로 웹페이지들을 수집함으로써 최신성을 높이면서도 서버의 부하를 줄이는 효과가 있다. 제안된 방식에서는 웹페이지 변경기록 데이터를 지속적으로 축적하고, 이를 주기적으로 분석하여 수집주기를 갱신함으로써 동적으로 웹페이지 수집주기를 변경해 나갈 수도 있다. 분석결과 최신성을 유지하면서도 서버의 부담을 절반정도로 줄이는 것으로 나타났다.

가존의 방식에서는 병렬처리를 위하여 쓰래드를 증가시키면 서버의 부하가 급격하게 높아지게 되어 데이터 수집을 차단하는 등의 문제가 있었으나 제안된 방식은 최신성을 유지하면서도 수집주기를 최소한으로 줄일 수 있으므로 서버 부담 문제를 완화하는데 도움이 된다.

본 논문은 특정 사이트에 대하여 일정기간의 웹페이지 변경 지식을 활용하여 웹페이지 수집주기를 결정하는 실용적인 방식을 제안한 것으로 다양한 유형의 웹사이트 (예를 들어 변경이 잦은 사이트와 그렇지 않은 사이트 등)에 적용하여 그 유용성을 검증하는 것이 필요하다.

참 고 문 헌

[1] 강한훈, 유성준, 한동일, “다양한 계층 트리구조를 갖는 쇼핑몰 상에서의 상품평 수집을 위한 웹크롤러 래퍼의 설계 및 구현”, 한국지능시스템학회 논문지, 제20권, 제3호, pp.318-325, 2010.

[2] 권성호, 이영탁, 김영준, 이용두, “고성능 웹크롤러의 설계 및 구현", 한국산업정보학회논문지, 제8권, 제4호, pp.64-72, 2003.

[3] 고일석, 최우진, 나윤지, 류승렬, “효율적인 웹문서 처리를 위한 HTTP 지연 개선에 관한 연구”, 한국콘텐츠학회논문지, 제2권, 제2호, pp.47-52, 2002.

[4] 김광영, 이원구, 이민호, “웹 자원 아카이빙을 위한 웹 크롤러 연구 개발”, 한국콘텐츠학회논문지, 제11권, 제9호, pp.9-16, 2011.

[5] 김성진, “웹 정보탐색행위 모형의 비교분석 연구”, 정보처리학회지, 제21권, 제2호, pp.211-233, 2004.

[6] 김경수, *웹 크롤링 수집주기의 동적 설계 및 구현*, 충북대학교 경영대학원 석사논문, 2011.

[7] 장문수, 정준영, “URL 패턴 스크립트를 이용한 효율적인 웹문서 수집방안”, 퍼지 및 지능시스템학회 논문지, 제17권, 제6호, pp.849-854, 2007.

[8] 황인수, “웹의 연결구조와 웹문서의 적합도를 이용한 효율적인 인터넷 정보추출”, 정보기술과 데이타베이스 저널, 제11권, 제4호, pp.49-60, 2004.

[9] C. Bertoli, V. Vrescenzi, and P. Merialdo, "Crawling Programs for Wraller-based Applications," In Proc. IEEE Intl. Conference on Information Reuse and Integration (IRI '08), pp.160-165, 2008.

[10] J. H. Cho, *Crawling the Web: Discovery and maintenance of Large-Scale Web Data*, Ph. D. Dissertation, Stanford University, 2001.

[11] S. Chakrabarti, M. van den Berg, and B. Dom, "Focused Crawling: A new Approach to Topic-Specific Web Resource Discovery," Computer Networks, Vol.31, No.11-16, pp.1623-1640, 1999.

[12] *TeraStream 제품소개서*, (주)데이터스트림즈 (www.datastreams.co.kr), 2008.

[13] Z. Guan, C. Wang, C. Chen, J. Bu, and J. Wang, "Guide Focused Crawler Efficiently and Effectively Using On-line Topical Importance Estimation," In Proc. of ACM SIGIR Conference on Research and Development in Infoprmation Retrieval, pp.757-758, 2008.

[14] B. He, C. Li, D. Killian, M. Patel, Y. Tseng, and K. C. C. Chang, "A Structure-Driven Yield-Aware Web Form Crawler: Building a Database of Online Databases," UIUC Technical

Report, 2006.
[15] J. Y. Yang, T. H. Kim, and J. M. Choi, "An Interface Agent for Wrapper-based Information Extraction," In Proc. Intl. Conf. on Principles of Practice in Multi-agent Systems(PRIMA '04), pp.291-302, 2004.
[16] Karthikeyan Anbarasan, *SQL Integration Services (SSIS) - Step by Step Tutorial*, in A SSIS eBook (www.f5Debug.net), 2011.
[17] Liu, Bing, *Web data mining: exploring hyperlinks, contents, and usage data*, Springer Verlag, 2007.
[18] G. Pant, P. Srinivasna, and F. Menczer, "Crawling the web," In Web Dynamics, pp.153-177, 2004.
[19] M. L. Vidal, A. S. da Silva, E. S. de Moura, and J. M. B. Cavalcanti, "GoGetIt!: a tool for generating structure-driven web crawlers," In Proc. 15th international conference on World Wide Web, pp.1011-1012, 2006.
[20] 김성진, 이상호, "웹 문서 변화에 관한 실험적 연구", 정보과학회논문지 : 데이터베이스, 제32권, 제2호, pp.151-160, 2005.
[21] http://bric.postech.ac.kr/myboard/list.php?Board=exp_qna
[22] http://news.nate.com/recent?cate=col&mid=n0108&type=t

저 자 소 개

조 완 섭(Wan-Sup Cho) 정회원

- 1985년 2월 : 경북대학교(학사)
- 1987년 2월 : KAIST(석사)
- 1996년 2월 : KAIST 전산학과(박사)
- 1987년 1월 ~ 1990년 12월 : ETRI 연구원
- 1997년 3월 ~ 현재 : 충북대학교 경영정보학과 교수

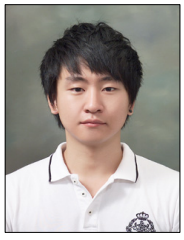
<관심분야> : DB, BI, ERP, 빅데이터

이 정 은(Jeong-Eun Lee) 준회원

- 2011년 2월 : 충북대학교 컴퓨터공학과(공학사)
- 2012년 3월 ~ 현재 : 충북대학교 비즈니스데이터융합학과 석사과정

<관심분야> : 빅데이터, 비즈니스 인텔리전스, ERP

최 치 환(Chi-Hwan Choi) 정회원



- 2009년 2월 : 충북대학교 경영정보학과(학사)
- 2011년 2월 : 충북대학교 바이오정보기술학과(석사)
- 2011년 3월 ~ 현재 : 충북대학교 바이오정보기술학과 박사과정

<관심분야> : 빅데이터, DB, 바이오인포매틱스, 분산컴퓨팅 플랫폼, 서버 아키텍처, BI, 데이터마이닝