J. Internet Comput. Serv.
ISSN 1598-0170 (Print) / ISSN 2287-1136 (Online)
http://www.jics.or.kr

최신 웹 크롤링 알고리즘 분석 및 선제적인 크롤링 기법 제안☆

A proposal on a proactive crawling approach with analysis of state-of-the-art web crawling algorithms

나 철 원[1] 온 병 원[2*]
Chul-Won Na Byung-Won On

요 약

오늘날 스마트폰의 보급과 SNS의 발달로 정형/비정형 빅데이터는 기하급수적으로 증가하였다. 이러한 빅데이터를 잘 분석한다면 미래 예측도 가능할 만큼 훌륭한 정보를 얻을 수 있다. 빅데이터를 분석하기 위해서는 먼저 대용량의 데이터 수집이 필요하다. 이러한 데이터가 가장 많이 저장되어 있는 곳은 바로 웹 페이지다. 하지만 데이터의 양이 방대하기 때문에 유용한 정보를 가진 데이터가 많은 만큼 필요하지 않은 정보를 가진 데이터도 많이 존재한다. 그렇기 때문에 필요하지 않은 정보를 가진 데이터는 거르고 유용한 정보를 가진 데이터만을 수집하는 효율적인 데이터 수집의 중요성이 대두되었다. 웹 크롤러는 네트워크 대역폭, 시간적인 문제, 하드웨어적인 저장소 등의 제약으로 인해 모든 페이지를 다운로드 할 수 없다. 그렇기 때문에 원하는 내용과 관련 없는 많은 페이지들의 방문은 피하며 가능한 빠른 시간 내에 중요한 페이지만을 다운로드해야한다. 이 논문은 위와 같은 이슈의 해결을 돕고자한다. 먼저 기본적인 웹 크롤링 알고리즘들을 소개한다. 각 알고리즘마다 시간복잡도와 장단점을 설명하며 비교 및 분석한다. 다음으로 기본적인 웹 크롤링 알고리즘의 단점을 개선한 최신 웹 크롤링 알고리즘들을 소개한다. 더불어 최근 연구 흐름을 보면 감성어휘 수집과 같은 특수한 목적을 가진 웹 크롤링 알고리즘의 대한 연구가 활발히 이루어지고 있다. 특수 목적을 가진 웹 크롤링 알고리즘에 대한 연구로써 선제적인 웹 크롤링 기법으로 감성 반응 웹 크롤링(Sentiment-aware Web Crawling) 기법을 소개한다. 실험결과 데이터의 크기가 커질수록 기존방안보다 높은 성능을 보였고 데이터베이스의 저장 공간도 절약되었다.

☞ 주제어 : 웹 페이지, 웹 크롤링 알고리즘, 웹 크롤러

ABSTRACT

Today, with the spread of smartphones and the development of social networking services, structured and unstructured big data have stored exponentially. If we analyze them well, we will get useful information to be able to predict data for the future. Large amounts of data need to be collected first in order to analyze big data. The web is repository where these data are most stored. However, because the data size is large, there are also many data that have information that is not needed as much as there are data that have useful information. This has made it important to collect data efficiently, where data with unnecessary information is filtered and only collected data with useful information. Web crawlers cannot download all pages due to some constraints such as network bandwidth, operational time, and data storage. This is why we should avoid visiting many pages that are not relevant to what we want and download only important pages as soon as possible. This paper seeks to help resolve the above issues. First, We introduce basic web-crawling algorithms. For each algorithm, the time-complexity and pros and cons are described, and compared and analyzed. Next, we introduce the state-of-the-art web crawling algorithms that have improved the shortcomings of the basic web crawling algorithms. In addition, recent research trends show that the web crawling algorithms with special purposes such as collecting sentiment words are actively studied. We will one of the introduce Sentiment-aware web crawling techniques that is a proactive web crawling technique as a study of web crawling algorithms with special purpose. The result showed that the larger the data are, the higher the performance is and the more space is saved.

☞ keyword : Web Page, Web Crawling Algorithms, Web Crawler

1 Department of Software Convergence Engineering, Kunsan National University., Gunsan-si Jeollabuk-do, 54150, Korea.
2 Department of Software Convergence Engineering, Kunsan National University., Gunsan-si Jeollabuk-do, 54150, Korea.
* Corresponding author: (bwon@kunsan.ac.kr)
[Received 24 December 2018, Reviewed 26 December 2018(R2 11 March 2019), Accepted 4 April 2019]

1. 서 론

스마트폰의 보급과 SNS의 발달로 정형/비정형 데이터 양이 기하급수적으로 증가하여 빅데이터 시대가 도래하였다. 이러한 빅데이터를 잘 분석한다면 미래 예측도 가능할 만큼 훌륭한 정보를 얻을 수 있다. 오늘날은 이러한 데이터에서 가치 있는 정보를 추출해 분석하는 것으로 경쟁하는 시대이다. 빅데이터를 분석하기 위해서는 우선적으로 대용량 데이터 수집이 필요하다. 이러한 데이터는 데이터들은 주로 웹 페이지 상에 존재한다. 데이터가 넘쳐나는 웹에서 웹 크롤링을 통해 업무를 자동화하고 유용한 정보를 찾을 수 있다. 하지만 데이터의 양이 방대하기 때문에 유용한 정보를 가지고 있는 데이터가 많은 만큼 필요하지 않은 정보를 가지고 있는 데이터도 많아졌다. 가능한 짧은 시간 내에 원하는 내용과 관련 있는 데이터를 가능한 많이 추출해내는 일은 매우 중요한 문제가 되었다. 짧은 시간 동안 대량의 웹 페이지를 통과하며 관련성에 따라 데이터를 수집할 수 있는 웹 크롤링 알고리즘을 사용함으로써 데이터 수집의 효율을 개선하기 위한 많은 연구들이 진행 중이다. 적시적인 데이터 수집은 생존을 위한 해결책이 된다. 현재 구글은 만 30억이 넘는 웹 페이지를 색인화하였고, 웹은 9-12개월 마다 두 배로 증가하며 변화율은 매우 급격하다[1, 2, 3]. 웹 크롤러를 통하여 필요한 데이터를 수집하기 위해 이러한 수많은 웹 페이지들을 거쳐야한다. 하지만 웹 크롤러는 네트워크 대역폭, 시간적인 문제, 하드웨어적인 저장소 등의 제약으로 인해 모든 페이지를 다운로드 할 수 없다. 그렇기 때문에 상황에 맞게 적절한 웹 크롤링 알고리즘을 선택하여 하는 내용과 관련 없는 많은 페이지들의 방문은 피하며 가능한 빠른 시간 내에 중요한 페이지만을 중복되지 않게 다운로드해야한다.

본 논문에서는 기존의 기본 웹 크롤링 알고리즘들을 이해하여 상황에 알맞은 알고리즘을 선택하고 더 효율적인 크롤링을 할 수 있는 개선된 웹 크롤링 알고리즘을 개발하는 것에 도움이 되는 것을 목적으로 한다. 우선 웹 크롤링에 관하여 알아야할 기본적인 사항들에 대해서 소개하며 웹 크롤러를 구현할 때 사용되는 기본적인 웹 크롤링 알고리즘을 소개한다. 소개된 각 알고리즘의 장단점과 시간복잡도를 설명하며 비교 및 분석한다. 다음으로 기본적인 웹 크롤링 알고리즘들의 단점을 개선한 최신 웹 크롤링 알고리즘을 소개한다. 대표적으로 'Breadth Fisrst Search' 알고리즘의 반복적인 노드들의 문제로 전체 프로세스 효율이 저하되는 문제점을 개선한 'An enhanced Breadth First Search' 알고리즘과 'Shark-Search' 알고리즘의 노이즈 링크 문제점을 개선한 'Shark-Search based on Multi Granularity' 알고리즘이 있다. 이러한 기본 웹 크롤링 알고리즘의 단점을 개선시킨 알고리즘뿐만 아니라 특수한 목적을 가진 웹 크롤링 알고리즘도 소개한다. 웹에 존재하는 데이터의 다양성과 각 사용자마다 원하는 데이터가 다르기 때문에 크롤러는 각각 다른 목적을 가지게 되었다. 최근 연구동향을 보면 감성분석을 위한 감성어휘 수집과 같은 특수한 목적을 가진 웹 크롤링 알고리즘에 대한 연구가 활발히 이루어지고 있다. 이처럼 특수한 목적을 가진 선제적 웹 크롤링 기법인 감성 반응 웹 크롤링 기법을 소개한다. 감성어휘를 수집하기 위해 기존에는 일단 모든 문서를 수집하고 수집된 전체 문서를 스캔하여 감성어휘의 유무를 판단하였다. 감성어휘 수집을 위한 기존방안은 일단 모든 문서를 저장하기 때문에 저장소가 낭비되고 저장된 전체 문서를 스캔하기 때문에 시간이 오래 걸리는 문제가 발생한다. 반면 감성 반응 웹 크롤링은 수집을 하는 동시에 감성어휘의 유무를 판단하여 필터링 후 저장한다. 실험결과 기존방안과 비교하여 감성 반응 웹 크롤링 알고리즘을 사용하였을 때 데이터의 크기가 증가할수록 높은 성능을 보였으며 데이터베이스의 저장 공간도 절약되었다. 이처럼 앞으로는 특수한 목적에 필요한 정보만 가지고 있는 데이터를 효율적으로 수집하는 방향의 연구가 진행되어야 한다.

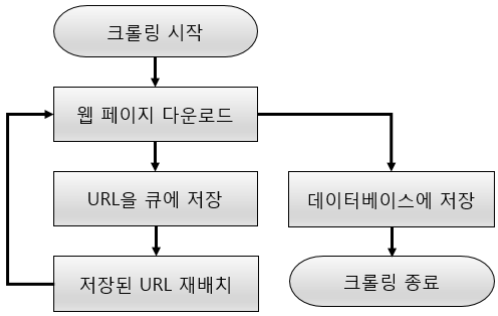
본 논문의 구성은 다음과 같다. 1장에서는 서론을 이야기하며 2장은 웹 크롤링의 기본사항에 대해 설명한다. 3장은 기존의 웹 크롤링 알고리즘들에 대해 설명한다. 더불어 앞서 설명한 각 알고리즘들의 요약한 정보를 비교하고 분석하며 재해석한다. 4장은 기존의 크롤링 알고리즘을 개선한 최신 크롤링 알고리즘과 더불어 특수한 목적을 가진 선제적 웹 크롤링 알고리즘인 감성 반응 웹 크롤링 알고리즘을 소개한다. 5장은 실제 크롤링에 도움이 될 만한 관련 툴을 소개한다. 6장에서는 결론을 다루며 마무리한다.

2. 웹 크롤링의 기본사항

이 장에서는 웹 크롤링에 관련된 간단한 용어 정리와 기본적으로 어떻게 동작하는지에 대해 설명한다. 또한 이러한 웹 크롤링이 갖고 있는 고유한 문제에 대해 설명하고, 웹 크롤링을 위해 알아 두면 도움이 될 만 한 특징들을 설명한다.

2.1 웹 크롤링 주기

기본적으로 동작하는 웹 크롤링 과정의 흐름도는 다음 그림 1과 같다.



(그림 1) 웹 크롤링 주기 흐름도
(Figure 1) Web Crawling Cycle Flowchart

먼저, 크롤링이 시작되면 초기 웹 페이지를 다운로드 한 후 웹 페이지에 있는 URL들을 큐에 저장한다. 큐에 있는 URL들을 중요도에 따라 재배치 해준다. 재배치 된 URL 큐에서 우선순위가 높은 URL에 해당하는 웹 페이지를 다운로드 한다. 다운로드하고, 큐에 URL 저장하고, URL 스케줄링 한다. 이와 같은 과정을 계속 반복한다. 모든 URL을 방문하게 되면 이 반복은 끝나게 된다. 반복이 끝나게 되면 필요한 데이터를 저장소에 저장하고 크롤링을 마치게 된다.

하지만 웹 크롤링이 동작하는 상세한 과정은 크롤러의 종류나 웹 크롤링 알고리즘에 따라 기본 사이클의 구성 요소가 추가되거나 변경될 수 있다.

2.2 웹 크롤링의 고유한 문제

웹 크롤링은 여러 고유한 문제들을 가지고 있다. 이러한 문제들을 해결할 수 있는 더 발전된 크롤링 알고리즘들이 필요하다. 다음은 웹 크롤링이 갖는 대표적인 문제들이다.

- Scale, 웹은 규모가 크고 계속 진화하고 있다. 웹 크롤링을 하는 과정에서 매우 높은 처리량을 달성하기 위해 많은 엔지니어링 문제가 발생한다.
- Content Selection Tradeoffs, 가치가 높은 콘텐츠는 수집하고, 품질이 낮거나 관련성이 없거나 악의적인 콘텐츠는 우회한다. 사이트의 제약 조건을 준수하는 것과 경쟁 목표(범위, 최신성) 사이의 균형을 유지하는 것이 필요하다.
- Social Obligations, 크롤링 하는 웹 사이트에 너무 많은 부담을 주지 않아야 한다. 올바른 보안 메커니즘이 필요하다.
- Adversaries, 일부 콘텐츠 제공자가 상업 목적으로 크롤러에 유용하지 않거나 오해의 소지가 있는 콘텐츠를 주입하는 경우가 있다.

2.3 웹 크롤링을 위한 특별한 특징

웹 크롤링을 하기 전에 알고 있으면 도움이 되는 몇 가지 특징들이 있다. 이러한 특징들을 미리 알고 있으면 어떠한 문제가 발생하였을 때 대처할 수 있을 것 이다. 다음은 웹 크롤링을 위해 알아두면 좋은 특징들이다.

- 접근 관리, 단일 웹 서버로 접근할 때 짧은 시간 동안 많은 수의 방문을 자제하도록 한다. 이것은 짧은 시간 동안 크롤링 대상이 되는 웹 서버에 과부하를 발생시키지 않기 위한 중요한 기능이다. 이런 기능이 없다면 대상 웹 서버에서는 많은 수의 방문을 DDoS 공격으로 오인할 수 도 있다.
- 함정 피하기, 크롤링 진행 중에 의미 없는 URL을 무한대로 발생시키는 특정 도메인에 갇혀 빠져 나오지 못하는 경우가 발생할 수 있다. 대표적인 예로 웹에서 구현된 달력이다. 달력의 경우에 웹에서 자동으로 다음 날짜들에 대해 URL을 생성하는 경우, 의미 있는 데이터를 수집 못 할 뿐 더러 무한루프에서 빠져 나오지 못하고 헤매게 되는 문제점이 발생한다. 이런 달력 같은 경우 calendar와 같은 키워드가 URL에 포함되어 있으면 접근하지 않도록 하는 URL Filtering 기법을 이용한다.
- 로봇 배제 표준, 웹 사이트에 로봇이 접근하는 것을 방지하기 위한 규약이다. 일반적으로 접근 제한에 대한 설명을 robots.txt에 기술한다. 이 규약은 1994년 6월에 처음 만들어 졌다. 이 규약은 권고안이고, 로봇이 robots.txt 파일을 읽고 접근을 중지하는 것을 목적으로 한다. 따라서 접근 방지 설정을 하였다고 해도, 로봇이 아닌 다른 사람들은 그 파일에 접근할 수 있다.
- URL 정규화, 물리적으로 동일한 웹 사이트일지라도 논리적으로는 서로 다르게 표현될 수 있다. 예를 들어 다음과 같이 문자가 서로 다른 URL이지만 물

리적으로 동일한 웹 사이트를 가리키는 경우가 있다.
-http://www.kunsan.ac.kr
-http://www.kunsan.ac.kr/
-http://www.kunsan.ac.kr/index.kunsan
-http://www.kunsan.ac.kr:8000

이처럼 하나의 웹 사이트에 논리적으로 다른 형태의 URL이 존재할 수 있기 때문에, 정규화를 통해 이 URL 들을 하나의 일관된 형태로 표현하는 처리가 필요하다.

- HTML Parsing, 일반적으로 웹 사이트는 주로 HTML 문서로 이루어져 있다. 하지만 이 HTML 문법은 견고하지 않다. 실제로 여러 웹 사이트를 확인해보면 빈번하게 HTML 표준 문법이 지켜지지 않는 것을 발견할 수 있다. 크롤러는 이처럼 정확한 문법을 지키지 않은 채 작성된 HTML 문서에 대해서도 파싱이 가능해야 한다.

3. 기본 웹 크롤링 알고리즘

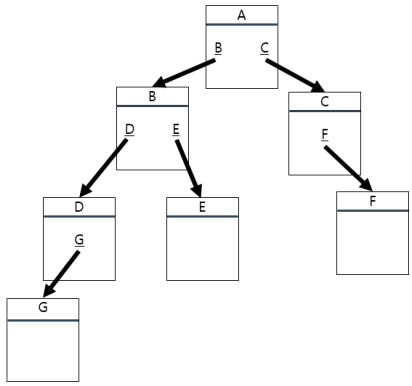
이 장에서는 기본 웹 크롤링 알고리즘들에 대해 설명한다. 기존 알고리즘의 기본적인 원리 및 장단점과 적용 분야에 대하여 소개하고 요약 및 분석한다.

3.1 기본 웹 크롤링 알고리즘 소개

3.1.1 Breadth First Search

너비 우선 탐색인 Breadth First Search 알고리즘은 가장 기본적인 알고리즘이다. 루트 노드부터 가까운 노드들을 먼저 방문하고 멀리 떨어져 있는 노드들을 나중에 방문하는 탐색 방법이다. 트리 구조인 노드들의 깊게 탐색하기 전에 넓게 탐색하는 방법이다. 크롤링의 궁극적인 목표인 전체 웹을 포괄하는 것에 잘 맞는 알고리즘이다[5].

그림 2에서 각각의 알파벳이 웹 페이지의 URL, 각 웹 페이지 안에 있는 알파벳들은 하이퍼링크가 걸려 있다고 생각한다. A점을 깊이가 1이라고 보면 각 점까지 연결된 선마다 깊이가 1씩 증가한다고 생각할 수 있다. 그렇다면 깊이가 2인 점은 'B, C', 깊이가 3인 점은 'D, E, F', 깊이가 4인 점은 'G'로 볼 수 있다. 깊이가 같으면 형제관계, 선이 연결되어있고 깊이가 깊으면 자식관계라고 생각한다. 너비 우선 탐색 알고리즘은 형제관계에 우선순위를 두어 방문한다. 가장 먼저 root URL인 A를 방문하고, 그 다음에 A와 연결된 깊이가 2인 B를 방문하고 형제관계가 우선순위 이므로 B와 형제관계인 C 순서로 방문을 한다. 그 다음에 깊이가 3인 D를 방문하고 D와 형제관계인 E, F 순서로 방문한다. 마지막으로 깊이가 4인 G를 방문한다.



(그림 2) 트리 구조의 웹 페이지
(Figure 2) Tree-structured web page

큰 트리 구조에서 중요한 정보가 깊이가 얕은 부분에서 많이 발견될 때 적합하기 때문에 SNS 친구의 친구 찾기 같은 경우에 활용된다. 하지만 트리 구조에서 가지가 많은 무한히 넓은 트리에서는 비효율적이다. 이 알고리즘이 가지고 있는 문제점을 개선시키기 위하여 제안된 'An enhanced Breadth First Search' 라는 방안을 5장에서 소개한다.

3.1.2 Depth First Search

깊이 우선 탐색인 Depth First Search 알고리즘은 root URL에서 시작하여 자식 URL을 통과하여 깊이 탐색하는 알고리즘이다[5].

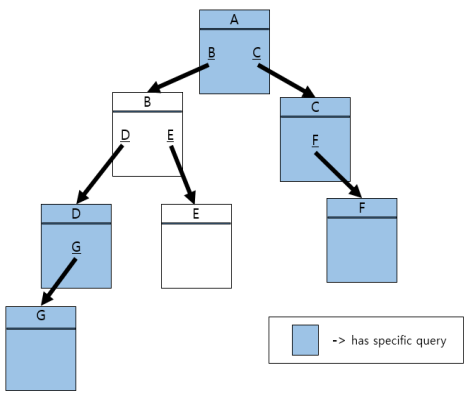
그림 2에서 각각의 알파벳이 웹 페이지의 URL, 각 웹 페이지 안에 있는 알파벳들은 하이퍼링크가 걸려 있다고 생각한다. A점을 깊이가 1이라고 보면 각 점까지 연결된 선마다 깊이가 1씩 증가한다고 생각할 수 있다. 그렇다면 깊이가 2인 점은 'B, C', 깊이가 3인 점은 'D, E, F', 깊이가 4인 점은 'G'로 볼 수 있다. 깊이가 같으면 형제관계, 선이 연결되어있고 깊이가 깊으면 자식관계라고 생각한다. 깊이 우선 탐색 알고리즘은 자식관계에 우선순위를 두어 방문한다. 가장 먼저 root URL인 A를 방문하고, 그 다음에 A와 연결된 깊이가 2인 B를 방문하고 자식관계가 우선순위 이므로 B와 자식관계인 알파벳 점들을 방문한다. B의 자식 관계인 D, D와 자식관계인 G 순서로 방문한다. 그 다음 D와 형제관계인 E를 방문한다. B와 형제관계

인 C를 방문하고 마지막으로 C와 자식관계인 F를 방문한다.

큰 트리 구조에서 중요한 정보가 깊이가 깊은 부분에서 많이 발견될 때 적합하기 때문에 일련의 검색어를 입력 받아 그와 관련된 웹 페이지를 불러오는 검색 엔진에서 활용된다. 하지만 트리 구조에서 가장 깊은 노드까지 방문하기 때문에 트리 구조가 큰 경우에는 알고리즘이 무한 루프에 빠질 수 있다. 때문에 대용량 데이터를 다루는 크롤링의 경우 이 알고리즘을 선호하지 않는다. 이 알고리즘을 사용하더라도 보통 그냥 사용하지 않고 깊이에 제한을 걸어두고 사용한다. 이 알고리즘이 가지고 있는 문제점을 개선시키기 위하여 제안된 'Modified version of Depth First Search Algorithm' 와 'A Cohesive Page Ranking and Depth-First Crawling' 라는 방안을 5장에서 소개한다.

3.1.3 Fish School Search

Fish School Search 알고리즘은 일련의 시드 페이지부터 시작하여 지정된 쿼리인 키워드나 정규 표현식과 일치하는 내용만 있는 페이지만을 고려한다. 크롤러는 지정된 쿼리가 포함된 정보를 찾아서 웹을 탐색한다. 지정된 쿼리를 발견하면 저장하고 다시 반복한다. 지정된 쿼리를 찾지 못하거나 대역폭 부족이면 종료한다. Fish School Search는 시드 페이지에서 고정된 깊이로만 탐색한다[6].



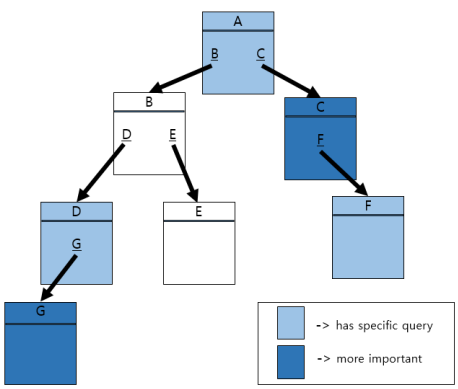
(그림 3) 쿼리를 포함한 트리 구조의 웹 페이지
(Figure 3) Tree-structured web page including query

그림 3에서 각각의 알파벳이 웹 페이지의 URL, 각 웹 페이지 안에 있는 알파벳들은 하이퍼링크가 걸려 있다고 생각한다. A는 홈 페이지의 root URL을 의미한다. 색이 없는 페이지는 지정된 쿼리와 일치하는 내용이 없다는 것을 의미한다. 반대로 A와 같은 색을 가지고 있는 페이지는 A와 지정된 쿼리와 일치하는 내용이 있다는 것을 의미한다. A와 같은 색인 C, D, F, G만을 저장한다.

사용자가 원하는 어떤 토픽에 대한 내용을 탐색하고 싶은 경우에 적합하기 때문에 사용자가 검색어를 입력했을 때 검색어를 포함한 웹 페이지 만을 반환해주는 검색 엔진에 활용된다. 하지만 고정된 깊이로만 탐색하기 때문에 대용량의 데이터를 수집하고자 할 때에는 비효율적이다. 이러한 문제점을 개선시키기 위하여 'Shark Search'라는 알고리즘이 제안되었다.

3.1.4 Shark Search

Shark Search는 fish school search를 개선한 알고리즘이다. 페이지 관련성을 결정하기 위해 코사인 측정과 함께 TF/IDF의 가중 방법을 사용한다. 크롤링해야 할 URL들을 페이지의 중요도에 따라 우선순위를 재정렬한다. 페이지의 중요도는 페이지를 가리키는 수에 의해 정해진다[6].



(그림 4) 쿼리를 포함한 트리 구조의 웹 페이지
(Figure 4) Tree-structured web page including query

그림 4. 에서 각각의 알파벳이 웹 페이지의 URL, 각 웹 페이지 안에 있는 알파벳들은 하이퍼링크가 걸려 있다고 생각한다. A는 홈 페이지의 root URL을 의미한다. 색이 없는 페이지는 지정된 쿼리와 일치하는 내용이 없다는 것을 의미한다. 반대로 색을 가지고 있는 페이지는 A와 지정된 쿼리와 일치하는 내용이 있다는 것을 의미한다. 더 짙은 색을 가지고 있는 페이지는 코사인 측정과 TF/IDF의 가중 방법을 사용하여 페이지의 중요도가 높다는 것을 의미한다. A의 색보다 더 짙은 색인 C, G를 우선순위로 저장하고, 같은 색인 D, F를 저장한다.

코사인 측정과 함께 TF/IDF 가중 방법을 사용하기 때

문에 원하는 페이지와의 관련성이 더 높은 페이지들 가져올 수 있기 때문에 'Fish School Search' 알고리즘 보다 효율적으로 검색어를 포함한 웹 페이지만을 반환해준다. 하지만 관련성 판단을 위해 정보 검색(IR)에 기반한 기술에 의존하게 된다는 단점이 있다. 이러한 문제점을 개선시키기 위하여 제안된 'Shark-search Algorithm based on Multi Granularity'라는 방안을 5장에서 소개한다.

3.1.5 By HTTP Get Request and Dynamic Web Page

이 알고리즘은 HTTP Get Request와 동적 웹 페이지를 사용하여 웹 크롤러 또는 스파이더 트래픽을 최소화하는 쿼리 기반 접근 방식이다. HTTP Get Request 및 동적 웹 페이지를 사용하여 웹 크롤러가 웹 사이트의 모든 업데이트를 알리는 쿼리 기반 접근 방식이다[7]. 이해를 돕기 위해 그림 3.을 보면 쿼리를 웹 사이트의 업데이트 정보라고 생각한다. 색이 있는 경우에 업데이트가 되었다고 생각한다. root URL인 A를 크롤링하고 이후 업데이트 된 C를 크롤링한다. 그 이후 또 업데이트 된 D, F, G까지 크롤링을 하게 된다.

이 알고리즘을 사용하는 크롤러는 마지막으로 업데이트 된 웹 페이지를 방문하기 전에는 볼 수 없지만 마지막 방문 이후에 최신 업데이트 된 웹 페이지만을 다운로드 할 수 있기 때문에 검색 엔진과 같은 사이트에서 데이터를 최신 상태로 유지하는 경우에 활용된다.

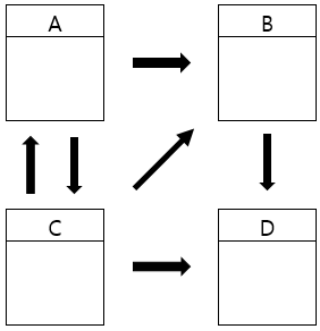
3.1.6 By the Use of Filter

이 알고리즘은 필터를 사용하여 크롤링에 대한 개선된 방법을 제안했고, 쿼리 기반 접근 방식이다. 필터는 항상 업데이트 된 웹 페이지를 리다이렉션 하고 크롤러는 마지막 방문 이후에 업데이트 된 모든 웹 페이지를 다운로드 한다[7]. 이해를 돕기 위해 그림 3.을 본다. 색이 있는 경우에 업데이트가 되었다고 생각한다. 페이지 사이의 화살표에 필터가 적용된다고 생각한다. root URL인 A를 크롤링하고 C를 크롤링 할 때 필터에서 업데이트가 되었기 때문에 리다이렉션 하고 마지막 방문 이후 업데이트 된 모든 웹 페이지를 다운로드 한다. 이런 식으로 D, F, G도 크롤링한다.

모든 웹 페이지를 방문하지 않고 필터에서 업데이트 된 웹 페이지만을 방문하기 때문에 웹 크롤링 또는 네트워크 트래픽을 줄여 검색 엔진에서 최신 업데이트 된 데이터를 확인하는 경우에 활용된다. 크롤러가 추가 작업을 수행하기 위해서는 추가 필터가 사용된다.

3.1.7 Page Rank

Page Rank 알고리즘은 웹 사이트에서 백 링크 수 또는 인용 횟수에 따라 웹 페이지의 중요도를 결정한다. 각 웹 페이지의 순위는 Page Rank 알고리즘에 따라 중요도가 계산된다[7].



(그림 5) 웹 페이지 네트워크
(Figure 5) Network of Web page

그림 5.는 웹 페이지 네트워크의 예를 나타낸다. 알파벳은 각 페이지의 URL이다.

$$PR(Page\ Rank) = \frac{PR of\ previous\ step}{\#\ of\ incoming\ node's\ outlink}$$

PR(Page Rank)값은 전 단계에서의 PR값을 들어오는 노드의 아웃링크개수의 합으로 나눈 것으로 구할 수 있다. 일단 각 페이지 마다 초기단계인 step0의 PR 값은 1/n의 값인 1/4를 부여한다. A로 들어오는 페이지는 C하나이다. C는 외부로 나가는 아웃링크는 총 3개이다. step1에서의 A의 PR 값은 step0에서의 C의 PR 값인 1/4에서 3을 나눠 1/12이 된다. 들어오는 노드가 여러 개일 경우에는 더해주면 된다. B의 PR값은 A와 C에서 들어오고 있으므로 1/4에서 2를 나눈 값과 1/4에서 3을 나눈 두 값을 더한 2.5/12가 된다. 같은 방법으로 C와 D의 PR값을 구한다. step1에서 구한 각 페이지의 값을 활용하여 같은 방식으로 step2를 계산한다. step2에서 값이 높을수록 더 많은 랭크를 갖고 있어 중요하다는 것을 의미한다.

(표 1) 페이지 랭크 결과
(Table 1) Result of Page Rank

	Step0	Step1	Step2	Page Rank
A	1/4	1/12	1.5/12	1
B	1/4	2.5/12	2/12	2
C	1/4	4.5/12	4.5/12	4
D	1/4	4/12	4/12	3

표 1.을 보면 C가 step2에서 값이 제일 높아 Page Rank 값이 4로 가장 높다. 이 예제에서 가장 중요한 역할을 하여 우선순위가 높다는 것으로 해석할 수 있다.

Page Rank 알고리즘은 웹 페이지에 대해서 우선순위 지정이 안정적이며 효과적이다. 대표적인 검색 엔진인 '구글(Google)'에서 검색 결과를 향상시키기 위해 이 알고리즘을 활용하여 실제로 월등히 좋은 검색 결과를 실현해 보였다. 하지만 웹 페이지의 관련성과 동적성에 대해서는 고려하지 않는다는 단점이 있다. 이 알고리즘이 가지고 있는 문제점을 개선시키기 위하여 제안된 'A Cohesive Page Ranking and Depth-First Crawling' 라는 방안을 5장에서 소개한다.

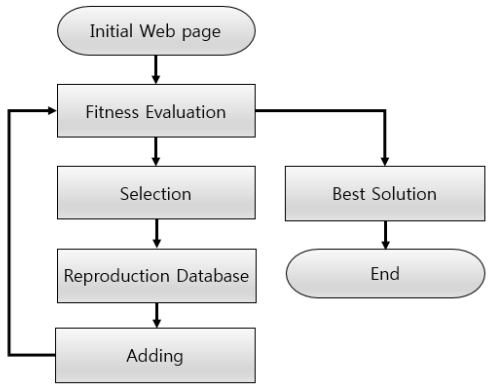
3.1.8 Online Page Importance Calculation (OPIC)

각 페이지마다 모든 링크에 캐시 값이 균등하게 분배되어 있으며 처음에는 모든 페이지의 1/n과 같다. 그림 5.를 보면 크롤링을 할 때 크롤러는 페이지를 다운로드하고 페이지를 다운로드 할 때 마다 페이지가 가리키는 모든 페이지에 현재 캐시를 균등하게 분배한다[7]. 즉, root URL이 A라고 가정하면 A를 크롤링하게 되면 A가 가르키는 페이지인 C와 B에 A가 가지고 있는 캐시 값을 2로 나누어 캐시를 분배한다. 페이지에 방문하기 전에 페이지가 잠시 기다리면 캐시가 누적되어 다음 방문 시에 더 많이 분배된다. 이런 식으로 캐시 값이 분배되고 각 페이지는 고유한 캐시 값을 갖게 된다. 크롤러는 캐시 값이 큰 웹 페이지를 우선으로 다운로드한다.

이 알고리즘은 캐시 값이 한 단계로 계산된다. 그렇기 때문에 매우 짧은 시간동안 계산이 되어서 부하가 많이 걸려있는 웹 페이지에서 빠르게 접근할 때 활용된다. 하지만 각 페이지가 여러 번 다운로드 되어 크롤링 시간이 늘어날 수 있다는 단점이 있기 때문에 각 페이지의 다운로드 횟수에 제한을 거는 방식으로 이 알고리즘을 개선시키는 연구의 필요성이 있다.

3.1.9 Genetic Algorithm (GA)

Genetic Algorithm은 자연 선택과 유전학의 원리에 기반한 검색 알고리즘이다. 문제 영역에서의 적합도 수준에 따라 개체를 선택하고 자연에서 수행된 유전 과정에서 빌린 연산자를 사용하여 개체를 함께 학습하는 과정에서 새로운 근사 값 집합이 만들어진다[7]. 기술적으로 말하면 어떠한 집합에서 적합한 함수를 통해 가장 우수한 개체를 선별한다고 볼 수 있다. 지정된 시간 내에 최상의 솔루션을 찾아내는 것이다.



(그림 6) 유전 알고리즘 크롤러 흐름도
(Figure 6) Flowchart of the genetic algorithm crwaler

그림 6은 Genetic algorithm crawler의 간단한 흐름도이다. 먼저 초기의 웹 페이지를 가져온다. 이 웹 페이지는 어떠한 함수에 의해 적합성 평가를 받고 평가에 따라 선택된 웹 페이지들은 재생성 된 데이터베이스에 추가한다. 이 과정을 반복하여 최적의 솔루션을 찾는다.

거대한 데이터베이스를 검색하는데 소비할 시간이 없거나 적을 때 매우 효율적인 장점이 있어 QoS(Quality of Service)를 예측해 주는데 활용된다. 하지만 자연에서 수행된 유전 과정에서의 교차나 돌연변이와 같은 문제가 발생할 수 있기 때문에 적합성을 평가하는 함수의 설계를 신중히 해야 할 필요가 있다.

3.1.10 Crawling through URL Ordering

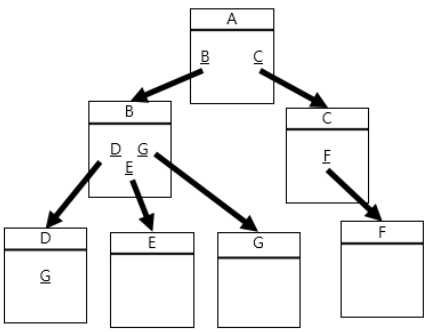
이 알고리즘은 URL이 있는 큐로부터 스캔하여 더 중요한 페이지를 먼저 찾는다. 이전 URL을 방문하여 driving 쿼리와 유사한 앵커 텍스트(link text) 또는 페이지의 링크 거리가 가까운 즉, PR(Page Rank) 값이 큰 것을 더 중요하다고 판단한다[7]. 여기서 말하는 앵커 텍스트(link text)란

하이퍼링크가 걸려져 있는 텍스트에서 강조 표시된 텍스트와 클릭해서 대상 웹 페이지를 열 수 있는 텍스트를 뜻한다.

웹의 일부를 크롤링 하려고 할 때 유용하기 때문에 웹 페이지의 특정 부분을 수집할 때 활용된다. 많은 클러스터가 웹 사이트에 존재하면 성능이 저하되는 단점이 있기 때문에 필터를 사용하여 네트워크 트래픽을 줄이는 장점을 가진 'By the Use of Filter'알고리즘을 접목시킨다면 좀 더 개선된 알고리즘을 기대해 볼 수 있다.

3.1.11 Crawling the Large Sites First

대기 중인 페이지가 많은 웹 페이지를 우선순위로 크롤링한다. 이 알고리즘은 먼저 크롤링 되지 않은 모든 웹 페이지에 점수를 부여하게 된다. 그 다음 우선 순위가 높은 웹 페이지를 찾고, 대기중인 URL이 많은 페이지를 먼저 크롤링하게 된다[7]. 이해를 돕기위하여 아래 그림을 참고한다.



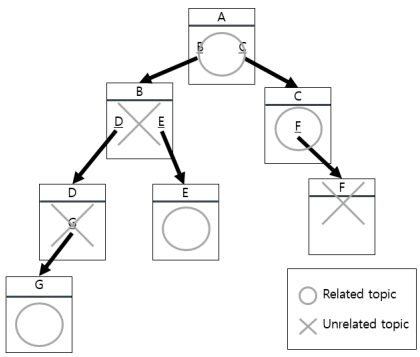
(그림 7) 큰 사이트 우선 크롤링 예제
(Figure 7) Example of Crawing the Large Sites First

그림 7은 Crawing the Large Sites First의 예제를 나타내기 위한 그림이다. 각각의 알파벳이 웹 페이지의 URL, 각 웹 페이지 안에 있는 알파벳들은 하이퍼링크가 걸려 있다고 생각한다. A 페이지에는 2개의 URL이 존재하고 B 페이지에는 3개의 URL, C와 D페이지에는 1개의 URL이 존재한다. 각 페이지에 존재하는 URL들을 대기 중인 상태라고 생각한다. 대기 중인 URL이 많을수록 높은 우선순위를 갖게 되어 B 페이지가 가장 높은 우선순위를 갖게 된다. 그러므로 크롤링이 시작되면 대기 중인 URL이 3개로 가장 많은 B 페이지를 제일 먼저 수집하게 된다. 다음으로 대기 중인 URL이 2개인 A 페이지와 대기 중인 URL이 1개 인 C, D 페이지 순으로 수집하게 된다.

대기 중인 URL이 많은 웹 페이지를 우선순위로 크롤링하여 확률적으로 중요한 정보가 많은 페이지를 먼저 크롤링하게 되어 덩치가 큰 웹 페이지를 방문할 때 활용된다. 하지만 중요한 정보가 있는 웹 페이지가 있다 하더라도 페이지가 적은 사이트에 속해 있다면 나중에 크롤링을 하게 되는 단점이 있다. 기존의 'Page Rank' 알고리즘을 접목시켜 우선순위의 개념을 활용한다면 더 개선된 알고리즘을 기대해 볼 수 있다.

3.1.12 Tunneling

동일한 주제의 페이지가 주제에 맞는 토픽을 가지고 있는 페이지를 직접 가리키지 않을 경우 제한된 수의 주제에 벗어난 페이지를 통과하도록 허용하는 것이 좋다. 베이스라인 크롤러의 근본적인 약점은 토픽 페이지의 경로를 따라 토픽 페이지로 터널링 할 수 없다[6].



(그림 8) 터널링의 표현
(Figure 8) Expression of tunneling

그림 8은 터널링을 표현한 그림이다. 각각의 알파벳이 웹 페이지의 URL, 각 웹 페이지 안에 있는 알파벳들은 하이퍼링크가 걸려 있다고 생각한다. A는 홈 페이지의 root URL을 의미한다. 우리는 시드 페이지인 A와 관련된 토픽이 있는 페이지만을 찾고 싶다. 여기서 관련된 토픽을 가지고 있는 페이지는 동그라미로 표시를 했고, 관련된 토픽을 가지고 있지 않은 페이지는 엑스로 표시를 했다. A와 같이 동그라미 표시가 된 C, E, G만을 저장한다. 그림 8.과 같이 관련된 토픽이 있는 웹 페이지를 찾기 위해서 관련된 토픽이 없는 웹 페이지도 거쳐야 한다. 따라서 제한된 수의 관련 없는 토픽을 가진 웹 페이지를 통과하도록 허용할 필요가 있다.

깊이가 깊어도 관련된 토픽이 있는 페이지들을 크롤링 할 수 있어 'Depth First Search'알고리즘과 'Fish School Search'알고리즘의 장점을 모두 살려 검색 엔진의 연관 웹 페이지 결과를 향상시킬 수 있다. 하지만 베이스라인 크롤러는 터널링을 모델링 할 수 없다는 문제점이 있다. 이러한 문제점을 개선하기 위한 두 가지 주목할 만한 프로젝트인 'Context-graph-based crawler'와 'Cora's focused crawler' 라는 크롤러가 터널링을 구현한다[8].

3.2 기본 웹 크롤링 알고리즘들의 성능 분석

표 2는 기본 웹 크롤링 알고리즘들의 개념, 이점, 한계점을 비교, 분석하기 쉽게 요약한 정보와 시간복잡도를 소개한다.

(표 2) 각 알고리즘 요약 및 시간복잡도
(Table 2) Summary and time-complexity

Algorithm	Concept	Advantage	Limitations	Time Complexity
Breadth First Search	전체 웹을 포괄하기 위해서 같은 레벨의 URL을 먼저 탐색한다.	큰 트리 구조에서 얕은 부분에서 발견되는 중요한 정보가 많을 때 적합하다.	트리 구조에서 가지가 많은 경우 잘 수행되지 않는다.	**O(N)**
Depth First Search	자식 URL을 통과하여 URL을 깊이 먼저 탐색한다.	중요한 정보가 깊이 있는 경우에 적합하다.	트리 구조가 큰 경우 알고리즘이 무한 루프에 빠질 수 있다.	**O(N)**
Fish School Search	지정된 쿼리와 일치하는 내용이 있는 페이지만을 고려한다.	어떤 토픽에 대한 내용을 탐색하고 싶은 경우에 적합하다.	고정된 깊이로만 탐색한다.	**O(N*logN)**
Shark Search	페이지 관련성을 측정하고 지정된 토픽과 관련이 있는 페이지만을 고려한다.	코사인 측정과 함께 TF/IDF 가중 방법을 사용한다.	관련성 판단을 위해 정보 검색에 기반한 기술에 의존한다.	**O(N*logN)**
By HTTP Get Request and Dynamic Web Page	마지막 방문 이후에 업데이트 된 웹 페이지만을 다운로드한다.	최근에 업데이트된 웹 페이지만을 다운로드 한다.	업데이트 된 웹 페이지를 방문하기 전에는 볼 수 없다.	**O(logN)**
By the use of filter	크롤링에 필터를 사용하고 쿼리 기반 접근 방식이다.	웹 크롤링 또는 네트워크 트래픽을 줄인다.	추가 작업을 수행할 수 있도록 추가 필터가 사용된다.	**O(logN)**
Page Rank Algorithm	웹 페이지의 백 링크 수 또는 인용 횟수에 따라 점수를 매겨 우선순위를 정한다.	우선 순위 지정이 안정적이며 효과적인 방법이다.	웹 페이지의 관련성과 동적성에 대해서는 고려하지 않는다.	**O(N+M)** **N: number of nodes** **M: number of arcs/edges**
Online Page Importance Calculation	캐시가 많은 웹 페이지를 다운로드하고 가리키는 페이지 사이에 캐시가 분산된다.	캐시 값이 한 단계로 계산되며 그에 따라 매우 짧은 시간 동안 계산된다.	각 페이지가 여러 번 다운로드 되어 크롤링 시간이 늘어나게 된다.	**O(N*logN)**
Genetic Algorithm	자연 선택과 유전학의 원리에 기반하여 문제를 해결한다.	적자생존의 원칙을 적용하여 더 나은 근사를 산출한다.	자연에서 수행된 유전 과정에서 교차나 돌연변이가 발생할 수 있다.	**O(N*M)** **N: number of population** **M: size of keyword**
Crawling through URL Ordering	URL이 있는 큐로부터 더 중요한 페이지를 먼저 찾는다.	웹의 일부를 크롤링 하려고 할 때 유용하다.	많은 클러스터가 웹 사이트에 존재하면 성능이 저하된다.	**O(logN)**
Crawling the Large Sites First	대기 중인 페이지가 많은 사이트를 먼저 크롤링한다.	큰 웹 사이트가 먼저 크롤링 된다.	중요한 웹 페이지더라도 작은 웹 사이트에 속해 있으면 나중에 크롤링 된다.	**O(logN)**
Tunneling	제한된 수의 나쁜 페이지를 통과하여 좋은 페이지에 도달한다.	깊이가 깊은 좋은 페이지를 크롤링 할 수 있다.	베이스라인 크롤러는 터널링을 모델링 할 수 없다.	**O(N*logN)**

4. 최신 웹 크롤링 알고리즘

크롤링에 대한 중요도와 관심이 높아지고 있다. 이에 따라 기존 크롤링의 문제점을 보완한 다양한 크롤링 알고리즘이 등장했다. 이번 장에서는 3장에서 소개했던 기본 크롤링 알고리즘을 토대로 성능을 개선시킨 최근에 제시된 방안의 크롤링 알고리즘을 소개한다.

4.1 Shark-search Algorithm based on Multi Granularity

이 논문에서는 Shark-search 알고리즘의 노이즈 링크를 개선하기 위하여 새로운 토픽 크롤링 방안을 제안한다. 웹 페이지의 깊이를 분석하고 특정 웹 페이지에 대한 VIPS(Vision-based Page Segmentation) 블록 처리 알고리즘을 채택한다. 관련 링크를 예측할 때 쿼리에 의존하는 HITS(Hyperlink Induced Topic Search) 알고리즘과 결합된 동시에 다중 단위 Shark-search 알고리즘을 채택한다. 이것은 Shark-search 알고리즘의 대역폭 부족 문제를 해결하여 노이즈 링크(noise link)를 줄였다. 콘텐츠기반 Shark-search 알고리즘은 고전적인 토픽 크롤링 알고리즘이다. URL 링크의 주제별 관련성을 계산할 때 알고리즘은 상위 노드 값을 상속 할뿐만 아니라 URL 링크 앵커 텍스트와 텍스트 문서 정보를 최대한 활용한다. 그러나 알고리즘 성능은 우수하지 않습니다. 실험 프로세스에 대한 심층적 인 분석을 통해 웹 페이지의 관련성 정도를 계산할 때 관련 링크에 대한 매우 좋은 예측이 될 수 없다는 것이 주된 이유라는 것을 발견했다. 그리고 페이지가 더 많은 링크를 포함 할 때, 노이즈 링크는 큰 비율을 차지한다. 다중 단위 Shark-search 알고리즘은 VIPS 알고리즘을 사용하여 하위 블록을 수행하고 페이지의 다른 섹션과 블록에서 링크 및 필터를 선택하여 많은 노이즈 링크를 제거한다[14].

4.2 A Cohesive Page Ranking and Depth-First Crawling

이 논문에서는 웹 크롤링의 더 나은 재현율(recall)과 정확도(precision)를 위해 효율적인 속도로 크롤링하는 것을 목적으로 한다. Page Rank와 Depth First Search를 통합하는 방안을 제안한다. 이 방안은 검색 엔진을 향상시킨다. 먼저 가장 왼쪽에서 자식을 찾고 링크 및 웹 페이지에서 크롤링을 시작한다. 가장 왼쪽 자식의 배열은 미리 정의된 주제와 가장 관련이 있는 것으로 결정한다. 가장 왼쪽 자식에 도착하면, 이제 가장 많은 수의 백 링크가 있는 자식을 확인하고 그곳에서부터 크롤링을 시작한다. 더 이상 링크를 사용할 수 없을 때까지 deep web을 사용한다. 그 다음 다시 같은 방식으로 반복한다. 여기서 자체 루프 검사기, 검색 함수 및 콘텐츠 필터가 추가된다. 자체 루프 검사기는 무한 루프에 빠지지 않게 주기를 확인하고 검색 함수는 웹 페이지의 그룹화를 수행하며 컨텐츠 필터는 링크 내용에 엑세스한다[15].

4.3 Less Invasive Crawling Algorithm (LICA)

P2P(Peer to Peer) 봇넷(Botnets)에서 알려진 기존 크롤링 접근 방식은 BFS(Breadth First Search) 및 DFS(Depth First Search)이다. 그러나 이러한 접근 방식은 봇넷에 크롤러가 쉽게 감지되어 봇넷 크롤링이 더욱 어려워진다. 이러한 문제점을 해결하기 위해서는 크롤링 속도가 빠르며 감지되지 않은 상태로 유지하기 위해 네트워크 활동을 최소화하여야한다. 봇넷을 크롤링 하는 기존 방안은 이웃 목록을 요청하는 것이지만 이웃 목록을 자주 요청하게 되면 봇넷에게 의심을 제기할 뿐만 아니라 크롤링 속도가 빠르지 않아 걸리게 된다. 본 논문에서는 네트워크 활동을 최소화하기 위해 덜 침략적인 LICA(Less Invasive Crawling Algorithm)을 제안한다. 크롤링 효율성을 목표로 한다. 매개변수를 통해 특정 봇넷에 연결할 수 있다. 매개변수는 특정 전체 크롤링 내에서 네트워크의 노드로 전송이 허용되는 최대 요청 수, 즉 후속 크롤링 반복 횟수이다. 윈도우 매개변수는 후속 요청들의 수를 결정한다. 윈도우 매개변수의 요청으로 나눈 이득은 크롤링 중 학습 곡선을 제공하여 임계 값 t가 0 이하로 떨어질 때 알고리즘은 종료된다. 전체 크롤링은 네트워크에 있는 모든 접촉 가능한 노드가 발견되면 종료된다[16].

4.4 An enhanced Breadth First Search

이 논문에서는 Peer to Peer(P2P) 콘텐츠 공유 네트워크(CSNs; Content Sharing Networks)에서 가장 큰 이슈인 트래픽 량과 공유 플랫폼의 측정과 분석에 대하여 제안한다. 대부분의 측정도구들은 P2P CSNs의 현재 상황에서 업데이트되지 않았다. 이 논문에서는 다재다능하고 순환적인 틀을 가진 LCrawler라는 측정도구를 제안한다. 이 LCrawler라는 크롤러에서 사용되는 크롤링 알고리즘은 기존의 Breadth First Search(BFS)를 향상시켜 사용한다. BFS

를 사용할 때 DFS보다 시작 시간은 빠르게 걸리지만 반복적인 노드들의 문제로 전체 프로세스 효율이 떨어진다. 이 부분을 해결하고자 enhanced BFS 알고리즘을 제안한다. 이 알고리즘은 다중 쓰레드(multi-thread)를 기반으로 동작되며 크롤링 하는 시간에 제한을 둔다. Send_thread, Receive_thread, Analyze_thread and Write_thread 총 4개의 쓰레드로 구성된다. 크롤링의 시간과 각 쓰레드의 비율의 임계값은 대상 노드 선택 방법을 제어한다. Send_thread에서는 항상 높은 크롤링 속도를 보장한다. 반복되는 노드 간가 폐기되고 노드 간의 연결이 Receive_thread안의 집합에 삽입된다. Analze_thread와 Write_thread는 검색한 노드에 초당 100 패킷의 속도의 무작위 요청을 보내 하나의 쿼리 함수를 도입하기 위해 약간의 최적화 기능을 가지고 있는데 이것은 메인 크롤링에 방해되지 않고 보다 중요한 인접 노드를 획득하는 것을 목표로 한다[17].

4.5 N-gram Based BFS, MWE BFS

이 논문은 크롤링 전력과 수집된 비교될만 한 말뭉치를 기반으로 statistical machine translation(SMT) 시스템에서 얻은 결과적인 변역간의 관계를 평가한다. 여러 URL에서 얻은 각 말뭉치를 사용하여 SMT 시스템의 결과로 번역에 대한 점수를 계산한다. 이 실험의 결과 URL의 질이 번역결과의 질에 큰 영향을 미친다는 것을 보여준다. 쉽게 확장가능하고 사용자 정의가 가능한 크롤러를 위해 이 논문에서 제안된 알고리즘 중 N-gram Based BFS(NBFS)와 Multiword Expression Factor BFS(MWE BFS)를 소개한다. NBFS는 BFS와 비슷하다. 주요한 차이점은 텍스트 단위가 한 단어에서 n-그램으로 바뀐다는 것이다. 따라서 관련성의 정의는 단어 대신 n-그램으로 고려된다. 코사인 유사성을 위한 벡터의 구성요소도 n-그램의 표현으로 바뀐다. 벡터 구성요소의 값은 n-그램이 문서에서 IDF와 발견될 수 있는 횟수를 곱한 것이다. MWE BFS는 NBFS를 기반으로한다. 다중 단어 표현 인자에 대한 약어인 MWE 인자를 추가하기 위해 원래의 TF/IDF모델을 채택한다. MWE 인자는 벡터 구성요소가 n-그램에 대한 주파수 정보를 포함하는 경우에만 적용할 수 있는데, 이는 벡터 구성요소가 n-그램 구성요소 단어 사이의 연결 강도에 기초하기 때문이다[18].

4.6 Modified version of Depth First Search Algorithm

이본 논문에서 크롤링 알고리즘은 Depth First Search Algorithm을 수정하여 사용한다. 먼저 검색을 시작해야 하는 위치에서 사용자가 루트 링크나 URL로 제공한다. 크롤러에 지정된 링크는 루트 역할을 하므로 검색의 노드로 자동 할당된다. 크롤러는 또한 특정 연구 목적으로 탐색을 좁힐 경우 크롤러가 접근할 수 있는 최대 링크 깊이를 지정할 수 있는 기회를 제공한다. 이는 분석기에서 지정한 노드가 필요에 따라 목표 상태와 동일한지 확인하는 것이다. 이 비교는 루프 종료의 조건으로 작용한다. 노드 링크가 목표 상태와 동일한 것으로 확인되면 크롤러는 여기에서 도달할 수 있는 다른 가능한 방향을 찾는다. 다른 링크 경로가 없는 경우 알고리즘이 해독되어, 지정된 조건 내에서 필요한 결과를 분석기로 나타낸다. 링크 깊이를 지정하는 경우 크롤러는 다른 비교 조건을 통해 현재 노드가 지정된 링크 깊이의 노드에 해당하는지를 점검한다. 이는 크롤러가 지정된 조건 내에 머물며 지정된 경계 밖으로 이동하지 않도록 하기 위한 것이다. 전술한 조건이 만족되지 않을 때에만 크롤링에 의해 메커니즘이 형성된다. 즉, 루트 노드는 목표 상태와 동일하지 않아야 하며, 만일 그렇다면 루트 노드에서 어떤 미개척 방향을 가져야 한다. 또한 현재 노드는 지정된 링크 깊이를 초과하면 안된다. 현재 노드가 지정된 링크 깊이의 상태에 이르면 크롤러는 경로를 역추적(Back-Track)할 수 있는 기능으로 개발된다. 이것은 크롤러에게 지정된 경계 내에서 가능한 모든 하이퍼링크를 통과할 수 있는 기회를 제공한다. 역추적은 스택의 도움으로 이루어진다. 노드가 루트에서 목표 상태로 이동함에 따라 하이퍼링크의 확장이 스택에 저장된다. 지정된 경계에 도달할 때 크롤링 알고리즘을 스택의 모든 노드에 대해 재귀적으로 부른다. 이 프로세스는 크롤러가 도달한 모든 링크에 대해 반복되므로 지정된 대로 하이퍼링크를 추적하고 액세스할 수 있다[19].

4.7 Event Focused Crawling

이슈가 되는 이벤트에 대한 웹 데이터를 수집하기 위한 통합 이벤트 집중(focused) 크롤링 시스템이 필요하다. 재난이나 다른 중요한 사건이 발생할 때, 많은 사용자들은 그 사건에 대한 최신 정보를 찾으려고 노력한다. 그러나 이벤트 정보의 체계적 수집과 보관은 거의 없다. 이 논문에서는 자동 이벤트 추적과 보관을 위한 지능형 이벤트 집중 크롤링은 궁극적으로 효과적인 접근으로 이어질 것을 제안한다. 이 논문에서는 주요 이벤트 정보를 캡처할 수 있는 이벤트 모델을 개발했고, 그 모델을 집중 크롤링 알고리즘에 통합했다. 집중된 크롤러가 웹 페이지 관련성

을 예측하는 데 이벤트 모델을 활용하기 위해 두 사건 표현 간의 유사성을 측정하는 기능을 개발했다. 그 후 캘리포니아 총격과 브뤼셀 공격이라는 두 가지 최근의 사건에 대한 시스템을 평가하기 위해 두 번의 실험을 실시했다. 첫 번째 실험 시리즈는 웹 페이지의 관련성을 평가할 때 제안한 이벤트 모델 표현의 효과를 평가했다. 이벤트 모델 기반 표현은 기준 방법을 능가했다. 그것은 F1 점수에서 20%의 향상으로 정밀도, 회상, F1 점수에서 더 나은 결과를 보여주었다. 두 번째 실험 시리즈는 WWW에서 관련 웹페이지를 수집하기 위한 이벤트 모델 기반 집중 크롤러의 효과를 평가했다. 이벤트 모델 기반 집중 크롤러는 최첨단 베이스라인 집중 크롤러(최우선)를 능가했다. 평균 40%의 개선으로 수확비율(harvest ratio)을 개선했다[20].

4.8 선제적 크롤링 알고리즘: 감성 반응 웹 크롤링 알고리즘

기존에는 일반적인 알고리즘의 성능을 개선하기 위한 알고리즘들이 개발되어왔다. 성능이 개선된 알고리즘의 등장과 더불어 최근에는 특수 목적을 가진 웹 크롤링 알고리즘들이 등장하기 시작했다. 이번 장에서는 선제적 웹 크롤링 알고리즘인 감성 반응 웹 크롤링(Sentiment-aware Web Crawler) 기법을 소개한다[21]. 기존 감성사전 구축은 감성 신조어에 대한 정보가 없는 문제점이 있다. 이와 같은 문제를 해결하기 위해 감성어가 풍부한 텍스트를 주기적으로 크롤링하여 감성 신조어를 추가하는 작업이 필요하다. 텍스트 문서 중 감성을 나타내는 긍/부정 어휘가 많은 텍스트 문서만을 수집하고자 할 때 기존 방안은 일단 모든 텍스트를 DB에 저장하고 저장된 모든 텍스트 문서를 스캔하여 필터링 하는데 이러한 작업은 시간이 오래 걸리게 된다. 이러한 방안은 물리적인 시간 낭비와 저장소 낭비가 발생할 수 있다. 모든 텍스트를 스캔하고 필터링하는 시간을 최소화하기 위하여 처리 속도가 빠르고 정확도가 높은 블룸 필터를 적용시킨다. 크롤링을 하는 동시에 긍/부정 어휘가 많은 텍스트 문서만을 필터링하여 수집하기 때문에 시간과 저장소의 절약을 기대할 수 있다.

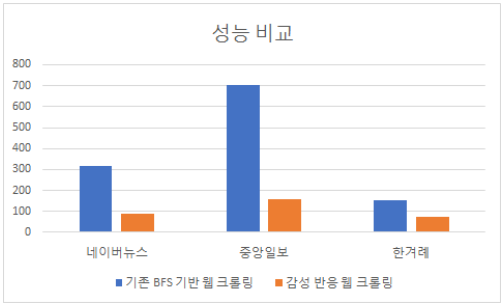
4.8.1 감성 반응 웹 크롤링 알고리즘

웹 문서를 크롤링하는 동안 긍/부정 어휘가 풍부한 텍스트 문서들을 보다 빠르고 정확하게 수집하기 위해 블룸 필터라는 자료 구조를 이용하여 감성 반응 웹 크롤링 알고리즘을 제안한다. 기본 크롤링 알고리즘은 BFS(Breadth First Search)알고리즘을 사용한다. 그림 9.를 보면 URL을 수집한 후 본문 텍스트 중 원하는 내용을 파싱한다. 파싱한 텍스트 문서를 전처리 과정을 진행한다. 전처리 과정은 불용어 처리와 어근을 추출한다. 기준이 되는 감성어는 블룸 필터라는 자료구조에 삽입한다. 블룸 필터란 m개의 비트 배열(bit array)과 k개의 해시 함수(hash function)를 이용해서 검사하고자 하는 원소가 이미 검사 되어 있는지를 판단한다. 이 때 사용되는 해시 함수란 데이터의 효율적 관리를 목적으로 임의의 길이의 데이터를 고정된 길이의 데이터로 매핑(mapping)하는 함수이다. 이러한 블룸 필터는 긍/부정 어휘를 식별해 주는 역할을 한다. 전처리 된 데이터에서 긍/부정 어휘가 많다고 판별되면 저장하고 그렇지 않으면 저장하지 않는다.

4.8.2 실험 결과

[21]에 따르면 3개의 SEED_URL(네이버뉴스, 중앙일보, 한겨레 신문)을 넣은 후 각 100개의 URL만 방문하게 제한하여 진행한다. URL 마다 본문 텍스트가 있을 수도 없을 수도 있기 때문에 총 본문 텍스트의 수는 다르다.

(시간 단위: Millisecond; ms)



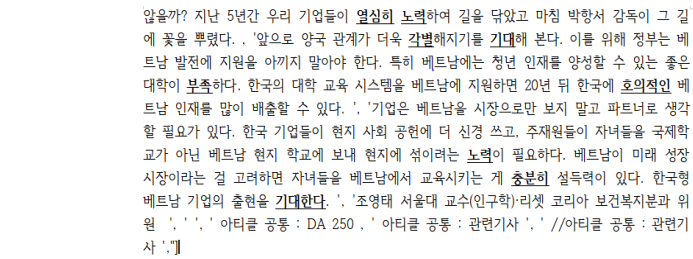
(그림 9) 실행 시간 비교

(Figure 9) Comparison of Execution times

그림 9는 긍/부정 어휘가 많은 텍스트를 수집할 때 기존방안에서 모든 문서를 스캔하는 시간과 블룸필터를 사용하여 필터링 하는 시간을 비교한 그래프이다. 기존방안으로 수집된 문서의 용량은 네이버뉴스 930KB, 중앙일보 1216KB, 한겨레 신문 432KB이고 저장된 모든 문서를 스캔하는 시간은 네이버뉴스는 317ms, 중앙일보는 703ms, 한겨레 신문은 152ms이다.

그에 반해 블룸필터를 사용했을 때 필터링 하여 수집된 문서의 용량은 네이버뉴스 249KB, 중앙일보 316KB, 한겨레 신문 138KB 이고 필터링 되는 시간은 네이버뉴

스는 90ms, 중앙일보는 104ms, 한겨레 신문은 73ms 이다. 실험결과 1875KB의 용량을 절약할 수 있었고 평균 302ms의 시간을 절약하는 것을 확인할 수 있다.


앞을까? 지난 5년간 우리 기업들이 열심히 노력하여 길을 닦았고 마침 박항서 감독이 그 길에 꽃을 뿌렸다. , '앞으로 양국 관계가 더욱 각별해지기를 기대해 본다. 이를 위해 정부는 베트남 발전에 지원을 아끼지 말아야 한다. 특히 베트남에는 청년 인재를 양성할 수 있는 좋은 대학이 부족하다. 한국의 대학 교육 시스템을 베트남에 지원하면 20년 뒤 한국에 호의적인 베트남 인재를 많이 배출할 수 있다. ', '기업은 베트남을 시장으로만 보지 말고 파트너로 생각할 필요가 있다. 한국 기업들이 현지 사회 공헌에 더 신경 쓰고, 주재원들이 자녀들을 국제학교가 아닌 베트남 현지 학교에 보내 현지에 섞이려는 노력이 필요하다. 베트남이 미래 성장 시장이라는 걸 고려하면 자녀들을 베트남에서 교육시키는 게 충분히 설득력이 있다. 한국형 베트남 기업의 출현을 기대한다. ', '조영태 서울대 교수(인구학)·리셋 코리아 보건복지분과 위원 ', ' ', ' 아티클 공통 : DA 250 , ' 아티클 공통 : 관련기사 ', ' //아티클 공통 : 관련기사 ',"]

(그림 10) 감성어가 많이 포함된 문서 중 일부분

(Figure 10) Part of documents that contains a lot of emotional words

그림 10은 '감성반응 웹 크롤러'로 감성을 나타내는 긍/부정 어휘가 많다고 판단되는 텍스트 문서를 저장하고 그 중 일부분을 캡처한 화면이다. 캡처한 부분에서 '열심히', '노력' 두 번, '각별', '기대' 두 번, '부족하다', '호의적인', '충분히' 등의 많은 감성어가 포함되어 있는 것을 확인할 수 있다.

5. 크롤링 관련 툴

이번 장에서는 실제로 크롤링을 할 때 도움이 될 만한 크롤링 관련 툴에 대한 정보들을 소개한다.

5.1 Selenium

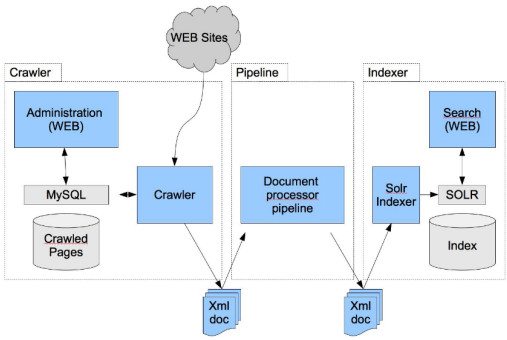
크롤링을 하는 다양한 방법 중 실제 브라우저를 다룰 수 있는 도구를 사용할 수 있다. 이러한 도구를 사용하면 브라우저 조작을 프로그래밍에서 할 수 있다. 다시 말해 버튼이나 링크를 클릭하는 이벤트 처리 상황을 모두 프로그래밍으로 조작할 수 있어 사용자가 실제 브라우저를 사용하는 상태를 재현할 수 있다. 하지만 실제로 브라우저를 실행하므로 자원을 많이 사용해야하고 조작 시점을 정확하게 맞출 수 있게 정밀한 프로그래밍이 필요하며 동작이 불안정하다는 단점도 존재한다. 이러한 단점들 때문에 대규모로 크롤링할 때에는 사용하기 어렵다. 그렇기 때문에 "Ajax를 활용한 웹 응용프로그램"이나 "SPA로 구축된 웹 사이트"같은 것들을 크롤링하는 경우에나 최후의 수단으로 사용한다. Selenium은 원래 웹 응용프로그램 테스트 도구이다. 이 안에 WebDriver라는 실제 브라우저의 조작을 자동화하기 위한 라이브러리가 있다. 이를 활용하면 웹 브라우저를 실행하고 이를 코드로 조작해서 자바스크립트를 활용한 웹 응용프로그램을 테스트할 수 있다. WebDriver를 사용하여 크롤러를 만들게되면 웹 브라우저를 다룰 수 있으므로 자바스크립트로 동적 생성되는 HTML같은 경우도 추출해 낼 수 있다. WebDriver는 파이어폭스, 크롬, 인터넷 익스플로러, 엣지 등의 다양한 브라우저를 지원하는 드라이버를 제공한다. 이러한 드라이버를 사용하면 실제 웹 브라우저를 실행한 뒤 자바 프로그램으로도 원격 조작할 수 있다[24].

5.2 PhantomJS

PhantomJS란 헤드리스 브라우저이다. 헤드리스 브라우저란 헤드리스 모드(브라우저 화면을 출력하지 않은 모드)로 웹 페이지를 읽어 들이는 브라우저를 의미한다. 따라서 자바스크립트 같은 경우도 일반적인 브라우저에서처럼 동작한다. 화면이 출력되지 않고 백그라운드에서 동작하므로 프로그램 조작으로 자동화를 구현해 낼 수 있는 장점이 있다. PhantomJS는 헤드리스 모드 전용 브라우저이지만 이 외에 크롬같은 일반적인 브라우저가 헤드리스 모드를 지원하는 경우도 있다. PhantomJS를 다운받을 때는 자신이 사용하는 PC의 OS에 맞게 바이너리 파일을 다운로드하고 적절한 경로 설정을 해주어 압축을 해제하면 된다. 파이어폭스 또는 크롬 등의 다른 웹 브라우저 전용 드라이버를 사용하는 경우에도 마찬가지로 각 드라이버에 대응하는 바이너리 파일을 따로 다운로드하고 적절한 경로를 설정해준다. 이러한 드라이버를 사용 시 실제 웹 브라우저에 화면이 출력되는 상태로 조작할 수 있다. 현재 까지 헤드리스 모드 전용 브라우저로써는 PhantomJS가 많이 사용되었지만, 크롬이 헤드리스 모드를 지원하게 되면서 2017년 4월에 PhantomJS를 만든 슬로보딘(Slobodin)이 "PhatomJS의 유지 및 보수를 중단한다"고 발표하였다. 더불어 그는 "크롬이 PhantomJS보다 빠르고 안정적이니 전환하기 바란다"는 말까지 하였다. 이후에는 PhantomJS보다 크롬이 더 많이 사용될 것으로 보인다[24].

5.3 Crawl Anywhere

Crawl Anywhere는 강력한 웹 사용자 인터페이스가 있는 웹 크롤러이다. 문서를 처리할 파이프라인과 Solr 인덱서, 모든 기능을 갖춘 맞춤형 검색 응용프로그램이다.



(그림 11) 크롤 애니웨어의 구성 요소
(Figure 11) Components of the Crawl Anywhere

그림 11은 Crawl Anywhere의 구성 요소이다. 크게 크롤러와 파이프라인 인덱서로 이루어져 있다. 크롤러부분에서 문서를 가져오면 파이프라인부분에서 문서를 처리하게 되고 그 처리된 문서로 인덱서부분에서 검색 응용프로그램을 활용할 수 있다[25].

5.4 Import.io

Import.io는 코드를 작성하지 않고 웹 사이트에서 데이터를 추출하기 위한 웹 기반 플랫폼이다. 이 도구는 사람들이 비정형 웹 데이터를 기계 학습, 인공 지능, 소매 가격 모니터링 및 학술 연구에 사용할 수 있는 구조화된 형식으로 변환할 수 있도록 해준다. 사용자는 URL을 입력하고 앱은 자동으로 필요한 데이터를 추출하려고 시도하며 자동으로 추출이 필요한 것을 제공하지 않는 경우 포인트와 클릭 인터페이스를 통해 추출할 수 있다. 수집된 데이터는 Import.io's 클라우드 서버에 저장되며 CSV, Excel, JSON 이나 API를 통해 다운도르 할 수 있다. 사용자는 자신의 응용 프로그램 또는 타사의 분석 및 시각화 소프트웨어에 라이브 웹 데이터를 쉽게 연동하여 수천 개의 데이터 소스를 동시에 추출할 수 있다[26].

5.5 Python Scrapy

최근 웹 페이지가 수 억개가 넘어가고 있다. 그러한 웹 페이지들은 수많은 정보를 가지고 있다. 페이지들의 정보를 모아 유의미한 정보를 도출하기 위하여 페이지를 수집하는 여러 가지 방법들이 논의되고 있는데 이를 Scraping 혹은 Crawling 이라고 한다. Scrapy는 Scraping을 돕기 위하여 파이썬에서 제공해주는 라이브러리이다. Scrapy를 이용하면 원하는 페이지로 접속하여 원하는 형태로 데이터를 가공해서 데이터를 저장할 수 있다. scrapy 명령어를 통해 프로젝트를 생성한다. 프로젝트 명으로 디렉토리가 생성되며 해당 디렉토리 안에 scrapy 기본 프로젝트가 실행된다. 프로젝트의 구조는 아래와 같다[27].

- project_name/scrapy.cfg: 프로젝트 파일구조 설정
- project_name/items.py: 크롤링한 결과가 저장될 형태 정의
- project_name/pipelines.py: 크롤링한 데이터를 가공 및 처리
- project_name/settings.py: 프로젝트 설정 파일
- project_name/spyders/: 실제 크롤링시 동작할 파일

5.6 OAuth

OAuth는 인터넷 사용자들이 비밀번호를 제공하지 않고 다른 웹사이트 상의 자신들의 정보에 대해 웹사이트나 애플리케이션의 접근 권한을 부여할 수 있는 공통적인 수단으로서 사용되는, 접근 위임을 위한 개방형 표준이다. 이 매커니즘은 여러 기업들에 의해 사용되는데, 이를테면 아마존, 구글, 페이스북, 마이크로소프트, 트위터가 있으며 사용자들이 타사 애플리케이션이나 웹사이트의 계정에 관한 정보를 공유할 수 있게 허용한다. OAuth가 사용되기 전에는 인증방식의 표준이 없었기 때문에 기존의 기본인증인 아이디와 비밀번호를 사용하였는데, 이는 보안상 취약한 구조이다. 기본인증이 아닐 경우는 각 애플리케이션들이 각자의 개발한 회사의 방법대로 사용자를 확인하였다. 예를 들면 구글의 AuthSub, AOL의 OpenAuth, 야후의 BBAuth, 아마존의 웹서비스 API 등이 있다. OAuth는 이렇게 제각각인 인증방식을 표준화한 인증방식이다. OAuth를 이용하면 이 인증을 공유하는 애플리케이션끼리는 별도의 인증이 필요없다. 따라서 여러 애플리케이션을 통합하여 사용하는 것이 가능하게 된다. OAuth인증은 소비자와 서비스 제공자 사이에서 일어나는데 이 인증 과정은 다음과 같다. 1.소비자가 서비스제공자에게 요청토큰을 요청한다. 2.서비스제공자가 소비자에게 요청토큰을 발급해준다. 3.소비자가 사용자를 서비스제공자로 이동시킨다. 여기서 사용자 인증이 수행된다. 4.서비스제공자가 사용자를 소비자로 이동시킨다. 5.소비자가 접근토큰을 요청한다. 6.서비스제공자가 접근토큰을 발급한다. 7.발급된 접근토큰을 이용하여 소비자에서 사용자 정보에 접근한다[28].

5.7 Win Web Crawler

Win Web Crawler는 검색 엔진이다. 가격은 $99.00이며

시스템 사용 조건은 Windows(95/98/2000/NT/ME/XP/Vista)와 32MB RAM, 1MB 하드디스크 공간, 인터넷 연결과 같은 조건을 충족하면 사용 가능하다. 데이터를 디스크 파일에 직접 저장하는 프로그램에는 URL 필터, 텍스트 필터, 데이터 필터, 도메인 필터, 수정 날짜 등 세션을 제한하는 수많은 필터가 존재한다. 사용자가 선택할 수 있는 시간 제한, 검색 스레드 등 많은 옵션을 사용할 수 있다. 인기 있는 모든 검색 엔진을 질의하고, 검색 결과에서 일치하는 모든 URL을 추출하여, 중복 URL을 제거한다음, 최종적으로 해당 웹사이트를 방문하여 데이터를 추출할 수 있다[29].

5.8 Elastic Search

Elastic Search(일래스틱서치)는 루씬 기반의 검색 엔진이다. HTTP 웹 인터페이스와 스키마에서 자유로운 JSON 문서와 함께 분산 멀티테넌트 지원 전문 검색 엔진을 제공한다. 일래스틱서치는 자바로 개발되어 있으며 아파치 라이선스 조항에 의거하여 오픈 소스로 출시되어 있다. 공식 클라이언트들은 자바, 닷넷(C#), PHP, 파이썬, 그루비 등 수많은 언어로 이용이 가능하다. 일래스틱서치는 가장 대중적인 엔터프라이즈 검색 엔진으로 그 뒤를 루씬 기반의 Apache Solr가 잇는다. 일래스틱서치는 로그스태시(Logstash)라는 이름의 데이터 수집 및 로그 파싱 엔진, 그리고 키바나(Kibana)라는 이름의 분석 및 시각화 플랫폼과 함께 개발되어 있다. 이 3개의 제품들은 연동 솔루션으로 사용할 목적으로 설계되어 있으며 이를 "일래스틱스택"(Elastic Stack, 과거 이름: ELK 스택)으로 부른다. 일래스틱서치는 모든 종류의 문서를 검색하는데 사용할 수 있다. 가변 검색 및 실시간에 가까운 검색을 제공하며 멀티테넌시를 지원한다. 일래스틱서치는 분산 방식이므로 인덱스를 여러 샤드로 나눌 수 있으며 각 샤드는 0개 이상의 복제물(replica)을 가지고 있을 수 있다. 각 노드는 하나 이상의 샤드를 관리하며 작업을 올바른 샤드로 할당시켜 주는 조율자 역할을 한다. 리밸런싱 및 라우팅은 자동으로 수행된다. 연관 데이터는 종종 동일한 인덱스에 저장되며 이는 하나 이상의 프라이머리 샤드와 0개 이상의 복제물(replica) 샤드로 이루어진다. 인덱스가 만들어지면 프라이머리 샤드의 수는 변경할 수 없다. 일래스틱서치는 루씬을 사용하며 JSON과 자바 API를 통해 모든 기능을 최대한 활용한다. 다면(facetting) 및 침투(precolating)을 지원하므로 새로운 문서들이 등록된 쿼리와 일치할 경우 통보하는데 유용할 수 있다. 그 밖의 기능으로 "게이트웨이"이 있으며 장기간의 인덱스 지속성을 관리한다. 이를테면 인덱스는 서버 충돌 시에 게이트웨이로부터 복구할 수 있다. 일래스틱서치는 실시간 GET 요청을 지원하므로 NoSQL 데이터스토어의 역할에 적합하지만 분산 트랜잭션 면에서는 부족하다[30].

6. 결 론

본 논문에서는 크롤링에 대한 기본사항과 기존의 기본적인 웹 크롤링 알고리즘을 소개한다. 더불어 기본 웹 크롤링 알고리즘의 단점을 개선한 최신 크롤링 알고리즘과 특수한 목적을 가진 선제적 크롤링 알고리즘인 감성반응 웹 크롤링 알고리즘을 소개한다. 또한 실제 크롤링에 도움이 될 만한 관련 툴에 대하여 소개한다. 데이터의 대다수는 웹 페이지 상에 존재하지만 네트워크 대역폭, 시간적인 문제, 하드웨어적인 저장소 등의 제약으로 인해 모든 페이지를 다운로드 할 수 없다. 나쁜 정보를 가진 데이터는 거르고 좋은 정보를 가진 데이터만을 수집하는 효율적인 데이터 수집이 필요하다. 효율적인 데이터 수집을 위해서 상황에 맞게 웹 크롤링 알고리즘을 사용해야한다. 웹 크롤링 알고리즘을 비교하고 분석하며 이해를 돕고 크롤링에 대한 심도있는 정보를 주는 것이 본 논문의 목적이다. 여러 웹 크롤링 알고리즘을 이해하면 어떠한 상황에서 어떤 알고리즘이 유용하게 활용 가능한지 알 수 있다. 나아가 앞으로 크롤링에 대한 연구 방향은 이 논문의 4장에서 소개하는 선제적 크롤링 알고리즘인 '감성 반응 웹 크롤링 알고리즘'처럼 특수한 목적에 필요한 정보만을 수집하는 알고리즘을 사용하여 좀 더 효율적인 데이터 수집을 할 수 있도록 진행되어야 한다. 이 논문을 바탕으로 웹 크롤링에 대한 정보를 심도있게 알아보고 기존 웹 크롤링 알고리즘에 대한 이해도를 높여 향후 더 개선된 웹 크롤링 알고리즘을 개발할 수 있기를 바란다.

Acknowledgement

이 논문은 2016년도 정부(미래창조과학부)의 한국연구재단의 중견연구자지원사업(No. NRF-2016R1A2B1014843)의 연구비 지원으로 수행하였습니다. 이 논문은 2017년 정부(과학기술정보통신부)의 재원으로 한국 연구재단의 지원을 받아 수행된 연구임(NRF-2017M3C4A7068188)

참고문헌(Reference)

[1] K. Bharat and A. Z. Broder, “A technique for measuring the relative size and overlap of public web search engines.” In Proceedings of the 7th World Wide Web Conference, vol. 30, pages 379-388, 1998. https://doi.org/10.1016/s0169-7552(98)00127-5

[2] S. Lawrence and C. L. Giles, “Searching the World Wide Web.” Science, 280(5360):98-100, 1998. https://science.sciencemag.org/content/280/5360/98

[3] C. Castillo, M. Marin, A. Rodriguez, and R. Baeza-Yates, “Scheduling algorithms for Web crawling.” In Latin American Web Conference (WebMedia/LA-WEB), Riberao Preto, Brazil, IEEE Cs. Press. 2004. https://doi.org/10.1109/webmed.2004.1348139

[4] C. Olston and M. Najork, ″Web Crawling,″ Foundations and Trends in Information Retrieval. Vol. 4, No. 3, 2010. https://www.nowpublishers.com/article/Details/INR-017

[5] R. kumar, A. Jain and C. Agrawal, ″Survey of Web Crawling Algorithms,″ Advances in Vision Computing: An International Journal (AVC), vol. 1, no. 2/3, Sep. 2014. https://doi.org/10.5121/avc.2016.3301

[6] C. Singh1, K. Singh2 and hansraj, ″A Survey on Web Crawling Algorithms Strategies″ International Journal of Research in Advent Technology, Sep. 2014.

[7] R. kumar, A. Jain and C. Agrawal, “Survey of Web Crawling Algorithms,” Advances in Vision Computing: An International Journal (AVC), vol. 3, no. 3, Sep. 2016. https://doi.org/10.5121/avc.2016.3301

[8] G. Parul, Arora, Vinay, “Similarity analysis of web crawled data”, THAPAR UNIVERS ITY, Master of Engineering in Computer Science and Engineering, Jul. 2015.

[9] J. Cho and U. Schonfeld, ″RankMass Crawler: A Crawler with High Personalized PageRank Coverage Guarantee ″ Very Large Data Base Endowment, Sep. 2007. https://dl.acm.org/citation.cfm?id=1325897

[10] M. B. Sahu and S. Bharne, ″A Survey on Various Kinds of Web Crawlers and Intelligent Crawler″ International Journal of Scientific Engineering and Applied Science (IJSEAS), vol. 2, no, 3, March. 2016.

[11] J. Cho, H. Garcia-Molina, L. Page, “Efficient Crawling Through URL Ordering”, Computer Networks and ISDN System, Vol 30, Apr, 1998 https://doi.org/10.1016/s0169-7552(98)00108-1

[12] Pavalam S M, S V Kashmir Raja, Felix K Akorli and Jawahar M, “A Survey of Web Crawler Algorithms,” International Journal of Computer Science Issues, vol. 8, issue. 6, no. 1, Nov. 2011.

[13] B. Novak, “A Survey of Focused Web Crawling Algorithms,” ailab.ijs.si, 2004. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.102.1837&rep=rep1&type=pdf.

[14] N. Liu, R. Yao, “The Crawling Strategy of Shark-search Algorithm based on Multi Grannularity”, 2015 8th International Symposium on Computational Intelligence and Design (ISCID). May. 2016. https://doi.org/10.1109/iscid.2015.273

[15] A. Kenechukwu, O. Samuel, A.Gloria, “A Cohesive Page Ranking and Depth-First Crawling Scheme For Improved Search Results”, West African Journal of Industrial & academic research, Vol.18, No.1, June. 2017. http://wajiaredu.com.ng/wp-content/uploads/2018/09/West-African-Journal-of-IndustrialVol18Tempzzz-1.pdf#page=35

[16] S. Karuppayah, M. Fischer, C. Rossow and M. Muhlhauser, “On Advanced Monitoring in Resilient and Unstructured P2P Botnets”, 2014 IEEE International Conference on Communications (ICC), Aug. 2014. https://doi.org/10.1109/icc.2014.6883429

[17] Q. Lu, B. Liu, H. Hu, “LCrawler: an Enhanced Measurement Tool for Peer-to-Peer Content Sharing Networks”, 2016 5th International Conference on Computer Science and Network Technology (ICCSNT), pp. 509-510, 2016. https://doi.org/10.1109/iccsnt.2016.8070210

[18] B. R. Laranjeira, V. P. Moreira, A. Villavicen cio, C. Ramisch, M. J. Finatto, “Comparing the Quality of Focuesd Crawlers and of the Translation Resources Obtained form them”, Language Resources and

Evaluation Conference (LREC), pp. 3574, 2014. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.680.7335&rep=rep1&type=pdf

[19] R. Suganya Devi, D. Manjula, and R. K. Siddharth, “An Efficient Approach for Web Indexing of Big Data through Hyperlinks in Web Crawling”, The Scientific World Journal, pp. 9-17, Feb. 2015. https://doi.org/10.1155/2015/739286

[20] Mohamed M. G. Frag, S. Lee, Edward A. Fox, “Focused crawler for events”, International Journal on Digital Libraries, vol. 19, pp. 3-9, Mar. 2018. https://doi.org/10.1007/s00799-016-0207-1

[21] C. Na, B. On, “A Bloom filter-based Sentiment-aware Web Crawling Algorithm”, The 30th Hangeul and Korean Information Processing Conference, Oct. 2018.

[22] S. Park, C. Na, M. Choi, D. Lee, B. On, “A proposal on the Establishment of Korean Emotional Dictionnary based on Bi-LSTM”, Journal of Inteligent Information System, 2018.

[23] Bloom filter, https://en.wikipedia.org/wiki/ Bloom_filter, (Accessed, 2018).

[24] Takejioe Naoki, Shinmamoto Takako, Tadokoro Shunsuke and 2 others, “Crawling Essentials Guide: Practical techniques for crawling websites”, p263-p265, wikibooks, 2018.

[25] Crawl Anywhere, “http://www.crawl-anyw here.com/”, (Accessed. 2018)

[26] Import.io, “https://en.wikipedia.org/wiki/-Import.io”, (Accessed. 2018).

[27] Python Scrapy, “http://www.incodom.kr/% ED%8C%8C%EC%9D%B4%EC%8D%AC/%EB%9D%BC%EC%9D%B4%EB%B8%8C%EB%9F%AC%EB%A6%AC/Scrapy”,(Accessed. 2018).

[28] OAuth, “https://ko.wikipedia.org/wiki/OAuth”, (Accessed. 2018)

[29] Win Web Crawler, “http://www.winwebcra wler.com/ index.htm”, (Accessed. 2018).

[30] Elastic Search, “https://en.wikipedia.org/wiki /Elasticsearch”, (Accessed. 2018).

◐ 저 자 소 개 ◑

나 철 원(Chul-Won Na)
2016년~현재 군산대학교 소프트웨어융합공학과 재학중
관심분야 : 웹크롤링, 데이터마이닝, 빅데이터, 자연어처리
E-mail : ncw0034@kunsan.ac.kr

온 병 원(Byung-Won On)
1998년 안양대학교 컴퓨터공학과(공학사)
2000년 고려대학교 대학원 컴퓨터학과(이학석사)
2007년 펜실베이니아주립대학교 대학원 컴퓨터공학과(공학박사)
2014년~현재 군산대학교 소프트웨어융합공학과 부교수
관심분야 : 데이터마이닝, 빅데이터, 기계학습, 인공지능
E-mail : bwon@kunsan.ac.kr