# 자바스크립트 함수 처리 기능을 포함한 분산처리 방식의 웹 수집 로봇의 설계

김대유* · 김정태*

## Efficient Design of Web Searching Robot Engine Using Distributed Processing Method with Javascript Function

Daeyu Kim* · Jung Tae Kim*


### 요 약

본 논문에서는 기존의 웹 수집 로봇에서 처리 하지 못하는 자바스크립트 함수 링크를 처리하기 위하여 인터넷 익스플로러의 “Active Script Engine”을 사용하여 웹 로봇을 구현하였으며, 또한 자바스크립트 함수 링크를 처리 하였을 경우 웹 수집 로봇의 수집량을 측정하기 위한 웹 수집 로봇을 개발하였다. 웹 수집 로봇을 개발하기 위해서 구글봇과 네이봇 등 웹 수집 로봇의 구조를 파악하여, 수집 로봇에 활용되는 구성요소를 구현하고 분산처리형태의 웹 수집 로봇을 설계하였다. 또한 제안된 웹 로봇에 제안된 자바스크립트 처리 모델을 추가하여 성능평가를 하였으며, 성능평가 방법은 자바스크립트를 사용하는 웹 사이트의 게시판을 대상으로 하여 웹 수집량을 비교 분석하였다. 웹 사이트 게시물 1000개인 경우, 일반 웹 로봇의 경우에는 1페이지밖에 수집하지 못하였고, 제안된 웹 로봇의 경우 1000개 이상의 웹 페이지를 수집하는 결과를 얻었다.

### ABSTRACT

This is an example of ABSTRACT format. 여기까지는 1단으로 편집하세요 In this paper, we proposed and implemented web robot using active script engine with internet explore to process javascript function link, which is not processing in conventional web searching robot. This web searching robot is developed to measure collecting amount of web searching robot with processing of javascript function link. We analysed the architecture of web searching robot with google and naybot to develope web searching robot, implemented element of configuration applicable to searching robot and designed with distributed processing type. In addition to, we estimated the proposed web robot employing javascript processing model and analysed the comparison of collecting amount of web site board using javascript. We obtained the result of 1,000 web page collecting compared to conventional method in case of 1,000 web site board.



키워드

웹 로봇 엔진, 웹 수집 로봇, 자바스트립트

Key word

Web engine, Web searching robot, javascript


* 목원대학교

접수일자 : 2009. 06. 30
심사완료일자 : 2009. 08. 27

## Ⅰ. 서 론

인터넷 이용이 활발해짐에 따라 수많은 정보들이 웹 문서의 형태로 공개되고 있으며, 이러한 웹 문서들을 효과적으로 검색하기 위하여 웹 검색 서비스들이 이용되고 있다. 웹 로봇은 지정된 URL 리스트에서 시작하여 웹 문서를 수집하고, 수집된 웹 문서에 포함된 URL들을 추출과정과 새롭게 발견된 URL에 대한 웹 문서 수집과정을 반복하는 소프트웨어로서 웹 검색 서비스의 구축을 위해서는 웹 로봇을 이용한 웹 문서 수집이 선행되어야 한다. 1990년대 중반의 웹 문서 수는 현재에 비하여 매우 적었기 때문에, 최초로 개발된 웹 로봇 Wanderer를 포함하여 이 당시 개발된 다수의 웹 로봇들은 대용량의 웹 문서들을 수집하도록 설계되지 않았다. 현재는 전 세계적으로 30억 개 이상의 웹 문서들이 존재하며, 국내에도 5천만 개 이상의 웹 문서들이 존재하고 있다. 따라서 이처럼 많은 수의 웹 문서들은 효율적으로 수집 할 수 있는, 즉 초당 수백 또는 수천 개의 웹 문서들을 수집 할 수 있는 웹 로봇의 필요성이 증가되고 있다.[1][2] 인터넷이 발달됨에 따라서 동적인 웹사이트가 증가하고 있다. 사용자들이 원하는 게시물을 등록하고, 삭제하고 수정할 수 있는 웹사이트가 대부분 차지하고 있으며, 웹 로봇은 이러한 특성들을 고려하여 설계되어야 한다. 웹 수집 로봇은 URL을 통해서 웹 문서를 수집하게 되는데, 웹 사이트 개발자에 의하여 만들어진 자바스크립트 함수로 링크가 연결되어 있는 경우에 해당 페이지를 수집 할 수 없으며, 해당 페이지에 연결된 웹 링크를 찾아갈 수 없기 때문에 많은 웹 페이지를 놓치게 된다.

또한 웹 수집 로봇은 대부분의 기업/업체(검색엔진)에서 사용되고 있지만 웹 수집 로봇의 소스는 공개되어 있지 않다. 본 논문에서는 그림 1과 같이 기존의 웹 로봇에서는 처리 하지 못하는 자바스크립트 함수 링크를 처리 하는 자바스크립트 모델을 제안하고, 제안된 스크립트 모델을 사용하는 웹 수집 로봇을 설계 및 구현 하였다. 또한 제안된 스크립트 모델을 사용하여 웹 수집 로봇의 수집 페이지의 양을 대상으로 웹 로봇의 수집량을 대상으로 성능평가 하였다. 웹 수집 로봇의 웹 페이지 수집량은 중요하기 때문이다. 마지막으로 결론 및 향후 연구에 대하여 알아본다.

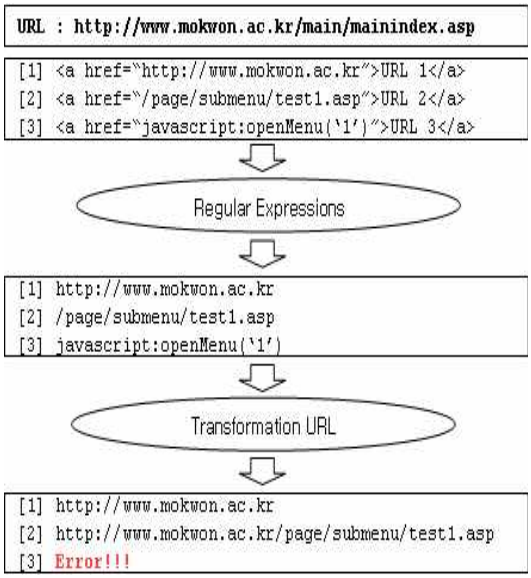


그림 1. 스크립트 처리가 불가능 한 예
Figure 1. Example of Impossibility of Script Processing

## Ⅱ. 관련 연구

### 2.1 구글봇(Google Bot)

구글봇(Googlebot)은 웹 검색 서비스를 제공하는 구글에서 사용하고 있는 웹 로봇으로, 스탠포드 대학의 학생이였던 Page & Brin에 의해 개발되었다.[3] 구글은 이러한 구글봇을 이용하여 전 세계를 대상으로 30억개 이상의 웹 문서를 수집하고 있다. 또한, 구글은 상업화된 이후에도 스탠포드 대학과 웹 문서 수집에 관련된 연구를 지속적으로 수행하고 있으며, 그 결과로는 웹 문서들이 병렬 수집[4], 중복된 문서들의 검출[5], 동적인 웹 문서들의 수집[6], 웹 문서들의 수정 주기 분석[7] 등이 있다. 그림 2는 구글봇 시스템의 구조를 보여 준다. 구글봇은 URL 관리기, 다운로더, 웹 문서 관리기, URL 추출기, URL 변환기로 구성되어 있으며, 각각의 구성 요소는 독립적인 프로세스로서 존재한다. URL 관리기는 수집할 웹 문서들의 URL들을 다수의 다운로더들에게 분배한다. 각각의 다운로더는 서로 다른 컴퓨터에서 실행되고, 웹 문서 관리기는 다운로드된 웹 문서들을 압축하여 디스크에 저장한다. URL 추출기는 디스크에 저장된 웹 문

서들로부터 URL들을 추출하고, URL 변환기는 이 URL들을 절대 URL로 변환하여 저장한다.

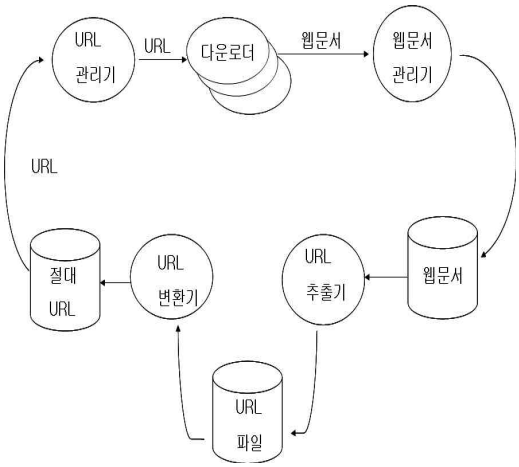


그림 2. 구글봇 시스템의 구조
Figure 2. Architecture of Google Bot System

### 2.2 네이봇(Nabot)

네이봇(nabot)은 웹 검색 포탈 네이버에서 사용하는 웹 로봇으로서 국내 및 일본의 웹 문서들을 수집한다. 네이봇은 데이터베이스 관리 시스템이 MySQL을 사용하여 수집된 웹 문서들을 관리하며, 또한 과거에 수집된 웹 문서들을 지속적으로 수집하기 위하여 지금까지 수집된 전체 웹 문서들의 URL을 관리한다. 네이봇은 관리중인 URL들이 지시하는 웹 문서만을 수집하고, 수집된 웹 문서들로부터 발견된 새로운 URL들을 URL 데이터베이스에 추가한다.

따라서 새롭게 발견된 URL들이 지시하는 웹 문서들은 다음번 네이봇 수행 시에 수집된다.[8] 그림 3은 네이봇 시스템의 구조를 보여준다. URL 분배기는 관리중인 URL들을 다수의 컴퓨터에 분산되어 있는 웹 문서 수집기들에게 분배하며, 웹 문서 수집기는 다음과 같은 작업들을 수행한다.

첫째, URL의 IP를 검사하여 국내 또는 일본의 웹 문서인지를 확인한다. 둘째, 로봇 배제 기준을 준수하기 위해서 웹 서버의 robots.txt 파일의 내용을 확인한다. 셋째, 웹 문서를 다운로드 한다. 넷째, 다운로드된 문서로부터 URL들을 추출하여 URL 검사기로 전달한다. 다섯째, 웹 문서의 내용을 분석하여 유해 또는 스팸 문서인가를 검사한다. 마지막으로, 웹 문서를 압축하여 데이터베이스에 저장한다. 한편, URL 검사기는 전달된 URL들 중에서 블랙리스트에 포함된 URL들과 기존의 URL 제거한 후, 나머지 URL들을 URL 데이터베이스에 추가한다.

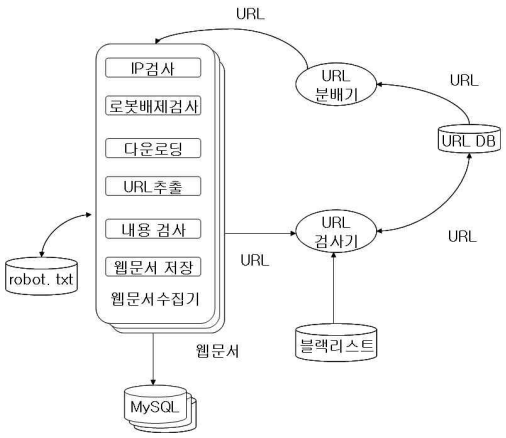


그림 3. 네이봇 시스템의 구조
Figure 3. Architecture of Nabot System

## Ⅲ. 제안된 웹 수집 로봇의 설계 및 구현

제안된 웹 수집 로봇은 로봇 관리자로부터 시작 되지 않으며, 큐를 중심으로 시작된다. 로봇 관리자는 현재 쌓여 있는 메시지 큐를 처리하기 위해서 각 관리자(수집 관리자, 다운로드 관리자, 에러 처리 관리자, 스크립트 관리자)를 생성하게 되며 처리된 큐는 삭제된다. 수집 처리 관리자에서는 페이지의 웹 주소를 가져오는데, 가져온 URL을 메시지 큐에 입력하고, 자바스크립트로 연결된 함수 링크의 경우에도 메시지 큐를 작성하여 등록한다. 이렇게 등록된 큐를 로봇 관리자에서 처리 하면, 웹 페이지의 모든 문서를 수집 할 수 있게 되는 것이다. 여기서 스크립트 관리자는 본 논문에서 제안된 관리자 이다. 이 스크립트 관리자는 수집 처리 관리자에서 처리 할 수 없는 링크 즉 자바스크립트 함수로 연결된 링크를 처리 하도록 도와주는 역할을 한다.

### 3.1 웹 로봇의 구성요소

1. 로봇 관리자

로봇 관리자는 수집 관리자와, 다운로드 관리자를 관리한다. 각 관리자의 개수를 정의하고 생성하는 관리를 하며, 수집 관리자로부터 검색된 URL을 중심으로 각 관리자들을 생성한다.

2. 수집처리 관리자

웹 페이지로 접속하여 페이지를 수집하는 역할을 하며 수집에 실패했을 경우, 에러 관리자가 다시 처리 할 수 있도록 에러 메시지를 작성하고, 수집하려는 URL의 정보가 파일형식일 경우는 다운로드 관리자에서 처리하도록 메시지를 작성하며, 자바 스크립트 함수로 연결된 링크 또한 스크립트 관리자에서 처리 할 수 있도록 메시지를 작성하는 역할을 한다.

수집 처리 관리자에서는 웹 수집 로봇에서 사용되는 파서가 요구된다. 웹 로봇용 파서는 웹 페이지의 소스코드에서 href 의 내용을 검출하여 주소를 획득할 수 있도록 처리 하면 된다. 또한 자바 스크립트 함수 링크로 연결되어 있을 경우에는 그에 따른 처리도 가능 하여야 한다.

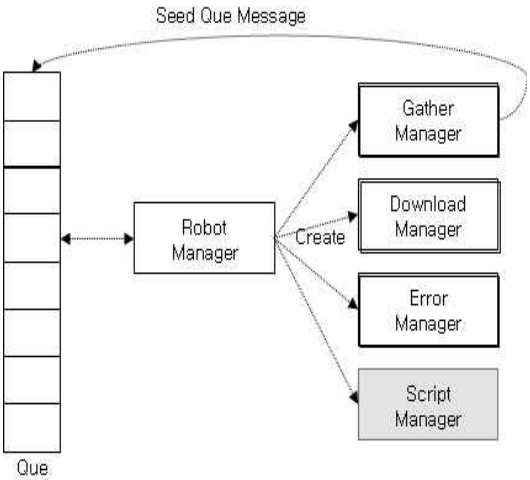


그림 4. 웹 수집 로봇의 기본 구조
Figure 4. Basic Structure of Web Searching Robot

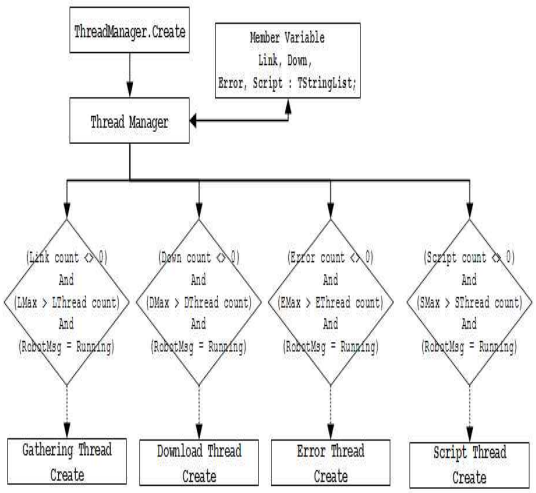


그림 5. 로봇 관리자의 구성
Figure 5. Configuration of Robot Administrator

3. 다운로드 관리자

수집처리 관리자로부터 URL헤더의 정보가 파일형식일 경우, 로봇 관리자는 다운로드 관리자에게 파일 URL을 전달하여 이 부분에서 처리 하게 된다. 파일을 다운로드 받아 웹 수집 로봇의 목적에 맞게 처리 하는 모듈이 있다.

4. 에러 관리자

수집처리 관리자에서 URL의 헤더를 다운로드 받거나, URL 페이지를 다운로드 받을 때 에러가 발생하였을 경우 특정 배열리스트에 URL 항목을 저장하게 되는데, 로봇 관리자는 재점검에 필요한 에러에 따라서 에러 관리자에게 항목을 전달하여 처리 하도록 한다.

5. 스크립트 관리자

수집처리 관리자에서 자바스크립트 함수 링크로 연결되어 있는 주소의 경우 이 스크립트 관리자에게 메시지를 전달하여 처리 한다.

### 3.2 제안된 스크립트 관리자

제 3.1절의 수집 처리 관리자에서 스크립트 함수 링크를 추출하게 되었을 경우 스크립트 처리 메시지를 작성하여 기록한다.

웹 로봇 관리자에서 해당 메시지가 있을 경우 이 관리자를 실행하게 되는데, 스크립트 관리자에서는 스크립트의 메시지를 분석하여 처리 한다. 스크립트 메시지를 파싱하여, 허용된 스크립트 함수일 경우 해당 메시지를 처리 하고 그렇지 않을 경우는 파기 하도록 설계해야 한다. 그 이유는 해당 처리 함수를 실행하였을 경우 익스플로러의 인쇄창이 뜨거나 해당 게시물이 삭제 될 수 있는 위험이 있기 때문이다. 우선 이 모듈을 개발하기 위해서는 MicroSoft Windows에서 제공하는 MSHTML.DLL 를 사용해야 합니다. 자바스크립트를 처리하기 위해해서 Internet Explorer에서 제공하는 SHOCVW.DLL과 MSHTML.DLL을 사용하면, ActiveX Control, Active X Script Engine, Java Applet, Plug-in 과 같은 여러 가지 객체를 제어 할 수 있게 됩니다.

그중 자바스크립트로 사용되는 링크의 결과 주소를 알아야 하기 위해서 사용된 부분이 바로 Java Script Engine이다. HDOCVW.DLL은 WebBrowser Control이며, MSHTML.DLL은 HTML 파서라고 볼 수 있다. SHOWDOCVW에서 지원되는 기능은 매우 다양하다. 그 중에 대표적인 기능이 Flash 와 ActiveX 컨트롤이다. MSHTML은 HTML과 Java Script와 같은 스크립트를 파싱하여 사용자에게 나타내주는 역할을 한다. 이 자바 스크립트 관리자 에서는 인터넷 익스플로러의 객체를 사용하여 자바스크립트를 처리 할 수 있는 기능을 가질 수 있는 것이다.

본 논문에서 구현된 자바스크립트 관리자의 경우에는 Internet Explorer에서 제공하는 기능을 사용하기 때문에, Windows 시스템 이외의 시스템(UNIX, Linux, Solraris)와 같은 운영체제에서는 사용이 불가능 하다. 모든 웹브라우저 컴포넌트는 "IWebBrowser2"라는 인터페이스를 사용 한다. VC++, VB, Delphi, 등으로 이 포함된다. Delphi의 경우 "TWebBrowser"라는 컴포넌트 역시 MS의 웹브라우저 컴포넌트를 포장해서 사용하는 관계로 "IWebBrowser2"의 인터페이스를 가지고 있다. 바로 "Ole Object property"를 사용하기 때문이다.

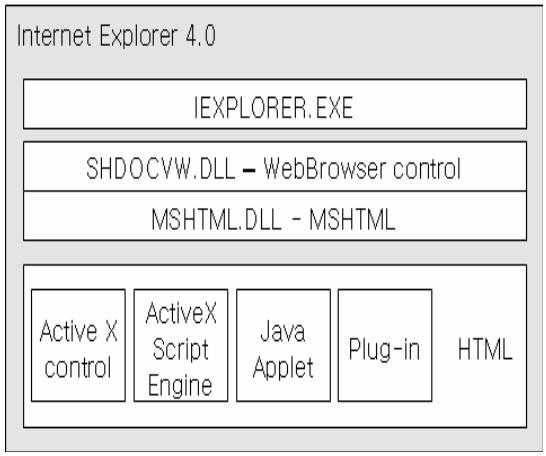


그림 6. 인터넷 익스플로러의 전체구조[10]
Figure 6. Total Architecture of Internet Explorer

또한 IWebBrowser2 의 값을 알고 있기 때문에 필요하다면 그 상위의 인터페이스들을 알아 낼 수도 있으며, IWebBrowser2로 IWebBrowserApp를 액세스하기 위해서는 다음의 웹 브라우저의 접근 예제와 같이 할 수 있다.

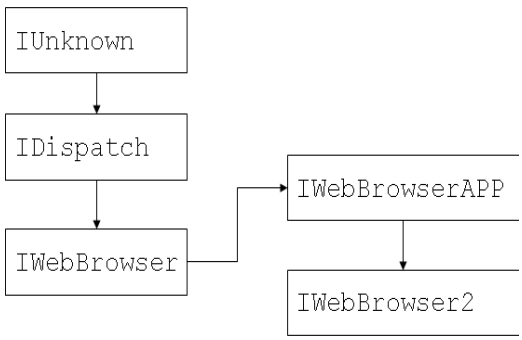


그림 7. 웹 브라우저의 레이아웃[11]
Figure 7. Layout of Web Browser

자바스크립트를 처리 하는 순서는 웹 오브젝트를 생성하고 생성된 웹 오브젝트에 해당 자바 스크립트의 페이지 소스를 삽입 후 자바스크립트를 실행하는 것이다. 스크립트 관리자의 처리 과정은 3단계로 나뉘어져있다. Enable, JBScript, Disable 이 스크립트 관리자는 DLL 로 구성되어 있다.

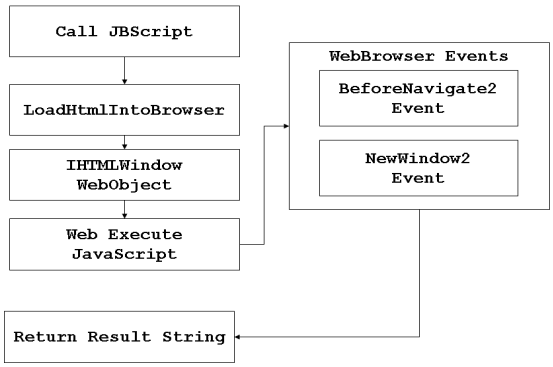


그림 8. 스크립트 관리자의 구성
Figure 8. Configuration of Script Administrator

## Ⅳ. 제안된 모델의 성능 분석

### 4.1 수집 모델의 쓰레드 처리 모델의 성능평가

90개의 웹 페이지를 생성하여 업로드 한 뒤에, 멀티 쓰레드 방식과 싱글쓰레드 방식으로 테스트 한 결과 멀티 쓰레드 방식은 9.250 Sec 이 소요 되었으며, 싱글 쓰레드 방식은 15.578 Sec 이 소요 되었음을 알 수 있다. 만약 웹 사이트에서 네트워크 장애가 발생하였을 경우 싱글 쓰레드의 경우에는 소요시간이 더 지체 될 수 있다.

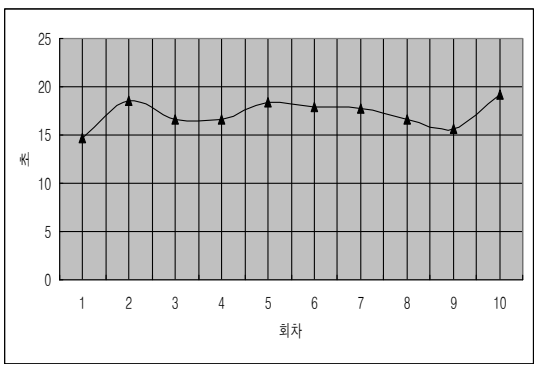


그림 9. 싱글 쓰레드 처리 다운로드 모델 성능평가
Figure 9. Evaluation of Performance for Single Thread Processing Download Model

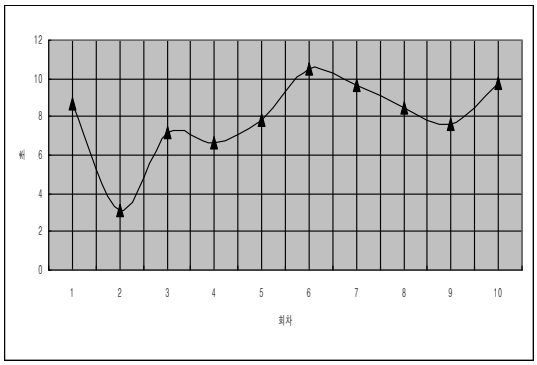


그림 10. 멀티 쓰레드처리 다운로드 모델 성능평가
Figure 10. Evaluation of Performance for Multi Thread Processing Download Model

### 4.2 제안된 웹 수집 로봇의 성능평가

웹 수집 로봇의 성능평가 기준에는 효율성, 지속성, 신성성, 포괄성, 정숙성, 유일성, 안정성 등이 있다.[2] 하지만 본 논문에서 웹 수집 로봇의 자바스크립트 함수로 연결된 링크의 해결 방안을 제안하므로 웹 수집 로봇에서 제안 모델을 삽입 한 것과 그렇지 않은 것에 대한 수집 개수를 측정하였다. 성능평가는 자바스크립트를 처리하지 않는 모델과, 자바스크립트를 처리하는 모델 2가지로 나뉘어 페이지 수집 량을 계산해 보았다.

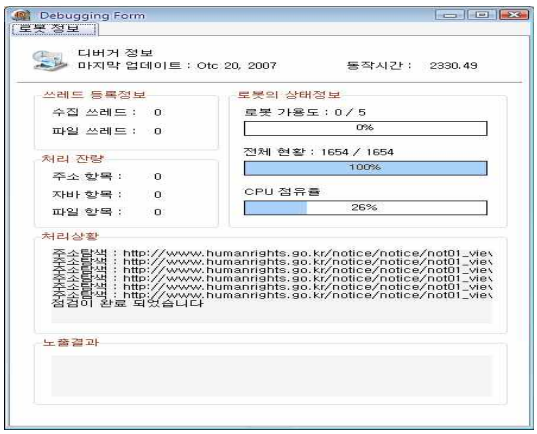


그림 11. 스크립트 처리
Figure 11. Script Processing

자바스크립트 처리 모델에 특정 사이트에 한가지의 정의 함수만 지정해놓고, 웹 사이트의 수집을 시작하였

다. 동작시간은 2330.49초 소요되었고, 수집 페이지의 개수는 1654 페이지를 수집하였다. 그에 반면, 스크립트를 처리 하지 않고 돌렸던 웹 수집 로봇의 경우, 72초 소요하였고, 수집 페이지의 개수는 153 페이지를 수집하였다.

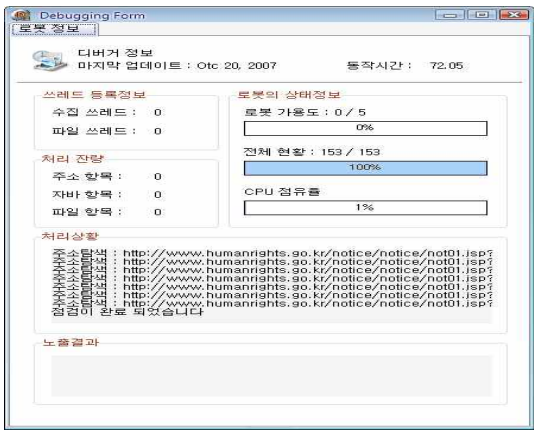


그림 12. 스크립트 미처리
Figure 12. Non Script Processing

웹 사이트에서 자바스크립트 처리 모델이 사용되지 않을 경우 웹 수집 로봇의 페이지 점검페이지의 개수는 동일 하지만, 스크립트가 사용된 웹 페이지이의 경우 페이지 수집 량에 차이가 나타난다. 스크립트를 처리 하여 처리된 웹 페이지를 처리 하게 될 경우 웹 사이트마다 차이가 있지만 더 많은 페이지의 링크를 수집 할 수 있다. 웹 사이트의 자바스크립트가 사용되는 게시판을 대상으로 수집을 테스트 한 결과를 표 1 도메인별 게시판의 수집량 테스트 결과에서 나타내고 있다.

표 1. 도메인별 게시판의 수집량 테스트
Table 1. Collecting Amount Test of Board on Domain Type

| 웹 수집 로봇 | | | |
|---|---|---|---|
| 기존 로봇 | | 제안된 로봇 | |
| 수집량 | 소요시간 | 수집량 | 소요시간 |
| 133 | 59.95 | 1464 | 395.67 |
| 1 | 1.61 | 76 | 17.36 |
| 22 | 3.2 | 70 | 18.16 |
| 155 | 100.94 | 1650 | 1587.42 |
| 20 | 8.83 | 30 | 9.95 |

자바스크립트가 사용되는 경우 수집량의 차이가 확연하게 나타나고 있는 것을 확인 할 수 있다. 기존의 로봇에서는 *****.gangwon.kr 도메인을 처리 했을 경우에는 자바스크립트 함수 링크를 처리 하지 못하여 1페이지 밖에 수집하지 못하는 문제점이 있었지만, 제안한 웹 로봇의 경우에는 76페이지를 수집하였다.

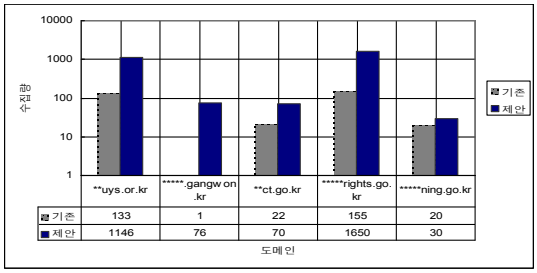


그림 13. 수집량의 성능평가
Table 13. Evaluation of Performance for Collecting Amount Test

## V. 결 론

본 논문에서는 웹 수집 로봇의 수집처리 관리자에서 페이지의 URL파싱처리 중 처리 하지 못하는 자바스크립트 함수 링크를 처리하기 위하여 인터넷 익스플로러 객체에서 제공하는 "Active Script Engine"을 활용하여 처리하는 방법을 제안하였다. 기존의 공개된 웹 로봇의 경우에는 단순한 상대경로의 URL링크만 절대 경로의 URL로 변경하여 사용하였기 때문에 자바스크립트 함수 링크로 사용된 URL 링크의 경우 대부분의 페이지를 수집 하지 못하는 방면, 제안된 웹 수집처리 모델에서 사용되는 자바스크립트 관리자를 추가한 경우 자바스크립트 함수 링크를 처리하기 때문에 웹 수집 로봇의 페이지 수집량이 증가하는 것을 확인 할 수 있었다. 특정 사이트의 자바스크립트 함수 링크로만 만들어진 게시판의 게시물의 개수가 1000개일 경우 일반 웹 수집로봇은 1페이지밖에 수집하지 못하였고 제한된 웹 수집로봇의 경우 게시물의 개수 1000개만큼 수집 할 수 있었다. 웹 수집 로봇의 속도도 중요하지만 많은 데이터를 확보하기 위해서는 페이지 수집이 정확해야 한다.

## 참고문헌


[ 1 ] M. Gray, "Internet Growth and Statistics: Credits and Background "http://www.mit.edu/people/mkgray/net/background.htm

[ 2 ] Kwang Hyun Kim, "A Methodology for Performance Evaluation of Web Robot, Korea Information Processing Society Vol. 11, No. 3, June 2004, pp. 563-565

[ 3 ] 김광현, 이준호, "웹 로봇의 성능 평가를 위한 방법론", 정보처리학회, 제11D권, 제3호, 2006. pp.563-570

[ 4 ] J. cho and H. Garcia-Molina, "Parallel Crawler," In Pro-ceedings of the 11th International World Wide Web Conference, Hawii, USA, 2002 , pp. 2-12.

[ 5 ] Beitzel et al., 2007 Beitzel, S. M., Jensen, E. C., Lewis, D. D., Chowdhury, A., & Frieder, O. (2007). Automatic classification of Web queries using very large unlabeled query logs. ACM Transactions on Information Systems, 25(2), Article No. 9.

[ 6 ] J. Cho and H. Garcia-Molina, "The Evolution of the Web and Implications for an Incremental Crawler," In Proceedings of the 26th International Conference on Very large Databases, Cairo, Egypt, 2000, pp. 5-20.

[ 7 ] J. Cho, N. Shivakumar and H. Garcia-Molina, "Finding replicated Web Collections," In Proceedings of the ACMSIGMOD International Conference on Management of Data, dallas, Texas, 2000.

[ 8 ] A.Heydon and M. Najork, "Mercator: A Scalable,: Extensible Web Crawler," InRecordings of the 8th World Wide Web Conference, Toronto, Canada, 1999, pp. 2-7.

[ 9 ] M. Najork and A. Heydon, "High-Performance Web Crawling," SRC Research Report 173, Compaq Systems Research Center, 2001, pp. 2-8.

[10] Microsoft MSDN Library "Internet Explorer Architecture",

[11] Han Back Bae "How to Use, TWebBrowser Object" http://www.delmadang.com/community/bbs_view.asp? bbsNo=3&bbsCat=43&indx=195149&keyword1=webbrowser&keyword2=


## 저자소개


**김대유(Dae-Yoo Kim)**

2006년 2월:목원대학교 전자공학과 학사
2006년 3월~2008년 2월 목원대학교 전자공학과 석사
2008년 3월~현재 : 목원대학교 전자공학과 박사과정 및 (주)위너다임 연구원
※관심분야: 웹보안, 개인정보보호, 프라이버시 시큐리티

**김정태(Jung-Tae Kim)**

2001년 8월 : 연세대학교 대학원 전자공학과 박사
1991년 8월～1996년 2월 : 한국전자통신연구원(ETRI)선임연구원
2002년 9월～현재 : 목원대학교 전자공학과 교수
※관심분야: Microwave photonics, Optically fed wireless communication system design, Information security system design, Network Security, ASIC Design.