메타 검색엔진을 위한 페이지 변경 탐지기 설계

박상위 오정석 이상호
숭실대학교 컴퓨터학부
{gypsy, dbstar}@orion.soongsil.ac.kr, shlee@computing.soongsil.ac.kr

Design of A Page Modification Detector for Meta-search Engines

Sang-Wee Park Jeong-Seok Oh Sang-Ho Lee
School of Computing, Soongsil University

요 약

웹 상의 HTML 문서들은 수시로 변경되고 있으며, 정보를 검색하는 웹사이트 또한 예외는 아니다. 다수의 웹 검색엔진들의 결과를 통합하는 메타 검색엔진은 각 검색엔진의 정보 변경에 민감해야 된다. 본 논문은, 수시로 변경되는 검색엔진들의 HTML 문서 정보를 메타 검색엔진에 반영하기 위해, 자동으로 검색엔진들의 질의 형태 변경과 검색 엔진의 검색 결과 HTML 문서의 구조 변경을 탐지하는 페이지 변경 탐지기를 설계하고자 한다. 검색 질의 형태 변경은 검색 엔진에 질의 시에 반환되는 HTTP 응답 코드를 분석하여 감지한다. HTML 문서의 구조 변경 탐지는 질의 결과가 반복되는 HTML 태그(tags) 문서 구조를 패턴(pattern)으로 이용한다. 패턴 발견 알고리즘은 문자열에서 규칙적으로 발생하는 패턴을 찾아내는 Jaak Vilo 알고리즘을 기반으로 하여 HTML 문서를 처리할 수 있도록 확장하였다. 발견된 HTML 문서 패턴과 기존의 검색 엔진 HTML 페이지의 구조적 패턴 정보를 비교하여 문서 구조 변경을 탐지한다.

1. 서론

다양한 특성을 가지는 검색 엔진들이 웹 상의 정보를 쉽게 접근하는 도구로서 많이 사용되고 있다[2]. 사용자들은 검색하려는 목적에 적합한 검색 엔진을 선택하여 사용하게 된다. 단일 검색엔진은 사용자 질의에 대해 부적절(irrelevant)하거나 질의 결과 페이지에서 연결(link)이 끊어진 페이지를 결과로써 출력하는 경우가 많다. 검색 엔진의 이러한 단점을 극복하는 메타 검색엔진은 동일한 사용자 환경(user-interface)을 통하여 특성이 다른 다수의 검색엔진으로 사용자 질의를 전달하고, 결과들을 병합하여 통일된(uniform) 형태로 보여준다[4]. 메타 검색엔진은 검색 엔진들의 정보를 기반으로 동작하기 때문에 각 검색 엔진에 대해 의존적이다. 검색엔진의 사용자 입력 페이지와 질의 결과 페이지의 구조는 수시로 변경되고 있으며 메타 검색엔진은 변경된 정보를 즉시 반영하여야 한다. 따라서, 메타 검색 엔진에 적용되는 검색 엔진들의 구성 정보가 변경되었는지를 자동으로 감지하는 도구가 필요하다.

본 논문에서는, 메타 검색 엔진에서 사용하고 있는 검색엔진의 웹 페이지의 변경을 탐지하는 페이지 변경 탐지기를 설계한다. 웹 페이지의 변경은 검색 질의의 변경과 검색 결과 페이지의 구조 변경으로 나뉘어서 탐지한다. 검색 결과 페이지 변경은 페이지 내의 반복 구조의 패턴을 이용하여 탐지한다. 검색 결과 페이지내의 패턴 발견 알고리즘은 문자열 내에서 규칙적으로 지정된 반복 횟수이상 발생하는 패턴을 발견하는 Jaak Vilo 알고리즘을 확장하여, HTML 문서를 처리할 수 있도록 수정하였다.

*본 논문은 한국과학재단 목적기초연구(2000-2-51200-002-3) 지원으로 수행되었음.

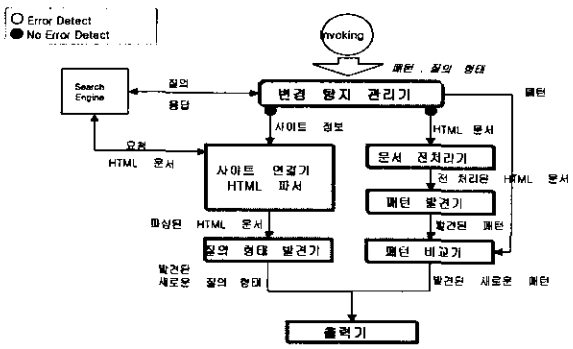
2. Jaak Vilo 알고리즘

Suffix Trie[1,3,6,7]는 TRIE나 PATRICIA와 유사한 자료구조로서, 텍스트 문자열에서 반복적인 부 문자열을 처리하는 자료 구조이다. Jaak Vilo 패턴 발견 알고리즘[5]은 Suffix Trie 자료 구조를 기반으로 하고, 길이가 n인 문자열 내에서 k회 이상 반복적으로 발생하는 패턴을 발견하는 알고리즘이다. Jaak Vilo 알고리즘은 문자열 내에 존재하는 모든 문자의 위치 정보와 출현 횟수를 이용하여 반복적으로 발생하는 패턴을 발견한다. 기존의 Suffix Trie는 문자열에서 발생하는 모든 Suffix를 트리(tree)로 표현하고 있기 때문에 메모리 사용량이 많았다. 그러나, Jaak Vilo 알고리즘에서는 문자열 내에 n회 이상 나타나는 문자들만으로 트리를 생성함으로서 Suffix Trie 자료구조의 단점인 메모리 사용량을 줄였다. Vilo 트리에서 문자열에서 n회 이상 발생하는 패턴

추출 방법은, 생성된 트리의 루트로부터 DFS(Depth-First-Search) 방식으로 노드를 검색한다.

3. 페이지 변경 탐지기 설계

[그림 1]은 페이지 변경 탐지기의 전체 구조도를 보이고 있다. 페이지 변경 탐지기는 크게 질의 형태 발견기와 패턴 발견기의 두 부분으로 설계되었다.



[그림 1] 패턴 변경 탐지기의 구조도

3.1 변경 탐지 관리기

변경 탐지 관리기는 시스템 전체 모듈을 관리한다. 변경 탐지 관리기는 검색 엔진의 질의를 입력으로 받아 각 검색엔진을 검색하고, 검색엔진으로부터 응답인 HTTP 처리 코드에 의해 질의 형태의 에러를 탐지한다. 변경 탐지 관리기는 질의 형태에서 에러가 탐지되면 질의 형태 발견기 모듈을 수행하고, 질의 형태에서 에러가 감지되지 않으면 패턴 발견기를 수행한다.

3.2 질의 형태 발견기

검색 엔진의 질의 형태가 변경됐음을 탐지하기 위해서 검색 엔진의 응답(response)인 HTTP 상태 코드를 분석하여 사용한다. 웹 서버(Web Server)의 HTTP 상태 코드는 [표 1]과 같다.

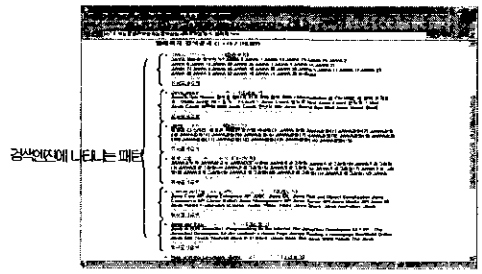
[표 1] HTTP 상태 코드

코드 범위	코드 의미
100-199	정보관련 코드 영역
200-299	클라이언트 요청 성공
300-399	클라이언트 요청이 리다이렉트됨
400-499	클라이언트 요청 불완전
500-599	서버 에러

질의를 검색엔진으로 전송 시에 에러가 발생한 때는 질의 형태가 변경된 경우이다. 질의 형태가 변경되었을 경우에는 변경된 새로운 질의 형태를 발견하는 모듈을 수행한다. 사이트 연결기와 HTML 파서 모듈은 검색 사이트의 첫 페이지 문서를 받아온 후 문서를 파싱(parsing)한다. 질의 형태 발견기는 파싱 된 페이지 내의 <FORM> 태그의 정보를 이용하여 새로운 검색 질의를 생성한다. <FORM>태그의 속성(attribute) 중 하나인 action 값은 웹 상에서 실행되는 프로그램을 지정하고 있으므로, 검색 사이트의 실제 검색 프로그램에 대한 정보를 추출할 수 있다. 또한, 질의 형태를 구성하고 있는 여러 가지 변수들에 대한 정보는 <FORM>의 구성요소(element)[8]의 name, value 값을 추출하여 이용한다. 추출한 값들을 순서대로 조합하여 검색 엔진에서 사용하는 형태의 새로운 검색 엔진 질의로 생성된다.

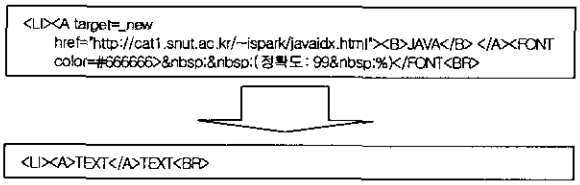
3.3 패턴 발견기

패턴 발견기는 검색 엔진에서 받아온 질의 결과 HTML 문서 내에서 반복되는 패턴을 찾는다. HTML 페이지는 [그림 2]와 같은 태그와 텍스트의 순서에 기반을 둔 문서 구조 패턴을 가지고 있다.



[그림 2] 검색엔진에서 나타나는 구조 패턴

HTML 문서는 문서 전체가 문자열로 취급되어 패턴 발견기로 입력된다. HTML 문서 전체를 정형화된 입력 형태로 만들어 주기 위해, 태그와 텍스트로만 나타나도록 변환하는 HTML 문서의 전처리(pre-processing)과정이 필요하다.



[그림 3] HTML 문서의 전처리

HTML 문서 전처리의 전략은 첫 번째, HTML 문서에서 태그가 아닌 문자열들은 상이한 형태이어서 정형화된 형태를 형성하기 위해, "TEXT" 문자열로 변환한다. 두 번째는 패턴의 구성요소에서 빠질 수 있는 태그를 고려하여 삭제하도록 한다. <HEAD>, <SCRIPT>, <FORM> 등의 태그들은 패턴을 구성하는 것이 아니라, 페이지의 동적인 실행과 관련된 태그이며, <META> 태그는 페이지의 메타 정보를 나타내기 위한 태그이므로 패턴과는 무관한 태그이다. 따라서, 삭제 가

능 태그는 <SCRIPT>와 </SCRIPT>, <FORM>과 </FORM>, <HEAD>와 </HEAD>이다. 세 번째는 HTML 문서에 나타난 모든 태그는 속성을 무시하고 태그 명만 가지도록 다음의 [예제]와 같이 처리한다.

[예제] HTML 태그의 전처리

<TABLE WIDTH="100%"> → <TABLE>

전처리 과정을 거친 문서는 태그와 "TEXT"의 조합으로 구성된 하나의 문자열로 취급되어 패턴 발견기로 입력된다. 패턴 발견기는 전처리 된 문서로부터 패턴을 발견하기 위해서 문서를 토크나이징(tokenizing)하고 트리를 생성한다. 문서의 토크나이징은 한 문자가 아니라, 태그와 "TEXT" 단위로 이루어진다.

패턴 발견기는 입력된 문자열의 Vilo 트리를 생성하여 패턴을 추출한다. Jaak Vilo 알고리즘의 단점은 첫 번째, 입력 문자열 S가 "ATACATA" 일 때, 반복횟수를 2로 지정한 경우, 문자열 S의 최장 패턴인 "ATA" 뿐만 아니라, "ATA"의 부-문자열(sub-string)인 "TA", "A"를 패턴으로 인정하기 때문에, 최장 패턴과 겹치는 패턴들을 생성하여 하나의 검색 페이지 당 평균적으로 50개 이상의 패턴을 출력하고 있다. 이 문제를 해결하기 위해, 부-문자열을 제거하였다. 두 번째, Vilo알고리즘은 문자열 내의 모든 문자에 대해 패턴을 발견하고 있다. 그러나, HTML 태그 중에는 문서의 구조와 상관없이 불규칙적으로 나타나는 태그들이 존재한다. 페이지에 불규칙적으로 나타나는 태그로 인해 페이지 내의 패턴을 찾는 것이 불가능하다. 이러한 태그는 <FONT>, <I>, <SMALL>, <STRONG>, <B>, <U>와 같은 스타일에 관련된 태그들이 있다. 이 태그들은 페이지 내에 나타나는 횟수와 위치를 예측할 수가 없다. 따라서, 문서의 구조와는 상관이 없는 태그들을 "TEXT"로 변환하는 과정을 Vilo 알고리즘에 추가하였다.

3.4 패턴 검증 작업

패턴 발견기에서는 1개 이상의 패턴을 추출하게 된다. 패턴 검증작업을 통하여 다수의 패턴 중에서 메타 검색엔진이 각 검색 페이지로부터 정보를 추출하는데 필요한 정보가 포함된 패턴을 찾아내야 한다. 패턴 검증의 전략은 첫 번째, 패턴으로 출력되는 결과들 중에는 "<A> TEXT </A> TEXT <A> TEXT </A> TEXT"처럼 3종류 이하의 태그들이 반복적으로 패턴을 구성하기도 하기 때문에, 이러한 결과들은 패턴 대상에서 삭제할 수 있다. 두 번째는, 메타 검색엔진에서 정보를 통합하는데, 사용하는 각 검색엔진의 결과 부분에는 반드시 <A>태그를 포함하여야 한다. 각 검색엔진들은 질의와 연관된 여러 개의 페이지 혹은 사이트들의 URL을 한 페이지에 보여주고, 사용자들이 직접 결과 링크를 선택하여 방문하도록 하고 있기 때문이다. <A> 태그가 포함되지 않은 패턴은 삭제하여 패턴에서 제외시켜야 한다. 패턴 검증 작업을 거치게 되면, 패턴의 개수를 평균적으로 5개 이하로 줄일 수 있다. 패턴 문자열과 기존의 패턴으로 정의되어 저장되어 있는 패턴 문자열을 비교하여, 패턴의 변경을 탐지한다.

4. 결론 및 향후과제

본 논문은 검색 엔진의 검색 결과 HTML 페이지가 반복되는 구조를 이루는 특징을 이용한 페이지 변경 탐지기를 설계하였다. 향후과제는 페이지 변경 탐지기를 구현하여, 실제 검색 엔진들을 대상으로 시험을 하는 것이며, 현재는 국내 검색 엔진 9가지를 대상으로 시험을 하고 있다. 향후 국내 검색 엔진뿐만 아니라, 해외 검색엔진으로 범위를 확장하여 시험하여야 한다.

페이지 변경의 탐지는 반복된 구조가 표현된 부분을 대상으로 하여, 패턴을 발견하고 기존의 패턴과 비교함으로써 수행되고 있다. 이 시스템의 특징은 태그와 텍스트의 순서에 기반한 구조를 패턴으로 인식하여, 패턴을 추출하여 사용하는 것이다. 메타 검색 엔진에서 필요한 정보를 추출하기 위해서 래퍼의 사용은 필수적이다. 현재까지 많은 래퍼(Wrapper)들이 개발되어져 있으나, 이들은 대부분 복잡한 데이터 추출 규칙을 사용하고 있다. HTML 페이지의 구조를 이용한 패턴을 래퍼 시스템에서 이용하게 되면, 기존의 래퍼 시스템보다 간단한 데이터 추출 규칙을 사용할 수 있는 장점이 있으므로, 패턴을 이용한 래퍼 시스템의 개발도 용이할 것이다.

5. 참고문헌

[1] M.Crochemore, W.Rytter, "Text Algorithms", Oxford University Press.

[2] D.Dreilinger, A.E.Howe, "Experiences with Selecting Search Engines Using Metasearch", TOIS 15(3): 195-222, 1997.

[3] D.Gusfield, "Algorithms on Strings, Trees, and Sequences", Cambridge University Press.

[4] E.Selberg, O.Etzioni, "Multi-Service Search and Comparison Using the MetaCrawler", Proceedings of the 4th International World Wide Web Conference.

[5] J.Vilo "Discovering Frequent Patterns from Strings", Technical Report C-1998-9 (pp. 20), Department of Computer Science, University of Helsinki, 1998.

[6] http://www.csse.monash.edu.au/~lloyd/tildeAlgDS/Tree/Suffix.html

[7] http://www.cs.mcgill.ca/~cs251/OldCourses/1997/topic7/

[8] http://www.w3.org/MarkUp/html-spec/