동적 색인 스토리지 및 통합 검색 서비스 개발

Dynamic index storage and integrated searching service development

이왕우*, 이석형*, 최호섭*, 윤화묵*, 김종환**, 허윤영**
상한국과학기술정보연구원*, 드림위즈**

Wang-Woo Lee*, Seok-Hyoung Lee*, Ho-Seop Choe*,
Hwa-Mook Yoon*, Jong-Hwan Kim**, Yoon-Young Hur**
KISTI*, DreamWiz Inc.**

요약

본 논문은 웹뉴스 및 리뷰 검색 서비스를 위해 만든 통합 검색 시스템을 소개한다. 검색 서비스를 위한 데이터 수집을 위해서 특정 사이트에서 수집한 뉴스와 리뷰 문서로부터 제목, 날짜, 저자, 본문처럼 특정한 영역의 데이터만 추출하는 XSLTRobot을 만들었다. XSLTRobot은 원하는 부분의 데이터만 추출하기 위해 XSLT 기술을 이용한다. 여러가지 검색 데이터 형식에 적합한 통합 검색엔진과 통합 검색엔진의 스토리지 모듈중 하나인 동적 색인 저장소(Dynamic Index Storage)를 소개한다. 동적 색인 저장소는 뉴스 데이터처럼 색인의 업데이트가 빨라야 하는 환경에 이용된다. 본 논문에서 제시하는 동적 색인 저장소는 대량의 실시간 업데이트 문서를 처리하지 않기 때문에 검색성능에 초점을 맞춰서 설계하였다.

Abstract

In this paper, the integrated search system made for the web news and review retrieval service is introduced. We made XSLTRobot that extract title, date, author and content from html document like news or reviews for search service. XSLTRobot used the XSLT technology in order to extract desired part of html page. The Intergrated Information Retrieval System(IIRS) is suitable for various search data format. And we introduce Dynamic Index Storage which is module of IIRS. Dynamic Index Storage is used to environment which needs fast index update like news. And it's design focused on retrieval performance because there was not many document that it has to update on a real time.

1. 서 론

검색 서비스 업체들의 기술은 시간이 갈수록 발전하고 있는데 그 중 하나가 동적 검색환경에서 색인데이터의 업데이트일 것이다. 몇 년 사이에 빠른 색인 업데이트가 가능한 검색 시스템들이 개발되었고 서비스되고 있다. 빠른 색인 업데이트가 필요한 대표적인 분야는 뉴스와 게시판 그리고 UCC일 것이다. 실례로, 기존의 많은 커뮤니티 사이트들에 있던 게시판은 데이터베이스를 이용한 Query검색이었는데 점점 검색엔진을 이용한 검색으로 변화하고 있다.

이 논문에서는 우리가 개발하고 서비스했던 시스템에 대해서 3 부분으로 나눠서 설명한다. 첫째는 xml과 xslt기술을 이용하여 웹상에 존재하는 html 페이지에서 특정한 데이터(제목, 저자, 가격, 내용, 날짜, 조회수...)만을 추출하는 로봇에 대해 설명할 것이다. 둘째는 색인 스토리지의 변경이 가능하고 여러 가지 다양한 검색 환경에서 사용 가능한 통합 검색엔진이다. 셋째는 동적 색인 스토리지를 구현하기 위한 설계에 대해서 설명한다.

2. 특정 데이터 추출을 위한 웹로봇

2.1 관련연구

일반적으로 HTML 페이지 전체를 저장하는 웹로봇과 달리 HTML 페이지에서 원하는 부분만을 추출하는 웹로봇을 만들기 위해서는 데이터 추출 규칙이 마련되어야 한다. HTML 페이지에서 특정 데이터를 추출하기 위한 방법은 여러 가지 연구가 있었다[8,10]. 온톨로지를 활용한 데이터 수집방법[13], 데이터 추출 언어를 이용하는 방법[12], 데이터 추출용 비주얼 툴을 이용하는 방법[11], XML의 표준기술을 이용하는 방법[8] 등 다양한 연구가 이뤄졌었다. 이 논문에서 활용한 방법은 가장 구현하기 쉬운 XML과 XSL기술을 이용하여 HTML 페이지에서 특정 데이터를 추출하는 방법이다[8]. XSL의 생성을 편리하게 하기 위한 GUI툴이나 wrapper를 개발하기 위한 여러가지 시도가 있었지만 이것도 사람의 수작업이 들어가야 하는 부분이고 툴 개발에 대한 시간이 많이 소요되기 때문에 XSL을 직접 작성하는 방법을 선택했다[11]. 물론 수집해야할 사이트가 아주 많거나 할 경우에는 좀 더 자동화 시키는 방법이 필요하지만 100개 이하의 사이트에서 데이터를 수집하는

데는 직접 XSL을 작성해도 큰 무리가 없었다. 우리가 만든 XSL을 이용하여 데이터를 수집하는 로봇을 XSLTRobot이라고 부른다.

2.2 XSLTRobot 설계

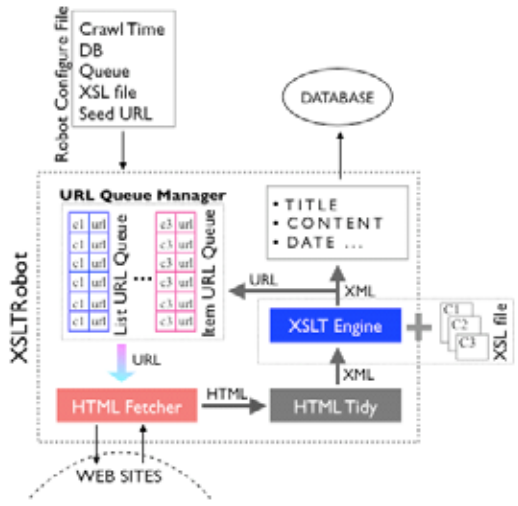
XSLTRobot은 Robot Configure File(xml)을 통해서 로봇 동작에 관한 전체적인 설정이 가능하다. Robot Configure File에는 데이터를 수집할 대상사이트의 부하를 줄이기 위한 문서 수집 간격의 휴식 시간, 프로그램의 동작 및 휴식 시간, 일일 수집할 페이지 개수 제한, Seed Url, Queue의 개수 및 Queue와 연결되는 CrawlTag(XSL file와 연결되는 Unique Tag)정의, DB접속 정보 등이 있다.

그림 1은 XSLTRobot의 전체적인 구성도이다. XSLT Robot의 기본적인 동작은 다음과 같다.

첫째, URL Queue Manager가 Queue의 우선순위에 따라 다운로드할 URL을 HTML Fetcher에게 넘겨준다. 각 URL은 다운로드된 문서를 XSL과 연결시키기 위해 CrawlTag가 부착되어 있다.

둘째, HTML Fetcher는 URL에 해당하는 HTML페이지를 다운로드 받고 HTML Tidy가 HTML 페이지를 XML형식으로 변환한다.

[3] XSLT Engine은 CrawlTag와 매칭되는 XSL 파일을 이용해서 원하는 데이터를 추출한다. 예를 들면, Item Page(게시글 페이지)에 접근하기 위한 List URL(게시판 리스트 페이지)인 경우에는 CrawlTag와 URL쌍이 추출될 것이고, Item URL인 경우에는 결과로 제목, 내용 그리고 날짜같은 데이터가 추출되서 Database에 저장된다.



▶▶ 그림 1. XSLTRobot 구성도

XSLT를 이용해서 HTML 페이지에서 원하는 정보를 추출하는 방법은 Jussi Myllymaki의 논문을 참고하였다[8].

DB에 저장된 데이터는 ITEM Sender에 의해 정의된 시간 단위로 IIRS(Intergrated Information Retrieval System)에게 전송되어 색인되고 서비스된다.

3. 통합 정보검색 시스템(IIRS)

드림위즈에서 사용하던 기존 검색엔진은 특정 검색 서비스(site, blog, knowledge, web, shopping, dic, news, image, movie, map etc…)에 맞추어진 엔진이라서 시스템의 성능에는 최적화되어 있지만 새로운 검색 서비스를 개발하기 위해서 기존 엔진의 내부를 수정하는 것은 불가피하였다.

그래서 검색엔진의 통합을 위한 통합 검색 시스템 개발이 필요하였고 개발에 고려한 조건은 다음과 같다.

첫째, 필드 및 구조가 다양한 문서를 처리한다.

필드 및 구조가 다양한 문서를 색인하도록 하기 위해서 색인할 문서의 구조로 XML을 이용하고, XML의 구조에 따라 색인할 부분이나 메타데이터로 갖고 있어야 할 부분을 메타정보로 처리한다. 그리고 검색(sorting, filtering, grouping …)에 필요한 필드 정보는 메모리에 로딩해서 처리한다.

둘째, 검색 서비스의 환경(색인의 업데이트 주기, 코퍼스의 크기)에 따라 적당한 색인 저장 스토리지를 선택할 수 있다.

실시간 검색환경에서 색인을 위해 사용하는 스토리지와 웹 문서같은 대량의 문서를 색인해야 하는 스토리지는 최적의 성능을 내기 위해 동일한 구조일 수가 없고 처리하는 방식도 다를 수 밖에 없다. 그래서 기존의 대량 데이터를 처리하는 스토리지 외에 동적 색인을 위한 스토리지를 추가적으로 개발하였고 기존의 엔진에서 사용하던 스토리지들도 모듈화해서 사용가능하도록 하였다.

셋째, 추가적인 검색기능이 요구될 경우를 대비하여 검색엔진의 검색기능을 확장하기 쉽도록 유지한다. 이를 위해서 검색엔진의 기능 설계에 OCP(Open-Closed Principle)의 개념을 적용하였다[6].

4. 동적 색인 저장소(DIS)

DIS(Dynamic Index Storage)는 IIRS에 속하는 모듈로 여러 가지 색인 저장 모듈중에 하나이다. DIS는 동적인 검색환경에서 빠른 업데이트가 필요할 때 사용하는 스토리지다. 실시간으로 큰 규모의 데이터를 처리하기에는 아직까지 개선할 점이 많다.

4.1 관련 연구

최근 들어 동적색인을 이용한 서비스도 많이 늘어나고 있고 관련 연구도 활발한 것 같다[1,2,3,5,9]. 인덱스의 업데이트 주기나 사용하는 장비의 상황에 따라 여러 가지 서비스 방법이 존재할 수 있는데 가장 간단한 방법은 새로운 인덱스를 만드는 동안 백업 인덱스로 서비스하는 것이다[2]. 이 방법은 업데이트 주기가 짧아지거나 실시간 색인과 검색을 요구할 경우에는 사용하기가 어려워진다.

다른 방법은 서비스 시간에는 auxiliary index에 새로운 포스팅 데이터를 저장해서 main index와 auxiliary index의 결과를 합쳐서 검색하고 검색량이 적은 밤에 auxiliary index를 main index로 업데이트하는 방법이 있다[1,2].

logarithmic merge방식도 실시간검색에 이용될 수 있지만 merge시간에는 검색응답이 느린 단점이 있다[3,4,9]. 하지만 대량의 문서도 처리할 수 있는 장점도 있다.

또 다른 연구로 Long-term에 대해서는 in-place update방식을 적용하고, short-term에 대해서는 lorgarithmic merge 방식을 이용한 하이브리드 방식이 있다[5]. Short-term에서 long-term을 구분하는 임계치를 하드 디스크의 특성에 따라 적절히 조절한다면 최적의 성능을 낼 것이라 생각된다. 이 논문은 시스템을 개발하고 난 뒤에 읽게 되었는데 long-term을 in-place방식으로 따로 관리하는 방식은 우리 시스템에서도 적용한다면 더 좋은 성능이 나올 것이라 생각된다. 그리고 하드디스크의 성능을 자동으로 측정해서 적절한 임계치 값을 알아내는 것도 범용 시스템 구축에 꼭 필요한 연구라고 생각된다.

4.2 동적 색인 저장소의 설계

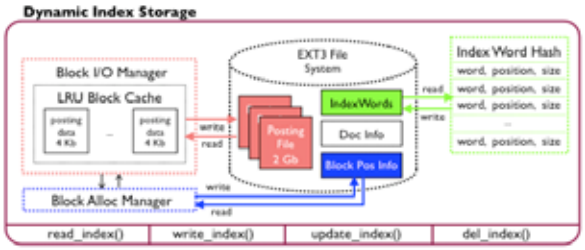
동적 색인 저장소를 만들 때 고려한 사항으로는

첫째, 파일에 데이터를 쓸 때 최적의 효율을 내기 위해 Ext3 파일시스템의 기본 block단위인 4k단위를 파일에 읽고 쓰는 최소 단위로 사용하였다. 그리고 디스크의 검색 시간을 최소화하기 위해 텀의 포스팅 데이터를 저장한 블록들이 연속적으로 저장되도록 한다. 그리고 한 블록이 가득 차서 저장될 수 있도록 메모리 캐시를 이용한다. 하지만 메모리 사용량의 제한으로 인해 항상 가득 찬 블록이 저장되도록 보장할 수는 없다.

둘째, 텀의 포스팅 데이터를 최대한 인접한 블록에 위치시킨다. Ext3 파일시스템은 1파일에 대해서 가능한한 128MB단위까지 물리적으로 블록들을 인접시키도록 구현되어 있다. 우리의 포스팅데이터는 2G단위로 세그먼트 파일을 구성하지만 텀의 포스팅 데이터를 인접하게 유지시켜서 가능하다면 물리적으로도 인접시키도록 노력한다.

셋째, 검색 성능과 색인어의 업데이트 성능 향상을 위해 메모리에 로딩된 블록들을 캐시로 이용한다. 제한된 메모리 관리는 LRU 캐시 방법을 이용하였다.

텀의 포스팅 데이터는 기본적으로 4 KB의 블록에 저장된다. 하지만 실제 많은 텀들이 1번 밖에 나오지 않는 경우가 많기 때문에 기본적으로 4 KB를 하나의 텀이 차지하기에는 공간의 낭비가 심해서 한 블록을 8로 나눈 512 바이트를 텀의 포스팅 데이터를 저장하는 기본 단위로 사용하였다.



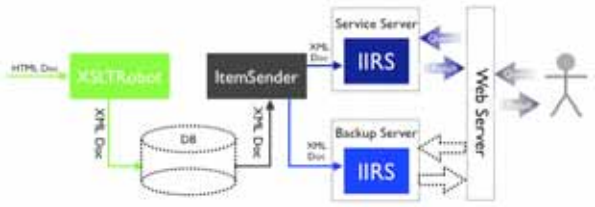
▶▶ 그림 2. 동적 색인 저장소의 구성

그림 2는 DIS의 전체적인 구조를 보여준다. 기본적인 인터페이스는 read_index(), write_index(), update_index(), del_index()를 제공한다. 색인어는 기본적으로 해시로 관리되고 색인어의 메타정보로는 포스팅 파일에서 위치와 포스팅 데이터의 사이즈를 가진다. Block Alloc Manager는 포스팅 파일에서 새로운 블록의 위치를 관리하고 클러스터된 블록을 관리한다. Block I/O Manager는 포스팅 데이터를 읽고 쓰는 역할을 한다. 그리고 지연된 쓰기를 통해 디스크 병목 현상을 최소화시키기 위해서 LRU Block Cache를 이용하였다.

포스팅 데이터의 저장방식은 10개, 100개 블록마다 서로 인접해서 저장되도록 구현하였다. 하지만 실제 서비스를 해 본 결과는 블록을 인접시키는데 소요되는 비용이 많이 높았고, 결과적으로 파일 I/O가 많아지게 되었다. 이런 문제로 인해 포스팅 데이터가 저장되는 블록을 클러스터링할 때 최적의 효과를 내기 위한 연구가 필요하다. 또한, 블록의 공간할당을 좀 더 효율적으로 할 수 있는 방법도 마련되어야 할 것이다.

5. 동적 검색 시스템

XSLTRobot에서 데이터를 수집해서 DB에 저장하면 ItemSender는 정해진 시간에 따라 수집된 데이터를 IIRS에 보내서 색인하도록 요청한다. IIRS는 색인과 동시에 검색이 가능해진다. 하지만 우리가 만든 DIS(Dynamic Index Storage)는 LRU Block Cache사용으로 인해서 블록에 저장된 포스팅 데이터가 파일에 바로바로 저장되지 않는 경우가 생기기 때문에 서버가 죽게되거나 정전이 되는 경우에는 문제가 될 수 있다. 그래서 서비스 서버외에 백업서버를 추가로 유지해야 했다. 이 부분은 DIS의 개선이 이뤄지면 백업서버는 필요없어질 것이다.



▶▶ 그림 3. 동적 검색 시스템의 구성

그림 3에서 보듯이 백업 서버는 문서를 색인만하고 서비스 서버에 문제가 있을 경우에 가동이 되어 동적 검색 서비스를 계속 유지할 수 있도록 한다.

6. 결론 및 향후 연구

우리가 의도했던 검색 시스템을 구축하는데는 만족스러운 결과이지만 시스템의 각 부분별로 개선시켜야 할 문제들이 남은 것 같다.

XSLTRobot이 대량의 사이트에서 문서를 수집하기 위해서는 XSL을 자동으로 생성하거나 태그패턴을 이용해서 데이터를 자동으로 추출하는 연구가 진행되어야 할 것이다.

동적 색인 스토리지를 처음 개발해서인지 성능적인 부분에 대해 개선해야 할 점이 많은 것 같다. 성능을 개선하기 위해 File의 I/O를 적절히 제어하는 연구가 필요하다. 또한 최적의 검색성능을 발휘하기 위해서 데이터를 읽고 쓸 때 하드디스크의 병목을 발생시키지 않는 절적한 임계치를 계산하기 위한 연구가 필요하다. 그리고 색인데이터가 커질수록 효율적인 처리를 위해서 하이브리드방식이 필요할 것 같다[5]. 또한 파일시스템에 최적화된 알고리즘도 연구되어야 할 것이다.

통합 검색 시스템은 여러 가지 유형의 문서를 색인하고 서비스할 수 있어서 우리가 원하던 요구를 해결하였지만 검색속도 향상을 위한 튜닝은 계속적인 숙제인 것 같다. 그리고 서버 구동 및 색인설정에 필요한 여러 가지 XML 설정 파일들로 인해 사용하기가 쉽지 않으므로 편리한 인터페이스의 개발도 필요할 것 같다.

▌참 고 문 헌▐

[1] D. Bahle, H.E. Williams and Justin Zobel, "Efficient Phrase Querying with an Auxiliary Index", Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval, pp. 215-221, 2002.

[2] C. Manning, P. Raghavan, and H. Schütze, "Introduction to Information Retrieval", Cambridge University Press, 2007, pp. 58-59.

[3] The Apache Lucene Project, http://lucene.apache.org

[4] Nicholas Lester, Justin Zobel, Hugh E. Williams, "In-place versus re-build versus re-merge: index maintenance strategies for text retrieval systems", Proceedings of the 27th Australasian conference on Computer science - Volume 26 ACSC '04, pp. 15-23, 2004.

[5] Stefan Büttcher, Charles L. A. Clarke, Brad Lushman, "Hybrid index maintenance for growing text collections", Proceedings of the 29th annual international ACM SIGIR conference onResearch and development in information retrieval, pp. 356-363, 2006.

[6] Meyer, Bertrand. "Object-Oriented Software Construction", 2d ed. Upper Saddle River, NJ: Prentice Hall, 1997.

[7] Hanny Yulius Limanto, Nguyen Ngoc Giang, Vo Tan Trung, Nguyen Quang Huy, Jun Zhang, Qi He, "An Information Extraction Engine for Web Discussion Forums", Special interest tracks and posters of the 14th international conference on World Wide Web, pp. 978-979, 2005.

[8] Jussi Myllymaki, "Effective Web data extraction with standard XML technologies", Proceedings of the 10th international conference on World Wide Web , pp. 689-696, 2001.

[9] Stefan Büttcher, Charles L. A. Clarke, "Indexing time vs. query time: trade-offs in dynamic information retrieval systems", Proceedings of the 14th ACM international conference on Information and knowledge management, pp. 317-318, 2005.

[10] Alberto H. F. Laender, Berthier A. Ribeiro-Neto, Altigran S. da Silva, Juliana S. Teixeira, "A brief survey of web data extraction tools", ACM SIGMOD Record Volume 31, pp. 84-93, 2002.

[11] Robert Baumgartner, Sergio Flesca, Georg Gottlob, "Visual Web Information Extraction with Lixto", Proceedings of the 27th International Conference on Very Large Data Bases, pp. 119-128, 2001.

[12] Robert Baumgartner, Sergio Flesca, Georg Gottlob, "The Elog Web Extraction Language", Proceedings of the Artificial Intelligence on Logic for Programming, pp. 548-560, 2001.

[13] Marc Ehrig, Alexander Maedche, "Ontology-focused crawling of Web documents", Proceedings of the 2003 ACM symposium on Applied computing, pp. 1174-1178, 2003.