RIA 기반 개인화 검색을 위한 Widget 응용의 구현

(RIA based Personalized Search with Widget Implementation)

박 차 라 † 임 태 수 ††
(Chara Park) (Taesoo Lim)
이 우 기 †††
(Wookey Lee)

요 약 쉽고 유연한 조작과 역동적인 화면구성에 초점을 맞춘 인터넷서비스 맞춤기술인 RIA(Rich Internet Application) 기술들은 웹2.0기술 중 사용자 편의성을 강조한 차세대 UI기술로 기대되고 있다. 본 논문은 평면적이고 순차적인 방법의 고급검색을 동적UI로 구현하고 사용자가 개인화 검색정보를 저장해서 검색에 활용할 수 있도록 구현하였다. 또한 사용자중심의 키워드 선호도를 통해 기존 웹 검색보다 개인화된 검색 결과물을 발견할 수 있는 검색구조를 설계하였다. 본 연구는 RIA 기술을 활용한 개인화 검색 관리자의 적용을 통해 검색된 페이지양의 감소를 입증하여 사용자에게 더욱 정제된 데이타를 제공하며 결론적으로 사용자들이 더욱 유연하고 편리한 방법으로 개인화된 웹 검색을 이용할 수 있음을 보였다.

키워드 : RIA, 웹 2.0, 개인화 검색, 사용자 인터페이스

Abstract Rich Internet Application(RIA) is one of the Web 2.0 technologies and is expected to be a next generation user interface technique which allows flexible and dynamic manipulation for Web searches. This paper addresses a personalization mechanism for advanced Web search using RIA for abundant user interactions. We devised a dynamic and graphical user interface instead of previous text-based searches and a client side application for storing personal preference information. In this research, we implemented the graphical personalized search manager using Yahoo web search API and widget, and demonstrated its effectiveness by performing some experiments with various query terms and representative predicates.

Key words : RIA(Rich Internet Application), Web2.0, Personalized Information Retrieval, User Interface

· This research was supported by the Ministry of I&C, Korea, under the College Information Technology Research Center Support Program, grant number IITA-2006-C1090-0603-0031.
· 이 논문은 2007 한국컴퓨터종합학술대회에서 'RIA 기술을 활용한 개인화된 웹 검색'의 제목으로 발표된 논문을 확장한 것임

† 학생회원 : 성결대학교 컴퓨터공학부
parkchala@hanmail.net
†† 정 회 원 : 성결대학교 컴퓨터공학부 교수
tshou@sungkyul.edu
††† 종신회원 : 인하대학교 산업공학과 교수
trinity@inha.ac.kr

논문접수 : 2007년 10월 2일
심사완료 : 2007년 11월 2일

정보과학회논문지: 컴퓨팅의 실제 및 레터 제13권 제6호(2007.11)

1. 서 론

기존의 웹 검색 엔진들은 사용자가 검색에 사용하였던 고급검색의 설정 내용들을 유지하거나 유연하게 변경하고 저장하는 것이 제한적이고, 사용자들이 검색엔진을 효율적으로 사용하기에 많은 제약이 있다. 또한 고급검색 설정이 텍스트기반 웹 페이지이기 때문에 사용자가 웹 서핑 시 제외하고 싶은 단어가 발생되면 다시 고급검색에서 질의를 추가하여 재검색해야 한다는 불편함이 있다. 국내 경우 고급검색 기능을 보다, 자동완성 기능이나 인기 검색어와 같은 추천단어 서비스를 제공하고 있다. 이러한 추천단어 서비스는 검색 사이트가 자체적인 가중치 평가 방식을 통해 계산된 단어를 추천하는 것이다. 가중치 평가 방식이란 웹을 유방향 그래프로 표현할 때 웹 페이지를 표현하는 웹 노드 집합과 이를 연결하는 하이퍼링크를 표현하는 웹 아크 집합에 대해 가중치를 부여하여 사용자의 질의응답 시 반영하는 방식이다[1]. 그러나 이러한 방법에서 가중치를 부여하는 기준이 사용자가 직접 평가한 것이 아닌 웹 검색엔진이 자체 분석한 결과이기 때문에 개인화된 검색을 충분히 지원해주지 못한다.

웹 2.0의 대표기술인 RIA기술은 기존 웹 어플리케이션이 갖고 있던 평면적인 표현과 순차적인 요소를 동적인 사용자 인터페이스와 Data API의 연동을 통해 저비용으로, 단일 인터페이스에서 모든 프로세스가 처리될 수 있도록 하는 기술들을 말한다. RIA기술들은 동적이고 확장 가능한 몇 가지 공용 기술들을 통해 개발되고 있다[2]. 대표기술로는 Ajax, Flash 기반의 Flex, 데스크톱 기반의 Opera와 야후 등의 Widget, 그리고 마이크로소프트사의 Windows Live Gadget등이 있다. 본 논문은 RIA기술들을 활용하여 개인 로컬 영역의 데이타뿐만 아니라 인터넷의 문서와 데이타들을 손쉽게 고급 검색할 수 있는 방법과 사용자의 의도와 요구에 맞는 결과 값을 검색할 수 있는 방향을 제시한다. 논문의 순서는 다음과 같다. 2장에서는 RIA의 기본 개념과 기술들을 알아보고, 3장에서 RIA 기술에 기반을 둔 사용자 중심의 검색구조를 제안한다. 마지막으로 4장에서 결론 및 향후 연구 방향을 제시하였다.

2. RIA 기술

웹 관련 기술발달로 개발자들은 로컬 PC 기반 어플리케이션만큼 풍부한 웹 어플리케이션을 구현할 수 있게 되었다[3].

2.1 RIA기술과 기존 어플리케이션과의 차이점

시맨틱웹의 대표기술인 RIA기술은 웹 어플리케이션으로 실행되면서 데스크톱 응용프로그램과 유사한 기능과 특징을 유지할 수 있게 하는 기술이다. Jesse James가 최초 소개한 AJAX(Asynchronous JavaScript and XML), MS의 Silverlight, Sun의 JavaFX, 최근에는 Adobe의 Air[4]등이 나오면서 RIA로 통칭되고 있으며 W3C에서도 Rich Web client로 워킹그룹[5]을 명명하고 2005년 말부터 활동이 시작되었다. 기존의 MS ActiveX Control은 OS의 호환성과 보안의 취약성이, Macromedia의 Flash는 오픈소스가 아닌 상업용 어플리케이션이라는 문제점을 갖고 있기 때문에 시맨틱웹의 개방성, 확장성, 표준화를 지원하는 Ajax기술이 가장 주목받아왔다. Ajax기술은 인터랙티브한 서비스 구현을 위하여 ActiveX의 설치를 요구하지 않는다. 이는 검증되지 않은 ActiveX를 설치함으로써 발생할 수 있는 보안문제도 해결할 수 있다. 이러한 RIA기술들의 공통점은 기존의 기술에 비해 풍부한 사용자경험(UX, User eXperience)을 보장 한다는 것이다[6]. 그림 1에서 보듯 RIA기술은 기존 웹기술들에 비해 서버측과 클라이언측 모두 동적이고 인터랙티브한 기술들을 말한다. 특히 컴파일 된 코드의 실행이 아닌 XML과 해당스크립트를 실시간에 해석하여 수행하기 때문에 사용자와의 상호작용을 더욱 긴밀하게 할 수 있다.

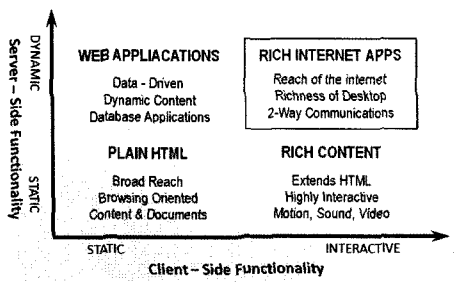

그림 1 동적이고 쌍방향적인 RIA기술

2.2 FLEX와 Laszlo

FLEX는 Adobe의 강력한 통합개발환경(IDE)을 배경으로 개발되고 있으며 사실상 표준으로 인식되는 Flash 기술을 기반으로 개발되었다. FLEX는 HTML의 한계에서 벗어나 화려하고 동적인 UI를 구현 가능하게 만든다. 그러나 무료로 제공되는 Ajax의 라이브러리나 프레임워크에 비해 오픈소스 플랫폼이 아니라는 것과 상업용 도구라는 것이 단점으로 존재한다. 또 다른 Flash 기반의 RIA기술인 Laszlo는 FLEX와는 달리 오픈 플랫폼을 지향한다. Laszlo는 LZX라는 XML을 기반으로 한 스크립트를 사용하고, 이 언어는 XML과 JavaScript를 이용하여 Flash를 생성하는 객체지향 태그기반으로 정의된다. 또한 Laszlo는 구현방식에 대해 Flash형식이나 DHTML형식을 선택할 수 있도록 하고 있다[7].

2.3 Firefox와 Silverlight

Mozilla사의 XUL(XML User Interface Language)은 XML기반 마크업 언어로서, 자사계열 브라우저에서 실행가능하며, 개발된 풍부한 라이브러리를 통하여 사용자 인터페이스 컨트롤을 기술한다. 또한 계층적 문서구조를 위하여 트리구조의 문서객체모델(DOM)을 사용한다. 대표적으로 Mozilla의 Firefox는 XUL파일과 JavaScript 그리고 CSS(Cascading Style Sheets)를 포함한 하나의 패키지로 구현되었다. 반면 마이크로소프트사는 WPF/e이란 코드네임으로 Silverlight라는 웹 프레젠테이션 기술을 개발하였는데, 이 기술은 다양한 플랫폼에서 풍부한 기능과 화려한 비주얼, 인터랙티브한 실행이 가능하도록 개발되었다[8]. 이 기술의 핵심은 XAML(eXtensible Application Markup Language)를 토대로 한 프레젠테이션기능이다. XAML은 텍스트기반인 XML이기 때문에 방화벽 환경에서도 편리하게 사용할 수 있고 JavaScript를 통하여 손쉽게 이벤트 핸들러를 작성하거나 컨텐츠의 상호작용을 가능하게 할 수 있다.

2.4 Widget과 Gadget

Widget은 Flash나 JavaScript기술등을 사용하여 일종의 Plugin과 같은 확장된 Client Application을 말한다. 기본적인 Widget의 형태는 XML을 기반으로 동적인 사용자 인터페이스를 제공하고 JavaScript와 CSS로 구성한다. 주로 Open API[9-12]에서 제공해주는 데이타를 GUI로 제공하거나 제어하는 형태로 개발되고 있다. 최근 많은 사이트들이 자사의 API를 개방함에 따라 이를 MashUp 혹은 Web Application Hybrid한 형태로도 많이 개발되고 있다. 이러한 Widget은 Opera나 Yahoo에서 활발한 개발이 이루어지고 있고, 마이크로소프트사는 Vista에서 새롭게 선보인 사이드바를 통해 Widget과 동일한 기능을 수행하는 Gadget을 관리할 수 있도록 서비스를 시작하였다.

2.5 Widget기반의 웹 검색 어플리케이션

다양한 RIA기술 중 Widget은 웹 브라우저와는 별개로 작동하면서도 클라이언트 영역에서 자유로운 프로세싱이 가능하다. Opera, Avedesk의 Widget이나 MS Vista 사이드바의 Gadget등 많은 종류의 어플리케이션이 존재하지만, Yahoo의 Widget은 Yahoo 검색 API와

유연한 연동이 가능하기 때문에 개인화된 웹 검색을 구현하고 평가하기 위한 어플리케이션으로 적합하다.

기존 Yahoo Widget의 경우 단순히 검색기능을 갖춘 검색 Widget이나 Open API를 이용하여 검색엔진을 선택하고 검색할 수 있는 선택검색 Widget은 존재하였지만 고급검색의 기능을 GUI로 제공하거나 이를 설정하여 저장 관리하는 Widget은 없다. 반면 Google에서는 검색기록이라는 베타 서비스로 사용자가 일정기간 동안 검색한 검색 질의문과 결과물들의 링크를 기억하고 관리할 수 있지만 이 정보를 사용자의 새로운 웹 검색에 반영하지는 않는다. 본 논문은 기존의 고급검색의 단점을 극복할 수 있는 RIA기술을 활용한 Widget 기반의 웹 검색 어플리케이션으로 기존의 검색 Widget과 단순한 검색기록의 저장과는 다른 개인화된 웹 검색구조를 설계하였다.

3. RIA기술을 활용한 개인화 검색시스템

3.1 검색 시스템 구조

본 논문에서 설계한 RIA를 활용한 검색시스템의 구조는 그림 2와 같다.

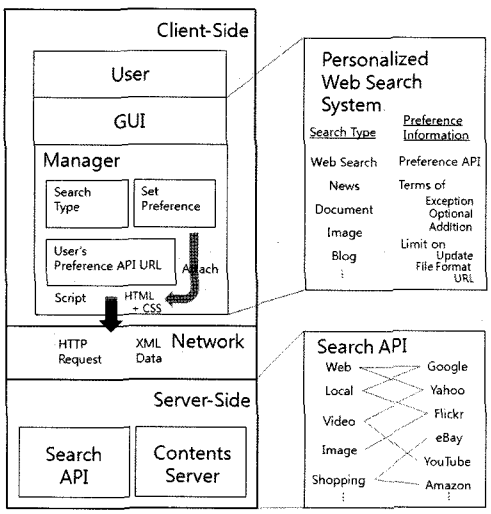

그림 2 RIA 기술을 활용한 개인화 검색시스템

• Personalized Web Search System

기존 ActiveX, NS Plug-in, Flash등의 기술들은 웹 브라우저 안으로의 활동에 제한되기 때문에, 풍부한 사용자경험을 제공하기엔 부족하고 이를 개선할 필요가 있다. Widget은 웹 브라우저와 별도로 작동하면서도 연계성을 유지할 수 있으며, 비동기적인 스크립트 언어를 사용하여 전체페이지 대신 페이지블록만을 갱신한다. 또한 동적인 GUI는 다른 클라이언트 영역에서 Text Type의 String 값이나 URL, 심지어 파일의 정보까지 관리할 수 있기 때문에 고급검색 기능을 사용자가 보다 쉽게 접근 이용할 수 있게 도와준다. 이 방법은 페이지를 갱신하는 단위를 줄임으로써, 애플리케이션의 처리 속도도 빨라지고, 가용성도 높아지며, 시각적인 효과도 더 좋아진다. 또한 개발자는 에러나 버그를 더욱 효율적으로 관리할 수 있다[13]. 기존 고급검색의 경우 사용자가 개인의도에 맞게 설정한 입력 값들이 페이지가 갱신되면 다시 재적용되지 않고 처음 디폴트 값으로 초기화 된다. 그러나 개인화 검색 관리자는 이 설정 값들을 저장 관리하여 매 검색 시 적용시켜준다. 또한 개인화 검색 관리자가 관리하는 사용자 프로파일에 의해 Public 검색엔진에서 제공하는 가중치가 부여된 검색 값의 정렬이 아닌 사용자의도에 가까운 검색결과값을 기대할 수 있게 된다. 이를 위해 관리자는 사용자 검색패턴이나 설정을 통해 선호하는 키워드와 비선호 키워드를 기억하고 검색 결과값을 필터링할 때 반영한다.

• Search API

API란 Application Programming Interface의 약어로 운영체제나 특정사이트가 제공하는 기능들을 활용하고 데이타를 제어할 수 있도록 만든 인터페이스를 말한다. 최근 Google이나 Amazon, E-bay등 많은 기업들이 자사의 API 소스를 공개하면서 누구나 RIA기술을 활용하여 제공되는 Open API와 조합을 통해 자신만의 어플리케이션을 제작할 수 있다. 따라서 RIA 기술은 자신이 선호하는 API를 선택할 수도 있으며 Open API를 사용하여 얻을 수 있는 데이타를 통해 더욱 가볍고 확장성이 높은 어플리케이션으로 구현할 수 있다[6].

3.2 질의처리과정

본 논문에서 설계한 시스템을 평가하기 위하여 Yahoo 검색 API에서 제공하는 고급검색과 Yahoo Widget engine[14]을 이용하여 구현한 기본처리과정은 그림 3과 같다. 질의처리과정을 위해 사용자에게 입력받은 SearchText 값과, 사용자가 설정하고 입력한 검색어를 기준으로 만든 SearchList, 이 SearchList와 사용자가 설정한값들을 바탕으로 한 PSMD(Personalized Search Manager data, 개인화 검색 관리자 데이타)를 객체로 생성한다. 2행에서 사용자로부터 검색어를 입력받고, 3행에서는 입력받은 값을 기존에 사용하였거나 설정되어있는 단어인지 SearchList에서 비교검색을 실행하고, 존재하지 않은 경우에 4행에서 새롭게 SearchList에 기록을 남기고 5행에서는 입력받은 SearchText를 API로 전달한다. 본 논문에서는 Yahoo Open API를 사용하였기 때문에 야후 검색 API URL[10]에 SearchText값만 추가하여 URL을 요청한다.

예를 들어 사용자가 'Apple'을 입력하면 검색어는 SearchText에 삽입되고 입력받은 검색어가 기존에 설정이 되었거나 검색한 적은 있는지 SearchList를 통해 비교한다. 비교결과 해당사항이 없을 경우 새롭게 SearchList

```
1   var SearchText, SearchList, PSMD
2   get(SearchText)
3   If(SearchList !=""){
4       SearchList = add SearchText
5       OpenURL("OpenAPI URL + SearchText)
6       }
7   Else {
8       SearchList = join SearchText
9       Run Personalized_Search_Manager()
10        If(Searchlist.Function == "None of these word")
11              put PSMD = "%2D" + SearchText
12      else If(Searchlist.Function == "Any of these word")
13              put PSMD = "%2B" + SearchText
14                      .....
15      get(PSMD)
16      OpenURL("OpenAPI URL + SearchText + PSMD)
17      }
```

그림 3 질의 처리 과정

에 추가하고 바로 검색어를 검색 API로 보내게 된다. 그러나 그렇지 않은 경우 해당단어를 개인화 검색 관리자가 참고하고 학습할 수 있도록 해당 값을 결합한다.

3.3 시스템의 구현

야후 Widget은 Widget엔진에서 제공하는 프레임워크와 라이브러리를 통하여 개발할 수 있다. 모델구현을 위해 야후에서 제공하는 웹 문서 검색 API와 검색 Widget을 바탕으로 구현하였다. 본 논문에서 제안하는 개인화된 검색 결과를 위하여 버튼과 메뉴를 추가적으로 구성하여 그림 4와 같이 구성하였다.

그림 4 개인화 화면구성

그림 3에서 설계한 처리과정을 구현하기 위해 기존 고급검색을 GUI로 구현하고, 보다 직관적인 인터페이스를 위해 타이핑, 클릭, Drag & Drop, hovering등을 통한 다양한 이벤트 등으로 입력과 수정, 관리를 할 수 있도록 그림 5와 같이 설계하였다. 고급검색에서 해당항목 입력 값을 바탕으로 Search List가 작성되었다 하더라도 여러 가지 제한 값들을 통해 검색결과를 더욱 세밀하게 필터링할 수 있다. 이러한 기능을 지원하기 위하여 날짜, 사이트, 파일포맷, 지역등을 제한할 수 있는 체크, 스크롤, 텍스트박스를 그림 6처럼 구성하였다.

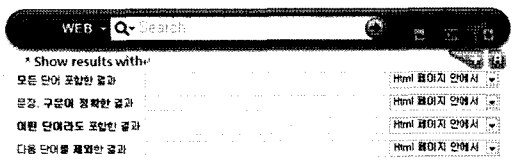

그림 5 SearchList에 단어를 정의하는 화면

각각 그림 5와 그림 6에서 GUI로 구현한 고급검색 및 제한기능들은 사용자가 설정함과 동시에 SearchList에 저장되고, 개인화 검색 관리자에 의해 API가 이해할 수 있는 연산자로 전환된다. 최종적으로 개인화 검색 관리자는 해당기능 연산자와 SearchList의 값들을 조합하여 API가 이해할 수 있는 데이타(PSMD)로 결과값을 생성한다.

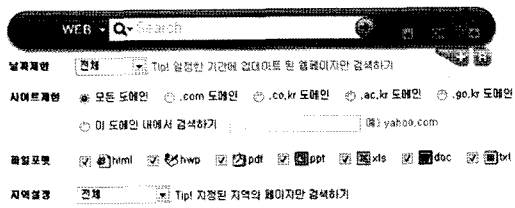

그림 6 검색 문서의 세부 항목을 제한하는 기능

Widget은 인터넷의 연결이나 웹 브라우저와는 별개로 작동하지만 그림 7처럼 사용자가 웹 서핑 도중 리스트에 추가하고 싶은 단어가 발생하였을 경우 해당단어를 선택하여 원하는 휴지통에 끌어다 놓기만 하여도 자동으로 리스트에 추가할 수 있는 직관적인 UI를 제공한다.

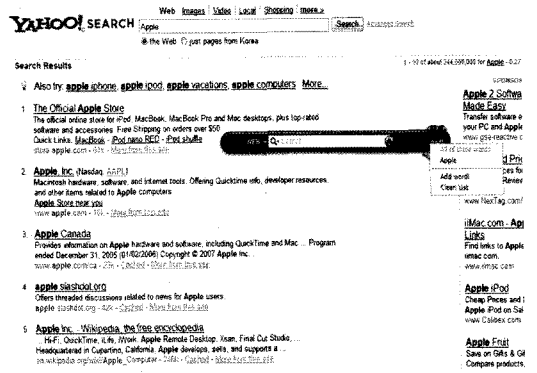

그림 7 제외단어를 Drag & Drop으로 추가하는 모습

3.4 시스템 평가

본 논문에서 제안한 구조가 사용자에게 얼마나 풍부한 사용자 경험을 제공하는지 평가하기 위하여 사용자 시나리오를 작성하고 이를 기반으로 평가하였다. 시나리오는 다음과 같다.

> *"뉴스를 중요시하는 사용자는 'Apple'이라는 단어를 검색어에 입력한다. 검색 값 중에 'Store'라는 단어를 제외하고, 3개월 이내의 문서 중 'IT'라는 단어는 선호하는 단어로 우선적으로 정렬된 페이지를 보고 싶다. 검색결과 사용자가 찾던 웹 문서의 주소는 "Apple.slash dot.org" 이다."*

일반검색을 사용하면 사용자는 'apple'이란 짧은 입력값만 타이핑한 반면, 사용자가 찾고자 하는 페이지까지 도달하는데 5개의 불필요한 상위랭크 페이지들을 검색해봐야 한다. 반면 고급검색을 이용하게 되면 사용자의 의도에 맞는 페이지를 더욱 상위에 랭크하여 불필요한

표 1 야후 검색 엔진을 이용한 평가 결과

	Rank	Input	Page	Key in	Mouse
Normal	6	**apple**	327,000,000	5	None
Advanced Search	2	apple, news, IT, store	127,000,000	18	5Clicks
Using RIA	1	**apple**	91,800,000	5	2 Clicks 4 Drag & Drop

페이지 검색을 줄일 수 있지만 사용자로부터 많은 입력 값들을 요구하게 된다. 표 1에서 보는 것처럼 고급검색의 경우 날짜와 파일형식을 설정하는 시간을 포함하여 18번의 키보드 입력과 5번의 마우스 조작이 소요된다. 그러나 입력한 설정 값들은 사용자가 새로운 검색어를 입력하게 되면 재적용되지 않기 때문에, 새롭게 입력 값을 설정해야 한다. 기존의 평면적이고 순차적인 고급검색의 설정을 본 논문에서 구현한 개인화 검색 관리자를 적용하게 되면 직관적이고 동적인 UI를 통해 선택하여 끌어다 놓기(Drag & Drop)만으로 간단한 설정을 할 수 있게 된다. 이와 같은 방법은 사용자가 이미 확인한 제목의 페이지나 중복된 링크를 개인화 검색 관리자에 끌어다 놓는 것만으로도 검색 결과에서 제외할 수 있다. 또한 페이지 단위의 전송이 아닌 사용자가 변경한 부분의 스크립트만을 재전송함으로써 새로운 검색어에도 유연하게 재적용 가능하여 사용자의 의도에 맞는 검색결과를 반환하여준다.

기존의 고급검색은 많은 옵션을 제공하고 있지만 각 항목에 대한 우선순위를 설정할 수는 없다. 그러나 인터랙티브한 RIA기술을 활용하면 고급검색 옵션에 해당되는 각각의 설정 값들을 우선순위와 혼합하여 설정할 수 있다. 이를 평가하기 위하여 선행시나리오와 유사하게 3개의 고급검색 옵션을 적용하여 무작위 100개의 검색어를 검색한 결과 다음과 같은 평균 페이지양을 보였다. 고급검색으로 검색된 페이지양보다 개인화 검색 관리자를 통한 검색이 더욱 감소한 평균페이지양을 보였다. 반면 30개의 임의단어를 순서에 상관없이 고급검색 설정을 통해 검색한 경우 검색에 소요되는 시간에 크게 영향을 주지 않지만, 개인화 검색의 경우 설정 값에 해당하는 우선순위를 무작위로 혼합할수록 검색에 소요되는 시간이 증가하였다.

4. 결론 및 향후 연구과제

방대한 자료 속에서 보다 개인의 선호와 의도에 맞는 검색 결과 값을 도출 하기 위하여 본 논문은 RIA기술과 사용자의 키워드 선호도에 따른 필터링을 통해 개인화된 검색 구조를 제안하였다. RIA기술들은 동적인 UI를 제공하면서도 가볍고, 클라이언트 어플리케이션과 같은 기능들을 수행할 수 있기 때문에 기존의 고급검색 이용 시 불편하였던 점을 개선에 적용될 수 있다. 또한 개인화 검색 관리자는 사용자의 프로파일을 통해 선호하는 키워드와 비

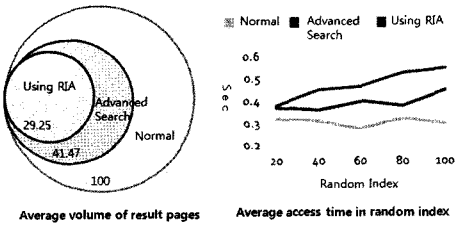

그림 8 평균 페이지양의 감소율과 검색소요시간

선호 키워드를 저장하고, 이를 활용하여 검색 결과값을 필터링할 수 있으며, 파일형식이나 업데이트 날짜 등을 제한하여 결과값을 더욱 세부적으로 관리할 수 있음을 보였다.

추후 연구과제로서 고객 맞춤 서비스를 제공하기 위하여 각각의 검색 사이트는 서버에 사용자의 프로파일을 저장관리하고 이를 통해 본 논문에서 제안한 개인화된 검색을 구현할 수도 있다. 즉, 이러한 RIA 어플리케이션이 반드시 클라이언트 영역에서만 국한되지 않고 서버 측에 적용될 수 있는 여지가 많다는 점이다. 또한 본 논문에서 제시한 키워드 선호도에 따른 필터링보다 더욱 개인화 된 검색을 위하여 개인화 검색 관리자가 사용자 검색어 패턴을 인식하고 스스로 학습하여 가중치를 계산한 후 결과 값을 부여해주는 방법도 더욱 연구해야 할 과제이다. 이런 연구를 통해 가공되지 않은 수많은 데이타들을 사용자에게 의미가 있는 정보로 제공하는 데 더욱 도움을 줄 수 있으며, 검색 서비스의 고객 맞춤화에 기여할 수 있을 것이다.

참 고 문 헌

[1] O. Etzioni et al., "Methods for Domain Independent Information Extraction from the web: An Experimental Comparison," In Proc. AAAI, pp. 391-398, 2004.
[2] J. Yu, B. Benatallah, F. Casati, R. Saint-Paul, "XUPClient - A Thin Client for Rich Internet Applications," In Proc. WISE, pp. 524-535, 2006.
[3] J. J. Garrett, The Elements of User Experience: User-Centered Design for the Web, New Riders Publishing, 2003.
[4] M. Chambers, labs.adobe.com/wiki/index.php/Apollo, 2007.
[5] Rich Web Clients, www.w3.org/2006/rwc/, 2006.
[6] J. Musser, T. O'Reilly, O'Reilly Radar Team, Web 2.0 Report, O'Reilly Media, 2006.
[7] V.V.Gadge,http://www-128.ibm.com/developerworks/library/wa-richiapp, 2006.
[8] L. Moroney, msdn2.microsoft.com/en-us/library/ bb190632.aspx, 2006.
[9] Flickr Open API, "http://flickr.com/services/api/"
[10] Yahoo Open API, "http://developer.yahoo.com/"
[11] Google Open API, "http://code.google.com/"
[12] Naver Open API, "http://openapi.naver.com/"
[13] A. Bozzon, "Conceptual modeling and code generation for rich internet applications", In Proc. ICWE, pp. 353-360, 2006.
[14] Yahoo! Widgets, http://widget.yahoo.com/