웹상에서 정보시각화도구 및 줌 브라우저의 구현

정 영 아[†] · 김 응 곤[††] · 한 승 조[†††]

요 약

본 논문에서는 웹상에서 검색결과를 시각화하기 위한 시각화 도구와 줌 브라우저(Zoom Browser)를 구현하였다. 시각화 도구는 검색엔진을 이용하여 검색한 결과를 아이콘으로 표현하고 나선형으로 배치하여 시각화함으로써 사용자가 필요로 하는 정보를 쉽게 찾을 수 있도록 해준다. 줌 브라우저는 많은 페이지로 구성된 긴 문서를 시각화하기 위하여 Focus+Context 기법을 이용하여 문서의 전체적인 윤곽을 파악하면서 관심있는 부분을 자세히 볼 수 있도록 한다. 본 시각화 도구는 자바로 구현되었으며, 웹상에서 자바를 지원하는 모든 브라우저에서 사용할 수 있다.

Development of a Visualization Tool and a Zoom Browser on the WWW

Young-A Jeong[†] · Eung-Kon Kim[††] · Seung-Jo Han[†††]

ABSTRACT

This paper implements a visualization tool and a zoom browser for visualizing the search results on the WWW. The visualization tool allows the user to find interested information through a display window as scattering the icons. The zoom browser uses a Focus+Context technique to visualize large documents with many pages. It offers an overview of a document, and gives users instant access to any part of interest.

These tools were implemented using Java and can be used by all browsers which support Java on the WWW.

1. 서 론

요즈음 인터넷을 통하여 많은 정보를 얻고 있다. 우리 주위에 정보가 폭발적으로 늘어나 홍수를 이루는데, 많은 정보 가운데 필요한 정보를 신속하고 정확하게 얻는 것이 무엇보다도 중요하다. 다행히 검색엔진이 개발되어 웹상에서 원하는 정보를 쉽게 찾을 수 있도록 해주고 있다.

그러나 대부분의 검색엔진은 문자중심이며, 그 결과는 스크롤되는 페이지로 나타나게 된다[1]. 문자는 그림보다 추상적이어서 많은 인지적 노력을 필요로 하며, 사용자가 쉽게 접근할 수가 없다. 뿐만 아니라 결과가 길어지면 여러 페이지에 걸쳐서 순차적으로 출력되기 때문에 앞 뒤 임의의 페이지로 넘어가기가 어렵다[2,3].

이러한 표현에서 문제점은 사용자가 필요로 하는 정보가 문서전체적인 경우는 문제가 되지 않으나, 특별히 문서중간의 어느 일정부분이고, 이미 그 부분을 지나쳐 왔을 경우 원하는 부분으로 바로 건너뛰기가 또한 쉽지 않다. 전체 문맥을 쉽게 파악하면서 동시에 어느 한 부분만을 집중해서 보고싶을 경우도 마찬가지이다. 컴퓨터 화면크기의 제약 때문에 나타나는 이러한 문제점들을 극복하기 위해서 그 동안 많은 정보표현의 시각화 기술들에 관한 논문들이 발표되었다[4,5,6]. Focus+

† 준 회 원 : 순천대학교 대학원 컴퓨터과학과
†† 정 회 원 : 순천대학교 컴퓨터과학과 교수
††† 정 회 원 : 조선대학교 전자정보통신공학부 교수
논문접수 : 1998년 10월 8일, 심사완료 : 1999년 3월 3일

Context 기법은 전체 정보(Context)의 개략적인 내용을 한 눈에 파악하면서 동시에 사용자가 원하는 부분(Focus)을 자세히 볼 수 있도록 하는 것이다[7].

따라서 본 논문에서는 새로운 시각화 도구를 만들어 사용자에게 편리한 인터페이스를 제공하고, 검색엔진으로부터 나온 검색결과를 화면에 시각화하여 보여줌으로써 원하는 정보를 쉽게 찾기 위한 시각화 도구를 개발하고, 검색된 문서의 내용을 효과적으로 시각화하기 위한 Focus+Context 기법을 이용한 줌 브라우저(Zoom Browser)를 구현한다.

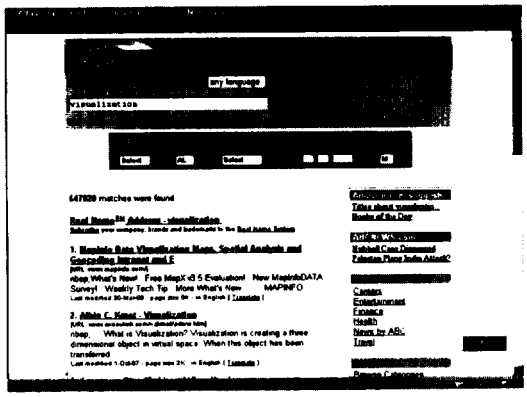
웹상에서 클라이언트가 입력한 검색어는 서버로 들어가서 검색엔진을 통한 결과가 시각화 도구를 거쳐 HTML과 자바를 사용해 클라이언트의 화면에 시각화되어 나타나고 검색결과의 문서들 중 사용자가 선택한 문서에 대하여 관심있는 부분을 확대축소할 수 있고, 문서의 전체 윤곽을 파악할 수 있게 된다. 본 시각화 도구는 알타비스타의 검색결과를 이용하고, 자바를 사용하여 클라이언트/서버 통신 환경을 구축하였다.

본 논문의 2장에서는 기존 웹 검색결과의 문제점과 이를 해결하기 위한 관련연구에 대하여 기술한다. 3장에서는 본 논문에서 구현한 시각화 도구의 구조와 기존의 시스템과의 비교를 살펴보고, 4장에서는 시각화 시스템의 설계에 대하여, 5장에서는 자바를 이용하여 구현한 검색결과의 시각화 도구와 줌 브라우저에 대하여 논하며, 6장에서는 결론과 향후 연구 과제에 대하여 기술한다.

2. 관련 연구

오늘날 대부분의 검색엔진은 검색결과를 문자 중심으로 출력하므로 다음과 같은 몇 가지 문제점이 발생한다[3]. 첫째, 문자중심의 출력은 상징적이고 추상화된 개념이기 때문에 사용자가 육안으로 검색결과를 쉽고 자연스럽게 관찰할 수가 없다. 둘째, 스크롤되는 문장들로 출력되므로 사용자는 출력결과들을 통합해야 하는 인지적인 부담을 갖게 한다. 셋째, 사용자는 전체 검색결과를 보지 못하고 화면에 출력된 부분적인 내용만 볼 수 있기 때문에 관심부분을 쉽게 찾거나 임의의 위치로 즉시 이동할 수 없다.

(그림 1)은 알타비스타의 검색결과로 위에서 언급한 문제점들을 드러내고 있다. 문자열이 나열되어서 시각적으로 한 눈에 들어오지 않고, 10개씩 부분적으로 출력되면서 스크롤 되어버리기 때문에 전체를 보거나 다시 임의의 항목으로 이동하는 것이 어렵다는 것을 알 수 있다.



(그림 1) 알타비스타의 검색결과
(Fig. 1) Search Results using Altavista

이러한 문제를 해결하기 위하여 웹상에서의 정보 시각화에 관한 연구가 많이 수행되었다[8]. 웹에서의 방향 상실에의 문제점을 해결하기 위한 다양한 도구들이 개발되어 왔는데 그 중에서는 브라우저나 렌즈와 같은 항해도구, 히스토리 리스트나 책갈피, 필터와 같은 수동적인 프로세스, 에이전트와 같은 능동적인 프로세스 등이 있다. Pittsburg 대학에서 개발한 정보시각화 도구인 CASCADE(Computer Augmented Support for Collaborative Authoring and Document Editing)는 문서들의 협력작업을 지원하는 항해도구와 수동적, 능동적 프로세스를 하나로 통합하여 계층적 구조의 개요를 통한 정보검색을 위한 시각적 인터페이스를 제공해 준다. WebBook은 사용자가 웹 활동에 대해서 바인딩 작업을 통해서 조직적으로 체계화할 수 있도록 구성되었다. 또한 Navigational View Builder는 정보공간의 유용한 시각화를 위하여 필터링과 클러스터링(clustering)으로 효과적인 개요를 제공해 주고 있다. Mukherjea 등은 웹상에서 검색결과를 VRML 코드를 이용하여 3차원으로 표현하는 AMORE라는 검색엔진을 개발하였으나, 문서를 인덱싱하고 검색하기 위하여 Harvest라는 별도의 소프트웨어와 VRML 코드 발생기를 필요로 하며, 검색결과를 시각화하는데 그친 반면, 시각화된 화면에서 선택한 문서를 줌 브라우징할 수 있는 기능은 제공하지 않고 있다[6]. 이외에도 웹상에서

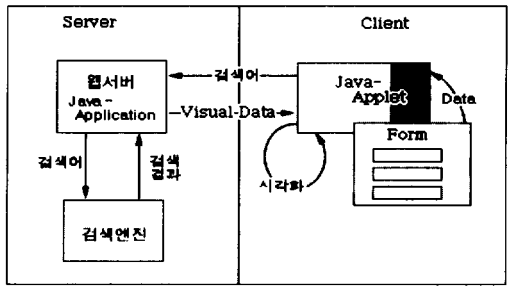
의 정보 시각화에 대한 연구가 끊임없이 수행되고 있으며, 사용자를 고려한 많은 도구들이 개발되고 있다. 하지만 대부분의 도구들은 플랫폼에 호환성이 없다[8]. 즉, 현재 웹 서버와 클라이언트를 위한 환경은 다양하지만 그러한 환경에서 모두 호환성있게 수행될 수 있는 도구가 필요하다. 노원빈 등은 기존의 도구들의 문제점인 플랫폼 비호환성을 해결하기 위한 정보시각화 도구를 개발하였는데 계층적인 문서 구조에 적합하고 파일들의 크기와 종류를 막대 그래프만으로 시각화하는데 역점을 두었다[8].

또한 긴 문서의 내용을 시각화하는 기능을 갖는 브라우저가 개발되고 있다. Brown 등은 HTML의 HEAD 태그에 지퍼(zipper)라 불리는 아이콘을 첨부하고, 사용자가 이 지퍼를 클릭함으로써 문서의 각 부분을 볼 수 있도록 구현했다[4]. 지퍼를 구현하기 위한 세 가지 방법이 제시되었다. 첫번째 방법은 지퍼를 삽입한 custom-built 웹 브라우저를 사용하는 것이고, 두 번째 방법은 지퍼를 웹 프록시(proxy)에 의해서 HTML 문서에 삽입하고, 지퍼의 변환된 상태를 나타내 주는 수정된 HTML을 생성한다. 마지막으로 세 번째 방법은 Javascript를 이용하여 지퍼를 구현하는 것이다. 이 지퍼에 의한 방법은 복잡한 계층구조를 갖는 문서에서는 효과적이나 섹션(section)이 긴 문서의 경우에는 적합하지 못하다.

또한, Holmquist와 Ahlberg는 정보의 양이 많은 경우에 전체 문맥을 파악하면서 동시에 특정부분을 자세히 알고 싶은 경우 임의접근이 가능한 Flip Zooming 기법을 제안하였다[5]. 그러나 이 기법에서는 문자 정보와 그림 정보를 같은 화면에 표현하지 못하고 있으며, 웹 브라우저에 적용하는 것을 향후 과제로 제시하고 있다.

3. 본 논문의 시각화 시스템 구조

본 논문에서는 검색엔진을 이용하여 검색한 결과를 시각화하는 도구와 긴 문서의 개략적인 내용을 한 눈에 파악하면서 원하는 부분을 자세히 볼 수 있도록 하는 줌 브라우저를 구현하였다[13,14]. 사용자는 웹상에서 검색어를 입력하고 검색 결과는 본 시각화 도구를 통하여 시각화되고 선택한 문서를 줌 브라우저를 통하여 볼 수 있다. (그림 2)는 본 논문에서 구현한 시각화 도구의 구조를 나타내고 있다. 입력한 검색어가 웹서버의 자바 Application을 통하여 검색엔진에 보내진 후, 얻어진 결과를 시각화하여 클라이언트의 화면에 나타내 준다.



(그림 2) 본 논문의 시각화 도구의 구조
(Fig. 2) Visualization Tool Architecture

클라이언트는 질의어를 입력받아 자바 Applet에서 검색식을 만들어 서버의 자바 Application을 통해 검색엔진으로 보내어 그 결과를 받아온다. 여기서 검색결과를 파싱하고 분석하여 시각화 데이타로 바꿔주며 클라이언트에서는 HTML과 자바를 이용하여 구현한 시각화 도구를 통해 원하는 결과를 시각화된 화면으로 볼 수 있게 된다. 검색결과가 시각화된 화면에서 선택한 문서에 대하여 줌 브라우징의 기능을 제공한다.

현재의 대부분의 검색엔진들은 검색결과를 문자중심으로 출력하므로 검색결과의 정보량이 많을수록 문서들을 스크롤하여 찾아가야 하므로 전체적인 검색결과를 보지 못하고 화면에 출력된 부분적인 내용만 볼 수 있기 때문에 관심부분을 쉽게 찾거나 임의의 위치로 즉시 이동할 수 없다.

따라서 많은 양의 검색 결과를 한정된 화면에 표현하기 위한 시각화 도구가 필요한데, 최근에 개발된 AMORE 시스템은 자체의 검색엔진에서 질의어를 인덱싱하고 검색하기 위하여 Harvest라는 소프트웨어를 사용하고, 검색결과를 시각화하기 위하여 각각의 문서를 영상으로 표현할 수 있는 VRML 코드로 변환해주는 VRML 코드 발생기를 Perl 프로그램을 통하여 구현하였다[6].

반면, 본 논문에서는 기존의 어떠한 검색엔진을 이용한 검색 결과에 대하여 시각화가 가능하며, 자바를 이용하여 구현하므로 자바를 지원하는 어떠한 웹 브라우저에서도 실행이 가능하다. 또한 기존의 시스템에서는 검색결과를 시각화하는데 그친 반면, 본 논문에서는 시각화 화면에서 사용자가 선택한 문서에 대하여 줌 브라우징할 수 있는 기능도 제공한다. 사용자가 선택한 긴 문서의 개략적인 내용을 한 눈에 파악하면서 원하는 부분을 자세히 볼 수 있도록 한다.

4. 설 계

본 논문에서는 검색엔진을 이용하여 검색한 결과를 시각화하는 도구와 긴 문서의 개략적인 내용을 한 눈에 파악하면서 원하는 부분을 자세히 볼 수 있도록 하는 줌 브라우저를 자바를 이용하여 클라이언트/서버 환경에서 구현하였다. 검색결과의 시각화 도구와 줌 브라우저를 포함하는 시각화 시스템의 구조를 (그림 2)와 같이 설계하였다[13,14].

기존의 시각화 시스템[6]이 검색결과를 시각화하는데 그친 반면, 본 시스템에서는 검색결과를 시각화할 뿐만 아니라 시각화된 화면에서 선택한 문서를 줌 브라우징할 수 있는 기능도 추가한다.

많은 양의 검색결과를 한정된 화면에 표시하기 위하여 각각의 문서를 아이콘으로 나타내어 나선형으로 배치하여 원하는 문서를 쉽고 빠르게 찾을 수 있도록 하였으며, 선택한 문서가 있는 웹 사이트로 즉시 이동할 수 있도록 하였다.

검색엔진으로부터 검색된 결과를 시각화 도구를 통하여 각 문서에 대한 아이콘을 나선형으로 배치하여 시각화한 후, 사용자가 원하는 문서를 클릭하게 되면 그 문서에 대한 전체적인 윤곽을 파악하면서 원하는 부분을 자세하게 볼 수 있도록 하는 줌 브라우저를 웹 브라우저에서 수행할 수 있도록 한다.

클라이언트에서 입력된 검색어는 서버로 넘겨지게 되고, 서버에서는 검색엔진인 알타비스타의 검색식에 맞추어 URL을 생성한 후 검색하게 된다. 이렇게 검색된 검색결과를 다시 클라이언트에서 받아들여 HTML 문서를 분석한다.

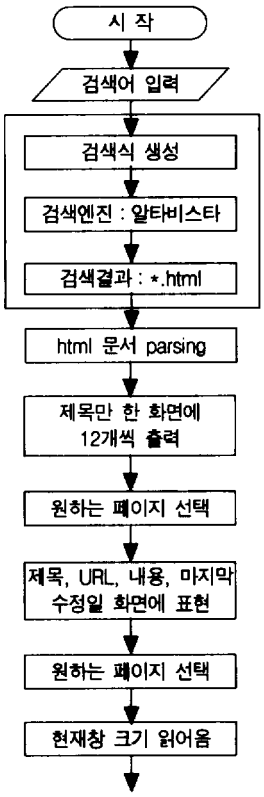
만저 제목만을 한 화면에 12개의 페이지로 나누어서 출력하도록 한 후 사용자가 원하는 페이지를 선택하면 그 페이지의 자세한 정보인 제목, URL, 내용, 마지막 수정일 등이 읽을 수 있는 크기로 확대되어 나타난다. 계속해서 사용자는 찾고자 하는 문서를 선택하고, 줌 브라우저에서는 문서의 길이가 현재 창보다 크다면 이 데이터를 여러 페이지로 나누어서 표현해 준다. 각 페이지를 자세히 읽어보기 위해서 확대하고자 할 때 사용자는 해당 페이지를 클릭하면 된다.

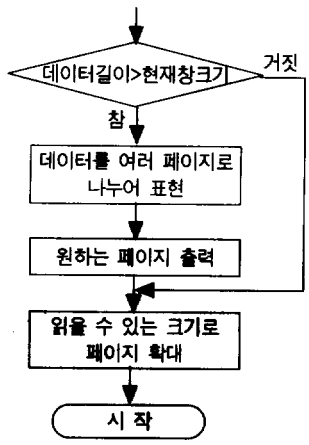
(그림 3)은 클라이언트/서버환경을 토대로 구현된 줌 브라우저의 전체 구현 과정을 나타내는 개략적인 순서도이다.

설계한 검색결과의 시각화 화면을 웹 브라우저를 통하여 볼 수 있도록 자바를 이용하여 클라이언트/서버 환경에서 구현하였다. 자바는 CGI와 달리 서버에 부담을 주지 않고, 어느 플랫폼이든 같은 결과를 보여준다. 자바 Applet은 인터넷 웹브라우저 상에서 실행되는 자바 프로그램이다. 따라서 Applet은 상당한 보안 문제를 안고 있다[11,12].

자바 Application으로 임의의 URL에 접속하여 데이타를 받아오는 것은 간단하지만 이것을 웹으로 옮기는 데에는 쉽지 않은 문제가 발생한다.

Applet은 자신이 실행된 서버이외의 네트워크로는 연결이 불가능하다. 이는 검색엔진 같은 네트워킹과 밀접한 관계가 있는 프로그램에서는 큰 문제가 아닐 수 없다. 다른 방법으로 검색 부분을 Application으로, 검색결과의 표시를 Applet으로 작성하여 Apolet에서 Application을 불러 사용할 수 있으나 이것 역시 보안 문제가 발생한다. 결국 클라이언트/서버 구조를 만들어 구현하였다. 클라이언트는 브라우저와 밀접한 관계가 있으므로 Applet으로 작성하고, 서버는 네트워킹이 가능한 Application으로 개발하였다.





(그림 3) 순서도
(Fig. 3) Flowchart

특정 포트를 사용하는 Socket을 제작하고, 그에 따라 클라이언트와 서버를 제작한다. 그리고 멀티 유저 환경을 위하여 서버 프로그램에서 검색 루틴을 따로 분리시킨다. 서버는 다음과 같이 항상 클라이언트의 접속을 기다린다.

```
ServerSocket ss=newServerSocket(2501)
        .....
Socket soc=ss.accept();
```

클라이언트의 접속요청이 들어오면 서버는 soc라는 Socket을 생성하여 클라이언트와 통신할 수 있도록 해준다. 클라이언트는 다음과 같이 서버에 접속요청을 한다.

```
Socket clientSocket = new Socket
(getCodeBase().getHostName(), 2501);
```

클라이언트는 Applet으로 작성된 코드이므로 서버 이외의 주소는 허용하지 않는다. 서버를 어디에 설치하더라도 작동할 수 있도록 getCodeBase()를 사용하여 서버의 주소를 알아낸다.

서버와 클라이언트간의 통신루틴인 Socket이라는 다리를 만든다. 이제는 이 다리를 건너기 위한 교통수단을 만들어야 한다. Socket은 자신을 입·출력 매개체로 삼을 수 있는 방법을 제공해 준다. 다음과 같이 입·출력 스트림으로 자료를 보냄으로써 클라이언트/서버는 대화를 할 수 있게 된다.

```
InputStream is =
        new InputStream(Socket.getInputStream());
OutputStream os =
        new OutputStream(Socket.getOutputStream());
```

검색된 결과는 이 입·출력 스트림을 통하여 서버에서 클라이언트로 보내어 지고, 클라이언트는 이 자료를 바탕으로 시각화하여 화면에 보여준다.

5. 구현 결과

5.1 클라이언트와 서버간의 네트워킹

주로 사용되는 검색엔진인 알타비스타를 사용하여 검색된 결과를 시각화하는 본 논문에서는 보통 일반적으로 사용되는 통신 가능한 PC 두 대를 클라이언트와 서버로 사용할 수 있도록 하였으며, 둘 사이의 통신은 자바에서 제공하는 소켓과 서버소켓을 사용하여 클라이언트/서버 환경을 구축하였다.

자바에서 제공하는 네트워킹 클래스로서 java.net.Socket과 java.net.ServerSocket 클래스가 있다. 소켓 클래스는 클라이언트가 서버에게 요청을 하기 위해 필요한 소켓을 생성하며, 이것은 신뢰할 수 있는 순차적 스트림(stream)을 제공한다. 여기에서 순차적 스트림이 의미하는 것은 데이터를 기록한 것과 같은 순서로 상대방에게 전달된다는 것이다.

다음은 클라이언트에서 소켓을 생성하여 지정된 호스트 이름과 포트에(2502 포트)에 접속한다.

```
s = new Socket(getCodeBase().getHost(), 2502);
```

서버소켓 클래스는 어떤 소켓 타입으로 통신할 것인지를 나타낸다. 서버소켓은 자신의 accept() 메소드가 호출될 때 주어진 포트에서 접속 요청을 듣게 된다. 서버소켓은 소켓 객체가 하는 것과 동일한 연결지향의 순차적 스트림 프로토콜을 제공한다. 일단 접속이 설정되면 accept() 메소드가 원격의 상대방과 대화할 소켓객체를 반환할 것이다.

다음의 서버소켓 클래스는 서버측에서 클라이언트의 요청이 들어오는지를 감시하기 위한 소켓(ss)을 생성한다.

```
ServerSocket ss = (ServerSocket)null;
                    ⋮
ss =    new ServerSocket(2502);
                    ⋮
soc = ss.accept();
createSocketThread();
soc.close();
```

위와 같은 방법으로 클라이언트/서버 환경이 구축되며 이를 바탕으로 검색하고자하는 내용이 클라이언트에서의 검색어 입력을 통해 서버로 넘어가면 클라이언트와 통신 가능해진 서버는 해당 검색어를 입력받아 알타비스타에서 검색하게 된다.

알타비스타에서. 검색하기 위해서는 입력된 검색어를 알타비스타의 검색식에 맞추어 주어야 한다. 검색식은 알타비스타에서 검색할 때 사용하는 URL에 해당 검색어를 첨가한다.

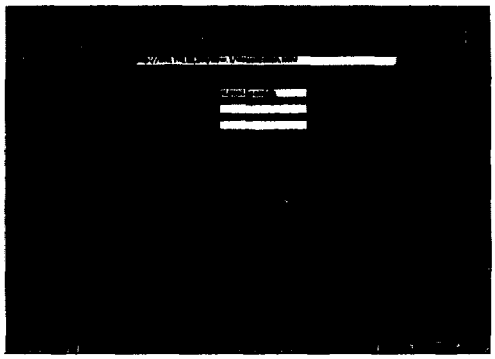
다음은 검색어를 알타비스타의 검색식에 맞추어 검색하는 알고리즘이다.

<서버에서 알타비스타로 연결하여 검색하는 알고리즘>

```
public static void start (String query) {
                    ⋮
string url = "http://www.altavista.digital.com/
query?&opt=on&text=on&pg=q&enc=iso88591&q=";
 for(int I=0; I<100; I+=10)
      fetch(url+query+"&stq="+I, I);
}
public static int fetch (String urls, int page) {
      URL url = null;
      DataInputStream dis = null;
      String sl = new String();
      try{
         url = new URL(urls);
         URLConnection uc = url.openConnection();
         dis = new DataInputStram(uc.getInputSt-
                    ream());
      }
         ⋮
}
```

5.2 검색결과의 시각화 도구

본 논문에서는 (그림 1)과 같은 형태로 출력되는 검색엔진으로부터 검색된 결과를 (그림 4)와 같이 시각화하여 사용자에게 편리한 인터페이스를 제공함으로써 원하는 정보를 쉽게 찾을 수 있는 시각화 도구를 자바를 이용하여 개발하였다. 문자의 틀을 벗어나 도형으로 보여주므로, 사용자에게 시각적 부담을 덜어주고 색상이나 명암 정보를 통해 검색 결과를 분류할 수 있으며, 많은 정보를 하나의 화면에 보여줄 수 있다.



(그림 4) 시각화 도구를 사용한 검색 결과
(Fig. 4) Search Results using the Visualization Tool

입력 창에 검색어를 입력하면 검색결과의 항목은 자료의 우선 순위에 따라 나선형으로 번호가 붙으며, 100개가 출력된다. 지금까지 문장을 스크롤시키는 기존의 방법은 화면의 크기에 따라 디스플레이하는 항목의 수가 제약을 받았으나 여기서는 조그마한 아이콘을 써서 더 많은 것을 한 화면에 보여주고 있다.

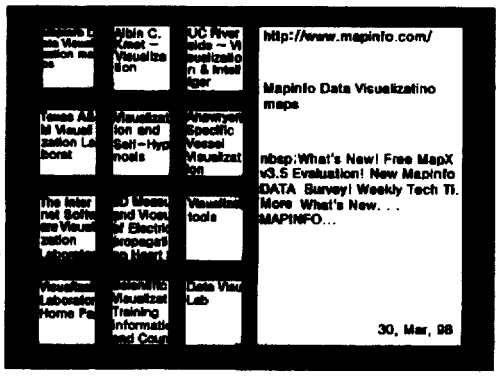
· 마우스의 포인터가 각 항목위로 지나가면 그 항목에 대한 URL, 제목, 내용, 날짜, 크기 등 자세한 내용이 출력된다. 이때 마우스를 클릭하면 바로 그 URL의 사이트로 갈 수 있다.

5.3. 줌 브라우저

본 논문에서는 검색엔진으로부터 검색된 결과를 (그림 4)와 같이 시각화 도구를 통하여 시각화한 후, 사용자가 원하는 문서를 클릭하게 되면 (그림 5)와 같이 그 문서에 대한 전체적인 윤곽을 파악하면서 원하는 부분을 자세하게 볼 수 있도록 하는 줌 브라우저를 자바를 이용하여 구현하였다.

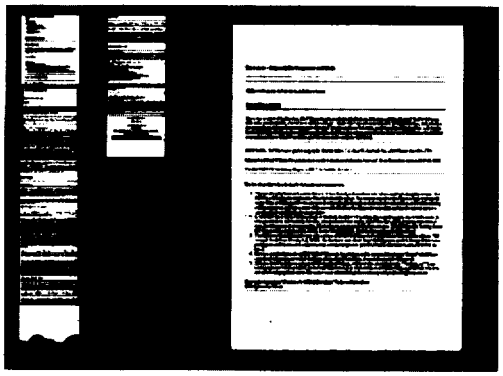
(그림 4)와 같은 검색결과의 시각화 화면에서 사용자가 원하는 문서를 클릭하면 (그림 5)와 같이 읽을 수 있는 크기로 페이지가 확대되어 나타나게 된다. 그림에서 보는 바와 같이 선택되지 않은 페이지들은 현재 확대되어진 페이지 왼쪽으로 재배열하여 계속해서

다른 페이지를 클릭할 수 있도록 하였다. 새로운 페이지가 클릭되면 그 페이지가 확대되어 나타나고 그 전의 것은 원래의 크기로 축소된다.



(그림 5) 검색결과에 대한 상세 정보 표현
(Fig. 5) Detailed Display of the Search Results

검색엔진으로부터 찾아진 결과를 사용자가 충분히 파악한 후에 더 자세히 알고싶은 내용을 클릭하면 사용자가 관심있는 정보가 어느 정도의 분량인지 한 화면에 걸쳐 전부 나타나게 되고, 사용자가 필요로 하는 페이지를 클릭하게 되면 (그림 6)과 같이 확대되어 앞에서와 마찬가지로 사용자가 읽기 쉬운 형태로 확대된다.



(그림 6) 페이지 확대
(Fig. 6) Page Zoom-in

줌 브라우저에서 사용자는 검색된 문서의 처음부터 페이지의 내용을 확인할 수도 있고 사용자 임의대로 중간 페이지부터 확인할 수도 있다. 사용자가 원하는 페이지를 클릭하면 그 페이지는 확대되면서 활성화되게 되고, 다른 페이지를 클릭 하면 다시 새로운 페이지가 확대되면서 그 이전 페이지는 원래 크기로 축소되게 된다.

클라이언트가 서버로부터 받아온 검색결과는 HTML 파일 형태이다. 이 HTML 문서를 페이지 단위로 나누어 표현해 주기 위해서는 HTML을 분석하는 과정이 필요하다.

먼저, HTML의 각 태그와 옵션을 파악한 후 데이터를 해당 태그와 옵션에 맞추어 자바 애플릿에 표현될 수 있도록 변환해 준다. 이 과정을 거쳐서, 일반적으로 넷스케이프 상에는 HTML의 태그 등은 나타나지 않고 필요한 정보만 나타나듯이 자바 애플릿에도 사용자가 필요로 하는 데이터만 나타나지게 된다.

다음은 알타비스타에서 검색된 결과에 나타내어진 제목과 URL, 내용, 마지막 수정일을 추출하기 위해서 클라이언트에서 HTML 문서를 분석하는 알고리즘이다.

<알타비스타 검색결과에서 제목과 URL, 내용, 마지막 수정일을 분석하는 알고리즘>

```
for (int i=0; i<perpage; i++) {
  try{
  mUrl[mlen] = matchList[i].substring(matchList[i].
     indexof("href=")+6, matchList[i].indexof("\">"));
                    ⋮
  mTitle[mLen] = matchList[i].substring(matchList[i].
 indexOf("<b>")+3,matchList[i].indexOf("</b>"));
  matchList[i] = matchList[i].substring(matchList[i].
 indexOf("<br>")+4);
  mContext[mLen] = matchList[i].substring(0,
    matchList[i].lastIndexOf("<font"));
  if (mContext[mLen].indexOf("<br>")!=-1)
  mContext[mLen] = mContext[mLen].substring(0,
  mContext[page+i].lastIndexOf("<br>"));
  mLastModified[mLen] =matchList[i].substring
 (matchList[i].indexOf("modified")+9,matchList[i].
  lastIndexOf(" - page"));
                    ⋮
                mLen++;
```

(그림 5)에서 보는 바와 같이 선택되지 않은 페이지들은 현재 확대되어진 페이지 왼쪽으로 재배열하여 계속해서 관심있는 다른 페이지를 선택 할 수 있도록 하

였다. 새로운 페이지가 선택되면 그 페이지가 확대되어 나타나고 이전의 페이지는 원래의 크기로 축소된다. 문서의 전체를 일정한 페이지 단위로 나누는 기준은 현재 창의 크기를 읽어 들여서 그 값과 데이터를 계속해서 읽어들여서 축적되는 데이터의 크기 값과 비교하여 분할해 준다.

다음은 현재 창의 크기에 맞추어서 데이터를 나누어주는 알고리즘이다.

```
String pages[] = new String[100];
curPage = 0;
                    :
if (lineWidth+fm.stringWidth(tmpStr) >= width) {
   switch (Align) {
   case 0 :   dPosX = 0;                    break;
   case 1 :   dPosX = (width-lineWidth) / 2; break;
   case 2 :   dPosX = width-lineWidth;   break;
   }
   dPosY += maxHeight;
   maxHeight = fm.getHeight();
}
                    :
if (dPosY > height) {
   curPagr ++;
   pages[curPage] ="@code\n"+
        "FontName="+fontName+"\n"+
        "FontSize="+fontSize+"\n"+
        "FontShape="+fontShape+"\n"+
        "FontColor="+fontColor+"\n"+
        "Align="+Align+"\n";
}
```

검색엔진으로부터 검색한 결과를 사용자가 충분히 파악한 후에 더 자세히 알고싶은 내용을 선택하면 그 문서의 내용이 어느 정도의 분량인지 한 화면에 거쳐 전부 나타나게 되고, 사용자가 필요로 하는 페이지를 선택하게 되면 (그림 6)과 같이 확대되어 사용자가 읽기 쉬운 형태로 나타나게 된다.

변환 프로그램에서는 현재 애플릿 창의 크기를 읽어들여서, 그 값을 일정 변수에 저장 후 읽어 들여진 정보의 폰트 크기 등을 더하며, 그림이 있을 경우 그림의 크기도 읽어들여 한 화면에 표현되도록 한다. 정보의 양이 많아 한 페이지를 넘길 경우 새로운 페이지에 계속 쓰여지도록 한다.

기존의 검색엔진이 보여주는 결과를 보다 더 시각적으로 표현해 주는 줌 브라우저는 웹 브라우저에 링크되는 정보의 크기가 점점 커져 가는 오늘날에 있어서 매우 유용하게 사용될 것이다. 긴 문서에서 자신이 원하는 특정부분의 검색시간을 줄여주면서 동시에 페이지 단위로 분할 표현함으로써 리스트 형태의 문서표현이 가져오는 지루함을 극복하였기 때문이다.

5.4 기존 브라우저와의 비교

Brown 등이 제안한 웹 브라우저에서는 HTML의 헤드태그에 지퍼라 불리는 아이콘을 첨부하고, 사용자가 이 지퍼를 선택함으로써 문서의 각 부분을 볼 수 있도록 하였다. 이 기법은 복잡한 계층구조를 갖는 문서에서는 효과적이나 한 섹션이 긴 문서의 경우에는 일반 문서표현과 마찬가지로 스크롤바를 제공하여 표현하여서 전체 정보량을 쉽게 파악하기가 힘들다.

또한 Holmquist와 Ahlberg이 제안한 Flip Zooming 브라우저는 문서의 전체정보를 여러 페이지로 나누어서 한 화면에 표시하여 사용자에게 편리한 시각적인 표현을 제공하지만 이것은 일반문서일 뿐 웹브라우저 상에서 실행되는 HTML 문서 등 정보검색결과를 나타내주지는 못한다. 또한 이것은 그림과 문자가 같이 들어있는 정보를 페이지 단위로 나타내도록 하는 것을 향후연구과제로 제시하고 있다.

Pad++ 웹 브라우저는 Pad++을 사용하여 구현된 다양한 기능의 그래픽 환경을 제공하는 브라우저이다. 이 브라우저에서는 Focus+Context 기법을 이용하여 정보를 표현하고 있고, 각 정보들의 상하위 계층관계를 링크로 표현하고 있어서 전체정보관계를 잘 파악할 수 있도록 하였다. 그러나 여기에서도 확대된 문서가 클 경우 문서의 처음부분이나 나중부분으로 커서를 옮기는 문제를 해결하지 못했다.

이에 반하여 본 논문에서 구현한 줌 브라우저는 일반 검색엔진을 사용하여 검색한 결과를 넷스케이프 상에서 페이지 단위로 나누어서 전체정보를 한 화면에 표현하였고, 사용자가 원하는 페이지를 선택하면 해당 페이지가 확대되게 된다. 또한 줌 브라우저는 검색결과에 문자 뿐 만 아니라 그림 정보가 같이 표현되어 있더라도 창의 크기에 맞추어서 적절히 표현해 주고, 웹 상에서 실행 가능함으로 어느 곳에서든지 검색결과를 페이지 단위로 나누어 볼 수 있다.

기존의 검색엔진이 보여주는 결과를 보다 더 시각적으로 표현해 주기 위해 본 논문의 줌 브라우저는 긴 문서에서 자신이 원하는 특정부분을 검색하는데 요구되는 검색시간을 줄여주면서 동시에 페이지 단위로 분할 표현함으로써 리스트된 문서의 표현이 가져오는 지루함을 없애 주고 편리한 인터페이스를 제공하므로 현재 사용중인 웹 브라우저에 적용함으로써 그 유용성을 확인하였다. 최근에 개발된 AMORE 시스템이 검색결과를 시각화한데 그친 반면, 본 논문에서는 시각화 화면에서 사용자가 선택한 문서에 대하여 줌 브라우징할 수 있는 기능도 제공한다.

6. 결론 및 향후 과제

본 논문에서는 검색엔진으로부터 검색된 결과를 시각화하여 주는 시각화 도구와 문서 전체의 개요를 파악하면서 관심있는 부분을 확대하여 보여주는 줌 브라우저를 구현하였다. 이 시각화 도구는 사용자에게 간편한 사용자 인터페이스를 제공하므로, 웹 상에서 관심있는 정보를 쉽게 찾는데 매우 유용하다.

검색 결과는 스크롤되는 문장 대신에 시각화된 아이콘을 나선형으로 배치하므로 많은 양의 정보를 나타낼 수 있고, 원하는 내용을 쉽고 빠르게 찾을 수 있으며, 선택한 문서의 웹 사이트로 즉시 이동할 수 있다.

CGI 대신 자바를 사용하였기 때문에 웹서버에 대한 부담을 줄이고, 어느 플랫폼에서든 사용할 수 있으며, 자바를 지원하는 모든 웹 브라우저상에서 실행된다.

줌 브라우저는 검색된 결과들을 페이지 단위로 나누어서 전체정보의 양을 한 화면에 표현하도록 하며 사용자가 원하는 부분을 클릭하면 그 페이지는 사용자가 읽기 쉬운 크기로 확대된다.

향후과제로 검색 결과를 VRML을 이용하여 3차원으로 시각화하여 공간을 돌아다니며 검색할 수 있도록 하고, 문자뿐만 아니라 이미지도 검색할 수 있는 데이터베이스 구축과 검색엔진을 개발하여 시각화 도구와 연동시키는 문제와 현재 문서의 내용뿐만 아니라 상위 문서들에 대한 전체적인 계층구조를 나타내주는 줌 브라우저의 기능을 확장하는 연구가 계속되어야 할 것이다.

참 고 문 헌

[1] Sougata Mukherjea, Kyoji Hirata and Yoshinori Hara, "Towards a Multimedia World-Wide Web Information Retrieval Engine," Sixth International World Wide Web Conference, 1997.

[2] Marc H. Brown, Hannes Marais, Marc A. Najork, William E. Weihl, "Focus+Context Display of Web Pages," http://gatekeeper.dec.com/pub/DEC...technical-notes/SRC-1997-010-html, 1997.

[3] Lous C. Vroomen, "Information Visualization : An Overview," http://www.crim.ca/~vroomen/writing/technical/reports/infovis.html, April, 1998.

[4] Marc H. Brown, Hannes Marais, Marc A. Najork, William E. Weihl, "Focus+Context Display of Web Pages," http://gatekeeper.dec.com/pub/DEC...technical-notes/SRC-1997-010-html, 1997.

[5] L. E. Holmquist, C. Ahlberg "Flip Zooming : A Practical Focus+Context Approach to Visualizing Large Data Sets," http://www.ling.gu.se/~leh/zoom/, 1997.

[6] Sougata Mukherjea, Kyoji Hirataand Yoshinori Hara, "Towards a Multimedia World-Wide Web Information Retrieval Engine," Sixth International World Wide Web Conference, 1997.

[7] Lous C. Vroomen, "Information Visualization : An Overview," http://www.crim.ca/~vroomen/writing/technical/reports/infovis.html, April, 1998.

[8] 노원빈, 허 신, "WWW상에서의 계층적 구조를 이용한 정보시각화 도구 개발", 정보처리논문지, 제5권 2호, pp.380-394, 1998.

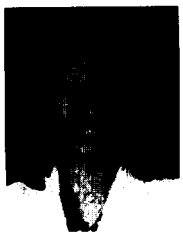
[9] Jason Wood, Ken Brodlie and Helen Wright, "Visualization Over The World Wide Web And Its Application To Environment Data," IEEE Computer Graphics and Applications, pp.81-86, 1996.

[10] Bo-yul Yoon and Eung-kon Kim, "The Design and Implementation of a Visualization Tool for the Search Results on the World-Wide Web," ICEIC'98, 1998.

[11] Elliotte Rusty Harold, "Java network programming", O'REILLY, 1997.

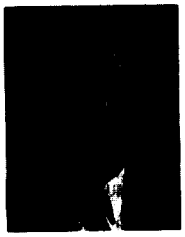
[12] Randall M. Rohrer and Edward Swing, "Web-Based Information Visualization," IEEE Computer Graphics and Applications, pp.52-59, Aug., 1997.

[13] 윤보열,전형민,정영아,김웅곤, "Java를 이용한 웹 검색결과의 시각화 도구 개발", 한국정보과학회 '98 가을학술발표논문집, 제25권 2호, pp.615-617.
[14] 정영아,김웅곤,허영남, "Focus+Context 기법을 이용한 Zoom Browser 구현", 한국정보과학회 '98가을 학술발표논문집, 제25권 2호, pp.618-620.



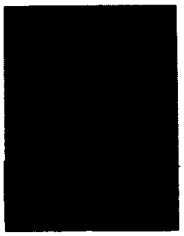
정 영 아

e-mail : younga@sunchon.sunchon.ac.kr
1996년 순천대학교 전자계산학과 (이학사)
1999년 순천대학교 대학원 컴퓨터과학과(이학석사)
1997년~현재 순천대학교 컴퓨터교육과 조교
관심분야 : 컴퓨터그래픽스



김 응 곤

e-mail : kek@sunchon.sunchon.ac.kr
1980년 2월 조선대학교 전자공학과 졸업
1987년 2월 한양대학교 전자공학과 공학석사
1992년 2월 조선대학교 전자공학과 공학박사
1984년 8월~1996년 8월 금성반도체(주) 연구소 연구원
1987년 3월~1991년 2월 국방과학연구소 선임연구원
1991년 3월~1993년 2월 여수수산대학 컴퓨터공학과 전임강사
1993년 3월~현재 순천대학교 컴퓨터과학과 부교수
1997년 3월~1998년 2월 Univ. of California, Santa Cruz Post Doc.
관심분야 : 컴퓨터그래픽스, 영상압축, 영상처리



한 승 조

e-mail : sjbhan@mail.chosun.ac.kr
1980년 조선대학교 전자공학과 학사
1982년 조선대학교 대학원 전자공학과 석사
1994년 충북대학교 대학원 전자계산학과 박사
1997년 조선대학교 전자·정보통신공학부 교수
1986년 6월~1987년 3월 Univ. of New Oleans 객원교수
1995년 2월~1996년 1월 Univ. of Texas 객원교수
관심분야 : 네트워크보안, ASIC설계, 음성합성, 영상압축