자동 Deep Tagging 에 의한 하이퍼비디오 브라우저의 설계와 구현

조명지, 김성환
서울시립대학교 컴퓨터과학부
e-mail : leisure7@uos.ac.kr, swkim7@uos.ac.kr

Design and Implementation of Hyper-Video Browser by Automatic Deep Tagging

Myung-Ji Cho and Seong-Whan Kim
Dept. of Computer Science, University of Seoul, Jeon-Nong, Seoul, South Korea 130-743

요 약

멀티미디어 자료는 빠르게 증가하고 있는 반면, 텍스트 기반의 검색엔진을 이용한 멀티미디어 자료 검색은 자료 내부를 검색할 수 없는 단점으로 인하여 검색된 정보의 정확성과 정확한 정보의 위치를 찾는 것이 어렵다. 그래서 이러한 문제를 해결하고자 멀티미디어 Deep Tagging 개념을 이용하여 비디오 파일에 자동으로 Deep Tagging 을 생성하고 또한 기존 하이퍼텍스트 기반의 하이퍼링크를 하이퍼비디오로 확장한 브라우저를 제안한다

1. 서론

학습을 위한 매체가 인쇄물에서 디지털 매체로 빠르게 이동하고 있다. 정부에서도 현 서책 형 교과서가 가지는 속보성 문제와 정보량의 제한성, 시각적 효과의 부족, 공간의 제한성 등의 문제를 극복하고자 2011 년 25 개 교과에 대한 디지털 교과서 [1]를 도입할 예정임을 발표하였다. 디지털 교과서의 개발로 기존 서책 형 교과서의 문제점을 극복하게 될 것이다. 하루에도 방대한 자료가 다양한 형식으로 새로 생성되고 사라지고 있는 이런 방대한 자료들 중에서 찾고자 하는 정보에 빠르고 정확하게 접근해야 한다는 점은 디지털 교과서에서 더욱 중요하게 요구될 것이다. 물론, 현재 텍스트 기반의 자료는 검색엔진을 통하여 자료의 세부 내용까지 추출되어 매우 높은 정확성을 확보하고 있고, 주기적으로 유효하지 않은 링크정보(dead link)를 갱신하여 접근 용이성도 보장하고 있다.

디지털 교과서는 학습 효과를 위하여 멀티미디어 데이터로의 접근을 반드시 요구한다. 하지만, 비디오, 오디오와 같은 멀티미디어 자료에 대한 검색 및 접근 방식에 대한 연구가 미약하며, 현재의 멀티미디어 검색 및 접근 방식은 생성시 만들어진 파일 이름, 사이즈, 포맷 타입 등의 기본적인 메타데이터 정보 통해서 혹은 생성자가 만들어 놓은 태그 (tag)를 바탕으로 이루어지고 있다. 최근에는 이러한 단점을 보완하고자 멀티미디어 파일에 Deep Tagging 개념을 반영한 텍스트 입력형 멀티미디어 편집기 개발이 활발히 이루어지고 있다. 이로써 접근의 용이성을 어느 정도 보완하였다. 하지만, 현재 비디오 형태로 제공되는 동영상들은 하이퍼미디어 기능을 가지지 않은 데이터 형태이기 때문에 비순차적 정보 접근이 어렵다. 이러한 제약은 디지털 교과서로 정보의 속보성 및 정보량의 제한성 문제 해결의 걸림돌이 된다.

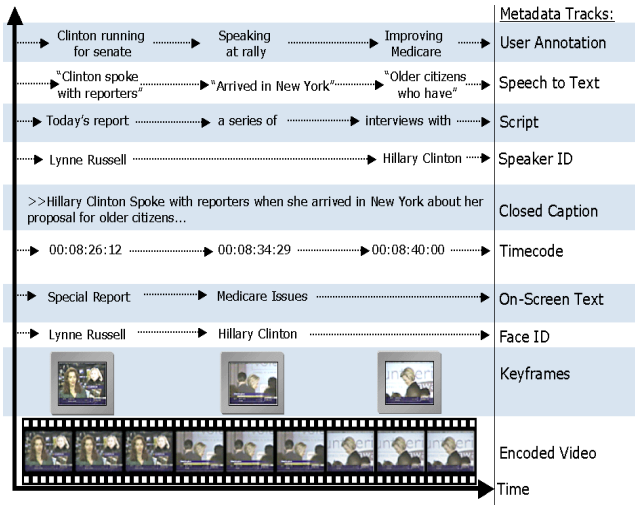
본 논문은 먼저 2 장의 관련 연구에서 현재 비디오 파일 내부에 Deep Tagging 개념을 도입한 기존의 연구 방법들에 대하여 설명한다. 3 장의 설계 및 구현 방법에서는, 비순차적 접근을 해결하기 위한 Deep Tagging 외에, 멀티미디어 정보 내의 오디오 검출을 통해, 멀티미디어 정보 간의 하이퍼미디어 기능을 제공하는 하이퍼비디오 브라우저 설계 및 구현에 대해 기술한다. 4 장 결론에서는 3 장에서 제안하는 방식의 이점과 디지털 교과서에서 적용하였을 경우의 이점에 대하여 기술한다.

2. 관련 연구

텍스트 기반의 정보를 비순차적으로 접근 가능하도록 하이퍼텍스트 개념을 제공하는 형태인 Web 문서는 HTML (hypertext markup language) 기반으로 구현된다. 하이퍼미디어는 텍스트 정보에 한적적인 하이퍼텍스트 개념을 멀티미디어 정보에 확장하여, 오디오 및 비디오 파일 내부에서 비순차적 접근을 가능토록 보완 한 방식이 Deep Tagging 개념을 이용한다. Deep Tagging 이란 오디오 혹은 비디오의 특정 장면에 대한 설명과 연관 텍스트를 연동하여, 해당 텍스트 정보를 통해 비디오 콘텐츠 내 검색과 비디오 콘텐츠 내의 링크 이동성을 높이는 개념이다. 이 기능을 사용하기 위해서는 기본적으로 비디오 타임코드와 Deep Tagging 의 동기화가 이루어져야 하며, 연관 텍스트 정보를 함께 저장하는 데이터 형태가 필요하게 된다.

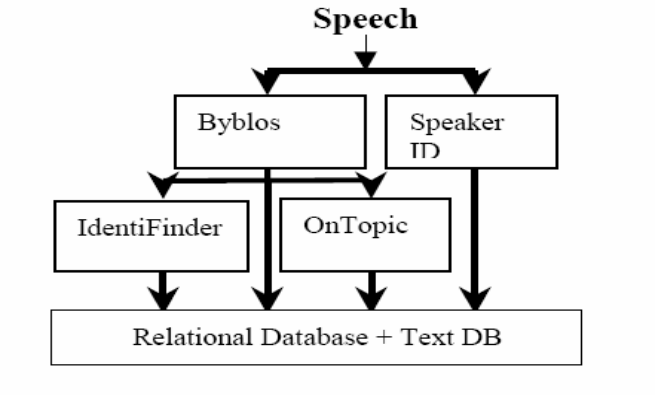
오디오 음성인식에 기반하여, 오디오트랙에서 자동으로 텍스트 정보를 구성하는 형태의 Deep Tagging 개념을 적용한 Virage VideoLogger [2] 소프트웨어에 대해 소개한다.

VideoLogger 는 우선 Virage SmartEncode 프로세스를 통하여 비디오 소스 인코딩 작업을 수행하며, 인덱싱 작업을 수행한다. Solution Server 프로세스는 VideoLogger 의 핵심으로 (그림 1)처럼 생성된 비디오 메타데이터를 HTML 형식으로 만들어 사용자들이 비디오 내 원하는 지점으로 이동할 수 있게 한다.



(그림 1) VideoLogger Metadata 생성

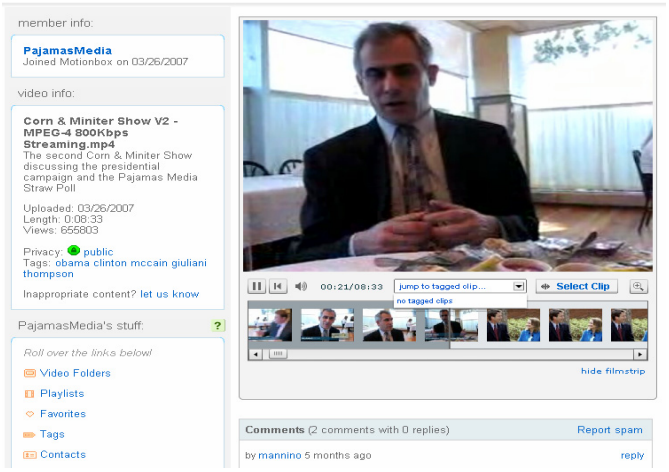
(그림 2)는 멀티미디어 정보의 오디오 트랙으로부터 해당 키워드를 인식하여 연관텍스트를 자동 생성하기 위한, Video Logger 시스템의 인덱싱 방법이다.



(그림 2) Virage Video Logger 의 인덱싱 방법

VideoLogger 에서 음성을 이용한 인덱싱 방법은 John Makhoul [3]이 제안한 방식으로 입력된 오디오 신호로부터 대화를 인지하고 그 내용을 자동으로 문서화하는 작업을 수행한다. 이 때 기존에 습득된 정보를 통해서 화자에 대한 인지 (speaker identification) 작업이 병행되며, 장소 혹은 인물 등의 고유명사를 검출하게 된다. 이후 Topic Indexing 프로세스는 HMM 기반의 분류 시스템인 OnTopic 컴포넌트에서 관련된 다양한 토픽으로 인덱스를 만들어낸다. 이로써 인덱싱이 된 정보는 DBMS (database management system)인 Solution Server 로 보내진다. 이후 클라이언트의 텍스트 기반으로 검색으로 접근이 가능하게끔 된다.

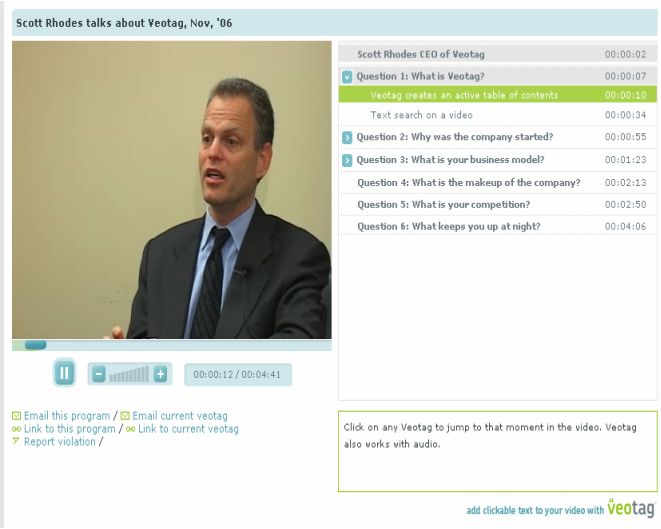
Motionbox™ [4] 시스템은, 음성인식을 기반으로 자동키워드텍스트 연관기법인 Video Logger 와는 달리, 사용자가 직접 tag 를 삽입하는 비디오 브라우저 서비스를 제공하는 시스템을 제공하고 있다. (그림 3) 에서 보는 바와 같이, 클라이언트로부터 업로드 된 비디오 파일은 전용의 멀티미디어 플레이어를 통해서 Deep Tagging 을 할 수 있게 한다.



(그림 3) Motionbox 를 통한 Deep Tagging 예

각 프레임이 플레이어의 하단부에 제공되며 비디오 파일이 재생되는 동안에 사용자가 직접 해당 프레임을 선택하여 텍스트를 삽입하는 방식이다. 따라서 파일 내부 검색은 사용자에 의해 삽입된 Deep Tagging 혹은 보여지는 키 프레임을 통하여 비디오 파일 내에서 이루어진다. Motionbox™ [4] 시스템은 사용자가 입력한 연관텍스트 정보를 HTML 형태로 제공하지 않기 때문에, 검색엔진 등에서 해당 멀티미디어 정보를 링크할 수 없다는 단점을 가지고 있다.

Veotag™ [4] (그림 4) 역시 업로드 된 비디오 혹은 오디오 파일을 재생하는 전용의 멀티미디어 플레이어를 제공한다. 전용의 멀티미디어 플레이어를 통해서 생성자가 직접 원하는 프레임 위치에 텍스트를 삽입하여 Deep Tagging 을 하는 방식으로 서비스를 제공한다. 이 때 편집된 Deep Tagging 은 Server 상에서 HTML 형식 기반으로 제공되기 때문에 기존의 텍스트 기반 검색엔진을 이용하여 외부에서 파일 내부의 특정 프레임으로 직접 접근할 수 있는 특징을 가진다. 하지만 Deep Tagging 으로 비디오 내 검색 및 링크 기능은 수행되지만 멀티미디어 데이터 간의 검색 및 링크 기능은 제공하지 못한다.



(그림 4) VeoTag 시스템의 Deep Tagging 예

3. 비디오 간 링크를 위한 Deep Tagging 기법의 설계

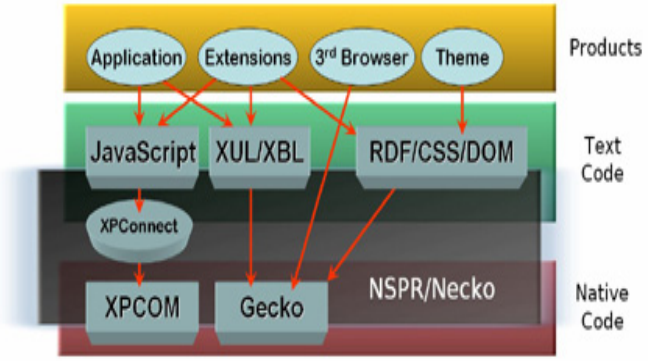
본 논문에서는 기존의 연구방법 또는 시스템들이, Deep Tagging 으로 비디오 내 검색 및 링크 기능은 수행되지만 멀티미디어 데이터 간의 검색 및 링크 기능은 제공하지 못하는 단점을 해결하고, 비디오 내에서의 링크 및 검색, 비디오 데이터 간의 링크 및 검색을 제공하기 위한 기법을 제안한다.

본 논문에서 제안하는 Deep Tagging 기법은, 비디오 파일 내 Deep Tagging 생성 방식은 파일 내 오디오를 통해서 미리 학습되어 있는 정보를 통해서 중요 단어를 순차적으로 검출하게 된다. 이때 검출된 내용으로부터 타임코드와 키 프레임을 기록한 메타데이터로 저장한다. 저장된 메타데이터는 다시 HTML 형식 기반으로 나타나게 하여 검색엔진에서 검색되도록 한다. 이렇게 하여 기존의 비디오 파일은 인덱싱 생성 과정을 거치고 난 후에 인덱싱이 포함된 파일로 브라우저에서 재생되도록 한다. 또한 이렇게 만들어진 Deep Tagging 은 키 프레임과 함께 미디어 링크 노드 값을 가지게 된다. Deep Tagging 된 정보를 제공하기 위해서 브라우저에 하이퍼미디어 뷰어를 설계한다. 뷰어는 원 비디오 파일과 메타데이터에 기록된 타임코드를 통해서 동기화가 이루지게 된 것이 나타나게 하고, HTML 기반으로 생성된 인덱싱 정보는 미디어 링크 노드 값을 가진 하이퍼텍스트와 키 프레임을 이용한 하이퍼비디오 정보가 나타나도록 한다.

4. 비디오간 링크를 제공하는 하이퍼비디오 브라우저의 구현

본 논문에서는 하이퍼미디어 브라우저의 기본형태로, Firefox 를 기본 브라우저로 이용하고 있다. 과거 브라우저 시장은 거의 Internet Explorer 의 독점이었으나, Mozilla Firefox [6]의 등장으로 새로운 전환점을 맞고 있다. 브라우저 기능 향상, 부가 기능 제공, 확장 기능 등의 용이성을 이유로 브라우저 시장의 새로운 강자로 Firefox 가 떠오르고 있다. 특히 웹 표준 기술에 충실한 개발 플랫폼에서 실행되는 독립적인 경량 애플리케이션이다. 또한 Firefox 의 특장점인 확장 기능을 활용하여 우리의 하이퍼미디어 브라우저 역시 Firefox 상에서 구현할 수 있다.

Firefox 는 (그림 5)과 같은 기술 구조를 가지고 소스코드는 모듈 형태로 파일형식인 컴파일 언어 C++과 객체 활용을 위한 JavaScript 로 구현된 오픈 소스 브라우저이다. Firefox 는 다양한 형태의 웹 브라우저 확장을 지원하기 위한 Products 계층과, 해당 플랫폼에 구현하기 위한 Native Code 계층, 그리고 기본 기능 확장 및 RDF 제공 등을 위한 Text Code 계층으로 구성된다. Firefox 의 렌더링 엔진 Gecko 는 표준 기술인 W3C 를 채택하고 플러그인 지원 및 기술 적용 비용이 무료이며 기본적으로 여러 개발 환경에 포함하여 자체적인 브라우저를 만들 수 있는 특징도 가지고 있다.



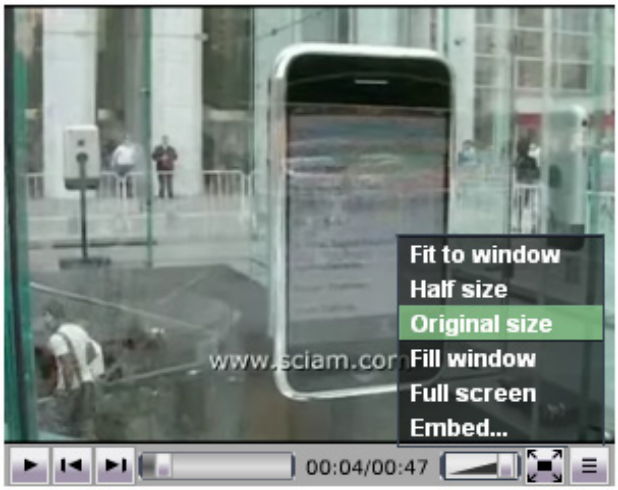
(그림 5) Mozilla Firefox 기반 구조

Gecko 를 이용해 브라우저를 만들 때 초기화와 종료에는 초기화 함수 (NS_InitEnbedding)와 셧다운 함수 (NS_TermEmbeding) 등 2 개의 C++함수를 사용한다. 초기화 동안에 nslWebBrowser 인터페이스를 사용하여 일반적인 브라우저의 윈도우를 표시하고 프로그램의 chrome 과 연결시키고 DOM 객체를 얻는다. nslWebBrowserSetup 은 브라우저가 열릴 때 그림을 표시할지 여부 등과 같은 일반적인 설정을 위해 사용되며 nslBaseWindow 는 윈도우와 기본 명령 (크기, 위치, 타이틀 검색 등)을 표현한다. nslWebNavigation 은 URL 방문 기록에서 앞, 뒤 이동을 제어하는 기능을 nslWebBrowserFind 는 검색과 실행을 제어한다. nslWebBrowserChrome 은 Gecko 웹 브라우저의 가장 기초 인터페이스로 nslWebBrowser 를 이용해 웹 브라우저를 만들 수 있는 필수 인터페이스이다. nslEmbeddingSitWindow 는 Gecko 에 윈도우 크기를 변경하거나 숨기거나 윈도우 타이틀 등을 변경할 때 사용한다.

Firefox 의 경우 OS 의 의존성을 제거하기 위해 독자적인 그래픽 사용자 인터페이스(GUI) 레이아웃을 정의했으며, 사용자 인터페이스를 정의하기 위해 웹 표준인 XML 기반 문법에 따라 XUL 을 만들었다.

XUL 은 브라우저의 외양을 매우 간단하게 바꿀 수 있게 한다. 따라서 제안하는 브라우저는 XUL 을 이용하여 제반 되는 외양을 구현한다.

본 논문에서는, Firefox 웹브라우저 상에서 실행되는 Flash 형태로 구현한다. Flash 비디오 플레이어로 오픈소스 프로그램인 FlowPlayer [7] 를 확장하여 사용한다. FlowPlayer 는 대부분의 운영체제 환경과 브라우저에서 구동되며 WMV, AV, ASF 등의 파일 포맷에 비해 보안성이 높고 작은 용량으로 고화질을 구현하는 FLV (Flash Video) 파일에 안정적으로 동작하면서 H.264 비디오 표준안 역시 지원하며 기존의 미디어 플레이어들이 가지고 있는 기능들을 모두 포함하는 장점이 있다. (그림 6) 은 FlowPlayer 를 사용하여 비디오 영상을 플레이 하는 예를 보이고 있다.



(그림 6) FlowPlayer 를 통한 비디오 플레이 예

(그림 7)은 브라우저 상에 FlashPlayer 를 임베디드 시키는 코드의 일부로 swfobject JavaScript 파일은 Adobe Flash content 를 임베딩하기 위한 것이다.

5. 결론

본 논문에서 제안하는 방법은 오디오 정보를 통해서 자동으로 멀티미디어 파일에 세부 정보를 삽입시켜 멀티미디어에서 멀티미디어로 이동하는 하이퍼미디어를 구현하는 것이다. 따라서, 멀티미디어 자료의 접근성 및 멀티미디어 자료 간의 이동성을 높이기 때문에 디지털교과서에서 멀티미디어를 통한 학습 시 학생들의 흥미유발 및 학습 효과 증대를 가져올 것이다. 또한, 이제는 사용자 인터페이스 기능과 내부 기반 기술을 지속적으로 축적하는 개발 플랫폼의 모양을 띄며, 크로스플랫폼을 지향하는 Firefox 상에서 구현하기 때문에 추후 디지털 교과서 도입에 따른 단말기, PDA, 컴퓨터 등의 하드웨어에 종속적이지 않고 안정적으로 구동될 수 있는 강점을 가질 수 있다. 패턴 인식 연구 등 기타 다른 연구 분야와 결합하여 멀티미디어 파일의 Deep Tagging 방식을 다양화 할 경우 디지털 교과서의 시각적 효과의 극대화 효과를 가져올 수 있을 것으로 보인다.

```
<script type="text/javascript"
src="/video/swfobject.js"></script>
<div id="fp1" class="flowplayer">Basic demo</div>
<script type="text/javascript">
  // <![CDATA[
  var fo = new SWFObject("/video/FlowPlayer.swf", "FlowPlayer",
"300", "250",
       "9", "#ffffff", true);
  fo.addParam("AllowScriptAccess", "always");
  fo.addParam("allowFullScreen", "true");
  fo.addVariable("config", "{
       playList: [
         { url: 'http://blip.tv/uploadedFiles/Zeitgeiber-
FirstInLineForIPhoneGuyDay3531-469.jpg', overlayId: 'play' },
         { url: 'http://blip.tv/file/get/Zeitgeiber-
FirstInLineForIPhoneGuyDay3531.flv?source=1' },
         { overlayId: 'play' } ],
       showPlayListButtons: true, loop: true, initialScale:
'orig', autoBuffering: false, useNativeFullScreen: true }");
  fo.write("fp1");
// ]]>
</script>
```

(그림 7) FlowPlayer 임베딩 코드

참고문헌

[1] 고범석, “디지털교과서 도입과 전망” 한국교육학술정보원, vol. 4 no. 1, pp.16-19, Spring, 2007.
[2] Chuck Fuller, “Virage Developer Program Overview,” http://www.broadcastpapers.com/whitepapers/ViragеDeveloperProgram.pdf.
[3] John Makhoul, Francis Kubala, Timothy Leek, Daben Liu, Long Nguyen, Rrchard Schwartz, and Amit Srivastava, “Speech and Language Technologies for Audio Indexing and Retrieval,” in Proceedings of the IEEE, vol. 88, pp. 1338-1353, 2000.
[4] MotionBox, http://www.motionbox.com
[5] VeoTag, http://www.veotag.com
[6] FireFox, http://www.mozilla.org/projects/
[7] FlowPlayer, http://flowplayer.org/