사용자와의 상호작용을 통한 HTML 문서의 XML 문서로의 변환

김승원[0] 민준기 정진완
한국과학기술원 전자전산학과
{swkim, jkmin, chungcw}@islab.kaist.ac.kr

Converting HTML Documents to XML Documents through Interactions with Users

Seung-Won Kim[0] Joon-Ki Min Chin-Wan Chung
Dept. of Electrical Engineering & Computer Science, KAIST

요 약

웹에 데이터를 나타내기 위해서 사용하는 HTML은 데이터를 표시(presentation)하기 위한 언어일 뿐 데이터의 의미를 나타내지는 못한다. 이러한 HTML의 단점을 극복하고 데이터의 표시(presentation)와 의미(semantic)를 나타낼 수 있도록 한 마크업 언어가 XML이다. HTML로 나타난 정보를 제대로 이용하기 위해서는 HTML 문서의 의미(semantic)정보를 알아내야만 한다. HTML 문서를 XML 문서로 변경할 수 있다면, 변경된 문서의 의미 정보를 이용할 수 있을 것이다. HTML 문서 포맷(format)을 XML 문서 포맷(format)으로 변경하기 위한 작업으로 [1]이 있다. [1]에서는 자동으로(automatic) 변환하는 방법을 사용했다. 이러한 방법은 프로그램이 HTML 문서의 의미를 파악하는데 한계가 있기 때문에 변환된 XML 문서에서 문서의 의미를 제대로 나타내기 어렵다는 단점을 안고 있다. 본 논문에서는 HTML 문서의 의미를 제대로 나타내는 XML 문서를 만들기 위해서 사용자가 어느 정도 개입하여 최종적인 XML 문서를 만드는 방법을 제안한다. 제안한 방법은 사용자의 약간의 개입으로 원래 HTML 문서의 의미를 보다 더 잘 나타내는 XML 문서를 만들어낸다.

1. 서론

HTML 문서는 데이터를 웹에 나타내기 위한 언어이다. 즉 데이터의 표시(presentation)에만 중점을 둔 언어로 데이터의 의미(semantic)를 담지 못한다. 이러한 단점을 극복하기 위해서 데이터의 의미(semantic)에 중점을 둔 마크업 언어인 XML이 탄생하게 되었다. XML 문서는 그것을 해석하는 응용 프로그램(application)에 의해서 다양하게 사용될 수 있다. 데이터베이스에 XML 문서의 내용을 저장할 수도 있고 웹 문서로 사용할 수도 있다. 단체에서 자신들만의 문서 포맷을 만들어 사용하기 위해서 XML을 이용할 수도 있다. XML은 의미에 중점을 둔 마크업 언어이기 때문에, 그것을 해석하는 응용 프로그램에 따라 다양하게 이용될 수 있는 것이다. 그에 반해 HTML은 단지 데이터의 표시(presentation)만을 위해서 사용될 뿐, 다른 용도로 사용될 수 없는 단점을 가진다. 현재 웹에 존재하는 수많은 HTML 문서의 내용을 XML 문서처럼 다양한 용도로 사용할 수 있다면 편리할 것이다. 이러한 목적으로 HTML 문서 포맷을 XML 문서 포맷으로 바꾸려는 노력이 있었으며, 본 논문에서는 기존에 제시된 방법보다도 HTML 문서의 의미를 더욱 잘 나타내는 XML 문서로 변경하기 위한 방법을 제안한다.

본 논문의 구성은 다음과 같다. 2장은 기존의 변환 방법에 대해서 간략하게 소개하고 그 방법의 단점에 대해서 이야기한다. 3장은 본 논문에서 제시된 방법에 대해서 설명한다. 마지막으로 4장에서는 결론에 대해서 논의한다.

2. 관련연구

[1]의 논문은 자동으로 HTML 문서 포맷을 XML 문서 포맷으로 변환한 논문이다. 이 논문에서 저자는 HTML 문서를 XML 문서로 변환하기 위해서 먼저 HTML 문서의 내용 구조(이 논문에서는 data hierarchy 라는 말을 사용했다.)를 추출해낸다. HTML 문서의 내용구조라는 것은 HTML 문서의 태그 (tag)들을 중요도에 따라 우선순위를 매기고 우선순위에 기초하여 문서의 계층구조(hierarchy)를 나타낸 것을 말한다. 여기서 우선순위(precedence)라는 것은 예를 들면 h1 태그가 h2 태그보다 더 큰 제목을 나타내므로 더욱 중요하다라는 사실을 의미한다. 이 우선순위에 기초해 나타난 계층구조를 나무구조(tree structure)로 나타내고 나무구조를 바탕으로 XML 문서를 생성해낸다. 이 과정에서 모든 HTML 태그는 제거되고, 원래 HTML 문서의 텍스트(text) 내용만을 가지고 변환될 XML 문서의 요소(element)를 생성한다. 이 방법은 몇 가지 단점을 갖게 된다.

첫째로 HTML 태그를 기준으로 우선순위를 매겨서 내용구조를 나타내려면, 선정된 우선순위가 정확하지 않으면 잘못된 내용구조를 나타낼 수 있다는 점이다. 즉, [1]의 논문에서 만든 태그 우선순위는 대부분의 사용자들이 태그를 그렇게 사용할 것이다라는 가정하에 만든 것이지만, 실제로 [1]의 기준에서 만든 우선순위대로 작성되지 않은 HTML 문서들도 많이 존재한다는 것이다. 예를 들면 HTML 의 태그의 경우에는 헤드 태그(head tag)보다도 더 높은 중요도를 가질 수 있으나 [1]의 우선순위 (presendence)에서 헤드 태그보다 낮은 우선순위를 갖게 됨으로써 잘못된 내용구조를 가지는 XML 문서를 만들 수 있다는 것이다.

둘째로 [1]에서는 자동으로 XML 문서로 변경하는 방법을 사용했기 때문에, 원래 HTML 문서의 태그를 제외한 텍스트 내용을 모두 XML 문서의 요소(element)로 대응(mapping)하는 방법을 사용한다. 이러한 방법은 HTML 문서의 텍스트 내용에 알맞은 XML 요소(element)를 제공할 수 없다는 단점을 가진다. 사용자가 원하는 XML 요소(element)를 제공할 수 없다면 그 XML 문서의 효용성은 그만큼 떨어질 것이다. 왜냐하면 사용자가 원하는 XML 문서 요소(element)를 가지는 문서여야만 그 문서를 사용하는 응용 프로그램에서 원하는 작업을 할 수 있기 때문이다.

3. HTML 를 XML 로 변환하기 위한 방법

3.1 변환 알고리즘

HTML 문서를 XML 문서로 변환하기 위해서는 HTML 문서의 내용을 파악해야 한다. 그래야만 의미 중심의 XML 문서를 생성할 수 있기 때문이다. HTML 태그정보를 기반으로 HTML 문서의 내용을 파악한다. HTML 문서에서 문서의 구조를 가장 명확하게 나타내주는 태그는 헤드태그이다. 헤드태그(head tag)는 말 그대로 문서의 한 부분의 제목을 나타내주는 태그이기 때문이다. 처음 헤드태그가 나온 부분에서 다음 헤드태그까지의 부분을 한 영역으로 간주하여 처리한다. 헤드태그들 사이는 명확한 우선순위 관계가 존재하기

때문에 그 관계에 맞추어 헤드태그들의 계층 구조를 형성한다. 즉 h1 태그는 h2 태그의 상위 개념이고 h2 태그는 h3 의 상위 개념이라는 것이다. 같은 헤드태그는 같은 계층에 속한 것으로 간주한다. 이렇게 전체 HTML 문서를 헤드태그 별로 나누어 헤드태그들 사이의 관계를 설정하면 전체적인 HTML 문서의 구조가 나오게 된다. 전체적인 구조가 나온 후, 각 헤드태그 영역별로 내용 구조를 추출하는 작업을 하게 된다. 즉 HTML 을 헤드태그 영역으로 나눈 후, 각 헤드태그 영역 사이의 포함 관계를 설정하여 문서의 전체적인 구조를 잡아낸 다음 각 헤드태그 영역 안에 있는 의미 구조를 알아내는 작업을 진행하는 것이다.

지금까지의 과정은 HTML 문서의 내용을 파악하기 위해서 단지 문서의 태그 사이의 포함 관계를 설정해 주는 작업에 불과하다. 즉 HTML 문서 내용들 사이의 계층 구조(hierarchy)를 설정하는 것이었을 뿐이다. 이 계층 구조를 설정함과 동시에 문서의 DTD (Document Type Definition)도 추출하게 된다. 사용자는 이 문서와 DTD 를 가지고 사용자가 원하는 XML tagging 을 하게 된다. 여기서 XML tagging 란 생성된 문서의 내용에 XML 요소 이름(element name)을 부여하는 작업을 말한다. 물론, 여기서 사용자는 XML tagging 을 하는 것뿐만이 아니라, 원하지 않는 문서의 구조를 변경할 수도 있다. 또한 사용자가 모든 문서의 내용에 XML tagging 을 하는 것은 아니다. 어느 정도 프로그램이 예측 가능한 내용에 대해서 자동적으로 XML tagging 을 수행하기 때문이다. 사용자의 XML tagging 이나 문서 구조 변경과 같은 이러한 작업은 위에서 언급한 DTD 를 통해서 이루어지게 된다.

변환과정을 정리하면 다음과 같다.

1. 입력된 HTML 의 헤드태그에 기초하여 문서를 구분한다. 구분된 각 헤드태그 영역들 사이의 포함관계를 헤드태그의 중요도에 기초하여 설정한다.
2. 각 헤드태그 영역 안에 있는 태그들의 관계를 설정한다. 기본적으로 태그들의 관계는 HTML 태그의 역할에 기반을 두고 알아낸다. HTML specification 4.01 에 있는 대부분의 태그들 사이의 관계를 설정할 수 있다.
3. HTML 태그들 사이의 내용 관계를 설정하면서 프로그램이 예측 가능한 범위에서 XML tagging 을 수행한다. 그리고 지금까지 생성된 문서의 DTD 를 추출한다.
4. 지금까지 생성된 문서를 보면서 사용자는 DTD 를 변경하여 필요한 XML tagging 을 수행하고 내용 구조를 변경한다.
5. 프로그램은 변경된 DTD 를 입력으로 하여 문서를 최종적인 XML 문서로 변환하게 된다.

다음에는 실제로 변환하는 예를 들어 보이겠다.

```
<html>
<head><title>TidyGUI</title></head>
<body>
  <h1 id="title">TidyGUI</h1>
  <p id="subtitle">A Free Windows GUI Version of HTML Tidy</p>
  <address>Author: <a href="http://perso.wanadoo.fr/ablavier/">
   Andre Blavier</a></address>
  <h2 id="intro">What is it?</h2>
  <p>TidyGUI is a Windows GUI encapsulation of HTML Tidy.</p>
  <h3>Platforms</h3>
  <p>all Win32 versions of Windows.</p>
  <h2 id="rel-notes">Release Notes</h2>
  <dl>
  <dt>Version 1.1.5 (26 May 2001)</dt>
  <dd>Based on HTML Tidy version 4th August 2000.</dd>
  </dl></body></html>
```

그림 1 HTML example

위 그림은 실제 웹에 존재하는 문서를 축소한 것이다. 이 문서는 TidyGUI 라는 소프트웨어 프로그램에 대해서 설명하고 있다. 첫 번째 헤드태그(h1)에서는 TidyGUI 라는 소프트웨어의 이름과 저자가 나오고 있으며 두 번째 헤드태그(h2)에서는 그 프로그램이 하는 일을 간략히 설명하고 있다. 세 번째 헤드태그(h3)에서는 프로그램이 동작하는 환경에 대해서 기술하고 있고 네 번째 헤드태그(h4)에서는 프로그램의 버전 정보를 나타내고 있다.

```
<h2 id="intro">What is it?</h2>
<p>TidyGUI is a Windows GUI encapsulation of HTML Tidy.</p>
<h3>Platforms</h3>
<p>all Win32 versions of Windows.</p>
```

그림 2 HTML code fragment

그림 2는 그림 1에서 일부분만을 추출한 것이다. 이 그림 2에 위에서 설명한 변환과정(1~5) 중에서 1번과 2번을 적용하고 나면 다음과 같다.

```
<intro id="intro" idn="6">
 <heading idn="7">What is it?</heading>
 <description idn="8">TidyGUI is a Windows GUI encapsulation of
   HTML Tidy.</description>
 <Platforms idn="9">all Win32 versions of Windows.</Platforms>
</intro>
```

그림 3 그림 2에 변환과정 1과 2를 적용한 결과

그림 2에는 HTML 태그 사이에 아무런 구조나 관계를 찾아볼 수 없다. 그러나 그림 3을 보면 <intro>라는 태그가 <heading>, <description>, <Platforms>의 부모 태그가 됨을 알 수 있다. 이러한 관계는 그림 2의 태그 정보를 기초로 추출할 수 있다. 즉, h2는 헤드태그로 다음에 나오는 p 태그를 포함할 수 있고, h3도 헤드태그로 그 다음에 나오는 p 태그를 포함할 수 있다. h2는 h3 보다 상위개념의 태그이기 때문에 h3를 포함할 수 있다. 따라서 그림 3과 같은 포함관계(hierarchy)가 성립될 수 있는 것이다. 그림 2와 그림 3을 보면 h2를 heading으로 바꾸고 그림 2의 h2 영역이 나타내는 부분은 그림 3에서 <intro> 태그가 나타내고 있음을 알 수 있다. 그림 3의 <intro> 태그는 그림 2의 h2 태그 안에 있는 id attribute 값을 이용하여 설정한 것이다. 이렇게 예측 가능한 태그 이름을 프로그램이 미리 설정하는 것이다. 그림 3에서 추출된 DTD를 보자.

```
<!ELEMENT intro (heading, description, Platforms)>
<!ATTLIST intro id ID #REQUIRED
               idn CDATA #FIXED "6" >
<!ELEMENT heading (#PCDATA)>
<!ATTLIST heading idn CDATA #FIXED "7" >
<!ELEMENT description (#PCDATA)>
<!ATTLIST description idn CDATA #FIXED "8">
<!ELEMENT Platforms (#PCDATA)>
<!ATTLIST Platforms idn CDATA #FIXED "9">
```

그림 4 그림 3의 DTD

그림 4의 DTD에 사용자가 변경을 가해서 변환과정 4번을 수행하게 되는 것이다. 그림 4에서 화살표가 가리키는 부분의 요소(element) 이름을 사용자가 원하는 XML 요소 이름으로 변경하기 위해서 화살표 부분을 다음과 같이 변경한다.

```
<!ELEMENT program_info_8 (#PCDATA, c?)>
```

그림 5 DTD 수정

그림 5에서 사용자는 description이라는 일반적인 이름 대신에 program_info 라는 좀더 특정한 이름을 사용하였다. 이름 뒤의 숫자는 변경되기 전에 이 element가 어떤 element였는지를 나타내준다. 즉 이 element는 변경되기 전에 idn 속성(attribute)값이 8인 element였다는 것을 나타내는 것이다. Element 선언문의 마지막에 c? 라는 마크는 이 element가 변경된 element라는 것을 나타내는 것이다. 프로그램은 위 그림 5의 변경된 DTD와 그림 3의 소스를 입력으로 받아서 사용자가 원하는 변경을 수행하게 되는 것이다. 이런 방법으로 DTD를 수정하여 요소 이름 변경하기(element renaming) 같은 XML tagging 작업이나 요소 삭제(element deletion)와 요소 삽입(element insertion), 속성 변경 같은 문서 구조를 변경하는 작업을 수행하도록 할 수 있다. 다음 그림은 그림 3을 변환한 최종결과를 나타내고 있다.

```
<intro id="intro">
 <heading>What is it?</heading>
 <program_info>TidyGUI is a Windows GUI encapsulation of
   HTML Tidy.</program_info>
 <Platforms>all Win32 versions of Windows.</Platforms>
</intro>
```

그림 6 그림 3을 변환한 최종결과

이런 식으로 그림 1의 모든 헤드태그 영역을 변경하면 다음과 같은 최종결과를 얻는다.

```
<?xml version="1.0"?>
<TidyGUI>
 <title id="title">
  <heading>TidyGUI</heading>
  <subtitle>A Free Windows GUI Version of HTML Tidy</subtitle>
  <address href="http://perso.wanadoo.fr/ablavier/">Author: Andre Blavier
  </address>
 </title>
 <intro id="intro">
  <heading>What is it?</heading>
  <program_info>TidyGUI is a Windows GUI encapsulation of HTML Tidy.
  </program_info>
  <Platforms>all Win32 versions of Windows.</Platforms>
 </intro>
 <rel-notes id="rel-notes">
  <version_data>
   <version>
    <name>Version 1.1.5 (26 May 2001)</name>
    <info>Based on HTML Tidy version 4th August 2000.</info>
   </version>
  </version_data>
 </rel-notes>
</TidyGUI>
```

그림 7 그림 1의 최종 변환 결과

최종결과에서는 그림 7의 XML 문서와 함께 다음의 DTD도 출력한다.

```
<!ELEMENT TidyGUI (title, intro, rel-notes)>
<!ELEMENT title (heading, subtitle, address)>
<!ATTLIST title id ID #REQUIRED >
<!ELEMENT heading (#PCDATA)>
<!ELEMENT subtitle (#PCDATA)>
<!ELEMENT address (#PCDATA)>
<!ATTLIST address href CDATA #REQUIRED >
<!ELEMENT intro (heading, program_info, Platforms)>
<!ATTLIST intro id ID #REQUIRED >
<!ELEMENT heading (#PCDATA)>
<!ELEMENT program_info (#PCDATA)>
<!ELEMENT Platforms (#PCDATA)>
<!ELEMENT rel-notes (version_data)>
<!ATTLIST rel-notes id ID #REQUIRED >
<!ELEMENT version_data (version)>
<!ELEMENT version (name, info)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT info (#PCDATA)>
```

그림 8 그림 7 문서의 DTD

3.2 제안한 방법에 대한 고찰

본 논문에서 제안한 방법은 프로그램이 HTML 문서의 내용 구조를 계층적으로 파악하여 생성된 문서와 DTD를 가지고 사용자가 필요한 XML tagging과 구조 변경을 수행하는 과정으로 되어있다. 이러한 방법은 [1]에 비해 몇 가지 장점을 가질 수 있다.

첫째로 사용자가 원하지 않는 XML 문서가 생성될 가능성을 현저히 줄일 수 있다. [1]의 방법은 HTML 태그의 우선순위에 기반하여 HTML 문서의 내용 구조를 추출하고 자동적으로 XML 문서로 변환한다. 그 우선순위가 맞지 않을 경우에 수정할 수 있는 방법이 없는 것이다. 또한 [1]의 경우 XML tagging을 하기 위해서 HTML 문서의 텍스트(text)내용을 그대로 사용한다. 그러나 이 경우 XML 요소(element)가 사용자가 원하지 않는 이름을 가질 수 있다. 사용자가 원하지 않는 요소 이름을 가진다는 것은 결국 그 문서를 사용하는 응용 프로그램이 제대로 동작할 수 없다는 것을 의미한다. 이러한 단점들을 제거하기 위해서 본 논문에서는 프로그램이 HTML 문서의 내용 구조를 추출하고 사용자가 원하는 XML tagging과 구조 변경을 수행하도록 하는 방법을 사용한 것이다.

둘째로 사용자의 개입을 가급적 줄일 수 있다는 장점을 가진다. 프로그램이 HTML 문서의 내용 구조를 추출하고 예측 가능한 XML tagging을 수행하기 때문에 사용자가 개입하여 수정할 부분은 그렇게 많지 않게 되는 것이다. 물론 본 논문에서의 방법은 [1]의 방법보다 많은 사용자 개입을 요구하지만, 사용자가 원하는 문서를 만들어내기 위해서 어느 정도의 사용자 개입은 불가피하게 된다.

셋째로 사용자의 개입이 있다고 하더라도 프로그램을 통하여 하기 때문에 발생 가능한 에러를 없앨 수 있다. XML 문서는 상당히 정형화(well-formed)된 문서이다. HTML 문서를 프로그램의 도움 없이 사용자가 직접 XML 문서로 코딩(coding)한다면, 발생할 수 있는 에러는 많을 수 밖에 없다. 문서가 크면 클수록 그런 문제는 심각해진다. 사용자는 DTD를 수정하고 프로그램에 입력하는 방법을 취함으로써 XML 문서의 정형성 문제는 프로그램이 담당하게 된다. 따라서 에러가 없는 원하는 XML 문서를 쉽게 생성할 수 있다.

4. 결론

HTML 문서를 XML 문서로 변경할 때 사용자가 직접 모든 변환을 수행할 수도 있겠지만, 긴 HTML 문서를 직접 사용자가 XML 문서로 코딩(coding)한다는 것은 시간이 많이 걸릴 뿐만 아니라 에러가 발생할 가능성이 월등히 높아진다. 따라서 HTML 문서를 XML 문서로 변환해주는 프로그램이 있다면 편리할 것이다. 그러나 모든 것을 자동으로 변환해주는 프로그램은 사용자가 원하는 의미 정보를 제대로 나타낼 수 없다는 단점을 가진다. 따라서 최소한의 사용자의 개입으로 사용자가 원하는 XML 문서를 만들어내는 것이 바람직할 것이다. 본 논문에서는 그러한 방법으로 변환을 수행할 수 있는 알고리즘을 제시하였다. 본 논문에서 제시한 프로그램을 가지고 사용자는 에러에 대한 걱정 없이 최소한의 개입으로 원하는 XML 문서를 얻을 수 있다.

참 고 문 헌

[1] Seung Jin Lim and Yiu-Kai Ng, "A Heuristic Approach for Converting HTML Documents to XML Documents.", Computational Logic 2000: 1182-1196

[2] Elliotte Rusty Harold, "XML Bible", IDG BOOKs 1999

[3] Dave Raggett, Arnaud Le Hors, Ian Jacobs, "HTML 4.01 Specification", http://www.w3.org/TR/html401

[4] Elliotte Rusty Harold and W. Scott Means, "XML in a Nutshell", O'reilly, 2001

[5] Brett McLaughlin, "Java & XML", O'reilly, 2001

[6] Tim Bray, Jean Paoli, C. M. Sperberg-McQueen, Eve Maler, "Extensible Markup Language (XML) 1.0 (Second Edition)", http://www.w3.org/TR/REC-xml

[7] Seung Jin Lim and Yiu-Kai Ng, "Converting the Syntactic Structures of Hierarchical Data to their Semantic Structures", Brigham Young University