블록 단위 그래프 모델을 통한 효율적인 정보 추출 Wrapper 생성과 유지 관리*

박주영 ○ 양재영 최중민
한양대학교 컴퓨터공학과
parkjy@gmail.com isconan@gmail.com jmchoi@hanyang.ac.kr

Effective Information Extraction Wrapper Generation and Maintenance by Using a Block-Based Graph Model

Juyoung Park○ Jaeyoung Yang Joongmin Choi
Department of Computer Science and Engineering, Hanyang University in Ansan

요 약

기존의 정보 추출에서는 웹 문서의 구조가 변경되었을 때 Wrapper가 원하는 정보를 추출할 수 없었다. 또한 웹 문서의 구조가 바뀌는 경우 동일한 정보를 Wrapping함에도 불구하고 사용자는 정보를 추출할 수 없었던 이유를 찾지 못하는 경우가 대부분이었다. 이 문제를 해결하기 위해 본 논문에서는 Web 페이지를 시각적 블록 단위로 잘라 인접한 블록들을 통해 그래프를 형성하여 웹 문서의 구조가 일부 변경되어도 기존의 Wrapper를 통해 정보를 추출할 수 있도록 보다 효율적으로 Wrapper를 생성하고 유지 관리 하는 방법을 제안한다. 또한 웹 문서를 블록 단위로 분할 하여 그래프를 생성함으로써 블록 내부에 추출하고자 하는 정보에 대한 규칙이 좀 더 유연하게 표현 될 수 있으며 문서의 구조가 아닌 추출하고자 하는 정보를 중심으로 규칙을 생성함으로써 그래프의 구조뿐 아니라 그래프를 구성하고 있는 블록 내부의 구조가 일부 변하더라도 기존의 규칙을 이용하여 정보를 추출할 수 있도록 하였다.

1. 서 론

인터넷이 발전할수록 웹에서 접할 수 있는 정보의 양도 늘어나게 되었고 많은 양의 데이터 속에서도 원하는 정보를 빠르게 획득 할 수 있는 방법이 필요하게 되어 웹 문서를 대상으로 하는 다양한 정보 추출 기법들이 연구 되었다[1]. Wrapper는 웹 문서에서 자동으로 데이터를 추출하고 구조적 형식에서 정보를 변환하는 특별한 프로그램이라고 할 수 있다.

대부분의 웹 문서는 자주 변화하며 계속적으로 발전하는 성향을 가지기 때문에 웹 문서의 구조 역시 빈번하게 변화한다. 그러나 초기에 생성된 Wrapper는 이러한 경우 정상적으로 정보를 추출할 수 없다. 이에 따라 바람직한 정보를 추출하기 위해 초기 생성 된 Wrapper를 업데이트 하는 문제는 중요하다. 기존의 정보 추출에서는 웹 문서의 구조가 변경 되었을 때 Wrapper는 변경 전 웹 문서의 구조만을 유지하므로 원하는 정보를 추출할 수 없었다. 이를 해결할 수 있는 방법으로 기존의 Wrapper를 통해 원하는 정보를 추출할 수 없을 시 새로운 문서를 이용하여 Wrapper를 다시 생성할 수 있지만, 이것은 효율적인 방법이라 할 수 없다. 이 같은 문제점을 해결하기 위해 본 논문에서는 VIPS(Vision based Page Segmentation) 알고리즘[2]을 사용하여 웹 문서를 시각적으로 구분되는 블록 단위로 분할하고 그 중 실제 추출하고자 하는 정보를 가지고 있는 주요 블록(Hot block)을 중심으로 하여 인접 블록들과의 연결성을 통해 그래프화 하는 방법을 제안한다. 이것은 그래프를 통해 Wrapper를 보다 효율적으로 관리하고자 함이다. 이로써 웹 문서의 구조가 변경되었을 경우 기존 그래프와의 비교를 통해 문서의 구조 변화 정도를 예측하여 일부 구조가 변경된 경우에도 기존의 Wrapper를 이용하여 정보를 추출할 수 있도록 한다.

2. 관련 연구

Wrapper generation에 관련된 연구는 예전부터 활발하게 이루어 졌다. 하지만 이들 연구의 대부분은 Wrapper의 maintenance에 관한 관점은 아니다.

[4]에는 웹 문서의 구조 변화가 매우 작다는 가정하에 자동적으로 wrapper를 유지관리 하는 문제에 대해 다룬다. 전통적인 Wrapper에 내용 기반 분류기와 역(backward) Wrapper를 추가하여 유효한 정보를 추출하고자 하였고, 기존의 Wrapper를 통해 정보를 추출할 수 없을 시에는 정상적으로 정보를 추출 하도록

* 본 논문은 "국가 IT 온톨로지 인프라 기술개발" 정보통신부 선도과제 성과의 일부입니다.

Wrapper를 수정하는 것을 시도하였다. 하지만 이것은 웹 문서의 구조 변화가 매우 작다는 가정하에 wrapper를 유지 관리 하기 때문에, 전반적인 웹 문서의 구조 변화 시 적절하지 않을 수 있다.

[5]에서는 구조(schema-guided)와 관련하여 Wrapper를 유지보수 방법을 제안한다. 이것은 비록 HTML문서의 변화가 빈번할지라도, 이러한 문서에서 추출된 데이터 아이템의 일부 특징(문법적 특징, 하이퍼링크, 주석 등)은 종종 유지된다고 가정한다. 때문에 변화된 문서에서 이러한 특징들을 인식한다. [5]에서 제안된 방법은 네 가지 단계로 이루어 진다. 먼저, 사용자 정의 구조로부터(이전에 추출 규칙) 몇 가지 특징들을 얻고, 결과를 추출한다. 둘째, 얻어진 특징과 함께 변화된 문서에서 데이터 아이템을 인식한다. 셋째, 구조에 따라 아이템을 그룹화 한다. 각 그룹은 의미 블록이라 불리며 인스턴스가 된다. 마지막으로 대표적인 인스턴스는 새로운 문서를 위한 규칙을 뽑기 위해 다시 선택된다. 그러나 이것 역시 웹 문서의 구조 자체가 변화한 경우 적절한 유지보수 방법이 아닐 수 있다.

[6]은 Wrapper가 작동하는 동안 어떤 질의의 결과를 수집하는 것을 기반으로 한다. 자동적인 유지보수를 위해 Wrapper가 동작하는 동안 유효한 질의로부터 나오는 일부 결과를 수집함으로써 wrapper를 체크하고, 페이지가 변화하였을 때 이러한 결과들은 다시 Wrapper을 수정하기 위해 레이블 된 예제의 새로운 training set을 생성하는데 사용한다. [6]은 위와 같이 특별한 질의로 결과 수집 등의 추가적인 작업이 필요하기 때문에 효율적으로 Wrapper를 유지보수 하는 방법이라고는 할 수 없다. 본 논문에서는 시각적으로 자른 블록의 그래프화를 통해 웹 문서의 구조가 바뀌어도 효율적으로 Wrapper를 유지 관리하는 방법을 제시하고자 한다.

3. 웹 문서의 전처리 과정

제안한 논문은 복잡하지 않으면서도 웹 문서의 구조적 정보를 보다 잘 표현하고 있는 그래프를 생성하기 위해 그림 1과 같은 몇 가지 사전 처리작업을 거친다.

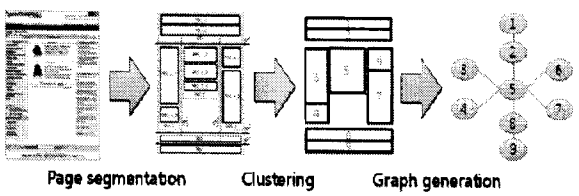

그림 1. 그래프 생성을 위한 작업흐름도

먼저 목표가 되는 웹 문서를 Cai가 제안한 VIPS 알고리즘[1]을 이용하여 시각적으로 구분되는 블록 단위로 분할한다. 다음 각 블록에서 구조적으로 유사한 블록을 군집화 하는 작업을 거친다.

군집화 작업은 쇼핑몰의 리스트 페이지와 같은 반복적인 구조가 나타나는 블록을 묶어 주기 위함이다. 이 작업은 각 블록들이 인접한 위치에 있으면서 구조적으로 유사한 경우에 이루어 진다. 각 블록이 구조적으로 얼마나 유사한 지를 구하기 위해 실제 웹 문서의 내용 정보 및 구조적 성향에 영향을 미치지 않는 <SPAN>, <SCRIPT>, <DIV> 등을 제거하고, <A>태그의 href 속성과 <IMG>태그의 src 속성 등 일부 태그의 속성에 대한 부분도 구조적으로 영향을 미치지 않는 경우 제거한다. 이후 인접한 블록에 대한 유사도 평가가 이루어 지는데, 이에는 문자열 편집거리(String edit distance)를 사용한다. 이것은 두 문자열에 대해 몇 번의 연산(바꾸기, 삽입하기, 삭제하기)으로 같아질 수 있는 지를 평가한다. 여기서 문자열 편집거리를 사용하되 블록 내에서 하나의 문자 단위로 하여 편집거리를 계산하는 것이 아닌 하나의 태그를 한 단위로 하여 두 블록 내의 유사도를 측정하였다.

$$
\begin{aligned}
&m[0,0]=0\\
&m[i,0]=i, i=1,2,...,|b_1|\\
&m[0,j]=j, j=1,2,...,|b_2|\\
&m[i,j]=\min(m[i-1,j-1]+p(b_1[i],b_2[j]),\\
&\qquad m[i-1,j]+1\\
&\qquad m[i,j-1]+1)
\end{aligned}
\quad \cdots (1)
$$

$$where\ i=1,2,...,|b_1|, j=1,2,...,|b_2|$$

$$p(b_1[i],b_2[j])=\begin{cases}0, & b_1[i]=b_2[j]\\ 1, & otherwise\end{cases} \quad \cdots (2)$$

$$Distance(b_1,b_2)=m[|b_1|,|b_2|] \quad \cdots (3)$$

두 블록에 대해 위의 수식을 통해 matrix m에 두 블록의 부분 거리가 점차 채워지게 되고, 두 블록의 최종 편집거리는 (3)이 된다.

$$
Cluster(b_1,b_2)
=\begin{cases}true, & if\ Distance(b_1,b_2)\le\alpha\ \ \&\\ & \quad isNeighbor(b_1,b_2)=true\\ false, & otherwise\end{cases}
\quad \cdots (4)
$$

최종 군집화는 두 블록이 인접한 위치에 있으면서 편집거리가 α 이하인 경우 이루어 진다. 두 블록이 인접한지 아닌지를 결정하는 함수에 대해서는 다음 장에서 논의하겠다. 또한 본 논문에서는 실험을 통해 α 의 값을 10으로 결정하였다.

4. 그래프 모델

4.1 그래프 생성

전처리 과정이 완료 된 후에는 그래프를 생성한다. 그래프 생성을 위해서는 먼저 각 블록들의 인접 블록을 계산한다. 이것을 계산하기 위해서는 VIPS에서 시각적으로 자른 것을 구분 해 주기 위한 사각형의 정보를 이용한다. 인접 블록은 각 블록에 대해 다른 모든 블록들의 좌표 값을 검사해 봄으로써 구해진다.(그림 2)

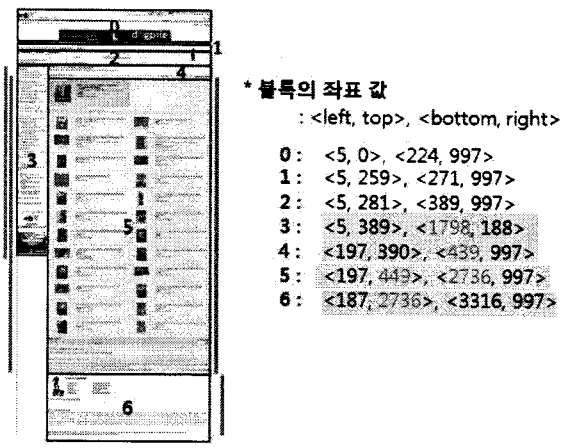

그림 2. 블록의 좌표 값 예제

```
Procedure isdNeighbor
input   : block i and block j
output  : true or false

temp1 = i_bottom - j_top
temp2 = i_top - j_top

Begin loop(segmented blok not exist)
    // the upside/downside block
    Begin if(temp1 < 50 && temp1 > -50)
    check = true
    End
    // the side block
    Begin if(temp2 < 50 && temp2 > -50)
        check = true
    End
    Begin if(i_top < j_top && i_bottom > j_bottom)
        check = true
    End
    Begin if((i_bottom > j_top) && (j_top > i_top))
        check = true
    End
End

return check;
```

그림 3. 인접 블록을 구하는 알고리즘

위의 그림 3은 인접 블록인지를 검사하는 알고리즘이다. 목표블록의 좌표 값을 중심으로 다른 블록들의 좌표 값을 검사한다. 이에 따라 그 좌표 값이 목표 블록과 물려있다면 인접 하다고 판단할 것이고, 맞물려 있지 않더라도 목표블록의 가장 상위 값과 비교블록의 가장 하위 값의 차가 적거나 목표 블록의 가장 하위 값과 비교블록의 가장 상위 값의 차가 적다면 또한 이 비교블록 역시 인접 하다고 판단한다. 이렇게 각 블록에 대한 인접 블록을 모두 체크하여 각 목표 블록에 대한 인접 블록이 구해 지면 그래프는 최종적으로 맵형태로 표현 된다.

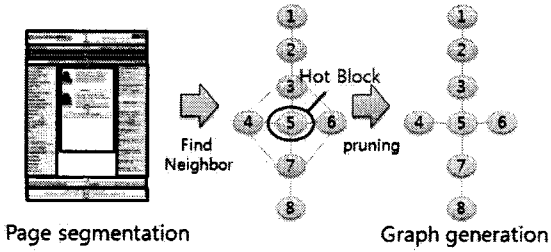

그림 4. 그래프의 생성과정

이후 주요 블록(Hot block)을 선택하는 과정이 필요하다. 주요 블록은 추출하고자 하는 정보가 있을 가능성이 가장 큰 블록으로써 이를 위해 군집화 된 각각의 블록마다 블록 내의 단어의 수와 이미지의 수를 고려하여 선택한다.

$$point(b_i) = w_1 |b_{token}| + w_2 |b_{img}|$$
$$|b_{token}| = \#token, \quad |b_{img}| = \#image \qquad \cdots\cdots (5)$$

(5)에서 가장 큰 값을 가지는 블록이 주요 블록으로 선택되고, 이 주요 블록을 중심으로 하여 그래프의 가지치기 과정을 진행한다. 이 과정을 통해 그래프의

주요 블록을 부각하되 조금 더 단순화 시킴으로써 다음 장에서 논의 될 그래프 비교 알고리즘을 좀 더 효율적으로 진행할 수 있도록 한다.

그림 5와 같이 가지치기 알고리즘은 주요 블록 내에 포함된 인접 블록들이 대상이 된다. 먼저 주요 블록 내에 있는 첫 번째 블록부터 목표 블록으로 하여 목표 블록에 있는 인접 블록들과 주요 블록에 있는 인접블록들을 비교 한다. 그리고 주요 블록 내에 있는 인접 블록이 목표 블록내의 인접블록으로 존재하게 되면 그 블록은 목표 블록의 인접블록에서 제외시킨다. 이로써 주요 블록을 중심으로 하여 생기는 그래프의 순환 구조를 제거하여 최종적으로 그래프가 생성된다.

```
Procedure pruneGraph
Input : clustered blocks
        with included neighbor blocks(graph)

Begin
    Begin loop(neighbor blocks not exist in hot block)
        target = neighbor i of hot block
        Begin loop(neighbor blocks
                not exist in target block)
            Begin loop(neighbor blocks not exist in
                    hot block)
                Begin if(neighbor block j and
                        neighbor block k are equals )
                    delete neighbor
                End
            End
        End
    End
End
```

그림 5. 가지치기 알고리즘

아래의 그림 6은 그림 4에서 보여준 그래프의 생성과정에서 순환 구조가 제거 된 상태이다. 초기 생성 된 그래프가 그림 6와 같이 보여지면 가지치기 알고리즘은 다음과 같이 적용된다.

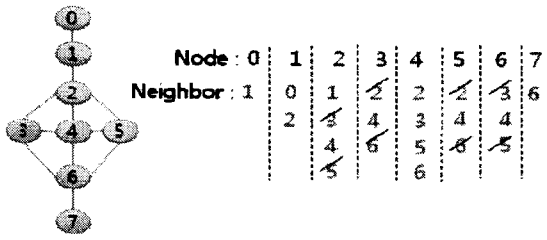

그림 6. 순환구조 제거

먼저 주요 블록내의 인접블록을 확인한다. 여기서 주요 블록의 인접 블록은 2, 3, 5, 6 블록이므로 가지치기 알고리즘이 적용되는 대상은 2, 3, 5, 6 블록이 된다. 그 뒤 이 네 개의 인접블록 들에 대해 순환구조를 가지는 블록들을 인접 블록 리스트에서 제거한다. 블록 2의 경우 네 개(1, 3, 4, 5)의 인접 블록을 가지고 있는데 이 중 가지치기 알고리즘에 따라 주요 블록이 가지고 있는 인접 블록과 2번 블록이 가지고 있는 인접 블록을 비교하여 동일한 블록이 존재할 경우 이를 제거 하게 된다. 이로써 2번 블록은 인접블록으로 최종적으로 두 개(1, 4)의 블록만을 가지게 되는 것이다. 3, 5, 6 블록 역시 이와 동일하게 적용되어 각각 하나의 인접블록만을 가지게 된다.

4.2. 그래프 비교

웹 문서에서 정보를 추출하고자 할 경우 먼저 기존에 생성된 그래프와 새로 정보를 추출하고자 하는 문서에서 그래프를 생성함으로써 두 그래프 간의 비교를 통해 웹 문서의 변화를 측정한다.

그래프의 비교는 두 맵을 비교함으로써 간단하게 이루어 진다. 그리하여 두 맵이 얼마나 유사한가를 측정하기 위해 블록 내에 포함 된 인접 블록의 수와 인접 블록의 블록 번호를 비교한다. 이로써 두 그래프의 차이가 특정 threshold값을 넘지 않는다면 두 그래프는 유사하다고 판단하여 기존에 생성된 Wrapper을 통해 정보를 추출한다. 그림 7의 예제는 웹 문서의 구조가 일부 변경되었음에도 불구하고 그래프로써의 표현에서는 동일한 그래프로 나타나게 됨을 보여주는 예제이다. 기존의 웹 문서에서 구조가 일부 변경되어 6번 블록이 3번과 4번 블록 사이로 이동하였다. 이때 본 논문에서 제안하는 그래프를 생성 알고리즘을 통해 두 그래프를 비교 해 본 결과 두 그래프가 완벽하게 일치함을 볼 수 있었다.

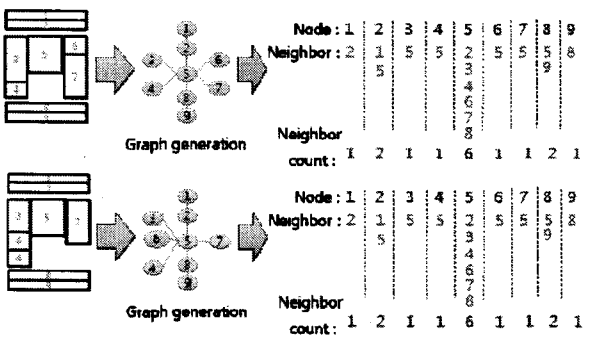

그림 7. 웹 문서의 구조 변화 예제

5. 실험 결과

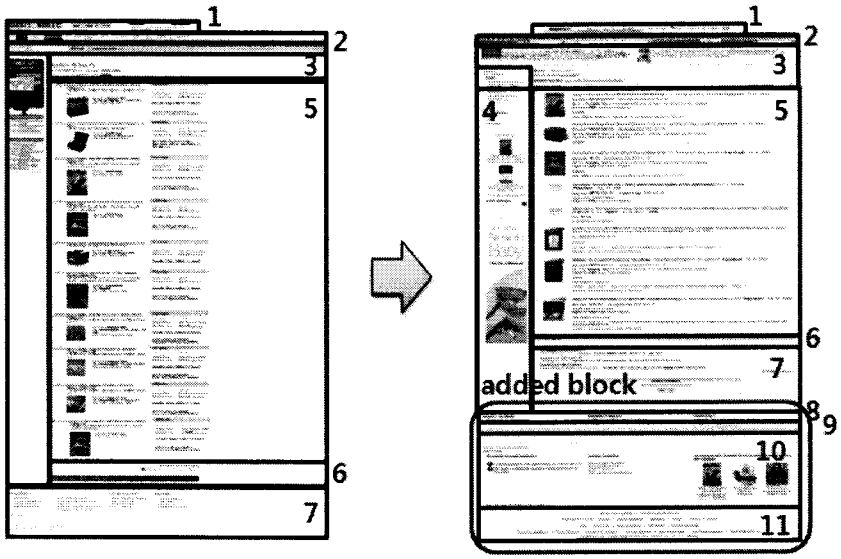

그림 8. 웹 문서의 구조 변화 예제

이 장에서는 본 논문이 제안하는 알고리즘 및 이론들을 시스템으로 구현해보고 실제 웹 문서를 이용한 실험을 통해 웹 문서의 구조가 바뀌었을 때에도 기존의 Wrapper를 통해 원하는 정보를 추출할 수 있음을 보고자 한다.

실험을 위해서는 특정 웹 문서를 일정시간을 두고 관찰하여 웹 문서의 구조가 변화하였을 때, 기존의 웹 문서를 바탕으로 만들어진 Wrapper를 통해 정보를 추출하여야 한다. 하지만 이 방법으로 실험을 할 경우 매우 오랜 시간이 걸릴 수 있다. 그러므로 본 논문에서는 웹 문서의 변화를 위해 같은 도메인의 두 웹 문서를 정하였다. 그래서 하나의 웹 문서를 원본 문서로 하고 다른 하나의 문서는 원본 문서의 구조가 바뀐 것으로 가정하였다. 원본 문서로 정한 웹 문서에서 Wrapper를 생성한 뒤 같은 도메인의 다른 문서에서 먼저 생성되었던 Wrapper를 통해 정보를 추출해 봄으로써 실험을 진행하였다. 이때에 추출하고자 하는 데이터 아이템은 쇼핑몰과 같은 도메인에서 특정한 순서로 나타난다는 특징을 고려하여 생성하였다.

표 1. 구조 변화를 위해 선택된 웹 문서 URL

원본 웹 문서	http://search.barnesandnoble.com/
변화된 웹 문서	http://www.amazon.com/

원본 문서와 변화된 문서를 위의 표1과 같이 정하였다. 이 두 웹 문서는 책에 대한 정보(제목, 저자, 가격 등)를 리스트 형태로 가지고 있다. 이 두 웹 문서의 변화는 그림 8과 같이 원본 웹 문서에서 일부 블록들이 추가되어 구조가 변화된 것을 가정한다. 실험결과 그림 9와 같이 제목과 저자에 대한 정보는 모두 정확하게 추출 되었으나, 가격에 대한 정보는 잘못된 정보를 추출하였음을 보여준다. 이 이유는 웹 문서의 전체적 구조뿐 아니라 내부 블록에서 데이터 아이템을 표현하는 구조 역시 바뀌었기 때문이다. 그래서 향후 Wrapper의 생성 시에 데이터 아이템의 순서를 고려하는 것이 아닌 태그의 속성과 구조적 특성을 이용한다면 전체적인 구조뿐 아니라 내부적으로 데이터 아이템의 순서가 바뀌었다 할지라도 데이터 아이템을 정확하게 추출할 수 있을 것이라 예상한다.

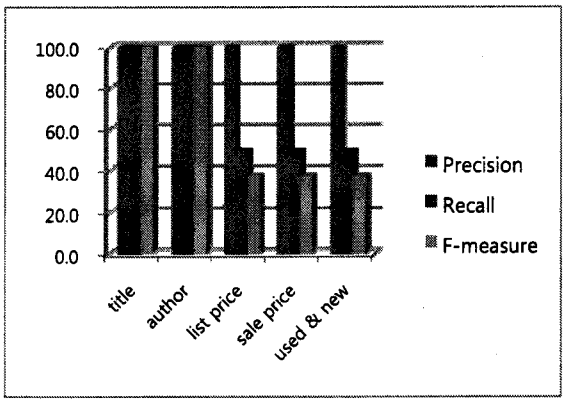

그림 9. 웹 문서에서 정보추출 성능

6. 결론

본 논문에서는 빈번하게 변하는 웹 문서의 구조를 시각적으로 구분 되는 블록단위로 분할하고 그 중 주요 블록을 중심으로 인접 블록들과의 연결성을 그래프로 표현함으로써 정보 추출 시 보다 효율적으로 Wrapper을 유지 관리 할 수 있는 방법을 제시하였다. 이 때 웹 문서를 블록 단위로 쪼개어 실제 뽑고자 하는 데이터가 들어있는 블록들을 통해 Wrapper를 생성하기 때문에 기존의 Wrapper에 비해 좀 더 일반적일 수 있는 장점을 가진다. 그렇지만 각 블록 내에서는 추출하고자 하는 데이터 아이템의 순서가 바뀔 경우 잘못 된 데이터를 추출할 가능성을 가지고 있다. 그래서 데이터 아이템의 순서를 고려하지 않고 Wrapper를 생성하는 방법을 통해 향후 이 문제점을 개선할 수 있을 것이며 이를 통해 Wrapper의 효율적인 유지 관리만이 아닌 다양한 환경에 쉽게 적응 가능한 Wrapper를 생성할 수 있을 것이라 예상된다.

참 고 문 헌

[1] A.H.F. Laender et al., A Brief Survey of Web Data Extraction Tools, ACM SIGMOD (2002)
[2] N. Kushmerick, Wrapper induction: Efficiency and expressiveness, Artificial Intelligence (2000)
[3] D. Cai, S. Yu, J R. Wen, and W Y. Ma, VIPS: a Vision-based Page Segmentation Algorithm, Microsoft Technical Report (2003)
[4] B. Chidlovskii, Automatic Repairing of Web Wrappers by Combining Redundant Views, ICTAI (2002)
[5] X. Meng, H. Lu, H. Wang, and M. Gu, Schema-Guided Wrapper Maintenance for Web-Data Extraction, WIDM (2003)
[6] J. Raposo, A. Pan, M. Álvarez, J. Hidalgo, Automatically Maintaining Wrappers for Web Sources, IDEAS (2005)
[7] K. Lerman, S. Minton, and C. Knoblock, Wrapper Maintenance: A Machine Learning Approach, JAIR (2003)
[8] Y. Zhai, B. Liu, Web Data Extraction Based on Partial Tree Alignment, IW3C2 (2005)