XML 기술과 스트링 매칭 기법을 이용한 구조 기반 정보 검색 알고리즘

한기덕○ 권혁철
부산대학교 컴퓨터공학과
{templer○, hckwon}@pusan.ac.kr

Structure Based Information Retrieval Algorithm Using XML Technology and String Matching Algorithm

Gi-deok Han○, Hyuk-chul Kwon
Department of Computer Science and Engineering, Pusan National University

요 약

Parsing 작업의 결과인 Parse Tree 정보는 문장에 관한 구조적 정보를 가지고 있는 Tree 정보로 이 정보를 이용하여 정보 검색에 활용하는 알고리즘을 제안한다. 제안하는 알고리즘은 XML 기술과 스트링 매칭 기법을 이용하였으며, 사용한 스트링 매칭 기법은 Approximate String Matching 기법이다. Query 정보와 문서 정보를 Parsing하여 얻은 Parse Tree를 XML 형태의 정보로 변환한 후, 두 정보를 가지고 Approximate String Matching 기법을 적용하여 Query 정보와 문서 정보 간의 유사도를 계산한다. 제안하는 알고리즘의 장점은 구조 기반의 정보 검색 기능이 가능하고 비슷한 정보에 대한 검색 기능이 가능하며 비슷한 구조에 대한 검색 기능이 가능하다는 것이다.

1. 서 론

정보 검색은 단어에 의한 정보 검색, 구조에 의한 정보 검색 등의 연구를 수행하는 분야로 최근에는 Semantic Web[1], Ontology 등 의미에 기반을 둔 정보 검색에 관한 활발한 연구가 진행되고 있다. 의미에 기반을 둔 정보 검색 기술의 핵심적인 정보 표현 언어는 RDF[3], OWL[4] 등이 있으며, 이 정보 표현 언어는 모두 XML[5]에서 파생된 언어이다. 따라서 XML 정보를 다루는 근본적인 기술의 개발은 의미에 기반을 둔 정보 검색 기술의 발전에 이바지할 수 있을 것이다. 본 논문은 Parsing의 결과물인 Parse Tree, XML 기술 및 Approximate String Matching[6] 기법을 활용하여 구조에 기반을 둔 정보 검색 알고리즘을 제안한다. 제안되는 알고리즘은 확장이 가능한 형태이며, Parse Tree가 의미에 기반을 둔 Parsing 기법을 이용하여 생성되었다면 제안하는 알고리즘의 적용 혹은 확장적용을 통해 의미에 기반을 둔 정보 검색 기능의 개발도 가능할 것이다. 제안하는 알고리즘을 간단하게 설명하자면 Query 정보와 문서 정보를 Parsing하여 얻은 Parse Tree를 XML 형태의 정보로 변환한 후, 두 정보를 가지고 Approximate String Matching 기법을 적용하여 Query 정보와 문서 정보 간의 유사도를 계산한다는 방식이다.

이 논문은 교육인적자원부 지방연구중심대학육성사업(차세대물류IT기술연구사업단)의 지원에 의하여 연구되었음)

제안하는 알고리즘의 장점은 구조 기반의 정보 검색 기능이 가능하고 비슷한 정보에 대한 검색 기능이 가능하며, 비슷한 구조에 대한 검색 기능이 가능하다는 것이다.

2. 관련 연구

2.1. XML

XML(extensible markup language)은 서로 다른 정보 시스템끼리의 데이터 교환 및 공유를 목적으로 개발된 태그 기반의 언어로 1996년 W3C(World Wide Web Consortium)에서 제안하였다. XML로 표현된 정보는 자유성, 가독성과 확장성이 뛰어나며, XML 기술을 이용하여 시스템을 관리하게 되면 뛰어난 시스템 확장성과 관리의 효율성이라는 장점을 얻을 수 있다. XML 기술을 이용한 통신 기법(SOAP), XML 기술을 이용한 보안 기법(SAML, XACML), XML Parser 기술 등 XML과 관련된 다양한 기술과 응용 예가 개발되고 있는 상황이다.

2.2. Approximate String Matching

Approximate String Matching은 두 문자열이 얼마나 차이나 나는지를 알려고 할 때 사용하는 스트링 매칭 기법이다. insertion(문자의 추가), deletion(문자의 삭제), substitution(문자의 변경) 이상 3가지 동작을 정의하고 두 문자열이 같아지려면 위의 3가지 동작을 각각 몇 번 수행해야 되는지를 계산하여 두 문자열이 같아지게 하기

위해 수행한 3가지 동작의 합계가 두 문자열이 얼마나 차이가 나는지를 나타내는 수치가 된다.

2.3. Parsing

Parsing은 각 문장의 문법적인 구성 또는 구문을 분석하는 작업, 즉 문장으로부터 추출한 토큰의 열을 이용하여 그 언어의 문법에 맞게 구문 분석 트리(parse tree)로 구성해 내는 작업을 말한다. Parsing 작업을 통해 얻어진 결과인 Parse Tree는 어떤 단어가 어떤 단어를 수식한다는 등의 구조적 정보가 포함되어 있는 Tree 정보이다.

3. 문장 Parsing 결과를 XML 문서로 변환

본 논문에서 다루는 정보는 일반적인 문장을 Parsing한 결과를 XML 문서로 변환한 정보이다. 즉, 정보 처리에 있어 Parsing 단계를 거친 이후의 정보 집합을 이용한다는 것이다. 영어 문장을 CFG(Context Free Grammar)를 이용하여 Parsing한 결과와 이를 XML 정보로 변환한 예를 다음 그림에서 보여준다.

영어 문장의 Parsing을 위해 사용되는 간단한 CFG

```
S -> NP VP
PP -> P NP
VP -> V NP
VP -> VP PP
NP -> NP PP
```

그림 1. Parsing을 위한 간단한 Context Free Grammar의 예

CFG를 이용하여 영어 문장을 Parsing한 예

영어 문장 : Astronomers saw stars with ears

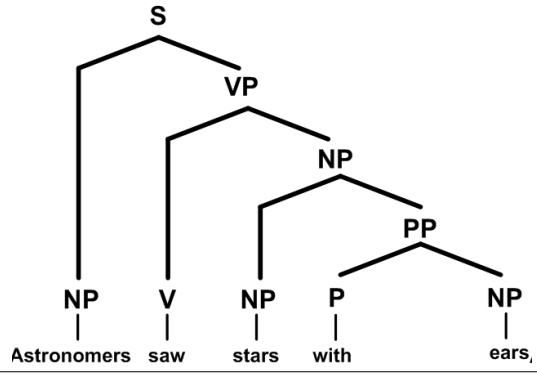

그림 2. CFG를 이용하여 영어 문장을 Parsing한 예

위에서 Parsing한 결과를 XML 정보로 변환한 예

영어 문장 : Astronomers saw stars with ears

```
<S>
    <NP>Astronomers</NP>
    <VP>
        <V>saw</V>
        <NP>
            <NP>stars</NP>
            <PP>
                <P>with</P>
                <NP>ears</NP>
            </PP>
        </NP>
    </VP>
</S>
```

그림 3. Parsing 결과를 XML 정보로 변환한 예

본 논문은 위와 같이 변환된 XML 정보를 가지고 정보를 검색하는 기법을 제시하는 것이 목적이며, Parsing의 성능 등은 고려 대상에 포함되지 않는다.

4. XML 기술과 스트링 매칭 기법을 이용한 구조 기반 정보 검색 기법

4.1. 사용자 Query와 문서 집합의 예

정보를 검색하려는 사용자의 Query와 실제 검색 대상이 되는 문서 집합은 모두 Parser에 의해 Parsing이 된 결과인 Parse Tree를 XML로 변환하여 처리한다. 아래의 그림 4와 그림 5는 사용자 Query가 “stars with ears"인 경우의 정보와 검색 대상이 되는 문서를 XML로 변환한 예를 보여준다.

사용자 Query와 Query 정보를 XML 형태의 정보로 변환한 예

star with ears

```
<NP>
    <NP>star</NP>
    <PP>
        <P>with</P>
        <NP>ears</NP>
    </PP>
</NP>
```

그림 4. 사용자 Query를 XML 형태의 정보로 변환한 예

검색 대상이 되는 문서 내용과 검색 대상이 되는 문서를 XML로 변환한 예

Astronomers saw stars with ears
Astronomers saw apples

```
<XMLDocumentBody>
 <S>
        <NP>Astronomers</NP>
        <VP>
                <V>saw</V>
                <NP>
                        <NP>stars</NP>
                        <PP>
                                <P>with</P>
                                <NP>ears</NP>
                        </PP>
                </NP>
        </VP>
 </S>
 <S>
        <NP>Astronomers</NP>
        <VP>
                <V>saw</V>
                <NP>apples</NP>
        </VP>
 </S>
</XMLDocumentBody>
```

그림 5. 검색 대상이 되는 문서를 XML 형태의 정보로 변환한 예

4.2. 스트링 매칭 기법의 적용 방법

Query에 적합한 문서를 검색하려면 4.1에서 언급한 사용자 Query와 검색 대상이 되는 XML 문서를 이용하여 유사도를 측정하는 방법이 명확하게 제시되어야한다. 본 논문에서는 스트링 매칭 기법을 이용한 유사도 측정 방법을 제시한다.

유사도 측정을 위해 사용하는 스트링 매칭 기법은 Approximate String Matching이며, 적용 방법은 다음과 같다.

1) XML 문서의 내용 중에 Query 정보의 Root Tag와 동일한 태그에 대해서만 Approximate String Matching 기법을 적용하며, Query 정보의 Root Tag와 동일한 태그의 태그 및 하위 정보들까지가 Approximate String Matching 기법의 적용 범위가 된다. (그림 6 참고)

사용자 Query의 XML 형태 정보

```
<NP>
        <NP>star</NP>
        <PP>
                <P>with</P>
                <NP>ears</NP>
        </PP>
</NP>
```

XML 문서의 내용 중에 Approximate String Matching 기법을 적용하는 부분 정보의 예
(아래의 XML 정보의 경우 Query의 Root Tag인 <NP>와 동일한 Tag가 6개가 존재하며, 6개 Tag 각각에 대한 Approximate String Matching 기법의 적용 범위는 점선으로 표시된 네모 공간과 같다.)

```
<XMLDocumentBody>
 <S>
        <NP>Astronomers</NP>
        <VP>
                <V>saw</V>
                <NP>
                        <NP>stars</NP>
                        <PP>
                                <P>with</P>
                                <NP>ears</NP>
                        </PP>
                </NP>
        </VP>
 </S>
 <S>
        <NP>Astronomers</NP>
        <VP>
                <V>saw</V>
                <NP>apples</NP>
        </VP>
 </S>
</XMLDocumentBody>
```

그림 6. 문서의 XML 정보로부터 스트링 매칭 기법을 적용할 XML 정보의 추출

2) 1)번 과정을 통해 XML 문서에서 추출한 정보들과 Query 정보와의 차이 수치를 Approximate String Matching 기법을 적용하여 계산하며, 계산된 차이 수치가 낮을수록 두 정보의 유사도가 높다는 의미가 된다. Approximate String Matching 기법을 적용할 때, 태그의 경우 하나의 여닫는 태그를 문자로 취급하여 계산하며, 태그의 값 정보는 각각의 문자를 나누어 처리한다.

사용자 Query와 XML 문서에서 추출한 정보에 Approximate String Matching 기법을 적용 - Data 1 (Query Data) : star with ears - Data 2 (추출된 Data) : stars	
Data 1	<NP> <NP>star</NP> <PP> <P>with</P> <NP>ears</NP> </PP> </NP>
Data 2	<NP>stars</NP>

그림 7. Query 정보와 추출된 XML 정보의 예

Approximate String Matching은 두 문자열이 얼마나 차이나 나는지를 알려고 할 때 사용하는 스트링 매칭 기법으로 두 문자열이 같아지려면 insertion(문자의 추가), deletion(문자의 삭제), substitution(문자의 변경) 이상 3가지 동작을 각각 몇 번 수행해야 되는지를 계산하여 두 문자열이 같아지게 하기 위해 수행한 3가지 동작의 합계가 두 문자열이 얼마나 차이가 나는지를 나타내는 수치가 된다. 다음은 표1에서 각 동작이 일어난 위치를 몇 군데 지정하여 각각의 동작이 어떤 것인지를 분류한 것이다.

insertion(문자의 추가) : (a), (d)
deletion(문자의 삭제) : (b), (e)
substitution(문자의 변경) : (c), (f)

표1의 위치 좌표 (i,j)는 (0,2)에서 (0,i)까지의 문자열과 (2,0)에서 (j,0)까지의 문자열을 비교한 결과 값을 넣어야 하는 위치이며, 비교 대상이 되는 문자들인 (0,i) 위치의 문자와 (j,0) 위치의 문자가 동일한지의 여부에 따라 계산 방식이 달라진다. 지금 계산해야 하는 문자들이 동일하고 지금 계산해야 하는 Table의 위치가 (i,j)라면, (i-1,j)에 insertion cost를 더한 값, (i,j-1)에 deletion cost를 더한 값, (i-1,j-1)의 값들 중에 가장 작은 값이 (i,j)의 값이 되며, 지금 계산해야 하는 문자들이 동일하지 않고 지금 계산해야 하는 Table의 위치가 (i,j)라면, (i-1,j)의 값에 insertion cost를 더한 값, (i,j-1)에 deletion cost를 더한 값, (i-1,j-1)에 substitution cost를 더한 값들 중에 가장 작은 값이 (i,j)의 값이 된다.

표1에서 Approximate String Matching 기법을 적용할 때 insertion cost, deletion cost, substitution cost의 값을 각각 1로 설정하여 계산하였으며, 각각의 cost를 1로 설정하였을 때의 계산 방식은 다음과 같이 표현된다.

표 1. Query 정보와 추출된 XML 정보에 Approximate String Matching 기법을 적용한 예

		<NP>	s	t	a	r	s	</NP>
	0	1	2	3	4	5	6	7
<NP>	1	0	1 (a)	2	3	4	5	6
<NP>	2	1 (b)	1 (c)	2	3	4	5	6
s	3	2	1	2	3	4	4	5
t	4	3	2	1	2	3	4	5
a	5	4	3	2	1	2	3	4
r	6	5	4	3	2	1	2 (d)	3
</NP>	7	6	5	4	3	2 (e)	2 (f)	2
<PP>	8	7	6	5	4	3	3	3
<P>	9	8	7	6	5	4	4	4
w	10	9	8	7	6	5	5	5
i	11	10	9	8	7	6	6	6
t	12	11	10	9	8	7	7	7
h	13	12	11	10	9	8	8	8
</P>	14	13	12	11	10	9	9	9
<NP>	15	14	13	12	11	10	10	10
e	16	15	14	13	12	11	11	11
a	17	16	15	14	13	12	12	12
r	18	17	16	15	14	13	13	13
s	19	18	17	16	15	14	13	14
</NP>	20	19	18	17	16	15	14	13
</PP>	21	20	19	18	17	16	15	14
</NP>	22	21	20	19	18	17	16	15

```
if( (0,i)의 문자 == (j,0)의 문자 ){
  (i,j)의 값 = min{(i-1,j-1)의 값, (i-1,j)의 값 + 1,
                   (i,j-1)의 값 + 1 };
}
else{
  (i,j)의 값 = min{(i-1,j-1)의 값 + 1, (i-1,j)의 값 +
                   1, (i,j-1)의 값 + 1 };
}
```

표1의 값을 채워나가는 계산은 왼쪽 상단에서부터 채워나가면 된다.

3) 2)번 과정을 통해 얻어진 Query와 XML에서 추출한 정보와의 차이 수치(표 1에서 Approximate String Matching 기법에 의해 계산된 결과는 15이다.)가 일정 값(Threshold)보다 적을 경우 해당 XML 정보를 Query와 관련 있는 정보라고 간주하여 처리한다.
(Threshold를 위한 수식은 다양하게 사용가능하다.)

5. 제안한 알고리즘에 관한 분석

5.1. 알고리즘의 유용성에 관한 분석

본 논문에서 제시한 알고리즘의 유용성은 다음과 같다.

1) 구조에 기반을 둔 정보 검색기법
Parsing 작업의 결과인 Parse Tree는 어떤 단어가 어떤 단어를 수식한다는 정보, 어떤 단어가 어떤 단어보다 먼저 출현했다는 정보(단순히 문장에서의 순서가 아니라 문장의 구조 측면에서의 순서를 의미함.) 등의 구조적 정보를 가지고 있다. 본 논문에서 제시하는 알고리즘은 Parsing의 결과인 Parse Tree의 정보를 정보 검색에 이용함으로써 구조에 기반을 둔 정보 검색기법이라고 말할 수 있다. 또한, 의미 기반의 Parsing 기법에 의해 생성된 Parse Tree에 본 논문에서 제시한 알고리즘을 적용한다면 높은 수준의 의미에 기반을 둔 정보 검색 기능의 구현이 가능할 것이다.
예를 들어 “붉은 옷을 입은 김태희”라는 Query와 관련된 문서를 검색하려고 할 때에는 “김태희”, “붉은 옷”, “입다”라는 단어만 가지고 검색을 하게 되면 정확한 문서를 검색하기가 어려우며 정확한 문서를 검색하기 위해서는 “김태희”, “붉은 옷”, “입다” 간의 관계 정보를 이해해야만 가능하다.

2) 비슷한 단어에 대해 검색이 가능
Approximate String Matching 기법을 사용함으로써 정확하게 일치되는 단어가 아니거나 오타가 발생한 단어에 대해서도 검색이 가능하다.

3) 비슷한 구조를 가진 정보에 대해 검색이 가능
Approximate String Matching 기법과 Parse Tree를 사용함으로써 동일한 구조가 아닌 비슷한 구조를 가진 정보에 대해 검색이 가능하다.

5.2. 알고리즘 구현에 관한 분석

본 논문에서 제시한 알고리즘을 실제 구현함에 있어 고려해야할 사항은 다음과 같다.

1) XML Parser의 이용 가능
Query와 문서 정보는 Parsing에 의해 Parse Tree로 표현되며, Parse Tree를 XML 정보로 변환하여 유사도 측정을 위해 Approximate String Matching 기법을 적용하는 것이 제시한 알고리즘으로 XML 정보의 검색은 표준화된 XML Parser 기법인 DOM이나 SAX를 구현한 XML Parser 프로그램이나 라이브러리가 존재하므로 해당 프로그램이나 라이브러리를 이용한다면 XML 검색을 위한 모듈은 쉽게 개발이 가능하다.

2) 색인 기법에 관한 고려가 필요
본 논문에서 제시한 알고리즘은 하나의 Query와 하나의 문서를 가지고 두 정보 간의 유사도를 측정하는 방법에 대해서만 설명하였으며, 정보 검색을 위한 색인 구조 및 방법에 대해서는 설명하지 않았다. 이 부분에 대해서는 더 많은 연구를 통해 제시한 알고리즘에 최적화된 색인 구조 및 색인 방법을 찾을 계획이며, 이 알고리즘을 문서 집합에서 Query와 연관된 문서를 검색하는 곳에 이용하려고 할 때에는 속도나 성능을 위해서 색인 기법에 대한 고려가 필요하다.

5.3. 알고리즘의 성능에 관한 분석

제안한 알고리즘의 속도라는 측면에서의 성능은 XML Parser의 처리 속도에 Approximate String Matching 기법을 적용하는 동작의 처리 속도를 더한 합이 된다. XML Parser의 처리 속도를 Tag 간의 비교를 하는 동작의 수를 이용하여 계산한다면 XML 정보의 모든 Tag와 Query의 Root Tag와의 동일 여부를 계산해야 하므로 $\ominus(N)$이 된다. Approximate String Matching 기법을 적용하는 동작의 처리 속도는 추출된 Tag의 수와 추출된 정보의 길이에 비례하게 되며, 추출된 Tag의 수를 M이라고 하고 Query 정보의 길이를 Q라고 하며, 추출된 정보의 길이를 L이라고 한다면 Approximate String Matching 기법을 적용하는 동작의 성능은 $\ominus(MQL)$이 된다. 앞의 계산 방식에 따라 계산된 제안한 알고리즘의 속도라는 측면에서의 성능은 다음과 같다.

$$\ominus(N+MQL)$$

N : XML 정보의 Tag의 수
M : XML 정보에서 Query의 Root Tag와 동일하다고 판단되어 추출된 Tag의 수
Q : Query 정보의 길이
L : XML 정보에서 추출된 정보의 길이

6. 결론 및 향후 과제

Parsing 작업의 결과로 얻어진 Parse Tree는 문장에

관한 구조적 정보를 가지고 있는 Tree 정보이다. 이 Tree 정보를 정보 검색에 이용함으로써 구조에 기반을 둔 정보 검색이 가능하다. 논문에서 제시한 구조에 기반을 둔 정보 검색 알고리즘은 파싱 기술과 Approximate String Matching 기법을 이용한다는 특징을 가지고 있다. Parsing 작업의 결과인 Parse Tree를 XML 형태로 변환함으로써 기존에 존재하는 XML 기술을 이용할 수 있다는 장점이 있으며, Query와 문서 정보를 Parsing한 결과에 대해 Approximate String Matching 기법을 이용한 유사도 측정 방법을 사용하여 구조에 의한 정보 검색, 약간 철자가 다른 단어나 오타가 발생한 단어에 대해서도 처리가 가능한 비슷한 단어에 대한 정보 검색, 비슷한 구조에 대한 정보 검색 등이 가능하다. 향후 과제로는 알고리즘의 검색 성능을 측정하기 위한 시스템 구축 및 실험을 할 계획이며, 제시한 알고리즘에 적합한 색인 구조에 관한 연구를 진행할 계획이다.

7. 참고문헌

[1] Berners-Lee, T., Hendler, J., Lassila, O., The Semantic Web, Scientific American, Vol. 284 (4). (2001) 34-43.
[2] The World Wide Web Consortium, http://w3c.org.
[3] RDF contents in W3C, http://www.w3.org/RDF/.
[4] OWL contents in W3C, http://www.w3.org/2004/OWL/.
[5] XML contents in W3C, http://www.w3.org/XML/.
[6] Alberto Apostolico, Zvi Galil, Pattern matching algorithms, ISBN:0-19-511367-5, pages 185-199, 1997.
[7] S.Brin and L.Page. The anatomy of a large-scale hypertextual Web search engine. Computer Networks and ISDN Systems, 30(1-7):107-117, 1998.
[8] J. M. Kleinberg. Authoritative sources in a hyperlinked environment. Proceedings of ACM SIGIR '98, pages 668 - 677, 1998.
[9] Qinghua Zou, Shaorong Liu, Wesley W.Chu, Using a compact tree to index and query XML data, Proceedings of the thirteenth ACM international conference on Information and knowledge management, pages 234-235, 2004.
[10] Chin-Wan Chung, Jun-Ki Min, Kyuseok Shim, APEX: an adaptive path index for XML data, Proceedings of the 2002 ACM SIGMOD international conference on Management of data, pages 121-132, 2002.
[11] Jun-Ki Min, Myung-Jae Park, Chin-Wan Chung, XPRESS: a queriable compression for XML data, Proceedings of the 2003 ACM SIGMOD international conference on Management of data, pages 122-133, 2003.