함수 변환과 FFT에 의한 XML 문서의 구조 비교

이 호 석
호서대학교 공과대학 뉴미디어공학과
hslee@office.hoseo.ac.kr

XML Document Structure Comparison based on Function Transform and FFT

°Ho Suk Lee
New Media Department, College of Engineering, Hoseo University

요 약

XML 문서의 유사성을 비교하는 연구는 XML 문서의 저장및 검색에 유용하기 때문에 많은 연구가 진행되었다. XML 문서의 유사성 연구는 크게 edit-distance를 이용하는 방법, 문서의 그래프 모델을 이용하는 방법, 문서의 매트릭스 모델을 이용하는 방법 등이 있다. 최근에는 문서를 encoding 하고 푸리에 변환을 이용하는 방법이 보고되었다. 본 논문에서는 XML 문서를 함수로 변환하고 FFT를 적용하여 문서의 구조적 유사성을 비교하는 새로운 방법을 제안한다. 제안하는 방법은 JAXP로 구현하였으며 XML 문서의 구조를 분석하여 함수로 변환하였다. 그리고 함수에 FFT를 적용하여 XML 문서의 구조적 유사성을 비교하였다. FFT 비교 결과는 XML 문서의 함수 변환이 적합한 것임을 보여주었으며 비교 결과는 예상된 결과를 보여주었다.

1. 서론

1.1 연구 동기

XML(Extensible Markup Language) 문서는 시스템 자료를 나타내는 표준이다[1]. XML 문서의 유사성을 파악하고 비교하는 것은 XML 문서의 저장과 검색에 매우 유용하다. 본 논문에서는 XML 문서를 함수로 변환하고 FFT(Fast Fourier Transform)를 적용하여 비교하는 새로운 방법을 제안한다. XML 문서를 JAXP(Java API for XML Processing)로[2] 분석하여 함수로 변환하였으며 변환된 함수에 FFT를 적용하여 문서를 비교하였다.

1.2 관련 연구

참고문헌 [3]에서는 이산 푸리에 변환(DFT, Discrete Fourier Transform)을 사용하여 time series를 주파수 영역으로 변환하여 질의어의 유사성을 비교한다. 푸리에 변환을 사용하는 중요한 이유는 소수의 저주파 성분이 중요한 요소를 표현하기 때문에 질의어 비교가 용이하기 때문이다. 참고문헌 [4]에서는 어휘뿐만이 아니고 구(phrase)를 사용하여 웹 문서의 유사성을 비교한다. 참고문헌 [5]에서는 S-GRACE 라고 불리는 계층적인 알고리즘을 제안하였다. 이 알고리즘은 s-graph 라고 불리는 구조 그래프(structure graph)를 사용하여 XML 문서간의 거리를 측정하였다. 참고문헌 [6]에서는 edit-distance를 사용하여 XML 문서의 유사성을 비교하였다. 참고문헌 [7]에서는 이전 버전과 현재 버전의 XML 문서 사이의 차이를 계산하는 효율적인 방법을 제안하였다. 참고문헌 [8]에서는 XML 문서의 태그(tag)들을 encoding하여 숫자로 표현한 다음에 숫자들의 발생 빈도를 조사하여 문서의 유사성을 측정하는 방법을 제안하였다. 즉, [8]에서는 XML 문서의 태그(tag)인 엘리먼트와 에트리뷰트를 encoding 함수를 정의하여 숫자로 변환한다. 이렇게 하면 XML 문서는 일련의 숫자들의 순서가 되며 이 숫자들을 디지털 신호의 신호 값으로 간주한다. 다음에 이 신호 값들에 대하여 이산 푸리에 변환을 수행하여 주파수 영역에서 XML 문서의 구조적 유사성을 비교하였다.

2. 본론

2.1 XML 문서 변환

XML 문서의 유사성을 검사하기 위해서는 XML 문서의 적합한 모델링이 필요하다고 생각한다. XML 문서의 모델링에는 어휘 모델링, 구조 모델링, 의미 모델링을 생각할 수 있다. 본 논문에서는 XML 문서의 구조를 함수로 변환하여 모델링하는 방법을 제안한다. 문서를 time series로 모델링하는 방법보다는 개념적으로 더 적합하다고 생각한다. 왜냐하면 XML 문서의 엘리먼트를 임펄스로 볼 수는 없기 때문이다. 우선 문서의 어휘 모델링으로서는 엘리먼트의 집합과 시퀀스 그리고 에트리뷰트의 집합을 생각한다. 여기서, 집합은 중복된 원소를 포함하지 않고 오직 시작 태그(start tag)만을 포함한다. 시퀀스는 원소의 나열로서 중복을 허용하며 시작 태그만을 포함하는 것으로 정의한다. 엘리먼트가 에트리뷰트를 가지고 있을 때에는 엘리먼트 시퀀스에 있는 엘리먼트에 관련된 에트리뷰트를 가리키는 주소를 표기하여 에트리뷰트를 가리킬 수 있도록 한다. XML 문서의 구조를 모델링하는 방법으로 문서 엘리먼트의 시퀀스를 함수로 변환하는 방법을 생각하였다. 예를 들어, 엘리먼트 시퀀스를 X 축으로 간주하고 시퀀스에 포함된 각 엘리먼트를 X 축의 값으로 간주한다. 다음에 엘리먼트의 계층레벨(nesting level)에 초점을 맞추고 계층레벨을 Y 값으로 간주하여 함수로 변환하는 것이다. 왜냐하면 계층레벨이 <element>와 </element>의 쌍에 의하여 문서의 구조를 나타낸다고 할 수 있기 때문이다.

함수로 변환하는 방법을 간략하게 설명하면 다음과 같다. (1) 시퀀스에 있는 첫 번째 엘리먼트를 X_1 위치에 놓는다. 첫 번째 엘리먼트의 Y 값 Y_1은 1로 한다. (2) 다음 엘리먼트 X_2를 조사한다. 만약 다음 엘리먼트의 nesting level이 동일하면 이전의 Y 값을 현재 엘리먼트 X_2의 Y 값으로 사용한다. (3) 만약 다음 엘리먼트의 nesting level이 하나 증가되어 있으면 X_2 의 Y 값은 Y_1 + 1로 한다. (4) 시퀀스에서 끝 태그(end tag)를 만나면 관련된 시작 태그를 시작할 때의 Y 값

을 해당 X_n 위치의 Y 값으로 사용한다. (5) 이 과정을 반복하여 수행한다. (6) 마지막으로 시퀀스에 있는 모든 엘리먼트를 사용하면 함수 변환 과정을 종료한다.

2.2 JAXP에 의한 구현

XML 문서의 함수 변환은 JAXP를 사용하여 구현하였다. 다음은 소스코드의 일부분을 보여준다.

```
public class XMLtree1 {

    public XMLtree1(File f1) {
        try {
            DocumentBuilderFactory factory
                = DocumentBuilderFactory.newInstance();
            DocumentBuilder builder
                = factory.newDocumentBuilder();
            Document doc = builder.parse(f1);
            f1( );
            f2( );

        } catch(Exception e) {
            e.printStackTrace();
        }
    }
    public void f1(Node node) {
        int type = node.getNodeType();
        switch(type) {
            case Node.DOCUMENT_NODE :

                Document doc = (Document) node;
                f1(doc.getDocumentElement());
                break;
            case Node.ELEMENT_NODE :

                NodeList clist1 = node.getChildNodes();
                for(int i=0; i<clist1.getLength(); i++) {
                    f1(clist1.item(i));
                }
                break;
        }
    }
    public void f2( ) {
        Complex[] cx = new Complex[];
        Complex[] cy = f3(cx);
    }
    public Complex[] f3(Complex[] x) {
        Complex[] e = f3(even);
        Complex[] o = f3(odd);
        return y;
    }
    public static void main(String[] args) {
        // car3.xml, car4.xml, car5.xml
        File file1 = new File("car3.xml");
        XMLtree1 xtree1 = new XMLtree1(file1);
    }
}
```

문서의 의미는 엘리먼트 태그와 태그가 포함한 문자열 그리고 에트리뷰트와 에트리뷰트가 나타내는 문자열을 비교하는 것으로 모델링한다. 비교 대상이 되는 XML 문서에 대하여 태그 배열과 태그 문자열 그리고 에트리뷰트 배열과 에트리뷰트 문자열을 비교함으로서 문서의 의미의 유사성을 비교한다.

2.3 실험과 결과

다음은 실험 대상 문서 car3.xml, car4.xml, car5.xml 그리고 각각의 FFT[9] 출력 결과를 보여준다. car3.xml 문서와 car4.xml 문서는 서로 비슷하지만 약간 다르다. car5.xml 문서는 구조가 좀 더 다르다.

car3.xml

```
<?xml version="1.0" encoding="euc-kr" standalone="no"?>
<!DOCTYPE car SYSTEM "car3.dtd">
<car>
   <number>서울 1가 1234</number>
   <model maker="Hyundai">
      <model_type>
         <model_name>EF 소나타</model_name>
         <brake>hand_brake</brake>
         <seat>5_seats</seat>
         <ventilation>air_conditioned</ventilation>
         <suspension>
            <type>McPherson</type>
         </suspension>
         <transmission>
            <level>4</level>
            <mode>centronic</mode>
         </transmission>
      </model_type>
      <engine type="gasoline">
         <hp>120 마력</hp>
         <cylinder>4 cylinder</cylinder>
      </engine>
      <body>chassis_body</body>
   </model>
   <owner>
      <name>홍길동</name>
      <gender>male</gender>
      <age>32</age>
      <phone>02-123-1345</phone>
      <addr>
         <street>서울시 동대문구 신당동</street>
         <zip_code>123-456</zip_code>
         <apartment>한양APT 801-1010</apartment>
      </addr>
      <company>
         <title>현대자동차</title>
         <dept>엔진연구소</dept>
         <position>연구1팀장</position>
         <desc1>새로운 가솔린 엔진 개발</desc1>
      </company>
   </owner>
</car>
```

car4.xml

```
<?xml version="1.0" encoding="euc-kr" standalone="no"?>
<!DOCTYPE car SYSTEM "car4.dtd">
<car>
   <number>서울 1가 1234</number>
   <model maker="Hyundai">
      <model_type>
         <model_name>EF 소나타</model_name>
         <brake>hand_brake</brake>
         <seat>5_seats</seat>
         <ventilation>air_conditioned</ventilation>
         <suspension_type>McPherson</suspension_type>
         <transmission>
            <level>4</level>
            <form>bar</form>
            <mode>centronic</mode>
         </transmission>
      </model_type>
      <engine type="gasoline">
         <hp>120 마력</hp>
         <cylinder>4 cylinder</cylinder>
      </engine>
      <max_speed>120</max_speed>
      <body>chassis_body</body>
   </model>
   <owner>
      <name>홍길동</name>
      <gender>male</gender>
      <age>32</age>
      <phone>02-123-1345</phone>
      <addr>
         <street>서울시 동대문구 신당동</street>
         <apartment>(123-456)한양APT 801-1010</apartment>
      </addr>
      <company>
         <title>현대자동차</title>
         <dept>엔진연구소</dept>
         <position>연구1팀장</position>
         <desc1>새로운 가솔린 엔진 개발</desc1>
      </company>
   </owner>
</car>
```

car5.xml

```
<?xml version="1.0" encoding="euc-kr" standalone="no"?>
<!DOCTYPE car SYSTEM "car5.dtd">
<car>
   <owner>
      <name>홍길동</name>
      <gender>male</gender>
      <age>32</age>
      <phone>02-123-1345</phone>
      <hand_phone>011-1111-2222</hand_phone>
      <address>서울시 동대문구 신당동 신라APT 801-1010
                                          </address>
      <company>현대자동차</company>
      <dept>엔진연구소</dept>
      <position>연구1팀장</position>
      <desc1>새로운 가솔린 엔진 개발</desc1>
   </owner>
   <model maker="Hyundai">
      <model_type>
         <car_number>서울 1가 1234</car_number>
         <model_name>EF 소나타</model_name>
         <brake>hand_brake</brake>
         <seat>5_seats</seat>
         <ventilation>air_conditioned</ventilation>
         <suspension>McPherson</suspension>
         <transmission>automatic</transmission>
         <level>4</level>
         <mode>centronic</mode>
```

```
        <body>chassis_body</body>
        <light>halogen</light>
        <roof>sliding_roof</roof>
    </model_type>
    <engine type="gasoline">
        <hp>120 마력</hp>
        <cylinder>4 cylinder</cylinder>
        <max_speed>120</max_speed>
        <injection>VGT</injection>
    </engine>
</model>
</car>
```

car3.xml 문서에 대한 FFT 결과

```
110.0 + 0.0i
-5.22985608434456 - 5.552714433067682i
-13.246193338372112 + 7.6583258058236811i
-4.516612358269251 + 0.030396384483083244i
-6.121320343559642 + 2.2928932188134521i
-5.096282159399493 + 2.2101417725099117i
-3.582722080200031 + 0.8210291114869468i
-1.2361763861534691 - 4.7407376678197011i
0.9999999999999999 + 1.0i
-0.9160645488239562 - 4.6390135272509241i
-3.0030643574265943 + 0.5783884243676611i
1.8616490241296728 - 3.9967781467192087i
-1.8786796564403572 - 3.7071067811865471i
-0.24875450646092778 - 0.5196692852536587i
3.8319797759990175 - 0.5843148812956014i
-0.6179029815880108 + 0.20586395699347593i
0.0 + 0.0i
-0.6179029815880122 - 0.20586395699347548i
3.8319797759990166 + 0.5843148812956023i
-0.24875450646092823 + 0.5196692852536593i
-1.8786796564403576 + 3.7071067811865481i
1.8616490241296772 + 3.9967781467192105i
-3.003064357426595 - 0.5783884243676611i
-0.9160645488239577 + 4.6390135272509241i
1.0 - 1.0i
-1.2361763861534711 + 4.7407376678197011i
-3.582722080200031 - 0.8210291114869459i
-5.096282159399493 - 2.2101417725099135i
-6.121320343559643 - 2.2928932188134531i
-4.516612358269251 - 0.030396384483084238i
-13.246193338372112 - 7.6583258058236685i
-5.229856083434563 + 5.552714433067681i
```

car4.xml 문서에 대한 FFT 결과

```
109.0 + 0.0i
-5.279264619659338 - 6.664187075927931i
-11.888768303211346 + 8.689855935831687i
-5.152659471734624 + 0.044770517356488961i
-6.828427124746419 + 1.5857864376269046i
-5.295476132690038 + 2.5004152846662621i
-1.0204582207319457 + 1.3113631413487552i
-3.9340333691222555 - 5.0665153522928481i
1.0 + 0.0i
0.6994002338524368 - 2.3516652819713161i
-2.7369010921487681 - 1.7597046705167211i
-0.5522829323321923 - 2.6241124342704671i
-1.1715728752538097 - 4.4142135623730951i
3.0004185367571985 + 0.5770970479121409i
-0.3538723839079392 - 2.3812118760337864i
0.5138977549291592 + 1.7075172438859818i
1.0 + 0.0i
0.5138977549291583 - 1.7075172438859805i
-0.3538723839079392 + 2.3812118760337881i
3.0004185367571976 - 0.5770970479121398i
-1.1715728752538102 + 4.4142135623730961i
-0.5522829323321932 + 2.6241124342704683i
-2.7369010921487695 + 1.7597046705167203i
0.6994002338524352 + 2.3516652819713175i
1.0 + 0.0i
-3.9340333691222575 + 5.0665153522928471i
-1.0204582207319461 - 1.3113631413487545i
-5.295476132690038 - 2.5004152846662624i
-6.828427124746419 - 1.5857864376269053i
-5.152659471734625 - 0.044770517356489625i
-11.888768303211345 - 8.689855935831691i
-5.279264619659344 + 6.664187075927931i
```

car5.xml 문서에 대한 FFT 결과

```
106.0 + 0.0i
-8.503382585705982 + 8.968443237870646i
-3.357425035160765 - 1.3826834323650896i
-7.323258742553076 + 1.7957581909198637i
-3.121320343559643 + 1.2928932188134521i
-7.026154046430754 - 1.8718433064509914i
-4.562263859468364 + 0.076120467488713041i
-1.9617818640177158 - 3.7375220616846394i
-3.0 - 1.0i
-0.010870643856434667 - 0.10800730431810668i
-2.266163265277825 - 1.9238795325112871i
-1.4704517162347335 - 0.7115948196240989i
1.1213203435596428 - 2.7071067811865471i
0.9914373804723762 - 1.5292746964918151i
2.185852159906955 + 0.6173165676349106i
1.3044702183263226 + 0.11267662099861031i
0.0 + 0.0i
1.3044702183263226 - 0.11267662099860765i
2.185852159906955 - 0.6173165676349099i
0.9914373804723744 + 1.5292746964918154i
1.1213203435596422 + 2.7071067811865481i
-1.470451716234734 + 0.7115948196240995i
-2.266163265277826 + 1.9238795325112867i
-0.010870643856435666 + 0.10800730431810779i
-3.0 + 1.0i
-1.9617818640177718 + 3.7375220616846391i
-4.562263859468365 - 0.076120467488713481i
-7.026154046430757 + 1.8718433064509894i
-3.121320343559643 - 1.2928932188134528i
-7.323258742553076 - 1.7957581909198643i
-3.357425035160765 + 1.3826834323650894i
-8.503382585705598 - 8.968443237870648i
```

FFT 결과는 함수를 시간 공간에서 주파수 공간으로 변환한 결과이다. 예를 들어, car3.xml의 두 번째 값인 -5.23-5.55i를 예를 들면, 그 크기는 7.626이고 두 성분 사이의 각도는 231.9도이다. 따라서 이 값들을 비교하여 문서 구조의 유사성을 파악할 수 있다. 즉, car3.xml의 1, 2, 3번째 FFT 결과는 110.0+0.0i, -5.23-5.55i, -13.25+7.66i이다. car4.xml의 결과는 109.0+0.0i, -5.23-6.66i, -11.80+8.69i이며 서로 비슷하지만 약간 다르다. 반면에 car5.xml의 결과는 106.0+0.0i, -8.50+8.97i, -3.35-1.38i로서 앞의 두 결과와는 좀 더 다르다. 이것은 car5.xml 문서의 구조가 좀 더 다르기 때문이다. 이것은 문서의 구조를 FFT 결과가 그대로 반영한다고 할 수 있다. 따라서 XML 문서의 함수 변환은 적합하며 FFT 결과를 바탕으로 문서의 구조적 유사성을 효과적으로 비교할 수 있다는 것을 알 수 있다.

3. 결론

본 논문에서는 XML 문서를 함수로 변환하고 FFT를 적용하여 문서의 구조적 유사점을 효과적으로 비교하는 방법을 제안하였다. 새로운 방법은 기존의 방법에 비하여 개념적으로 자연스럽고 예상된 결과를 보여주었다. 다음 연구로는 다양한 문서에 대하여 실험을 실시할 계획이다. 그리고 FFT 결과를 효과적으로 비교할 수 있는 메트릭 공간을 개발할 계획이다.

참고문헌

[1] Elliotte Rusty Harold, XML 1.1 Bible (3rd ed.), John Wiley & Sons, 2003.

[2] Elliotte Rusty Harold, Processing XML with Java, Addison-Wesley, 2003.

[3] Rakesh Agrawal, Christos Faloutsos, Arun Swami, "Efficient Similarity Search in Sequence Databases," Proc. of the 4th Int'l Conf. of Foundations of Data Organization, pp.69-84, 1993.

[4] Khaled M. Hammouda, Mohamed S. Kamel, "Efficient Phrase-Based Document Indexing for Web Document Clustering, IEEE Trans. on Knowledge and Data Engineering, Vol. 16, No. 10, pp.1279-1296, 2004.

[5] Wang Lian, David Wai-lok Cheung, Nikos Mamoulis, Siu-Ming Yiu, "An Efficient and Scalable Algorithm for Clustering XML Documents by Structure," IEEE Trans. on Knowledge and Data Engineering, Vol. 16, No. 1, pp82-96, Jan. 2004.

[6] Andrew Nierman, H. V. Jagadish, "Evaluating Structural Similarity in XML Documents," Proc. of the 5th Int'l Workshop on Web and Databases, 2002.

[7] Kyong-Ho Lee, Yoon-Chul Choy, Sung-Bae Cho, "An Efficient Algorithm to Compute Differences between Structured Documents," IEEE Trans. on Knowledge and Data Engineering, Vol. 16, NO. 8, pp.965-979, Aug. 2004.

[8] Sergio Flesca, Giuseppe Manco, Elio Masciari, Luigi Pontieri, Andrea Pugliese, "Fast Detection of XML Structural Similarity," IEEE Trans. on Knowledge and Data Engineering, Vol. 17, No. 2, pp.160-175, Feb. 2005.

[9] Alan V. Oppenheim, Ronald W. Schafer, John R. Buck, Discrete-Time Signal Processing(2nd ed.), Prentice-Hall, 1999.