ベンチマークのための実データを利用したスケーラブルなXMLデータ生成器の提案

原崎真奈美† 横山 昌平† 福田 直樹† 石川 博†

† 静岡大学情報学部情報科学科 〒 432–8011 静岡県浜松市中区城北 3–5–1
E-mail: †cs05078@s.inf.shizuoka.ac.jp, ††{yokoyama,fukuta,ishikawa}@inf.shizuoka.ac.jp

あらまし 近年，XML 形式で記述されたデータの急速な増加に伴い，大規模な XML データを扱う技術においてスケーラビリティの評価が特に重要となってきた．スケーラビリティの評価実験では，データの規模の増大に対してアプリケーションがどの程度適応できるかを調査するため，様々なサイズのデータセットが必要になる．XMark や XBench では，あらかじめ用意されたスキーマに従った任意のサイズのデータを生成するが，それらはあくまで人工データであり，実データでは無い．XML が様々な分野で多様な使われ方をしている現状を鑑みると，その性能検証で使用するテストデータは実データ，あるいは実データと同じ特徴を持ったデータである事が重要だと考える．しかしながら，性能検証に利用するための様々なサイズの実データをあらかじめ用意することは容易ではない．そこで本研究では，実データを解析することでしか得られない特徴，例えば，要素の偏りや要素の並び順，文字列などを利用することで，一つの実データから，その特性を持つ任意のサイズのデータセットを効率的に生成する手法を提案する．
キーワード XML，データ生成器

Proposal of Scalable XML Data Generator Using Real Data for Benchmark

Manami HARAZAKI†, Shohei YOKOYAMA†, Naoki FUKUTA†, and Hiroshi ISHIKAWA†

† Department of Computer Science, Faculty of Informatics, Shizuoka University Johoku 4–5–6, Nakaku, Hamamatsu-shi, Shizuoka, 432–8011 Japan
E-mail: †cs05078@s.inf.shizuoka.ac.jp, ††{yokoyama,fukuta,ishikawa}@inf.shizuoka.ac.jp

Abstract Recently, as the data described by XML increases rapidly, therefor the scalability has become especially desirable property in the technology that handles large-scale XML data. To investigate if the application is suitably efficient and practical when applied to large scale situations, various size datasets are needed. Though XMark and XBench generate the arbitrary size dataset according to the schema prepared beforehand, it isn't real data but absolutely artificial data. Considering the current state for which XML is used in various fields, it is thought that preferable test data used by the performance verification is data with the same feature as real data or real data. However, it is not easy to prepare real data of various sizes to use for the performance verification. In this paper we propose a novel XML data generator by using features obtained only by analyzing real data for benchmark. We use the occurrence rate of element, the order of elements and character string of element value as the features.
Key words XML, Data Generator

1. はじめに

近年，XML(Extensible Markup Language) は標準的なデータ記述フォーマットとして急速に普及するとともに，様々な分野で XML が利用されるようになり，XML で記述された大規模データを対象とした関連技術の研究開発が盛んに行われるようになった．

データサイズの拡大に伴い，XML データを扱う技術におけるスケーラビリティは特に重要な評価指標の一つとなってきている．スケーラビリティの評価実験はデータ規模の増大に対するシステムの適応力の検証が目的であるため，様々なサイズの実データ，あるいは実データの特徴を持ったデータセットを用意する必要がある．テストデータとしてシステムの対象となる実データを使用することが最も理想的であるが，実データのサ

イズを拡張することは出来ないため，任意のサイズのデータセットを複数必要とする評価実験には適していない．データサイズを拡張出来ない実データに対し，XMark [1] や XBench [2] のような XML ベンチマークを利用すると，あらかじめ用意されたスキーマに従った任意のサイズのデータを生成，利用することが可能であるが，それらはあくまで人工データであり，実データではないため，アプリケーションの対象として常に最適なデータセットが得られるとは限らない．

大規模な XML データが増加の一途をたどる中，開発技術の正確な性能評価を行うために，対象となる実データに近いデータ内容を持つ任意のサイズの XML データセットの効率的な生成手法の必要性は高まってくる．そこで本研究では，実データを解析することでしか得られない特徴，例えば，要素の偏りや要素の並び順，文字列などを利用することによって，利用者が指定する実データの特性を持った任意のサイズのデータセットを生成する手法を提案する．

2. 関 連 研 究

大規模な XML データを扱う技術開発においてスケーラビリティの評価実験が行われている．スケーラビリティの評価を正確に行うためには，実データの特性を持つ様々なサイズのデータセットを用いた実験，検証を行う必要がある．実データでは，サイズの拡張が出来ないため様々なサイズのデータセットを用意することは難しい．XML ベンチマークである XMark, XBench では，任意のサイズのデータセットを用意することが出来るが，データはあくまで人工的に作られているため，実データに近いコンテンツ，特性を持ったデータセットが得られるとは限らない．

本手法では，実データを解析することによって得られる要素の割合や要素の並び順，要素値の文字列のような実データの特徴となる情報を利用することによって，アプリケーションの対象とする実データに近いコンテンツ，実データの特性を保持したデータセットを生成している．

すでに研究開発され広く利用されている XML データ生成器として，ToXgene [3] や Cohen の手法 [4] などが挙げられる．ToXgene とは，利用者がパラメータを入力したテンプレートから生成データに関する情報（データ構造，深さ，要素値のタイプ，要素数など）を読み込み，それに従った XML データを生成する手法である．また Cohen の手法とは，要素数や要素値を指定するために利用者が作成した XML 文書例とアプリケーションの対象データの DTD(Document Type Definition) を併せて利用することによって XML データを生成する手法である．これら既存の XML データ生成器は利用者側で生成データの構造，特徴をある程度指定することが出来る．生成するデータセットの特徴を利用者が指定するということは，実データの特性を持ったデータセットを生成したい場合，実データが持つ特性を利用者が把握していなければならないが，実データの特性を完全に理解し，それを生成するデータに反映させることは容易ではない．特に，各要素の出現割合や並び順のような実データ全体を分析してみなければ発見できないような特徴を生成データに反映させることは難しい．また，既存の XML データ生成器の場合，データセットを作成するまでにテンプレートへのパラメータの入力や構造を指定する文書例の作成など利用者側での手間が多い．

本手法では，利用者が実データの特性を正確に認識していなくても，アプリケーションの対象となる実データを一つ与えるだけで，簡単かつ効率的に実データの特性を保持した任意のサイズのデータセットを生成することが可能である．

3. 提 案 手 法

XML データセットを生成する上で，XML データセットの特徴抽出，データ生成の二段階で処理を行っていく．一つの実データを基に複数のデータセットを生成するとき，データ生成の度に特徴抽出するという無駄な処理を省くため，特徴抽出器とデータ生成器を別々に用意している．また，特徴情報を一度 XML データとして出力させることで，大規模な XML データを持っていなくても，サイズの小さい特徴情報のみ保持していれば任意のサイズのデータ生成が出来るようになる．

実際にアプリケーションとして提供する際，データ特徴抽出器，データ生成器と共に，様々なタイプ，また抽出時期の異なる特徴情報データも公開し，誰でも利用出来るようにしておきたいと考えている．XML データ特徴抽出器が様々な分野の人たちに利用されれば，多様な実データから取り出された特徴情報データを大勢の利用者の間で共有することが出来る．また，実データが更新される度，データサイズの大きい実データを保存しておかなくても，サイズの小さい特徴情報ファイルであれば，古いデータから最新のデータまで，あらゆる時期の実データの特徴情報を保存しておくことが容易となり，様々な時期の特性を持ったデータセットを生成することが可能である．

本章では，実データからの特徴情報抽出手法，特徴情報を利用した任意のサイズのデータセット生成手法について説明する．

3.1 XML データ特徴抽出手法

本手法で生成データセットに反映させる実データの特徴量は，データの構造，各要素の出現割合の偏り，要素の出現確率，要素の分布傾向，要素値の文字列である．そのために必要となる情報を実データから抽出する．そこで，XML データ特徴抽出器では，実データから以下のような特徴量を抽出し，XML 形式のデータとして出力する．

- データ構造
- 要素の出現回数
- 要素の出現確率
- 要素の出現順序
- 要素値の文字列

これらの特徴量を種類ごとに別々のファイルへ出力させると，データ生成時に読み込むファイル数が増え，計算量が増加することが予想される．そこで，一つのファイル内に出来るだけ多くの情報を組み入れるため，データ構造，要素の出現回数，要

```
DataGuide G の生成{
   Nodes(G)={{root}}
   Edges(G) =φ
   currentNode = root
    stack s;
   while (文書の解析が終わるまで){
      if(<a>を見つけた){
         Edges(currentNode) に<a>を加える;
         currentNode = a;     s.push(a);
      }else if(</a>を見つけた){
         currentNode = s.pop();
      }
   }
}
```

図 1　構造取得アルゴリズム

```
<root>
  <A  count="A 要素出現回数">
    <C  count="A 直下の C 要素出現回数">
      A 直下の C 要素出現確率
    </C>
  </A>
  <B  count="B 要素出現回数">
    <C  count="B 直下の C 要素出現回数">
      B 直下の C 要素出現確率
    </C>
  </B>
</root>
```

図 3　構造特徴出力形式

素の出現確率をまとめて一つの XML 形式ファイルに出力させ，データ生成時における特徴情報データの読み込み処理の高速化を図る．したがって，特徴抽出器から出力される特徴データは，データ構造ファイル，要素の出現順序ファイル，要素値の文字列ファイルの三種類である．次節では，各データの記述形式および，特徴抽出手法の詳細について説明する．

3.1.1　データ構造の抽出

データセットを生成するためには，まず，基準となる実データ全体の構造を取得する必要がある．また XML データ全体の構造概要を把握できていないと効率的な情報抽出は難しい．本手法では，実データの構造概要を知る手掛かりとして，Goldman らの DataGuide [5] のアルゴリズムを利用する．図 1 に DataGuide を参考にした構造取得アルゴリズムを示す．DataGuide とは，情報源において共通のパスを持つノードを一つのノードに集約し，半構造データの現時点での構造をラベル付き有向グラフで表現するものである．

DataGuide で作られる有向グラフの一部を図 2 に示す．作り出される有向グラフから，情報源とした実データ内に含まれる要素名，および各要素が持つ子要素名を読み取ることができる．

DataGuide によるデータ構造の取得と同時に，各要素の出現回数および出現確率を算出する．これらの情報は，各要素の出現頻度の偏り，親要素から各子要素への経路を通る確率を生成データに反映させるために抽出する．構造データに各要素の出現回数および出現確率を付与したデータを XML 形式で出力する．データ記述形式を図 3 に示す．構造特徴データの要素の属性値 count は要素の出現回数を，要素値はその要素の出現確率を示している．出現確率とは，その要素が親要素の下に現れる確率である．そのため同じ要素でも親要素の種類によって出現

```
<root>
  <A>要素 A が連続して現れる回数（≧ 1）</A>
  <B>要素 B が連続して現れる回数（≧ 1）</B>
  <A>要素 A が連続して現れる回数（≧ 1）</A>
</root>
```

図 4　要素順序情報出力形式

確率は異なる．例えば，図 3 の親要素 A の子要素 C の出現確率は，式 1 で求められる．

$$A \text{ 直下の } C \text{ 出現確率} = C \text{ を持つ } A \text{ の数}/A \text{ の総数} \cdots (1)$$

3.1.2　データの順序情報の抽出

要素の出現順序もまた実データを解析することでしか得られない特徴量の一つである．例えば，検索システムのスケーラビリティの検証実験をする際，本手法で生成したデータセットを用いて検索時間を測定するとき，異なる種類の要素がランダムに現れるようなデータセットと各要素がまとまって現れるようなデータセットとでは，検索時間に差が生じる可能性がある．基盤となる実データの要素の出現順序が厳密に決まっているのならば，その順序性を反映させたデータセットである必要がある．また，生成データに求める実データとの類似度の高低は利用者によって異なるが，要素の分布の仕方が異なるデータセットが用意出来れば，要素の分布傾向を反映させる度合いを利用者側で選択出来るようになり，柔軟に利用者が求めるデータセットを生成することが出来る．

構造データ同様，要素の出現順序に関する情報データもまた XML 形式で出力する．データ記述形式を図 4 に示す．要素の順序は，実データの要素が現れる順番をそのまま出力しているため，同じ要素が連続して現れることが一度も起こらないようなデータセットの場合，実データと変わらないデータサイズのファイルが出力されるという問題点がある．

この問題に対する解決策として，要素の出現順序を完全に抽出するのではなく，要素の分布傾向を抽出し，それを利用したデータセット生成も行えるよう改良を加えた．完全な並び順ではなく，大まかな分布の偏りを反映させるため，データ全体を分割し，要素ごとにクラス内の出現割合から各要素のデータ全体における分布傾向を把握する．ここでいうクラスとは，データセット全体を分割した 1 つの集合のことである．各要素が各クラスに出現する割合は，式 2 で算出し，データの分割数は利用者側で任意に指定出来るものとする．

$$f(x,i) = (i \text{ クラス内の } x \text{ 要素数})/(x \text{ 要素数}) \cdots \qquad (2)$$

分割数が大きければ大きいほど，厳密な分布傾向を得ることが出来るため，生成データを利用するシステムで要素の分布傾向をどの程度反映させる必要があるのかを利用者が判断し，分割数を指定してもらうことになる．この分布データを利用して，実データの持つ各要素の分布の偏りを加味したデータセットを生成している．これによって，様々な分布のデータセットを生成することが可能となり，実データとの類似度を利用者側の必要に合わせてより柔軟にデータセットを生成することが出来る．

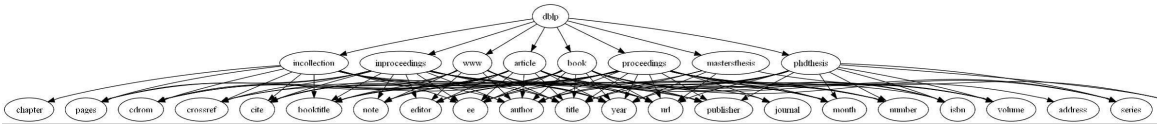

図 2　DataGuide で生成される木構造例

```
SpaceSaving(k){
    T=φ;
    for(i){
        if(i ∈ T){
            c[i]=c[i]+1;
        }else if(|T|<k-1){
            T=T ∪{i};
            c[i]=1;
        }else{
            j = min(T);
            c[i] = c[j]+1;
            T =T ∪{i}/{j};
        }
    }
}
```

図 5　頻出要素値取得アルゴリズム

```
<root>
  <A  item="獲得した A の要素値総数">
    <data  id="0">要素値（文字列）</data>
    <data  id="1">要素値（文字列）</data>
  </A>
</root>
```

図 6　頻出要素値出力形式

3.1.3　要素値の抽出

本節では，データ生成時に各要素の要素値として代入する文字列を取得することについて述べる．一般的な XML ベンチマークでは，要素値に特に意味のない文字列やオークションデータからランダムに取り出した文字列を代入しているため，常にアプリケーションの対象として適切な XML データを生成出来ているとは限らない．本手法では，実データ解析時に各要素の要素値の文字列を取得し，この文字列を生成データの要素値に代入することで，生成データのコンテンツをより実データに近付ける．

利用者によって生成するデータセットに求める要素値の種類の豊富さは異なると考えられるため，各要素の獲得要素値数の上限を利用者が指定出来るものとする．また，実データに近いデータを生成するために，頻出要素値上位 k 個を取得する．k は利用者が指定した取得数の上限を意味する．本手法では，上位 k 個の頻出要素を求める手法として提案されている Metwally らの Space-Saving [6] のアルゴリズムを利用する．頻出要素値の取得アルゴリズムを図 5 に示す．図 5 中の T が上位 k 個の頻出要素値文字列の集合となる．

出力する要素値データの記述形式を図 6 に示す．要素の属性 item の値は獲得した要素値数を表している．

3.2　XML データ生成手法

データ生成器では，XML データ特徴抽出器から出力された構造特徴，順序情報，要素値の文字列を解析しながら，実データの特性を保持したデータセットを生成する．以下のような段階を経て，実データの特徴を保持したデータセットを生成する．

```
<root>
  <book>
    <title>Contingency Table</title>
    <author>Alexey Tsymbal</author>
  </book>
  <article>
    <title>E-Action Rules</title>
    <author>Colin Frayn</author>
  </article>
</root>
```

図 7　初期データセット例

次節より各過程における処理の詳細について述べる．

1 ：データセット生成の前処理
2 ：要素の割合を満たしたデータセットの生成
3 ：順序性を満たしたデータセットの生成

3.2.1　データセット生成の前処理

利用者が実データよりもサイズの小さいデータセットを必要とした場合，出現割合が極端に小さい要素が，生成データセット内に現れない可能性がある．実データにおいてその要素の重要度を測ることが出来ない以上，生成されるデータセット内には，実データ内の要素を全て含んでいる必要がある．確実に実データ内の全ての要素が現れるデータセットを生成するために，前処理として，提供するデータセットの基礎となるデータセットを生成する．

要求サイズのデータセットを生成する前処理として，実データ内に現れる要素が一回以上現れるようデータを書き出す．例えば，要素の種類が book，article，author，title の 4 種類であれば，図 7 のようなそれぞれが最低一度は現れるデータセットとなる．このデータセットは要素の出現割合を考慮せず，読み込んだ構造概要に従って各要素を出力するため，各要素の出現割合が等しいデータセットである．

3.2.2　要素の割合を満たしたデータセットの生成

前段階で用意された初期データセットを基に，各要素の出現確率を維持したまま，要求されたデータサイズに達するまでデータセットを拡張させていく．

まず，前処理で生成したデータセット内の各要素の出現確率を算出し，算出した要素出現確率と実データの要素出現確率を比較する．この差が最も大きい要素を次に追加すべき要素とみなすことによって，実データにおける各要素の出現確率の維持を図る．

追加時における出現確率から追加すべき要素が決定したら，その要素の値を要素値データからランダムに選び出し，生成データセットに追加する．このとき，追加するデータのバイト数を算出しておく．

要素を加える度に，その時点における要素の出現確率を算出し，理想値（実データの要素出現確率）に近付くよう加えるべき要素を選択し，追加する．この処理をデータサイズに達するまで繰り返し行う．

追加する要素の出現回数とその親要素の出現回数の比率を構造データから読み込み，その比率を維持するように追加要素数を調節することで，一つの要素に複数の同一子要素を持たせることが可能となる．

出力するデータセットのバイト数が要求サイズを超えるまで要素の追加を繰り返す．この時点で，各要素の出現割合を保持した，出現順序がランダムなデータセットが得られる．利用者が順序性を保持させる必要がないと指定した場合，この段階の出現順序がランダムなデータセットを渡す．

3.3 順序性を満たしたデータセットの生成

最後に，前段階で生成された，要素の出現確率，出現割合を維持した要求サイズのデータセットに実データの持つ順序性を持たせる．この処理は，順序性を維持することを選択した場合のみ実行される．実データの持つ順序性を保持しない場合は，要素出現確率を維持した出現順序がランダムなデータセットを提供するため，ここでの生成処理は行わない．

順序性を保持する場合は，ランダムなデータセット生成時に各要素の出力数を保存しておく．順序データを読み込み，順序情報を参照しながら実データの順序が維持されるように出力数分，要素を書き出していく．3.1.2 節で述べたように順序データは，一つの要素が連続して出現する回数を示しているため，その要素の出現回数の総数と連続して出現した回数の比から，その時点で連続して出現する回数を決定し，その数だけ連続してその要素を出力させる．

大まかな分布傾向を保持する場合も同様に，分布傾向データを読み込み，データセット全体を分割し，各要素がそれぞれのクラス内に現れるべき回数を算出し，その回数分連続して出現させることで，実データ本来の完全な順序は考慮せず，分布の傾向を維持させる．

このデータセット生成処理を終えて，実データの持つ要素の順序の特徴を考慮して生成したデータセットが利用者に提供される．

順序性を保持させた場合，保持しない場合に比べて処理が増え，生成時間がより掛かってしまう．順序性の保持については選択制とし，順序性を求めない利用者には，より高速にデータセットを提供する．これらのデータから読み込んだ実データの特徴情報を手掛かりとして実データの特徴を持ったデータセット生成を行う．

4. 実　　験

本章では，特徴情報抽出器およびデータセット生成器の動作確認，処理実行時間について検証する．さらに，本手法の精度評価のため，生成された XML データが実データの特徴を保持出来ていることを確認する．

4.1 動 作 確 認

実装した特徴情報抽出器並びにデータセット生成器の動作確認，処理速度について評価検証を行う．本実験で基盤とした実データは，DBLP [7] の論文書誌データ（データサイズ 498MB）である．なお，CPU：Intel Xeon 2.33GHz，メインメモリ：16GB の PC を用いて実験を行った．

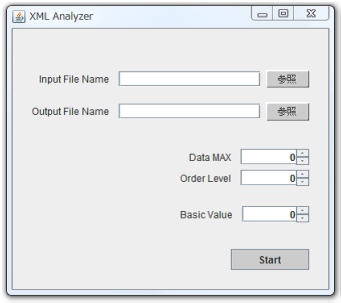

図 8　XML データ特徴抽出器

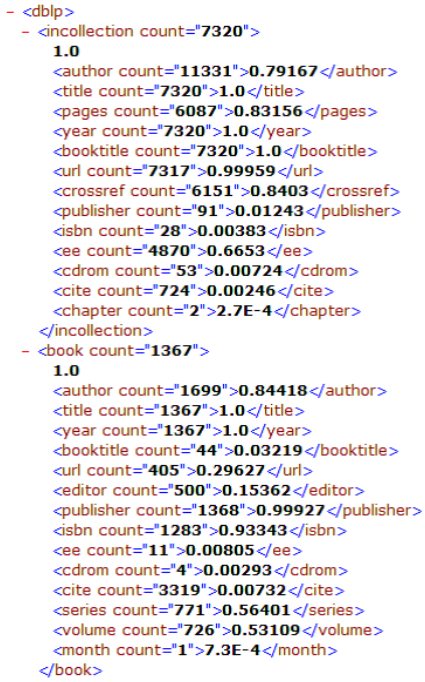

図 9　XML 構造特徴データ

表 1　特徴データのサイズ

特徴ファイル名	データサイズ
構造特徴データ	5KB
順序情報データ	695KB
分布傾向データ	2KB
頻出要素値データ	1,040KB

4.1.1 特徴抽出処理

実装した特徴抽出器のインタフェースを図 8 に示す．図 8 中の Input File Name に基盤とする実データ名を，Output File Name に出力するデータファイル名を入力する．Data Max にて要素値獲得数の上限を指定し，Order Level にて分布傾向を測定するための実データの分割数を指定する．

今回の実験では，DBLP の XML 形式で記述された論文書誌データを基盤となる XML データセット，各要素から取り出す要素値の上限を 1000 件，実データ分割数を 5 と指定し，特徴

```
<incollection>33</incollection>
<book>2</book>
<incollection>34</incollection>
<book>98</book>
<incollection>23</incollection>
<book>44</book>
<incollection>247</incollection>
<book>19</book>
<incollection>11</incollection>
<book>19</book>
<incollection>28</incollection>
<book>15</book>
<incollection>47</incollection>
<book>4</book>
<incollection>122</incollection>
<book>49</book>
<incollection>13</incollection>
<book>1</book>
<incollection>850</incollection>
<book>154</book>
<incollection>30</incollection>
<book>4</book>
```

図 10　順序データ

```
- <DataSet>
    <dblp item="0" />
    <incollection item="0" />
  - <author item="100">
      <data id="0">Alexey Tsymbal</data>
      <data id="1">Colin Frayn</data>
      <data id="2">Carlos Justiniano</data>
      <data id="3">Julian Togelius</data>
      <data id="4">Simon M. Lucas</data>
      <data id="5">Renzo De Nardi</data>
      <data id="6">Georgios N. Yannakakis</data>
      <data id="7">John Hallam</data>
      <data id="8">Heather Barber</data>
      <data id="9">Daniel Kudenko</data>
      <data id="10">Kevin Burns</data>
      <data id="11">Maria Fasli</data>
      <data id="12">Michael Michalakopoulos</data>
      <data id="13">Kyung-Joong Kim</data>
      <data id="14">Sung-Bae Cho</data>
      <data id="15">zs Ga</data>
      <data id="16">Istv</data>
      <data id="17">n Vass</data>
      <data id="18">rgy Kozmann</data>
      <data id="19">Efstratios K. Theofilogiannakos</data>
```

図 11　頻出要素値データ

情報の抽出を行い，実データの各特徴情報が抽出されていることを確認する．

出力された XML 構造特徴データの一部を図 9，要素の順序情報データの一部を図 10，頻出要素値データの一部を図 11 にそれぞれ示す．

図 9 の incollection 要素に着目すると，incollection 要素の出現回数が 7320 回，incollection 要素下の author 要素出現回数が 7320 回，incollection 要素が author 要素を持つ確率が 79.167% であることが分かる．incollection 要素には，author 要素を持たない要素も在れば，author 要素を複数個持つ要素も存在しているということが読み取れる．データセット生成時に，このような要素間の出現回数の比率，出現確率を利用している．

構造特徴として各要素の出現回数，出現確率，要素の並び順情報が取得されていること，要素値が各要素最大 1000 件ずつ取り出されていることが確認できた．また，出力された特徴情報データそれぞれのサイズを表 1 に示す．サイズが約 500MB である実データに対して，出力データはいずれもサイズがかなり小さいことが確認できる．頻出要素値データに関しては，上限値によってデータサイズが変わるが，基盤とする実データよりもサイズが大きくなることはない．分布傾向データについても分割数を変えても順序情報データ以上のサイズにはならない．たとえ実データの規模が今後拡大しても，要素の種類劇的

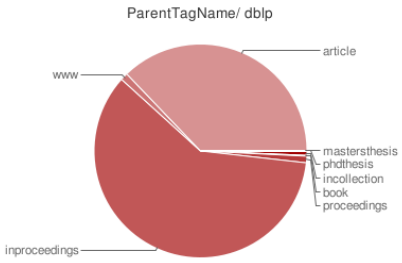

図 12　DBLP 論文書誌データにおける要素の出現割合

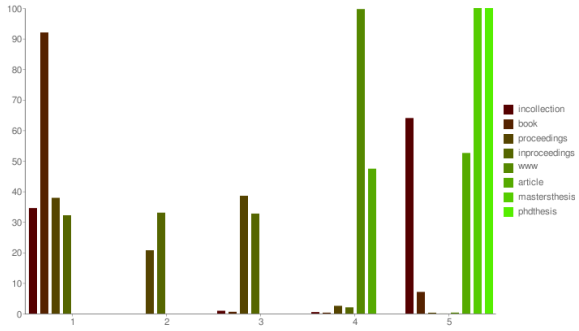

図 13　DBLP 論文書誌データにおける各要素の分布傾向

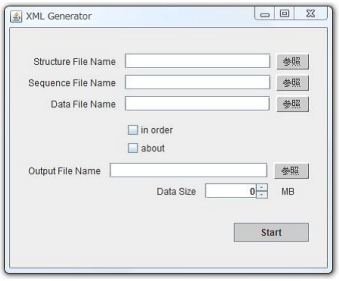

図 14　XML データ生成器

な増加をしない限り，これら構造特徴データ，順序情報データのデータサイズが大きく変化することはなく，小さいサイズが保たれると考えられる．

データ特徴抽出によって解析された実データの特徴を示すグラフを図 12，図 13 にて示す．図 12 は，DBLP 論文書誌データセットのルート直下の要素の出現割合である．DBLP データは大部分が inproceedings 要素，article 要素で構成されており，要素によって出現回数にばらつきがあることが確認出来る．

図 13 は DBLP 論文書誌データの各要素の分布傾向を示す．ここでは，データ全体を先頭から順に五分割し，各要素が各クラスに出現する割合を算出している．このグラフから，要素によって分布に偏りがあることが分かる．例えば，全 incollection 出現数のうち 35 %の要素はクラス 1 に，残り 65 %はクラス 5 に現れることや mastersthesis，phdthesis は全てクラス 5 に現れることなど，各要素のデータ全体における分布傾向を捉えることが出来る．

4.1.2　データ生成処理

実装したデータ生成処理器のインタフェースを図 14 に示す．図 14 中の Structure File Name に構造データファイル名を，Sequence File Name に順序情報データ，あるいは分布傾向デー

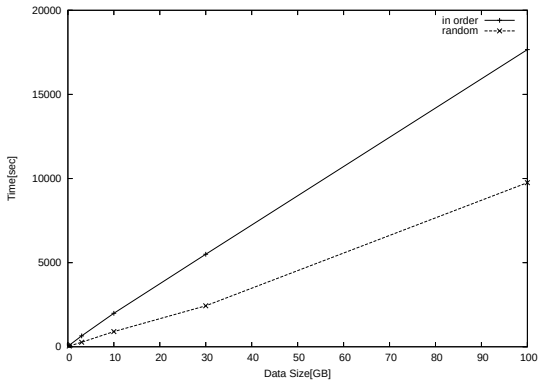

図 15　データ生成時間

表 2　生成データのサイズと生成時間

要求サイズ	生成データサイズ	生成時間 [sec]	
		順序性あり	順序性なし
300MB	300.73MB	88	44
3GB	3.02GB	645	287
30GB	30.54GB	5,497	2,436
100GB	100.73GB	9,751	17,664

タファイル名を，Data File Name に要素値の文字列データファイル名をそれぞれ入力する．in order は，順序性を加味したデータセットを生成する場合にチェックする．about は，分布傾向を加味したデータセットを生成する場合にチェックする．Output File Name に生成するデータセット名を入力し，Data Size にて生成するデータサイズを指定する．

特徴抽出器を用いて得られた DBLP の論文書誌データの三種類の特徴データのファイル名，生成するデータのサイズを入力し，実験を行う．今回，生成したデータセットのサイズは，300MB，3GB，30GB，100GB である．各サイズ毎に実データの持つ順序性を保持したデータセット，順序性を保持しないデータセットを生成し，全部で 4 つのデータセットが得られた．まず生成されたデータセットのサイズ，データ生成時間の評価を行う．

生成したデータセットのサイズと順序性を保持した場合と保持しない場合それぞれの生成時間を表 2 に，生成データサイズによる生成時間の変化のグラフを図 15 に示す．図 15 中の in order は順序性を保持させた場合，random は順序性を保持しない場合の生成時間の変化をそれぞれ示している．図 15 より，順序性を保持した場合と保持しない場合のどちらにおいても，生成するデータセットのサイズに比例した生成時間であることが確認出来る．しかしながら，30GB のデータセットを生成するのに要した時間は，順序性を保持した場合 1 時間 30 分以上，保持しない場合でも 50 分近く掛かっており，とても高速にデータセット生成が出来ているとは言えない．今後の課題として，生成時間の短縮方法について検討していく余地がある．

また，サイズについては，要求されたサイズよりもやや大きいサイズのデータセットが出力されていることが確認出来る．書き込むデータサイズが要求サイズを超えた時点で出力データセットへのデータ追加を停止しているため，最後に追加したデータが必要な子要素を持たないような不完全なデータ構造になる可能性がある．最後に追加するデータのバイト数を調整して要求通りのサイズでデータセットを出力する方法の検討が必要である．生成時間を見ると，順序性を保持した場合とそうでない場合とでは二倍ほど違うが，データサイズに対して線形的に増加しており，スケーラブルな生成器であると言える．生成された XML データの一部を図 16 に示す．


```
- <dblp>
  - <incollection>
      <author>Piraveenan Mahendra</author>
      <author>Astrid Zeman</author>
      <title>Privacy-Preserving Naive Bayesian Classification over Horizontally Partitioned
      <pages>131-159</pages>
      <year>2006</year>
      <booktitle>Encyclopedia of Algorithms</booktitle>
      <url>db/series/sci/sci100.html#PereiraT08</url>
      <crossref>series/sci/2008-115</crossref>
      <publisher>IEEE Computer Society</publisher>
      <isbn>978-3-540-72376-9</isbn>
      <ee>http://dx.doi.org/10.1007/978-3-540-72377-6_1</ee>
      <cdrom>ibmTR/rj2598.pdf</cdrom>
      <cite>conf/vldb/LanzelotteV91</cite>
      <chapter>27</chapter>
    </incollection>
  - <incollection>
      <author>Justin Zhan</author>
      <author>Peter Wang</author>
      <title>Privacy-Preserving Naive Bayesian Classification over Horizontally Partitioned
      <pages>47-69</pages>
      <year>2002</year>
      <booktitle>Encyclopedia of Cryptography and Security</booktitle>
      <url>db/series/sci/sci66.html#JainTL07</url>
      <crossref>series/sci/2007-66</crossref>
    </incollection>
  - <incollection>
      <title>Decentralized Multi-Agent Clustering in Scale-free Sensor Networks.</title>
      <year>2005</year>
      <booktitle>Statistical Implicative Analysis</booktitle>
      <url>db/series/sci/sci118.html#ZhanMC08</url>
      <crossref>series/sci/2008-115</crossref>
      <ee>http://dx.doi.org/10.1007/978-3-540-72377-6_11</ee>
    </incollection>
  - <incollection>
      <author>lia da Costa Pereira</author>
      <author>Andrea Tettamanzi</author>
```


図 16　生成した XML データセット

表 3　要素出現確率 [%]

要素名	DBLP	300MB	30GB
incollection	0.6583	0.6582	0.6583
book	0.1229	0.123	0.1229
proceedings	0.9924	0.9924	0.9924
inproceedings	59.9189	59.9187	59.9189
www	1.12138	1.1214	1.12138
article	37.1773	37.1773	37.1773
phdthesis	0.008093	0.0082	0.0081
mastersthesis	0.00001	0.00001	0.000001

4.2　生成データ精度検証

生成された XML データセットが，基盤の XML データセットの特徴を維持していることを確認する．データ生成実験で作成した 300MB と 30GB のデータセットの特徴抽出を行い，生成データセットの特徴情報と，基盤とした XML データセットの特徴情報を比較し精度検証を行う．

図 12 のグラフで示したルート要素が持つ子要素数の出現割合を数値化し，実データと生成データそれぞれの各要素出現割合を表 3 に示す．実データと生成データ（300MB，30GB）の各要素出現割合を比較すると，差がほとんど無いことが確認できる．DBLP 論文書誌データから抽出した特徴（データ構造，要素の出現割合）を維持した XML データセットが生成されて

表 4　実データの要素分布傾向

DBLP 論文書誌データ [%]					
要素名	1	2	3	4	5
incollection	34.5	0.0	0.92	0.49	64.1
book	92.0	0.0	0.585	0.29	7.10
proceedings	37.9	20.7	38.6	2.53	0.0029
inproceedings	32.2	33.0	32.7	2.04	0.0131
www	0.0	0.0	0.0	99.7	0.313
article	0.0	0.0	0.0	47.4	52.6
phdthesis	0.0	0.0	0.0	0.0	100.0
mastersthesis	0.0	0.0	0.0	0.0	100.0

表 5　生成データの要素分布傾向

生成データセット (3 GB)[%]					
要素名	1	2	3	4	5
incollection	34.6	0.0	0.925	0.49	64.0
book	92.1	0.0	0.60	0.30	6.97
proceedings	38.8	21.0	39.6	0.6	0.0
inproceedings	32.2	33.0	32.7	2.10	0.0
www	0.0	0.0	0.0	99.7	0.312
article	0.0	0.0	0.0	47.4	52.6
phdthesis	0.0	0.0	0.0	0.0	100.0
mastersthesis	0.0	0.0	0.0	0.0	100.0

いると言える．

図 13 で示した各要素の分布傾向を数値化し，DBLP 論文書誌データの分布傾向を表 4，生成データセットの分布傾向を表 5 にて提示する．表内の数値は，データセット全体を五分割したとき，各要素ごとに 1 つのクラス内に出現する要素数の割合を表している．表 4 の実データと表 5 の生成データそれぞれの各要素の分布傾向を比較すると，各要素における著しい偏り（例えば，incollection 要素，book 要素がクラス 2 に全く存在しないことや phdthesis 要素がクラス 5 にのみ存在することなど）は，生成データセットでも再現されており，実データの要素の分布傾向をほぼ維持出来ていると言える．しかし，実データでクラス内の要素出現割合が 0.1% 以下になっていた部分については，生成されたデータセットの場合，そのクラス内に要素が全く入っていない状態になっている．実データの持つ順序性，各要素の分布傾向を生成データセットへより正確に反映させることは，今後検討すべき課題である．

5. おわりに

本研究では，実データの特徴情報としてデータの構造，要素の割合，要素の並び順，文字列，各要素の分布傾向を抽出し，これらの特徴を維持したデータセット生成手法の提案，実装を行い，生成されたデータセットが実データの特性を維持していることを確認できた．

今後は，データセット生成時間短縮方法について検討し，効率的かつ高速な手法を目指して改良していくとともに，データ生成時に各要素の出現の偏りを利用して分布傾向を保持する手法の検討，要素に属性を付与させる手法の検討，実装を行っていく．今回実験で使用した DBLP 論文書誌データでは，全要素の属性 key が含まれているが，本手法では実データから属性を取得していないため，要素の属性に key が含まれない．DBLP 論文書誌データの DTD にて属性 key を持つことを決定付けられているため，DTD を満たすには，要素の属性を取得する必要がある．

また，利用者が入力したデータを任意の位置に挿入できるような機能の追加を検討している．例えば，検索システムの検索時間を測定したい場合，生成データセット内に検索対象となるデータが存在しなくてはいけない．検索対象となるデータを任意の位置に挿入できる機能が追加されれば，この検索対象データがデータセット内のどの位置に存在するか把握した状態で検索時間の測定が可能となる．解決策として，XPath を利用したデータ挿入を検討している．

本研究では，DBLP 論文書誌データを利用したデータ生成実験のみ行っているので，今後 DBLP 以外の XML 形式の実データを使ったデータ生成実験，さらに，他分野の XML ベースの記述言語 (数式を記述する mathML や化学分子式の記述言語 CML など）で記述された実データを使ったデータ生成実験も行っていきたいと考えている．様々な分野において，本手法が汎用的な手法として広く利用できるデータ生成器となることを目指して改良を重ねていきたい．

謝辞　本研究の一部は，科学研究費補助金基盤研究（B）（課題番号 19300026）の助成による．

文　献

[1] A. Schimidt, F. Waas, M. Kersten, M. Carey, I. Manolescu, and R. Busse, “ XMark: a benchmark for XML data management ”, VLDB(2002)
[2] B. Yao, M. Özsu, and N. Khandelwal, “ XBench：benchmark and performance testing of XML DBMSs ”, ICDE(2004)
[3] D. Barbosa, A. Mendelzon, J. Keeleyside, and K. Lyons, “ ToXgene: an extensible template-based data generator for XML ”, WebDB(2002)
[4] Sara Cohen, “ Generating XML Structure Using Examples and Constraints ”, VLDB(2008)
[5] R. Goldman and J. Widom, “ DataGuides: Query Formulation and Optimization in Semistructured Databases ”, ICDE (2002)
[6] A.Metwally, D.Agrawal, and A.E.Abbadi, “ Efficient computation of frequent and top-k elements in data streams ”, ICDT(2005)
[7] The DBLP Computer Science Bibliography, http://www.informatik.uni-trier.de/~ley/db/