データベースシステム 125−53
（２００１．７．１８）

二段階圧縮法のＸＭＬへの適用

大塚真吾 †　　宮崎収兄 ‡

† 千葉工業大学大学院　工学研究科　情報工学専攻　‡ 千葉工業大学　情報科学部　情報工学科

千葉県習志野市津田沼 2-17-1

Tel 047-478-0541

{otsuka, miyazaki}@mz.cs.it-chiba.ac.jp

あらまし　最近、ビジネスデータや行政データにXMLファイルが使われはじめ、そのデータ量は急速に増加している。XMLファイル利用のために、XMLファイルの保存と検索についての研究は重要である。XMLファイルはタグ構造のため、従来のテキスト検索より高度な検索が可能である。また、圧縮に関しても従来のテキスト圧縮より圧縮率が良くなる傾向がある。我々は以前、インデックスを用いてテキストファイルを符号化し、そのファイルを更に他の符号化法で符号化を行うことにより、圧縮率が良く高速な検索が可能となる二段階圧縮法を提案した。本稿では二段階圧縮法のXMLファイルに対しての適用法について検討し、英文で書かれたXMLファイルでの評価を示す。

テキスト検索、テキスト圧縮、XML

The Application of Two-Stage Compression Method for XML

Shingo Otsuka and Nobuyoshi Miyazaki

Department of Computer Science,
Chiba Institute of Technology

2-17-1 Tsudanuma, Narashino, Chiba, Japan
Tel +81-47-478-0541
{otsuka, miyazaki}@mz.cs.it-chiba.ac.jp

Abstract　Recently, XML files are used for business data and public data, and the amount of various XML data has grown very large. It is important to investigate how to store and search XML files. We are able to use more sophisticated conditions in searching XML files than ordinary text files because XML files are TAG structure. Compression ratio of XML files tend to be better than traditional text compression. We have proposed a two-stage compression method that compresses text files using index files and compresses the result again with another algorithm. This paper discusses application of the two-stage compression method for XML files. We also discuss results of experiments of the proposed method for XML files written in English.

text search, text compression, XML

1　はじめに

最近、XML ファイルは急激に普及している。XML ファイルはアプリケーションに依存せずにデータを共有できる。今後、ビジネスや行政のデータが XML で書かれ、その情報量が増大すると予想できる。また、XML ファイルは例えば「打率が一番良いバッターは誰か？」など、従来のテキスト検索にはない高度な検索が可能である。しかし、XML ファイルはタグ情報が大量にあるため通常のテキストファイルに比べ冗長な部分が多い。大量な XML ファイルの管理を考えた場合、これは重大な問題となる。この解決策として XML ファイルを圧縮する事が必要である。しかし、XML ファイルを圧縮してしまうと高度な検索ができない。そこで、属性名や要素名などのタグ情報を損なわずに圧縮を行う方法を考えなければならない。

XML ファイルを圧縮する研究は以前から行われており、XMill[7] や XMLZIP[8] 等のソフトがある。XML ファイルは冗長な部分が多い為、元のファイルの 10%程度に圧縮できる。しかし、これらの研究はファイルの圧縮に重点を置いているため、圧縮されたファイルに対して直接検索する事はできず、検索には復号処理を伴う。一方、要素名等の検索や DOM による操作を考慮に入れた圧縮法として Simplified Element XML[9] がある。これは圧縮率は 50%前後であるが、ファイルの復号を行わずに検索処理などが行える。XML ファイルでも、テキストファイルと同様に圧縮率と検索時間のトレードオフになっている。

我々は以前、索引ファイルを用いた二段階圧縮法を用いて検索効率と良好な圧縮率を両立させることを提案した [2, 3]。また、新聞や雑誌データなど比較的小さいファイルが大量にある場合の二段階圧縮法の適応についても提案を行った。本稿ではこの圧縮法を XML ファイルに適用した場合について検討を行う。まず、2 節で二段階圧縮法について述べる。3 節では二段階圧縮法の XML ファイルへの適用について述べ、4 節で評価を行い、5 節でまとめを行う。

2　二段階圧縮法

2.1　基本原理

二段階圧縮法は元のファイルから索引を作成し、これを圧縮と検索に用いる。索引ファイルは元のファイルから単語を抽出して作成するため，元のファイルの部分集合と考える事ができる。従って，索引ファイル中の情報は全て元のファイル中にあり，この冗長性が圧縮率の低下につながっていると考えられる。そこで，この冗長性を少なくするために，索引ファイル中の単語の位置情報を用いて元のファイルを圧縮する。このファイルと索引ファイルを更に他の符号化法で圧縮を行う事により，圧縮率を高める事ができる。両ファイル間の冗長性が少なくなるので，両ファイル圧縮後のサイズの合計は原理的には元のテキストファイルを単独で圧縮したものと同程度になることが期待できる。また，索引ファイルの符号化法は検索時間と圧縮率のどちらに重点を置くかで，適応符号化法，静的符号化法，または索引の圧縮を行わないなど任意の方式を用いることができる。

複数のテキストファイルに対して検索を行う場合，シーケンシャルサーチは全てのファイルを検索しなければならない。また，予め索引ファイルを作成しておく場合は索引のみ検索を行えば良い。一方，二段階圧縮法はテキスト各々に付加してある索引ファイルを検索するため，全てのファイルをシーケンシャルサーチするより速く検索を行えるが，予め索引ファイルを作成する方法より検索時間が劣る。

2.2　処理方式

二段階圧縮法は図 1 のように索引ファイルを生成し，第一段階圧縮，第二段階圧縮，索引圧縮の計 3 回の圧縮を行う。また，符号化されたファイルは索引部と本体部とに分かれ，索引部は検索に用いることができる。ここでは索引ファイルの生成法とそれぞれの符号化処理について述べる。二段階圧縮法の処理ステップは以下のようになる。

1. 索引ファイルの作成
2. 第一段階圧縮
3. 索引圧縮・第二段階圧縮

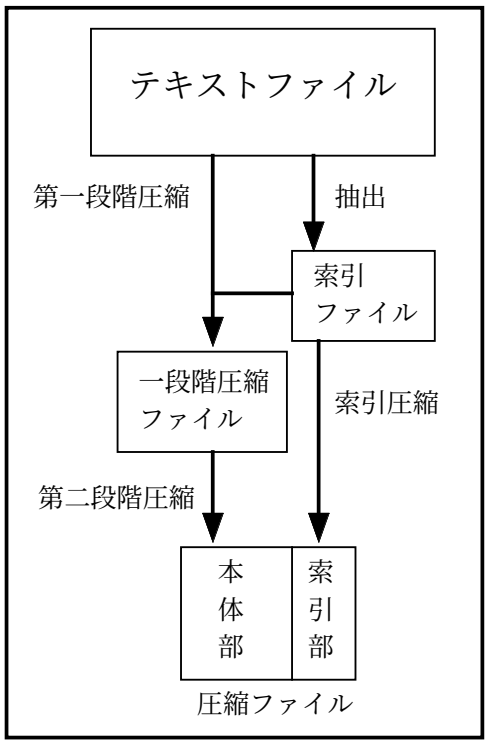

図 1: 圧縮の概要

2.3 索引ファイルの作成

テキストファイルから単語を抽出し索引ファイルを作成する。英単語は「アルファベットの連続した文字列」と定義する。このようにして単語を抽出した索引ファイルの例を図 2 に示す。索引ファイル中の単語の順番は次の処理である第一段階圧縮を行う前であれば，どのような順番でもかまわない。単語が重複する場合，索引ファイルには一度だけ登録する。また，‘And’ と ‘and’ のように大文字と小文字は区別して登録する。

2.4 第一段階圧縮

索引ファイル中の単語の位置情報を用いてその単語の符号化を行う。二段階圧縮法の符号語長は 2 バイトまたは 3 バイトの可変長とし，符号語長以下の単語は符号化を行わない。単語の位置情報とは索引ファイル中の単語に通し番号を付けたものである。図 2 の例では以下のようになる。

I 0

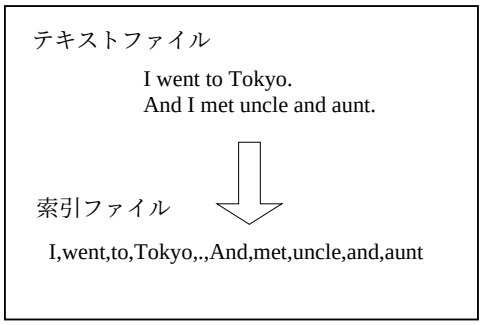

図 2: 索引ファイルの例

went 1

to 2

・

・

・

aunt 9

この番号を用いてテキストファイルの符号化を行う。また，通し番号は 2 進数で表現する。ASCII コードでは 1 バイトコードの一番左のビットは常に ‘0’ である。そこで，符号語は一番左のビットを ‘1’ にする事でその文字が符号語かそうでないか識別する事ができる。また，英文は語と語の間にスペースがある事が非常に多いため，その次のビットは ‘符号化される単語の次がスペースかどうか’ を判別するのに用いる。‘10...’ ならば符号語の次の文字がスペースで無いことを表し，‘11.....’ ならば符号語の次の文字がスペースであることを表している。これにより，テキストファイルから大部分のスペースを取り除くことができる。

符号語の識別とスペースの有無を考慮すると，符号語長が 2 バイトなら約 16,000 単語を表現することができる。また，索引ファイル中の単語数がこの数を超えた場合，索引ファイル中の単語の通し番号がこれよりも大きい単語については符号化を行わない。一方，符号語長が 3 バイトならば約 800 万単語を表現でき事実上全ての単語を表現できる。2 バイトの符号語長を用いた符号化の例を図 3 に示す。索引ファイル中の ‘went’ は通し番号に ‘1’，‘Tokyo’ は ‘3’ に対応する。‘went’ の次の文字がスペースなので符号語は ‘1100000000000001’ となり 16 進数で表現すると ‘c001’ となる。また，

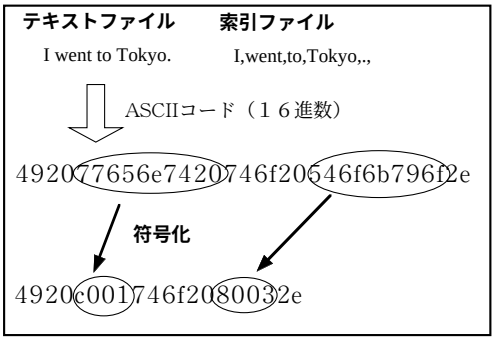

図 3: 第一段階圧縮

‘Tokyo’ の次の文字はスペースではないので符号語は ‘1000000000000011’ となり 16 進数表示は ‘8003’ となる。この例では，文章自体は約 60%に圧縮されるが，索引ファイルの容量を考えると実際にはその容量は減少していない。しかし，文章中には同じ単語が何度も頻繁に使われるので，文章が大きくなれば圧縮率は良くなる。

2.5 第二段階圧縮・索引圧縮

第二段階圧縮は第一段階圧縮によって生成された一段階圧縮ファイルを他の符号化法で符号化を行う。これにより，更に圧縮率を高める事ができる。索引圧縮は検索時間を重視し，復号時間が短い符号化法または，直接検索が行える符号化法で符号化を行う。また，検索時間に重点を置く場合は索引部の圧縮を行わない事もできる。

このように，第二段階圧縮，索引圧縮とも，既存の符号化法を取り入れているので，新たに高性能な符号化法が提案されても，二段階圧縮法は柔軟に対応することができる。

3 二段階圧縮法の XML への適用

二段階圧縮法で圧縮したファイルに対する検索は索引を用いて行う。したがって、検索語に該当する単語のみ検索が行えるが、検索語がファイル中のどこで使われているかを検索する事はできない。XML ファイルに対する検索は「要素名 ”NAME ” の値は？」のような文書構造に対する検索が重要である。そこで、文書構造に対する検索が行える検索方式について検討を行う。

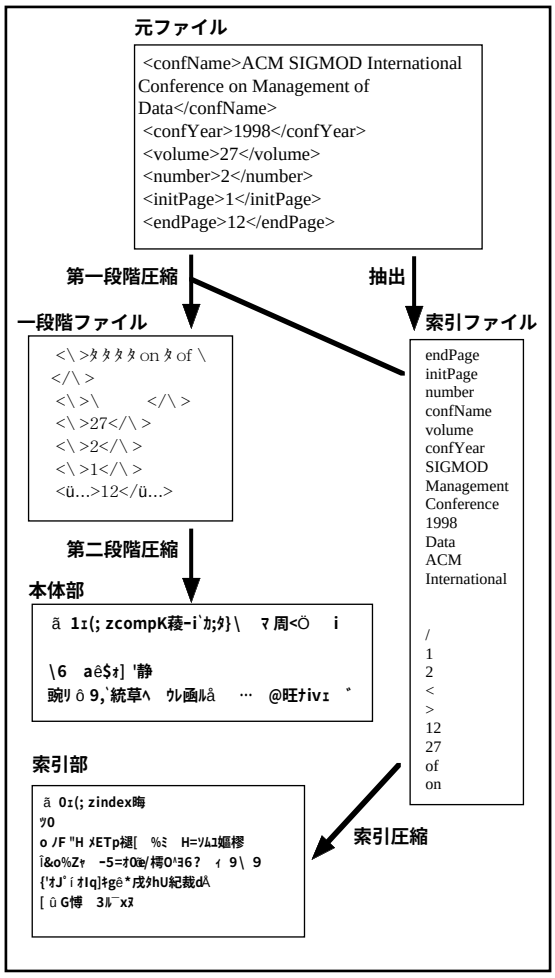

図 4: XML ファイルを圧縮した時のそれぞれのファイルの内容

3.1 XML ファイルの二段階圧縮

XML ファイルに対して通常のテキストファイルとして扱い二段階圧縮法を適用した場合のファイルの中身について図 4 に示す。ここでは第二段階・索引圧縮には gzip を用いている。本体部と索引部に対しては直接検索は行えない。一方、一段階ファイルはタグ情報が残っている。タグを表すカッコ (⟨⟩) は 1 バイトであり、符号語長より短い文字なので符号化されずに残っている。タグの中

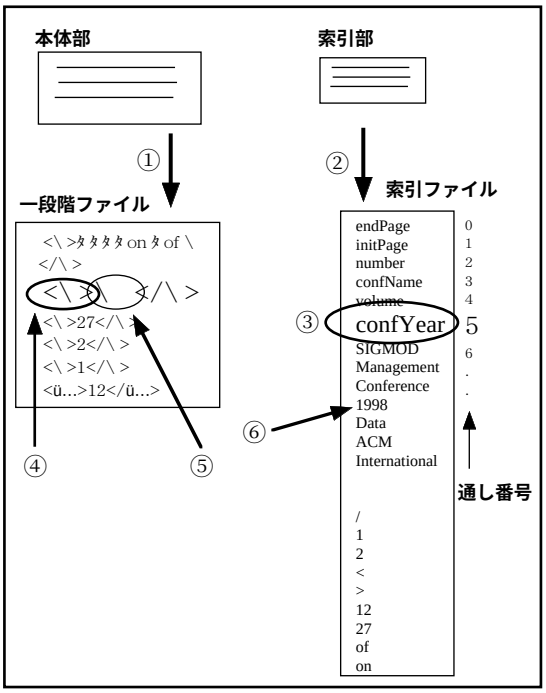

図 5: 二段階圧縮した XML ファイルの検索例

は索引ファイル中の単語の通し番号が書いてあるので検索する事が可能である。

二段階圧縮法で圧縮した XML ファイルに対してタグの内容の検索を行う場合、以下の手順で行う。

1. 本体部を復号し一段階ファイルを作成する。
2. 索引部を復号し索引ファイルを作成する。
3. 索引ファイルを検索し該当するタグの通し番号 1 を得る。
4. 一段階ファイルから「⟨ 通し番号 1⟩」を検索する。
5. 「⟨ 通し番号 1⟩」の次の通し番号 2（複数の場合もある）を得る。
6. 通し番号 2 に該当する単語を索引ファイルから探す。

このように、二段階圧縮法で圧縮された XML ファイルに対して高度な検索はファイルの一部分の復号で行える。属性名”confYear”の値を検索する場合の例を図 5 に示す。上に述べた手順で検索が行え、結果”1998”を得る事ができる。しかし、索引ファイルを 2 回参照するので効率の良い検索とは言えない。

3.2 DTD ファイルを利用した二段階圧縮

ほとんどの XML ファイルでは属性名等の情報は DTD ファイルに書かれている。XML ファイルの冗長性は属性名等のタグ情報が原因であることは前に述べた。そこで、DTD ファイル中のタグ情報を用いた二段階圧縮を提案する。この方式は

1. DTD ファイルから索引ファイルを作成する。
2. 索引ファイルを用いて XML ファイルの第一段階圧縮を行う。
3. 索引ファイルと一段階ファイルを他の符号化法で圧縮する。

の手順で行う。これにより、タグ情報だけを圧縮する事になる。DTD ファイルを用いた圧縮ファイルの内容について図 6 に示す。一段階ファイルはタグ情報だけ圧縮されている事が分かる。この方式で圧縮されたファイルに対する検索は以下の手順で行う。

1. 本体部を復号し一段階ファイルを作成する。
2. 索引ファイルを検索し該当するタグの通し番号を得る。
3. 一段階ファイルから通し番号を検索する。
4. 通し番号の次の情報を得る。

今度はタグの内容は符号化されていないため、簡単に解を得ることができる。また、従来の二段階圧縮法は符号語長が 2 または 3 バイトだが、DTD ファイルのサイズが小さいため単語数はそれほど多くならない。そこで、符号語長を 1 または 2 バイトにする。この符号語長で約 16,000 単語を表現する事ができる。

二段階圧縮法は索引を用いて圧縮を行うので、同じ単語が頻繁に使われれば圧縮率が良くなる。また、ファイルサイズが大きくなるほど索引部が占める割合も減り圧縮率が良くなる。従って、ファイルサイズが小さいと圧縮率は良くない。図 7 のように同じ DTD ファイルを用いた XML ファイルが大量に集まっている環境では 1 つのファイルごとに二段階圧縮を行うと、どの索引ファイルにもタグ情報が入り圧縮率の向上が期待できない。また、XML ファイルを追加するごとに索引ファイルが増えるので検索時間も増加する。一方、DTD

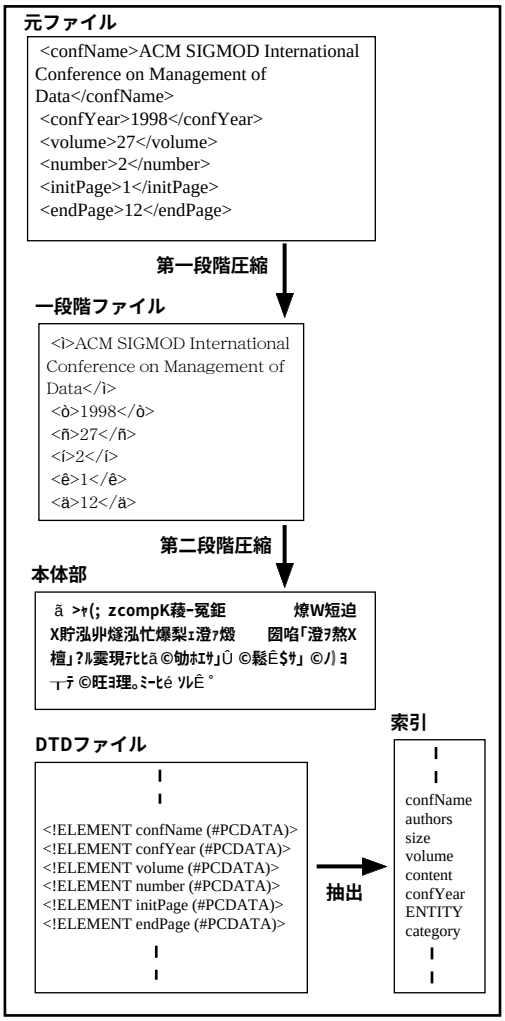

図 6: DTD ファイルを利用して圧縮した時のそれぞれのファイルの内容

ファイルを利用して二段階圧縮を行うとタグ情報が冗長にならず圧縮率の向上が期待できる。また、同じ DTD ファイルを使った XML ファイルを追加しても、索引ファイルは増加しないので検索時間は DTD を用いない方法より速く行える。

4 評価

実験は Linux 上で ACM SigmodRecord の XML ファイル [1] を用いて行った。ファイルの内容について表 1 に示す。表中の平均サイズとはファイルサイズをファイルの個数で割ったものである。

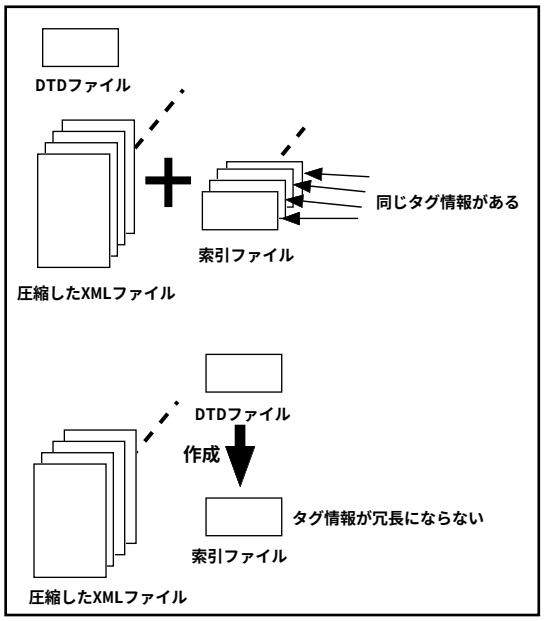

図 7: DTD ファイルを利用した場合の利点

表 1: 実験に用いたファイル

ファイルの内容	ファイルの個数	ファイルサイズ	dtdサイズ	平均サイズ
IndexTermsPage	920個	2,112,609	1,229	2,296
OrdinaryPage	51個	268,717	1,057	5,269
ProceedingPage	16個	589,215	1,357	36,826
SigmodRecord	1個	478,343	619	478,343

単位：byte

IndexTearmPage は小さいファイルが大量にある環境である。反対に SigmodRecordPage は約 500K バイトのファイル 1 つだけである。比較対象として XMill と gzip でも実験を行った。評価は圧縮率と検索時間について行った。圧縮率とは圧縮ファイルの容量と元ファイルの容量の比である。検索時間は構造に対する検索を行う事を前提とし、二段階圧縮法に関しては前節で提案した手順で検索時間を測定している。また、XMill と gzip は復号時間と検索時間を合計である。検索には grep コマンドを用い、時間の測定には time コマンドを用いた。二段階圧縮法の第二段階圧縮には gzip を用いた。実験結果を表 2 に示す。

IndexTermsPage では DTD を利用した二段階圧縮法が圧縮率、検索時間共に一番良い結果が得られた。ファイルサイズが大きい SigmodRecordPage は XMill の圧縮率が良く、検索時間はあまり差が

表 2: 実験結果

ファイルの内容	圧縮方式	圧縮率	検索時間
IndexTermsPage	二段階圧縮	55.56%	12.19
	二段階圧縮(DTD利用)	42.27%	4.54
	XMill圧縮	48.66%	7.70
	gzip圧縮	46.91%	4.62
OrdinaryPage	二段階圧縮	33.28%	0.63
	二段階圧縮(DTD利用)	23.83%	0.37
	XMill圧縮	24.67%	0.51
	gzip圧縮	25.27%	0.33
ProceedingPage	二段階圧縮	19.86%	0.20
	二段階圧縮(DTD利用)	15.29%	0.11
	XMill圧縮	14.34%	0.15
	gzip圧縮	15.71%	0.10
SigmodRecord	二段階圧縮	16.34%	0.05
	二段階圧縮(DTD利用)	15.42%	0.04
	XMill圧縮	12.06%	0.06
	gzip圧縮	17.01%	0.02

なかった。従来の二段階圧縮法の圧縮率は他の圧縮法と比べてあまり良くない。また、gzip は圧縮率に関しては DTD を利用した二段階圧縮法より少し悪い結果になった。

実験の結果から、小さい XML ファイルが大量にある環境では DTD ファイルを利用して二段階圧縮法を行えば、圧縮率と検索時間の向上が見込める事が分かった。また、1 つのファイルサイズが大きくなると XMill の圧縮率が優れている事が分かる。

5 おわりに

本稿では二段階圧縮法の XML ファイルへの適用について検討を行った。XML ファイルに対する検索は属性が持つ要素についてなど、高度な検索が可能である。そこで本稿では従来の二段階圧縮法の検索方法を変更する事により、ファイルの一部を復号して複雑な検索を可能にする方法について検討を行った。また、DTD ファイルを利用する事によって、従来の二段階圧縮法よりも圧縮率、検索時間どちらも向上させる事ができた。実験結果から同じ DTD ファイルを利用する XML ファイルが大量にある環境では XMill よりも圧縮率が良くなる事が分かった。

今後の課題は、大規模な XML ファイルを用いての評価と XML ファイルの保存、検索システムの構築を行う事である。

参考文献

[1] ACM SIGMOD Record XML Version. *http://www.acm.org/sigs/sigmod/record/xml/*.

[2] 大塚真吾, 宮崎収兄. ファイル検索のための二段階圧縮法. 電気学会論文誌 C　No7 掲載予定, 2000.

[3] 大塚真吾, 宮崎収兄. 高速検索を可能とする日本語テキストの二段階圧縮法. 情報処理学会論文誌：データベース (TOD-10)　掲載予定, 2000.

[4] 定兼邦彦, 今井浩. 転置ファイルおよび接尾辞配列の効率的圧縮法. 電子情報通信学会データ工学研究会, Vol. 40, pp. 57–62, 1999.

[5] 須藤真理. 情報検索とデータ圧縮とを統合したシステム mg の日本語化. 情報処理学会情報学基礎研究会, Vol. 40, pp. 33–40, 1995.

[6] 植松友彦. 文書データ圧縮アルゴリズム入門. ＣＱ出版, 1994.

[7] XMill. *http://www.research.att.com/sw/tools/xmill/*.

[8] XMLZIP. *http://www.xmlzip.com*.

[9] 横山昌平, 太田学, 石川博. 要素名圧縮による xml データ圧縮手法の提案. *DBWeb2000*, pp. 331–337, 2000.

[10] J. Zobel and A. Moffat. Adding compression to a full-text retrieval system. *Software-Practice and Experience*, Vol. 25-8, pp. 891–903, 1995.