An Empirical Evaluation of Simple DTD-Conscious Compression Techniques

James Cheney

Database Group/Digital Curation Centre
University of Edinburgh

WebDB 2005

June 17, 2005

Always start with a joke...

Why did the chicken cross the road?

To get to the other side!

Always start with a joke...

```
<?xml version="1.0"?>
<!DOCTYPE joke SYSTEM "joke.dtd">
<joke type="question-answer">
  <setup>
    Why did the chicken cross the road?
  </setup>
  <punch-line>
    To get to the other side!
  </punch-line>
  <laughter type="optional"/>
</joke>
```

XML is verbose.

XML Compression

The term XML compression has been used in several different contexts:

1. minimum-length encoding for efficient XML storage and transmission

2. compact binary formats for efficient XML stream processing

3. techniques for efficient in-database XML storage and query processing

For us, XML compression means (1).

Prior work: XML compression

- State of practice: use gzip or bzip2 (or library variants) to compress XML as text

- [Liefke, Suciu 2000] XMill: transform XML document to bring similar text closer together, then use gzip/bzip2

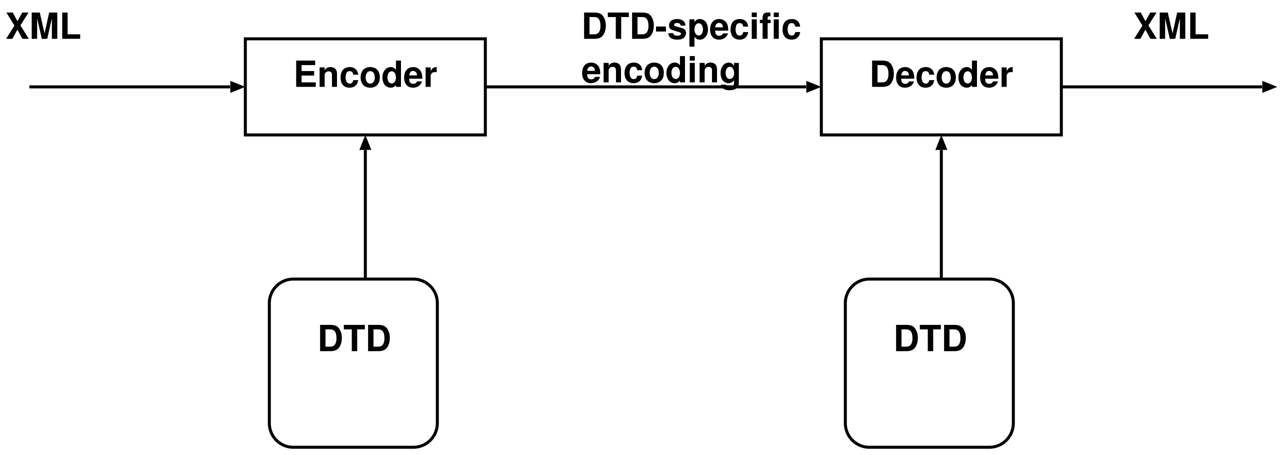
- [Cheney 2001] XMLPPM: compress XML by leveraging advanced statistical text compression techniques

  – XMLPPM/variants have best published results so far.

DTD-conscious compression

DTD/schema information tells us what valid XML documents to expect, so “obviously” should help compression

Assume encoder and decoder have access to (identical) DTD



Prior work: DTD-conscious compression

[Levene and Wood, 2002]: use DTD regexp content models to encode element structure

Example: In regexp model $(c + d)(ab)^{*}d^{?}$, encode

$$cabababd$$

as

$$011101$$

Bits indicate decisions made at choice points during validation.

Prior work: DTD-conscious compression

While likely much more compact than XML text, LW02 technique does not compress better than XMLPPM

Why? XMLPPM already “learns” a lot about data structure, and uses a more advanced statistical model than Levene and Wood’s encoding.

Moreover, LW02’s technique is not easy to incorporate into XMLPPM

Why? LW02’s encoding breaks *byte alignment*, confusing later text compression stages

Lesson: Need to avoid stepping on toes of later stages

Why DTDs vs XML Schemas?

- Pro: DTDs simpler, more stable, less work to validate; techniques should generalize

- Con: XML Schemas more descriptive (especially datatypes), appear to be more popular now

It is a lot of work to implement DTD-conscious, let alone XML Schema-conscious compression; is it worth the effort?

Our approach

Look for simple techniques for leveraging DTD information in XMLPPM.

Easier to implement, easier to test, easier to incorporate into XMLPPM.

If simple techniques are effective, more complex techniques may be worthwhile.

Implemented in DTDPPM, an XMLPPM variant that simultaneously validates and compresses

Four simple optimizations

- Strip *ignorable* (non-PCDATA) whitespace — obvious but necessary for good compression due to properties of underlying compressor

- Re-use element, attribute, default symbols found in DTDs

- Predict element symbols (open and close-element tags) using regular expression context

- Sort and encode attribute lists using bitmaps; use types and default information also

Example

Given element declaration

```
<!ELEMENT book   (title,author+)>
<!ELEMENT title  (#PCDATA)>
<!ELEMENT author (#PCDATA)>
```

Encode

```
            <book>
              <title>Title</title>
              <author>Auth1</author>
              <author>Auth2</author></book>
```

as

```
   00 ’f’ ’o’ ’o’ ’A’ ’u’ ’t’ ’h’ ’1’ 01 ...  FF FF
```

Example: attribute list coding

Given attribute list declaration

```
<!ATTLIST elt att1 CDATA   #FIXED "foo"
              att2 (x|y|z) #REQUIRED
              att3 CDATA   #IMPLIED
              att4 CDATA   "bar">
```

we can encode the attribute list of

```
<elt att1=’foo’ att2=’y’ att4=’baz’>
```

as

01000000_2 `01 ’b’ ’a’ ’z’ 00`

Evaluation

- “XMLPPM benchmark”: corpus used in [Cheney 2001]; mostly historical interest (5MB, mixed sources)
- NewsML: Reuters news reports (2.7MB total, 11KB avg)
- MusicXML: Musical scores (1.8MB total, 101KB avg)
- Medium data sets (Washington corpus, 3MB total, mixed sources)
- Large data sets (DBLP, XMark, PSD, Medline, 100-700MB each)

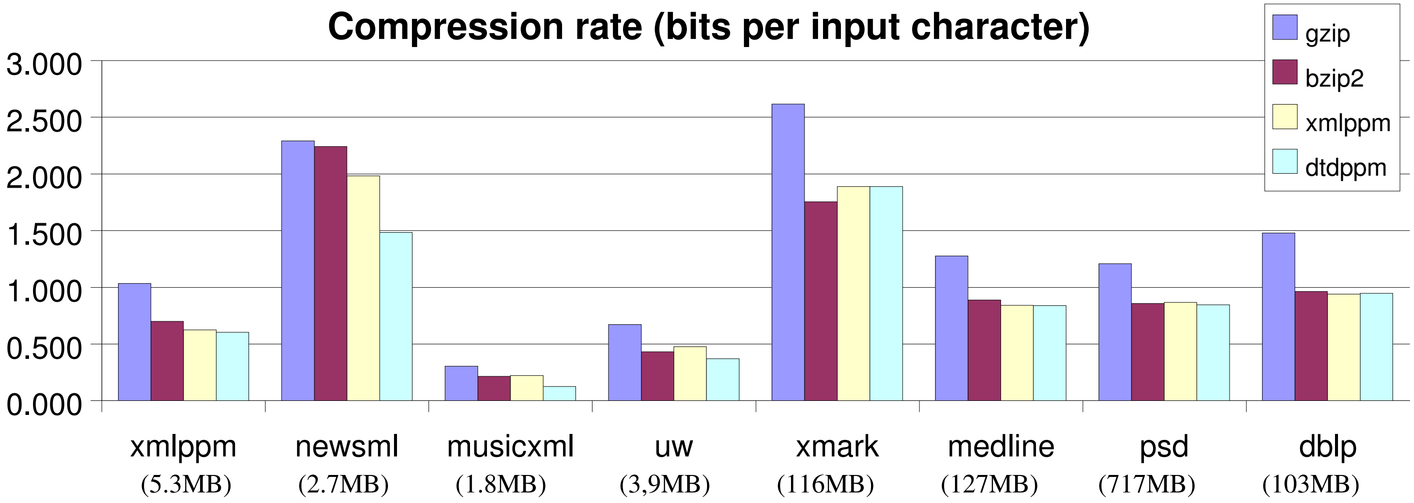
Setup

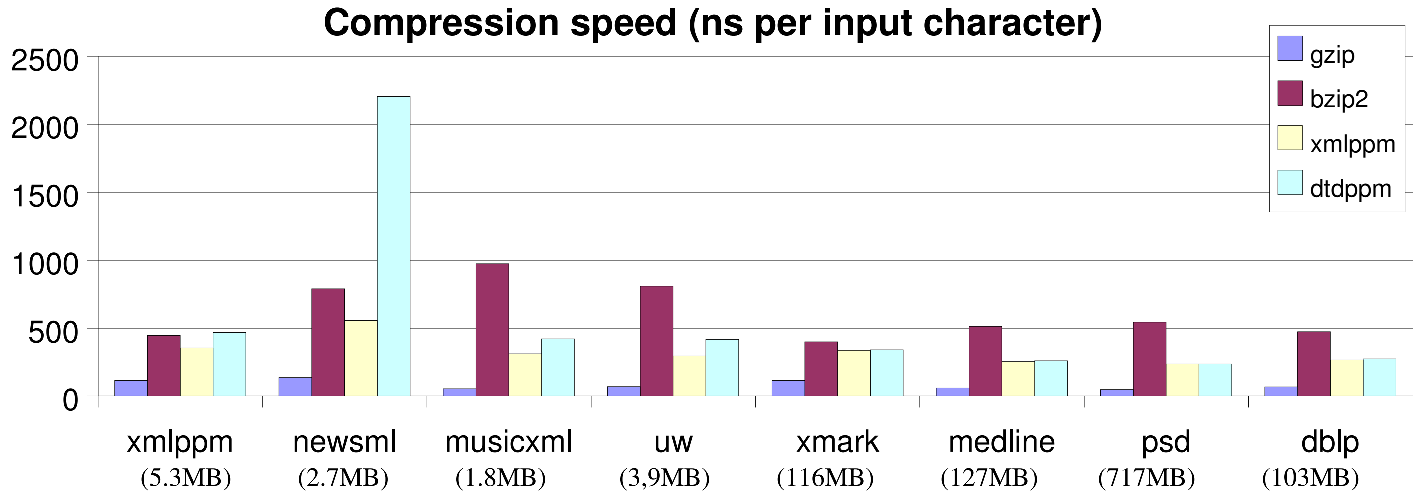
Experimental setup: AMD64 3000+, 512MB RAM, FC3

Measured

- compression effectiveness (compressed bits per input character)
- compression time (ns per input character)

Note: Decompression for PPM techniques $\approx$ compression time (but gzip, bzip2 decompress faster than they compress)


Compression rate (bits per input character)
3.000
2.500
2.000
1.500
1.000
0.500
0.000
gzip
bzip2
xmlppm
dtdppm
xmlppm (5.3MB)
newsml (2.7MB)
musicxml (1.8MB)
uw (3,9MB)
xmark (116MB)
medline (127MB)
psd (717MB)
dblp (103MB)


Compression speed (ns per input character)
2500
2000
1500
1000
500
0
gzip
bzip2
xmlppm
dtdppm
xmlppm (5.3MB)
newsml (2.7MB)
musicxml (1.8MB)
uw (3,9MB)
xmark (116MB)
medline (127MB)
psd (717MB)
dblp (103MB)

Observations

Short documents (NewsML) compress better, but re-parsing DTD is very expensive.

Highly-structured documents (MusicXML) compress much better

Flat data sets or very large irregular documents compress no better than bzip2, but xmlppm/dtdppm are faster than bzip2

XMark compresses no better, but may not be a realistic compression benchmark (since randomly generated)

Which technique is best?

No single technique dominates.

In particular, improvement is not all from WS stripping; each technique can account for 0-80% of improvement.

Need a variety of techniques because XML data structure varies widely.

WS stripping is probably the best value for effort: everyone should (and many already) do it when compressing XML.

Conclusions

DTD information: “obviously” should be useful for compression

However, real improvements over advanced XML-only techniques do not come easily

We have explored many alternatives and identified four that do work (in the context of one XML compressor, XMLPPM).

Future work: Improving efficiency, more advanced techniques, XML Schema

`http://sourceforge.net/projects/xmlppm`