Tradeoffs in XML Database Compression

James Cheney

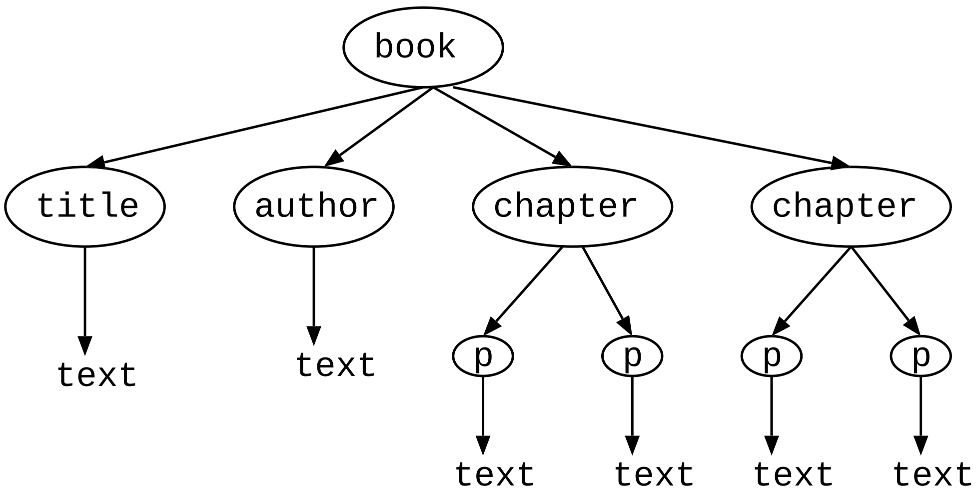
University of Edinburgh

Data Compression Conference

March 30, 2006

XML Compression

- XML: a format for tree-structured data
- Increasingly used for large data collections (e.g. bibliographic/scientific databases)




```
<book>
 <title>text</title>
 <author>text</author>
 <chapter><p>text</p></chapter>
 <chapter><p>text</p></chapter>
</book>
```

- Verbose, so gzip or bzip2 usually used to compress XML
- Can XML-specific compression techniques do better?

Prior work

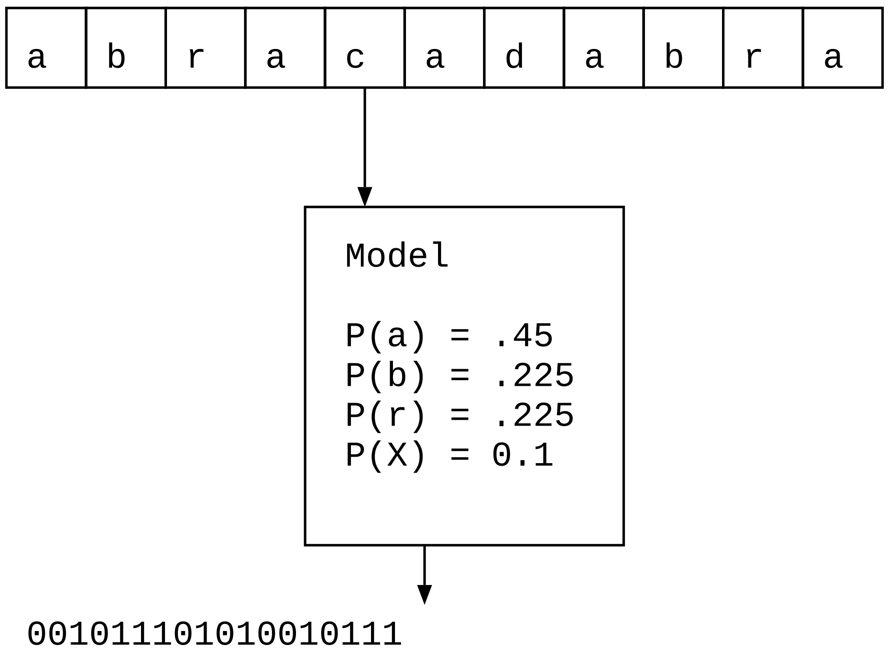
- XMill (Liefke, Suciu 2000): first (serious) XML compression work
  - transform XML document, then compress with gzip/bzip2
- XMLPPM (Cheney, DCC 2001): uses statistical modeling, better compression than XMill but slower
- SCMPPM (Adiego, de la Fuente, Navarro, DCC 2004), XAUST (Hariharan, Shankar, CIAA 2005): use different statistical models, report improvement over XMLPPM
- Other approaches have been explored but statistical methods have best performance

Motivation

- Most experimental evaluations have focused only on *compression rate* (and often only for large files)
- Other relevant factors such as memory requirements, rate of convergence neglected
- Thus, experiments demonstrating improved compression are valid, but don't tell the whole story.
- Goal of this talk: detailed comparison of *memory vs compression rate* and *rate of convergence*
- Focus on *unstructured text compression* behavior of *statistical XML compression models* used in XMLPPM, SCMPPM, and XAUST

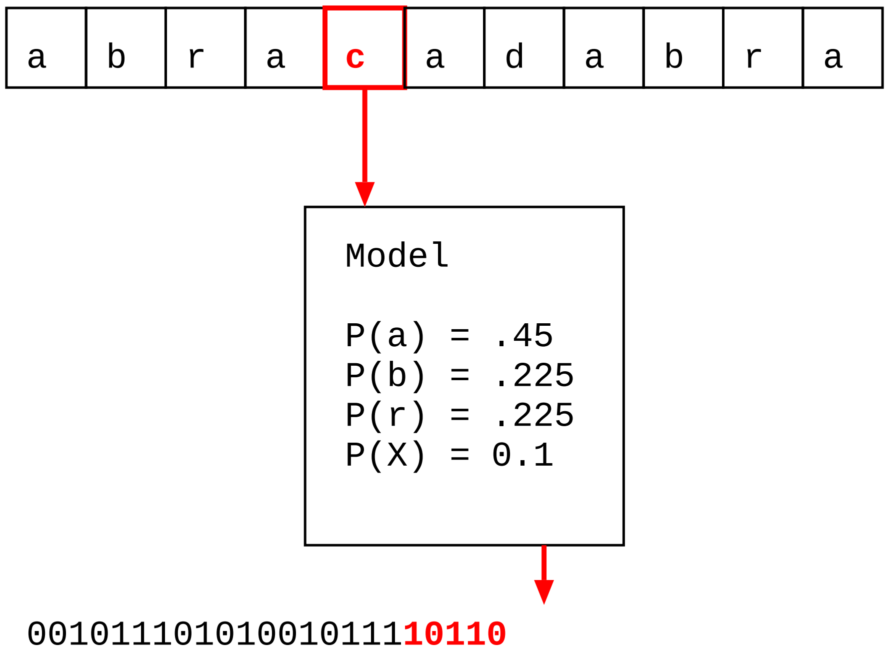
Statistical models

- Statistical text compression: compresses text by building a *model* that predicts next symbol
- *Adaptive* approach: interleave model building and prediction/compression. Requires only one pass over data, but has to “learn” model as it goes



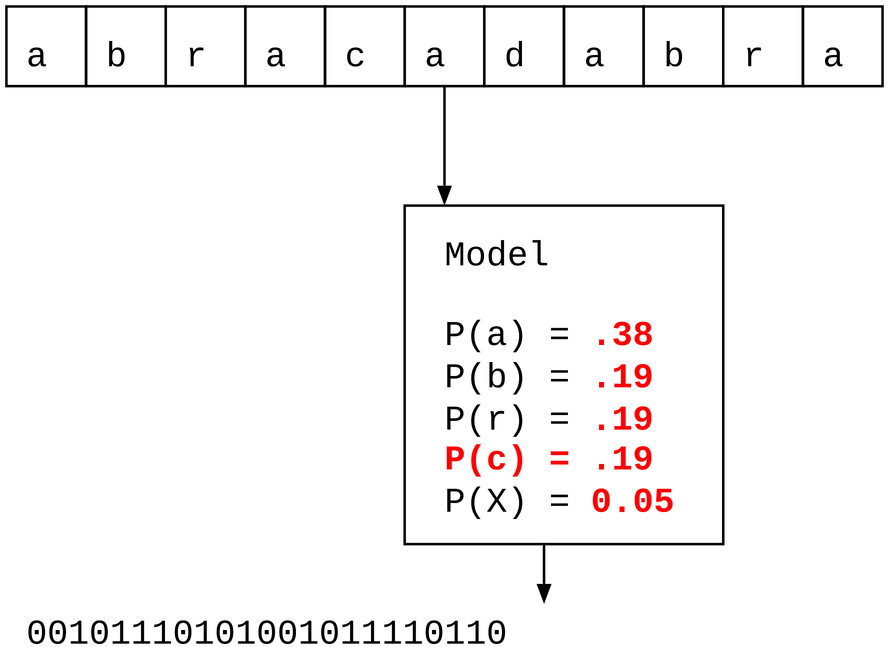
Statistical models

- Statistical text compression: compresses text by building a *model* that predicts next symbol
- *Adaptive* approach: interleave model building and prediction/compression. Requires only one pass over data, but has to “learn” model as it goes



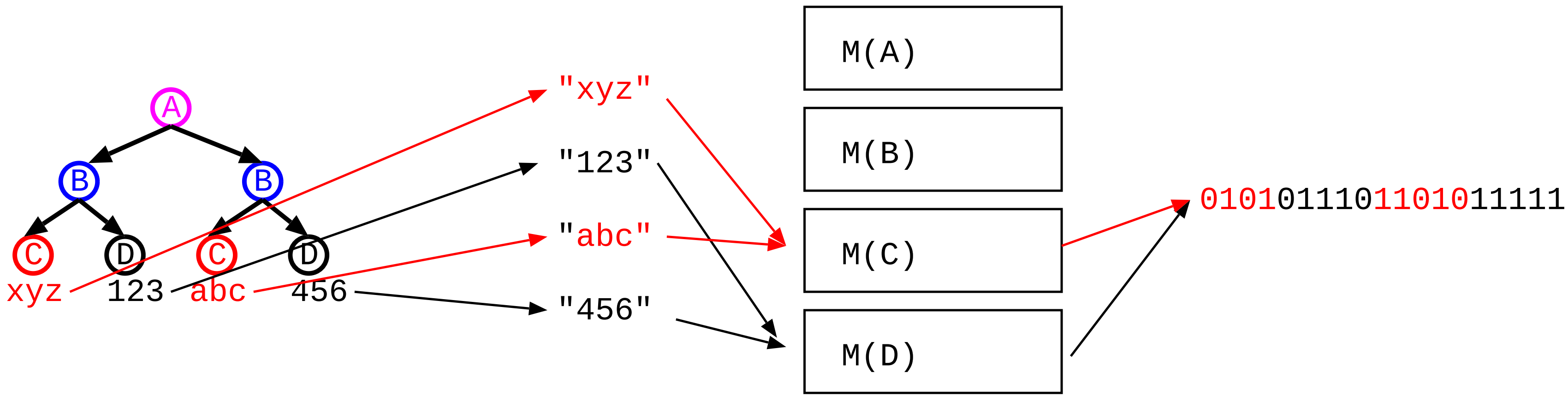
Statistical models

- Statistical text compression: compresses text by building a *model* that predicts next symbol
- *Adaptive* approach: interleave model building and prediction/compression. Requires only one pass over data, but has to “learn” model as it goes



Approach #1: Multi-model

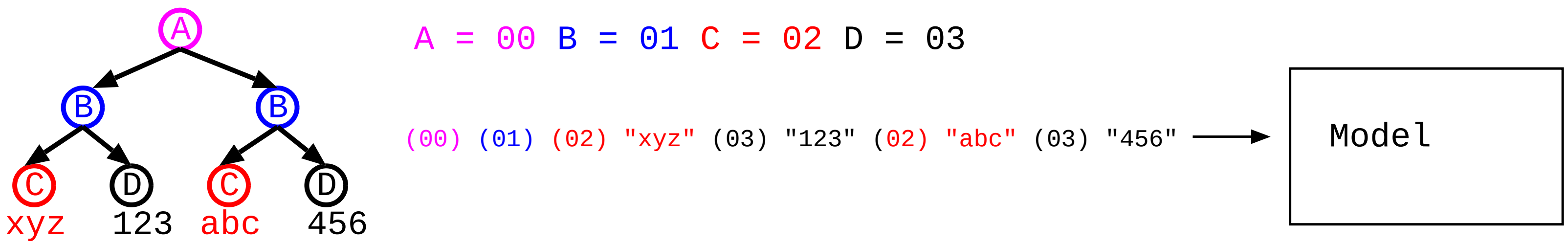
- Idea: Switch between n models, one model $M(e)$ per element name e
- Use $M(e)$ to encode the text immediately under e



- Used in SCMPPM, XAUST
- I'll call this the *Structured Contexts Model* (SCM) approach

Approach #2: Single-model

- Idea: Use a *single* model for text, but "prime" model with element symbols
- Priming symbols are "free" since can be inferred from tree context (this is part of the fixed cost we're ignoring)



- where (00), (01) etc are priming symbols for various element tags
- Used in XMLPPM, so I'll call it the XMLPPM approach

Prior experiments

- XMLPPM: wide variety of XML documents, max size <1MB, used 1MB memory for statistical models
  - When limit reached, statistical model restarts
- SCMPPM: used large TREC documents with 8 elements, very little structure; statistical models used 1MB each (maximum of 8MB for TREC)
- XAUST: used large documents such as DBLP; no memory upper limit

Flaws in prior experiments

- XMLPPM: didn't consider large documents, memory variation
- SCMPPM, XAUST: didn't consider small documents, memory variation
  - Can't tell whether reported compression gain is due to using more memory or more accurate modeling
  - SCM approach may allocate much more memory than it ever uses
  - SCM approach may eventually attain much better compression, but may converge very slowly (benefiting only large files)
- Not enough data to draw any conclusions about relative merits of these approaches

Text is the dominant factor

- Most of the "interesting" content of most XML documents is unstructured text

	gzip			xmlppm		
fi le	struct	total	%struct	struct	total	%struct
DBLP	9.9MB	52.4MB	19%	667KB	33.4MB	2.0%
Medline	2.7MB	20.2MB	14%	539KB	13.7MB	3.9%
XMark	4.1MB	38.1MB	11%	287KB	27.6MB	1.0%
PSD	13.6MB	108MB	12%	2.5MB	79.6MB	3.1%

- Existing techniques already compress structure well (less than 1–20% of document)
- So, in this work, focus only on modeling/compression of unstructured text in XML
- Compressing the structure is treated as a small fixed cost

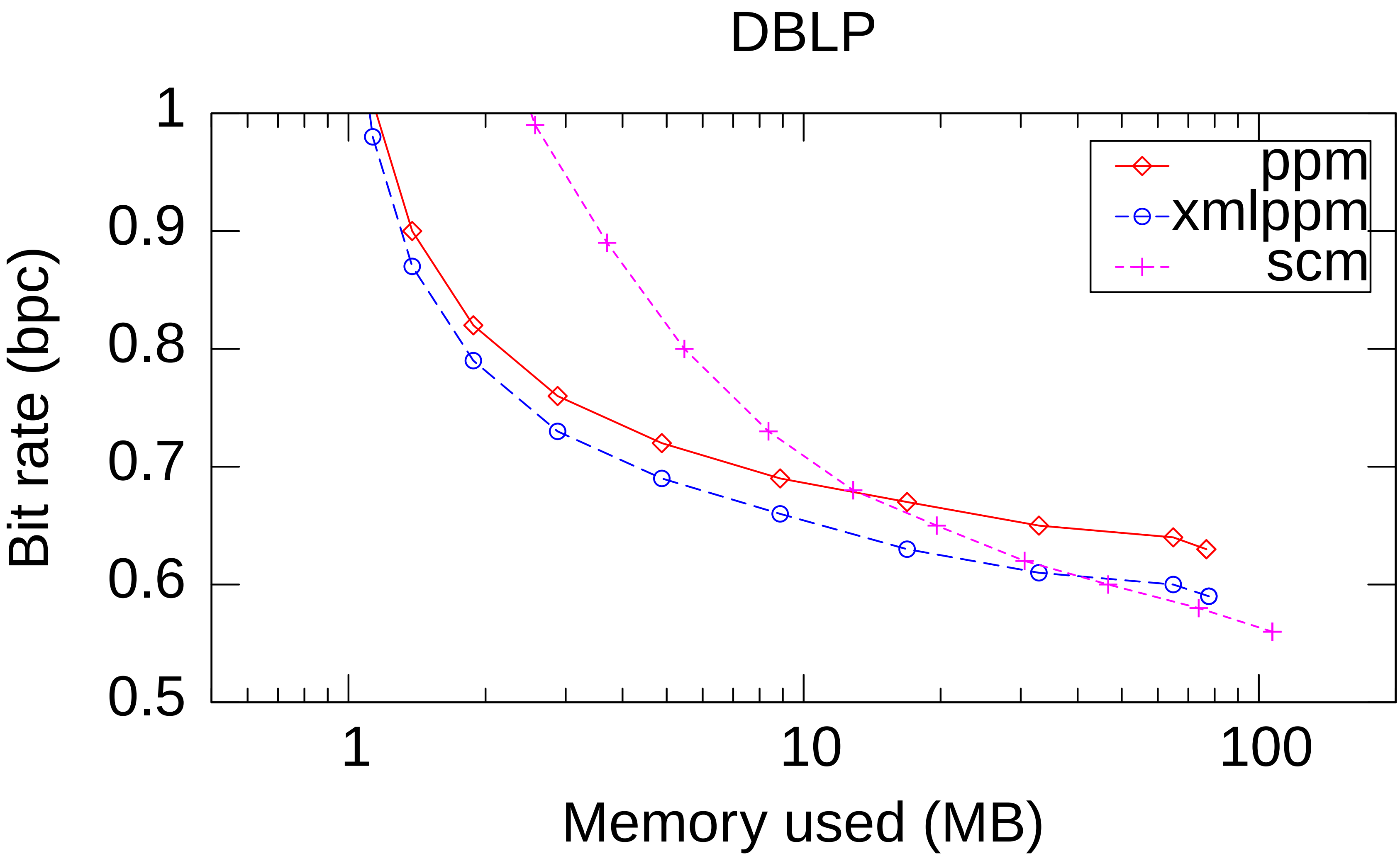
Experimental methodology

- Three experiments:
  1. Memory vs. compression rate: for a wide range of model sizes, measured *compression rate* vs. *memory used*
  2. Convergence rate: compressed *prefixes* of large files, and measured *prefix length* vs. *compression rate*
  3. Memory footprint (not shown): for a wide range of model sizes, measured *memory allocated* vs. *memory used*

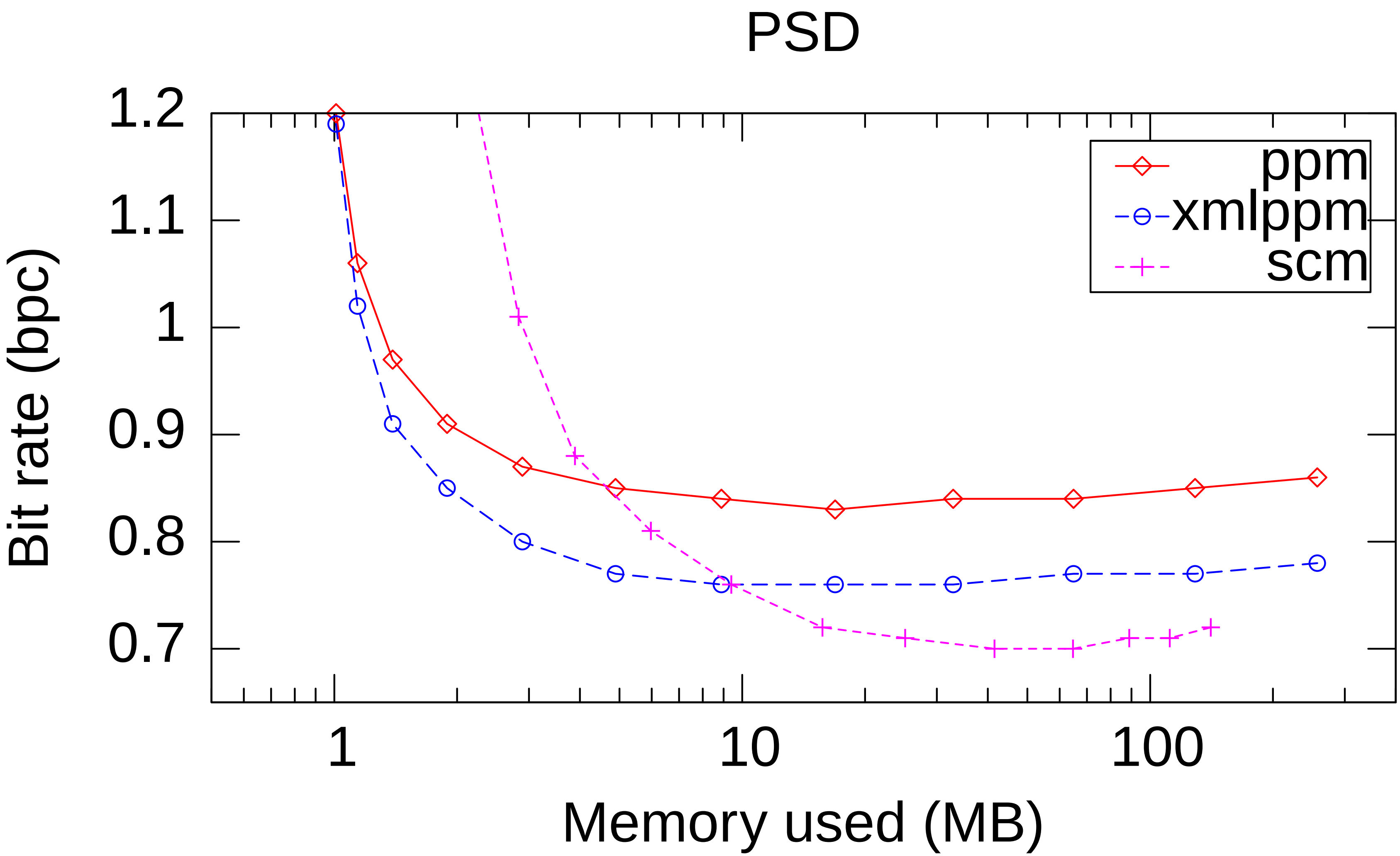
Experiments

- Used two large “typical” data sets:
  - DBLP (bibliography, 300MB uncompressed)
  - PSD (protein sequence database, 717MB uncompressed).
- Tested plain PPM, XMLPPM, SCM, and a “hybrid” (not shown)
- Further details in paper

Memory use vs. compression rate



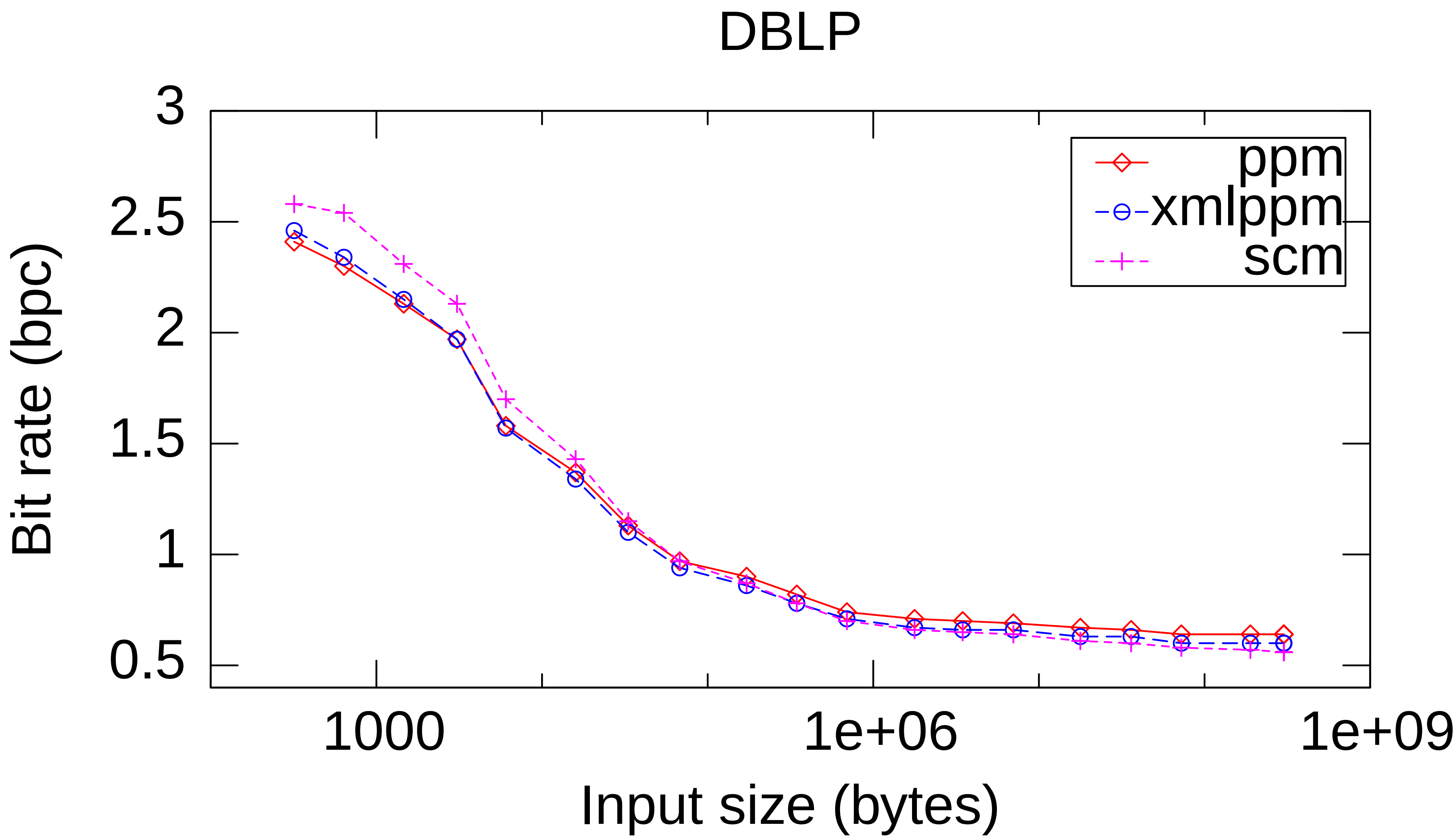
Memory use vs. compression rate



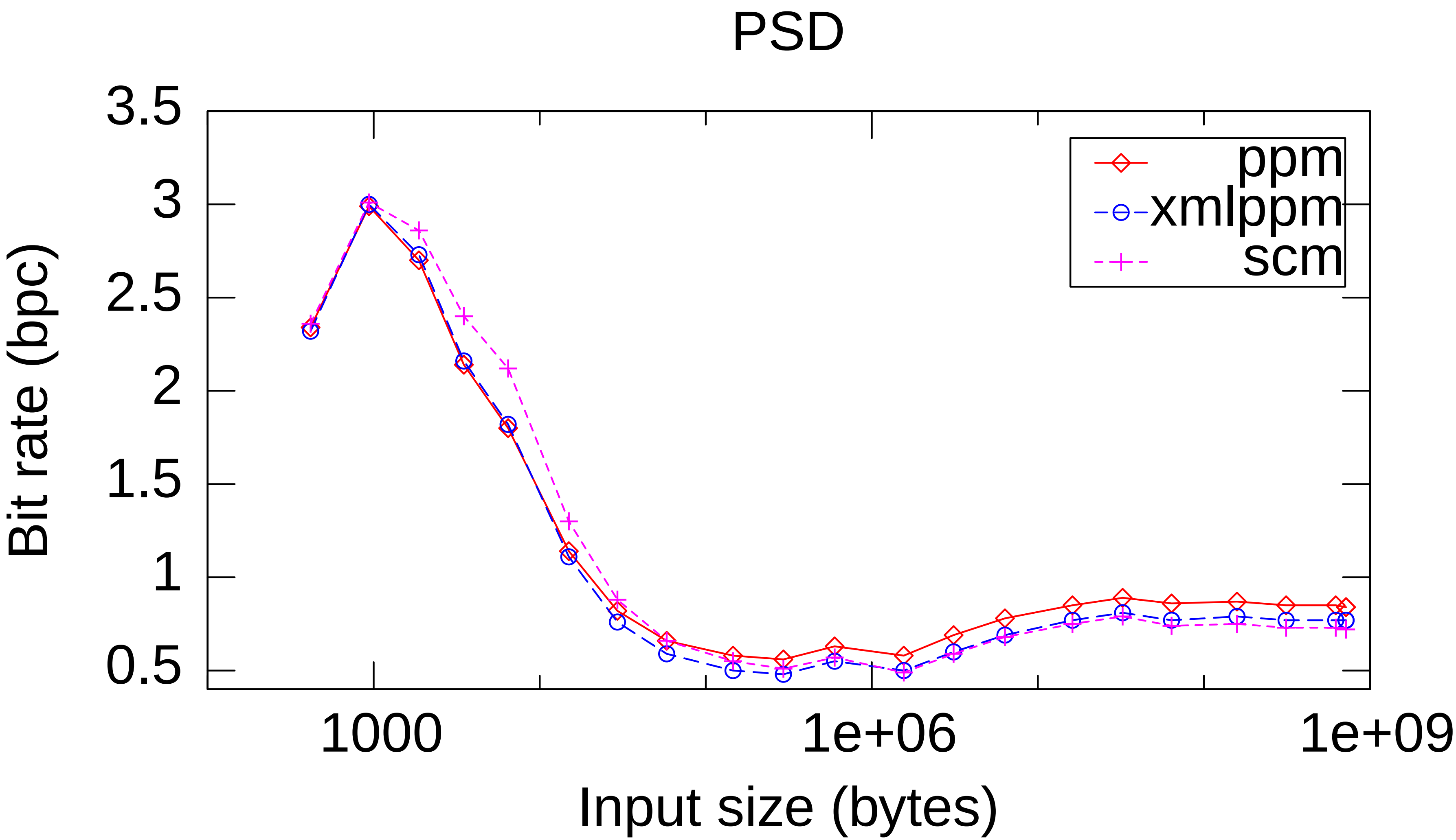
Memory use vs. compression rate

- For DBLP, improvement for SCM is minor (5%), needs over 40MB to achieve this.
- For PSD, SCM can perform around 10% better, improves after 10MB.
- Why?
  - Small XMLPPM models benefit from sharing common statistics
  - But large SCM models benefit from specialization

Convergence rate



Convergence rate



Convergence rate

- Overall trend: SCM performs worse early, but eventually better
- Why?
  - because SCM separates text under different elements, each model learns any common text separately
  - but because XMLPPM lumps all text into a single model, eventually it does worse because of averaging

Conclusions

- The SCM approach does provide better compression...
- provided you give it lots of memory and lots of data
  - Of course, for “archiving” XML DBs (DBLP, PSD, etc), this is fine!
- However, the XMLPPM approach is better for small documents or using small amounts of memory
  - This may make it preferable for on-the-fly compression of XML “messages”
  - webpages, RDF, RSS feeds, SOAP RPCs
  - Or low-memory devices such as PDAs, mobile phones

Meta-conclusions

- XML compression research is still wide open area
- However, so far experiments have focused on compression rate and ignored other factors
- More generally, standards for benchmarking and evaluating XML compression systems needed!
- Source code should also be made available to allow repeatability

`http://xmlppm.sourceforge.net`