XTRACT: A System for Extracting Document Type Descriptors from XML Documents

Minos Garofalakis
Aristides Gionis
Rajeev Rastogi
S. Seshadri
Kyuseok Shim

Bell Laboratories
Stanford University
KAIST and AITrc

XML Primer I

- Standard for data representation and data exchange
- Looks like HTML but it isn't
- Collection of elements
  - Atomic (raw character data)
  - Composite (sequence of nested sub-elements)
- Example

```
<book>
   <title>
      A Relational Model for Large Shared Data Banks
   </title>
   <author>
      <name> E.F. Codd </name>
      <affiliation> IBM Research </affiliation>
   </author>
</book>
```

XML Primer II

- XML documents can be accompanied by Document Type Descriptors (DTDs)
- DTDs serve the role of the schema of the document
- Specify a regular expression for every element
- Example

```
<!ELEMENT book (title, author*)
<!ELEMENT title (#PCDATA)>
<!ELEMENT author (name, affiliation)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT affiliation (#PCDATA)>
```

XTRACT System

- Goal:

  Infer DTDs from XML Documents

Why are DTDs useful?

- Efficient storage of XML data

  [Deutsch, Fernandez, Suciu '99],

  [Shanmugasundaram et al '99]

- Optimization of XML queries

  [Goldman, Widom '97],

  [Fernandez, Suciu '97]

Why need to infer DTD?

- DTDs are not mandatory $\Rightarrow$ might be missing
- XML databases that have been generated automatically,

  $\Rightarrow$ Relational databases

  $\Rightarrow$ Flat files

  $\Rightarrow$ HTML documents

  $\Rightarrow$ Bibliographies

Related Work I

- Database Community
  - Schema extraction from semistructured data
    [Nestorov, Abiteboul, Motwani '98]
    [Goldman, Widom '97]
    [Fernandez, Suciu '97]
- Focus on finding a typing for semistructured data
- Un-ordered vs. ordered

Related Work II

- Machine Learning Community
  - DFA inference
    [Gold '67], [Gold '78]
    [Angluin '78]
    [Pitt '89]
  - Roughly equivalent regular expression inference
    [Brazma '93]
  - Pattern language inference
    [Kipeläinen, Mannila, Ukkonen '95]

Problem Simplification and Abstraction

- Element types $\Rightarrow$ alphabet Σ
- Infer DTD for each element type separately
- Example sequences: instances of nested subelements

  $\Rightarrow$ Only one level down in the hierarchy
- Problem definition:
  - Given a set example sequences for element e
  - Infer a "good" DTD for e
- Hard problem since DTDs can be general, complex regular expressions

Example

Original XML document:

```
<book>
   <title> </title>
   <author>
      <name> </name>
      <affiliation> </affiliation>
   </author>
</book>
```

```
<paper>
   <title> </title>
   <author>
      <name> </name>
      <affiliation> </affiliation>
   </author>
   <conference> </conference>
   <year> </year>
</paper>
```

```
<book>
   <title> </title>
   <author>
      <name> </name>
      <address> </address>
   </author>
   <author>
      <name> </name>
      <address> </address>
   </author>
   <editor> </editor>
</book>
```

Example (Ctd)

Simplified example sequences:

```
<book> :- <title><author>,
          <title><author><author><editor>

<paper> :- <title><author><conference><year>

<author> :- <name><affiliation>,
            <name><affiliation>,
            <name><address>,
            <name><address>
```

Desirable solution:

```
<!ELEMENT book (title, author*, editor?)
<!ELEMENT paper (title, author, conference, year)>
<!ELEMENT author (name, affiliation?, address?)>
```

DTD Inference Requirements

- Requirements for a good DTD:
  - Generalizes to intuitively correct but previously unseen examples
  - It should be concise (i.e., small in size)
  - It should be precise (i.e., not cover too many sequences not contained in the set of examples)

- Example: Consider the case
  `p :- ta, taa, taaa, ta, taaaa`

Candidate DTD	Concise	Precise
`ta|taa|taaa|taaaa`	no	yes
`(t|a)*`	yes	no
`ta*`	yes	somewhat

DFA Minimization Doesn't Work

- Suggestion:
  - Consider all example sequences as positive examples of a language of a DFA
  - Find the minimum DFA that accepts the language
- However:
  - Minimum DFA does not imply an intuitive simple regular expression
  - DFA minimization algorithm finds equivalent language $\Rightarrow$ Does not generalize
  - How to infer ta^* from ta, taa, $taaa$, $taaaa$?

The XTRACT Approach: MDL Principle

- Minimum Description Length (MDL) quantifies and resolves the tradeoff between DTD conciseness and preciseness

- MDL principle states:

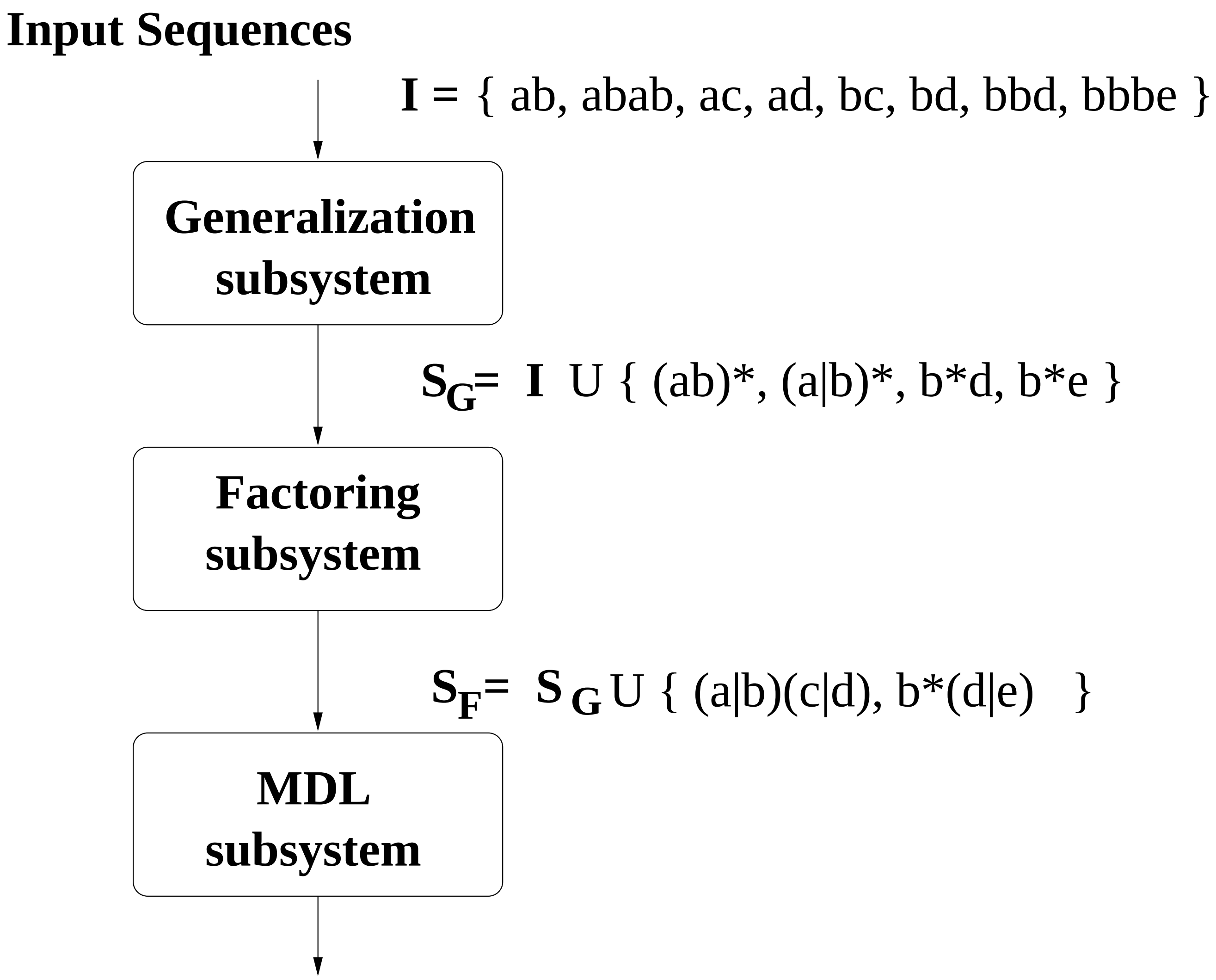
  The best theory to infer from a set of data is the one which minimizes the sum of

(A) the length of the theory, in bits, plus
(B) the length of the data, in bits, when encoded with the help of the theory.

- Part (A) captures conciseness, and
  Part (B) captures preciseness

Overview of the XTRACT System

XTRACT consists of 3 subsystems



Inferred DTD: (ab)* | (a|b)(c|d) | b*(d|e)

MDL Subsystem

- MDL principle:

  Minimize the sum of

  – Theory description length, plus
  – Data description length given the theory

- In order to use MDL, need to:

  – Define theory description length (candidate DTD)
  – Define data description length (input sequences) given the theory (candidate DTD)
  – Solve the resulting minimization problem

MDL Subsystem – Encoding Scheme

- Description Length of a DTD:
  - Number of bits required to encode the DTD
  - $|$Size of DTD$| * \lg |\Sigma \cup \{(,),|,*\}|$
- Description Length of a sequence given a candidate DTD:
  - Number of bits required to specify the sequence given DTD
  - Use a sequence of encoding indices

  - Encoding of a given a is the empty string ϵ
  - Encoding of a given $(a|b|c)$ is the index 0
  - Encoding of aaa given a^* is the index 3

  - Example: Encoding of $ababcabc$ given $((ab)^*c)^*$ is sequence 2,2,1

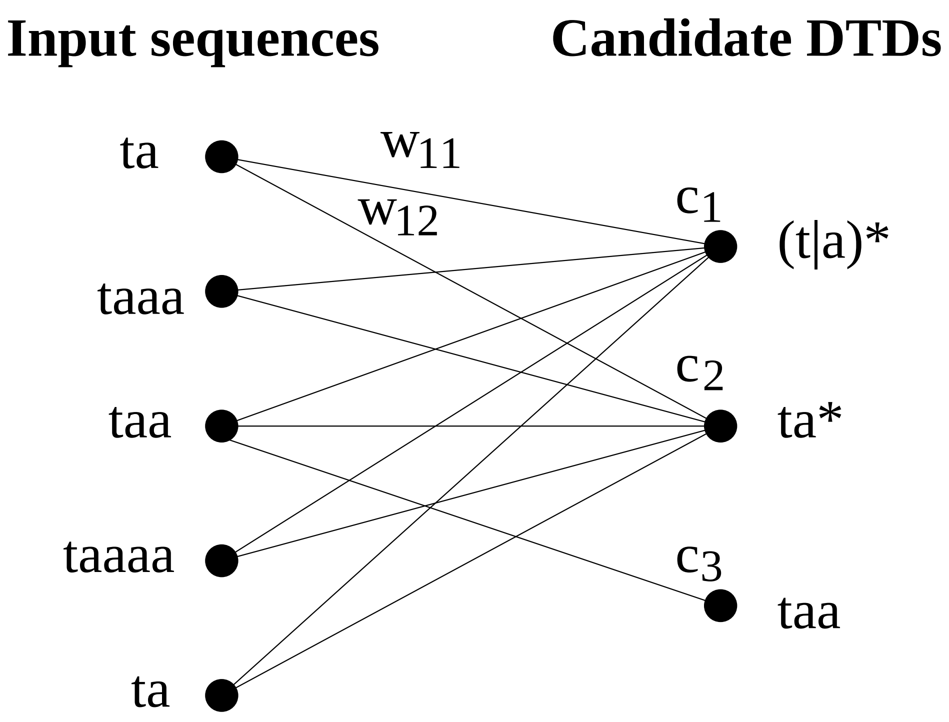
MDL Example

- Consider again the case

`p :- ta, taa, taaa, taaaa`

Theory Description	Data Description (given the theory)	Total Length
ta|taa|taaa|taaaa	0, 1, 2, 3	17 + 7 = 24
(t|a)*	201, 3011, 40111, 501111	6 + 21 = 27
ta*	1, 2, 3, 4	3 + 7 = 10

MDL Subsystem - Minimization



- Facility Location Problem (NP-hard)
- XTRACT Employs fast algorithms proposed by the Algorithms [Charikar, Guha '99] and Operations Research [Hochbaum '82] communities

Generalization Subsystem

- Goal:
  - Introduce metacharacters $*$, $|$
  - Produce candidate DTDs of the form
    a^*bc^*, $(abc)^*$, $(a|b|c)^*$, $((ab)^*c)^*$, etc.

- Combines two generalization heuristics

  1. $\textsc{DiscoverSeqPattern}(s,\ r)$
     $s = \alpha tt \ldots t\beta \Rightarrow \alpha t^* \beta$
     [Brazma '93]

  2. $\textsc{DiscoverOrPattern}(s,\ d)$
     Symbols $a_1, a_2, \ldots, a_m$ in close proximity $\Rightarrow$ $(a_1|a_2|\ldots|a_m)^*$

- Candidate DTDs are generated by calling the above functions for appropriate values of r and d

Generalization - Examples

1. $\textsc{DiscoverSeqPattern}(s,\, r)$

   Ex. $s = abababcabcababc,\ r = 2$
   $abababcabcababc \Rightarrow$
   $(ab)^*cabcababc \Rightarrow$
   $(ab)^*cabc(ab)^*c \Rightarrow$
   $((ab)^*c)^*$

2. $\textsc{DiscoverOrPattern}(s,\, d)$

   Ex. $(s = abcbac,\ d = 2)$
   $abcbac \Rightarrow a(b|c)^*ac$
   Ex. $(s = abcbac,\ d = 3)$
   $abcbac \Rightarrow (a|b|c)^*$

Factoring Subsystem

- Goal: Combine different candidates to derived more compact, factored DTDs forms
  - E.g., ac, ad, bc, $bd \Rightarrow (a|b)(c|d)$
- MDL modeling justification: Reduces description length of the candidate DTDs
- Intuitive justification: Captures DTDs like

```
<!ELEMENT article(title, author*,
(workshop | conference | journal),
(compute science | physics | chemistry | ...))
```

- Captures optional elements
  $ab, a \Rightarrow a(b|\epsilon) \Rightarrow ab?$
- Adaptation of the factoring algorithms for boolean expressions in logic optimization literature
- Novel heuristic algorithms for selecting subsets of candidate DTDs that give good factored forms

Experimental Results

- Compare with IBM alphawork DDbE extraction tool (Data Description by Example) [1]

- Use both Synthetic and Real-life DTDs

- Data Sets: Generate 1000 random example sequences for each DTD
  - Feed input sequences into XTRACT and DDbE and compare output with original DTD

[1] DDbE tool can be found at `http://www.alphaworks.ibm.com/`

Model for Random Sequence Generation

Goal: Generate random example sequences for a given DTD

- Need to instantiate metacharacters $|$ and $*$
- Pick instantiation of $*$ using geometric distribution
- Among operands of $|$ pick with equal probability
- $Pr[\text{Generate } aab \text{ from } (a|b)*] = p^{1/3}(1-p) * (\frac{1}{2})^3$

IBM Alphawork DDbE Extraction Tool

- Java component library for inferring DTD from XML data instances
- Version 1.0
- Control parameters:

  c Maximum number of consecutive identical tokens not replaced by a list (default $= 1$)
  d Maximum depth of factorization (default $= 2$)

Results on Synthetic DTDs

- Test on 10 DTDs of increasing complexity

No.	Original DTD
1	$abcde\|efgh\|ij\|klm$
2	$(a\|b\|c\|d\|f)^*gh$
3	$(a\|b\|c\|d)^*\|e$
4	$(abcde)^*f$
5	$(ab)^*\|cdef\|(ghi)^*$
6	$abcdef(g\|h\|i\|j)(k\|l\|m\|n\|o)$
7	$(a\|b\|c)d^*e^*(fgh)^*$
8	$(a\|b)(cdefg)^*hijklmnopq(r\|s)^*$
9	$(abcd)^*\|(e\|f\|g)^*\|h\|(ijklm)^*$
10	$a^*\|(b\|c\|d\|e\|f)^*\|gh\|(i\|j\|k)^*\|(lmn)^*$

- XTRACT recovers each single one of them

- DDbE behavior is inconsistent
  - Recovers correctly only the first one (simplest)
  - Deduced DTDs are over-generalizations
  - Do not even cover all the examples
  - Not good in factoring

Real-life DTDs

Test on 5 real-life DTDs [2]

<table>
<tr><th>No</th><th>Original DTD</th><th>Simplified DTD</th></tr>
<tr><td>1</td><td><!ENTITY included-elements (audio-clip|blind-box-reply| link|pi-char|video-clip)></td><td>$a|b|c|d|e$</td></tr>
<tr><td>2</td><td><!ELEMENT communications-contacts (phone|fax|email|pager|web-page)*></td><td>$(a|b|c|d|e)^*$</td></tr>
<tr><td>3</td><td><!ELEMENT employment-services (employment-service.type, employment-service.location* (e.zz-generic-tag)*)></td><td>ab^*c^*</td></tr>
<tr><td>4</td><td><!ENTITY location (addr*,area?, city?,state?,zip-code?,country?)></td><td>$a^*b?c?d?$</td></tr>
<tr><td>5</td><td><!ELEMENT transfer-info (transfer-number,(from-to,co-id)+, contact-info)*></td><td>$(a(bc)^+d)^*$</td></tr>
<tr><td>6</td><td><!ELEMENT real-estate-services (r-e.type,r-e.location?, r-e.response-modes*,r-e.comment?)*></td><td>$(ab?c^*d?)^*$</td></tr>
</table>

[2] Taken by NAA Classified Advertising Standards Task Force

Results on Real-life DTDs

No	Simplified DTD	DTD Obtained by XTRACT	DTD obtained by DDbE
1	$a\|b\|c\|d\|e$	$a\|b\|c\|d\|e$	$a\|b\|c\|d\|e$
2	$(a\|b\|c\|d\|e)^*$	$(a\|b\|c\|d\|e)^*$	$(a\|b\|c\|d\|e)^*$
3	ab^*c^*	ab^*c^*	$(ab^+c^*)\|(ac^*)$
4	$a^*b?c?d?$	$a^*b?c?d?$	$(a^+b(c\|(c?d))?)\|$ $((b\|a^+)?cd)\|$ $((a^+\|b)?d)\|$ $((a^+\|b)?c)\|a^+\|b)$
5	$(a(bc)^+d)*$	$(a(bc)^*d)^*$	$(a\|b\|c\|d)^+$
6	$(ab?c^*d?)*$	–	$(a\|b\|c\|d)^+$

Concluding Remarks

- Inference of DTDs can be used for improving database storage and for query optimization
- Hard problem! – Naive approaches fail to produce intuitive, meaningful DTDs
- XTRACT employs MDL to control the tradeoff between model simplicity and model generalization
- XTRACT architecture can be used in other applications that need to extract regular expressions
- Experimental results provide strong support