An efficient Java implementation of the immediate successors calculation

Clément Guérin
Karell Bertet
Arnaud Revel
Context

• eBDtheque project
  – 2 PhD candidates, 4 professors

• Two research axis
  – Extraction of comic books visual items
  – Reasonning on these items

• One goal: how to help comic books to take the right turn on their digital metamorphosis?
Comic books domain model

- A model combining comic books’ and image’s domain knowledge

Reference
C. Guérin, K. Bertet, A. Revel
Ontologies and spatial relations applied to comic books reading
EKAW’s PhD Symposium, Galway, October 2012
What for?

• Tools to ease the conversion from printed to digital comic books
  – Panels and balloons ordering
  – Association of balloons to characters
  – Scene understanding

• Information retrieval
  – Word spotting
  – Linguistic study
  – Semantic Content Base Image Retrieval
Semantic Content Based Image Retrieval

• From classical Content Based Image Retrieval (CBIR)

  Exemple as query input → Visually similar results as output → Results labelling → Refinement of the query

• Use the model description of the panels to retrieve semantically and/or visually similar panels

Reference
C. Guérin, K. Bertet, A. Revel
An approach to Semantic Content Based Image Retrieval using Logical Concept Analysis. Application to comicbooks.
FCA4AI Workshop of ECAI, Montpellier, August 2012
What annotations?

• Panels attributes
  – shape, size, position, rank, shot type, etc.

• What is “inside” a panel
  – speech balloons, characters, words, objects, visual features...

• What is “outside” a panel
  – the author of the comic, its style, its year of publication, etc.
### Example

<table>
<thead>
<tr>
<th></th>
<th>p1</th>
<th>p2</th>
<th>p3</th>
<th>p4</th>
<th>p5</th>
</tr>
</thead>
<tbody>
<tr>
<td>contains:balloon</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>shape:wide</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>shape:high</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>size:medium</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>contains:tree</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Browsing context

• Formal Concept Analysis
  – Concept lattices gather panels described by the same set of attributes in concepts
  – Set of panels (resp. attributes) ordered in a browsable hierarchical structure

• Need to get the successors and predecessors of a concept very quickly

Reference
C. Demko, K. Bertet
Generation algorithm of a concept lattice with limited object access
CLA, Nancy, October 2011
Lattice Java API

- **Java language**
  - Widely used programming language
  - Lattice Java API

- **Implemented immediate successors algorithm from Bordat’s (or Lindig’s) theorem**

  **Data:** A context $K = (O, I, (\alpha, \beta))$; A closure $B$ of the closure lattice of $K$

  **Result:** The immediate successors of $B$ in the closure lattice

  begin
  Init the set system $F_B$ with $\emptyset$;
  foreach $x \in I \setminus B$ do
    Add $\varphi(x + B)$ to $F_B$
  end
  $Succ=$minimal inclusion subsets of $F_B$;
  return $Succ$
  end

- **Not fast enough**
Limited Object Access algorithm

• Improved efficiency by reducing the subset of observations to its cardinality

• The count function $c$ associates to any subset $X$ of attributes the cardinality of the subset $\beta(X)$

\[ \varphi(B + X) \subseteq \varphi(B + x) \iff c(B + X + x) = c(B + X) \]

• Refer to the reference paper for proof and details
Good performances using SQL indexing mechanism

How to get it efficient without SQL?

Reference

C. Demko, K. Bertet
Generation algorithm of a concept lattice with limited object access
CLA, Nancy, October 2011
LOA Java implementation

• Naive implementation of LOA using “classical” data containers (TreeSet, HashSet...)

• The most time consuming task in the immediate successors computation is the calculation of the extent

• The most time consuming task in the extent computation is the intersection between two sets of elements

• Time can be won here

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Extent</th>
</tr>
</thead>
<tbody>
<tr>
<td>contains:balloon</td>
<td>{p1, p2, p3, p4}</td>
</tr>
<tr>
<td>shape:wide</td>
<td>{p2,p5}</td>
</tr>
<tr>
<td>shape:high</td>
<td>{p1, p4}</td>
</tr>
<tr>
<td>size:medium</td>
<td>{p3,p5}</td>
</tr>
<tr>
<td>contains:tree</td>
<td>{p4,p5}</td>
</tr>
</tbody>
</table>
Binary words

- Objects and attributes are reduced to their index
  - They have to be ordered though
  - The index is stored to retrieve
- Extents and intents are expressed as binary words using Java’s BitSet
- Intersection is performed by a logical AND
  - \( E(n/w) \) operations

<table>
<thead>
<tr>
<th></th>
<th>p1</th>
<th>p2</th>
<th>p3</th>
<th>p4</th>
<th>p5</th>
</tr>
</thead>
<tbody>
<tr>
<td>contains:balloon</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>shape:high</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>contains:tree</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Extent of (b,h,t)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Experiment

• First dataset:
  – 848 comic books’ panels
  – 100 attributes (avg. 7 attributes/panels)

• Second dataset:
  – 848 comic books’ panels
  – 3533 attributes (avg. 15 attributes/panels)
  – 3403 of these are shared by less than 3 panels
Results

• Calculation of the *immediate successors* of the *bottom concept* and *immediate predecessors* of the *top concept*

<table>
<thead>
<tr>
<th>Computation time (in ms)</th>
<th>Immediate successors</th>
<th>Immediate predecessors</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(</td>
<td>O</td>
</tr>
<tr>
<td></td>
<td>(</td>
<td>I</td>
</tr>
<tr>
<td>Classical + TreeSet</td>
<td>3.06</td>
<td>11767.52</td>
</tr>
<tr>
<td>Classical + BitSet</td>
<td>0.77</td>
<td>196.58</td>
</tr>
<tr>
<td>LOA + TreeSet</td>
<td>0.29</td>
<td>11.26</td>
</tr>
<tr>
<td>LOA + BitSet</td>
<td>0.02</td>
<td>0.15</td>
</tr>
</tbody>
</table>

• 0.18 second on 500 randomly picked concepts
Conclusion & perspectives

• Conclusions
  – Processing time kept below the second on a reasonable machine
  – Efficient way to browse a context without generating its lattice

• Perspectives
  – Limit the impact of the amount of attributes on the performances
  – Make that library available
Thank you.
Any questions?