Topic Map Erotica

RDF and Topic Maps
"*in flagrante*"

Steve Pepper, CEO, Ontopia
Convenor SC34/WG3, Editor XML Topic Maps
pepper@ontopia.net

 http://www.ontopia.net/

The Ontopia MapMaker

Leveraging RDF to autogenerate Topic Maps

Steve Pepper, CEO, Ontopia
Convenor SC34/WG3, Editor XML Topic Maps
pepper@ontopia.net

 http://www.ontopia.net/

Ontopia's autogeneration approach

- **There are many sources of topic map data**
  - Document metadata, structured content, databases, even semi- and unstructured data
- **Many operations are simple manipulations of flat blocks of property/value pairs**
  - Especially true in extracting topics and associations from metadata, relational databases and XML documents
  - Suggests that RDF might be a suitable technology to exploit
- **There are a large number of common operations**
  - Extracting metadata property/value pairs from XML
  - Converting data types, normalising values
  - Splitting single values into multiple values
  - Traversing HTTP links, etc.
  - Suggests a modular approach in order to foster reuse

Ontopia's MapMaker™ Toolkit

- **A framework and toolkit for autogenerating topic maps**
- **Used extensively in project work, but not (yet) a separate product**
- **Consists of a set of configurable processing modules, which are chained together according to the needs of each individual autogeneration application**
- **Modules have access to an RDF model that is constructed during processing**
  - Some modules produce, others consume, and some modify
- **The RDF is cleaned and extended, finally converted to a topic map**

Module examples

- **RDF-based processing**
  - Producers:
    - Directory walking, POP3 readers, DMS extractors...
  - Modifiers:
    - XML extractor, Regexp processor, Token collector, File splitter, ...
  - Consumers:
    - RDF2TM converter, RDF writer, ...
  - Advanced processing:
    - NLP processing, inferencing, querying...
- **Topic Map-based processing**
    - TM loader, DuplicateSuppressor, TM exporter...

MapMaker demo

- **The goal:**
  - To create a topic map from a USENET news group archive
- **Starting point:**
  - A set of archive files, located in a subdirectory structure
  - Each file contains a number of news articles
  - Each article has an RFC822 header containing metadata
  - The metadata is the principle source of topic map data
  - This example uses a particularly interesting archive of news articles
    - The interest is in the type of metadata fields it contains

Sample text (expurgated)

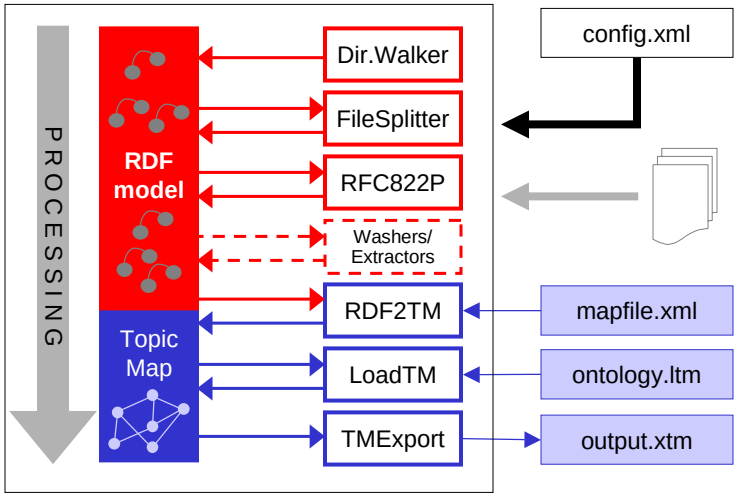
```
Article: 1107 of rec.arts.erotica
Path: ifi.uio.no!nntp.uio.no!trane.uninett.no!sunic!pipex!uunet!
news.mtholyoke.edu!news.amherst.edu!not-for-mail
From: CMSJMETZ@UGA.cc.uga.edu (J Metz)
Newsgroups: rec.arts.erotica
Subject: Switcheroo
Followup-To: alt.sex.stories.d
Date: 22 Feb 1994 21:50:05 -0500
Organization: University of Georgia
Lines: 691
Sender: erotica@unix.amherst.edu
Approved: erotica@unix.amherst.edu (Tim Pierce)
Message-ID: <2keg8t$ju5@amhux3.amherst.edu>
NNTP-Posting-Host: amhux3.amherst.edu
Keywords: mf ff trans sf
X-Moderator-Review: 3: reads like Pepe Le Pew on acid

Archive-name: switcheroo

Switcheroo

     Up on the mountain, Johar was bored.  Bored, bored bored...
```


The MapMaker demo



RDF processing chain

- Setup
- Collect list of file names
- Split into individual files
- Process news group headers
- Split complex headers
- Reinterpret certain relationships

(1) Set up templates

- URI template declarations

```
<uri-template prefix  ="mm"
              template="http://psi.ontopia.net/mapmaker#%s"/>
<uri-template prefix  ="rfc822"
              template="http://psi.ontopia.net/mapmaker/rfc822#%s"/>
```

  - Now "mm" and "rfc822" can be used as prefixes when creating and using URIs
  - This simplifies and shortens the code in the remainder of the chain configuration

(2) Collect list of file names

- **Uses DirectoryWalker module**
  - Input parameters: root directory, file name pattern
  - RDF statements used: none

```
<directory-walker directory = "src-data"
                  pattern   = "text*.txt">
  <out property = "rdf:type" value = "mm:Infile"/>
</directory-walker>
```

  - Output: One RDF statement for each file to be processed:

```
• ("file:/.../src-data/text1.txt",
   "rdf:type",
   "http://psi.../mapmaker:Infile")
  etc.
```


(3) Split into individual files

- **Uses FileSplitter module**
  - Input parameters: Name of file, separator string, output directory
  - RDF statements used: rdf:type="mm:Infile"

```
<file-splitter separator     = "&#10;&#10;Article: "
               out-directory = "src-data/split"
               filename      = "story-%s.txt">
  <in seek-property = "rdf:type" seek-value = "mm:Infile"/>
  <out property = "rdf:type" value = "mm:Story"/>
</file-splitter>
```

  - Output: A new set of files (one article per file) and one RDF statement for each article to be processed:

```
• ("file:/.../src-data/split/story-1.txt",
   "rdf:type",
   "http://psi.../mapmaker#Story")
  etc.
```

(4) Process news group headers

- **Uses RFC822Parser module**
  - Input parameters: None
  - RDF statements used: rdf:type="http://psi.../mapmaker#Story"
  - Output: One RDF statement per header field for every story:
  - ```
    ("file:/.../src-data/split/story-1.txt",
     "http://psi.../mapmaker/rfc822#From",
     "CMSJMETZ@UGA.cc.uga.edu (J Metz)")
    ```
  - ```
    ("file:/.../src-data/split/story-1.txt",
     "http://psi.../mapmaker/rfc822#Subject",
     "Switcheroo")

    ("file:/.../src-data/split/story-1.txt",
     "http://psi.../mapmaker/rfc822#Keywords",
     "mf ff trans sf")

    etc.
    ```


(5) Split complex headers

- **Uses RegExpGrouper, StringExtractor, and ValueSplitter modules**
- **Example 1:**
  - Split "From" property into Author and Email, e.g.
    - From: "CMSJMETZ@UGA.cc.uga.edu (J Metz)"
  - becomes
    - Author: "J Metz"
    - Email: "CMSJMETZ@UGA.cc.uga.edu"
- **Example 2:**
  - Split "Keywords" property into multiple "Keyword" properties, e.g.
    - Keywords: "mf ff trans sf"
  - becomes
    - Keyword: "mf"
      Keyword: "ff"
      etc.

(6) Reinterpret certain relationships

- **Uses PropertyMover module**
- **Initially, the subject of every property generated by processing a story is the story itself**
- **This is also the case after properties have been split**
- **In some cases this hides relationships between the objects of a group of properties,**
  - For example
    - ("story1", "author", "J Metz") and ("story1", "email", "CMSJMETZ@UGA.cc.uga.edu")
  - should really be
    - ("story1", "author", "J Metz") and ("J Metz", "email", "CMSJMETZ@UGA.cc.uga.edu")
- **The PropertyMover rearranges properties to handle this**

Topic map processing chain

- **Map RDF to topic map**
- **Suppress duplicate topic characteristics**
- **Merge in pre-existing topic map containing typing topics**
- **Export to XTM syntax**

(7) Map RDF to topic map

- **Uses RDF2TM module**
- **A generic mapping from RDF to topic maps is not useful, because of the difference in levels of semantics**
- **Mappings are better performed at the schema level**
- **RDF2TM uses a mapping file that explains how to map each predicate to a topic map construct**
- **Predicates may be mapped to:**
  - subjectIndicator
  - instanceOf
  - baseName
  - occurrence
  - association

(8) Suppress duplicate topic characteristics

- **Uses DuplicateSuppressor module**
- **Document metadata typically contains a lot of redundancy**
- **For example**
  - Every article contains both the name and the email of the author
  - Every article contains both the author and her affiliation
- **This leads to many duplicate base names, occurrences and associations**
  - DuplicateSuppressor takes care of this situation

(9) Merge in ontology topic map

- Uses TMLoader module
- Nothing in the source data provides the names of the topic types, occurrence types, association types and association role types
- These are stored in a separate topic map (in our case, using LTM syntax)

```
[keyword = "Keyword" @"http://psi.garshol.priv.no/tekst#Keyword"]
[story   = "Story"   @"http://psi.garshol.priv.no/tekst#Story"]
[person  = "Person"  @"http://psi.garshol.priv.no/tekst#Person"]

[dom    = "Domination"         @"http://psi.garshol.priv.no/tekst#dom"]
[hist   = "Historical"         @"http://psi.garshol.priv.no/tekst#hist"]
[gothic = "Gothic"             @"http://psi.garshol.priv.no/tekst#gothic"]
[humor  = "Humorous"           @"http://psi.garshol.priv.no/tekst#humor"]
```

- This topic map is merged in using TMLoader

(10) Export to XTM syntax

- Uses TMExporter module
- Writes out the resulting topic map from its internal representation in the topic map engine to either XTM, HyTM or LTM

Demo

(For Adults Only)

Concluding remarks

- **This demo has shown the use of the MapMaker in typical batch mode**
  - Most useful with “dirty data” that requires manual clean-up
- **The MapMaker can also be used in event-driven mode with a topic map server**
  - In this case events in a data source trigger MapMaker chains and send requests to the server
  - Scoping assertions by source allows us to deal with updates
- **Conclusions**
  - Automated topic map generation is an feasible, efficient, and cost-effective way of creating and maintaining topic maps
    - But, as always, the GIGO principle applies!
  - RDF and topic maps are complementary: The goal should be to exploit their synergies

Want to know more about topic maps?

- **Ontopia Web Site: http://www.ontopia.net**
  - Literature and references
  - Online demos
- **Download Ontopia's free Omnigator**
  - Play with the enclosed topic maps
  - Experiment with your own topic maps (step-by-step tutorial included!)
- **Call the Ontopians for consultancy and training**
  - +47 23233080
  - info@ontopia.net