FORMAL CONCEPT DISCOVERY IN SEMATIC WEB DATA

Markus Kichberg\textsuperscript{1} \hspace{1cm} Erwin Leonardi\textsuperscript{1} \hspace{1cm} Tan Yu Shyang\textsuperscript{1} \hspace{1cm} \\
Sebastian Link\textsuperscript{2} \hspace{1cm} Ryan Ko\textsuperscript{1} \hspace{1cm} Francis Lee\textsuperscript{1} \\

\textsuperscript{1}Hewlett-Packard Laboratories Singapore \hspace{1cm} \textsuperscript{2}Auckland University, New Zealand
Outline

• Introduction
  • Semantic Web
  • Linked Open Data
• Related Work
• Web of Concept
• Applying FCA to Semantic Web Data
• Experimental Results
• Conclusion
Introduction

• Semantic Web
  • To enable machines to understand WWW contents
  • A machine-processable Web of Knowledge

Web of Documents \(\longrightarrow\) Web of Data (WoD) \(\longrightarrow\) Web of Knowledge (WoK)

• Web of Data
  • Integration and combination of data drawn from diverse sources
  • Resource Description Framework (RDF)
  • Linked Open Data
RDF

• W3C recommendation
• A standard model for data interchange on the Web
• Used to represent information about resources on the Web by using *Uniform Resource Identifiers*, or URIs

• RDF triple
  • In the form of Subject – Predicate – Object

<conf:ICFCA2012>  <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>  <conf:Conference>

• RDF quad
  • Add *context* into triple
  • In the form of Subject – Predicate – Object – Context
Linked Open Data
Linked Open Data

• LOD Core Stack

• SPARQL
  • A RDF query language
  • http://www.w3.org/TR/rdf-sparql-query/
Our Vision

• The large gap between WoD and WoK
  • LOD’s rapid growth
  • Varying data-set compliance w.r.t. LOD guidelines
  • Complexity of computations over large data-sets
  • Lack of tools and applications

Web of Documents  → Web of Data      → …      → Web of Knowledge

• Barriers for WoK
  • Creation
  • Usage
• To supplement the Web of Data with a concept layer to bridge this gap
• Partitions RDF triples into equivalence classes of semantically related facts
Related Work

- **FCA and Semantic Web**
  - Ontology querying, browsing and visualization [Tane et al., 2006]
  - Interactive ontology merging [Maedche et al., 2001]
  - Ontology learning from text [Cimiano et al., 2005]
  - Interactive completion of ontologies [Völker et al., 2008]
  - Extracting representative questions over a given RDF data-set [d’Aquin et al., 2011]

- **FCA Algorithm**
  - CbO algorithms (i.e., PCbO, FCbO, and PFCbO)
  - In-Close algorithm
Web of Concept

- Concepts can be computed using the well-founded FCA approach
- Benefits and value to Semantic Web efforts
  - Concepts facilitate understanding without having ontologies at hand;
  - Concepts will lower costs of creating, maintaining, integrating and exchanging ontologies;
  - Concepts can be automatically computed from RDF-encoded facts (at scale) and are context-aware;
  - Concepts reduce the difficulty in writing SPARQL queries and semantic rules as data can be better understood; and
  - Concepts are easier to understand by consumers and allow for verification of data sources by following the path of computation, i.e., supporting trust and provenance.
Applying FCA to Semantic Web Data
Semantic Web Data Properties

- LOD consists of 295 officially acknowledged data-sets;
  - media,
  - geographic,
  - government (largest wrt. triples),
  - publications (largest wrt. number of data-sets),
  - cross-domain,
  - life sciences (largest wrt. out-links)
  - user-generated content
- Diverse and dynamic data sources
- Potentially differs from what FCA algorithms have been applied to in the past
## Semantic Web Data Properties

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Size (in byte)</th>
<th># of Objects</th>
<th># of Attributes</th>
<th>Matrix Density</th>
<th>Matrix Size</th>
<th># of Concepts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adult (USA Census Income)</td>
<td>N/A</td>
<td>32,561</td>
<td>124</td>
<td>12.09%</td>
<td>4,038k</td>
<td>1,064,875</td>
</tr>
<tr>
<td>Mushroom</td>
<td>N/A</td>
<td>8,124</td>
<td>119</td>
<td>21.01%</td>
<td>967k</td>
<td>238,710</td>
</tr>
<tr>
<td>DBPedia Languages</td>
<td>82,958</td>
<td>316</td>
<td>169</td>
<td>0.931%</td>
<td>53k</td>
<td>187</td>
</tr>
<tr>
<td>DBPedia Drugs</td>
<td>365,150</td>
<td>2,162</td>
<td>459</td>
<td>0.234%</td>
<td>992k</td>
<td>504</td>
</tr>
<tr>
<td>DBPedia Drugs v2</td>
<td>2,923,649</td>
<td>4,726</td>
<td>1,245</td>
<td>0.287%</td>
<td>5,884k</td>
<td>9,012</td>
</tr>
<tr>
<td>DBPedia Country</td>
<td>2,948,530</td>
<td>2,345</td>
<td>5,709</td>
<td>0.115%</td>
<td>13,388k</td>
<td>8,316</td>
</tr>
<tr>
<td>UK Crime Locations</td>
<td>6,910,550</td>
<td>31,936</td>
<td>20,707</td>
<td>0.005%</td>
<td>661,299k</td>
<td>20,708</td>
</tr>
<tr>
<td>DBPedia Alma-mater</td>
<td>7,595,917</td>
<td>27,383</td>
<td>5,407</td>
<td>0.029%</td>
<td>148,060k</td>
<td>13,605</td>
</tr>
<tr>
<td>DBPedia Genre</td>
<td>10,553,958</td>
<td>26,672</td>
<td>2,145</td>
<td>0.113%</td>
<td>57,211k</td>
<td>21,805</td>
</tr>
</tbody>
</table>

SW data has very low matrix density values (less than 1%)

SW data can have thousands (if not millions) of objects

- [http://fcarepository.com/](http://fcarepository.com/)
- [http://fimi.ua.ac.be/data/](http://fimi.ua.ac.be/data/)
Questions

1. Obtaining data from the Web (instantaneously) causes a considerable overhead.
   - Can the overhead be quantified wrt. downloading the data and converting it to FCA input formats (FIMI versus CXT)?
   - Will the overhead of obtaining and converting data mitigate concept computation time?

2. Considering the significant differences in FCA matrix density, how will common FCA implementations perform on Web data?

3. How good or bad do common FCA implementations scale wrt. the number of objects?
Experiments

• Web Data Extraction and Preparation
  • To determine the overhead of downloading Web data and converting it to FCA input formats (FIMI and CXT)

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Download time *</th>
<th>Object set to DB *</th>
<th>Attr. set to DB *</th>
<th>Context matrix to DB *</th>
<th>Generate FIMI format *</th>
<th>Generate CXT format *</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBPedia Languages</td>
<td>3.56</td>
<td>0.324</td>
<td>0.256</td>
<td>0.515</td>
<td>0.576</td>
<td>0.209</td>
</tr>
<tr>
<td>DBPedia Drugs</td>
<td>4.94</td>
<td>0.721</td>
<td>0.348</td>
<td>0.962</td>
<td>0.180</td>
<td>0.249</td>
</tr>
<tr>
<td>DBPedia Drugs v2</td>
<td>20.22</td>
<td>0.153</td>
<td>0.644</td>
<td>0.547</td>
<td>0.803</td>
<td>1.694</td>
</tr>
<tr>
<td>DBPedia Country</td>
<td>16.48</td>
<td>0.736</td>
<td>0.189</td>
<td>0.504</td>
<td>0.695</td>
<td>4.405</td>
</tr>
<tr>
<td>UK Crime Locations</td>
<td>30.72</td>
<td>1.359</td>
<td>0.932</td>
<td>1.336</td>
<td>0.206</td>
<td>161.52</td>
</tr>
<tr>
<td>DBPedia Alma-mater</td>
<td>40.18</td>
<td>1.305</td>
<td>0.199</td>
<td>1.468</td>
<td>0.236</td>
<td>39.714</td>
</tr>
<tr>
<td>DBPedia Genre</td>
<td>63.66</td>
<td>1.203</td>
<td>0.187</td>
<td>2.351</td>
<td>0.305</td>
<td>15.077</td>
</tr>
</tbody>
</table>

* in seconds
Experiments

- Web Data Extraction and Preparation Overall
Experiments

- FCA Algorithms
  - PCbO, FCbO, PFCbO, and In-Close2
- Ensuring fairness
  - PCbO and FCbO only accept FIMI-formatted input files,
  - In-Close2 only supports CXT-formatted input files;
  - PFCbO supports both input formats.
  - Since CXT input files are significantly larger, we differentiate between input file reading time and execution time
  - For concept computation performance evaluation, we mainly consider execution time
  - We modified all algorithms to only return the intent portion of computed concepts.
Experiments

- Algorithms Modifications
  - PCbO:
    - Several memory leaks were closed.
  - PFCbO:
    - Several memory leaks were closed, memory management was tweaked to support in-memory data structures holding more than 2GB of data (on 64bit operating systems), and static output buffer memory allocation was adjusted to gather for large data-sets.
  - In-Close2:
    - Code was ported from the original Windows implementation to Linux; interactive parts were removed and two memory configurations were prepared to suit the tested data-sets best (as memory allocation is static).

Modified FCA algorithms, data-sets and complete results can be accessed at http://icfca2012.markuskirchberg.net/
FCA Algorithms Performance

- Traditional vs. Web Data
  - Conducted on an HP Cluster with Intel 8-core CPU, 2.7GHz, 16GB RAM, 16GB SWAP, and Ubuntu Linux 64bit
  - Average of 5 complete runs
  - Algorithms and data sets were matched in round-robin fashion
  - Default time-out is 3,600 seconds
FCA Algorithms Performance

<table>
<thead>
<tr>
<th>Data Sets</th>
<th>In-Close2 (CXT)</th>
<th>PCbO (FIMI)</th>
<th>PCbO -P8 (FIMI)</th>
<th>FCbO (FIMI)</th>
<th>PFCbO (FIMI)</th>
<th>PFCbO -C8 (FIMI)</th>
<th>PFCbO -L8 (CXT)</th>
<th>PFCbO -L8 (FIMI)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adult</td>
<td>1.950</td>
<td>664.830</td>
<td>207.936</td>
<td>6.485</td>
<td>1.328</td>
<td>0.394</td>
<td>0.382</td>
<td>1.795</td>
</tr>
<tr>
<td>Mushroom</td>
<td>0.697</td>
<td>96.530</td>
<td>27.404</td>
<td>1.250</td>
<td>0.376</td>
<td>0.184</td>
<td>0.173</td>
<td>0.803</td>
</tr>
<tr>
<td>DBPedia Language</td>
<td><strong>0.001</strong></td>
<td>0.002</td>
<td>0.003</td>
<td>0.003</td>
<td>0.005</td>
<td>0.004</td>
<td>0.002</td>
<td>0.002</td>
</tr>
<tr>
<td>DBPedia Drugs</td>
<td>0.024</td>
<td>0.035</td>
<td>0.033</td>
<td>0.026</td>
<td>0.023</td>
<td><strong>0.020</strong></td>
<td>0.024</td>
<td>0.017</td>
</tr>
<tr>
<td>DBPedia Drugs v2</td>
<td><strong>0.093</strong></td>
<td>1.318</td>
<td>0.575</td>
<td>0.247</td>
<td>0.209</td>
<td>0.150</td>
<td>0.159</td>
<td>0.166</td>
</tr>
<tr>
<td>DBPedia Country</td>
<td><strong>0.361</strong></td>
<td>3.597</td>
<td>2.026</td>
<td>1.489</td>
<td>10.034</td>
<td>8.146</td>
<td>6.966</td>
<td>3.284</td>
</tr>
<tr>
<td>DBPedia Alma-mater</td>
<td><strong>4.192</strong></td>
<td>45.855</td>
<td>26.675</td>
<td>24.055</td>
<td>11.144</td>
<td>5.580</td>
<td>5.484</td>
<td>6.543</td>
</tr>
<tr>
<td>DBPedia Genre</td>
<td>1.704</td>
<td>25.208</td>
<td>9.294</td>
<td>3.312</td>
<td>1.433</td>
<td><strong>1.063</strong></td>
<td>1.090</td>
<td>1.165</td>
</tr>
</tbody>
</table>

- PFCbO and In-Close2 have best overall performance
- In-Close2 seems suitable for sparse datasets
- On contrary, CbO-based algorithms seems good for dense datasets
## Input File Read Performance

<table>
<thead>
<tr>
<th>Dataset</th>
<th>In-Close2 (CXT)</th>
<th>PFCbO -C8 (FIMI)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBPedia Languages</td>
<td>0.001</td>
<td>0.000</td>
</tr>
<tr>
<td>DBPedia Drugs</td>
<td>0.004</td>
<td>0.001</td>
</tr>
<tr>
<td>DBPedia Drugs v2</td>
<td>0.017</td>
<td>0.003</td>
</tr>
<tr>
<td>DBPedia Country</td>
<td>0.035</td>
<td>0.005</td>
</tr>
<tr>
<td>UK Crime Locations</td>
<td>1.122</td>
<td>0.076</td>
</tr>
<tr>
<td>DBPedia Alma-mater</td>
<td>0.278</td>
<td>0.024</td>
</tr>
<tr>
<td>DBPedia Genre</td>
<td>0.128</td>
<td>0.017</td>
</tr>
</tbody>
</table>

- Input file reading times has no significant impact on the results
## Context Extraction and Concept Computation Performance

<table>
<thead>
<tr>
<th>Dataset</th>
<th>In-Close2 (CXT Input, Reading &amp; Execution)</th>
<th>PFCbO -C8 (FIMI Input, Reading &amp; Execution)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBPedia Languages</td>
<td>4.866</td>
<td>5.236</td>
</tr>
<tr>
<td>DBPedia Drugs</td>
<td>7.248</td>
<td>7.176</td>
</tr>
<tr>
<td>DBPedia Drugs v2</td>
<td>23.367</td>
<td>22.528</td>
</tr>
<tr>
<td>DBPedia Country</td>
<td>22.709</td>
<td>25.576</td>
</tr>
<tr>
<td>UK Crime Locations</td>
<td>221.018</td>
<td>244.008</td>
</tr>
<tr>
<td>DBPedia Alma-mater</td>
<td>87.336</td>
<td>48.896</td>
</tr>
<tr>
<td>DBPedia Genre</td>
<td>84.309</td>
<td>68.812</td>
</tr>
</tbody>
</table>
Experiments on Web-Scale Data

- Billion Triple Challenge 2011
  - Extract `rdf:type` property
  - The first 10K, 50K, 100K, 250K, 500K, and 750K objects

<table>
<thead>
<tr>
<th>Data Sets</th>
<th>10K</th>
<th>50K</th>
<th>100K</th>
<th>250K</th>
<th>500K</th>
<th>750K</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of Objects</td>
<td>10,000</td>
<td>50,000</td>
<td>100,000</td>
<td>250,000</td>
<td>500,000</td>
<td>750,000</td>
</tr>
<tr>
<td>No. of Attributes</td>
<td>1,548</td>
<td>1,548</td>
<td>1,548</td>
<td>1,552</td>
<td>1,552</td>
<td>1,552</td>
</tr>
<tr>
<td>No. of Concepts</td>
<td>220</td>
<td>449</td>
<td>601</td>
<td>885</td>
<td>1,159</td>
<td>1,416</td>
</tr>
</tbody>
</table>

- Conducted on an HP Cluster with Intel 8-core CPU, 2.7GHz, 16GB RAM, 16GB SWAP, and Ubuntu Linux 64bit
- Average of 5 complete runs
- Algorithms and data sets were matched in round-robin fashion
- Default time-out is 10,800 seconds
Experiments on Web-Scale Data
Experiments on Web-Scale Data (2)
Conclusion

• On-going efforts to apply FCA concepts and algorithms to the Semantic Web
• Web of Concepts
• Differences between properties of Web data and traditional FCA data-sets
• Performance measurements for various different types of Web data
Thank you

Any questions?

Modified FCA algorithms, data-sets and complete results can be found at [http://icfca2012.markuskirchberg.net/](http://icfca2012.markuskirchberg.net/)
Datasets

- **Dbpedia Language:**
  - Language of each country (dbpedia-owl:officialLanguage property)
  - Object: countries
  - Attribute: languages

- **Dbpedia Drug:**
  - Administration route of drugs (dbpprop:routesOfAdministration property)
  - Object: drugs
  - Attribute: administration routes

- **UK Crime Location:**
  - Location of reported crime (crime:location property)
  - Object: crime reports
  - Attribute: locations
Concept Recipe

```json
{
    "dataSource": [
        {
            "id": 0,
            "type": "SPARQL_ENDPOINT",
            "endpoint": "http://dbpedia.org/sparql",
            "sparql": "select distinct ?s ?o where { ?s
            <http://dbpedia.org/ontology/officialLanguage> ?o }
        }
    ],
    "fcaObjectSet": [
        {
            "source_id": 0,
            "binding": "?s"
        }
    ],
    "fcaAttributeSet": [
        {
            "source_id": 0,
            "binding": "?o"
        }
    ]
}
```