Smushing RDF instances: are Alice and Bob the same open source developer?

Lian Shi, Diego Berrueta, Sergio Fernández, Luis Polo and Silvino Fernández
Outline

- Introduction
- RDF instances smushing
- Data recollection
- Experimental results
- Conclusions and further directions
• **Data integration** is one of the key points in the ExpertFinder initiative. Expertise evidences have to be gathered from heterogeneous and distributed data sources.

• **Identity problem** is a complex challenge in data integration processes. There would be a widespread flooding of virtual identifiers for the same individual
  - How can I know that “resource 1” from data source A is the same as “resource 2” from data source B.
What is the identity problem

- The reference and the identity of a person is somehow hidden behind partial descriptions in:
  - Her different participation profiles in the company
  - Her different identities in some communities
  - Her different characterizations ...

- Our hypothesis is based on the idea that “Data smushing” techniques are useful for spotting redundant instances and consolidating datasets
  - We will present an experiment in the context of open source communities
Smushing is the process of normalising an RDF dataset in order to unify \textit{a priori} different RDF resources which actually represent the same thing.
A smusher is a valuable tool for identifying the co-occurrence of the same person in different online communities. We have explored two different approaches to implement our smusher:
- Logical approach, based on Inverse Functional Properties
- Heuristics approach, based on Label similarity

Pairs of redundant resources spotted by the smusher are automatically related with the property `owl:sameAs`
- They behave as a single resource for OWL-aware applications
Inverse Functional Properties

• If a property $P$ is an $\text{owl:InverseFunctionalProperty}$ (IFP), then a value $y$ can only be the value of $P$ for a single instance $x$.
  - If $a \text{ priori}$ two distinct persons Alice and Bob, such $(Alice, y)$ and $(Bob, y)$ are instances of $P$, it should be the case that Alice and Bob are the same person.

• Some FOAF properties can be used as IFPs: mbox, jabberID, openid, etc.
  - These properties are identity criteria for a person
  - We restrict our experiment to the $\text{foaf:mbox_sha1sum}$ property.
• We discarded the usage a DL reasoner to smush the instances of the dataset
  - Scalability and performance arguments (when large and OWL-Full datasets)
  - It is not allowed to define datatype properties as `owl:InverseFunctionalProperties`.

• We use light-weight rules to define IFPs
  - Rules can be customized to any kind of properties
  - Datatype properties can be treated as IFPs
  - Rules have been written as SPARQL CONSTRUCT sentences
CONSTRUCT {
   ?person1 owl:sameAs ?person2
}
WHERE {
   ?person1 rdf:type foaf:Person .
   ?person2 rdf:type foaf:Person .
   ?person1 foaf:mbox_sha1sum ?email .
   FILTER (?person1 != ?person2)
}
When smushing people's descriptions, labels refer to personal names (*foaf:name*)

- Proper names can be used to detect possible redundancies

Smushing based on label similarity is not a precise operation, i.e., two resources can share the same label with any guarantee that they are the same.

- “George W. Bush” is not the same person as his father “George Bush”
Label similarity implementation

- Previous considerations about personal names
  - Names can be miss-spelled, but the probability is very low if the names are entered by the user.
  - Traditional similarity comparison functions, such as Levenshtein distance, are not really useful

- Our approach just consider strict string equality comparison
  - The label-based smusher has been implemented as a SPARQL CONSTRUCT sentence
  - SPARQL doesn't have rich built-in string comparison functions
CONSTRUCT {
   ?person1 owl:sameAs ?person2
}
WHERE {
   ?person1 rdf:type foaf:Person .
   ?person2 rdf:type foaf:Person .
   ?person1 foaf:name ?name1 .
   FILTER (name1 = name2)
}
• A corpus of RDF data with many `foaf:Person` was assembled from five online communities
  - Two million triples were extracted
  - OpenLink Virtuoso server as back-end

• A smusher prototype was implemented, according to the previous definitions, to consolidate the dataset
  - IFP: `foaf:mbox_sha1sum`
  - String strict equality: `foaf:name`

• The results of the two smushing techniques have been compared and evaluated
Data recollection

- **Advogato Community**
  - Exports its data as FOAF files, collected using a RDF crawler

- **The RDFohloh project**
  - Exposes the information as Linked Data. Data were extracted using Ohloh's API

- **GNOME Desktop mailing lists**
  - Data exported to RDF using SWAML.

- **Debian mailing lists**
  - The information was scrapped from the HTML archives (XSLT)

- **Debian packages**
  - They were extracted from *APT* database
Instances generation

- A single URI was assigned to every generated instance
  - Different namespace for each source
- Some sources directly produce instances of `foaf:Person`
  - Advogato community, Debian Packages and the Ohloh project
- Other sources produce instances of `sioc:User`
  - Gnome and Debian mailing lists
  - Our assumption: for each `sioc:User`, there exists a `foaf:Person` (automatically created by the system when missing)
Experimental results (1)

- Number of smushed instances of *foaf:Person* using *foaf:mbox_sha1sum*

<table>
<thead>
<tr>
<th>Source</th>
<th>Debian Pkgs</th>
<th>Advogato</th>
<th>Gnome ML</th>
<th>Ohloh</th>
<th>Debian ML</th>
</tr>
</thead>
<tbody>
<tr>
<td>Debian Pkgs</td>
<td>0</td>
<td>81</td>
<td>37</td>
<td>74</td>
<td>762</td>
</tr>
<tr>
<td>Advogato</td>
<td>19</td>
<td>270</td>
<td>106</td>
<td>141</td>
<td></td>
</tr>
<tr>
<td>Gnome ML</td>
<td>364</td>
<td>112</td>
<td>161</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ohloh</td>
<td></td>
<td>0</td>
<td>115</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Debian ML</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
</tbody>
</table>
- Number of smushed instances of `foaf:Person` with exactly the same `foaf:name`

<table>
<thead>
<tr>
<th>Source</th>
<th>Debian Pkgs</th>
<th>Advogato</th>
<th>Gnome ML</th>
<th>Ohloh</th>
<th>Debian ML</th>
</tr>
</thead>
<tbody>
<tr>
<td>Debian Pkgs</td>
<td>98</td>
<td>170</td>
<td>101</td>
<td>58</td>
<td>1319</td>
</tr>
<tr>
<td>Advogato</td>
<td>49</td>
<td>592</td>
<td>95</td>
<td>305</td>
<td></td>
</tr>
<tr>
<td>Gnome ML</td>
<td>1716</td>
<td>148</td>
<td>432</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ohloh</td>
<td></td>
<td></td>
<td>13</td>
<td>208</td>
<td></td>
</tr>
<tr>
<td>Debian ML</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2909</td>
</tr>
</tbody>
</table>
• Number of smushed instances of foaf:Person combining both smushing techniques

<table>
<thead>
<tr>
<th>Source</th>
<th>Debian Pkgs</th>
<th>Advogato</th>
<th>Gnome ML</th>
<th>Ohloh</th>
<th>Debian ML</th>
</tr>
</thead>
<tbody>
<tr>
<td>Debian Pkgs</td>
<td>98</td>
<td>188</td>
<td>113</td>
<td>104</td>
<td>1418</td>
</tr>
<tr>
<td>Advogato</td>
<td>55</td>
<td>669</td>
<td>167</td>
<td>342</td>
<td></td>
</tr>
<tr>
<td>Gnome ML</td>
<td>1765</td>
<td>227</td>
<td>462</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ohloh</td>
<td>13</td>
<td>287</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Debian ML</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2909</td>
</tr>
</tbody>
</table>
## Details of the top people

<table>
<thead>
<tr>
<th>Name</th>
<th>Debian Pkgs</th>
<th>Advogato</th>
<th>Gnome ML</th>
<th>Ohloh</th>
<th>Debain ML</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frederic P.</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Dan K.</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Raphael H.</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Julien D.</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rob B.</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Daniel R.</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Federico Di G.</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ross B</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Francis T.</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Feedback from the community

• We have contacted the top people from the five online communities to recollect their feedback.

• The aim was to experimentally checked the reliability of our conclusions
  - Measure the precision and recall of the smushing process

• Some important privacy and ethical issues arose, even if the information was publicly available on the Web
Conclusions

- We have tested smushing techniques in a large dataset
  - More than 36,000 instances of foaf:Person automatically extracted have been smushed.
- The label-based smusher draws more conclusions than the IFP-based one
  - IFP results are largely contained in the results of the label-based smusher
- The differences of the results between both approaches are:
  (a) There are people who have more than one e-mail account
  (b) There are different people with the same name
Further directions

- **Enrich** the dataset with more detailed descriptions about people
  - We have only used the IFP property `foaf:mbox_sha1sum`
  - There are more personnel properties useful for smushing processes in some **controlled environments**: homepage, Ids, preferences, etc.
- **Explore and evaluate more intelligent string comparison functions**
  - Proper names can be written in different ways: “J.F.K”, “J.F. Kennedy”, “John Fitzgerald Kennedy”, etc.
  - String similarity techniques can be applied to other datatype properties. For instance, address, nicknames, etc.
thanks

questions?