Training Neural Language Models with SPARQL queries for Semi-Automatic Semantic Mapping

Giuseppe Futia*, Antonio Vetrò*, Alessio Melandri**, Juan Carlos De Martin*

* Politecnico di Torino (DAUIN) - Nexa Center for Internet & Society
** Synapta Srl
Outline

- Context
- Approach and Implementation
- Results
- Conclusion and Future Works
Context
Semantic Mapping

<table>
<thead>
<tr>
<th>people</th>
<th>birth dates</th>
<th>birth places</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alessandro Manzoni</td>
<td>1860-06-08</td>
<td>Milan</td>
</tr>
<tr>
<td>Walter Scott</td>
<td>1771-08-15</td>
<td>Edinburgh</td>
</tr>
<tr>
<td>Wolfgang Goethe</td>
<td>1749-08-28</td>
<td>Frankfurt</td>
</tr>
</tbody>
</table>
Semantic Mapping

<table>
<thead>
<tr>
<th>people</th>
<th>birth dates</th>
<th>birth places</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alessandro Manzoni</td>
<td>1860-06-08</td>
<td>Milan</td>
</tr>
<tr>
<td>Walter Scott</td>
<td>1771-08-15</td>
<td>Edinburgh</td>
</tr>
<tr>
<td>Wolfgang Goethe</td>
<td>1749-08-28</td>
<td>Frankfurt</td>
</tr>
</tbody>
</table>
Semantic Mapping (RML)
Semantic Mapping (RML)

```xml
<#Writers>
  rml:logicalSource [
    rml:source "authors.csv"
    rml:referenceFormulation ql:CSV
  ];
  rr:subjectMap [
    rr:template
    "http://ex.com/writers/{people}"
    rr:class myontology:Person
  ];
  rr:predicateObjectMap [
    rr:predicate myontology:name
    rr:objectMap [
      rml:reference "people";
    ]
  ];
  rr: predicateObjectMap [
    rr: predicate myontology:bornIn;
    rr: objectMap [
      rr: parentTriplesMap <#Place>;
    ]
  ].
</#Writers>
```
Semantic Mapping (RML)

```xml
<#Place>
  rml:logicalSource[
    rml:source "authors.csv"
    rml:referenceFormulation ql:CSV
  ];
  rr:subjectMap[
    rr:template
      "http://ex.com/cities/{birth places}"
    rr:class myontology:City
  ];
  rr:predicateObjectMap[
    rr:predicate myontology:name
    rr:objectMap[
      rml:reference "birth places"
    ]
  ].
```

Semantic Mapping (RML)

```
<#Writers>
  rml:logicalSource [
    rml:source "authors.csv"
    rml:referenceFormulation ql:CSV
  ];
  rr:subjectMap [
    rr:template
    "http://ex.com/writers/{people}"
    rr:class myontology:Person
  ];
  rr:predicateObjectMap [
    rr:predicate myontology:name
    rr:objectMap [
      rml:reference "people";
    ]
  ];
  rr: predicateObjectMap [
    rr: predicate myontology:bornIn;
    rr:objectMap [
      rr:parentTriplesMap <#Place>;
    ]
  ].
</#Writers>

<#Place>
  rml:logicalSource [
    rml:source "authors.csv"
    rml:referenceFormulation ql:CSV
  ];
  rr:subjectMap [
    rr:template
    "http://ex.com/cities/{birth places}"
    rr:class myontology:City
  ];
  rr:predicateObjectMap [
    rr:predicate myontology:name
    rr:objectMap [
      rml:reference "birth places";
    ]
  ].
</#Place>
```
Semantic Mapping Issues

• Manual generation of semantic mappings requires a significant effort and expertise

• The automation of this task is currently a challenging problem
  – Semantic Types Detection
  – Semantic Relations Discovery
Name Conflicts

- Schemas and ontologies use different conventions to name their artifacts (modeling the same domain)
  - Short identifiers vs long-speaking names
  - Plural vs singular names
  - Different tokenization techniques
Approach
A semi-automatic approach for the construction of semantic mappings based on the training of Word2Vec with SPARQL queries
Word2Vec as Language Model

• The goal of a Language Model (LM) is to learn the joint probability function of sequences of words

• Word2Vec is a LM based on a two-layered neural network to learn *word embeddings*
  - Dense and low-dimensional vectors that convey syntactic and semantic information of words
SPARQL queries as sequence of words that incorporate semantic information expressed through domain ontologies
SPARQL Queries


SPARQL Variables

\(?person\) dbp:birthPlace \(?birthPlaces\).
\(?birthPlaces\) dbp:latitude \(?lat\).
\(?birthPlaces\) dbp:longitude \(?long\).

\(?person\) dbp:birthPlace \(?bp\).
\(?bp\) dbp:latitude \(?lat\).
\(?bp\) dbp:longitude \(?long\).
SPARQL Variables


SPARQL Variables and Data Attributes

• Syntactic similarities between attributes of a data source and SPARQL query variables

• Name conflicts between traditional data sources and ontologies are mitigated:
  – Short names also in SPARQL variables
  – Plural instead of singular expressions
SPARQL Query
Variables

Data Attributes

people
birth dates
birth places
SPARQL Query

Variables

Data Attributes

- people
- birth dates
- birth places
Implementation
SPARQL Extractor

It conducts a pre-processing stage, in order to prepare a set of SPARQL queries as input of the Neural Language Model

- **SPARQL Importer**: it downloads SPARQL queries of LSQ project (AKSW Group)
- **SPARQL Parser**: it extracts triples patterns of SPARQL queries (427,186 triple patterns)
- **SPARQL Enricher**: it harmonizes the context of SPARQL queries that express the same semantics
SPARQL Enricher Details


DBpedia Class: http://dbpedia.org/ontology/person
DBpedia Label: person
SPARQL Enricher Details


Neural Language Engine

It assigns an embedding representation to variables included in triples retrieved by the SPARQL Extractor

• **Sentence Generator**: it transforms triples in sentences
• **Neural Language Model**: it exploits Word2Vec to assign embeddings to SPARQL variables
Cluster Manager

It assigns an RML template to clusters built on the embedding representation of SPARQL variables (semi-automatic)

- **Embedding Cluster**: it aggregates in clusters SPARQL variables vectors located in a close proximity (DBScan + K-means)
- **Cluster Labeler**: it provides a console through which the user can assign the RML templates to each of the cluster
<#Writers>
   rml:logicalSource [
      rml:source "authors.csv"
      rml:referenceFormulation ql:CSV
   ];
   rr:subjectMap [
      rr:template
      "http://ex.com/writers/{$P}"
      rr:class myontology:Person
   ];
   rr:predicateObjectMap [
      rr:predicate myontology:name
      rr:objectMap [
         rml:reference "{$P}"
      ]
   ].

<#Place>
   rml:logicalSource [
      rml:source "authors.csv"
      rml:referenceFormulation ql:CSV
   ];
   rr:subjectMap [
      rr:template
      "http://ex.com/cities/{$C}"
      rr:class myontology:City
   ];
   sm:variables ?birthplace, ?birthPlace, ?bp, ?birth_place;
   rr:predicateObjectMap [
      rr:predicate myontology:name
      rr:objectMap [
         rml:reference "{$C}"
      ]
   ].
Mapper Coordinator

It generates the semantic mapping between the data source and the domain ontology

• **Attributes Reconciler**: it takes as input the RML template + the data source
  – Goal: reconciling variables mentioned in the RML template and the attribute of the data source (Normalized Levenshtein)

• **Mapper Generator**: it produces the final output, that consists in a RML file
<#Writers>
  rml:logicalSource [
    rml:source "authors.csv"
    rml:referenceFormulation ql:CSV 
  ];
  rr:subjectMap [
    rr:template "http://ex.com/writers/{people}"
    rr:class myontology:Person 
  ];
  rr:predicateObjectMap [
    rr:predicate myontology:name
    rr:objectMap [
      rml:reference "people";
    ]
  ]
  sm:variables ?person, ?p, ?people 
  sm:score 0.89
];
rr: predicateObjectMap [
  rr: predicate myontology:bornIn;
  rr:objectMap [
    rr:parentTriplesMap <#Place>;
  ]
].

<#Place>
  rml:logicalSource [
    rml:source "authors.csv"
    rml:referenceFormulation ql:CSV 
  ];
  rr:subjectMap [
    rr:template "http://ex.com/cities/{birth places}" 
    rr:class myontology:City 
  ];
  sm:variables ?birthplace, ?birthPlace, ?bp, ?birth_place; 
  sm:score 0.93
  rr: predicateObjectMap [
    rr: predicate myontology:name
    rr:objectMap [
      rml:reference "birth places";
    ]
  ]
].
Results
Datasets test + Evaluation

1. The Wikipedia infobox template for a person
2. Web tables of the Famous Birthdays website
3. Web tables of the Biography.com website

<table>
<thead>
<tr>
<th></th>
<th>Famous Birthdays.com</th>
<th>Biography.com</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>NONE</td>
<td>name</td>
</tr>
<tr>
<td>birth_date</td>
<td>birthday</td>
<td>birth date</td>
</tr>
<tr>
<td>birth_place</td>
<td>birthplace</td>
<td>place of birth</td>
</tr>
<tr>
<td>death_date</td>
<td>death date</td>
<td>death date</td>
</tr>
<tr>
<td>death_place</td>
<td>NONE</td>
<td>place of death</td>
</tr>
</tbody>
</table>
Evaluation Approach + Results

\[
\text{precision} = \frac{\text{triples}(DSM) \cap \text{triples}(SSM)}{\text{triples}(DSM)}
\]

1. The Wikipedia infobox template \(\rightarrow\) Precision: 1
2. Web tables of the Famous Birthdays \(\rightarrow\) Precision: 0.3
3. Web tables of the Biography.com website \(\rightarrow\) Precision: 0.6
Discussion

1. **Wikipedia**: there is a complete overlap between the RML triples generated by the system and the domain experts

2. **FamousBirthdays.com**: the name of the subject is not directly reported in the Web table, but it is reported in another section of the Web page

3. **Biography.com**: “place of birth” and “place of death” strings are never used as variables in our SPARQL training set
Conclusions
• A semi-automatic approach for the construction of **semantic mappings** based on the training of **Word2Vec** with **SPARQL queries**

• Exploiting the potential **syntactic closeness between SPARQL variables and attributes of a data source**, we reconstruct the semantics of the data source
Future Works
• We plan to extend SPARQL variables using lexical databases like Wordnet

• We want to empower our approach using also the attribute values

• We desire to explore approaches to directly train neural networks with semantic models
Thank you!

Training Neural Language Models with SPARQL queries for Semi-Automatic Semantic Mapping

Mail: giuseppe.futia@polito.it
Twitter: giuseppe_futia
GitHub: https://github.com/giuseppefutia/