Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 00 (2018) 000–000

Procedia Computer Science

www.elsevier.com/locate/procedia

SEMANTiCS 2018 – 14th International Conference on Semantic Systems

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

Giuseppe Futia[a], Antonio Vetro'[a], Alessio Melandri[b], Juan Carlos De Martin[a]

[a]*Politecnico di Torino (DAUIN), Nexa Center for Internet and Society, Corso Duca Degli Abruzzi 24, Turin 10124, Italy*
[b]*Synapta Srl, Via S. Quintino 31, Turin 121, Italy*

Abstract

Knowledge graphs are labeled and directed multi-graphs that encode information in the form of entities and relationships. They are gaining attention in different areas of computer science: from the improvement of search engines to the development of virtual personal assistants. Currently, an open challenge in building large-scale knowledge graphs from structured data available on the Web (HTML tables, CSVs, JSONs) is the semantic integration of heterogeneous data sources. In fact, such diverse and scattered information rarely provide a formal description of metadata that is required to accomplish the integration task. In this paper we propose an approach based on neural networks to reconstruct the semantics of data sources to produce high quality knowledge graphs in terms of semantic accuracy. We developed a neural language model trained on a set of SPARQL queries performed on knowledge graphs. Through this model it is possible to semi-automatically generate a semantic map between the attributes of a data source and a domain ontology.

Keywords: Knowledge Graph; Semantic Mapping; SPARQL; Neural Language Model

1. Introduction

Knowledge graphs (KG) are labeled and directed multi-graphs, which encode information using the Resource Description Framework (RDF) [7] as data model. KGs are gaining importance in different areas of computer science,

* Corresponding author. Tel.: +39-011-090-7219 ; fax: +39-011-090-7216.
E-mail address: giuseppe.futia@polito.it

from the improvement of search engines[1] to the development of intelligent systems[2]. KGs exploit RDF to express data in the form of subject-predicate-object and domain ontologies [17] adopted on top of RDF enable to formally define semantic types, properties, and relationships of concepts.

The integration of the vast amount of scattered data available on the Web through different formats (HTML tables, CSVs, JSONs, and -more generally- information retrieved through API services) is a key ingredient to build large-scale KGs. In the Semantic Web studies, a common approach to combine information from multiple heterogeneous sources exploits domain ontologies to produce a semantic description of data sources through a process called semantic mapping (SM). Such process implies the construction of a bridge between the attributes of a specific data source -for instance the fields of a table- and semantic types and relationships described by a domain ontology.

Considering the variety of the data released on the Web, manual generation of SMs requires a significant effort and expertise and, although desirable, the automation of this task is currently a challenging problem. For these reasons, we propose a semi-automatic approach for the construction of SMs based on the training of a neural language model (NLM), i.e. Word2Vec [9], exploiting SPARQL queries performed on knowledge graphs as training set.

SPARQL queries incorporate semantic information expressed through domain ontologies. As a matter of fact, we may consider the triple "<http://dbpedia.org/resource/Alessandro_Manzoni> <http://dbpedia.org/ontology/birthPlace> ?birthPlace"[3], which allows the user to retrieve the Alessandro Manzoni's place of birth. This query is particularly suitable to become a sentence for training a NLM for two reasons: (i) articles, conjunctions, adverbs, and prepositions are not present; (ii) the vocabulary used is limited, but rich in semantics: properties like dbo:birthPlace are defined by a domain ontology.

In our paper we describe an approach that employs similar vector representation, generated by a NLM, of SPARQL variables that express the same meaning, for instance ?birthPlace, ?birthplace, ?birth_place. We label all these variables with a specific SM and we exploit the potential syntactic closeness between such variables and the attributes of a data source. Consequently, we reconstruct the semantics of the data source to produce knowledge graphs. We evaluate our approach considering the semantic accuracy metric, defined by the ISO standard 25012 [13] as the closeness of the data values to a set of values defined in a domain considered semantically correct.

The remainder of the paper is structured as follows. Section 2 provides deep insights on the problem statement. Section 3 provides references to research works developed around the automatic and semi-automatic generation of SMs. Section 4 describes the foundations of our proposed approach. Section 5 presents details about the SM generation process, followed by results and a discussion in Section 6. Finally, Section 7 provides conclusions and suggests activities for future works.

2. Problem Statement

Fig. 1 shows an example of semantic mapping of a sample data source.

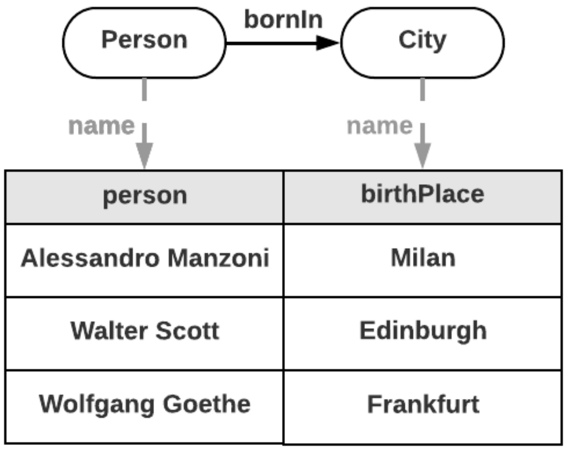
The construction of the semantic map is based on two different steps. The first step is to infer semantic types of data fields. In other words, each source attribute has to be labeled with a class and/or a data property of the domain ontology. In Fig. 1, examples of semantic types of the table attributes are the name of Person and the name of City. The second step is to identify the relationship between semantic types. In this case, in fact, we need to assume if City is the place where a person was born, or where a person lives, or even where a person died. Therefore, to build a complete semantic map we need to recognize the semantic type of the attributes of a dataset and then to establish the relationship between these semantic types.

SMs can be formalized using specific languages in compliance with the W3C recommendations, including for instance R2RML [2] and RML [3]. Both these languages define SMs according to RDF principles: R2RML maps

[1] In 2012 Google presents its knowledge graph (currently known as Knowledge Vault) to enhance its search engine's results with information gathered from a variety of sources

[2] Personal assistants like Apple Siri, Microsoft Cortana, and Amazon Alexa exploit high quality knowledge graphs in order to improve their services.

[3] Hereafter we use abbreviated versions for URIs for brevity reasons. In case of resources such as <http://dbpedia.org/resource/Alessandro_Manzoni> we use dbpedia:Alessandro_Manzoni and in case of ontology properties such as <http://dbpedia.org/ontology/birthPlace> we use dbo:birthPlace



person	birthPlace
Alessandro Manzoni	Milan
Walter Scott	Edinburgh
Wolfgang Goethe	Frankfurt

Fig. 1. Example of Semantic Mapping

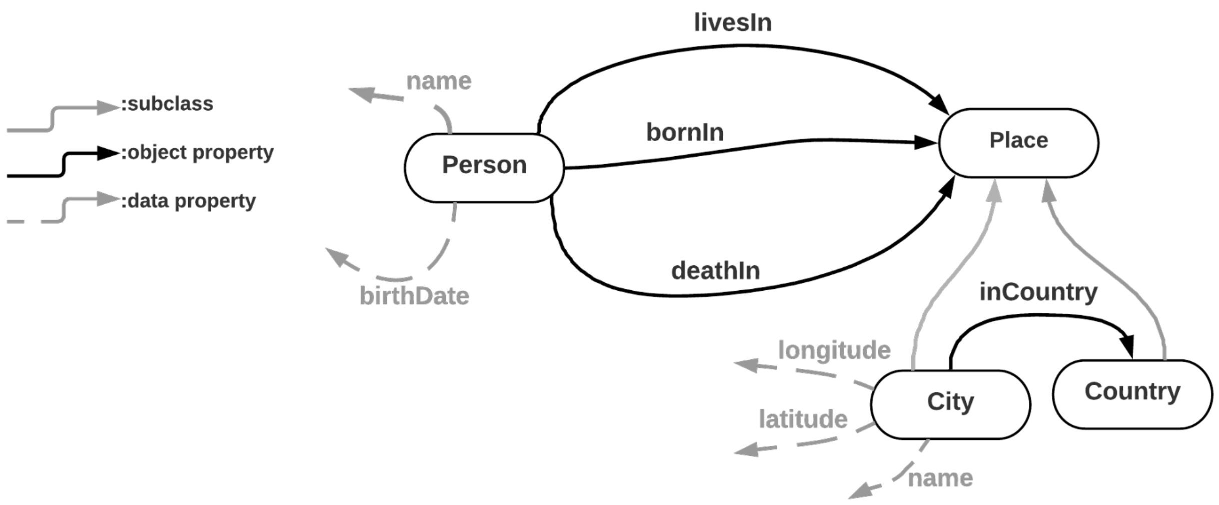
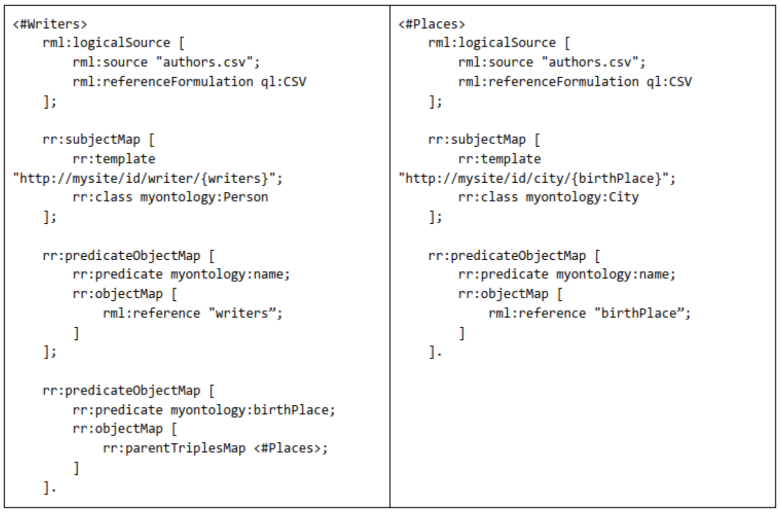


Fig. 2. Example of Domain Ontology

data of relational databases to the RDF data model, while RML extends R2RML's applicability and broaden R2RML's scope to define mappings of data released in other formats such as CSV or JSON. Assuming that the table reported in Fig. 1 is published in the CSV format with the name "authors.csv", the RML file that describes the semantic map to the domain ontology of Fig. 2 includes RDF statements reported in Fig. 3.

Automation of this task is currently a challenging problem because the system should be able to recognize the semantic types of data source fields and the semantic relationships between such fields. Furthermore, one of the main issues in traditional approaches for the generation of SMs is based on the fact that data sources schemas (for instance relational databases) and domain ontologies use different conventions to name their artifacts even when they model the same domain and thus should use a similar terminology. As underlined by Pinkel et. al [10], such differences have different causes: short identifiers for table attributes of data source and long speaking names for ontologies; plural vs. singular class types; different tokenization systems. As explained in following sections, in our approach we consider



```
<#Writers>
    rml:logicalSource [
        rml:source "authors.csv";
        rml:referenceFormulation ql:CSV
    ];

    rr:subjectMap [
        rr:template
"http://mysite/id/writer/{writers}";
        rr:class myontology:Person
    ];

    rr:predicateObjectMap [
        rr:predicate myontology:name;
        rr:objectMap [
            rml:reference "writers”;
        ]
    ];

    rr:predicateObjectMap [
        rr:predicate myontology:birthPlace;
        rr:objectMap [
            rr:parentTriplesMap <#Places>;
        ]
    ].
```

```
<#Places>
    rml:logicalSource [
        rml:source "authors.csv";
        rml:referenceFormulation ql:CSV
    ];

    rr:subjectMap [
        rr:template
"http://mysite/id/city/{birthPlace}";
        rr:class myontology:City
    ];

    rr:predicateObjectMap [
        rr:predicate myontology:name;
        rr:objectMap [
            rml:reference "birthPlace”;
        ]
    ].
```

Fig. 3. Example of RML file

instead potential syntactic similarities between attributes of a table and SPARQL query variables. The assumption is that traditional problems related to the name conflicts between traditional data sources and ontologies are mitigated.

3. Related Work

SM is a well-known problem in the field of ontology-based data integration systems [15] [16]. In fact, a number of systems to support the SM generation has been developed and, consequently, generic and effective benchmarks for their evaluation are also available [10]. According to the benchmarks, there are two different dimensions for classifying such systems: (i) availability of a domain ontology; (ii) automation level of the the tool. Regarding the first dimension, the ontology could already be described in details when the SM construction starts, or it could be incomplete. For the second dimension, tools could follow a full-automatic or a semi-automatic process. In particular, for systems that adopt a semi-automatic approach, the operation is in general iterative, because the mapping phase can be interspersed by human feedback to incorporate corrections and gradually achieve better semantic accuracy.

Among the most prominent automatic tools, we mention BootOX [5] and IncMap [11]. BootOX is based on direct mapping[4]: every table in the database is mapped into a class of the ontology, data attributes are mapped on data properties, and foreign keys to object properties. IncMap, instead, runs in two phases: firstly, it uses lexical and structural matching and, secondly, it represents with a meta-graph the ontology and the schema of the input dataset in order to preserve their structure. In the field of more general purpose tools, we remark MIRROR [8] and D2RQ [1]. Both tools do not necessarily exploit an existing domain ontology, but they can generate an ontology on-the-fly based on the input data schema. In details, MIRROR produces R2RML direct mappings exploiting the morph-RDB[5] engine. D2RQ, instead, uses its own native language to define the SMs.

In the field of semi-automatic tools, there is a recent proliferation of tools that exploit all the background knowledge of RDF data, its production and its queries. Heyvaert et. al [4], for instance, propose an ontology-based data mapping generation method that uses extended knowledge from existing RDF data, schemas, query workload, and already-built SMs, and combines it with knowledge provided by the mapping process to generate new SMs. Other approaches [6], instead, focus only on the reuse of already-built SMs in a specific domain in order to learn the semantics behind data sources.

[4] For more details, see: âĂIJA Direct Mapping of Relational Data to RDFâĂİ, W3C Recommendation 27 September 2012. More information available at: https://www.w3.org/TR/rdb-direct-mapping/.
[5] The GitHub repository of the engine is available at: https://github.com/oeg-upm/morph-rdb

Our system adopts a semi-automatic approach: it receives as input a domain ontology for the SM generation and suggests to the user several possible SM schemes. Although the approaches that exploit already-built SMs are very promising, the current problem is that high-quality (in terms of semantic accuracy) SMs are complex to understand and reuse, and in many cases are not publicly available, because they are developed by private organizations for their own purposes. For these reasons, we decide to use SPARQL queries instead of existing SMs on a specific domain. The SPARQL queries, in fact, offer a variety of advantages with respect to SMs: they are easier to understand by users with basic skills and they are recently made available by research projects like LSQ, the Linked data Sparql Queries dataset[6], maintained by the AKSW Research Group at the University of Leipzig. Furthermore, at the best of our knowledge, there are no systems that exploit a NLM like Word2Vec that are able to generate valuable SMs exploiting the vector representation of SPARQL variables in triple patterns.

4. Approach

SPARQL (a recursive acronym for SPARQL Protocol and RDF Query Language) is a semantic query language that allows users to retrieve and manipulate data stored in the RDF format. Our proposal is to harness SPARQL queries to train a NLM that reconstruct the semantics of a data source.

This section is structured as follows: (i) we provide details on NLM principles and we describe the key ideas behind a specific implementation called Word2Vec; (ii) we explain how we use Word2Vec to assign a vector representation to SPARQL variables; (iii) we clarify the potential behind syntactic similarities between SPARQL query variables and attributes of a data source, in order to reconstruct the semantics of the latter.

4.1. Word2Vec as Neural Language Model

The goal of a Language Model (LM) is to learn the joint probability function of sequences of words in a specific language. LMs are becoming a fundamental part of many systems that attempt to solve natural language processing tasks, such as machine translation and speech recognition. Currently, LMs developed using neural networks also known as Neural Language Models define the state of the art in terms of learning efficiency and accuracy [12].

The Word2Vec Neural Language Model is an approach based on a two-layered neural network to learn *word embeddings*, i.e. dense and low-dimensional vectors that convey syntactic and semantic information. Word2Vec is characterized by two different models, the Continuous Bag-of-Words model (CBOW) and the Skip-Gram model (Fig. 4).

Although it is not the focus of this paper to provide the reader with an explanation of how neural networks for language modeling work, below we briefly summarize the working mechanisms of CBOW and Skip-Gram models.

The goal of CBOW is to predict a word given its context, which is defined as the window of words to the left and to the right of a target word in a sentence. In the CBOW, the neural network has the following behaviour:

1. The input layer is characterized by the surrounding words of the target, whose embedding representations are retrieved from the input weight matrix and projected in the projection layer.
2. Then, using the output weight matrix between the projection layer and the output layer, a score for each word in the vocabulary is computed, which is the probability of the word being a target word.

The goal of Skip-Gram is inverse of CBOW, because it tries to predict the context words from the target word:

1. The input layer is constituted by the target word, whose embedding representation is retrieved from the input weight matrix and projected in the projection layer.
2. Then, using the output weight matrix between the projection layer and the output layer characterized by the surrounding words of the target, a score for each word in the vocabulary is computed, which is the probability of the word being a context word.

[6] More details on the available datasets and the SPARQL storage model is available at: `http://aksw.github.io/LSQ/`.

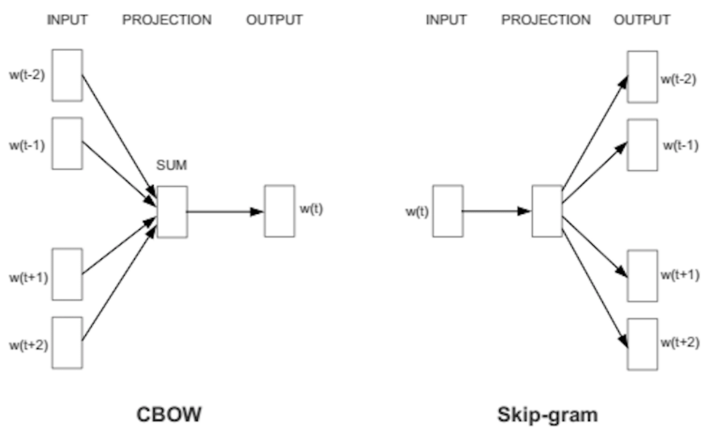


Fig. 4. Architecture of Word2Vec models: CBOW and Skip-Gram

4.2. Embedding Representation of SPARQL Variables

Triple patterns mentioned in SPARQL queries like "?person dbo:birthPlace ?birthPlace" are assertions characterized by a subject, a predicate, and an object. For such reason, they can be considered as natural language sentences, composed by three words, corresponding to the elements of the triples.

Furthermore, variables mentioned in such triples, for instance ?birthPlace, have two peculiar features: (i) they contain a well defined semantic type (a place in our case); (ii) they have specific relationships with other entities in the KG (someone's birthplace in our case).

To better understand such peculiarities, consider the 2 following examples of queries:

```
?person dbo:birthPlace ?birthPlace .
?birthPlace dbp:latitude ?lat .
?birthPlace dbp:longitude ?long .

?person dbo:birthPlace ?bp .
?bp dbp:longitude ?lat .
?bp dbp:longitude ?long .
```

In this case the variables ?birthPlace and ?bp are characterized by a similar context: they are preceded by either dbo:birthPlace and ?person, while they are followed by dbp:latitude and ?lat.

As explained by Ristoski and Paulheim [14], the goal of a NLM like Word2Vec is "to estimate the likelihood of a specific sequence of words appearing in a corpus, explicitly modeling the assumption that closer words in the word sequence are statistically more dependent". Therefore, NLMs are able to generate embedding representations of words that express semantic similarities. According to these principles, in our specific case variables like ?birthPlace and ?bp are characterized by a similar vector representation.

4.3. SPARQL Variables and Data Attributes

In our approach we consider potential syntactic similarities between attributes of a data source and SPARQL query variables. The assumption is that traditional problems related to the name conflicts between traditional data sources and ontologies are mitigated. For instance, users tend to write short names also in SPARQL variables than long speaking names. In other cases, users tend to use plural instead of singular expressions, where more than a single result is expected: in a SPARQL query that retrieves all actors of a specific movie, users are inclined to use ?actors instead of ?actor.

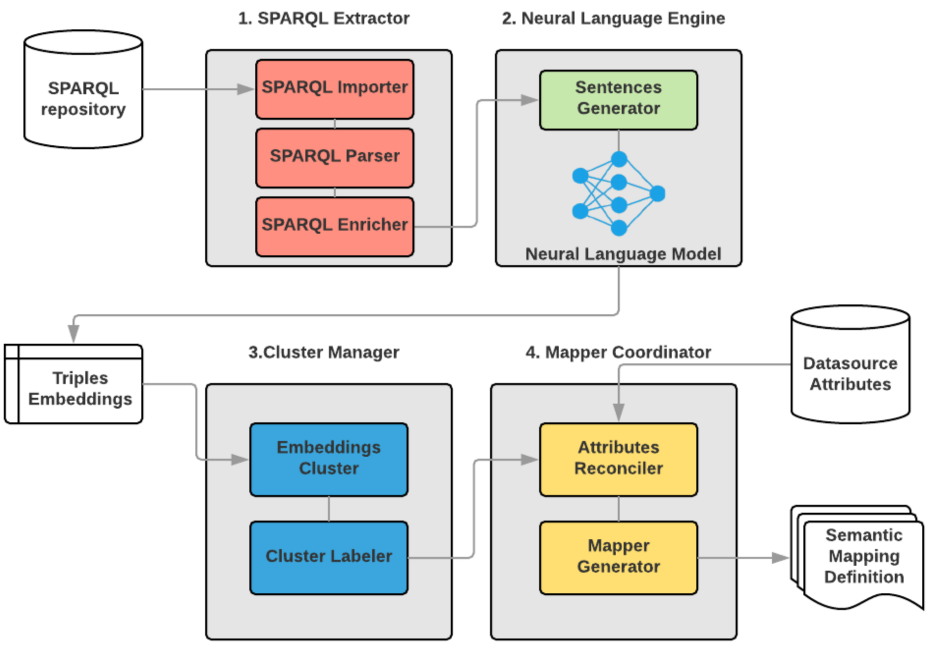


Fig. 5. Modules and components of the pipeline for the generation of the semantic model

In the following Section we explain how the embedding representation of SPARQL queries and the syntactic similarities between such variables and the attributes of a data source can be exploited to produce SMs in a semi-automatic way.

5. Semantic Mapping Generation Process and Implementation

In this Section we illustrate all stages to generate a SM, exploiting SPARQL queries as training sentences for a NLM. Each stage is performed by different software modules that are shown in Fig. 5

5.1. SPARQL Extractor

The SPARQL Extractor module (1 in Fig. 5) conducts a pre-processing stage in order to prepare a set of SPARQL queries as input of the NLM. To accomplish such goal, it uses three software components: the SPARQL Importer, the SPARQL Parser, and the SPARQL Enricher.

The SPARQL Importer downloads SPARQL queries made available by the LSQ project. Such SPARQL queries are published according to a specific ontology and can be retrieved from an endpoint available on the Web[7]. The download process is performed through a pipeline of 3 different queries:

1. we retrieve all properties mentioned in the body of stored queries. The result of the query provides the URIs of properties (e.g., dbp:birthPlace);

[7] The LSQ SPARQL endpoint is available at: `http://lsq.aksw.org/sparql`.

2. we retrieve all URIs of queries data that contain such properties. For instance, the URI http://lsq.aksw.org/res/DBpedia-q135894 describes a query that contains the property dbp:birthPlace in the WHERE and in the OPTION clauses[8];
3. we retrieve the body of each query identified by a URI. Considering http://lsq.aksw.org/res/DBpedia-q135894, we are able to obtain the entire text of the query, exploiting the property http://spinrdf.org/sp#text defined by the LSQ ontology.

The SPARQL Parser component extracts triples from text of queries retrieved by the SPARQL component. Such triples are located inside the WHERE and the OPTION clauses, in which you can find for instance "dbpedia:Alessandro_Manzoni dbo:birthPlace ?birthPlace". Triple patterns like this compose the training set for the NLM (see Section 4.2). The SPARQL Parser component is developed by means of SPARQL.js library[9].

Finally, the goal of the SPARQL Enricher is to harmonize the context of SPARQL variables that express the same semantics. To clarify the behaviour of this component, please consider the following triple examples:

```
dbpedia:Alessandro_Manzoni dbo:birthPlace ?birthPlace .

dbpedia:Walter_Scott dbo:birthPlace ?birthPlace .

?person dbo:birthPlace ?birthplace .
```

In this case, the context of ?birthPlace and ?birthplace are not very similar, because the first one is included in sentences with different subjects: dbpedia:Alessandro_Manzoni, dbpedia:Walter_Scott. The third one, instead, is included in a sentence where ?person is the subject. Nevertheless, Alessandro Manzoni and Walter Scott are all concepts categorized under the DBpedia class Person (http://dbpedia.org/ontology/person).

The SPARQL Enricher component retrieves the label of the highest-level classes of the concepts mentioned in the SPARQL queries (in this case the label of the class http://dbpedia.org/ontology/person is "person") and adds a new triple for each concept. In our case, 2 new triples in the form ?person dbo:birthPlace ?birthPlace, are added to the training set.

5.2. *Neural Language Engine*

The goal of the Neural Language Engine module (2 in Fig. 5) is to assign an embedding representation to variables included in triples retrieved by the SPARQL Extractor. To perform this task, the module uses two software components: the Sentence Generator and the Neural Language Model.

The Sentence Generator component takes as input SPARQL triples and transform them in sentences to train the NLM. In details, the triple "dbpedia:Alessandro_Manzoni dbo:birthPlace ?birthPlace", for instance, is treated as a sentence made of different words: (i) dbpedia:Alessandro_Manzoni; (ii) dbo:birthPlace; (iii) ?birthPlace.

The NLM component generates the vector representation of the SPARQL variables, taking as input the sentences produced by the Sentence Generator. In our context, variables like ?birthplace and ?bp share a similar context and therefore they can be aggregate in the same cluster as it is described in the following stage.

5.3. *Cluster Manager*

The objective of the Cluster Manager module (3 in Fig. 5) is to assign a RML template[10] to clusters built on the embedding representation of SPARQL variables. Unlike previous software modules that work fully automatically, the Cluster Manager needs user intervention to correct the results of the clustering process and the definition of the RML template. To achieve its goal, such module exploits two software components: the Embedding Cluster and the Cluster Labeler.

[8] Including triples mentioned in the OPTION clause, we are able to extend the training set.
[9] The GitHub repository of the tool is available at `https://github.com/RubenVerborgh/SPARQL.js/`
[10] An RML template has essentially the same contents of the RML file reported in Section 2, except that some parts of the code are replaced by parameters whose content is filled by the Cluster Manager itself and by the Mapper Coordinator module described in the next Section

The Embedding Cluster component aggregates vectors representing SPARQL variables located in a close proximity, according to the cosine similarity[11]. The algorithm used for the clustering is a combination between DBScan and K-means. At this stage, the user can adjust through a GUI the results of the clustering process in case of a wrong variables grouping.

The Cluster Labeler component provides to the user another GUI through which he can assign the RML templates according to the clusters generated by the previous component.

We provide an example of how the Cluster Manager module works. Consider the following example of different clusters of SPARQL variables created by the Embedding Cluster component:

- Cluster 1: ?person, ?p, ?people.
- Cluster 2: ?birthplace, ?birthPlace, ?bp, ?birth_place.

On the basis of these clusters, the RML template is composed by 3 elements (see Fig. 6):

- #[1] (in blue in Fig. 6) assigns the semantic type dbpedia:Person to SPARQL variables ?person, ?p, ?people.
- #[2] (in red in Fig. 6) assigns the semantic type dbpedia:Place to SPARQL variables ?birthplace, ?birthPlace, ?bp, ?birth_place.
- #[3] (in green in Fig. 6) assigns the relation dbo:birthPlace between the just mentioned semantic types.

SPARQL variables are included in the RML template and they are directly linked with the reference element (in bold in Fig. 6). The next module adds further information to the RML template that finally contribute to reconstruct the semantics of a data source.

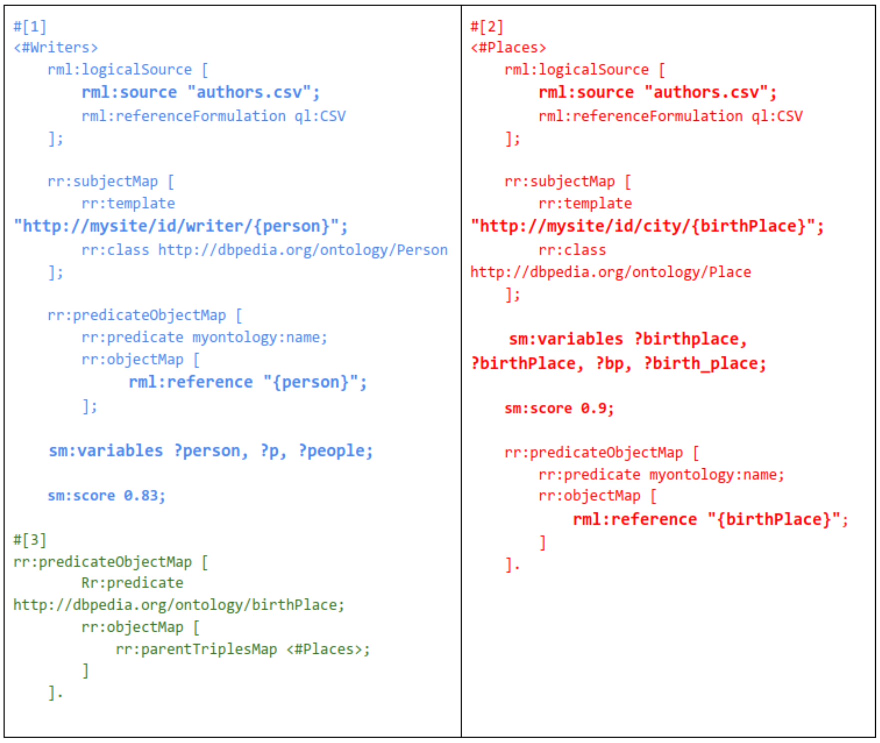
5.4. Mapper Coordinator

The Mapper Coordinator module generates the SM (in the form of RML template) between the data source and the domain ontology. To reach this result, the module makes use of 2 different software components: the Attributes Reconciler and the Mapper Generator.

The Attributes Reconciler takes two inputs: (i) the RML template generated by the Cluster Manager module; (ii) the data source for which a domain ontology-base semantic reconstruction is needed. The goal of such component is to reconcile the variables mentioned in the RML template and the attributes of the data source, considering syntactic similarities. To map the single attribute to the correct cluster of variables we compute a score, comparing the median value of the normalized Levenshtein distances between the attribute and each element of the cluster. According to the highest value of this score, the component proceeds with the assignment.

To clarify this step, consider the table mentioned in Section 2 with the two columns "person" and "birthPlace" as a CSV file entitled "authors.csv". The Attribute Reconciler component compares the word "person" with all the variables mentioned in the RML templates: in our example case, it is compared with the Cluster 1 (?person, ?p, ?people) and the Cluster 2 (?birthplace, ?birthPlace, ?bp, ?birth_place) defined in the previous Section. The attribute author is assigned to the first element of the RML template (#[1] (in blue in Fig.) with a specific score (see sm:score value in Fig. 6). The same process is executed for the "birth_place" attribute, that is assigned to the second element of the RML template (#[2] (in blue in Fig. 6) with a specific score (see sm:score value in Fig. 6).

[11] Cosine similarity is a measure of similarity between two non-zero vectors of an inner product space that measures the cosine of the angle between them.



```
#[1]
<#Writers>
    rml:logicalSource [
        rml:source "authors.csv";
        rml:referenceFormulation ql:CSV
    ];

    rr:subjectMap [
        rr:template
"http://mysite/id/writer/{person}";
        rr:class http://dbpedia.org/ontology/Person
    ];

    rr:predicateObjectMap [
        rr:predicate myontology:name;
        rr:objectMap [
             rml:reference "{person}";
        ];

    sm:variables ?person, ?p, ?people;

    sm:score 0.83;

#[3]
rr:predicateObjectMap [
        Rr:predicate
http://dbpedia.org/ontology/birthPlace;
        rr:objectMap [
            rr:parentTriplesMap <#Places>;
        ]
    ].
```

```
#[2]
<#Places>
    rml:logicalSource [
        rml:source "authors.csv";
        rml:referenceFormulation ql:CSV
    ];

    rr:subjectMap [
        rr:template
"http://mysite/id/city/{birthPlace}";
        rr:class
http://dbpedia.org/ontology/Place
    ];

    sm:variables ?birthplace,
?birthPlace, ?bp, ?birth_place;

    sm:score 0.9;

    rr:predicateObjectMap [
        rr:predicate myontology:name;
        rr:objectMap [
            rml:reference "{birthPlace}";
        ]
    ].
```

Fig. 6. Example of RML template

6. Evaluation

6.1. Design

The goal of our experiment is to measure the semantic accuracy of the SMs generated with our approach[12]. As mentioned in previous Sections, the semantic accuracy can be defined as the closeness of the data values to a set of values defined in a domain considered semantically correct. For our purposes, we therefore compare the SM generated by our system with a SM produced by a domain expert on a specific field of knowledge. Assuming that the RML file of the SM produced by the domain expert is identified by DSM and the RML file of the SM generated by the system is identified by SSM, we compute the precision considering the intersection between the triples mentioned in the two RML files and the number of triples in the RML file created by the domain expert.

$$precision = \frac{triples(DSM) \bigcap triples(HSM)}{triples(DSM)}$$

The precision value is between 0 and 1. For the evaluation of the semantic accuracy, we consider only the precision and not the recall metric, because domain experts can add much more RML triples to define the SM than a semi-automatic system, even if these are not specified in the original data source. For instance, they can include triples related to the description of an entity, using the property http://dbpedia.org/ontology/description of the DBpe-

[12] The source code for the evaluation is available at https://github.com/giuseppefutia/semantics2018

dia ontology. Nevertheless, this new assertion includes additional information which is not strictly related to semantic accuracy, even though it may be useful to the user to better understand the meaning of a specific resource.

To reach their goal, human experts and our system share some details in order to avoid differences in RML triples that are not strictly related to the semantic mapping process. In particular, (i) they share knowledge about the ontology as starting point to create the SM, (ii) they share the root of the URI in order to create resources of concepts mentioned in the data source. To clarify this second point, assume that for identified entities in the data source: both the domain expert and the system have to use the URI http://mydomain/entities/_NAME_, where _NAME_ is the value of the attribute of the data source. In this way, we avoid to create differences in terms of RML triples not related to the semantic mapping process, that can lead to a decrease in the value of precision.

6.2. *Data Sources*

To build the training set, we exploit SPARQL queries published by the LSQ project. Such project provides SPARQL queries performed on endpoints of different research projects: DBpedia[13], Linked Geo Data[14], Semantic Web Dog Food[15], British Museum[16]. From SPARQL queries retrieved from LSQ, we were able to extract 427.186 triples. Each of this triple constitutes a training sample for the Word2Vec model.

For the SM generation task we decided to use only the DBpedia ontology[17], whose properties are the most used in the SPARQL queries.

To show the potential of our approach, we test the mapping process with 3 different data sources, that cover the same data using different attributes:

1. The Wikipedia infobox template for a person[18].
2. Web tables of the Famous Birthdays website[19].
3. Web tables of the Biography.com website[20].

All these data sources include the following data attributes:

Table 1. Attributes of data sources reported in Wikipedia, Famous Birthdays.com, Biography.com

Wikipedia	Famous Birthdays.com	Biography.com
name	NONE	name
birth_date	birthday	birth date
birth_place	birthplace	place of birth
death_date	death date	death date
death_place	NONE	place of death

6.3. *Results and Discussion*

We report the precision value of the semi-automatic mapping generation process on the three specific cases:

- Wikipedia: 1
- Famous Birthday.com: 0.3

[13] Project website: http://dbpedia.org
[14] Project website: http://linkedgeodata.org
[15] Project website: http://data.semanticweb.org
[16] Project website: http://bm.rkbexplorer.com
[17] More information on the DBpedia ontology is available at: http://wiki.dbpedia.org/services-resources/ontology.
[18] More information at: https://en.wikipedia.org/wiki/Template:Infobox_person
[19] More information at: https://www.famousbirthdays.com
[20] More information at: https://www.biography.com

- Biography.com: 0.6

In the case of Wikipedia, there is complete overlap between the RML triples generated by the system and by the domain expert. The issue with FamousBirthdays.com is that the name of the subject entity is not directly reported in the Web table, but it is reported in another section of the Web page. In the case of Biography.com, the low of precision is related to the strings "place of birth" and "place of death", because they are never used as variables in SPARQL queries and the module that exploits the normalized Levenshtein distance was not able to recognize the semantic affinity with other expressions like "birth_place" and "death_place" syntactically very distant.

7. Conclusions and Future Works

In our paper we have described an approach that employs similar vector representations of SPARQL variables that express the same meaning, for instance ?birthPlace, ?birthplace, ?birth_place. Such vectors are generated exploiting a well-known neural language model like Word2Vec. We have grouped variables that express same semantics with clustering algorithms and we have labeled each cluster with a specific semantic mapping. Exploiting the potential syntactic closeness between SPARQL variables and attributes of a datasource, we have reconstructed the semantics of the data source. We have evaluated our approach considering the semantic accuracy metric, comparing the semantic mapping generated by our system with a semantic mapping produced by a domain expert, computing the precision value. In our future works we plan to extend SPARQL variable using lexical databases like Wordnet and to combine our approach with systems that also use the values of the attributes for the semantic mapping.

References

[1] Bizer, C., Seaborne, A., 2004. D2rq-treating non-rdf databases as virtual rdf graphs, in: Proceedings of the 3rd international semantic web conference (ISWC2004), Proceedings of ISWC2004.

[2] Das, S., 2011. R2rml: Rdb to rdf mapping language. http://www. w3. org/TR/r2rml/ .

[3] Dimou, A., Vander Sande, M., Colpaert, P., Verborgh, R., Mannens, E., Van de Walle, R., 2014. Rml: A generic language for integrated rdf mappings of heterogeneous data., in: LDOW.

[4] Heyvaert, P., Dimou, A., Verborgh, R., Mannens, E., 2017. Ontology-based data access mapping generation using data, schema, query, and mapping knowledge, in: European Semantic Web Conference, Springer. pp. 205–215.

[5] Jiménez-Ruiz, E., Kharlamov, E., Zheleznyakov, D., Horrocks, I., Pinkel, C., Skjæveland, M.G., Thorstensen, E., Mora, J., 2015. Bootox: Practical mapping of rdbs to owl 2, in: International Semantic Web Conference, Springer. pp. 113–132.

[6] Knoblock, C.A., Szekely, P., Ambite, J.L., Goel, A., Gupta, S., Lerman, K., Muslea, M., Taheriyan, M., Mallick, P., 2012. Semi-automatically mapping structured sources into the semantic web, in: Extended Semantic Web Conference, Springer. pp. 375–390.

[7] Lassila, O., Swick, R.R., 1999. Resource description framework (rdf) model and syntax specification. w3c recommendation, 1999.

[8] de Medeiros, L.F., Priyatna, F., Corcho, O., 2015. Mirror: Automatic r2rml mapping generation from relational databases, in: International Conference on Web Engineering, Springer. pp. 326–343.

[9] Mikolov, T., Chen, K., Corrado, G., Dean, J., 2013. Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781 .

[10] Pinkel, C., Binnig, C., Jiménez-Ruiz, E., Kharlamov, E., May, W., Nikolov, A., Sasa Bastinos, A., Skjæveland, M.G., Solimando, A., Taheriyan, M., et al., 2016. Rodi: Benchmarking relational-to-ontology mapping generation quality. Semantic Web , 1–28.

[11] Pinkel, C., Binnig, C., Kharlamov, E., Haase, P., 2013. Incmap: pay as you go matching of relational schemata to owl ontologies., in: OM, pp. 37–48.

[12] Press, O., Wolf, L., 2016. Using the output embedding to improve language models. arXiv preprint arXiv:1608.05859 .

[13] Rafique, I., Lew, P., Abbasi, M.Q., Li, Z., 2012. Information quality evaluation framework: Extending iso 25012 data quality model. World Academy of Science, Engineering and Technology 65, 523–528.

[14] Ristoski, P., Paulheim, H., 2016. Rdf2vec: Rdf graph embeddings for data mining, in: International Semantic Web Conference, Springer. pp. 498–514.

[15] Sahoo, S.S., Halb, W., Hellmann, S., Idehen, K., Thibodeau Jr, T., Auer, S., Sequeda, J., Ezzat, A., 2009. A survey of current approaches for mapping of relational databases to rdf. W3C RDB2RDF Incubator Group Report 1, 113–130.

[16] Spanos, D.E., Stavrou, P., Mitrou, N., 2012. Bringing relational databases into the semantic web: A survey. Semantic Web 3, 169–209.

[17] Wache, H., Voegele, T., Visser, U., Stuckenschmidt, H., Schuster, G., Neumann, H., Hübner, S., 2001. Ontology-based integration of information-a survey of existing approaches, in: IJCAI-01 workshop: ontologies and information sharing, Citeseer. pp. 108–117.