SeMi: A SEmantic Modeling machIne to build Knowledge Graphs with graph neural networks

SoftwareX 12 (2020) 100516

Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx

Original software publication

SeMi: A SEmantic Modeling machIne to build Knowledge Graphs with graph neural networks

Giuseppe Futia *, Antonio Vetrò, Juan Carlos De Martin

Nexa Center for Internet and Society (DAUIN), Politecnico di Torino, Corso Duca Degli Abruzzi, 24, Torino, Italy

ARTICLE INFO

Article history:
Received 12 August 2019
Received in revised form 27 March 2020
Accepted 19 May 2020

Keywords:
Knowledge Graphs
Semantic Modeling
Graph neural networks

ABSTRACT

SeMi (SEmantic Modeling machIne) is a tool to semi-automatically build large-scale Knowledge Graphs from structured sources such as CSV, JSON, and XML files. To achieve such a goal, SeMi builds the semantic models of the data sources, in terms of concepts and relations within a domain ontology. Most of the research contributions on automatic semantic modeling is focused on the detection of semantic types of source attributes. However, the inference of the correct semantic relations between these attributes is critical to reconstruct the precise meaning of the data. SeMi covers the entire process of semantic modeling: (i) it provides a semi-automatic step to detect semantic types; (ii) it exploits a novel approach to inference semantic relations, based on a graph neural network trained on background linked data. At the best of our knowledge, this is the first technique that exploits a graph neural network to support the semantic modeling process. Furthermore, the pipeline implemented in SeMi is modular and each component can be replaced to tailor the process to very specific domains or requirements. This contribution can be considered as a step ahead towards automatic and scalable approaches for building Knowledge Graphs.

Code metadata

Current code version	v0.1
Permanent link to code/repository used for this code version	https://github.com/ElsevierSoftwareX/SOFTX_2019_258
Legal Code License	GNU General Public License v3.0
Code versioning system used	git
Software code languages, tools, and services used	Node.js, Python, PyTorch, Elasticsearch.
Compilation requirements, operating environments & dependencies	Linux Operating System
If available Link to developer documentation/manual	https://github.com/giuseppefutia/semi/blob/master/README.md/
Support email for questions	giuseppe.futia@polito.it

1. Introduction

Knowledge Graphs (KGs) encode relations between real-world facts through nodes and edges associated to semantic entities. For example, the statement "Mona Lisa is created by Leonardo da Vinci" , is represented by the nodes "Mona Lisa" and "Leonardo da Vinci" connected by the edge "is created by". The KG is modeled through the Resource Description Framework (RDF) [1], that shapes data in the form of subject (Mona Lisa), predicate (is created by), and object (Leonardo da Vinci). Therefore, *subject* and *object* are the nodes in the graph, while the *predicate* is the edge.

Because of their flexibility, KGs are currently employed in several fields, including semantic search, fraud detection, chatbots development, drug discovery, and risk analysis (e.g., [2–4]). However, the construction of KGs is not yet a scalable process, especially because of low automation of the semantic modeling step. In fact, there is often lack of the metadata that link the attributes of an input data source – for instance the fields of a CSV file – to the classes and properties defined by an ontology. As a consequence, semantic models needed for building KGs are often created manually. This is a notable problem for building large-scale KG, since a manual approach requires a high effort and several domain experts, due to the magnitude and the variety of data available on the Web.

We developed SeMi (SEmantic Modeling MachIne) to address this issue, providing a modular pipeline to semi-automatically

* Corresponding author.
E-mail address: giuseppe.futia@polito.it (G. Futia).

https://doi.org/10.1016/j.softx.2020.100516

reconstruct the semantic model of a data source. The semantic model is formalized as a graph where leaf nodes are the attributes of the data source and the other nodes and edges are derived from the ontology (see Fig. 1). The semantic model generation relies on the following steps:

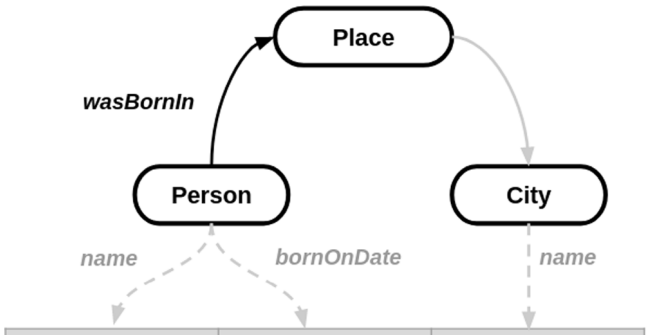
1. The *semantic type detection*, that assigns a class and a property of the ontology (*semantic label*) to each attribute of the target source. In SeMi this is a semi-automatic step, because it suggests a ranking of semantic labels for each attribute of the source, to be validated by the user or by the domain expert. The implemented approach is based on the work of Ramnandan et al. [5].
2. The *semantic relation inference*, that establishes the correct semantic relations between the target source attributes. In SeMi this step is fully-automatic and characterizes the novelty of our approach, based on a graph neural network that is trained on background linked data.

Our approach takes inspiration from the work of Taheriyan et al. [6]. In this article the authors describe a method that exploits linked data as background knowledge to infer semantic relations within a data source. They perform a manual extraction of features from linked data with SPARQL [7] queries. These features include various types of complex graph patterns, that represent semantic relations of different lengths. The extraction of these patterns requires a compound feature engineering process, it is not scalable as the length of the semantic relation increases, and it requires prior knowledge of the linked data structure. On the contrary, our method to extract features from linked data is automatic. Our proposed approach is based on a graph neural network that automatically learns latent features from the local neighborhood structures of the linked data graph. These latent features are aggregated into a vector representation of entities and properties (*embeddings*), which are employed to predict the semantic relations within the target source. In this paper, we demonstrate that the adoption of the embeddings increases the accuracy in reconstructing the correct relations in complex data sources, compared to manually-selected and compound features. Furthermore, we show that the graph neural network training is a more scalable procedure than the extraction of increasingly complex graph patterns through SPARQL queries.

The remainder of this paper includes the following sections. Section 2 presents an overview of the research into automatic semantic modeling, underlying our contribution in the field. Section 3 provides details on the system goal of SeMi and its main architectural requirements. Section 4 introduces the pipeline components for the semantic model generation, whose implementation details are reported in Section 5. Details on the evaluation method the and results obtained by SeMi are described in Section 6. Section 7 shows the package components and practical uses of the tool. In Section 8 we summarize our contribution and give notice of on-going applications of SeMi and future planned developments.

2. Related work

The semantic modeling process involves two main tasks: (i) the semantic type detection (or semantic labeling) and (ii) the semantic relation inference. Among the most recent contributions in semantic labeling, we mention the work of Pomp et al. [8], in which the authors propose a semantic concept recommendation system for data attributes based on the "data representatives". Data representatives define multiple representations of information that are generated from the data values, rather than from the labels of the source attributes. On the same topic, Ruemmele et al. [9] developed three different systems for the detection



people	birth dates	birth places
Bob Dylan	1941-05-24	Duluth
Elvis Presley	1935-01-08	Tupelo
Little Richard	1932-12-05	Macon

Fig. 1. Example of semantic model of a tabular data source.

of semantic types: a classification model based on a manual feature engineering process and two deep learning models that exploit the Convolutional Neural Network and the Multi-Layer Perceptron architecture respectively. Ramnandan et al. [5] adopt a different perspective for semantic labeling, that consider an holistic view of the data values corresponding to a semantic label. The goal of this approach is to capture the features of data instances that are related to a semantic type as a whole. Their classifier assigns a semantic label to each attribute of the target data source: the algorithm predicts candidate semantic types by computing the cosine similarity[1] between the TF–IDF[2] vectors of the labeled values in the training data, and the unlabeled values coming from an attribute of a new target source.

In SeMi we implement the method proposed by Ramnandan et al. [5]. Their approach has the following advantages: (i) *efficiency and scalability*: their method is about 250 times faster than methods that use other algorithms such Conditional Random Fields; (ii) *accuracy in different fields*: their approach improves accuracy of competing methods on a plethora of diverse sources; (iii) *generality*: the method is agnostic in terms of ontology and schema for the semantic labeling purpose. At the end of the semantic labeling process, SeMi suggests a ranking of semantic labels for each attribute of the source, to be validated by the user or by the domain expert.

In their influential works [6,10,11] Taheriyan et al. indicates that research efforts in semantic modeling focused so far mainly on the detection of semantic types (or semantic labeling), while less attention has been given to the automatic inference of semantic relations. The motivation for this observed trend has to be found in the complexity of the second step: in fact, even when semantic labels are properly refined with human intervention, as expected in the functioning of SeMi, inferring the relations through an automatic mechanism is not trivial and it is still an open issue in research. In addition, in more complex (but not unusual) situations, semantic types can be connected through multiple paths that include different sequences of ontology classes and object properties. As a consequence, without

[1] Cosine similarity is a measure of similarity between two vectors, obtained computing the cosine of the angle between them.

[2] The term frequency–inverse document frequency (TF–IDF) reflects how important a word is to a document from a collection or corpus.

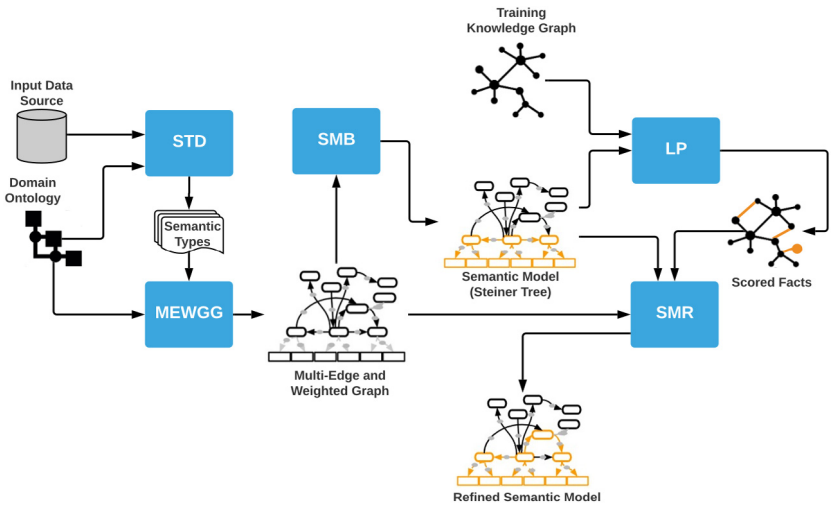


Fig. 2. Pipeline components of SeMi.

explicit and additional background context, it is difficult to identify which paths – or in other words which semantic relations – define the actual meaning of the data.

Recent works that cover the entire semantic modeling process, including the semantic relation inference, exploit as background knowledge existing semantic models for similar sources, to learn the semantics of the target source. In this context, the work of Vu et al. [12] employs probabilistic graphical models to identify the most plausible semantic model of a data source within a combinatorial space. Among the advantages offered by this approach, the authors mention the robustness against noisy information and a straightforward method for taking advantage of relations within the data. Taheriyan et al. [11] propose a system that exploits existing semantic models and an ontology to build a weighted graph that includes all plausible semantic models for the target source. Then, on the basis of the assigned weight, the system computes a ranked list of candidate semantic models.

The main limitation of both approaches is that accuracy is hugely dependent on the availability of semantic models. However, in many domains existing semantic models are not available and manually create them is a very expansive process.

Among other approaches proposed in the literature to address the semantic modeling problem, and in particular the relation inference, a promising one is the exploitation of linked data repositories as background knowledge. Linked data available on the Web [13] or in private repositories include a vast amount of meaningful information, that can be used to learn how different entities are related to each other. As demonstrated by the work of Taheriyan et al. [6], the results of this learning process are helpful to select a path representing the correct semantic interpretation of the target source. We took inspiration from this work to integrate in SeMi a novel mechanism for inferring semantic relations using background linked data. The most important difference between our approach and the work of Taheriyan et al. [6] is that the latter adopts a manual extraction of compound features (e.g., complex graph patterns to represent semantic relations of different lengths), while our method automatically learns latent features for entities and properties, encoding them in a vector space. These features are learnt by the graph neural network, exploiting the local neighborhood structures within the linked data graph.

3. Goal and main architectural requirements

The goal of the SeMi tool is to produce a semantic model, given a data source, a domain ontology, and a background linked data as input. To achieve such a general goal, we identified two architectural requirements:

- *A flexible and modular pipeline*: the first requirement consists in the development of a pipeline where each component can be individually improved or replaced, for a variety of purposes, such as tailoring the tool to a very specific domain, or injecting user input in an intermediate step. In the current implementation of SeMi, only the semantic labeling step enables the possibility of user-based refinements.
- *A graph neural network model included in a production pipeline*: the implementations of graph neural networks in research literature (i) show results in comparison to other models (ii) on available benchmarks and (iii) according to specific evaluation metrics. Nevertheless, the usage of these models in a data pipeline to reach a specific purpose is a complex task. According to this requirement, we needed to include and re-engineer an existing graph neural network to the purpose of semantic relations inference.

4. SeMi pipeline components

The pipeline to produce semantic models includes five main components (Fig. 2):

- Semantic Type Detector (STD);
- Multi-Edge and Weighted Graph Generator (MEWGG);
- Semantic Model Builder (SMB);
- Link Predictor (LP);
- Semantic Model Refiner (SMR).

The input *artifacts* to this pipeline are (i) the input data source, (ii) the domain ontology, (iii) the background linked data; herein we describe them.

1. *Input data source*: an example of an input data source is in Fig. 3. The title of the columns of this table are also known as attributes of the source.
2. *Domain ontology*: an example of an ontology is in Fig. 4. The nodes of the ontology are defined as classes of the ontology, while the edges can be defined as follows:

   (a) *Object properties* connect entities belonging to two different classes. In Fig. 4 “lives in” links a “Person” to a “Place”.

people	birth dates	birth places
Bob Dylan	1941-05-24	Duluth
Elvis Presley	1935-01-08	Tupelo
Little Richard	1932-12-05	Macon

Fig. 3. Example of input data source.

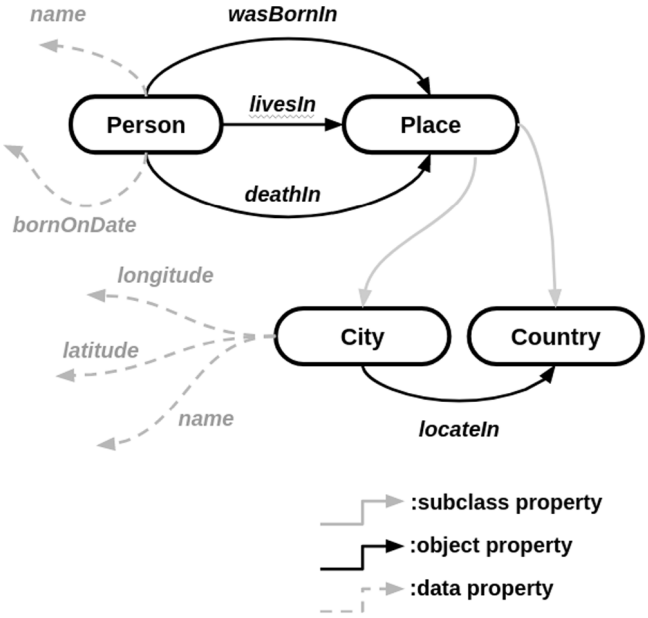


Fig. 4. Example of domain ontology.

(b) *Subclass properties* are a special case of object properties that connect entities belonging to two different classes. In particular, they establish a connection from a broader concept to a more specific concept. In Fig. 4 "Place" and "City" are linked by a subclass property.

(c) *Data properties* connect an entity belonging to a class and a literal value. In our example, "bornOnDate" connects a "Person" with his birth date.

3. *Background linked data*: an example of linked data adopted for training purposes is depicted in Fig. 5. This background knowledge covers data belonging to the same domain of the target source, it adopts a subset of the properties declared in the ontology, and it includes instances of classes of the domain ontology.

The domain ontology provides a formal definition of entity types and properties that connect entities within a specific domain. It generalizes the model of the data, but it does not include details on the entities themselves. The background linked data collects specific information on the entities, that are described and interconnected in a graph structure using types and properties defined by the domain ontology. In other words, the ontology does not contain facts related to the data, but it provides the necessary semantic scaffolding to structure the collection of data facts within the linked data graph.

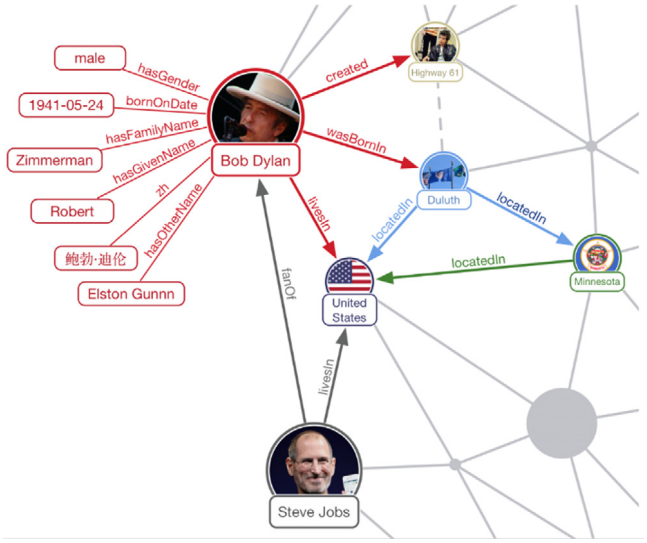


Fig. 5. Example of background linked data.

4.1. Semantic type detector

The STD component takes the data source and the ontology as input. The goal of the STD is to assign a semantic type (also called *semantic label*) to each attribute of a data source. Such semantic label consists of a combination of an ontology class and an ontology data property. As shown in Fig. 1, the semantic label of the attribute "people" is "Person_name", where "Person" is a class and "name" is a data property.

4.2. Multi-edge and weighted graph generator

The MEWGG component receives the semantic types produced by the STD and the domain ontology. The goal of the MEWGG is to build a multi-edge and weighted graph that includes all plausible semantic models for the target source. More specifically, MEWG reconstructs all possible semantic relations between all the semantic types, according to the object properties of the domain ontology. The weights assigned to these object properties are only based on the structure of the ontology (see Section 5.2).

The semantic relation is a path that connects two semantic types. In the simplest case, the semantic relation is an object property. In more complex cases, the path includes different classes and more object properties. For instance, with reference to the semantic model of Fig. 1, "Person" is connected to "City" with a path that includes the object property "wasBornIn", the class "Place", and the subclass property that links "Place" to the class "City". We implemented the algorithm of Knoblock et al. [14] to build the multi-edge and weighted graph.

4.3. Semantic model builder

The SMB component takes the graph produced by the MEWGG as input. The goal of the SMB is to select an initial semantic model, among all plausible semantic models, which encompasses the minimum cost path on the graph that connects all semantic types. The detection of this path in the graph can be considered a steiner tree problem [15]. To resolve the steiner tree problem, we adopt the approach provided by Kou et al. [16]. In our case, the classes of the semantic types are the steiner nodes and the generated semantic model describes the target source in terms of classes and properties of the domain ontology. The detection of the steiner tree in the graph produced by the MEWGG has the

following limit: the object property weights are based only on the structure of the ontology and they do not necessarily reflect the correct semantic interpretation of the target source.

4.4. Link predictor

The LP component is characterized by an offline and online stage. In the offline stage, the input of the LP are RDF facts from background linked data repositories. This background knowledge is adopted for training the graph neural network model, whose output are the embeddings of entities and object properties of RDF facts seen during the training process. In the online stage, these embeddings are used to score the unseen RDF facts, resulting from all plausible semantic models produced by the MEWGG. In details, the RDF fact scores are properly used to adjust the weights assigned to the graph by the MEWGG, incorporating information from the background knowledge provided by linked data. The intuition behind this refinement is that properties used by other people to semantically describe data in a domain are more likely to represent the semantics of the target source in the same domain.

The adoption of the LP component represents a step towards in the semantic modeling process. In fact, in the work of Taheriyan et al. [6] the weights of the semantic relations derived from complex graph patterns are assigned through an inverse relation of their frequency in the background knowledge. In our approach, we consider latent information embodied in the background knowledge which are not the result of a manual and compound features extraction from the linked data, but are automatically extracted by means of the graph neural network (see implementation details in Section 5.4).

4.5. Semantic model refiner

The SMR component receives the initial semantic model built by the SMB, the embeddings to score unseen RDF facts, and all plausible semantic models included in the MEWGG. The SMR computes the aggregation of the RDF score facts that refer to the same semantic relation. Comparing such aggregated values, the SMR replaces (or confirms) the semantic relations that link semantic types classes and performs a new steiner tree detection. Therefore, the goal of the SMR is to provide a new semantic model that describes in a more accurate and rich way the semantics of the target source.

5. Implementation details

This section discusses the implementation of the current SeMi release.

5.1. Semantic type detection querying indexes

To assign a semantic label, the STD measures the similarity [17] between the data values of unlabeled attributes of the target source and the data values of sources with labeled attributes. Such labeled data sources are stored within an Elasticsearch [18] Lucene index: therefore, the STD composes and performs a Lucene query to obtain a ranking of scored semantic types for each attribute of the target source. After this ranking, the user can select which semantic types correctly label the attributes of the target source.

The current implementation of the STD is available in Node.js, because it is suitable to interact with RESTful services provided by Elasticsearch.

5.2. Incremental generation of the multi-edge and weighted graph

The MEWGG component incrementally creates a graph G through the following steps (see Algorithm 1 for more details):

- ***Addition of semantic types*** (lines 1–5 of Algorithm 1): for each semantic type the algorithm creates and adds to G the following graph structures: (i) a class node, (ii) a data node, (iii) a weighted edge between the two nodes. Then, it assigns a weight of 1 to this edge.
- ***Addition of closure nodes*** (lines 6–9 of Algorithm 1): for each class node in G the algorithm performs a SPARQL query [7] to get the related ontology classes. Such classes are added as new class nodes (*closures*) to G.
- ***Addition of edges between class nodes*** (lines 12–21 of Algorithm 1): object properties (see Section 4) connecting all class nodes in G are retrieved from the ontology through a SPARQL query. Such properties are added to G as new edges.
- ***Assignment of a weight to each new edge***: different types of object property p can connect the class nodes c_u and c_v in G:
  - *direct properties* (lines 14–15 of Algorithm 1): p is a direct property between c_u and c_v if they are respectively defined as domain and range of the p in the ontology. The algorithm assigns a weight of 100 to edges corresponding to direct properties;
  - *inherited properties* (lines 16–17 of Algorithm 1): p is an inherited property between c_u and c_v if its domain contains one of the super classes of c_u and its range contains one of the super classes of c_v. The algorithm assigns a weight of $100 + \epsilon$ to edges corresponding to inherited properties;
  - *subclass properties* (lines 18–19 of Algorithm 1): p is a subclass property between c_u and c_v if they are linked by a special property in the ontology called *rdfs:subClassOf*. The algorithm assigns a weight of $100/\epsilon$ to edges corresponding to subclass properties.

This component is implemented in Node.js, because it is suitable to manage the response of SPARQL queries in an asynchronous way.

5.3. Semantic model definition through steiner trees and SPARQL syntax

The detection of the shortest path within the graph that connects the class nodes of the semantic types is a steiner tree problem. The time complexity of the steiner algorithm (see Section 4.3) is equal to $O(|N_d||N_c|^2)$ in which N_d is the set of data nodes and N_c is the set of class nodes in G. Considering the dataset adopted in our experimental evaluation (see Section 6.1 for more details), time complexity for this step is negligible compared to other steps, for instance the training time of the graph neural network. However, it can be an indicator for other users that intend to adopt SeMi for building their own KGs.

The detected steiner tree represents, in the form of a graph, the initial semantic model. Nevertheless, the semantic model needs to be converted using the syntax of a mapping language, such as R2RML [19], RML [20], or SPARQL. Semantic models expressed using a mapping language can be processed by engines such as MIRROR [21], RML Mapper [20], and TARQL [22] to generate KGs. In our implementation we chose a specific SPARQL syntax that can be processed by the JARQL library [23]. The source code that converts the graph representation of the semantic model in SPARQL is implemented in Node.js, while JARQL is implemented in JAVA within an external library.

Algorithm 1: Generate the Multi-Edge and Weighted Graph

Input: Semantic Types $STs\{class, data\_prop, source\_attr\}$
Input: Domain Ontology $O\{classes, object\_props\}$
Output: Graph G

1 **for** *loop on STs* **do**
2 $G.add\_class\_node(ST\{class\})$;
3 $G.add\_data\_node(ST\{source\_attr\})$;
4 $G.add\_edge(ST\{class\}, ST\{source\_attr\}, ST\{data\_prop\}, w = 1)$;
5 **end**
6 $closures \leftarrow sparql\_closure(O\{classes\})$;
7 **for** *loop on closures* **do**
8 $G.add\_class\_node(closure)$;
9 **end**
10 $c_u s \leftarrow get\_class\_nodes(G)$;
11 $c_v s \leftarrow get\_class\_nodes(G)$;
12 **for** *loop on* $c_u s$ **do**
13 **for** *loop on* $c_v s$ **do**
14 $direct\_props \leftarrow sparql\_direct(c_u, c_v, O\{object\_props\})$;
15 $G.add\_edges(c_u, c_v, direct\_props, w = 1)$;
16 $inherited\_props \leftarrow sparql\_inherited(c_u, c_v, O\{object\_props\})$;
17 $G.add\_edges(c_u, c_v, inherited\_props, w = 1 + \epsilon)$;
18 $subclass\_props \leftarrow sparql\_subclass(c_u, c_v, O\{object\_props\})$;
19 $G.add\_edges(c_u, c_v, subclass\_props, w = 1/\epsilon)$;
20 **end**
21 **end**

5.4. Graph neural network architecture for link prediction

The LP is a graph neural network composed by two main units: (i) an *encoder*, which is a neural architecture called Relational Graph Convolutional Networks (R-GCN) [24]. The goal of the R-GCN is to assign embeddings, to entities within the background linked data (ii) a *decoder*, which manipulates such vectors with a factorization method in order to predict new facts in the background linked data (see Algorithm 2 for more details). The propagation model of the R-GCN, for each layer l of the network, is defined as follows (line 6 of Algorithm 2):

$$h_i^{(l+1)} = \sigma\left(\sum_{r \in \mathcal{R}} \sum_{j \in \mathcal{J}_i^r} \frac{1}{c_{i,r}} W_r^{(l)} h_j^{(l)} + W_0^{(l)} h_i^{(l)}\right) \quad (1)$$

where:

- $h_i^l \in R^{d^{(l)}}$ denotes the hidden state of the node v_i (its vector representation) in the lth layer of the neural network
- σ is an element-wise activation function such as ReLU;
- $\mathcal{J}_i^r$ is the set of indices of the neighbors of node i under the relation $r \in \mathcal{R}$;
- $c_{i,r}$ is a hyperparameter of the application that controls the entity of the normalization;
- W is a weight matrix.

The computation of Eq. (1) is performed in parallel for all nodes at each network update. By stacking up several layers, it is possible to capture and encode the relations between nodes across multiple steps.

The purpose of the decoder is to predict new facts within the background linked data, by scoring candidate new RDF facts in the form *(subject, relation, object)* using a factorization function. In practice, the decoder could employ any factorization or scoring function, so we chose to employ DistMult [25], as done by Schlichtkrull et al. [24]. Given a background linked data, we want to assign a certain score $f(s, r, o)$ to a candidate fact (s, r, o). DistMult associates each relation r with a diagonal matrix R_r, and the score for a given candidate fact is computed as follows (line 12 of Algorithm 2):

$$f(s, r, o) = e_s^T R_r e_o \quad (2)$$

The model is trained by means of the negative sampling: for each training sample, a set of negative samples is generated by randomly corrupting either the subject or the object of the fact. The model is optimized so that the positive facts are scored higher than the negative ones.

To include the LP in the semantic model pipeline we modified the implementation of [24], in order to produce embeddings that are used by next block to score RDF facts generated by the plausible semantic models of the target source. The LP is implemented in Python adopting the Deep Graph Library [26] based on the PyTorch framework.

Algorithm 2: Link prediction with the graph neural network

Input: $complete\_set$, $train\_set$, $valid\_set$, $test\_set$, adj_matrix
Output: $entity\_embs$, $property\_embs$

1 $entity\_dict \leftarrow create\_entity\_dict(complete\_set)$;
2 $property\_dict \leftarrow create\_property\_dict(complete\_set)$;
3 $entity\_embs \leftarrow initialize\_embs(entity\_dict)$;
4 $property\_embs \leftarrow initialize\_embs(property\_dict)$;
/* `Forward: aggregate and update entity embeddings using the local neighborhood structure` */
5 $entity\_embs \leftarrow update\_features(entity\_embs, adj\_matrix)$;
/* `Backward: computing gradients and update entity and property embeddings to reconstruct the facts` */
6 $s\_tr_{id}, p\_tr_{id}, o\_tr_{id} \leftarrow extract\_ids(train\_set)$;
7 $s\_va_{id}, p\_va_{id}, o\_va_{id} \leftarrow extract\_ids(valid\_set)$;
8 $s\_te_{id}, p\_te_{id}, o\_te_{id} \leftarrow extract\_ids(test\_set)$;
9 **for** *epochs* **do**
10 $grads \leftarrow compute\_grads(emb(s\_tr_{id}) * emb(p\_tr_{id}) * emb(o\_tr_{id}))$;
11 $entity\_embs, property\_embs \leftarrow update\_weights(grads)$;
12 $model \leftarrow best\_on\_validation(emb(s\_va_{id}), emb(p\_va_{id}), emb(o\_va_{id}))$;
13 **end**
/* `Evaluation: evaluate the best model on the test set and return computed embeddings` */
14 $fact\_scores \leftarrow evaluate(emb(s\_te_{id}), emb(p\_te_{id}), emb(o\_te_{id}), model)$;

5.5. Semantic model refinement based on fact scores

The SBR determines whether to refine or validate the initial semantic model. The weight of each semantic relation of all plausible semantic models are updated according to the aggregated scores of the related RDF facts. These scores are then weighted on the number of occurrences of each semantic relation. On the basis of this aggregated score, the algorithm performs a new steiner tree detection and propagates any changes to the initial semantic model. This component is implemented in Python.

6. Evaluation

Our objective is to investigate the capability of SeMi in inferencing accurate semantic relations within a target source. In this

Section, we described the dataset adopted for the evaluation and we reported details on the validation procedure, discussing the obtained results.

6.1. Evaluation dataset

The evaluation dataset includes 15 target sources available in JSON format on the advertising domain. The domain ontology is an extension of Schema.org [27], which contains 736 classes and 1081 properties. To prepare the background knowledge for each target source we employed the leave-one-out setting. In practice, if k is the number of sources in our dataset, the background linked data assigned to each target source is created from the RDF facts obtained by the other $k - 1$ sources. In other words, each background knowledge includes RDF facts which come from all the sources, except those obtained from the target source. The leave-one-out setting guarantees that the background knowledge does not contain the facts related to the semantic relations within the target source, that have to be predicted in the experiment. Nevertheless, we included the RDF facts from the target source that are specifically related to the semantic types. This step has no impact on the performance of semantic relation inference, because semantic type facts are not considered during the experimental evaluation. However, additional facts derived from semantic types are required within the training process, because the graph neural network has to learn the latent features of all entities, including the target source entities, in order to execute the link prediction task. To ensure the quality of linked data repositories adopted as background knowledge, the RDF facts are generated using the ground-truth semantic models.

The original dataset, including the target sources, the ontology, and the ground-truth semantic models are available in the Taheriyan GitHub repository.[3] We chose to perform the evaluation on this specific dataset, because it represents the case in which the approach of Taheryan et al. [6] obtained the best performance. The background linked data for training the graph neural network are novels and constructed specifically for our experiment. To simplify the access to all data employed in the evaluation, we created two folders in the SeMi GitHub repository: (i) input sources, ontologies, and background linked data are available at https://github.com/giuseppefutia/semi/tree/master/data/taheriyan2016/task_04; this folder includes also the semantic models generated by our system. The ground truth semantic models and the semantic models computed by baseline algorithms (see Subsection 6.2.2 for more details) are available at https://github.com/giuseppefutia/semi/tree/master/evaluation/taheriyan2016/task_04. Details on the input sources (number of attributes), on the background linked data (number of entities and object properties), and on ground-truth semantic models (number of nodes and semantic relations) are reported in Table 1.

Table 1
Details on target sources, background linked data, and ground truth semantic models.

Sources	#attrs	Background LD		Ground-Truth SMs	
		#entities	#facts	#nodes	#links
alaskaslist	8	3396	6954	12	3
armslist	20	3396	6793	15	4
dallasguns	15	3379	6940	23	7
elpasoguntrader	8	3396	7044	13	4
floridagunclassifieds	16	3396	6904	23	6
floridaguntrader	10	3396	6774	15	4
gunsinternational	10	3396	6945	19	4
hawaiiguntrader	7	3396	7122	11	3
kyclassifieds	10	3396	6945	14	3
montanagunclassifieds	9	3396	7104	14	4
msguntrader	11	3375	7086	16	4
nextechclassifieds	20	3396	6198	32	11
shooterswap	11	3396	7041	15	3
tennesseegunexchange	14	3396	7104	21	6
theoutdoorstrader	12	3396	6784	18	5

6.2. Evaluation procedure and results

The evaluation procedure relied on two different steps: (i) the validation of the graph neural network; (ii) the validation of the semantic relation inference task.

6.2.1. Validation of the graph neural network

The first step of the evaluation procedure consisted in the validation of the link prediction (LP) mechanism, based on the graph neural network trained with background linked data. Each background linked data assigned to a target source was splitted into three different datasets: the training set, the validation set, and the test set. Then, these datasets are taken as input by the graph neural network and used to perform the LP. To measure the performance of the LP we employed as metric the standard Mean Reciprocal Rank (MRR) [28]. In our case the MRR provides an insight on the correctness of the facts reconstructed by the graph neural network, exploiting the learned embeddings of entities and object properties. For each background linked data, Table 2 reports: (i) details on the number of facts included in the training set, the validation set, and the test set respectively; (ii) the resulting MRR on the test dataset. An overview of the most significant hyperparameters used to train the graph neural network is available in Table 3.

To understand the effectiveness of the graph neural network on our background linked data, we compared our results with the MRR values obtained by the graph neural network on FB15-k237[29], one of the most well-known dataset for benchmarking KG completion tasks. These MRR values reported in literature [24] are:

- MRR Raw: 0.158
- Hits 1: 0.153
- Hits 3: 0.258

MRR values obtained on background linked data (Raw and Hits 1) are higher than the MRR values obtained on FB15-k237, therefore the graph neural network performed very well on our dataset. In practice, this means that entity and object property embeddings encode in a proper way the local neighborhood structure of the background knowledge. As a consequence, these embeddings can be suitable to score facts derived from all plausible semantic relations in the target source, identifying the most accurate ones.

6.2.2. Validation of the semantic relation inference task

In the second step of the experimental evaluation we investigated if SeMi performs better than the approach of Taheriyan et al. [6], that is currently implemented in Karma [14]. Furthermore, we compared our approach with two different base lines: (i) a method exploiting only the frequency of semantic relations of length 1 within linked data (no heuristics to extract and rank complex the graph patterns, as done by Taheriyan et al. [6]); (ii) a method based on the detection of a steiner tree on a multi-edge and weighted graph, whose weights are assigned using only the ontology structure (no background knowledge). We focused the evaluation procedure on the semantic relation inference task and for this reason we considered that the correct semantic types are already available. We evaluated the accuracy of computed semantic models in terms of precision and recall, by comparing them

[3] https://github.com/taheriyan/iswc-2016/blob/master/weapon-ads.zip.

Table 2
Number of facts in the training, the validation, and the testing set and the MRR values obtained by the graph neural network on each background linked data.

Sources	Background LD - #Facts			Mean Reciprocal Rank (MRR)		
	Training	Validation	Testing	Raw	Hits @1	Hits @3
alaskaslist	6264	345	345	0.202556	0.171014	0.221739
armslist	6123	335	335	0.189313	0.156716	0.214925
dallasguns	6250	345	345	0.222723	0.201449	0.233333
elpasoguntrader	6344	350	350	0.175496	0.135714	0.198571
floridagunclassifieds	6214	345	345	0.213165	0.191304	0.224638
floridaguntrader	6104	335	335	0.207233	0.174627	0.229851
gunsinternational	6264	345	345	0.205095	0.188406	0.211594
hawaiiguntrader	6412	355	355	0.208059	0.180282	0.223944
kyclassifieds	6255	345	345	0.191376	0.163768	0.207246
montanagunclassifieds	6394	355	355	0.233740	0.212676	0.245070
msguntrader	6386	350	350	0.209148	0.188571	0.222857
nextechclassifieds	5588	305	305	0.204046	0.177049	0.216393
shooterswap	6341	350	350	0.226965	0.205714	0.241429
tennesseegunexchange	3694	355	355	0.203350	0.180282	0.214085
theoutdoorstrader	6114	335	335	0.185680	0.159701	0.205970

Table 3
Graph neural network hyperparameters.

Hyperparameters	Values
Dropout	0.2
Hidden layers	2
Hidden neurons	100
Learning rate	1e−2
Epochs	6000
Regularization	0.01
Edges sample size	1000
Negative sampling	10

with the ground-truth semantic models. If the correct semantic model of the source s is denoted as sm and the semantic model computed by the system is denoted as sm', precision and recall are defined by Taheriyan et al. [6] as follows:

$$precision = \frac{rel(sm) \cap rel(sm')}{rel(sm')} \tag{3}$$

$$recall = \frac{rel(sm) \cap rel(sm')}{rel(sm)} \tag{4}$$

where $rel(sm)$ is the set of triples (u, v, e), where e is an object property from the ontology class u to the ontology class v. Table 4 reports the results in terms of precision and recall obtained by: (i) SeMi; (ii) the approach of Taheriyan et al. [6]) (Tahe in the Table); (iii) the baseline exploiting only the frequency of semantic relations of length 1 (Occs in the Table); (iv) the baseline using the steiner tree performed on a weighted graph based on the ontology structure (Stei in the Table).

In our evaluation experiment SeMi always obtained a better accuracy in terms of precision and recall, compared to: (i) the baseline that captures the frequency of semantic relations of length 1; (ii) the baseline of the steiner tree built on the graph weighted according to the ontology structure. In this experiment we employed the dataset in which the Taheriyan et al. [6] approach obtained the best results. The results show that our approach outperforms the state of the art in case of the following data sources: "dallasguns", "floridagunclassifieds", "gunsinternational", and "shooterswap". These sources have the most complex structure in terms of number of nodes and links in the ground-truth semantic models (see Table 1 for more details.) The accuracy improvement in these specific cases can be explained considering the different levels of features extraction. The system of Taheriyan identifies the best semantic relation considering two metrics: (i) cost and (ii) coherence. The cost of the semantic relation derived from a graph pattern is computed according to an inverse function of its popularity. As a consequence, computing the minimum cost is equivalent to select the most frequent semantic relation. On the other side, the coherence gives priority to longer semantic relations. From a different perspective, SeMi assigns to each plausible semantic relation a cost, aggregating the scores obtained by each RDF fact of the semantic relation. This score is computed by the DistMult factorization method (see Section 5.4) that takes as input the embeddings of the subject, the predicate, and the object of the RDF fact. Embeddings represent more granular and latent information that incorporate hidden regularities in the data that cannot be detected adopting a manual approach for the extraction of compound features. For many other data sources, SeMi reaches the accuracy in terms of precision and recall of the state of the art [6]. Therefore, we believe the our approach allows to distinguish very well the correct relation, exploiting the neighborhood structure of the graph. On the other side, we noticed that the performance of SeMi in terms of precision drastically low in presence of many data attributes within sources that are characterized by the same semantic type (see "elpasoguntrader" and "nextechclassifieds"). For instance, the "nextechclassifieds" source includes 5 different attributes that are labeled with the ontology class "schema:Offer". According to the ground-truth semantic model of this source, the class of the semantic type "schema:Offer1" is linked to the other 4 entities classes with the object property "schema:relatedTo". Nevertheless, this type of graph structure represents an anomaly because it never appears in the semantic models of the other sources, that have been exploited for creating the background knowledge of "nextechclassifieds". We believe that including in the background linked data analogous graph structures the performance of SeMi should increase.

Regarding the scalability issues, in their work, Taheriyan et al. [6] underlines that performing a single SPARQL query on Virtuoso [30] repository with more than three million of triple requires approximately one hour for the graph patterns of length five (Mac OS X System, 2.3 GHz Intel Core i7 CPU, 16 GB of RAM). This time is expected to grow exponentially as the dimension of semantic relations increase. The training of the graph neural network is the operation performed by SeMi, that specifically involves the background linked data, as the SPARQL queries performed by Taheriyan et al. [6]. The training execution time is less than 30 min and does not encounter issues related to the growing of semantic relations complexity. The training step has been performed on a Centos 7 - OpenHPC 1.3 System, nVidia Tesla V100 SXM GPU, 32 GB of memory, 5120 cuda cores.

7. Package components and practical uses

The current version of SeMi includes the following components in the GitHub repository:

Table 4
Results of the semantic relation inference in terms of precision and recall.

Sources	Precision				Recall			
	Semi	Tahe	Occs	Stei	Semi	Tahe	Occs	Stei
alaskaslist	**1**	1	0.667	0	**1**	1	0.667	0
armslist	**0.750**	0.750	0.500	0	**0.750**	0.750	0.500	0
dallasguns	**0.667**	0.570	0.500	0	**0.570**	0.570	0.428	0
elpasoguntrader	0.500	1	0.500	0.250	0.500	0.750	0.500	0.250
floridagunclassifieds	**0.833**	0.800	0.167	0	**0.833**	0.670	0.167	0
floridaguntrader	**1**	1	0.750	0	**1**	1	0.750	0
gunsinternational	**0.750**	0.600	0.250	0	**0.750**	0.750	0.250	0
hawaiiguntrader	**1**	1	1	0	**1**	1	1	0
kyclassifieds	**1**	1	0.333	0.333	**1**	1	0.333	0.333
montanagunclassifieds	0.750	1	0.500	0	0.750	1	0.500	0
msguntrader	**0.670**	0.670	0.667	0	**0.500**	0.500	0.500	0
nextechclassifieds	0.454	1	0.182	0	**0.454**	0.360	0.182	0
shooterswap	**1**	0.750	1	0	**1**	1	1	0
tennesseegunexchange	0.667	1	0.500	0.167	0.667	1	0.500	0.167
theoutdoorstrader	0.800	0.830	0.200	0.200	0.800	1	0.200	0.200

- ***semantic_typing*** (see Section 4.1):
  - *semantic_label.js* performs the query on the Elasticsearch index to obtain the semantic labels ranking;
- ***semantic_modeling*** (see Sections 4.2 and 4.3):
  - *graph.js* generates the multi-edge and weighted graph that includes all plausible semantic models;
  - *steiner_tree.js* runs the steiner tree algorithm on such graph to identify the initial semantic model;
  - *jarql.js* serializes the steiner tree graph representation in SPARQL syntax;
- ***link_prediction*** (see Section 4.4):
  - *link_predict.py* performs the link prediction task using the R-GCN encoder and the DistMult decoder both implemented in *model.py*;
  - *utils.py* produces as output the scores of the facts generated through all plausible semantic models;
- ***semantic_refinement*** (see Section 4.5):
  - *refine.py* refines the graph produced by *steiner_tree.js* according to the aggregation of the RDF facts.

8. Conclusion and future work

Automatic and semi-automatic techniques for semantic modeling are needed to build large-scale KGs. In this paper we described SeMi (SEmantic Modeling machIne), a tool that is able to semi-automatically generate semantic models for KG generation, adopting a graph neural network trained with background linked data. SeMi extracts features from linked data in an automatic way, learning latent features from the local neighborhood structures of the linked data graph. These latent features are aggregated into a vector representation of entities and properties (*embeddings*), which are employed to predict the semantic relations within the target source. We demonstrated that the adoption of the embeddings increases the accuracy in reconstructing the correct relations, compared to manually-selected and compound features. Furthermore, we show that the graph neural network training is a more scalable procedure than the extraction of increasingly complex graph patterns through SPARQL queries. At the best of our knowledge, this is the first technique that exploits a graph neural network within the semantic modeling process. In addition, we built SeMi with a modular pipeline where each component can be easily replaced: we deem this feature relevant because it enables higher reuse and makes possible to replace modules with alternative ones in the future.

At the moment, we are experimenting the link prediction mechanism of SeMi (Section 5.4) in the field of scholarly literature, improving a former approach used to support and partially automatize systematic literature reviews [31,32]. The prediction mechanism is embedded in a platform called Geranium, that allows users to search for scientific publications and authors by inferred semantic topics rather than keywords, and to show implicit connections between researchers from different groups or departments. The platform works on top of the research institutional repository of Politecnico di Torino (https://iris.polito.it/), it is currently in alpha version and it is being tested by a pool of directors of Politecnico.

As far as future developments are concerned, we intend to extend SeMi to compute semantic models, in a context of a constant evolution of the ontology or of the KG [33]. This feature can be particularly useful in real-world situations, in which new data sources and the related semantic models are included in a continuous data integration process. The current implementation of our system is able to parse XML and JSON files with basic structures, which do not include tag parameters or deep nested structures. We aim to increase the robustness of our parsers, in order to process files with more complex structures. Moreover, we intend to employ SeMi in a specific scenario related to the public procurement, that we covered and analyzed in previous works [34,35]. Finally, we plan to develop an interactive user interface that shows the construction of the semantic model along each block of the pipeline, enabling the possibility to the user to intervene in each step of the semantic modeling process. We believe that a valuable solution is presenting a list of candidates as output of each block, instead of the most plausible output from the implemented algorithms. For instance, the current implementation of the Semantic Model Refiner (Sections 4.5 and 5.5) selects the correct semantic relation, taking into account the best score obtained from the Link Predictor (Sections 4.4 and 5.4). On the contrary, an interactive system would propose a list of possible refinements, that are ranked according to different scores computed by the Link Predictor. Therefore, the interactive system will guide the user in the definition of the correct semantic model during the entire process.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

Computational resources provided by hpc@polito, which is a project of Academic Computing within the Department of Control and Computer Engineering at the Politecnico di Torino (http://www.hpc.polito.it).

References

[1] Lassila O, Swick RR, et al. Resource description framework (rdf) model and syntax specification. Citeseer; 1998.
[2] Hooi B, Song HA, Beutel A, Shah N, Shin K, Faloutsos C. Fraudar: Bounding graph fraud in the face of camouflage. In: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining. ACM; 2016, p. 895–904.
[3] Athreya RG, Ngonga Ngomo A-C, Usbeck R. Enhancing community interactions with data-driven chatbots–the DBpedia chatbot. In: Companion proceedings of the the web conference 2018. International World Wide Web Conferences Steering Committee; 2018, p. 143–6.
[4] Hasnain A, Kamdar MR, Hasapis P, Zeginis D, Warren CN, Deus HF, et al. Linked biomedical dataspace: lessons learned integrating data for drug discovery. In: International semantic web conference. Springer; 2014, p. 114–30.
[5] Ramnandan SK, Mittal A, Knoblock CA, Szekely P. Assigning semantic labels to data sources. In: European semantic web conference. Springer; 2015, p. 403–17.
[6] Taheriyan M, Knoblock CA, Szekely P, Ambite JL. Leveraging linked data to discover semantic relations within data sources. In: International semantic web conference. Springer; 2016, p. 549–65.
[7] DuCharme B. Learning SPARQL: querying and updating with SPARQL 1.1. O'Reilly Media, Inc.; 2013.
[8] Pomp A, Poth L, Kraus V, Meisen T. Enhancing knowledge graphs with data representatives. 2019.
[9] Ruemmele N, Tyshetskiy Y, Collins A. Evaluating approaches for supervised semantic labeling. 2018, arXiv preprint arXiv:1801.09788.
[10] Taheriyan M, Knoblock CA, Szekely P, Ambite JL. A graph-based approach to learn semantic descriptions of data sources. In: International semantic web conference. Springer; 2013, p. 607–23.
[11] Taheriyan M, Knoblock CA, Szekely P, Ambite JL. Learning the semantics of structured data sources. Web Semant Sci Serv Agents World Wide Web 2016;37:152–69.
[12] Vu B, Knoblock C, Pujara J. Learning semantic models of data sources using probabilistic graphical models. In: The world wide web conference. ACM; 2019, p. 1944–53.
[13] Bizer C, Heath T, Berners-Lee T. Linked data: The story so far. In: Semantic services, interoperability and web applications: Emerging concepts. IGI Global; 2011, p. 205–27.
[14] Knoblock CA, Szekely P, Ambite JL, Goel A, Gupta S, Lerman K, et al. Semi-automatically mapping structured sources into the semantic web. In: Extended semantic web conference. Springer; 2012, p. 375–90.
[15] Hwang FK, Richards DS. Steiner tree problems. Networks 1992;22(1):55–89.
[16] Kou L, Markowsky G, Berman L. A fast algorithm for Steiner trees. Acta Inform 1981;15(2):141–5.
[17] Mihalcea R, Corley C, Strapparava C, et al. Corpus-based and knowledge-based measures of text semantic similarity. In: Aaai, vol. 6. 2006, p. 775–80.
[18] Gormley C, Tong Z. Elasticsearch: the definitive guide: a distributed real-time search and analytics engine. O'Reilly Media, Inc.; 2015.
[19] Das S, Sundara S, Cyganiak R. R2RML: RDB to RDF mapping language. W3C recommendation (2012). 2016.
[20] Dimou A, Vander Sande M, Colpaert P, Verborgh R, Mannens E, Van de Walle R. RML: A generic language for integrated RDF mappings of heterogeneous data. In: Ldow, vol. 1184. 2014.
[21] de Medeiros LF, Priyatna F, Corcho O. MIRROR: Automatic R2RML mapping generation from relational databases. In: International conference on web engineering. Springer; 2015, p. 326–43.
[22] Cyganiak R. Tarql (sparql for tables): Turn csv into rdf using sparql syntax. Technical report, Jan. 2015, 2015, http://tarql.github.io.
[23] Schiavone L, Morando F, Allavena D, Bevilacqua G. Library data integration: the CoBiS linked open data project and portal. In: Italian research conference on digital libraries. Springer; 2018, p. 15–22.
[24] Schlichtkrull M, Kipf TN, Bloem P, van den Berg R, Titov I, Welling M. Modeling relational data with graph convolutional networks. In: European semantic web conference. Springer; 2018, p. 593–607.
[25] Yang B, Yih W-t, He X, Gao J, Deng L. Embedding entities and relations for learning and inference in knowledge bases. 2014, arXiv preprint arXiv:1412.6575.
[26] Wang M, Yu L, Gan Q, Zheng D, Gai Y, Ye Z, et al. Deep graph library. 2018, URL http://dgl.ai.
[27] Guha RV, Brickley D, Macbeth S. Schema.org: evolution of structured data on the web. Commun ACM 2016;59(2):44–51.
[28] Craswell N. Mean reciprocal rank. In: Encyclopedia of database systems, vol. 1703. 2009.
[29] Toutanova K, Chen D. Observed versus latent features for knowledge base and text inference. In: Proceedings of the 3rd workshop on continuous vector space models and their compositionality. 2015, p. 57–66.
[30] Erling O, Mikhailov I. RDF support in the virtuoso DBMS. In: Networked knowledge-networked media. Springer; 2009, p. 7–24.
[31] Rizzo G, Tomassetti F, Vetro A, Ardito L, Torchiano M, Morisio M, et al. Semantic enrichment for recommendation of primary studies in a systematic literature review. Digit Scholarsh Humanit 2017;32(1):195–208.
[32] Tomassetti F, Rizzo G, Vetro A, Ardito L, Torchiano M, Morisio M. Linked data approach for selection process automation in systematic reviews. In: 15th annual conference on evaluation & assessment in software engineering. IET; 2011, p. 31–5.
[33] Pomp A, Lipp J, Meisen T. You are missing a concept! enhancing ontology-based data access with evolving ontologies. In: 2019 IEEE 13th international conference on semantic computing. IEEE; 2019, p. 98–105.
[34] Futia G, Morando F, Melandri A, Canova L, Ruggiero F. ContrattiPubblici.org, a semantic knowledge graph on public procurement information. In: AI approaches to the complexity of legal systems. Springer; 2015, p. 380–93.
[35] Futia G, Melandri A, Vetrò A, Morando F, De Martin JC. Removing barriers to transparency: A case study on the use of semantic technologies to tackle procurement data inconsistency. In: European semantic web conference. Springer; 2017, p. 623–37.