

# Extracting Entity-specific Substructures for RDF Graph Embeddings

Muhammad Rizwan Saeed<sup>a,\*</sup>, Charalampos Chelmis<sup>b</sup> and Viktor K. Prasanna<sup>a</sup>

<sup>a</sup> *Ming Hseih Department of Electrical Engineering, University of Southern California, CA, USA*  
E-mails: saeedm@usc.edu, prasanna@usc.edu

<sup>b</sup> *Department of Computer Science, University at Albany - SUNY, NY, USA*  
E-mail: cchelms@albany.edu

**Abstract.** Knowledge Graphs (KGs) have become useful sources of structured data for information retrieval and data analytics tasks. Enabling complex analytics, however, requires entities in KGs to be represented in a way that is suitable for Machine Learning tasks. Several approaches have been recently proposed for obtaining vector representations of KGs based on identifying and extracting relevant graph substructures using both uniform and biased random walks. However, such approaches lead to representations comprising mostly *popular*, instead of *relevant*, entities in the KG. In KGs, in which different types of entities often exist (such as in Linked Open Data), a given target entity may have its own distinct set of most *relevant* nodes and edges. We propose *specificity* as an accurate measure of identifying most relevant, entity-specific, nodes and edges. We develop a scalable method based on bidirectional random walks to compute specificity. Our experimental evaluation results show that specificity-based biased random walks extract more *meaningful* (in terms of size and relevance) substructures compared to the state-of-the-art and the graph embedding learned from the extracted substructures perform well against existing methods in common data mining tasks.

Keywords: Graph Embedding, Linked Open Data, Data Mining, Recommender Systems, RDF, SPARQL, Semantic Web, DBpedia

## 1. Introduction

*Knowledge Graphs* (KGs), i.e., graph-structured knowledge bases, store information as entities and the relationships between them, often following some schema or ontology [1]. With the emergence of Linked Open Data [2], DBpedia [3], and Google Knowledge Graph<sup>1</sup>, large-scale KGs have drawn much attention and have become important data sources for many data mining and other knowledge discovery tasks [4–10]. As such algorithms work with the propositional representation of data (i.e., feature vectors) [11], several adaptations of language modeling approaches such as word2vec [12]

and GloVe [13] have been proposed for generating graph embedding for entities in a KG. As a first step for such approaches, a *representative* neighborhood for each target entity in the KG must be acquired. To accomplish this task approaches based on biased random walks [14, 15] have been proposed. Such approaches use weighting schemes to make a particular set of edges or nodes more likely to be included in the extracted subgraphs than others. However, weighting schemes based on metrics such as frequency or PageRank [14, 16] tend to favor *popular* (or densely connected) nodes in the *representative* subgraphs of target entities at the expense of semantically more relevant nodes and edges.

---

\* Corresponding author. E-mail: saeedm@usc.edu.

<sup>1</sup> <https://www.blog.google/products/search/introducing-knowledge-graph-things-not/>

In this paper, we focus on RDF<sup>2</sup> KGs which are encoded using the *Resource Description Framework* (RDF) syntax and constitute Linked Open Data [17, 18]. We assert that the *representative* neighborhoods of different types of entities (e.g., book, film, drug, athlete) in RDF KGs may comprise distinct sets of defining relationships. Our objective is to automatically identify such relationships and use them to extract entity-specific representations. This is in contrast to the scenario where extracted representations are KG-specific because of the inclusion of popular nodes and edges, irrespective of their semantic relevance to the target entities. Additionally, we want to identify the most relevant neighborhood of a target entity without venturing into *unrelated* neighborhoods of close-by entities. For example, when identifying the most relevant representation for a film, the director’s name or style of film-making should be more likely to be included in the representation than her year or place of birth.

To address this challenge:

- We propose *specificity* as an accurate measure for assigning weights to those semantic relationships which constitute the most intuitively interpretable representations for a given set or type of entities.
- We provide a scalable method of computing specificity for semantic relationships of any depth in large-scale KGs.
- We show that specificity-based biased random walks can extract more meaningful (intuitively relevant and compact) entity representations as compared to the state-of-the-art.
- To demonstrate the potential of our specificity-based approach for real-world applications, we train neural language model (Skip-Gram [19]) for generating graph embedding from the extracted entity-specific representations and use the generated embeddings for the task of entity recommendation, regression, and classification for select entities in DBpedia.

This paper considerably extends [20], in which we introduced the metric of *specificity* for the first time. We provide additional experiments to show that the vector embeddings can not only be used for entity recommendation but also for regression

and classification tasks. We also propose a variation of specificity called *Specificity<sup>H</sup>* which takes into account the hierarchy of classes in the schema ontology associated with a KG, discussed in Section 4.1.

The rest of this paper is structured as follows. In Section 2, we provide a brief overview of related work. In Section 3, we provide the necessary background. In Section 4 we motivate and introduce the concept of specificity. In Section 5, we present a scalable method for computing specificity. In Section 6, we present results highlighting beneficial characteristics of specificity on DBpedia. In Section 7 we conclude with a summary and an outlook on future work.

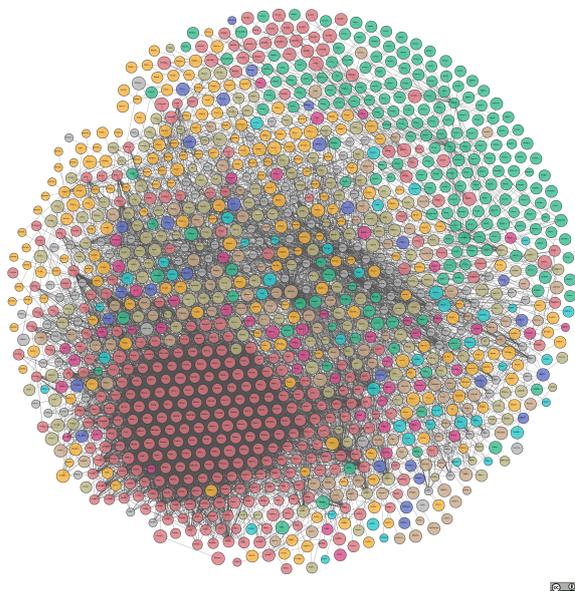


Fig. 1. Linked Open Data [Source: <http://lod-cloud.net/>]

## 2. Related Work

Linked Open Data (LOD) [21] is a massive KG shown in Figure 1 where each node itself is a Knowledge Graph. Each of these individual KGs comes from semantically annotating and integrating unstructured data and publishing as structured data on the web using the principles of Linked Data or Semantic Web [22–24]. LOD cloud currently comprise 1205 KGs<sup>3</sup>. Few of the KGs

<sup>2</sup><https://www.w3.org/TR/rdf-concepts/>

<sup>3</sup>Source: <http://lod-cloud.net/>

part of LOD are *Wikidata*<sup>4</sup>, *Freebase*<sup>5</sup>, and *DBpedia*<sup>6</sup> [25, 26].

Due to its open availability, heterogeneity, and cross-domain nature, LOD is increasingly becoming a valuable source of information in many data mining tasks. However, most data mining algorithms work with a propositional feature vector representation of the data [11]. Recently, graph embedding techniques have become a popular area of interest in the research community. An embedding maps entire graph or individual nodes into low dimensional vector space, preserving as much information as possible related to its neighborhood [27–29].

Numerous techniques have been proposed for generating appropriate representations of KGs for knowledge discovery tasks. Graph kernel-based approaches simultaneously transverse the neighborhoods of a pair of entities in the graph to compute kernel functions based on metrics such as the number of common substructures (e.g., paths or trees) [30, 31] or graphlets [32, 33]. Neural language models such as word2vec [12] and GloVe [13], proposed initially for generating word embedding, have been adapted for KGs [11, 15, 34]. Deep Graph Kernel [33] identifies graph substructures (graphlets) and uses neural language models to compute a similarity matrix between identified substructures. For large scale KGs, embedding techniques based on random walks have been proposed in the literature. DeepWalk [35] learns graph embedding for nodes in the graph using neural language models while generating truncated uniform random walks. node2vec [15] is a more generic approach than DeepWalk and uses 2<sup>nd</sup> order biased random walks for generating graph embedding, preserving roles and community memberships of nodes. RDF2Vec [11], an extension of DeepWalk and Deep Graph Kernel, uses BFS-based random walks for extracting subgraphs from RDF graphs, which are converted into feature vectors using word2vec [12]. Random walk-based approaches such as RDF2Vec have been shown to outperform graph kernel-based approaches in terms of scalability and their suitability for ML tasks for large-scale KGs [11, 27]. The main limitation of approaches using uniform (or unbiased)

random walks is the lack of control over the explored neighborhood which can lead to inclusion of less relevant nodes in identified subgraphs of target entities. To address this challenge biased random walks based approaches have been recently proposed [3, 14, 20]. Such approaches use different weighting schemes for nodes and edges. The weights create the *bias* by making specific nodes or edges more likely to be visited during random walks. The work closest to ours is biased RDF2Vec [14] which uses frequency-, degree-, and PageRank-based metrics for weighting schemes. Our proposed approach resembles this approach in terms of extracting entity representations and using neural language model for generating embeddings. Unlike [14], we use our proposed metric of *specificity* as an edge- and path-weighting scheme for biased random walks for identifying most relevant substructures for extracting entity representations from KGs.

Semantic similarity and relatedness between two entities have been relatively well explored [23, 36–38]. Searching for similar or related entities given a search query is a common task in the field of Information Retrieval. To facilitate the search for *similar* entities the notion of similarity and the set of attributes used for its computation must first be defined. Semantic similarity and relatedness are often used interchangeably in literature [23, 36, 39], where the similarity between two entities is sometimes computed based on common paths between them. This definition allows computation of similarity between any two given entities, including entities of different types. For example, *Kobe Bryant* and *Kareem Abdul-Jabbar* (athletes) are each *related* to *LA Lakers* (team) based on path-based similarity. The other kind of similarity is between *Kobe Bryant* and *Kareem Abdul-Jabbar* who are entities of the same type, i.e., athletes. Both are athletes, basketball players, and have played for the same team. These attributes in KG, e.g. DBpedia, are represented through semantic relationships *rdf:type* and *dct:subject*. For this paper, our objective is to automatically identify such semantic relationships that constitute the representative neighborhoods of entities of the same given type. Therefore, we limit the computation of similarity to be between two entities of the same type.

<sup>4</sup>[https://www.wikidata.org/wiki/Wikidata:Main\\_Page](https://www.wikidata.org/wiki/Wikidata:Main_Page)

<sup>5</sup><https://developers.google.com/freebase/>

<sup>6</sup><https://wiki.dbpedia.org/>

### 3. Preliminaries

An RDF graph is represented by a knowledge base of triples [40]. A triple consists of three parts:  $\langle \text{subject } (s), \text{predicate } (p), \text{object } (o) \rangle$ .

**Definition 1. RDF Graphs:** Assuming that there is a set  $U$  of Uniform Resource Identifiers (URIs), a set  $B$  of blank nodes, a set  $L$  of literals, a set  $O$  of object properties, and a set  $D$  of datatype properties, an RDF graph  $G$  can be represented as a set of triples such that:

$$G = \{ \langle s, p, o \rangle \mid s \in (U \cup B), p \in (D \cup O), (o \in (U \cup B), \text{ if } p \in O) \wedge (o \in L, \text{ if } p \in D), ((D \cup O) \subseteq U) \} \quad (1)$$

We can also represent an RDF graph  $G$  as  $G = \{V, E\}$  such that  $V \in (U \cup B \cup L)$  and  $E \in (O \cup D)$ , where  $E$  is a set of directed labeled edges. In an RDF graph, URIs are used as location-independent addresses of entities (both nodes and properties), whereas blank nodes are assigned internal IDs. Literals can have any values conforming to the XML schema definitions. Thus, (1) signifies that any entity with a URI can be subject, predicate or object. In practice, only entities with URIs defined as Object or Data type properties are used predicates. However, these properties can also be subjects or objects in other triples, which is an interesting feature of an RDF graph that an edge can be between two edges which makes it different from traditional definition of a graph. Blank nodes can either be subject or object whereas literals can only be objects. An RDF graph is a set of all such RDF triples [40, 41]. Ontologies are a key concept in the domain of Semantic Web. An ontology is a formal conceptualization of a particular domain containing a hierarchy of concepts (or classes), their relationships (object properties) and attributes (data properties). Such semantic relationships are the fundamental aspect of knowledge representation in Semantic Web as they provide information about how entities are linked together. Thus, the ontologies are used to enforce a schema over RDF instance data [24].

**Definition 2. Semantic Relationship:** A semantic relationship in an RDF graph can be defined as  $\langle s, p^d, o \rangle$  where  $p^d$  can be a single predicate or a path represented by successive predicates and intermediate nodes between  $s$  and  $o$ .

For this paper, we define semantic relationship  $p^d$  of depth or length  $d$ , as a template for a path (or a walk) in  $G$ , that comprises of  $d$  successive predicates  $p_1, p_2, \dots, p_d$ . Thus,  $\langle s, p^d, o \rangle$  represents all paths (or walks) between any two entities  $s$  and  $o$  that traverse through  $d - 1$  intermediate nodes, using the same  $d$  successive predicates that constitute  $p^d$ .

**Definition 3. Graph Walk:** Given a graph  $G = \{V, E\}$ , a single graph walk of depth  $d$  starting from a node  $v_0 \in V$  comprises a sequence of  $d$  edges (predicates) and  $d + 1$  nodes:  $v_0 \xrightarrow{e_1} v_1 \xrightarrow{e_2} v_2 \xrightarrow{e_3} \dots \xrightarrow{e_d} v_d$ .

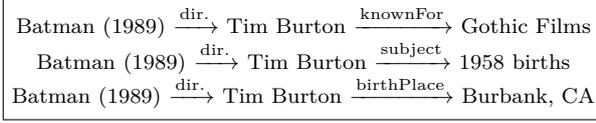
Random graph walks provide a scalable method of extracting entity representations from large scale KGs [15]. Starting from a node  $v_0 \in V$ , in the first iteration, a set of randomly selected outgoing edges  $E_1$  is explored to get a set of nodes  $V_1$  at depth 1. In the second iteration, from every  $v \in V_1$ , outgoing edges are randomly selected for exploring next set of nodes at depth 2. This is repeated until a set of nodes at depth  $d$  is explored. The generated random walks are the union of explored triples during each of the  $d$  iterations. This simple scheme of random walks resembles a randomized breadth-first search. In literature, both breadth-first and depth-first search strategies and interpolation between the two have been proposed for extracting entity representations from large-scale KGs [11, 15, 35].

### 4. Specificity: An Intuitive Relevance Metric

In this section, we introduce and motivate the use of *specificity* as a novel metric for quantifying relevance.

#### 4.1. Specificity

Consider the example shown in Figure 2. Starting from the entity *Batman (1989)* in DBpedia, a random walk explores the three shown semantic relationships (descriptive names used for brevity instead of actual DBpedia URIs). Our intuition suggests that the style of a director (represented by *Gothic Films*) is more relevant to a film than his year and place of birth. Frequency-, degree-, or PageRank-based metrics of assigning relevance may assign higher scores to nodes represent-

Fig. 2. Random walks from node *Batman (1989)*

ing broader categories or locations. For example, PageRank scores (non-normalized) computed for DBpedia entities *Gothic Films*, *1958-births*, and *Burbank, CA* are 0.586402, 161.258, and 57.1176 respectively (calculated based on [16]). PageRank-based biased random walks may include these popular nodes and exclude intuitively more relevant information related to the target entity. Our objective is to develop a metric that assigns a higher score to more relevant nodes and edges in such a way that the node *Gothic Films* becomes more likely to be captured for *Batman (1989)* than *1958 births* and *Burbank, CA*. This way, the proposed metric captures our intuition behind identifying more relevant information in terms of its specificity to the target entity.

To quantify this relevance we determine if *Gothic Films* represents information that is *specific* to *Batman (1989)*. We trace all paths of depth  $d$  reaching *Gothic Films* and compute the ratio of the number of those paths that originate from *Batman (1989)* to the number of all traced paths. This gives specificity of *Gothic Films* to *Batman (1989)* as a score between 0.0-1.0. A specificity score of 1.0 means that all paths of depth  $d$  reaching *Gothic Films* have originated from *Batman (1989)*. For  $G = \{V, E\}$ , this node-to-node specificity of a node  $n_1$  to  $n_2$ , such that  $n_1 \in V$ ,  $n_2 \in V$  and  $p^d$  being any arbitrary path, can be defined as:

$$\text{Specificity}(n_1, n_2) = \frac{|\langle n_2, p^d, n_1 \rangle \in G|}{|\langle v, p^d, n_1 \rangle \in G : v \in V|} \quad (2)$$

Computing specificity between every pair of nodes in a large scale KG is impractical. Instead of defining specificity as a metric of relevance between each pair of entities (or nodes) we make two simplifying assumptions. First, we assert that each class or type of entities (e.g., films, books, athletes, politicians) has a distinct set of characteristic semantic relationships. This enables us to compute specificity as a metric of relevance of a node

(*Gothic Films*) to a class or type of entities (*Film*), instead of every instance of that class (*Batman (1989)*). Second, we measure the specificity of a semantic relationship (*director, knownFor*), instead of an entity (*Gothic Films*), to the class of target entities. Here, we are assuming that if the majority of the entities (nodes) reachable via a given semantic relationship represents entity-specific information, we consider that semantic relationship to be highly *specific* to the given class of target entities. From our example, this means that instead of measuring specificity of *Gothic Films* to *Batman (1989)*, we measure specificity of semantic relationship *director, knownFor* to the class or entity type *Film*. Based on these assumptions we redefine specificity as

**Definition 4. Specificity:** Given an RDF graph  $G = \{V, E\}$ , a semantic relationship  $p^d$  of depth  $d$ , and a set  $S \subseteq V$  of all entities  $s$  of type  $t$ , let  $V_{S, p^d} \subseteq V$  be the set of all nodes reachable from  $S$  via  $p^d$ . We define the specificity of  $p^d$  to  $S$  as

$$\text{Specificity}(p^d, S) = \frac{1}{|V_{S, p^d}|} \sum_{k \in V_{S, p^d}} \frac{|\langle s, q^d, k \rangle \in G : s \in S|}{|\langle v, q^d, k \rangle \in G : v \in V|} \quad (3)$$

where  $q^d$  represents any arbitrary semantic relationship of length  $d$ . Figure 3 provides visual representation of Equation 3.  $S$  is a set of all nodes with a given type  $t$  and  $S' = V - S$ .  $S$  and  $V_{S, p^d}$  are shown as disjoint sets only for illustrative purposes.  $S \cap V_{S, p^d} \neq \emptyset$  when  $p^d$  creates a loop or when for  $d = 1$  is a self-property.

We also present an extension of specificity which takes into account the class hierarchy of the schema ontology of the KG for its computation. In Equation 3, the numerator only counts those semantic relationships that originate from  $s \in S$ . This definition is rigid because a given semantic relationship can be specific to multiple classes of entities. In other words, certain semantic relationships can be applicable to a broader class and by extension to multiple of its subclasses. When computing specificity of a given semantic relationship to instances of one of these subclasses e.g., *Film*, the specificity score may get penalized due to relevance of the semantic relationships to the other subclasses such as *TelevisionShow*. To rectify this and make the specificity computations more

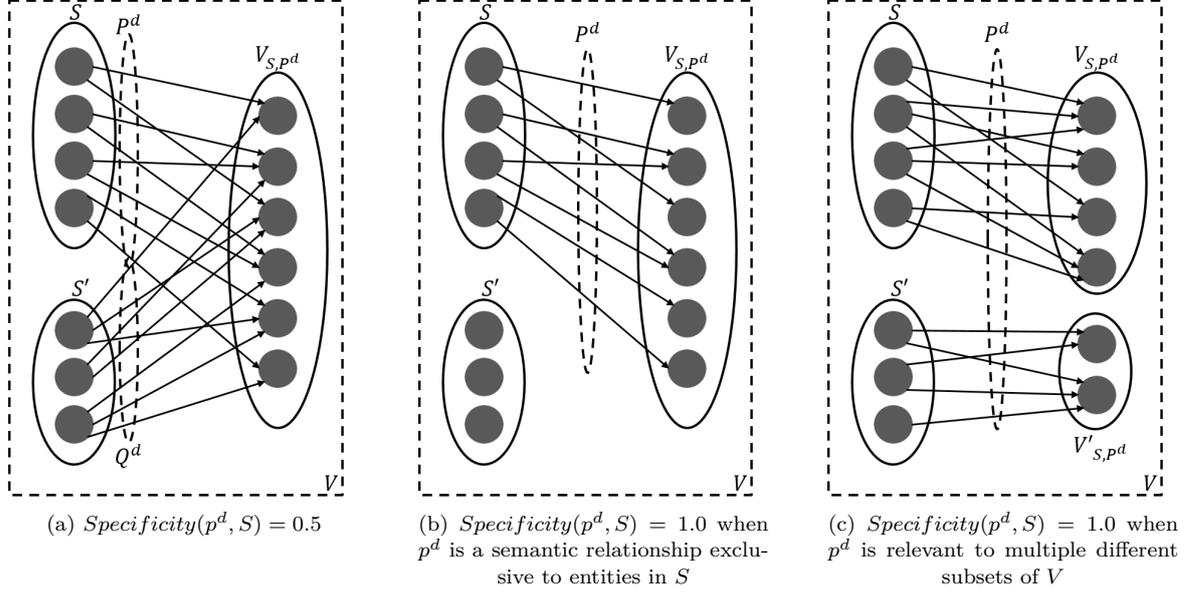


Fig. 3. Illustration of specificity

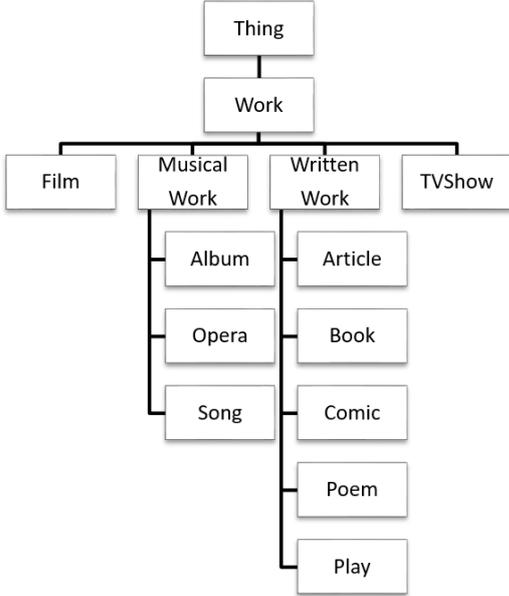


Fig. 4. A subset of DBpedia class hierarchy

flexible, we leverage the hierarchical relationships represented by the schema ontology. One of the most common relationships in an ontology is hyponymy, also called is-a relationship [42, 43]. Hyponymy represents the relationship between hypernym (broader category) and hyponyms (more specific categories). Figure 4 shows a subset of the

hierarchical structure of DBpedia ontology. *Film*, *MusicalWork*, *WrittenWork* are hyponyms to the hypernym *Work* and co-hyponyms of each other. Assume that the class hierarchy in the schema ontology of the KG is structured as an n-ary tree i.e., a class can have more than one subclasses but not more than one superclass. Moreover, there is only one class that has no superclass which is the root of the entire class hierarchy.

Let the given entity type  $t$  be at height (or depth)  $h$  from the root such that  $t$  and root can be labeled as  $C_0$  and  $C_h$  respectively, which results in a hierarchy of classes  $C_0 \subseteq C_1 \subseteq \dots \subseteq C_h$ . Subclasses of  $C_1$  are co-hyponyms of  $C_0$ . Our objective is to include the instances of hypernyms of entity type  $t$  ( $C_0$ ) in the computation of specificity. In Equation 3, we only consider  $s \in S$ , all of which are of type  $t$  (or instances of class  $t$ ). We add a term to the numerator to include instances of hypernyms of  $t$  in the computation of specificity as follows:

$$\begin{aligned}
 & Specificity(p^d, S) \\
 &= \frac{1}{|V_{S,p^d}|} \sum_{k \in V_{S,p^d}} \frac{1}{|\langle v, q^d, k \rangle \in G : \forall v \in V|} \\
 & \left( \sum_{j=0}^h \beta^j |\langle v_j, q^d, k \rangle \in G : \forall v_j \in V \wedge type(v_j) = C_j| \right)
 \end{aligned} \tag{4}$$

Additional constraints for Equation 4 are:  $v_0 \in S$  and  $v_1 \cap v_2 \cap \dots \cap v_h \cap S = \emptyset$ . This is to ensure that each instance of type  $t$  or any of its superclasses is only considered once in the computation. The factor  $\beta$  ensures diminishing influence of broader concepts on the computation of specificity for more specific concepts. For example, from Figure 4, when computing specificity of a given semantic relationship for class *Film* other instances of class *Work* that are not films have a higher influence than the more broader class *Thing*.

## 5. Bidirectional Random Walks for Computing Specificity

Computing Equation 4 requires accessing large parts of the knowledge graph. In this section, we present an approach that uses bidirectional random walks to compute specificity. To understand, consider an entity type  $t$  and a semantic relationship  $p^d$ , for which we want to compute  $Specificity(p^d, t)$ . We start with a set  $S$  containing a small number of randomly selected nodes of type  $t$ . From nodes in  $S$ , forward random walks via  $p^d$  are performed to collect a set of nodes  $V_{S,p^d}$  (ignoring intermediate nodes, for  $d > 1$ ). From nodes in set  $V_{S,p^d}$ , reverse random walks in  $G$  (or forward random walks in  $G^r = reverse(G)$ ) are performed using arbitrary paths of length  $d$  to determine the probability of reaching any node of type  $t$ . Specificity is computed as the number of times a reverse walk lands on a node of type  $t$  divided by the total number of walks. This idea is the basis for the algorithm presented next which builds a list of most relevant semantic relationships up to depth  $d$  sorted by their specificity to a given entity type  $t$ .

Specifically,  $Q_{paths}$  and  $Q_{spec}[]$  hold the set of semantic relationships, unsorted and sorted by specificity respectively.  $Q_{spec}[]$  is initialized as an array of size  $d$  to hold sorted semantic relationships for every depth up to  $d$ .  $N_{paths}$  specify the size of  $Q_{paths}$ .  $N_{walks}$  is the number of bidirectional walks performed for computing specificity for each semantic relationship in  $Q_{paths}$ . A set  $S$  of randomly selected nodes of type  $t$  is generated in line 3. For each  $i^{th}$  iteration ( $i \leq d$ ), a set of semantic relationships  $Q_{paths}$  is selected in line 5. The function  $computeSpecificity$ , in line 6, computes specificity for each semantic relationship in

---

### Algorithm 1 $rankBySpecificity(G, d, t)$

---

**Input:** RDF graph  $G = \{V, E\}$ ,  $d$  is the maximum depth of semantic relationships to be considered, originating from entities of type  $t$ .  
**Output:** Returns ranked list  $Q_{spec}[]$  of semantic relationships for depths  $\leq d$ , with a score of 0.0 – 1.0

- 1: initialize  $Q_{paths}, Q_{spec}[]$  to null/empty
- 2: initialize  $N_{paths}, N_{walks}, \beta$
- 3:  $S \leftarrow$  Generate random nodes of type  $t$
- 4: **for**  $i \leftarrow 1, d$  **do**
- 5:      $Q_{paths} \leftarrow selectPaths(G, S, i, N_{paths})$
- 6:      $Q_{spec}[i] \leftarrow computeSpecificity(G, Q_{paths}, S, t, i, N_{walks}, \beta)$
- 7: **end for**
- 8: **return**  $Q_{spec}[]$

---

$Q_{paths}$  and returns results in  $Q_{spec}[i]$ . Each element of  $Q_{spec}$  is an array of dictionaries. Each dictionary contains *key–value* pairs sorted by *value*, where *key* is the semantic relationship and *value* is its specificity. For each  $i^{th}$  iteration of *for* in Algorithm 1,  $Q_{paths}$  can be populated from scratch with semantic relationships of depth  $i$  by random sampling of outgoing paths from  $S$ . Alternatively, for iterations  $i \geq 2$ ,  $Q_{paths}$  can be populated by expanding from most specific semantic relationships in  $Q_{spec}[i - 1]$ .

Algorithm 2 shows the function  $computeSpecificity$  which computes specificity for a given set of semantic relationships in  $Q$  ( $Q_{paths}$  from Algorithm 1). In lines 7 and 8, for each semantic relationship  $q \in Q$ , a node  $s \in S$  is randomly selected to get a node  $v$  reachable from  $s$  via  $q$  in  $G$  (*forward walk*). In line 9, by using  $v$  and selecting an arbitrary path  $q'$  of depth  $d$ , a node  $v'$  is reached (*reverse walk*, reverse walk in  $G$  or forward walk in  $G^r$ ). If implementing specificity based on Equation 3, the algorithm needs to check if  $t$  is one of the types associated with  $v'$  and simply increment an integer variable *count* [20]. To implement specificity based on Equation 4, we first need to acquire the class hierarchy of entity type  $t$  in  $H$  in line 2. The first index of  $H$  holds  $t$  and every subsequent entry  $H[i]$  for  $i > 1$  holds the hypernym or superclass of entry  $H[i - 1]$ . For example, for entity type *Film* (Figure 4),  $H = [“Film”, “Work”, “Thing”]$ . Starting from the first element in  $H$  at index  $i = 1$ , the existence of triple  $\langle v', rdf : type, H[i] \rangle$  is checked in  $G$ . If such a triple exists, count is incremented by a factor of  $\beta^{i-1}$ . This process of bidi-

**Algorithm 2** *computeSpecificity*( $G, Q, S, t, d, N, \beta$ )

---

**Input:** RDF graph  $G = \{V, E\}$ ,  $S$  is a set of random entities with a common type  $t$ ,  $Q$  is a set of semantic relationships of length  $d$  to be processed,  $N$  is number of bidirectional walks to be performed for each semantic relationship in  $Q$

**Output:** Returns list  $L$  of semantic relationships sorted by specificity (0.0 – 1.0)

- 1:  $G^r = reverseEdges(G)$
- 2:  $H = hierarchy(t)$
- 3: initialize *dictionary*  $L$
- 4: **for all**  $q \in Q$  **do**
- 5:      $count \leftarrow 0.0$
- 6:     **repeat**
- 7:          $s \leftarrow$  randomly pick a node from  $S$
- 8:          $v \leftarrow$  randomly explore node from  $s$  using any path  $q$  in  $G$
- 9:          $v' \leftarrow$  randomly explore node from  $v$  using any path  $q'$  in  $G^r$
- 10:         **for**  $i \leftarrow 1, size(H)$  **do**
- 11:             **if**  $\exists < v', rdf : type, H[i] > \in G$  **then**
- 12:                  $count \leftarrow count + \beta^{i-1}$
- 13:             **break**
- 14:         **end if**
- 15:         **end for**
- 16:     **until**  $N$  times
- 17:     insert  $(q, \frac{count}{N})$  in  $L$
- 18: **end for**
- 19: **return**  $L$

---

rectional walks is repeated  $N$  times for each  $q$ . At line 17, specificity is computed as  $\frac{count}{N}$ . Lines 4-18 are repeated until specificity for each  $q \in Q$  has been computed. The return variable  $L$  contains semantic relationships and their specificity scores as *key – value* pairs.

## 6. Evaluation

We evaluate our approach in five different ways. First, we analyze the behavior of specificity computed for most relevant semantic relationships up to depth 3. Second, we study the sensitivity of specificity to the parameter  $N_{walks}$  (number of bidirectional walks) and  $|S|$  (seed set size). Third, we evaluate the compactness of the extracted subgraphs by specificity-based biased random walk scheme against few baselines. Fourth, we generate embeddings from the extracted subgraphs to perform a few tasks, i.e., entity recommendation, regression, and classification. Finally, we analyze the ability of generated embeddings in preserving

the semantics associated with the entities from the KG.

### 6.1. Datasets

We evaluated our approach on DBpedia which is one of the largest RDF repositories publicly available [44]. We used the English version of DBpedia dataset from 2016-04 (<http://wiki.dbpedia.org/dbpedia-version-2016-04>). We created graph embeddings for 5000 entities each of types: *Film*, *Book*, and *Album*. We also generated embeddings for 500 and 3000 entities of type *Country* and *City* respectively. We use three different datasets from different domains for the tasks of classification and regression. The datasets provide classification and regression targets for given DBpedia entities.

The *Mercer Cities*<sup>7</sup> dataset contains a list of cities and their quality of living as numeric ratings and discrete labels. The *Metacritic Movies*<sup>8</sup> and *Metacritic Music Albums*<sup>9</sup> datasets contain the average ratings of all time reviews for a list of movies and albums respectively. More details of these datasets can be found in [11] and they are accessible at [http://data.dws.informatik.uni-mannheim.de/rmlod/LOD\\_ML\\_Datasets/data/datasets/](http://data.dws.informatik.uni-mannheim.de/rmlod/LOD_ML_Datasets/data/datasets/).

### 6.2. Experimental Setup

The first step is to compute *specificity* and *specificity<sup>H</sup>* to find the set of most relevant semantic relationships for entities of selected types. Unless otherwise specified we used following values for the parameters of the Algorithms 1 and 2:  $N_{walks} = 8000$ ,  $|S| = 1000$ ,  $\beta = 0.25$ . Starting from a random seed set  $S$  of each type of entity, we randomly sample semantic relationships originating from entities in  $S$  and select *top – 50d* semantic relationships based on the frequency of occurrence for each depth  $d$ . After specificity has been computed for each semantic relationship, we only consider those as relevant that have specificity scores greater than 50%. These *specific* semantic relationships are used for subgraph extrac-

---

<sup>7</sup><https://mobilityexchange.mercer.com/Insights/quality-of-living-rankings>

<sup>8</sup><http://www.metacritic.com/browse/movies/score/metacore/all>

<sup>9</sup><http://www.metacritic.com/browse/albums/score/metacore/all>

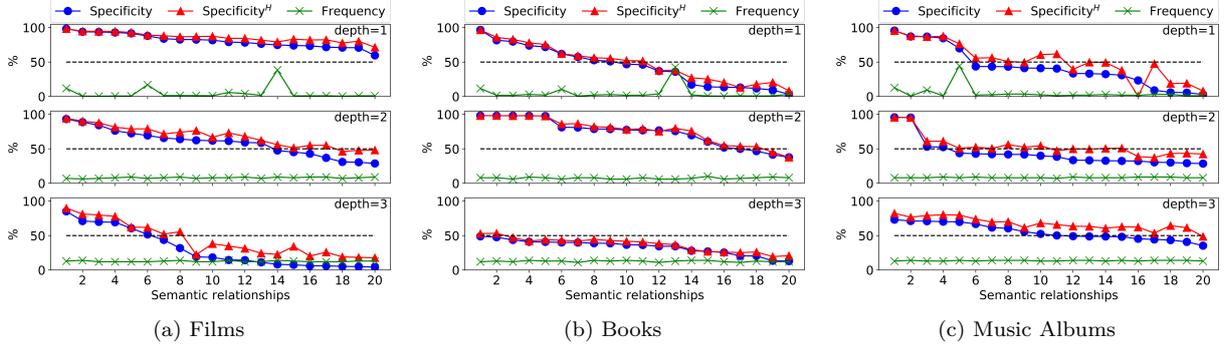


Fig. 5. Comparison of frequency- and specificity-based metrics for top-20 semantic relationships

tion using biased random walks for subgraph extraction. Due to this reason, we only extracted relevant subgraphs up to depth = 3 because none of the semantic relationships of depth (or length) equal to 4 had a specificity score > 50% for any of the selected entity-types.

We have only included select plots in the paper to support the discussion. Complete experimental results are available at <https://github.com/mrizzwansaeed/Specificity>.

### 6.3. Specificity as a Metric for Measuring Relevance

Figure 5 shows the top 20 semantic relationships for entities of three types sorted by their *specificity* up to depth 3. The frequency represents the % of number of times a path representing a particular semantic relationship is traversed when the neighborhoods of randomly selected nodes are explored. For depth 1, the top-most plot of each subplot in Figure 5 shows that there are only a few relatively high-frequency semantic relationships (represented by the peaks) whereas the rest show a uniform trend of frequency. As depth  $d$  increases, frequency exhibits a flattened trend due to a rapid increase in the number of possible semantic relationships at each depth. This trend makes it difficult to define a frequency-based cut-off value for choosing a certain set of semantic relationships as most relevant. This may require manual examination of the set of semantic relationships for choosing an appropriate frequency threshold such that most relevant semantic relationships. Alternatively, we can instead choose a value  $k$  such that *top* -  $k$  high-frequency semantic relationships are selected as

Table 1

Top semantic relationships based on frequency with corresponding specificity scores for *dbo:Film*

Semantic Relationship	Freq. (%)	Spec. (%)	Spec. <sup>H</sup> (%)
rdf:type	38.46	74.69	79.15
dct:subject	16.94	87.9	89.09
owl:sameAs	11.88	98.4	98.4
dbo:starring	5.82	79.06	84.43
dbo:writer	1.68	76.48	81.86
dbo:producer	1.34	83.9	88.22
dbo:director	1.08	81.76	87.42
dbo:musicComposer	0.92	70.77	80.28
dbo:distributor	0.84	74.08	83.22
dbo:language	0.72	38.65	53.42
dbo:editing	0.66	91.65	92.82
dbo:cinematography	0.64	92.99	94.49

the most relevant. For example, in Table 1, assuming that we wish to include the intuitively relevant semantic relationship *dbo:director* in the selected semantic relationships, we can either choose a frequency threshold of < 1.08 or choose  $k \geq 7$ . However, this ad-hoc method of selecting thresholds is impractical since it has to be done for every different class of entities, for every depth, and for every different RDF KG.

The threshold of specificity is drawn at 50% in all plots in Figure 5. The specificity of a semantic relationship is the probability of reaching any node of a given type from a set of nodes ( $V_{S,p^d}$  in Definition 4) by reverse walks in  $G$  (or forward walks in  $G^r$ ) using any arbitrary path of length  $d$ . We can define a universal cut-off for specificity at 50%. Specificity score above this threshold means

that the selected semantic relationship links the instances of given class (or entity type)  $t$  to the set  $V_{S,p^d}$  such that more than half the incoming edges to this set originate from instances of class  $t$ . This means that the information represented by nodes in  $V_{S,p^d}$  on average is more *specific* to  $t$ .

In literature, other approaches such as [23] have employed a decaying factor  $\alpha^d$  (where  $\alpha \in [0.0 - 1.0]$ ) as a function of depth  $d$  that is used to equally penalize the relevance score of all nodes at depth  $d$  from target nodes. This is done to implement the idea that the relevance of nodes decreases as we move farther away from the target nodes. Specificity, on the other hand, has a more fine-grained mechanism of assigning relevance score across different  $d$ 's. Figure 5 shows that there are multiple instances of semantic relationships at depth  $d$  that have higher specificity than semantic relationships at depth  $< d$ . For example, Figure 5c shows that the top-2 semantic relationships at depth=2 have a specificity score of 100% which is higher than all but one of the top-20 semantic relationships at depth=1. This way specificity exhibits a more fine-grained behavior of relevance across depths, meaning that all semantic relationships do not simultaneously become less relevant as we move farther away from target nodes. There are variations in the specificity-based relevance scores of semantic relationships at the same depth as well as across depths. This allows both shallow (breadth-first) and deep (depth-first) exploration of the relevant neighborhoods around target entities by specificity-based biased random walks.

Table 2

Comparison of relevance metrics for example in Figure 2

Semantic Relationships	Spec.	PR	Freq.
director,knownFor	59.14	6.2	345
director,subject	1.05	823.53	73752
director,birthPlace	0.03	200.33	7087

Table 2 shows computed relevance of the three semantic relationships from example in Figure 2 (Section 4.1) based on their specificity, PageRank, and frequency. The given PageRank values in column 3 are the average of non-normalized PageRank scores [16] of top-25 nodes linked to DBpedia entities of type *Film* by corresponding semantic relationship. The values of frequency in the last column represent the number of occurrences of corresponding semantic relationship in DBpe-

dia dataset. We argued that the semantic relationship *director,knownFor* is more relevant to a film as compared to the other two. Table 2 shows that the proposed specificity based relevance metric is closer to our intuition as compared to other metrics.

### 6.3.1. Specificity<sup>H</sup>

As discussed in Section 4.1 that *specificity* only includes those nodes in its computation that are dominantly exclusive or specific to nodes of a given type. However, many types of entities require both specific as well as generic nodes and relationships for complete characterization. For example, assume that a node representing *language* is associated with all instances of the class *Work* and its subclasses (Figure 4). Since nodes representing languages can be linked to multiple different entity types, therefore the specificity of the semantic relationship that links language-related nodes will be low. In other words, if the algorithm performs reverse walks from *dbo:English*, it can potentially land on multiple different types of nodes (e.g., films, books, songs, plays, games). Table 1 shows that the DBpedia property *dbo:language* has a specificity score below the threshold resulting in the exclusion of language-based information from the representative neighborhoods of instances of *dbo : Film*. *Specificity<sup>H</sup>* computed using Algorithm 2 takes into account the general applicability of language-based nodes to co-hyponyms and hypernyms of *dbo:Film*, resulting in increasing the specificity score. This is evident in all plots in Figure 5 where the curve of *specificity<sup>H</sup>* mostly lie on or above the corresponding curve of *specificity*.

## 6.4. Parameter Sensitivity

The algorithms for computing specificity use bidirectional random walks, governed by two parameters: number of bidirectional walks ( $N_{walks}$ ) and size of seed set  $S$ . In this section, we evaluate the sensitivity of specificity to both of these parameters.

### 6.4.1. Sensitivity of Specificity to $N_{walks}$

The algorithm for computing specificity uses the parameter  $N_{walks}$  for controlling number of bidirectional walks. To understand the effect of this parameter on specificity, we first computed specificity for  $N_{walks} \in [2500, 20000]$ . Figures 6a and 6b show the comparison between specificity scores

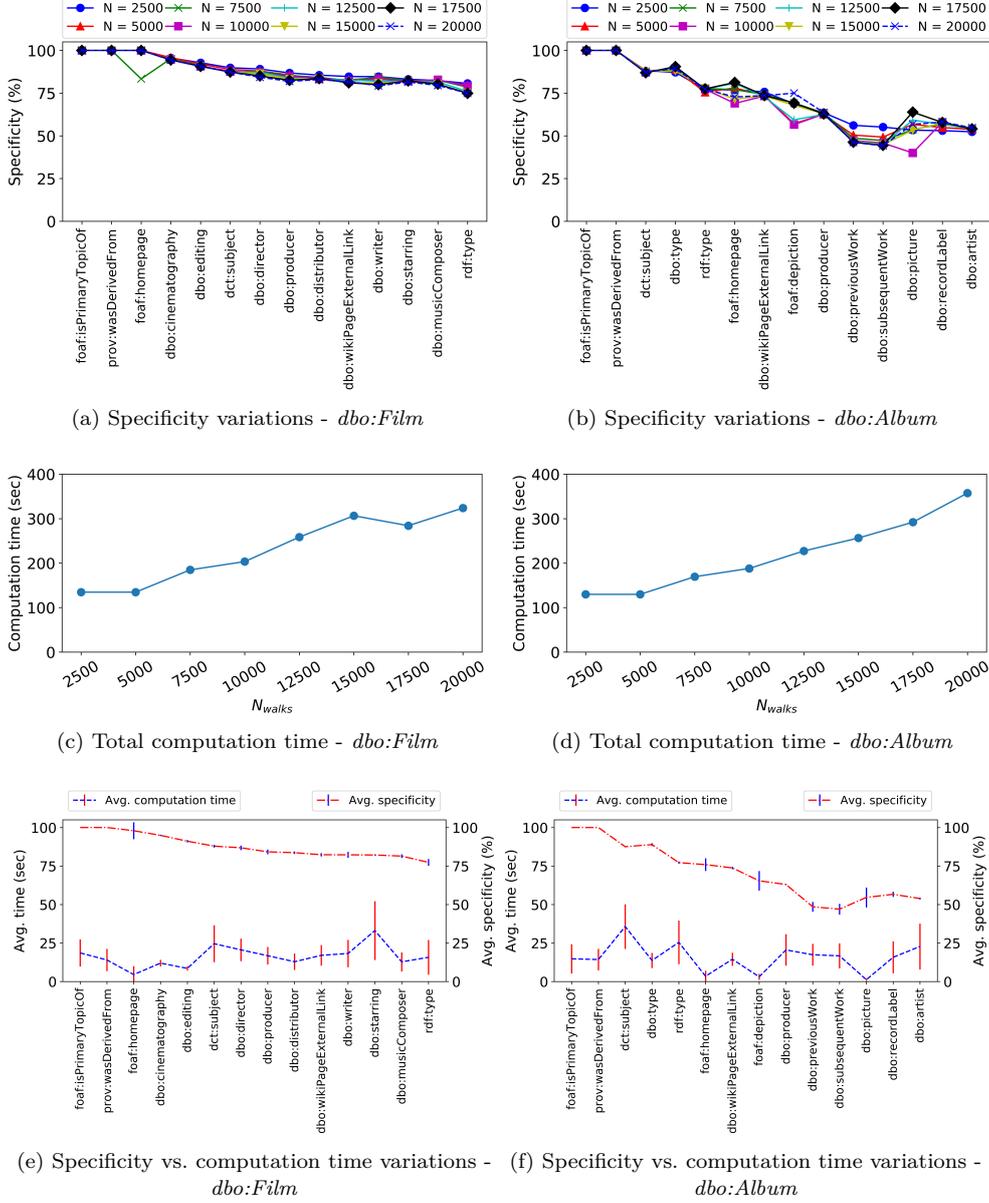
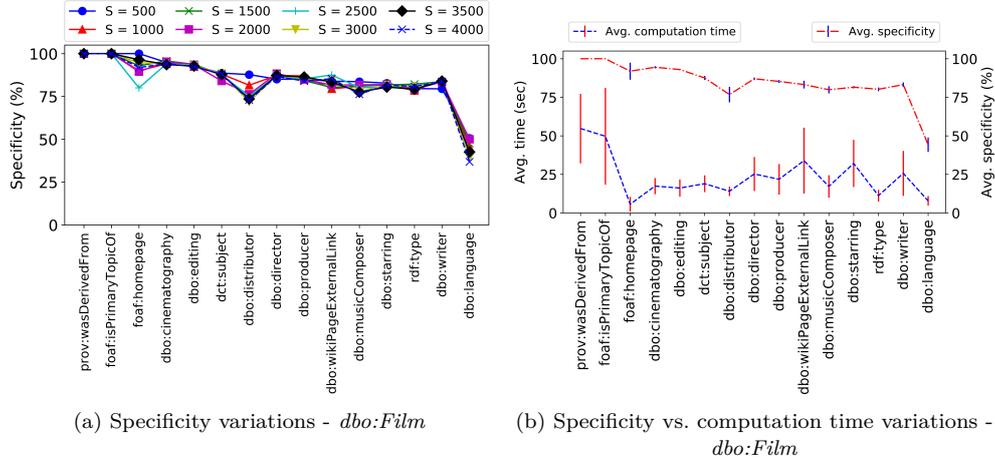


Fig. 6. Effect of  $N_{walks}$  on computation of specificity

computed for different values of  $N_{walks}$ . The semantic relationships are sorted on the x-axis (from left to right) in order of decreasing specificity compute based on  $N_{walks} = 2500$  for comparison. Figures 6c and 6d show the total time taken to compute specificity for the shown semantic relationships for *dbo:Film* and *dbo:Album* respectively. The computation time increases with increasing number of bidirectional walks. Figures 6e and 6f

show the average specificity and corresponding computation time for each semantic relationship for all values of  $N_{walks}$ . We use standard deviation to show the variations in the specificity scores and computation times observed for different values of  $N_{walks}$ . It can be observed that (e.g. for *dbo : starring* in Figure 6e and *dct : subject* in 6f) large variations in computation time do not lead to major changes in the specificity scores. Choos-

Fig. 7. Effect of  $|S|$  on computation of specificity

ing a larger value of  $N_{walks}$  will simply increase the total computation time without having significant effect on the computed specificity scores. If we select  $N_{walks} = 8000$  as a suitable value for computing specificity, it can be seen from Figures 6c and 6d that the time for computing specificity scores w.r.t either entity-type is  $\approx 200s$ . This computation is only needed to be performed once for each type of target entities in a KG i.e., once for *dbo:Film*, once for *dbo:Book*, and so on.

#### 6.4.2. Sensitivity of Specificity to $|S|$

Similar to previous discussion for  $N_{walks}$ , Figure 7b also shows that the large variations in computation time do not lead to major change in the specificity. Choosing a larger value of  $|S|$  increases the total computation time without having a significant effect on the computed specificity scores. If we chose  $|S| \approx 1000$  as a suitable value for computing specificity, it will result in computation time of  $\approx 200s$  for computing specificity of all of the shown semantic relationships.

#### 6.5. Comparison of Size of Extracted Subgraphs

For baselines, we have implemented uniform and biased random walk schemes for subgraph extraction [11, 14]. We have used the DBpedia PageRank [16] dataset available at [http://people.aifb.kit.edu/ath/#DBpedia\\_PageRank](http://people.aifb.kit.edu/ath/#DBpedia_PageRank) in our experiments. For the DBpedia dataset, we generated 500 walks (based on [11]) with depths 1, 2, and 3.

Figure 8 shows that specificity-based random walk schemes enable the extraction of representative subgraphs with fewer number of walks. Specificity-based schemes use extraction template based on semantic relationships with  $\geq 50\%$  specificity which enables collection of comparatively fewer but more relevant nodes and edges than the baselines for all three of the chosen entity-types.

#### 6.6. Suitability for Entity Recommendation task

We have shown that the specificity-based biased random walks extract more compact entity representations as compared to other schemes. However, to prove that the compactness of extracted

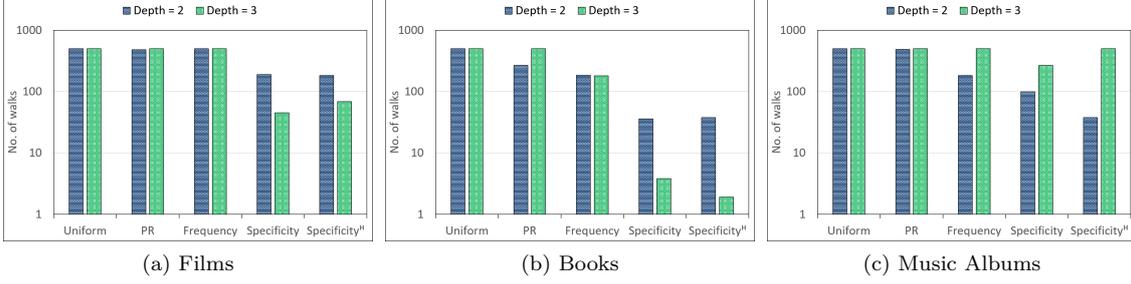


Fig. 8. Average number of walks per entity for relevant subgraph extraction

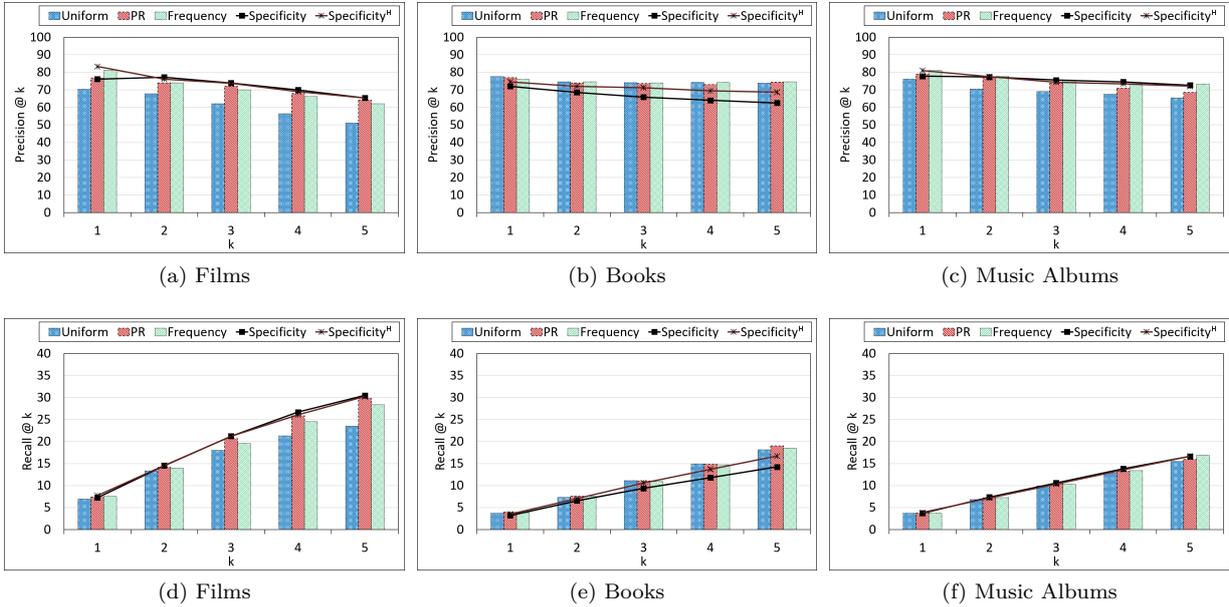


Fig. 9. Comparison for precision and recall for entity recommendation tasks

subgraphs is not a disadvantage, we use the graph embeddings generated from the extracted substructures for the task of entity recommendation. Using the extracted subgraphs (represented as sequence of labels), we trained Skip-gram models using the following parameters: dimensions of generated vectors = 500, window size = 10, negative samples = 25, iterations = 5 for each scheme and depth. All models for depth  $d > 1$  are trained using sequences generated for both depths 1 and  $d^{10}$ .

Given a vectorized entity as the search key, we list its top- $k$  most similar results. We use the metrics of precision@ $k$  and recall@ $k$  to quantify the

performance of the recommendation tasks. However, evaluating retrieved results requires ground truth. For music albums and books, the ground truth straightforwardly consists of other works by the same artists and authors respectively. For films, we selected franchises or series as ground truth. Films in a franchise/series usually share one or more common attributes (e.g., director, actors, genre, characters) and are more likely to be *similar* to each other. For example, for any of the three *The Lord of the Rings* (LOTR) films the other two films in the trilogy are its more likely top-2 similar results because of the same cast members, director, genre, and characters. In our experiments, *King Kong (2005)* also frequently appeared among the results similar to LOTR since it also has the

<sup>10</sup>The parameters for this experiment are based on RDF2Vec [11].

Table 3  
Results of Regression and Classification Tasks - *Specificity*<sup>H</sup>

Model	Depth	Classification			Regression		
		Cities	Films	Albums	Cities	Films	Albums
Uniform	1	1.0012	0.9783	1.0491	1.1783	0.9983	1.0125
	2	0.9795	0.9802	1.0299	0.9832	1.0024	1.0006
	3	1.0246	0.9967	0.9446	1.0251	1.002	0.9914
PageRank	1	1.0456	1.0063	1.0384	1.1629	0.9975	1.0154
	2	0.9608	0.9935	1.0161	0.9338	0.9994	0.9969
	3	0.9989	1.0531	1.0203	0.9757	1.0025	0.9947
Frequency	1	0.9945	1.0294	1.02	1.106	1.0049	1.0014
	2	1.0591	0.9936	0.9881	0.9627	1.0008	1.0008
	3	0.9376	1.0101	0.9642	1.0102	0.9987	0.9976
<i>Specificity</i>	1	1.0112	1.0137	1.07	1.1004	1.005	1.004
	2	1.0567	0.9663	0.9933	1.0157	0.9975	0.9996
	3	0.9597	1.0153	0.9365	1.0733	0.998	0.9952

same director. Other films may also occur among top-k results based on any number of other factors, e.g., cinematography, editing, distributor, all of which have high specificity to *dbo:Film* (from Table 1). Creating exhaustive lists of films for the ground truth to encompass all such scenarios is laborious. That is why we included only those films in the ground truth that are either in a franchise (e.g., all Batman movies) or are part of a series (e.g., prequels or sequels). Similarity among such films is relatively stronger and easier to interpret. Assuming that there are  $n$  entities in the ground truth for a given franchise, series, author, or artist (e.g.,  $n = 3$  for LOTR or  $n = 71$  for Agatha Christie in our DBpedia dataset), we chose one entity at a time to retrieve top-k similar entities (for  $k = 1$  to  $n - 1$ ), resulting in a total of  $n(n - 1)$  recommendation tasks per franchise, author, or music artist. We performed 92880, 236160, 187872 recommendation tasks for films, books, and music albums for each random walk scheme respectively.

Figure 9 shows the results of recommendation tasks over all depths for each scheme. It can be seen that specificity-based schemes has similar performance as compared to the baselines: slightly better in case of *dbo:Film* and *dbo:Album* but slightly lower for *dbo:Book*. However, if interpreted in conjunction with Figure 8, this difference in accuracy is achieved with embeddings generated from smaller extracted subgraphs using fewer walks than the baseline approaches.

### 6.7. Suitability for Regression and Classification Tasks

We performed the tasks of classification and regression on the *Mercer Cities*, *Metacritic Movies*, and *Metacritic Music Albums* datasets using the embeddings generated for DBpedia entities. We used SVM for classification and Logistic Regression for regression tasks. We measured *accuracy* for classification tasks and *root mean squared error* (RMSE) for regression tasks. The numeric values shown in Table 3 can be interpreted as factor of improvement (or *lift*) when *specificity*<sup>H</sup>-based embeddings are used compared to the baselines and *specificity*. The results show that the results of all schemes are comparable with no scheme consistently outperforming the others. However, these results are achieved using embeddings generated from smaller extracted subgraphs, proving that specificity-based embeddings can be suitable for data mining tasks.

### 6.8. Semantics of Specificity-based Vector Representations

To analyze the semantics of the vector representations, we employ Principal Component Analysis (PCA) to project the generated embeddings into a two-dimensional feature space. We selected seven countries (similar to evaluation done for RDF2Vec [11]) and their capital cities and visualized the vectors as shown in Figure 10. Figure 10d shows that specificity-based embeddings are capable of organizing entities of different types and preserving

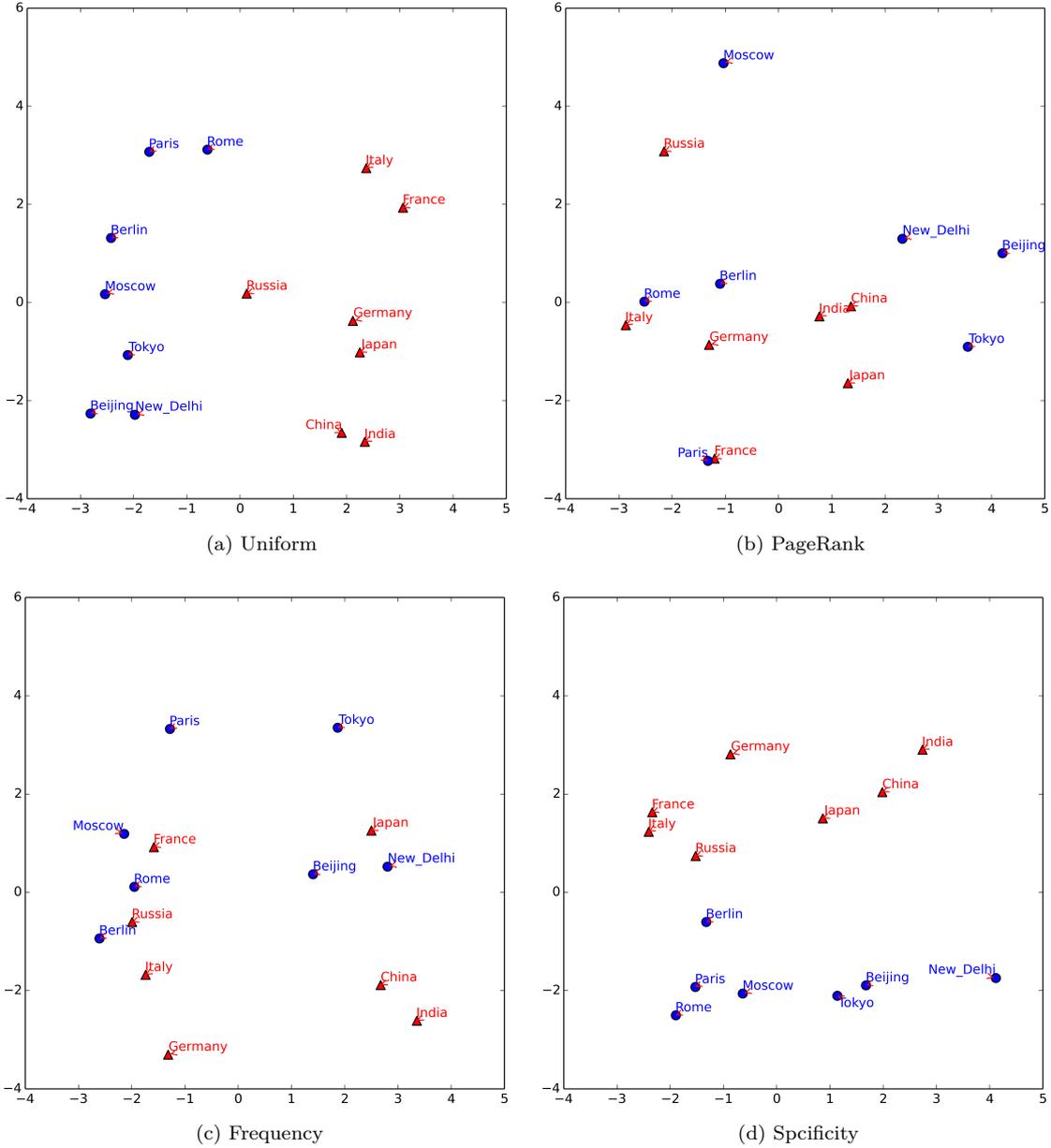


Fig. 10. Projection of countries and capitals in 2D space using embedding generated from RDF subgraphs of depth 2

the semantic associations between different entities. First of all, there is a clear separation between two different types of entities: *dbo:Country* and *dbo:City*, preserving the *rdf:type* relationship. Second, countries and their respective capitals are organized correspondingly, preserving the *explicit* relationship *dbo:capital*. Third, the *implicit* relationship *dbp:continent* is also captured as countries belonging to the same continents are positioned

closer to each other. The information regarding continents is represented via the semantic relationship *det:subject* which links each country to multiple DBpedia categories including those that represent continents, e.g. *dbc:Countries\_in\_Asia*. Another interesting observation is that, within the group of Asian countries, *Japan* is positioned closer towards the European countries. The reason could be that *Japan* shares many other DB-

pedia categories with them which it does not with *India* and *China* e.g., *dbc:G8\_nations* and *dbc:G20\_nations*.

Figures 10b and 10c show that i) *rdf:type* relationship is not preserved by frequency and PageRank-based embeddings. There is, although inconsistent, organization of countries and capitals with respect to each other as well as within groups based on their continents. Figure 10a shows that semantic associations similar to Figure 10d are also preserved using uniform random walks. However, Figure 11 shows that although the size of extracted subgraph for each entity (in terms of number of walks) is similar for countries but a ten-fold reduction can be observed for cities between the uniform and specificity-based random walk schemes.

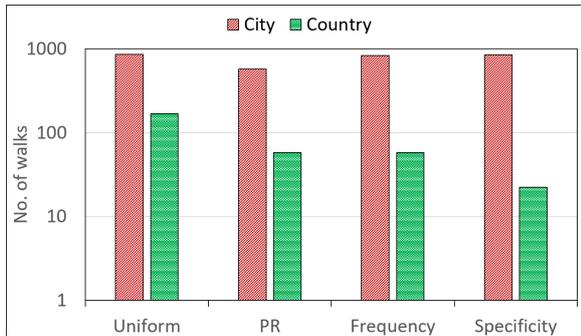


Fig. 11. Average number of walks per entity of type *dbo:City* and *dbo:Country* for relevant subgraph extraction

## 7. Conclusion

Graph embedding is an effective method of preparing KGs for AI and ML techniques. However, to generate appropriate representations, it is imperative to identify the most relevant representative subgraphs for target entities. In this paper, we presented specificity as a useful metric for finding the most relevant semantic relationships for target entities of a given type. Our bidirectional random walks-based approach for computing specificity is suitable for large-scale KGs of any structure and size. We have shown through experimental evaluation that the metric of specificity incorporates a fine-grained decaying behavior for semantic relationships. It has the inherent ability to interpolate between the extreme

exploration strategies: BFS and DFS. We used specificity-based biased random walks to extract compact representations of target entities for generating graph embedding. These generated representations have similar performance as compared to baseline approaches when used for our selected tasks of entity recommendation, regression, and classification. For future work, we intend to study other tasks for which specificity-based graph embedding can be used such as KG completion.

## Acknowledgment

This work is supported by Chevron Corp. under the joint project, Center for Interactive Smart Oilfield Technologies (CiSoft), at the University of Southern California.

## References

- [1] T. Berners-Lee, J. Hendler and O. Lassila, The semantic web, *Scientific american* **284**(5) (2001), 34–43.
- [2] C. Bizer, T. Heath and T. Berners-Lee, Linked data—the story so far, *International journal on semantic web and information systems* **5**(3) (2009), 1–22.
- [3] M. Atzori and A. Dessi, Ranking DBpedia Properties, in: *2014 IEEE 23rd International WETICE Conference, WETICE 2014*, 2014, pp. 441–446.
- [4] D. Diefenbach and A. Thalhammer, Pagerank and generic entity summarization for RDF knowledge bases, in: *European Semantic Web Conference*, Springer, 2018, pp. 145–160.
- [5] Y. Huang, V. Tresp, M. Nickel, A. Rettinger and H.-P. Kriegel, A scalable approach for statistical learning in semantic graphs, *Semantic Web* **5**(1) (2014), 5–22.
- [6] M. Nickel, K. Murphy, V. Tresp and E. Gabrilovich, A Review of Relational Machine Learning for Knowledge Graphs, *Proceedings of the IEEE* **104**(1) (2016), 11–33, ISSN 0018-9219. doi:10.1109/JPROC.2015.2483592.
- [7] G. Piao and J.G. Breslin, Transfer Learning for Item Recommendations and Knowledge Graph Completion in Item Related Domains via a Co-Factorization Model, in: *The Semantic Web*, A. Gangemi, R. Navigli, M.-E. Vidal, P. Hitzler, R. Troncy, L. Hollink, A. Tordai and M. Alam, eds, Springer International Publishing, 2018, pp. 496–511.
- [8] A. Rettinger, U. Löscher, V. Tresp, C. d’Amato and N. Fanizzi, Mining the Semantic Web, *Data Mining and Knowledge Discovery* **24**(3) (2012), 613–662.
- [9] P. Shiralkar, A. Flammini, F. Menczer and G.L. Ciampaglia, Finding streams in knowledge graphs to support fact checking, in: *IEEE International Conference on Data Mining (ICDM)*, IEEE, 2017, pp. 859–864.

- [10] L. Zhu, M. Ghasemi-Gol, P. Szekely, A. Galstyan and C.A. Knoblock, Unsupervised entity resolution on multi-type graphs, in: *International Semantic Web Conference*, Springer, 2016, pp. 649–667.
- [11] P. Ristoski and H. Paulheim, RDF2Vec: RDF Graph Embeddings for Data Mining, in: *The Semantic Web - ISWC 2016 - 15th International Semantic Web Conference, Kobe, Japan, October 17-21, 2016, Proceedings, Part I*, 2016, pp. 498–514.
- [12] T. Mikolov, I. Sutskever, K. Chen, G.S. Corrado and J. Dean, Distributed Representations of Words and Phrases and their Compositionality, in: *27th Annual Conference on Neural Information Processing Systems - NIPS 2013*, 2013, pp. 3111–3119.
- [13] J. Pennington, R. Socher and C.D. Manning, Glove: Global Vectors for Word Representation, in: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014*, 2014, pp. 1532–1543.
- [14] M. Cochez, P. Ristoski, S.P. Ponzetto and H. Paulheim, Biased graph walks for RDF graph embeddings, in: *Proceedings of the 7th International Conference on Web Intelligence, Mining and Semantics, WIMS 2017*, 2017, pp. 21–12112.
- [15] A. Grover and J. Leskovec, node2vec: Scalable Feature Learning for Networks, in: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining 2016*, 2016, pp. 855–864.
- [16] A. Thalhammer and A. Rettinger, PageRank on Wikipedia: Towards General Importance Scores for Entities, in: *The Semantic Web: ESWC 2016 Satellite Events, 2016*, Springer International Publishing, Cham, 2016, pp. 227–240. ISBN 978-3-319-47602-5.
- [17] T. Di Noia, R. Mirizzi, V.C. Ostuni, D. Romito and M. Zanker, Linked open data to support content-based recommender systems, in: *Proceedings of the 8th International Conference on Semantic Systems*, ACM, 2012, pp. 1–8.
- [18] A. Lausch, A. Schmidt and L. Tischendorf, Data mining and linked open data—New perspectives for data analysis in environmental research, *Ecological Modelling* **295** (2015), 5–17.
- [19] T. Mikolov, K. Chen, G. Corrado and J. Dean, Efficient Estimation of Word Representations in Vector Space, *CoRR abs/1301.3781* (2013).
- [20] M.R. Saeed and V.K. Prasanna, Extracting Entity-specific Substructures for RDF Graph Embedding, in: *9th IEEE International Conference on Information Reuse and Integration (IRI), Salt Lake City, Utah, USA*, IEEE, 2018.
- [21] L. Yu, Linked Open Data, in: *A Developer's Guide to the Semantic Web*, Springer Berlin Heidelberg, Berlin, Heidelberg, 2011, pp. 409–466. ISBN 978-3-642-15970-1.
- [22] P. Hitzler and F. Van Harmelen, A reasonable semantic web, *Semantic Web* **1**(1, 2) (2010), 39–44.
- [23] C. Paul, A. Rettinger, A. Mogadala, C.A. Knoblock and P.A. Szekely, Efficient Graph-Based Document Similarity, in: *The Semantic Web. Latest Advances and New Domains - 13th International Conference, ESWC 2016*.
- [24] M.R. Saeed, C. Chelmiss and V.K. Prasanna, Automatic Integration and Querying of Semantic Rich Heterogeneous Data: Laying the Foundations for Semantic Web of Things, in: *Managing the Web of Things: Linking the Real World to the Web*, 2017, pp. 251–273. doi:10.1016/B978-0-12-809764-9.00012-3. <https://doi.org/10.1016/B978-0-12-809764-9.00012-3>.
- [25] M. Färber, F. Bartscherer, C. Menne and A. Rettinger, Linked data quality of dbpedia, freebase, open-cyc, wikidata, and yago, *Semantic Web*, 1–53.
- [26] A. Ismayilov, D. Kontokostas, S. Auer, J. Lehmann and S. Hellmann, Wikidata through the eyes of DBpedia, *Semantic Web* **9**(4) (2018), 493–503.
- [27] P. Goyal and E. Ferrara, Graph embedding techniques, applications, and performance: A survey, *Knowledge-Based Systems* **151** (2018), 78–94, ISSN 0950-7051. doi:<https://doi.org/10.1016/j.knosys.2018.03.022>. <http://www.sciencedirect.com/science/article/pii/S0950705118301540>.
- [28] P. Ristoski, S. Faralli, S.P. Ponzetto and H. Paulheim, Large-scale taxonomy induction using entity and word embeddings, in: *Proceedings of the International Conference on Web Intelligence*, ACM, 2017, pp. 81–87.
- [29] B. Shi and T. Weninger, ProjE: Embedding Projection for Knowledge Graph Completion, in: *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.*, 2017, pp. 1236–1242. <http://aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14279>.
- [30] G.K.D. de Vries and S. de Rooij, Substructure counting graph kernels for machine learning from RDF data, *J. Web Sem.* **35** (2015), 71–84.
- [31] U. Lössch, S. Bloehdorn and A. Rettinger, Graph Kernels for RDF Data, in: *The Semantic Web: Research and Applications - 9th Extended Semantic Web Conference, ESWC 2012*, 2012, pp. 134–148.
- [32] N. Shervashidze, S.V.N. Vishwanathan, T. Petri, K. Mehlhorn and K.M. Borgwardt, Efficient graphlet kernels for large graph comparison, in: *Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics, AISTATS 2009*, 2009, pp. 488–495.
- [33] P. Yanardag and S.V.N. Vishwanathan, Deep Graph Kernels, in: *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, August 10-13, 2015*, 2015, pp. 1365–1374.
- [34] M. Cochez, P. Ristoski, S.P. Ponzetto and H. Paulheim, Global rdf vector space embeddings, in: *International Semantic Web Conference*, Springer, 2017, pp. 190–207.
- [35] B. Perozzi, R. Al-Rfou and S. Skiena, DeepWalk: online learning of social representations, in: *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2014*, 2014, pp. 701–710.
- [36] N. Aggarwal, K. Asooja, H. Ziad and P. Buitelaar, Who are the American Vegans related to Brad Pitt?: Exploring Related Entities, in: *Proceedings of the 24th*

- International Conference on World Wide Web Companion, WWW 2015*, 2015, pp. 151–154.
- [37] E. Gabrilovich and S. Markovitch, Computing Semantic Relatedness Using Wikipedia-based Explicit Semantic Analysis, in: *IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence, 2007*, 2007, pp. 1606–1611.
- [38] J.P. Leal, V. Rodrigues and R. Queirós, Computing Semantic Relatedness using DBpedia, in: *1st Symposium on Languages, Applications and Technologies, SLATE 2012*, 2012, pp. 133–147.
- [39] Y. Sun, J. Han, X. Yan, P.S. Yu and T. Wu, PathSim: Meta Path-Based Top-K Similarity Search in Heterogeneous Information Networks, *PVLDB* 4(11) (2011), 992–1003.
- [40] Y. Tzitzikas, C. Lantzaki and D. Zeginis, Blank Node Matching and RDF/S Comparison Functions, in: *The Semantic Web - ISWC 2012 - 11th International Semantic Web Conference, Boston, MA, USA, November 11-15, 2012, Proceedings, Part I*, 2012, pp. 591–607.
- [41] X. Ning, H. Jin, W. Jia and P. Yuan, Practical and effective IR-style keyword search over semantic web, *Inf. Process. Manage.* 45(2) (2009), 263–271.
- [42] S. Cederberg and D. Widdows, Using LSA and noun coordination information to improve the precision and recall of automatic hyponymy extraction, in: *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, Association for Computational Linguistics, 2003, pp. 111–118.
- [43] M.A. Rodríguez and M.J. Egenhofer, Determining semantic similarity among entity classes from different ontologies, *IEEE transactions on knowledge and data engineering* 15(2) (2003), 442–456.
- [44] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P.N. Mendes, S. Hellmann, M. Morsey, P. van Kleef, S. Auer and C. Bizer, DBpedia - A large-scale, multilingual knowledge base extracted from Wikipedia, *Semantic Web* 6(2) (2015), 167–195.