Semantic Web 0 (0) 1
IOS Press

Extracting Entity-specific Substructures for RDF Graph Embeddings

Muhammad Rizwan Saeed [a,*], Charalampos Chelmis [b] and Viktor K. Prasanna [a]
[a] *Ming Hseih Department of Electrical Engineering, University of Southern California, CA, USA*
E-mails: saeedm@usc.edu, prasanna@usc.edu
[b] *Department of Computer Science, University at Albany - SUNY, NY, USA*
E-mail: cchelmis@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: Relevance Metrics, 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[1], 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 subgraph* for each target entity in the KG must be acquired. Each entity (represented by a node) in a heterogeneous KG is surrounded by other entities (or nodes) connected by directed labeled edges. For a node representing a film, there can be a labeled edge *director* connecting it to a node representing the director of the film. Another labeled edge *releaseYear* may exist that connects the film to an integer literal, also represented by a node in the KG. If we want to extract a subgraph representing the film, ideally such relevant information (labeled edges and nodes) must be a part of the extracted subgraph. On the other hand, relationships linked to, say, the *Director* such as *birthYear* or *deathYear*, which are at two hops from a film, e.g.

*Corresponding author. E-mail: saeedm@usc.edu.
[1] https://www.blog.google/products/search/introducing-knowledge-graph-things-not/

$Batman\_1989 \xrightarrow{director} Tim\_Burton \xrightarrow{birthYear}$ 1958, may not be useful or relevant for representing a film. Therefore, in order to extract a useful representation (as a subgraph) of a given entity, we first need to automatically determine the most relevant edges and nodes in its neighborhood. An extracted representation of any target entity[2] in a KG can only be considered representative if it includes only the most relevant nodes and edge w.r.t the target entity. 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 and 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.

In this paper, we focus on RDF[3] KGs which are encoded using the *Resource Description Framework* (RDF) syntax and constitute Linked Open Data [17, 18]. We assert that the *representative* subgraphs of different types of entities (e.g., book, film, drug, athlete) in RDF KGs may comprise distinct sets of 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.

The main contributions of this paper are as follows:

- We propose *Specificity* as an accurate measure for assigning weights to those semantic relationships which constitute the most intuitively relevant 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 enable more compact extraction of relevant subgraphs for target entities in a KG as compared to the state-of-the-art.

To demonstrate the usefulness of our specificity-based approach for real-world applications, we train neural language model (Skip-Gram [19]) for generating graph embedding from the extracted subgraphs and use the generated embeddings for the tasks 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 regression and classification tasks. We also propose a variation of specificity called $Specificity^H$ which takes into account the hierarchy of classes in the schema ontology associated with a KG, discussed in Section 4.2.

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.

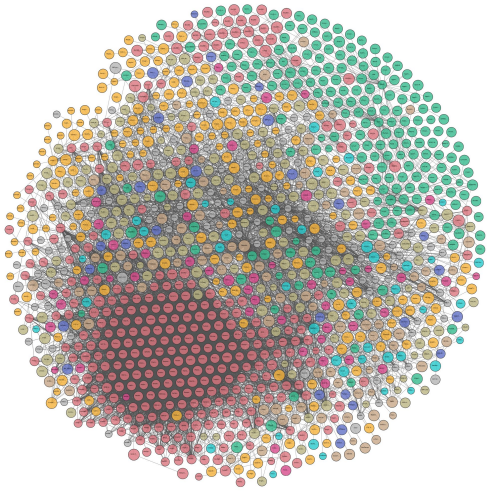


Fig. 1. Linked Open Data [Source: http://lod-cloud.net/]

[2]Since entities are represented as nodes in a KG; we use entity and node interchangeably throughout the text.

[3]https://www.w3.org/TR/rdf-concepts/

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 [22–25]. LOD cloud currently comprise 1205 KGs[4]. Few of the KGs part of LOD are *Wikidata*[5], *Freebase*[6], and *DBpedia*[7] [26, 27].

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 [28–30].

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) [31, 32] or graphlets [33, 34]. Neural language models such as word2vec [12] and GloVe [13], proposed initially for generating word embedding, have been adapted for KGs [11, 15, 35]. Deep Graph Kernel [34] 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 [36] 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^{nd} 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, 28]. 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 approach [14] which uses frequency-, degree-, and PageRank-based metrics for weighting schemes. Our proposed approach also uses biased random walks to extract entity representations. However, unlike [14], we use our proposed metric of *specificity* as an edge- and path-weighting scheme for biased random walks for identifying most relevant subgraphs for extracting entity representations in the KGs.

Semantic similarity and relatedness between two entities have been relatively well explored [23, 37–39]. Searching for similar or related entities given a search query is a common task in the field of Information Retrieval [17, 40, 41]. 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, 24, 37, 42], 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.

[4]Source: http://lod-cloud.net/
[5]https://www.wikidata.org/wiki/Wikidata:Main_Page
[6]https://developers.google.com/freebase/
[7]https://wiki.dbpedia.org/

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.

3. Preliminaries

An RDF graph is represented by a knowledge base of triples [43]. A triple consists of three parts: *<subject (s), predicate (p), object (o)>*.

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:*

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

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 Datatype 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 the traditional definition of a graph. Blanks nodes can either be subject or object whereas literals can only be objects. An RDF graph is a set of all such RDF triples [43, 44]. 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 [45].

Definition 2. ***Semantic Relationship:*** *A semantic relationship in an RDF graph can be defined as $< s, p^d, o >$ where p^d represents a path comprising d successive predicates and $d-1$ intermediate nodes between s and o. When $d = 1$, $< s, p^1, o >$ becomes equivalent to a single triple $< s, p, o >$, where p^1 or p represents a single predicate or edge between two nodes 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 $p_1, p_2, \ldots, p_d$. Thus, $< s, p^d, o >$ 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. To understand the difference between a triple, a path, and a template, assume that an RDF KG consists of only the following four triples: $< s_1, a, x >$, $< x, b, o_1 >$, $< s_2, a, y >$, and $< y, b, o_2 >$. $< s_1, a, x >$ and $< x, b, o_1 >$ constitute a complex relationship between s_1 and o_1. These two triples together are an example of a semantic relationship of form p^2 which consists of two successive predicates a and b and one intermediate node x. If we treat a, b as a template of a path, then in this KG there are two instances of this template: the semantic relationship between s_1 and o_1 (first and second triples) and the semantic relationship between s_2 and o_2 (third and fourth triples). This also means that for an arbitrary s and o, $| < s, p^2, o > | = 2$, where $p^2 = a, b$. We will formulate the expression for specificity using this notation in Section 4.

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: $v_0 \xrightarrow{e_1} v_1 \xrightarrow{e_2} v_2 \xrightarrow{e_3} \ldots \xrightarrow{e_d} v_d$.*

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, 36].

Definition 4. ***Representative Subgraph:*** *The representative subgraph (neighborhood) of an entity or a node v is the set of other nodes S in the graph that represent the most relevant information related to v. The edges or paths that link v to every $s \in S$ are also part of the representative subgraph.*

Using our running example from Section 1, the representative subgraph of a film may contain depth-1 edges such as *director, producer* that link it to nodes representing its director(s) and producer(s). Some examples of depth-2 relationships are $Film \xrightarrow{basedOn} Book \xrightarrow{writtenBy} Author$ and $Film \xrightarrow{director} Director \xrightarrow{knownFor} Style$. On the other hand, our intuition suggests that semantic relationships such as $Film \xrightarrow{director} Director \xrightarrow{birthYear} Year$ is not as relevant to represent a film and hence must not be part of the representative subgraph.

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 representing 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.

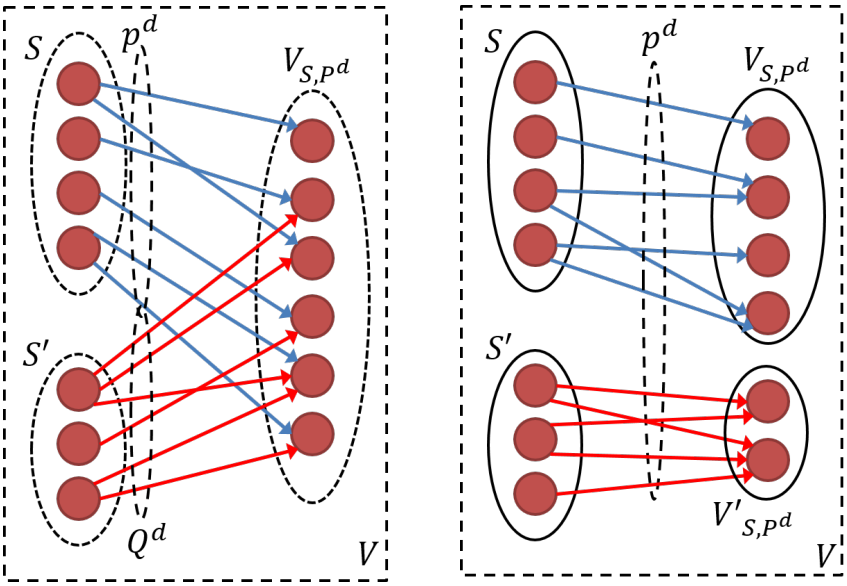
Batman (1989) $\xrightarrow{\text{dir.}}$ Tim Burton $\xrightarrow{\text{knownFor}}$ Gothic Films
Batman (1989) $\xrightarrow{\text{dir.}}$ Tim Burton $\xrightarrow{\text{subject}}$ 1958 births
Batman (1989) $\xrightarrow{\text{dir.}}$ Tim Burton $\xrightarrow{\text{birthPlace}}$ Burbank, CA

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

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:

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

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



(a) $Specificity(p^d, S) = 50\%$ when equal number of paths terminate on V_{S,p^d} from set S and its complement S'

(b) $Specificity(p^d, S) = 100.0\%$ when p^d is relevant to multiple different subsets of V but the information V_{S,p^d} it links to S is exclusively connected to S

Fig. 3. Illustration of specificity

(*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 5. ***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*

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

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.

4.2. $Specificity^H$: Incorporating Hierarchy of Classes into Specificity computations

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 apply to a broader class and by extension to

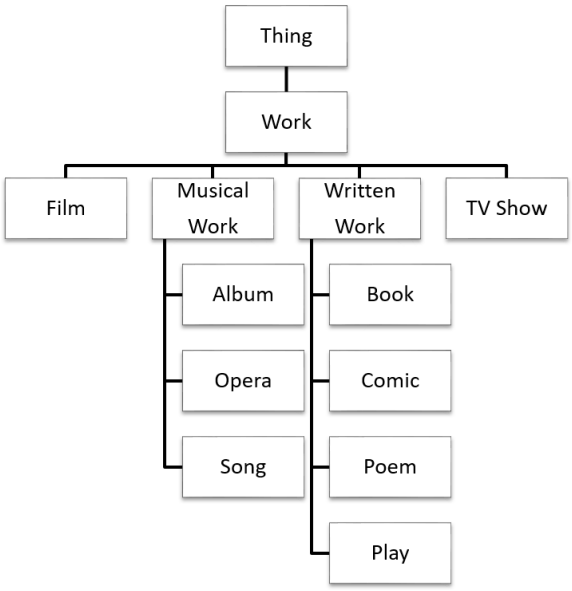


Fig. 4. A subset of DBpedia class hierarchy

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 the relevance of the semantic relationships to the other subclasses such as *TelevisionShow*. To rectify this and make the specificity computations more 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 [46, 47]. Hyponymy represents the relationship between hypernym (the 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 $C_0 \subseteq C_1 \subseteq \ldots \subseteq C_h$. 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:

$$Specificity(p^d, S) = \frac{1}{|V_{S,p^d}|} \sum_{k \in V_{S,p^d}} \frac{1}{|< v, q^d, k >\in G : \forall v \in V|} \cdot \Big(\sum_{j=0}^{h} \beta^j |< v_j, q^d, k >\in G : \forall v_j \in V \wedge type(v_j) = C_j| \Big) \quad (4)$$

Additional constraints for Equation 4 are: $v_0 \in S$ and $v_1 \cap v_2 \cap \ldots \cap v_h \cap S = \emptyset$. This is to ensure that each instance of type t or any of its superclasses is only considered once in the computation. The factor β 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.

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 $depths \leqslant d$, 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: $\quad Q_{paths} \leftarrow selectPaths(G, S, i, N_{paths})$
6: $\quad Q_{spec}[i] \leftarrow computeSpecificity(G, Q_{paths}, S, t, i, N_{walks}, \beta)$
7: **end for**
8: **return** $Q_{spec}[]$

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 ($i \leqslant d$), 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 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 $i \geqslant 2$, 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 semantic relationship q in G (*forward walk*). In line 9, by using v and selecting an arbitrary path via any semantic relationship 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, 48]. 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), $H = [\text{“}Film''$, “$Work''$, “$Thing'']$. Starting from the first element in H at index $i = 1$, the existence of triple $< v', rdf : type, H[i] >$ is checked in G. If such a triple exists, count is incremented by a factor of β^{i-1}. This process of bidirectional 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.

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: $\quad count \leftarrow 0.0$
6: $\quad$ **repeat**
7: $\quad\quad s \leftarrow$ randomly pick a node from S
8: $\quad\quad v \leftarrow$ randomly explore node from s using any path using q in G
9: $\quad\quad v' \leftarrow$ randomly explore node from v using any path using q' in G^r
10: $\quad\quad$ **for** $i \leftarrow 1, size(H)$ **do**
11: $\quad\quad\quad$ **if** $\exists < v', rdf : type, H[i] > \in G$ **then**
12: $\quad\quad\quad\quad count \leftarrow count + \beta^{i-1}$
13: $\quad\quad\quad\quad$ break
14: $\quad\quad\quad$ **end if**
15: $\quad\quad$ **end for**
16: $\quad$ **until** N times
17: $\quad$ insert $(q, \frac{count}{N})$ in L
18: **end for**
19: **return** L

6. Evaluation

We evaluate our approach in multiple ways. We analyze the behavior of specificity computed for most relevant semantic relationships up to depth 3. We evaluate the compactness of the extracted subgraphs by specificity-based biased random walk scheme against other metrics used as baselines. We generate embeddings from the extracted subgraphs to perform tasks of entity recommendation, regression, and classification. We analyze the ability of generated embeddings in preserving the semantics associated with the entities extracted from the KG. We study the sensitivity of specificity to the parameter N_{walks} (number of bidirectional walks) and $|S|$ (seed set size). We provide an empirical analysis of the running time of Algorithm 2.

6.1. Datasets

We use DBpedia for evaluation which is one of the largest RDF repositories publicly available [49]. We have used the English version of DBpedia dataset from 2016-04 (http://wiki.dbpedia.org/dbpedia-version-2016-04). We create graph embeddings for 5000 entities each of types: *Film*, *Book*, and *Album*. We also generate embeddings for 500 and 3000 entities of types *Country* and *City* respectively.

We use three different datasets for the tasks of classification and regression which provide classification and regression targets for DBpedia cities, films, and music albums:

- The *Mercer Cities*[8] dataset contains a list of cities and their quality of living as numeric scores and discrete labels ("high", "medium", "low").
- The *Metacritic Movies*[9] and *Metacritic Music Albums*[10] datasets contain the Metacritic score (0-100) which were used as regression targets. The classification targets are provided as either "good" (score $\geqslant$ 50) and bad (score $<$ 50) [11]. These datasets are are accessible at http://data.dws.informatik.uni-mannheim.de/rmlod/LOD_ML_Datasets/data/datasets/.

6.2. Experimental Setup and Methodology

For our experiments, we hosted the DBpedia dataset using OpenLink Virtuoso[11] on a server. All data transactions between the implemented modules and the repository occurred as SPARQL queries.

Implementation of Baselines and Proposed Approach: In the first step, we computed $Specificity$ and $Specificity^H$ 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: $N_{walks} = 5000$, $|S| = 500$, $\beta = 0.25$. Starting from a random seed set S for each type of entities, we randomly sampled semantic relationships originating from entities in S and selected *top-25d* semantic relationships based on the frequency of occurrence for each depth d. This became the frequency-based baseline, where the relevance of semantic relationships were based on how frequently they occurred in the KG. After computing specificity for each of the most frequent semantic relationship, we only considered those as relevant that had specificity scores $\geqslant 50\%$. For creating the PageRank-based baseline, we used the PageRank DBpedia dataset provided by Thalhammer et al. [16] (available at http://people.aifb.kit.edu/ath/#DBpedia_PageRank).

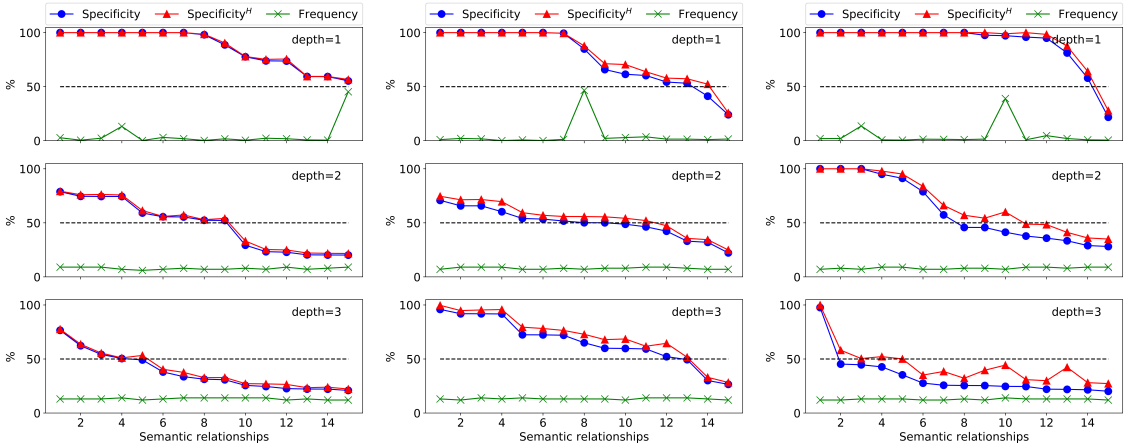
Biased Random Walks for Subgraph Extraction: Using the lists of most relevant semantic relationships based on frequency-, PageRank-, and specificity-based metrics, we performed subgraph extraction using biased random walks. We used weighted randomized DFS for subgraph extraction for each target entity. The DFS algorithm traversed the paths starting from each target node using the list of most relevant semantic relationships. The nodes linked by more relevant semantic relationships had a greater likelihood to become part of the extracted subgraphs. The subgraphs are extracted as a set of unique graph walks (Definition 3) and can be represented as a sequence of edge and node labels, in the order in which they were

[8] https://mobilityexchange.mercer.com/Insights/quality-of-living-rankings

[9] http://www.metacritic.com/browse/movies/score/metascore/all

[10] http://www.metacritic.com/browse/albums/score/metascore/all

[11] https://virtuoso.openlinksw.com/



(a) Books (b) Music Albums (c) Films

Fig. 5. Comparison of frequency- and specificity-based metrics for top-15 semantic relationships

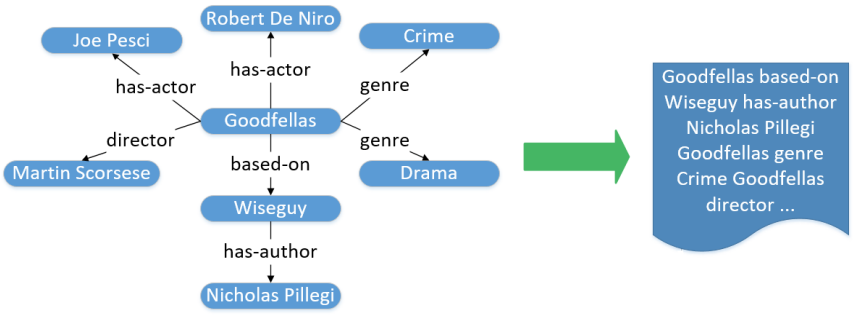


Fig. 6. Extracted subgraphs are represented as sequence of edge and node labels resulting in a document-like representation

visited [11, 14]. An example of document-based representation of extracted subgraph is shown in Figure 6. For our experiments, we include up to a maximum of 1000 distinct paths in the extracted subgraphs for each entity for proposed and baseline approaches.

Embedding Generation using word2vec: With the document-based representation, we used the Python library gensim[12] which provides the implementation of word2vec [12] to estimate representation of each label in the generated document into vector space creating embedding for the RDF entities [11].

[12]https://radimrehurek.com/gensim/index.html

6.3. *Specificity as a Metric for Measuring Relevance*

Figure 5 shows the top 15 semantic relationships for entities of three types sorted by their *Specificity* up to depth 3. The frequency represents the percentage 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 topmost 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. Alternatively, we can choose a value k such that *top-k* high-frequency semantic relationships are selected as 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 $k \geqslant 7$. However, this ad-hoc method of selecting thresholds is impractical since it has to be done for every different class of entities, every depth, and every different RDF KG.

Table 1
Top semantic relationships based on frequency with corresponding specificity scores for *dbo:Film*

Semantic Relationship	**Freq. (%)**	**Spec. (%)**	**Spec.^{H} (%)**
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 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 5) 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 α^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 $< d$. This way specificity exhibits a more fine-grained behavior of relevance across depths, meaning that all semantic relationships at depth d do not simultaneously become less relevant as compared to those on depth $d-1$, as d increases. 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.

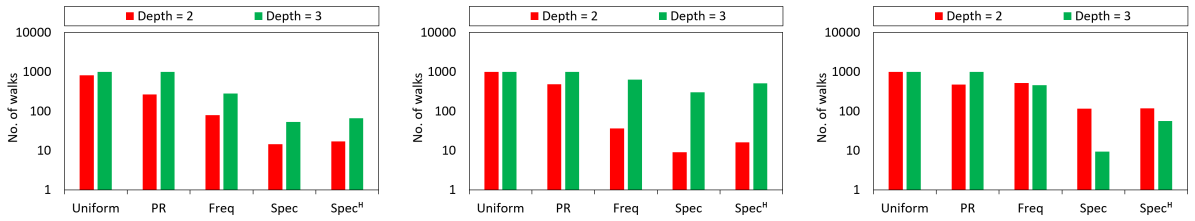
Table 2
Comparison of relevance metrics for example in Figure 2

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 the corresponding semantic relationship in DBpedia 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. $Specificity^H$

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



(a) Books (b) Music Albums (c) Films

Fig. 7. Average number of walks per entity for subgraph extraction

Table 3
Comparison of Specificity and $Specificity^H$ scores (β = 0.25)

Semantic Relationships	**Entity Type**	**Spec. (%)**	$Spec^H$ (%)
dbo:language	dbo:Film	38.65	53.42
dbo:genre	dbo:Album	43.29	50.8
dbo:recordLabel	dbo:Album	43.41	56.34
dbp:genre	dbo:Book	46.85	52.29
dbo:previousWork	dbo:Book	40.0	55.0
dbo:litrerayGenre	dbo:Book	46.22	51.52

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).

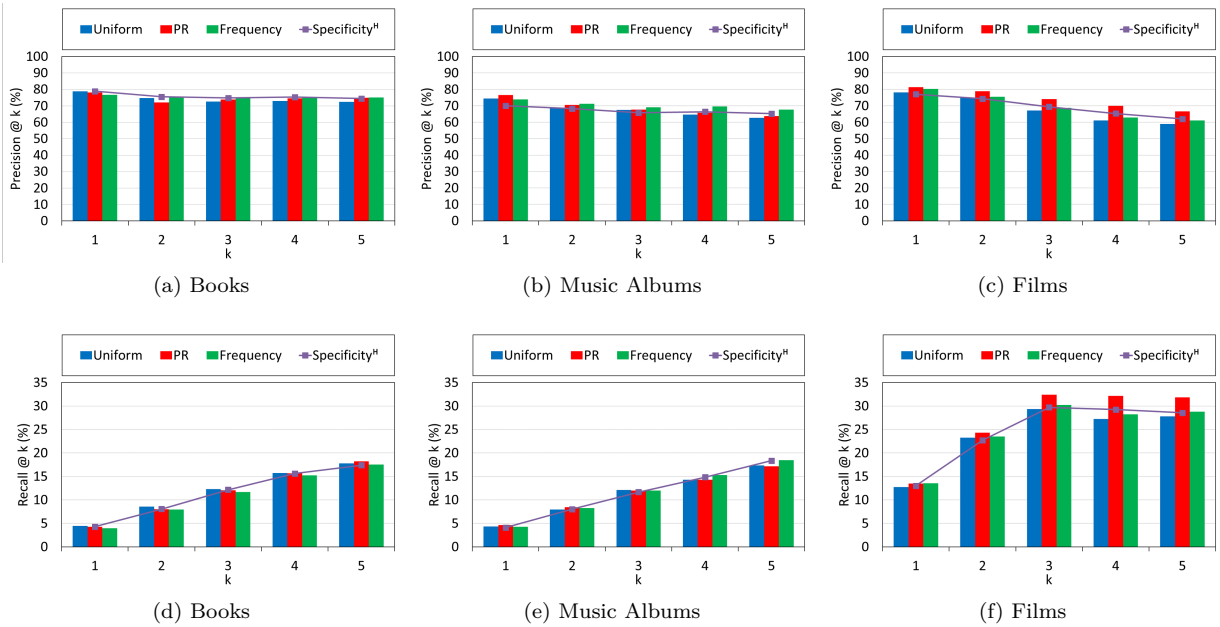
Table 3 shows a few examples of intuitively relevant semantic relationships with specificity scores below the threshold of 50%, resulting in the exclusion from the representative subgraphs of instances of corresponding entity types. $Specificity^H$ computed using Algorithm 2 takes into account the applicability of such relationships to co-hyponyms and hypernyms of the given entity types, resulting in an increase in the specificity score. This is evident in all plots in Figure 5 where the curve of $Specificity^H$ lie on or above the corresponding curve of $Specificity$.

6.4. Comparison of Sizes of Representative Subgraphs

Figure 7 shows that specificity-based random walk schemes enable the extraction of relevant subgraphs with fewer number of walks. Specificity-based schemes use extraction template based on semantic relationships with $\geqslant$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. $Specificity^H$, as seen in Figure 5 assigns higher specificity scores, resulting in a few more semantic relationships meeting the 50% threshold. This results in an increase in the collected number of walks which, nevertheless, is still below the baseline approaches.

Figure 7a shows that the average size of the subgraph of each entity of type dbo:Book for $Specificity^H$ is smaller by a factor of 48, 16, and 5 w.r.t uniform, PageRank, and Frequency-based approaches respectively. Similarly, Figure 7b shows that the average size of the subgraph of each entity of type dbo:Book for $Specificity^H$ is smaller by a factor of 62, 30, and 2 w.r.t uniform, PageRank, and Frequency-based approaches respectively. We will revisit this discussion in the context of its impact on the recommendation task in Section 6.5.1.1.

Figure 7a shows that the average subgraph size is larger for depth-3 than depth-2. One of the reasons is that the most specific depth-2 property for dbo:Book entities add on average one walk to each of the extracted subgraphs, whereas the most specific depth-3 property add 15 walks on average. On the other hand, Figure 7c shows the opposite effect where depth-2 subgraph size is larger than depth-3 for Specificity (β = 0.0). The top specific semantic relationships at depth-2 and depth-3 add 100.8



(a) Books (b) Music Albums (c) Films

(d) Books (e) Music Albums (f) Films

Fig. 8. Comparison of precision and recall for entity recommendation tasks ($\beta = 0.25$ for $Specificity^H$)

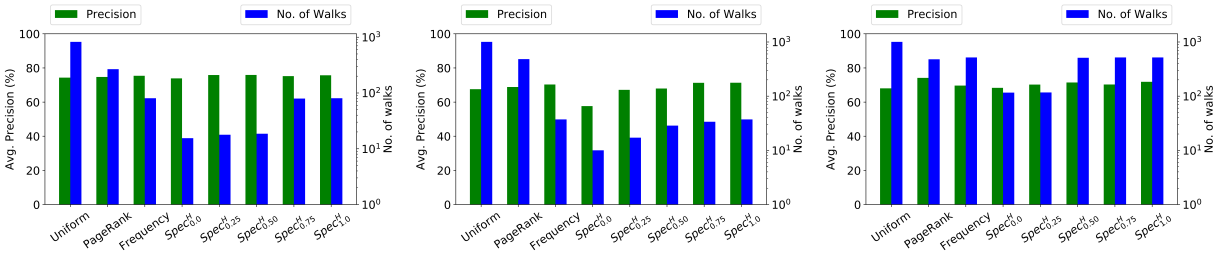
and 9.4 walks on the average to each extracted subgraph. The main underlying reason is that having a higher number of specific semantic relationships does not necessarily indicate a larger extracted subgraph or vice versa. The size of the subgraph depends on the number of nodes a particular semantic relationship connects to the target entities to include in the extracted subgraph. For example, for film nodes, on average dbo:director and dbo:cinematographer add 1.06 and 1.07 walks to the extracted subgraphs whereas rdf:type and dct:subject add 28 and 8 walks respectively.

6.5. Embeddings as an Application of Specificity-based Extracted Subgraphs

We have shown that the specificity-based biased random walks extract more compact subgraphs representing entities as compared to other schemes. However, to show that the compactness of the extracted subgraphs is not a disadvantage, we use the graph embeddings as an application to evaluate their effectiveness. Using the extracted subgraphs extracted as documents (Section 6.2), 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. The parameters for this experiment are based on RDF2Vec [11].

6.5.1. Suitability for Entity Recommendation task

To show that the compactness of the extracted subgraphs is not a disadvantage, we use the generated graph embeddings for the task of entity recommendation. Given a vectorized entity as the search key, we list its top-k most similar results. We use the metrics of precision@k to quantify the performance of the recommendation tasks. 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 or a series usually share 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 director, cast members, genre, and characters. In our experiments, *King Kong (2005)* also frequently appeared among the results similar to LOTR since it also has the same director. Other films may also occur among top-k



(a) Books (b) Music Albums (c) Films

Fig. 9. Effect of β on recommendation task

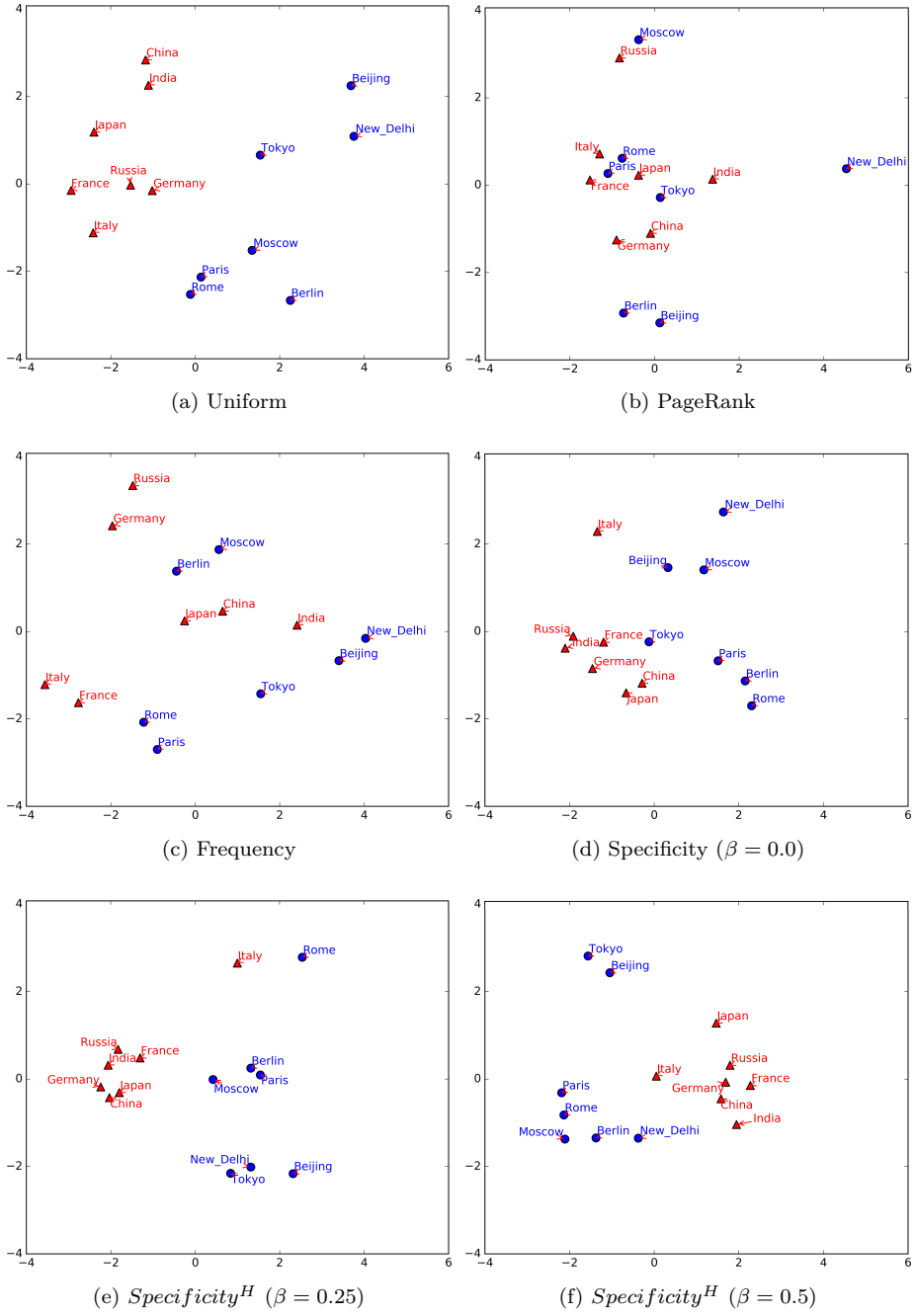
results based on any number of other factors, e.g., cinematography, editing, distributor, all of which have high specificity to *dbo:Film* (from Table 1). Creating exhaustive lists of films for the ground truth to encompass all such scenarios is laborious. That is why we included only those films in the ground truth that are either in a franchise (e.g., all Batman films) 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 59616, 221280, and 124032 recommendation tasks for films, books, and music albums for each random walk scheme respectively.

6.5.1.1. Results Figure 8 shows the results of recommendation tasks for each scheme. Here, we have chosen $\beta = 0.25$ for computing $Specificity^H$. The baselines are shown as colored bars whereas $Specificity^H$ is drawn as a line to make the comparison clearer. $Specificity^H$ is generally slightly better than other baseline schemes except in Figure 8c where PageRank-based extracted subgraphs have a better performance.

Here, it is important to note that the compactness of the extracted subgraphs is the main contribution of our approach. We use recommendation as an application to show that despite extracting less information from the KG, there is not a significant deterioration in performance when using the specificity-based approach with such applications. The results of the recommendation task must be interpreted in conjunction with our primary metric of the size of the extracted subgraph. Figure 9 shows the For *dbo:Book*, Figure 9a shows that the average size of the subgraph for each entity is 48, 16, and 5 times smaller for $Specificity^H$ w.r.t uniform, PageRank, and Frequency-based approaches respectively, whereas the precision values for $Specificity^H(\beta = 0.25)$ and the three baselines lie in $75.1 \pm 0.78\%$. Similarly, for music albums (Figure 9b), the subgraph size for $Specificity^H$ ($\beta = 0.25$) is reduced by factors of 62, 30, and 2 respectively with precision lying $68.7 \pm 1.6\%$. Figure 9c shows that the average size of the subgraph for PageRank-based approach for each entity is 4.2 times larger than $Specificity^H$ with $\beta = 0.25$ with an advantage of 5% in precision. The size of the subgraph for each entity for unbiased (uniform) approach is ten times larger than $Specificity^H$-based approach for films. The range of precision values for $Specificity^H$ ($\beta = 0.25$) and the baselines lie in the range $71.1 \pm 3\%$. This shows that a substantial reduction in extracted information still allows us to have comparable performance in entity recommendation task with the specificity-based approach.

6.5.1.2. Effects of β on Specificity Increasing the value of β means that the $Specificity^H$ score (for $beta > 0.0$) for any semantic relationship will be equal or higher than its $Specificity$ score. This is also evident from Figure 5 where $Specificity^H$ ($\beta = 0.25$) $\geqslant Specificity$ ($\beta = 0.0$). This will result in more semantic relationships having higher specificity than the cut-off set at 50%. With a higher number of relevant semantic relationships used for subgraph extraction, the size of extracted subgraphs can increase.

Figure 9 also shows the effect of changing β on the size of the extracted subgraphs and the rec-



(a) Uniform (b) PageRank

(c) Frequency (d) Specificity ($\beta = 0.0$)

(e) $Specificity^H$ ($\beta = 0.25$) (f) $Specificity^H$ ($\beta = 0.5$)

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

ommendation task. Here first we refer back to discussion around Algorithm 1 in Section 5. In Line 5 Q_{paths} holds the set of selected semantic relationships which are then re-ordered and filtered in Line 6 using the metric of specificity. In our implementation, the semantic relationships for populating Q_{paths} are selected based on their frequency of occurrence. For $\beta = 1.0$, every semantic relationship in Q_{paths} gets a high specificity score in Line 6, i.e., the *count* variable in Line 12 gets incremented by 1 irrespective of the type of v' in Line 9. This means that all *frequent* semantic relation-

Table 4
Results of Regression and Classification Tasks - $Specificity^H(\beta = 0.25)$

Model	Depth	Classification			Regression		
		Cities	Films	Albums	Cities	Films	Albums
Uniform	1	0.935	1.0585	0.983	0.9011	1.0079	1.0002
Uniform	2	0.9359	1.0273	1.0374	0.9131	0.9976	1.0056
PageRank	1	1.0985	1.029	0.9663	0.9546	0.9955	1.0033
PageRank	2	1.0135	0.9849	1.0484	0.9347	0.9935	1.0019
Frequency	1	1.0709	1.0118	0.9421	0.9506	0.998	0.9926
Frequency	2	0.9558	1.0294	0.9886	1.0328	1.0002	0.9996
SpecificityS	1	1.1098	1.0341	0.9894	1.0601	0.9967	1.0017
SpecificityS	2	0.9161	1.023	1.0401	0.8882	0.9985	1.0027

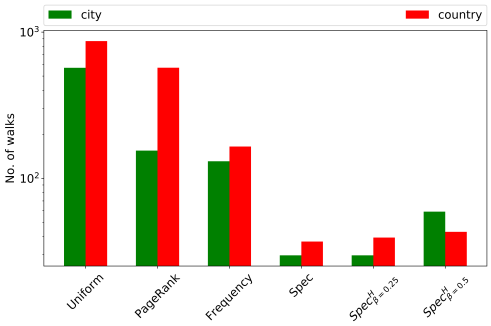


Fig. 11. Average number of walks per entity of types *dbo:City* and *dbo:Country* for subgraph extraction for different values of β

ships are also deemed *specific* for $\beta = 1.0$. Figure 9 shows the average sizes of extracted subgraphs for Frequency and $Specificity^H$ ($\beta = 1.0$) are the same in each subplot.

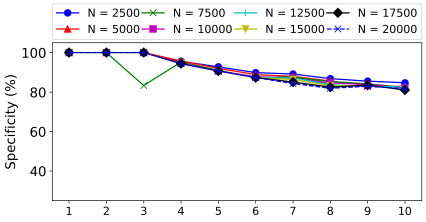
6.5.2. Semantics of Specificity-based Vector Representations

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. Figures 10d and 10e show that specificity-based embeddings are capable of organizing entities of different types and preserving some semantic context among them. For instance, there is a separation between two different types of entities: *dbo:Country* and *dbo:City*, preserving the *rdf:type* relationship. Compared to $Specificity$, $Specificity^H$ has comparatively better, although not perfect, organization of capitals in correspondence to their respective countries. Third, in Figure 10e, cities are grouped together based on the continents. The information regarding continents is represented via the semantic relationship *dct:subject* which links each city to different DBpedia categories including one that represents information about continents, e.g. *dbc:Capitals_in_Asia*.

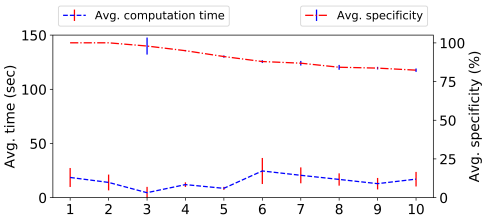
Figures 10b and 10c show that *rdf:type* relationship is not preserved by frequency and PageRank-based projections respectively. There exists some, but inconsistent, organization of countries and capitals with respect to each other and a grouping based on continents. There is also no clear segregation based on entity types. Figure 10a shows that semantic associations are also preserved using uniform random walks. However, it can be seen in Figure 11 that the average size of extracted subgraphs is order of magnitude more than the Specificity-based approach.

6.5.3. Suitability for Regression and Classification Tasks

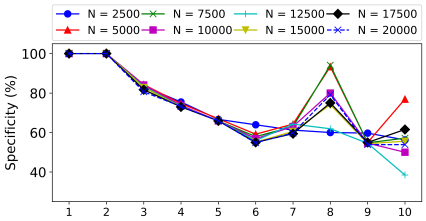
We performed the tasks of classification and regression on the *Mercer Cities*, *Metacritic Movies*, and *Metacritic Music Albums* datasets using the embeddings generated for DBpedia entities. We used SVM for classification and Logistic Regression for regression tasks. We measured *accuracy* for classification tasks and *root mean squared error* (RMSE) for regression tasks. The numeric values shown in Table 4 can be interpreted as factor of improvement ($lift$) when $Specificity^H(\beta = 0.25)$-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,



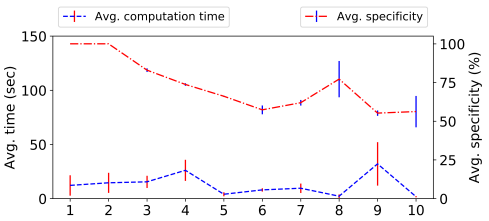
(a) Specificity variations - *dbo:Film*



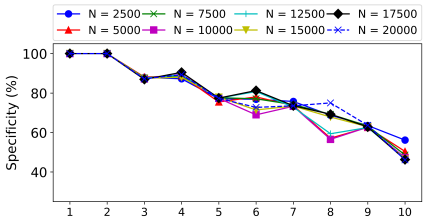
(b) Specificity vs. computation time variations - *dbo:Film*



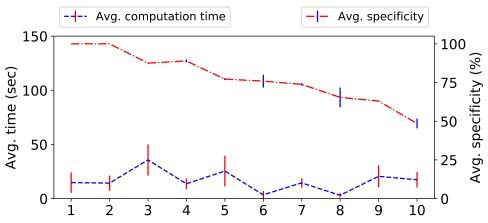
(c) Specificity variations - *dbo:Book*



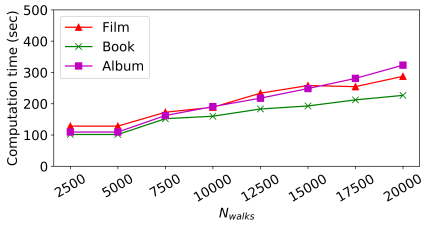
(d) Specificity vs. computation time variations - *dbo:Book*



(e) Specificity variations - *dbo:Album*



(f) Specificity vs. computation time variations - *dbo:Album*



(g) Total computation time of top-10 semantic relationships

Fig. 12. Effect of N_{walks} on computation of specificity

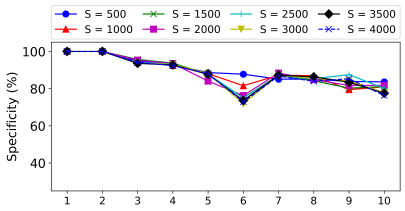
proving that specificity-based embeddings can be suitable for data mining tasks.

6.6. Parameter Sensitivity

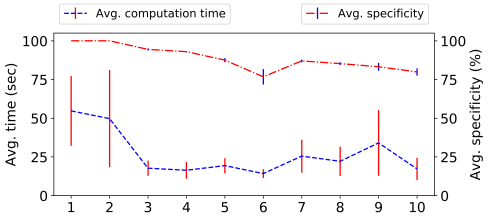
The algorithms for computing specificity use bidirectional random walks, governed by two parameters: number of bidirectional walks (N_{walks}) and size of seed set S. In this section, we evaluate the sensitivity of specificity to both of these parameters.

6.6.1. Sensitivity of Specificity to N_{walks}

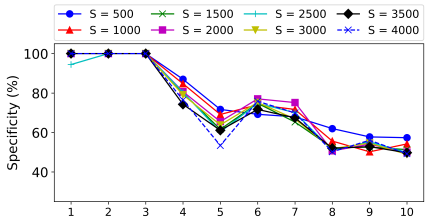
The algorithm for computing specificity uses the parameter N_{walks} for the number of bidirectional walks. To understand the effect of this parameter on specificity, we first computed specificity for $N_{walks} \in [2500, 20000]$. Figures 12a, 12c, and



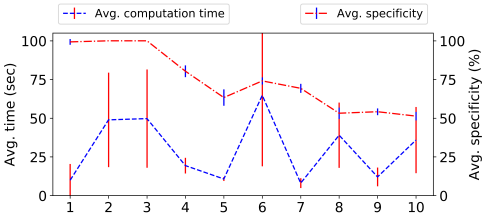
(a) Specificity variations - *dbo:Film*



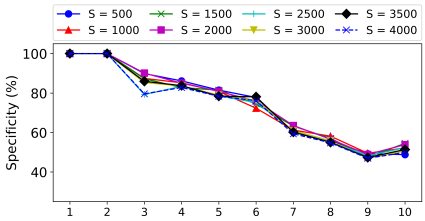
(b) Specificity vs. computation time variations - *dbo:Film*



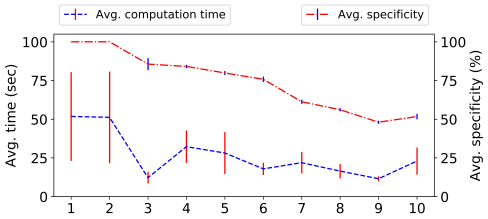
(c) Specificity variations - *dbo:Book*



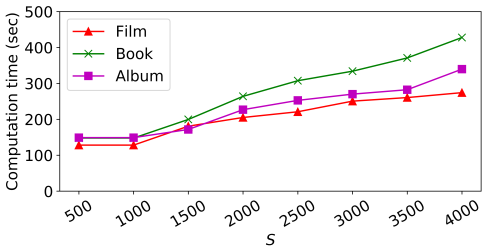
(d) Specificity vs. computation time variations - *dbo:Book*



(e) Specificity variations - *dbo:Album*



(f) Specificity vs. computation time variations - *dbo:Album*



(g) Total computation time of top-10 semantic relationships

Fig. 13. Effect of $|S|$ on computation of specificity

12e show the comparison between specificity scores computed for different values of N_{walks} for top-10 semantic relationships for entity type *dbo:Film*, *dbo:Book*, and *dbo:Album*. The semantic relationships are sorted on the x-axis (from left to right) in order of decreasing specificity computed based on $N_{walks} = 2500$ for comparison. Figure 12g shows the total time taken to compute specificity for the top-10 semantic relationships for each value of N_{walks}. The computation time increases with increasing number of bidirectional walks. Figures 12b, 12d, and 12f show the average specificity and corresponding computation time for each semantic relationship for all values of N_{walks} for the three chosen entity types. 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 generally large variations in computation time do not lead to significant changes in the specificity scores. Choosing 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 $N_{walks} = 5000$ as a suitable value for computing specificity, it can be seen from Figure 12g that the time for computing specificity scores w.r.t either entity-type is $\leqslant 150s$. Moreover, this computation is only needed to be performed once for each type of target entities in a KG.

6.6.2. Sensitivity of Specificity to $|S|$

Similar to previous discussion for N_{walks}, Figures 13b, 13d, and 13f 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 choose $|S| \in [500 - 1000]$ as a suitable value for computing specificity, it will result in computation time of $\approx 150s$ for computing specificity of top-10 most specific semantic relationships.

6.7. Analysis of Running Time of Specificity Computations

The specificity computations are performed by Algorithm 2. Algorithm 1 is used for setting up parameters and inputs for Algorithm 2. Since, the algorithm for computing specificity is based on random sampling governed by specific parameters, so instead of purely theoretical formulation here we provide a limited complexity analysis supplemented by empirical analysis of the running time of the algorithm.

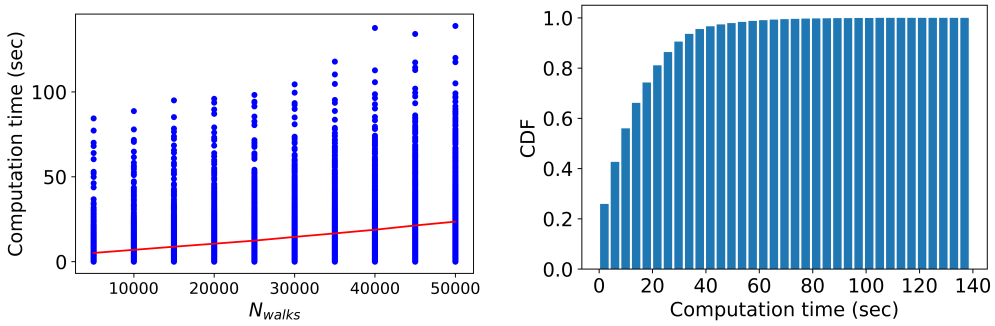
Algorithm 2 computes specificity for a list of semantic relationships provided as input parameter Q. The complexity of Algorithm 2 as presented is the complexity of computing specificity of a single semantic relationship multiplied by the size of Q. Therefore, we analyze the complexity of computing specificity of a single semantic relationship which is accomplished in Lines 5-17. Lines 6-16 repeat N (or N_{walks}) times, which is the number of bidirectional walks. In each iteration, the Algorithm 2 performs a forward walk in G and G^r ($reverse(G)$) or a forward and then a reverse walk in G (Lines 8 and 9). After completing the bidirectional walk, a node v' is identified in Line 9. Lines 10-15 determine if v' has the type t or some other type in the class hierarchy, and the count is updated accordingly.

There are two distinct tasks being performed in each iteration of the loop (Lines 6-16): (i) a bidirectional walk starts from a random node s of type t and lands at a node v' and (ii) determining the position of the type of v' in the class hierarchy. Assuming a tree-like class hierarchy where the height of the tree is H, the running time of the Lines 6-16 can be represented as $O(Nn_1 + NHn_2)$ where n_1 and n_2 correspond to the two tasks described above. Tasks n_1 and n_2 require information that is gathered by issuing SPARQL queries in our implementation. Therefore, the complexity of these tasks depends on how the query engine of Virtuoso optimizes the query execution. Instead of further expanding the relation for complexity mathematically, we treat the query engine as a black box and measure the query execution times of the issued SPARQL queries. In our implementation, we compute specificity using three sets of SPARQL queries, each of which performs one of the tasks below:

1. Starting from the seed set S, get a set V of N number of nodes using the semantic relationship of depth d, i.e., N iterations of Lines 7-8 of Algorithm 2.
2. Starting from the acquired set V, perform N number of reverse walks to get a set V' using any arbitrary path of depth d, i.e., N iterations of Line 9 of Algorithm 2.
3. Determine type of each $v' \in V'$ and update count in Line 12 accordingly.

Table 5 shows the distribution of classes at different heights in the class hierarchy of DBpedia. Level 0 corresponds to *dbo:Thing*, the root of the class hierarchy. The maximum height of the tree H is 8. We replace H with a constant term and rewrite the running time relation for our DBpedia-based experiments as $O(N[n_1 + kn_2])$. Since the time complexity of tasks n_1 and n_2 depends on the query optimizations performed by the SPARQL query engine, therefore, we empirically measure the behavior of the term in the parentheses.

In the Figure 14a, each marker represents a single computation of specificity. We computed specificity for five classes (*dbo:Film, dbo:Book,*



(a) Computation time for specificity for different values of N_{walks}

(b) Distribution of computational time

Fig. 14. Empirical analysis of computation time of specificity

Table 5
Distribution of classes at different heights in DBpedia hierarchy

Height in the Class Hierarchy	Number of Classes
0	1
1	49
2	126
3	209
4	271
5	72
6	23
7	4

dbo:Album, *dbo:City*, and *dbo:Country*, depths $d \leqslant 3$, different seed set S sizes (100-1000), and $N_{walks} = [5000, 50000]$, resulting in 23908 data points. The red curve shows the average of computation time averaged against each value of N_{walks}. Figure 14b shows the distribution of computation times which shows 75% of all the data points shown in Figure 14a represent a computation time of less than $20s$ for DBpedia KG. Moreover, this computation is only needed to be performed once for each type of target entities in a KG.

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 nodes and edges in the neighborhood of each 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 bi-directional 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. Lösch, 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. Chelmis and V.K. Prasanna, Smart Oilfield SafetyNet - An Intelligent System for Integrated Asset Integrity Management, in: *SPE Annual Technical Conference and Exhibition (ATCE)*, 2018.

[25] M. Saeed, C. Chelmis, V.K. Prasanna, R. House, J. Blouin and B. Thigpen, Semantic Web Technologies for External Corrosion Detection in Smart Oil Fields, in: *SPE Western Regional Meeting, April 27-30, 2015, Garden Grove, California, USA*, 2015. doi:10.2118/174042-MS.

[26] M. Färber, F. Bartscherer, C. Menne and A. Rettinger, Linked data quality of dbpedia, freebase, opencyc, wikidata, and yago, *Semantic Web*, 1–53.

[27] A. Ismayilov, D. Kontokostas, S. Auer, J. Lehmann and S. Hellmann, Wikidata through the eyes of DBpedia, *Semantic Web* **9**(4) (2018), 493–503.

[28] 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.

[29] 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.

[30] 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, Califor-*

nia, USA., 2017, pp. 1236–1242. http://aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14279.

[31] 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.

[32] U. Lösch, 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.

[33] 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.

[34] 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.

[35] 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.

[36] 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.

[37] 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.

[38] 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.

[39] 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.

[40] B. Heitmann and C. Hayes, Using Linked Data to Build Open, Collaborative Recommender Systems., in: *AAAI spring symposium: linked data meets artificial intelligence*, Vol. 2010, 2010.

[41] P. Lops, M. De Gemmis and G. Semeraro, Content-based recommender systems: State of the art and trends, in: *Recommender systems handbook*, Springer, 2011, pp. 73–105.

[42] 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.

[43] 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.

[44] 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.

[45] M.R. Saeed, C. Chelmis 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.

[46] 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.

[47] 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.

[48] M.R. Saeed, C. Chelmis and V.K. Prasanna, Not all Embeddings are created Equal: Extracting Entity-specific Substructures for RDF Graph Embedding, *CoRR* **abs/1804.05184** (2018). http://arxiv.org/abs/1804.05184.

[49] 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.