Manuscript Details

Manuscript number	FGCS_2019_3100
Title	A Distributed Affinity-Preserving Random Walk Strategy for Instance Matching on Knowledge Graphs

Abstract

Instance Matching (IM) is the process of matching instances that refer to the same real-world object (e.g., the same person) across different independent Knowledge Bases (KBs). This process is considered as a key step, for instance, in the integration of KBs. In this paper, we propose SBIGMat, a novel approach for the IM problem based on Markov random walks. Our approach bears in mind the local and global information mutually calculated from a pairwise similarity graph. Precisely, we first build an expanded association graph consisting of pairs of IM candidates. Then, we rank each candidate pair through the stationary distribution computed from the Markov random walk on the association graph. We propose semantic and bipartite graph-based post-processing strategies that operate on the obtained random walk ranks to optimize the final assignment of co-referents. We provide a scalable distributed implementation of our approach on top of the Spark framework and we evaluate it on benchmark datasets from the instance track of the Ontology Alignment Evaluation Initiative (OAEI). The experiments show the efficiency and scalability of SBIGMat compared to several state-of-the-art IM approaches.

Submission Files Included in this PDF

File Name [File Type]

Highlights.pdf [Highlights]

paper_sbigmat.pdf [Manuscript File]

To view all the submission files, including those not included in the PDF, click on the manuscript title on your EVISE Homepage, then click 'Download zip file'.

Highlights

We propose a random walk-based approach for instance matching in knowledge graphs.

We build an expanded candidates association graph consisting of pairs of IM candidates.

We transform the intance matching problem into a node ranking and selection problem.

We propose semantic and bipartite graph-based strategies to optimize the assignments.

Experimental results on benchmark datasets show the superiority of our approach.

A Distributed Affinity-Preserving Random Walk Strategy for Instance Matching on Knowledge Graphs

Ali Assi[a], Wajdi Dhifli[b,*]

[a] *University of Quebec At Montreal 201, av. President-Kennedy, Montreal (Quebec) H2X 3Y7, Canada*
[b] *Univ. Lille, EA 2694, 3 rue du Professeur Laguesse, F-59000 Lille, France*

Abstract

Instance Matching (IM) is the process of matching instances that refer to the same real-world object (*e.g.*, the same person) across different independent Knowledge Bases (KBs). This process is considered as a key step, for instance, in the integration of KBs. In this paper, we propose SBIGMat, a novel approach for the IM problem based on Markov random walks. Our approach bears in mind the local and global information mutually calculated from a pairwise similarity graph. Precisely, we first build an expanded association graph consisting of pairs of IM candidates. Then, we rank each candidate pair through the stationary distribution computed from the Markov random walk on the association graph. We propose semantic and bipartite graph-based post-processing strategies that operate on the obtained random walk ranks to optimize the final assignment of co-referents. We provide a scalable distributed implementation of our approach on top of the Spark framework and we evaluate it on benchmark datasets from the instance track of the Ontology Alignment Evaluation Initiative (OAEI). The experiments show the efficiency and scalability of SBIGMat compared to several state-of-the-art IM approaches.

Keywords: Affinity-preserving random walk, Data linking, Instance matching, Knowledge graph, Web of data.

1. Introduction

In recent years, the amounts and availability of openly accessed data have witnessed an impressive growth. Combined with the advances in algorithmic techniques for information extraction, this have facilitated the design and structuring of information giving rise to large-scale Knowledge Bases (KBs) in a "Big Data" context. By now, the Linked Open Data (LOD) cloud comprises 1239 datasets structured as Knowledge Graphs (KGs) and covering several domains such as geography, biology, *etc.* These knowledge graphs are considered as centralized repositories represented as Resource Description Framework (RDF). They store billions of facts about entities (*e.g.*, persons, locations), denoted by Uniform Resource Identifiers (URIs), their attributes and interdependence relationships. However, These knowledge graphs are often created independently from each other. Thus, they may introduce entities (with distinct descriptions) that are co-referring to the same entity in the real-world. Yet, such a connection is not explicitly defined. By establishing semantic links between entities described in these different KBs, intelligent agents are able to navigate between them as if they operate on a local integrated database. As a result, a richer and more enriched information is provided in response. *Instance Matching* (IM) [1, 2] is defined as the process of

*Corresponding author
Email addresses: assi.ali@courrier.uqam.ca (Ali Assi), wajdi.dhifli@univ-lille.fr (Wajdi Dhifli)

establishing a specific kind of semantic links called the identity link. The latter is expressed by the "*owl:sameAs*" property of the OWL (Web Ontology Language) vocabulary. It allows to explicitly link two instances that refer to the same entity in the real-world.

Example 1. *Fig. 1 shows two distinct KGs: DBpedia and Freebase. These KGs contain co-referent instance pairs*[1]*. For instance,* `dbpedia:Stanley_Kubrick` *and* `fbase:m.06mn7` *refer to the american film director "Stanley Kubrick". Similarly,* `dbpedia:Manhattan` *and* `fbase:m.0cc56` *refer to the american city "Manhattan". In this context, the aim of the IM process is to detect that each of these pairs is co-referent and thus link them through an identity link owl:sameAs.*

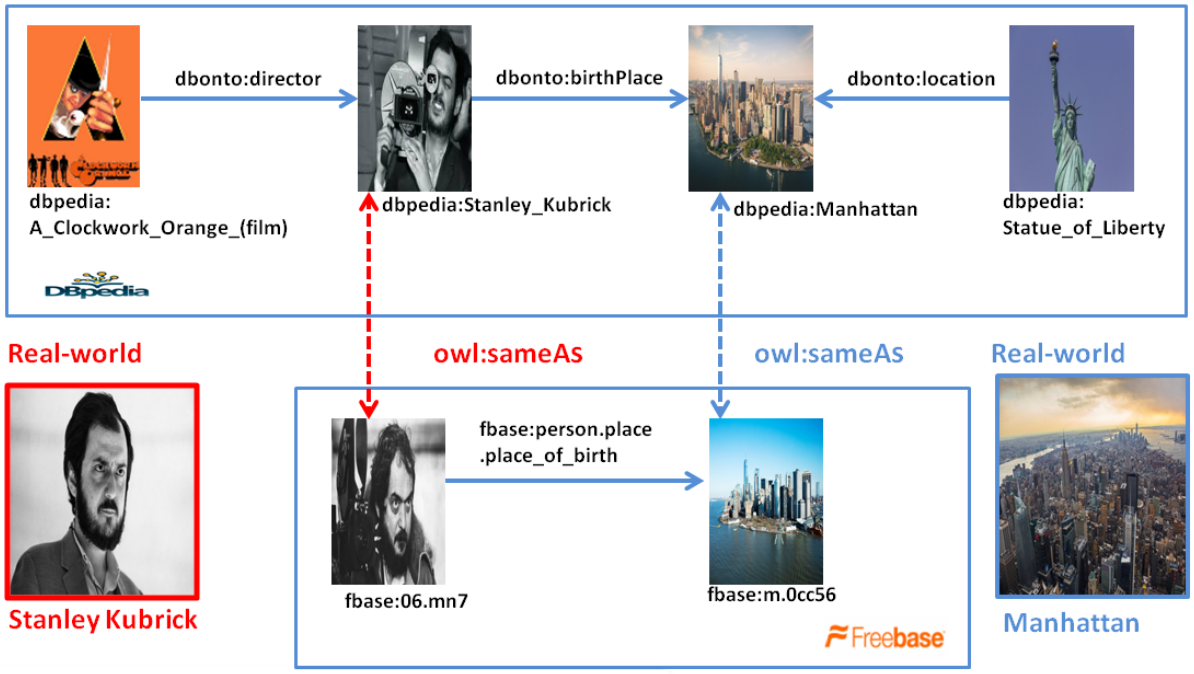

Figure 1: An illustrative example of the instance matching process. The images are given for simplicity and clarity of the example. In reality, each is given through a URI.

Graphs provide a powerful tool to represent complex data [4]. They are widely used in data modeling in order to formulate a structure description of resources (as in the RDF data model in this thesis). The aim of instance matching problem is to find one-to-one correspondences among "nodes" of a source and a target (KB) graphs. In terms of graph theory, this correspondence in known as graph matching problem [5] which is a critical problem in computer science. This problem can be seen as a quadratic assignment problem (QAP) [6] which is an NP-complete [7, 8]. Indeed, the graph matching also is an NP-complete problem [9]. Thus, only approximate methods can be defined and exploited to find the matching in a polynomial computational complexity [10]. Indeed, in IM, processing large-scale KBs can be very costly. As a result, besides the existing qualitative challenges, additional quantitative and scalability challenges are introduced. Parallel and distributed programming models (such as Spark [11], and Map-Reduce [12]) present important opportunities to improve the scalability of the matching process. Indeed, distributed architectures allow to fragment the instance matching process into several smaller sub-matching tasks that could

[1]The figure is adapted from [3]

be executed and resolved in parallel. Random Walk (RW) strategies present another important opportunity for solving the IM problem. This technique has long been used to solve a wide range of tasks presenting qualitative and quantitative challenges as in the IM problem. These tasks include outlier detection [13], web search [14], entity embedding [15], *etc.* The common idea behind RW-based techniques is to design the input data as a stochastic graph where the vertices usually correspond to the data objects. Then, a RW is executed on all the paths in the graph to deduce the importance (*i.e.*, rank) of the vertices. These ranks are then used to reconstruct the mapping between the nodes of the source and target graphs.

In this paper, we propose SBIGMat (**S**emantic and **B**ipartite **G**raph-based **I**nstance **Mat**ching), an approach for the instance matching problem based on the affinity-preserving RW. We represent the IM problem as a graph-based node ranking [14] and selection problem in a constructed candidates association graph. Our approach bears in mind the local and global information mutually calculated from a pairwise similarity graph.

In summary, SBIGMat gives the following contributions:

- We first build an expanded candidates association graph consisting of pairs (each as a graph node) of IM candidates. The edges between nodes reflect their pairwise structural harmony.
- We then rank each candidate pair through the stationary distribution vector computed from a Markov random walk strategy on that graph. Candidate pairs with higher rank scores in this vector are more likely to be co-referents. This novel random walk technique preserves the initial similarities of instance matching pairs.
- An absorbing node is also introduced in the candidates association graph in order to approximately separate nodes of estimated true candidates (inliers) from the false ones (outliers).
- We further present semantic and bipartite graph-based post-processing strategies to filter out the final set of co-referents among the ranked nodes of the candidates association graph.

We experimentally evaluate our approach on several KBs from the benchmark instance matching track of OAEI 2009 and 2010. We also compare it with a wide range of existing state-of-the-art IM techniques. Furthermore, we perform a scalabilty test on our spark-based implementation to show the potential speedup of SBIGMat on large-scale datasets when leveraging distributed computation resources. The obtained results show the efficiency and scalability of our approach.

The remainder of this paper is structured as follows. Section 2 gives the preliminary definitions of the RDF data model, the instance matching problem and the random walk technique. Section 3 describes the affinity-preserving random walk technique (Section 3.1) used for the instance matching task, as well as the association graph of candidates instance matching pairs (Section 3.2). Section 4 describe the semantic and bipartite graph-based post-processing strategies used in our approach. Section 5 depicts the workflow of our approach. Evaluation and experimental results on KBs from OAEI benchmarks are reported in Section 6. Section 7 presents and discusses existing related works from the literature. Finally, Section 8 concludes the paper and discusses our future work.

2. Preliminaries and Problem Statement

In this section, we provide the preliminary definitions and the basic terminology about the RDF data model. We also define the problem of "instance matching" that is targeted by this work.

2.1. RDF Data Model

The RDF data model [16] represents the descriptions of the resources (*i.e.*, the concepts and the instances[2]) by RDF expressions, called statements (*i.e.*, triples), in the form `<subject, predicate, object>`. A `subject` can be a URI or a blank node. The latter represents an anonymous resource. An `object` can be a URI, a blank node, or a basic value (*e.g.*, a string, a date, an integer, *etc*). A `predicate` allows to model a relationship between the `subject` and the `object`. When the `object` is a URI, the `predicate` is called an object-type property, and when it is a basic value it is called a data-type property. In regard to this model, the triples in the form `<subject, rdf:type, object>` declare that the `subject` is an instance of the `object` (*i.e.*, class).

Definition 1. ***(Instance matching)*** *Given two sets of instances $\mathcal{S}$ and $\mathcal{T}$ belonging to two KBs (KB_1 and KB_2), the aim of IM is to discover the set $\mathcal{M}$ of identity links owl:sameAs which are not already defined in the KBs. Formally, $\mathcal{M} = \{< i_1, owl{:}sameAs, i_2 >| \ (i_1, i_2) \in \mathcal{S} \times \mathcal{T}\}$ where $\mathcal{M} \not\subset KB_1 \cup KB_2$.*

Definition 2. ***(RDF knowledge graph)*** *An RDF knowledge graph is a set of facts in the form `<subject, predicate, object>` $\in (\mathcal{E} \cup \mathcal{B}) \times \mathcal{P} \times (\mathcal{E} \cup \mathcal{L} \cup \mathcal{B})$, where $\mathcal{E}$ is the set of instances, $\mathcal{B}$ is the set of blank nodes, $\mathcal{P}$ is the set of predicates and $\mathcal{L}$ is the set of literals (basic values).*

Alternatively, an RDF knowledge graph is a labeled multi-digraph $G = (V, E)$ where V is the set of nodes (including URIs and literals), and E is the set of edges which are URIs corresponding to predicates.

It is important to note that the IM task is different from the Ontology Alignment (OA) problem. The latter seeks to find a mapping between the concepts and properties of two ontologies, while the former is to determine a mapping between the instances of these ontologies. The majority of IM approaches exploit either the correspondences between the elements of the two ontologies, or use their common elements. These correspondences are determined either manually by an expert or automatically via an alignment tool. However, using such a tool in a big data context is highly expensive [17], and drastically affects the computational efficiency of the consequent IM approaches [18]. Even if the two KBs conform to the same ontology, the identity link detection remains an important and challenging problem.

Instance matching is a difficult task mainly due to textual variation of the property values, incompleteness, presence of erroneous information, multilingualism, *etc.*

2.2. Random Walks on Graphs

Consider a graph $G = (V, E)$ and an imaginary surfer s which starts at time t_0 a walk at an arbitrary vertex $X^{(0)} = x_0 \in V$. At time t_1, the surfer is located at $X^{(1)} = x_1$ picked uniformly at random from the 1-hope neighbors of x_0. By considering the time as discrete (*i.e.*, $t \in \mathbb{N}$), after N-time steps, the surfer will be at a vertex $X^{(N)} = x_n$. Thus, the sequence of visited vertices $x_0, x_1, \ldots, x_N$ is called a simple Random Walk in G [19]. Formally, uniformly at random means that $\forall x, y \in V$, $\forall t \in \mathbb{N}$, the probability to reach y from x is the following:

$$T_{xy} = P(X^{(t+1)} = y | X^{(t)} = x) = \begin{cases} \frac{1}{d(x)} & (x, y) \in E \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

where $d(x)$ is the degree of x.

[2] In this paper, instance and entity will be used interchangeably.

This way to define a RW is a specification of the discrete stochastic (*i.e.*, random) process. In general, when the surfer is at vertex x at time t, the neighbor y to move toward at time $t+1$ is chosen with a probability value that is proportional to the weight w_{xy} of the edge $(x, y) \in E$. This means that y is not chosen uniformly at random. Therefore, this RW is equivalent to a specific family of random process, that is a Markov chain [20]. Formally:

Definition 3. ***(Random walk)*** *A random walk is defined as a Markov chain with a discrete sequence of states (i.e., random variables) drawn from a discrete state space* $X = (X^{(n)})_{n\in\mathbb{N}} = (X^{(0)}, \ldots, X^{(n)})$ *and a given matrix* P *of transition probabilities:*

$$T_{xy} = P(X^{(t+1)} = y | X^{(t)} = x) = \begin{cases} \frac{w_{xy}}{\sum_{k\in V} w_{xk}} & \{(x, y), (x, k)\} \subseteq E \\ 0 & otherwise \end{cases} \tag{2}$$

The normalization that is either performed by $d(x)$ or by $\sum_{k\in V} w_{xk}$, $\forall (x, k) \in E$, is called "democratic normalization". The latter is equivalent to the notion of "Internet Democracy" in [21]. This means that each neighbor vertex of x can be reached with the same probability. Indeed, P can be converted to a row stochastic matrix (*i.e.*, $\forall x \in V, \sum_{y\in V} T_{xy} = 1, (x, y) \in E$) by $T = D^{-1}W$ where D is a diagonal matrix with $\forall i \in V$, $D_{ii} = \sum_{k\in V} w_{ik}$ and W is the adjacency matrix of G with $W_{ij} = w_{ij}$. Note that, the Markov chain is time-homogeneous (*i.e.*, T remains the same for any time $t \geq 0$).

The RW leads to a probability distribution $\vec{p} = (p(v))_{v\in V}$ at time t [21]. The value $p(v)$ reflects the likelihood that the vertex v is visited. This probability distribution is updated at each time step until it reaches (*i.e.*, converges to) a "stationary distribution", *i.e.*, the probability distribution does no change again. The stationary distribution of the RW will be determined by solving the following iterative equation $\vec{p}^{(t+1)^T} = \vec{p}^{(t)^T} \times T$, where $\vec{p}^T$ is a row vector, until the $\left\|\vec{p}^{(t+1)} - \vec{p}^{(t)}\right\|_{\ell-norm}$ becomes equal to 0 or less or equal to a predefined threshold.

Since the update is done by performing moves toward neighbors, the RW can go to a trap in a dead vertex (*i.e.*, a sink vertex). To avoid such a situation, an imaginary action (*i.e.*, jump) can be taken by the surfer when it is at a current vertex with a small "restart probability" α [22]. This new move allows the surfer to jump to any randomly selected vertex in G. Such a new action makes the transition matrix both *irreducible* and *aperiodic* (according to Perron-Frobenius theorem [23]). As a result, the existence and the uniqueness of the stationary distribution (*i.e.*, convergence) is guaranteed. This action can be illustrated by defining a "personalization vector" $\vec{v}$ (being of length $|V|$) which curbs the surfer to restart from a set of vertices, called seed(s). Only the seed vertices will get values different from zero in $\vec{v}$. In the RW scenario where the surfer can jump to any vertex in the graph, each value in $\vec{v}$ is initialized to $\frac{1}{|V|}$. Formally, the RW model can be represented by:

$$\vec{p}^{(t+1)^T} = (1 - \alpha) \times \vec{p}^{(t)^T} \times T + \alpha \vec{v}^T \tag{3}$$

where $\alpha \in [0, 1]$ is known as a damping factor (*i.e.*, restart probability) and is usually predefined as 0.15 or 0.1. Note that when the set of seed(s) is explicitly defined (*i.e.*, surfer jump back to a given set of starting vertices), the RW will be called Random Walk with Restart (RWR).

3. Random Walks for Instance Matching

As mentioned in the previous Section 2.1, the KBs are multi-digraphs. With respect to this consideration, the IM can be transformed to a graph matching problem between the source and target KBs given by $G^1 = (V^1, E^1)$ and $G^2 = (V^2, E^2)$, respectively. To solve this problem, we adopt the random walk technique to compute the similarities among the vertices of G^1 and G^2.

More precisely, to be able to run this technique, we create a stochastic graph called candidates association graph G^{rw} (rw stands for random walk) (see Section 3.2). The vertices of G^{rw} are simply the candidate pairs between G^1 and G^2. The weight (also known as affinity) of edges between the candidate pairs in G^{rw} are encoded in a matrix W (*i.e.*, affinity matrix) which can be transformed into T by $T = D^{-1}W$. Consequently, the values in the stationary distribution vector will be interpreted as vertices (*i.e.*, candidate pairs) ranking scores.

In principle, not all the vertices in G^{rw} represent true candidate pairs. Indeed, only the ones with "high" similarity values represent true candidate pairs. They are considered as inlier candidate pairs. The rest (*i.e.*, with "low" similarity values) represent false candidate pairs and are considered as outliers. Another important remark to mention here is that the number of outliers are bigger than the number of inliers. Thus, by detecting the outliers, the inliers constitute the bounded set of candidate pairs that includes the true matches. The outliers can be detected through the "stationary distribution" vector [24]. Intuitively, the rare a vertex $v \in G^{rw}$ is visited by the RW, the more probably that v is an outlier. In other words, the smaller is the score $p_t(v)$, the more presumably v is an outlier.

The affinity sum of the outgoing edges of an outlier vertex is usually smaller than that of an inlier vertex. Thus, the democratic normalization applied in this stochastic graph will leads to the following issues:

1. **Swamping problem:** the affinity sum of such an outlier vertex will be amplified making it be miss-classified as an inlier.
2. **Masking problem:** the affinity sum of an inlier vertex will be abated and, thus, it will be miss-classified as outlier.
3. **Ranking list:** as a result, the RW will generate a distorted probability distribution vector where only few vertices will have their right ranking values.

In our IM task, these issues will lead to several False Positives (FP) and a few True Positive (TP) matching pairs. We deduce that the democratic normalization is not suitable for our case and a new normalization method is required (See Section 3.1).

3.1. Affinity-Preserving Random Walk

The outliers are not known a priori by the surfer. Thus, the required normalization method should guide the surfer to distinguish between an outlier and an inlier vertex. This guidance can be illustrated by preserving the original affinity relations between the candidate IM pairs through a special stochastic graph called the "absorbing association graph" denoted by $G^{arw} = (V^{arw}, E^{arw})$. G^{arw} adds a special vertex to G^{rw} called "absorbing vertex", denoted by v_{abs}, and which constitutes a trapped vertex which once visited, the random walk cannot move out, *i.e.*, $T_{v_{abs}v_{abs}} = P(v_{abs}, v_{abs}) = 1$. Executing the RW on G^{arw} leads to a special Markov chain called "Affinity-Preserving Random Walk" (APRW) which can be defined as follows:

Definition 4. ***(Affinity-preserving random walk)*** *An affinity-preserving random walk on a graph $G^{arw} = (V^{arw}, E^{arw})$ is an absorbing Markov chain [25] with discrete sequence of states drawn from a discrete state space $X = (X^{(n)})_{n\in\mathbb{N}} = (X^{(0)}, \ldots, X^{(n)})$ and a given transition probabilities matrix T:*

$$T_{xy} = P(X^{(t+1)} = y | X^{(t)} = x) = \begin{cases} \frac{w_{xy}}{W_{max}} & (x,y) \in E^{arw} \\ 1 - \frac{\sum_{k\in V^{arw}} w_{xk}}{W_{max}} & \forall x \in V^{arw} \text{ and } y = v_{abs} \\ 0 & \forall y \in V^{arw} \text{ and } x = v_{abs} \end{cases} \quad (4)$$

where W_{max} is the maximum affinity weight in G^{arw}.

Note that $W_{max}=\|W\|_1=\max_{1\leq j\leq n}\sum_{i=1}^{n}|w_{ij}|$. Note also here that in practice v_{abs} does not need to be materialized in definition 4 and thus the graph $G^{arw}=(V^{arw},E^{arw})$ can be simply reduced to $G^{rw}=(V^{rw},E^{rw})$. The trick here is to find a way that prevents the surfer from moving out to another vertex once it is at an outlier vertex. In fact, the surfer will be redirected toward the absorbing vertex. In other words, once the surfer is at a current vertex in G^{arw} with a big affinity in G^{rw}, it will less probably move toward v_{abs}. The surfer will get an opposite behavior if its affinity sum is small in G^{rw}. It will more probably walk and be absorbed by v_{abs}. In this way, the inaccurate ranking scores of the outliers will be absorbed by the absorbing vertex. The new normalization of the affinity matrix W can be transformed to T by $T=D^{-1}\frac{W}{\|W\|_1}$. It is worth pointing here that some rows in T can get a sum less than 1. In the following, we adopt the formulation used in [26] for T:

$$T=\begin{pmatrix}\frac{W}{\|W\|_1} & \vec{1}-\frac{\vec{a}}{\|W\|_1}\\ \vec{0}^T & 1\end{pmatrix} \tag{5}$$

and for the absorbing Markov chain (*i.e.*, APRW):

$$\begin{pmatrix}x^{(t+1)^T} & x^{(t+1)}{}_{v_{abs}}\end{pmatrix}=\begin{pmatrix}x^{(t)^T} & x^{(t)}_{v_{abs}}\end{pmatrix}\times T \tag{6}$$

where $\frac{W}{\|W\|_1}$ is a $|V^{rw}|\times|V^{rw}|$ square matrix (called sub-stochastic matrix), $\vec{1}_{|V^{rw}|\times 1}$ denotes an all-ones vector with size $|V^{rw}|$, $\vec{0}^T_{|V^{rw}|\times 1}$ denotes an all-zeros vector with size $|V^{rw}|$ and $\vec{a}=(W_1,W_2,\ldots,W_n)^T_{1\leq n\leq|V^{rw}|}$ with $W_i=\sum_{j\in V^{rw}}w_{ij},\quad\forall i\in V^{rw}$. The distribution probability for an absorbing Markov Chain admits a fixed solution $\vec{p}=(\vec{0}^T1)$. It is trivial that $\vec{p}$ in this case cannot be used for vertex ranking as in PageRank [14]. To alleviate this problem, [26] defines the following conditional probability:

$$\bar{x}^{(t)}_{ia}=P(X^{(t)}=v_{ia}|X^{(t)}\neq v_{abs})=\frac{x^{(t)}_{ia}}{1-x^{(t)}_{v_{abs}}} \tag{7}$$

which refers to the distribution of unabsorbed vertex $X^{(t)}=v_{ia}\in G^{arw}$ at time t. Hence, the distribution probability where the surfer stays at time t is represented now by $\bar{x}^{(t)}$. The solution (*i.e.*, stationary distribution) of equation (6) is called "quasi-stationary distribution" denoted by the $(|V^{rw}|\times 1)$ $\bar{x}$ vector. It is obtained when $\bar{x}^{(t+1)}=\bar{x}^{(t)}$.

3.2. Candidates Association Graph

Vertices generation. We rely on the predicates values of the instances to build the IM candidate pairs (*i.e.*, vertices in the association graph). For this, purpose, we build for each dataset (with a Source (G^1) and a Target (G^2)) an inverted index. Each of them will have $<Token,\{i_1,i_2,\ldots,\}>$ as a schema where i is an instance in the considered dataset (we denote by s_i and t_j instances in G^1 and G^2, respectively). Then, we join both inverted-indexes on the common $Token$ to produce a table $Tab=<Token,<\{s_1,s_2,\ldots,s_n\},\{t_1,t_2,\ldots,t_m\}>>$. The IM candidate pairs with their initial similarities are then generated by transforming Tab to a new matrix I of size $n\times m$ where $I=<v_{ij},sim_{ij}>$ and $sim_{ij}=\frac{1}{\|s_i\|}\times\sum_{w\in s_i\cap t_j}TIF_1(w)\times TIF_2(w)$ is the similarity between s_i and t_j. TIF is a metric permitting to quantify the infrequency of a given word in the given dataset [27, 1]. $TIF(w)=\frac{1}{log_2(F(w)+1)}$ where $F(w)$ is equal to the number of instances containing w in their description.

Neighbors. Two instances s and o are considered as neighbors in a given KB graph if there is a statement between them (*i.e.*, $< s, p, o >$). In case where o is a blank node, we adopt the idea in our previous works [27, 1] to generate the neighbors of s. In fact, we consider all the leaf nodes of the k-hops paths starting from o and ending in an object node as neighbors of s.

Edges generation. We suppose that an edge exists between every two nodes in the candidates association graph. The weights of these edges are determined in a way that reflects the intuition that if two instances form a potential co-referent pair, then there is a high possibility that their neighbors are also potential co-referent pairs. Setting the values of the weights of these edges is detailed in the following paragraph.

CA-graph construction. Let us define $G^{rw} = (V^{rw}, E^{rw})$ as the association graph where $V^{rw} \subseteq G^1 \times G^2$ is the set of vertices and E^{rw} the set of edges between the vertices. The edges are weighted according to different schemas. Let us consider two nodes $v^k = (v_i, v_a)$ and $v^p = (v_j, v_b)$ in G^{rw}. If v_j is a neighbor of v_i in G^1 and v_b is a neighbor of v_a in G^2, then the weight of the edge $e = (v^k, v^p)$, denoted by $w_{ia;jb}$, is equal to $\exp^{\frac{sim_{ia}+sim_{jb}}{2}}$. Otherwise, it is equal to $\frac{1}{\mid V^{rw} \mid}$. The weights $w_{ia;jb}$ are the affinities that are encoded in the affinity matrix W. Algo. 1 shows the full algorithm for computing the values of the weights for the edges E^{rw} of G^{rw}. Note that the function *Neighbors* allows to identify the set of neighbors of a query node in its graph.

Algorithm 1: Computation of the weights values of the edges in the candidates association graph

Input : I: IM candidate pairs (V^{rw}), $G^1 = (V^1, E^1)$: a source dataset, $G^2 = (V^2, E^2)$: a target dataset and $G^{rw} = (V^{rw}, E^{rw})$: the CA-graph
Output: W: The affinity matrix of G^{rw}

$W \leftarrow 0$
foreach $v^k = (v_i, v_a) \in V^{rw}$ **do**
  $v_i \leftarrow$ the entity $v_1^k \in V^1$
  $v_a \leftarrow$ the entity $v_2^k \in V^2$
  if $v_j \in$ *Neighbors*(v_i, G_1) **then**
    foreach $v^p \in$ *Neighbors*(v^k, G^{rw}) **do**
      $v_j \leftarrow$ the entity $v_1^p \in V^1$
      $v_b \leftarrow$ the entity $v_2^p \in V^2$
      if $v_b \in$ *Neighbors*(v_a, G_2) **then**
        $w_{ia;jb} \leftarrow \exp^{\frac{sim_{ia}+sim_{jb}}{2}}$
foreach $w_{m;n} \in W$ **do**
  if $w_{m;n} = 0$ **then**
    $w_{m;n} \leftarrow \frac{1}{\mid V^{rw} \mid}$
return W

Optimization. A naive solution to build the candidates association graph can be done by taking all the different possible instances' pairs from the source and target datasets. Such a solution raises two main issues. Firstly, the seemingly large size of the compared KBs yields a scalability challenge whether on the number of nodes (*i.e.*, $n \times m$ where n and m are the total number of instances in source and target KBs, respectively) or the number of iterations until the random walk converges.

Secondly, a large number of false positive pairs can be comprised in the association graph yielding a large number of noisy edges between the nodes which may hinder the quality of the IM results. In iterative algorithms such as RW, this drawback increases in magnitude since the erroneous matching results will be propagated between the nodes over the iterations. A pruning step can highly alleviate these issues.

In order to reduce the number of nodes (*i.e.*, candidate pairs) in the association graph, it is possible to leverage the background knowledge about the used data to restrain the search space (*i.e.*, the number of nodes in the candidates association graph). Indeed, we rely on the instances' labels to reduce the number of candidate pairs, such that two instances $v_i \in V^1$ and $v_a \in V^2$ (where $G^1 = (V^1, E^1)$ and $G^2 = (V^2, E^2)$ resemble the graphs of a source and a target dataset, respectively) are directly considered as a match, if they have an identical unique label and there doesn't exist any other instances $\nexists v_p \in \{V^1_{\setminus\{v_i\}} \cup V^2_{\setminus\{v_a\}}\}$ that shares this label. As a result, we remove from G^{rw} all the nodes $v^k \in G^{rw}$ that include in their component v_i or v_a. The labels (*i.e.*, names) are used to refer to the instances in real world. They can be considered as a very meaningful information source. Intuitively, humans refer to the names as a first *key* to identify objects (*i.e.*, instances) in real world. In KBs, the labels could explicitly be defined by the KB's designer (for instance, through predefined predicates like `rdfs:label` and `foaf:name`) or not. To automatically determine the predicates that could act as labels (*i.e.*, most distinctive), we use three known metrics namely: discriminability, support (*i.e.*, coverage) and their harmonic average (*i.e.*, priority).

Let T, $\mathcal{P}$, $\mathcal{E}$ and $\mathcal{L}$, be the sets of statements, predicates, instances and literals in a KB, respectively. Let also T_p be the set of statements in T, $T_p \subseteq T$, with p as a data-type property, *i.e.*, $T_p = \{t =< s, p, o > \mid s \in \mathcal{E},\ p \in \mathcal{P},\ o \in \mathcal{L}\}$. Indeed, we refer to the set of subjects (*e.g.*, s) and objects (*e.g.*, o) where a predicate p appears by $sub(p)$ and $obj(p)$, respectively. Formally: $sub(p) = \bigcup\{s \mid t =< s, p, o >,\ t \in T_p\}$ and $obj(p) = \bigcup\{o \mid t =< s, p, o >,\ t \in T_p\}$.

Definition 5. ***(Discriminability)*** *The discriminability of a property is computed as the ratio of its range size on the number of triples that instantiated it. It measures its* `objects` *diversity. The discriminability of p is defined as:*

$$disc(p) = \frac{|obj(p)|}{|sub(p)|} \tag{8}$$

Definition 6. ***(Support)*** *The support of a property is determined as the ratio of the number of instances that instantiated it, on the total number of instances in the KB. It measures the instance-wise frequency of a property. The support of p is defined as:*

$$sup(p) = \frac{|sub(p)|}{|\mathcal{E}|} \tag{9}$$

Definition 7. ***(Priority)*** *The priority of a data-type property is determined as the harmonic mean of its discriminability and support. The priority of p is defined as:*

$$priority(p) = 2 \times \frac{sup(p) \times disc(p)}{sup(p) + disc(p)} \tag{10}$$

Intuitively, only properties p with high priority values will be used in the optimization phase. For non-unique labels' instances, we select for each source instance the 25% (up from the 3^{rd} quartile) most similar target instances (based on the similarity values computed as in section 3.2.a) that share at least one word with it on their local descriptions.

4. Post-processing Strategies

4.1. Bipartite Graph-based Post-processing

To extract the final set of co-referents from $\bar{x}$, it is possible to build a weighted bipartite graph $G^b = (V^1, V^2, E^b)$ where the rank $\bar{x}_{ia}^{(t)}$ of each vertex $v_{ia} \in G^{arw}$ is assigned to the edge $(v_i, v_a) \in E^b$. Then, the final set of co-referents can be obtained by adopting a post-processing optimization strategy including Stable Marriage Problem (SMP) [28], Hungarian algorithm (Kuhn-Munkres) [29], Symmetric Best Match strategy (SBM) [30] or simply by getting the Best Match (BM) among the targets for each source instance. Algo. 2 shows the procedure for performing a bipartite graph-based post-processing.

4.2. Semantic Similarity Strategy

It possible to find multiple nodes from the CA-graph having the same source instance and ranking score (*e.g.*, (v_i, v_j) and (v_i, v_k)). In these cases, the previously mentioned post-processing strategies in Section 4.1 randomly select one of the target instances among those that equally have the best ranks. To prioritize (*i.e.*, to select v_j or v_k) one node over the others, it is possible to take advantage of a semantic similarity strategy. We will use the semantic similarity described in [1]. To compute the semantic similarity between two given instances, we first build their corresponding BOWs (VD_1 and VD_2) as in [1] and create a combined list of unique vocabulary, denoted by L, *i.e.*, $L = VD_1 \cup VD_2$. Then, we compute the pairwise similarity of each word v_L in L with every word v_1 in VD_1. This leads to create a vector $\vec{V_1}$ that contains the maximum similarities between each word in L and all words in VD_1 (respectively $\vec{V_2}$ for VD_2). Formally, $\vec{V_1}$ is defined as:

$$\vec{V_1} = \max_{1 \leq i \leq |VD_1|} Sim(v_{1i}, v_{Lj}) : \forall v_{Lj} \in L \tag{11}$$

$$\vec{V_2} = \max_{1 \leq i \leq |VD_2|} Sim(v_{2i}, v_{Lj}) : \forall v_{Lj} \in L \tag{12}$$

where Sim (we use the cosine similarity) computes the similarity between two words based on their embedding vectors. These embedding vectors could be obtained by training a semantic model (such as FastText [31]) on a big text corpora (*e.g.*, Wikepedia, DBpedia, *etc.*). This allows us to take into account words semantics. Once $\vec{V_1}$ and $\vec{V_2}$ are determined, we compute the cosine similarity between them.

Example 2. *In Fig. 2, the cosine similarity between the embedding vector of "auto" in L and the embedding vector of "new" in VD_1 is equal to "0.28". Similarly, the cosine similarity between "auto" and "car" in VD_1 is equal to* 0.8*. Thus, the resulting similarity value for "auto" in L has the highest score of* 0.8 *with the embedding of "car" in VD_1, i.e., $v_{13} = \max(0.28, 0.8) = 0.8$. In a similar way, the semantic distance vector $\vec{V_2}$ of VD_2 is computed.*

5. Approach overview of SBIGMat

The workflow of our approach is depicted in Fig. 3. In the first step, we determine the candidate instance pairs (*i.e.*, nodes) based on the common token in their local descriptions. These nodes are refined by the unique label heuristic defined at the end of Section 3.2. Then, based on the join of both inverted indexes, we determine the vertices of the candidates association graph and then we create the edges between them. The affinity-preserving random walk phase allows to compute the "quasi" distribution probability vector which represents the final nodes' ranks. The final step (*i.e.*, co-referents assignment) permits to extract the one-to-one match between the source and target instances by applying a post-processing strategy as detailed in Section 4. We term this hybrid approach SBIGMat.

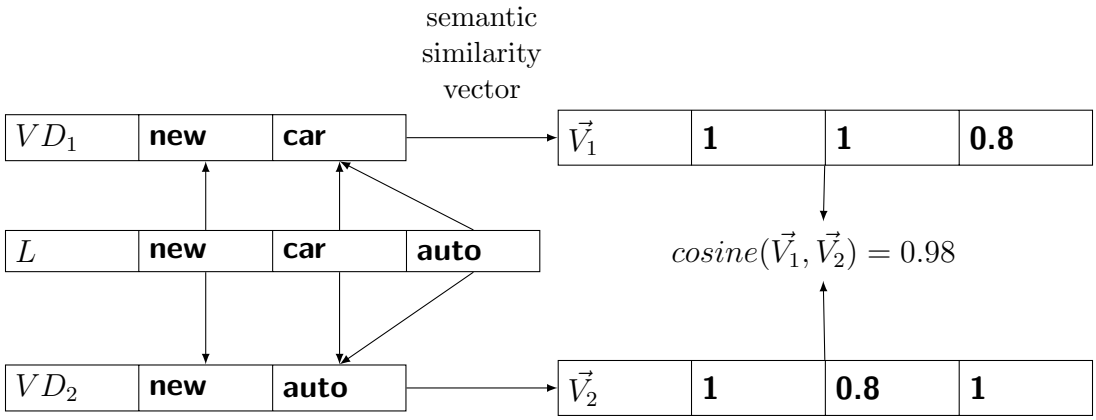

Figure 2: An example of computation of the semantic similarity between the virtual documents of two instances.

Algorithm 2: Bipartite graph-based post-processing

Input : $\bar{x}$: Quasi distribution probability vector, $G^1 = (V^1, E^1)$: a source dataset, $G^2 = (V^2, E^2)$: a target dataset

Output: M: the final set of co-referents

$E^b \leftarrow \emptyset$
foreach $v^k \in V^{rw}$ **do**
  $v_i \leftarrow$ the entity $v_1^k \in V^1$
  $v_a \leftarrow$ the entity $v_2^k \in V^2$
  $w_{ia} \leftarrow \bar{x}_{ia}$
  $E^b \leftarrow E^b \cup \{(v_i, v_a, w_{ia})\}$
$M \leftarrow postprocessing(V^1, V^2, E^b)$
return M

6. Experimental Evaluation

6.1. Datasets

We evaluate our approach on three benchmark datasets: PR (synthetic), DI (real) and A-R-S (real). The first two benchmarks are used in OAEI 2010 while A-R-S is used in OAEI 2009. Table 1 reports the statistics about the considered benchmarks. From PR, which is a small benchmark, we choose only the restaurants RDF datasets. The latter includes structural and value modifications only. DI includes four RDF subsets named *Sider*, *DrugBank*, *Diseasome* and *Dailymed* related to health and drug domains. A-R-S includes three RDF files named *eprints*, *rexa* and *dblp* related to scientific publications RDF data. This benchmark encompasses very noisy values' data as well as ambiguous labels (authors names and paper titles). The structure of these datasets includes blank nodes. Although the latter include valuable additional information about the subjects they are usually ignored by IM approaches. Incorporating this information constitutes another challenge for the matching process. For each set, the OAEI provides a mapping file ("gold standard") including the co-referent pairs between the source and the target RDF files. PR benchmark provides a complete gold standard that allows an accurate IM evaluation.

6.2. Experimental Setup

Our approach is implemented in Python 3.5 and Spark 2.3. To conduct our experiments, we deploy a Linux cluster of 5 nodes. Each node has 128 GB RAM and an Intel E5-2683 v4 "Broadwell"

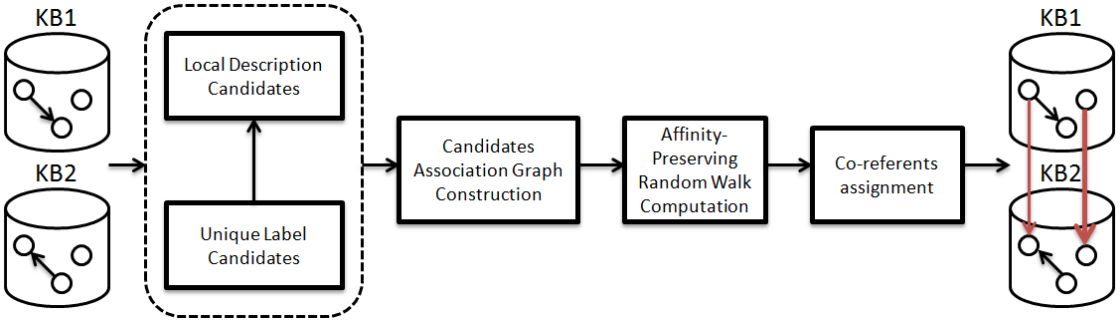

Figure 3: General architecture of SBIGMat.

Benchmarks	ID	Datasets	Source	Target	Gold Standard
PR	**D1**	Restaurant1-Restaurant2	399 URIS	2256 URIs	112
A-R-S	**D2**	eprints-rexa	1130 URIS	18,492 URIs	777
	D3	eprints-dblp	1130 URIs	2, 650, 832 URIs	554
	D4	rexa-dblp	18, 492 URIs	2, 650, 832 URIs	1540
DI	**D5**	Sider-Diseasome	16187 URIs	20064 URIs	238
	D6	Sider-Dailymed	16187 URIs	12515 URIs	349

Table 1: Description of the experimental benchmark datasets.

CPU of 2.1 Ghz and 32 cores. Indeed, we allow the paths to be composed at most of 2 blank nodes when we determine the neighbors in the candidates association graphs as described in Section 3.2. Following [32], we set up the predefined threshold for random walk convergence to be 10^{-6}. Indeed, we fix the maximum number of iterations of the random walk to be 300. For word embedding vectors, we used a dataset trained on Wikipedia corpus using FastText [31]. If a word does not exist in the pre-trained embeddings, a unique vector will be created at test time.

6.3. Baselines

We compare SBIGMat with several IM systems which we categorize into three blocks:

- *Supervised approaches:* These approaches require training data and sometimes the intervention of a domain expert. In this category, we compare with AdaBoost [33] and cLink [34].
- *Unsupervised approaches:* Here, the approaches do not go through a training phase but they use knowledge declared in an ontology or provided by an expert. In this category, we compare with SERIMI [35], PARIS [2], VMI [36], SIGMa [37], RiMOM [38], HMATCH(I) [39], DSSIM [40] and FBEM [41].
- *Semi-supervised approaches:* In this category, the methods self-learn [42] by labeling unlabeled examples and extend the labeled training set. In this category, we compare with ObjectCoref [43].

6.4. Results and Discussion

Analysis the effect of the affinity-preserving random walk on the IM. We first analyze the effect of the affinity-preserving random walk on the accuracy of the IM process. Table 2 reports the obtained results for our approach with the APRW (SBIGMat (APRW)) and with the traditional random walk (SBIGMat (RW)). For both SBIGMat (RW) and SBIGMat (APRW), we use

BM for post-processing. This means that for each source instance, we assign the target instance having the best rank. We notice that APRW permits to avoid the drawback of the democracy internet problem in the IM task. This new variant of random walk allows to minimize the effect of low quality data (*i.e.*, the case of eprints-rexa) and therefore to increase the accuracy of the IM process. As a result, the F-measure over the considered datasets are improved approximately between 1% to 4%.

Datasets	**D1**	**D2**	**D3**	**D4**	**D5**	**D6**
SBIGMat(RW)	0.99	0.83	0.79	0.92	0.87	0.76
SBIGMat(APRW)	**1**	**0.87**	**0.81**	**0.94**	**0.88**	**0.78**

Table 2: F-Measure of SBIGMat using random walk (RW) or affinity-preserving random walk (APRW).

Effect of the co-referents assignment (post-processing) method on the IM. In this section, we study the effect of different post-processing strategies on the quality of the obtained co-referents results. We will test different methods with SBIGMat(APRW) and the best method will be used in the rest of this paper. We test the assignment methods: BM, SMP [28], Kuhn-Munkres [29] as well as the semantic similarity (Semantic), previously detailed in Section 4. Table 3 shows the obtained results. We can see that, overall, the Semantic post-processing strategy allows to obtain the best F-measure results. Indeed, by using the semantics of words in the descriptions of the instances, SBIGMat was able to unravel for source instances the most semantically similar target instances among those equally having the top APRW ranks.

Approach\Dataset	**D1**	**D2**	**D3**	**D4**	**D5**	**D6**
BM	**1**	0.87	0.81	0.94	**0.88**	0.78
Semantic	**1**	**0.88**	**0.82**	**0.95**	**0.88**	0.78
SMP	**1**	**0.88**	0.81	0.94	0.84	**0.79**
Kuhn-Munkres	**1**	0.85	0.80	0.93	0.83	0.78

Table 3: F-Measure results of SBIGMat using APRW and different one-to-one assignment strategies.

Comparative analysis with state-of-the-art approaches. In the following, we report the results of multiple state-of-the-art IM approaches on the used benchmark datasets from PR, A-R-S (Table 4) and DI (Table 5) and we compare our approach against them. SBIGMat achieves better results compared to the other approaches. By analyzing the false matching pairs obtained on the A-R-S benchmark, we noticed that several of these instances in A-R-S were isolated nodes in the original RDF graph and thus their neighborhood in the candidate association did lack inlier nodes which can boost their ranks. As for DI benchmark, we noticed that instances with very similar descriptions exist too. These instances also possess highly similar neighbors which have by their turn recursively similar descriptions. This scenario is challenging for IM approaches in general.

Scalability Analysis. In Fig. 4, we report the scalability test of SBIGMat on three datasets from OAEI 2009 and 2010. This test shows the effect of the number of used CPU cores, in the deployed cluster, on the running time and the speedup of SBIGMat. In each sub-figure (in Fig. 4), the left (respectively right) vertical axis records the running time (respectively speedup). In principle, Spark creates a task for each partition of the initial data and runs the obtained tasks on

Datasets	D1	D2	D3	D4
SBIGMat(Semantic)	**1**	**0.88**	0.82	0.95
VDLS [1]	0.98	0.87	**0.90**	**0.97**
RIMOM [38]	0.81	0.80	0.73	0.73
PARIS [2]	0.91	-	-	0.91
VMI [36]	-	0.85	0.66	0.76
SIGMa [37]	0.96	-	-	0.94
DSSIM [40]	-	0.38	0.13	-
HMATCH(I) [39]	-	0.62	0.65	0.45
FBEM [41]	-	0.18	0.28	0.21

Table 4: Results of the compared approaches on PR and A-R-S benchmarks

Datasets	D5	D6
SBIGMat(Semantic)	**0.88**	**0.78**
VDLS [1]	0.81	0.76
RIMOM [38]	0.46	0.63
SERIMI [35]	0.87	0.66
AdaBoost [33]	0.79	0.73
PARIS [2]	0.11	0.15
ObjectCoref [43]	0.74	0.71
cLink[34]	0.82	**0.78**

Table 5: Results of the compared approaches on DI benchmark

the cluster. The best practice use of Spark recommends 2 to 3 tasks per CPU core in the cluster. As a result, the number of tasks will be equal to the number of the available cores (in the cluster) multiplied by the number of executed tasks per CPU core. In our experimentation, we set it to 3 and we fix the number of tasks among all the datasets tests.

The results in Fig. 4 show that SBIGMat benefits from the increased number of the available cores. Its running time was shortened according to the number of CPU cores used in each experimentation. On the small restaurant datasets, SBIGMat finished in 40 seconds. This running time is due to the overhead of the switch of tasks initialized in Spark platform. This case shows that Spark may hinder the running time on small datasets. In contrast, for large-scale datasets as in rexa-dblp, our approach records 6 minutes as running time while VMI took 19min52s, RIMOM 36h34min, DSSim 20h32min, respectively. For sider-dailymed datasets, SBIGMat took 4min3sec to finish his job. To the best of our knowledge, none of the state-of-the-art approaches that we compare with on sider-dailymed report its running time.

7. Related work

In the literature, a myriad of IM approaches have been proposed. Recently and thanks to the availability of distributed and parallel programming models (such as Spark [11], and Map-Reduce [12]), several IM approaches have been introduced to handle large scale KBs. In fact, using Spark, the work in [44] applies bloom filters to an encrypted data in order to link disparate socioeconomic and health-care databases. A parallelization of the adaptive windowing Sorted Neighborhood blocking method with the data load balancing problem among the executors are addressed in the

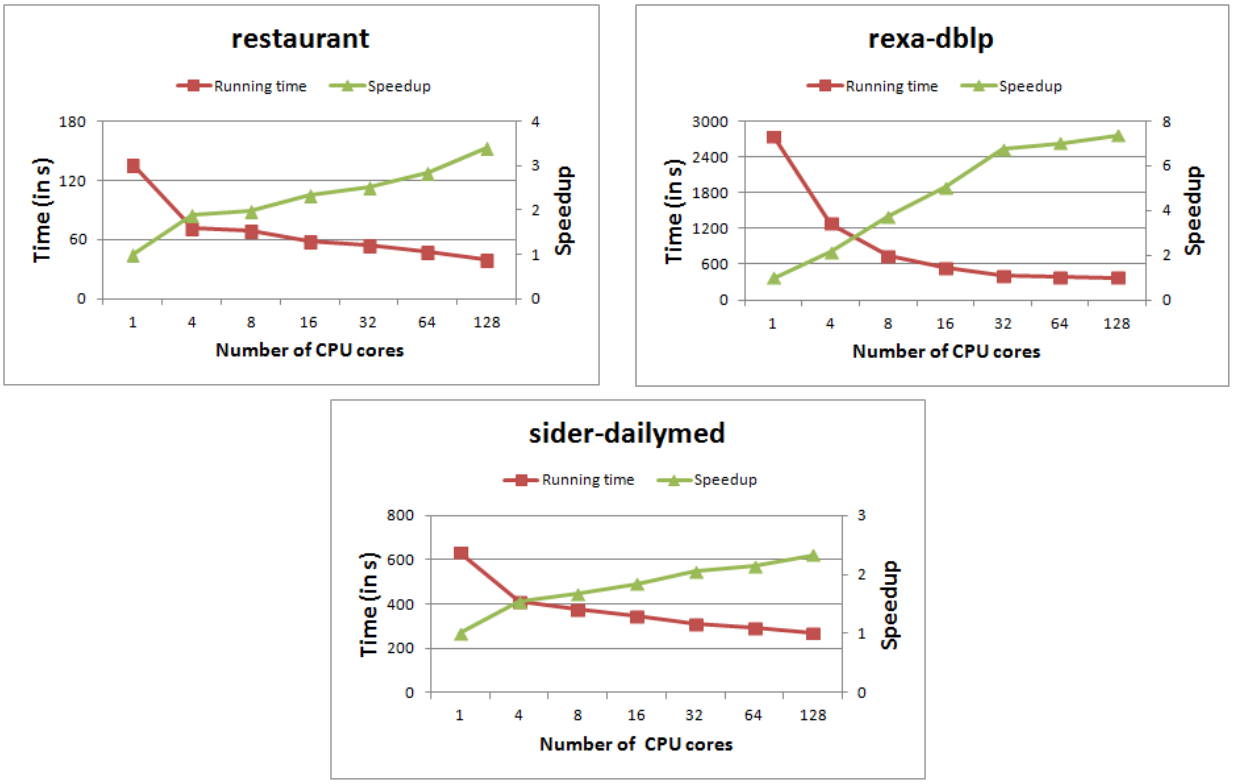

Figure 4: Scalability test of SBIGMAT. The left vertical axis shows the running time of the approach and the right vertical axis shows the respective speedup.

S-DCS++ approach [45]. A Spark-based kNN classification approach is aborted in [46] to carry out with the unbalanced distribution of duplicate and non duplicate of record pairs in the dataset. MinoanER [47] implements a Spark-based approach which relies on a blocking scheme defined as a disjunction of different evidences coming from the values, names, and neighbors of the candidate instances. Four threshold-free heuristics are applied on these blocks to filter them in order to discover identity pairs. HolisticEM [48] is a Map-Reduce based approach that constructs a similarity graph from the candidate pairs, generated based on the overlap across their attributes and their neighbors. The local and global information from the similarity graph are propagated using PPR to update and compute the final similarity scores for the candidate pairs.

Other sequential-based approaches for IM exist. In the following, we cite some of the approaches that fall in this category. SIGMa [37] starts by identifying similar instances having identical instances names. Then, it uses them as seeds to iteratively propagate the matching from already detected identical instances to their neighbors. SERIMI [35] starts by selecting for each instance a set of candidates by applying a direct matching between the value of the most discriminating property (in the instance's class) and the instances in the target KB sharing that value. Then, it applies class disambiguation to capture for each source instance its co-referent. For more details about IM approaches, we refer the reader to [3, 49, 50].

More recently, with the advances in deep learning, several knowledge graph embedding-based models are proposed for IM. Initially, these models have been developed in order to improve the completeness of knowledge graph (in terms of missing instances or relations) by leveraging the existing triples. MTransE [51] and MTransE-Af [52] learn an embedding space for each KG separately and propose to learn a transition matrix, based on large seed instance mappings, to map the em-

bedding space from one KG to the other. IPTransE [53] and BootEA [54] are two self-training methods which embed two KGs in a unified space and iteratively identify new identity links. These methods align entities mostly based on the relationship triples (*i.e.*, 1-hop relational neighbors) in a KG. JAPE [55] and AttrE [56] complement the embedding of the relational structure of KGs by incorporating the embedding of the attributes or the attributes' values (*i.e.*, more features) to refine the instances embeddings and project them into the same vector space. The similarity of entities is measured by the distance of entity embeddings. Note that, IPTransE [53] and AttrE [56] assume the existence of a seed relation (*i.e.*, object-type property) and attribute (*i.e.*, data-type property) alignment before generating the instances embedding. However, the seed alignments between two KGs are rarely available and are hard to obtain. Indeed, generating the relations and attributes alignments between KGs in a big data context becomes highly expensive [17], and drastically affects the usability and computational efficiency of the consequent IM approaches [18].

Our approach does not suffer these limitations. In fact, the neighborhood context considered in SBIGMat is n-hops. In other words, SBIGMat leverages more evidential information about the matching of instances from their n-hops neighbors in their respective KBs. In addition, the incorporation of topological relations between instances and the propagation of the matching decisions through the neighbors allow SBIGMat to increase its accuracy and efficiency in matching other instances.

8. Conclusion

In this paper, we proposed a novel approach to the instance matching problem based on random walks. This approach represents and exploits the global interdependence among the different candidate pairs to infer the final co-referents. Our constrained candidates association graph plays an important role permitting our approach to be scalable. This scalability is furthermore increased by leveraging the distributed computation platform Spark. We established an affinity-preserving random walk technique which is a variant of random walk to overcome the drawback of democratic normalization. Our approach achieves competitive results on benchmark datasets compared to state-of-the-art approaches.

To deal with instances having multiple target instances with the same score (*i.e.*, ties), we use a semantic-similarity strategy to capture the most semantically similar target. For future work, we plan to address the possible integration of the semantic-similarity strategy over the RW iterations and not only during the post-processing. It will be interesting to integrate more pruning heuristics into our association graph. This will help in reducing more the running time of our approach and in removing false positives.

References

[1] A. Assi, H. Mcheick, A. Karawash, W. Dhifli, Context-aware instance matching through graph embedding in lexical semantic space, Knowledge-Based Systems 186 (2019) 104925.

[2] F. M. Suchanek, S. Abiteboul, P. Senellart, Paris: Probabilistic alignment of relations, instances, and schema, PVLDB 5 (3) (2011) 157–168.

[3] V. Christophides, V. Efthymiou, K. Stefanidis, Entity Resolution in the Web of Data, Synthesis Lectures on the Semantic Web: Theory and Technology, Morgan & Claypool Publishers, 2015.

[4] D. J. Cook, L. B. Holder, Mining Graph Data, John Wiley & Sons, Inc., USA, 2006.

[5] C. C. Aggarwal, H. Wang, Managing and Mining Graph Data, 1st Edition, Springer Publishing Company, Incorporated, 2010.

[6] E. L. Lawler, The quadratic assignment problem, Management Science 9 (4) (1963) 586–599.

[7] S. Sahni, T. Gonzalez, P-complete approximation problems, J. ACM 23 (3) (1976) 555–565.

[8] R. E. Burkard, E. Çela, P. M. Pardalos, L. S. Pitsoulis, The quadratic assignment problem (1998).

[9] S. Gold, A. Rangarajan, A graduated assignment algorithm for graph matching, IEEE Transactions on Pattern Analysis and Machine Intelligence 18 (4) (1996) 377–388.

[10] M. Gori, M. Maggini, L. Sarti, Graph matching using random walks, in: 17th International Conference on Pattern Recognition, ICPR 2004, Cambridge, UK, August 23-26, 2004., 2004, pp. 394–397.

[11] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, I. Stoica, Spark: Cluster computing with working sets, in: HotCloud, 2010, pp. 10–10.

[12] J. Dean, S. Ghemawat, Mapreduce: Simplified data processing on large clusters, Commun. ACM 51 (1) (2008) 107–113.

[13] C. You, D. P. Robinson, R. Vidal, Provable self-representation based outlier detection in a union of subspaces, in: CVPR, 2017, pp. 4323–4332.

[14] L. Page, S. Brin, R. Motwani, T. Winograd, The pagerank citation ranking: Bringing order to the web., Technical Report 1999-66, Stanford InfoLab (November 1999).

[15] A. Grover, J. Leskovec, Node2vec: Scalable feature learning for networks, in: ACM SIGKDD, 2016, pp. 855–864.

[16] G. Klyne, J. J. Carroll, Resource description framework (rdf): Concepts and abstract syntax, `http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/` (2004).

[17] X. L. Dong, D. Srivastava, Big data integration, Synthesis Lectures on Data Management 7 (1) (2015) 1–198.

[18] G. Simonini, G. Papadakis, T. Palpanas, S. Bergamaschi, Schema-agnostic progressive entity resolution, IEEE Trans. Knowl. Data Eng. 31 (6) (2019) 1208–1221.

[19] L. Lovász, Random walks on graphs: A survey, in: Combinatorics, Paul Erdős is Eighty, Vol. 2, 1996, pp. 353–398.

[20] S. Karlin, H. Taylor, A First Course in Stochastic Processes, Elsevier Science, 2012.

[21] A. N. Langville, C. D. Meyer, Deeper inside PageRank, Internet Mathematics 1 (3) (2004) 335–380.

[22] J.-Y. Pan, H.-J. Yang, C. Faloutsos, P. Duygulu, Automatic multimedia cross-modal correlation discovery., in: W. Kim, R. Kohavi, J. Gehrke, W. DuMouchel (Eds.), KDD, ACM, 2004, pp. 653–658.

[23] D. L. Isaacson, R. W. Madsen, Markov chains, theory and applications, Wiley, New York, 1976.

[24] H. D. K. Moonesignhe, P. Tan, Outlier detection using random walks, in: ICTAI, 2006, pp. 532–539.

[25] E. Seneta, Non-Negative Matrices and Markov Chains, Springer, 2006.

[26] M. Cho, J. Lee, K. M. Lee, Reweighted random walks for graph matching., in: ECCV, Vol. 6315, Springer, 2010, pp. 492–505.

[27] A. Assi, H. Mcheick, W. Dhifli, Context-aware instance matching through graph embedding in lexical semantic space, in: Advances and Trends in Artificial Intelligence. From Theory to Practice - 32nd International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems, IEA/AIE 2019, Graz, Austria, July 9-11, 2019, Proceedings, 2019, pp. 422–433.

[28] D. Gale, L. S. Shapley, College admissions and the stability of marriage, The American Mathematical Monthly 69 (1) (1962) 9–15.

[29] J. Munkres, Algorithms for the assignment and transportation problems, Journal of the society for industrial and applied mathematics 5 (1) (1957) 32–38.

[30] S. Melnik, H. Garcia-Molina, E. Rahm, Similarity flooding: a versatile graph matching algorithm and its application to schema matching, in: ICDE, 2002, pp. 117–128.

[31] P. Bojanowski, E. Grave, A. Joulin, T. Mikolov, Enriching word vectors with subword information, Transactions of the Association for Computational Linguistics 5.

[32] D. Smedley, S. Köhler, J. C. Czeschik, J. Amberger, C. Bocchini, A. Hamosh, J. Veldboer, T. Zemojtel, P. N. Robinson, Walking the interactome for candidate prioritization in exome sequencing studies of Mendelian diseases, Bioinformatics 30 (22) (2014) 3215–3222.

[33] S. Rong, X. Niu, E. W. Xiang, H. Wang, Q. Yang, Y. Yu, A machine learning approach for instance matching based on similarity metrics, in: ISWC, Springer, 2012, pp. 460–475.

[34] K. Nguyen, R. Ichise, Linked data entity resolution system enhanced by configuration learning algorithm, IEICE TRANSACTIONS on Information and Systems 99 (6) (2016) 1521–1530.

[35] S. Araujo, D. Tran, A. DeVries, J. Hidders, D. Schwabe, Serimi: Class-based disambiguation for effective instance matching over heterogeneous web data., in: WebDB, 2012, pp. 25–30.

[36] J. Li, Z. Wang, X. Zhang, J. Tang, Large scale instance matching via multiple indexes and candidate selection., Knowledge-Based Systems 50 (2013) 112–120.

[37] S. Lacoste-Julien, K. Palla, A. Davies, G. Kasneci, T. Graepel, Z. Ghahramani, Sigma: Simple greedy matching for aligning large knowledge bases, in: ACM SIGKDD, 2013, pp. 572–580.

[38] J. Li, J. Tang, Y. Li, Q. Luo, Rimom: A dynamic multistrategy ontology alignment framework, IEEE TKDE 21 (8) (2009) 1218–1232.

[39] S. Castano, A. Ferrara, S. Montanelli, D. Lorusso, Instance matching for ontology population, in: SEBD, 2008, pp. 121–132.

[40] M. Nagy, M. Vargas-Vera, E. Motta, Dssim - managing uncertainty on the semantic web, in: OM, Vol. 304 of CEUR Workshop Proceedings, 2007.

[41] H. Stoermer, N. Rassadko, Results of okkam feature based entity matching algorithm for instance matching contest of oaei 2009, in: P. Shvaiko, J. Euzenat, F. Giunchiglia, H. Stuckenschmidt, N. F. Noy, A. Rosenthal (Eds.), OM, Vol. 551 of CEUR Workshop Proceedings, 2009.

[42] Z.-H. Zhou, M. Li, Semi-supervised learning by disagreement, Knowledge and Information Systems 24 (3) (2010) 415–439.

[43] W. Hu, J. Chen, Y. Qu, A self-training approach for resolving object coreference on the semantic web, in: WWW, ACM, 2011, pp. 87–96.

[44] R. Pita, C. Pinto, P. Melo, M. Silva, M. Barreto, D. Rasella, A spark-based workflow for probabilistic record linkage of healthcare data, in: EDBT/ICDT Workshops, Vol. 1330, 2015, pp. 17–26.

[45] D. G. Mestre, C. E. S. Pires, D. C. Nascimento, A. R. M. de Queiroz, V. B. Santos, T. B. Araújo, An efficient spark-based adaptive windowing for entity matching, Journal of Systems and Software 128 (2017) 1–10.

[46] C. Wang, S. Karimi, Parallel duplicate detection in adverse drug reaction databases with spark, in: EDBT 2016, 2016, pp. 551–562.

[47] V. Efthymiou, G. Papadakis, K. Stefanidis, V. Christophides, Minoaner: Schema-agnostic, non-iterative, massively parallel resolution of web entities, in: EDBT, 2019, pp. 373–384.

[48] P. Maria, M. Yakout, K. Chakrabarti, Holistic entity matching across knowledge graphs, in: IEEE Big Data, 2015, pp. 1585–1590.

[49] A. Ferraram, A. Nikolov, F. Scharffe, Data linking for the semantic web, Semantic Web: Ontology and Knowledge Base Enabled Tools, Services, and Applications 169 (2013) 326.

[50] M. Nentwig, M. Hartung, A.-C. Ngonga Ngomo, E. Rahm, A survey of current link discovery frameworks, Semantic Web 8 (3) (2017) 419–436.

[51] M. Chen, Y. Tian, M. Yang, C. Zaniolo, Multilingual knowledge graph embeddings for cross-lingual knowledge alignment (2017) 1511–1517.

[52] M. Chen, T. Zhou, P. Zhou, C. Zaniolo, Multi-graph affinity embeddings for multilingual knowledge graphs, 2011.

[53] H. Zhu, R. Xie, Z. Liu, M. Sun, Iterative entity alignment via joint knowledge embeddings, in: Proceedings of the 26th International Joint Conference on Artificial Intelligence, IJCAI'17, AAAI Press, 2017, pp. 4258–4264.

[54] Z. Sun, W. Hu, Q. Zhang, Y. Qu, Bootstrapping entity alignment with knowledge graph embedding, in: Proceedings of the 27th International Joint Conference on Artificial Intelligence, IJCAI'18, 2018, pp. 4396–4402.

[55] Z. Sun, W. Hu, C. Li, Cross-lingual entity alignment via joint attribute-preserving embedding, in: International Semantic Web Conference, Springer, 2017, pp. 628–644.

[56] B. D. Trisedya, J. Qi, R. Zhang, Entity alignment between knowledge graphs using attribute embeddings, in: AAAI, 2019.