Holistic and Scalable Ranking of RDF Data

Axel-Cyrille Ngonga Ngomo
Data Science Group
University of Paderborn
Paderborn, Germany
Email: axel.ngonga@upb.de

Michael Hoffmann
AKSW Research Group
University of Leipzig
Leipzig, Germany
hoffmann@informatik.uni-leipzig.de

Ricardo Usbeck
Data Science Group
University of Paderborn
Paderborn, Germany
Email: ricardo.usbeck@upb.de

Kunal Jha
Data Science Group
University of Paderborn
Paderborn, Germany
Email: kunal.jha@upb.de

***Abstract*—The volume and number of data sources published using Semantic Web standards such as RDF grows continuously. The largest of these data sources now contain billions of facts and are updated periodically. A large number of applications driven by such data sources requires the ranking of entities and facts contained in such knowledge graphs. Hence, there is a need for time-efficient approaches that can compute ranks for entities and facts simultaneously. In this paper, we present the *first holistic ranking approach for RDF data*. Our approach, dubbed HARE, allows the simultaneous computation of ranks for RDF triples, resources, properties and literals. To this end, HARE relies on the representation of RDF graphs as bi-partite graphs. It then employs a time-efficient extension of the random walk paradigm to bi-partite graphs. We show that by virtue of this extension, the worst-case complexity of HARE is $O(n^5)$ while that of PageRank is $O(n^6)$. In addition, we evaluate the practical efficiency of our approach by comparing it with PageRank on 6 real and 6 synthetic datasets with sizes up to 10^8 triples. Our results show that HARE is up to 2 orders of magnitude faster than PageRank. We also present a brief evaluation of HARE's ranking accuracy by comparing it with that of PageRank applied directly to RDF graphs. Our evaluation on 19 classes of DBpedia demonstrates that there is no statistical difference between HARE and PageRank. We hence conclude that our approach goes beyond the state of the art by allowing the ranking of all RDF entities and of RDF triples without being worse w.r.t. the ranking quality it achieves on resources. HARE is open-source and is available at http://github.com/dice-group/hare.**

***Index Terms*—Semantic Web, Ranking, PageRank, Data Volume, Scalability**

I. Introduction

Improving the access to data is one of the core goals behind the Semantic Web [27], [28]. Search engines and question answering systems [6], [29] as well as Semantic Web browsers [4], [15] and entity summarization techniques [4], [5] are only a few of the types of frameworks which require the ranking of RDF[1] resources or triples to support this goal. While some approaches have been devised to rank resources [2], [8], [10], [14], [16], [21], [26], [27], properties [7], [20] or triples [9], there is currently no unified and holistic approach that is able to generate a ranking for triples as well as for entities.[2] The provision of such functionality is however of central importance for all the types of frameworks mentioned above [4], [6], [15], [29]. For example, user interfaces for entity search [6], [29] have to display ranked resources as well as summaries of these resources, which are commonly based on ranked triples, resources, properties and/or literals [4], [5].

In addition to being *holistic* (i.e., being able to rank resources, properties, literals and triples), ranking approaches for the Semantic Web need to be *efficient* so as to deal with the *volume* of the constant stream of updates which some knowledge graphs[3] are faced with. For example, DBpedia Live[4] contains $\approx 10^9$ facts (i.e., RDF triples) and is updated as often as Wikipedia, i.e., hundreds of time each day. Other knowledge graphs such as LinkedTCGA [23] are updated at even higher *velocity*. Hence, being able to recompute the ranking of entities and triples even on large knowledge graphs within a few minutes is of central importance when computing ranks on RDF knowledge graphs.

In this paper, we address the research gaps aforementioned by presenting HARE (<u>H</u>olistic R<u>a</u>nking for <u>R</u>DF <u>E</u>ntities and Triples). To the best of our knowledge, HARE *is the first ranking approach able to generate comparable ranks for resources, properties, literals and triples*. HARE is highly efficient and relies on the representation of RDF graphs using bi-partite graphs that can capture all triples and entities in the graphs. Like networks of websites connected through hyperlinks, bi-partite graphs are not ergodic [3]. Like PageRank [3], HARE enforces the characteristic of ergodicity upon the bi-partite representation of RDF graphs to derive means to index this RDF graph representation efficiently. Formally, our approach relies on the concept of *two-hop graphs*: Given a graph $\mathcal{G}$ with adjacency matrix $\mathbf{A}(\mathcal{G})$, the two-hop graph of $\mathcal{G}$ is the graph $\mathcal{G}^2$ with adjacency matrix $\mathbf{A}^2(\mathcal{G})$.[5] HARE then relies on the dependencies between the blocks in $\mathbf{A}^2(\mathcal{G})$ to derive a time-efficient indexing strategy which outperforms PageRank's runtime while retaining PageRank's ranking accuracy. The contributions of this paper are hence as follows:

1) We show that given a bi-partite graph whose nodes are partitioned across the sets V_1 and V_2, we can derive the stationary distribution of the nodes in V_1 from the stationary distribution of the nodes in V_2.
2) We use this insight to derive an approach for the efficient computation of stationary distributions of bi-partite graphs and derive the *first ranking approach for RDF*

[1]Resource Description Framework, see https://www.w3.org/RDF/
[2]We use *entity* to mean resources (including properties) and literals.
[3]Also called knowledge bases.
[4]http://live.dbpedia.org
[5]$\mathbf{A}^2(\mathcal{G}) = \mathbf{A}(\mathcal{G}) \cdot \mathbf{A}(\mathcal{G})$.

graphs that allows the ranking of resources, properties, literals and triples using comparable scores.

3) We study HARE's runtime by comparing it with PageRank. Our results suggest that HARE has a better time complexity as well as a better practical runtime than PageRank. Our results also suggest that HARE achieves a ranking accuracy comparable to that of PageRank.

The rest of this paper is structured as follows: In Section II, we present the notation that we use throughout this paper. We present the representation of RDF graphs as bi-partite graphs in Section III. In Section IV, we introduce HARE and how it computes stationary distributions efficiently. The evaluation section, i.e., Section V, presents runtime and ranking accuracy experiments, which show that our approach outperforms PageRank significantly w.r.t. its runtime while retaining its ranking accuracy. The last two sections give an overview of related works and conclude the paper.

II. Preliminaries

A. Notation

We use upper-case letters to denote sets, e.g., A and B. The elements of sets are denoted using the lower-case letter corresponding to the set name. For example, the elements of A are denoted a_1, a_2, a_3, etc. Matrices and vectors are denoted using symbols in bold font, e.g., **M**, **P**. We denote the cells of a matrix in a manner akin to the denotation of elements of sets, ergo using small letters. For example, the entry at the ith row and jth column of **M** is denoted m_{ij}. To make the dimension of a matrix **M** explicit, we write $\mathbf{M}_{n\times m}$ to denote a matrix with n rows and m columns. However, for the sake of legibility, we partly omit the dimension information from matrices if these dimensions are known. We write $\mathbf{M}\cdot\mathbf{P}$ to signify the product of the matrices **M** and **P**. The notation for vectors is handled analogously. For graphs, we use the calligraphic typeset, e.g., $\mathcal{G}$ and $\mathcal{H}$. The set of nodes of a graph $\mathcal{G}$ is denoted $V(\mathcal{G})$ while we write $E(\mathcal{G})$ for the set of edges of the same graph. The adjacency matrix of $\mathcal{G}$ is denoted $\mathbf{A}(\mathcal{G})$.

B. RDF Knowledge Graphs

An RDF *knowledge graph* K can be modeled as a set of triples $(s,p,o)\in(R\cup B)\times P\times(R\cup B\cup L)$ where

- R is the set of all RDF resources, which stand for things of relevance in the domain to model. For example, these could be publications, authors and conferences in a knowledge graph on publications.
- B the set of all RDF blank nodes, i.e., which describe existential quantifications of entities that do not need a unique global identifier.
- $P\subseteq R$ is the set of all RDF predicates and stands for the sets of binary relations which can exist between resources or literals. For example, the predicate `:authorOf` can be used to link a person with a publication (s)he (co-)wrote.
- L is the set of all literals, i.e., of all data values used in a given knowledge graph. For example, the predicate for the year of publication links a publication to a literal value for the year in which the publication was accepted.

Each RDF triple stands for a fact described in the knowledge graph. For example, the triple (`:NgongaNgomo`, `:authorOf`, `:HARE`) is to be understood as stating the fact that Ngonga Ngomo co-authored the paper HARE. We use *entities* to refer to all elements of $R\cup P\cup B\cup L$.

III. Representation of RDF Knowledge Graphs

A. RDF Knowledge as Graphs

The random surfer model has been shown to be one of the best performing ranking models over recent years [3] and is well defined for directed graphs with unlabeled edges. However, the direct translations of facts expressed in RDF into graphs that are commonly used [24] either (1) fail to represent RDF facts correctly or (2) cannot be processed by the random surfer model.

```
:BarackObama :spouse :MichelleObama .
:BarackObama :party :Democrats .
```

Listing 1: Example of a graph representation of RDF

The approach foreseen by the RDF specification[6] represents RDF triples as directed graphs with labeled edges. Here, the set of nodes of the graph is the set of all subjects and objects. A triple (s,p,o) is represented by two nodes s and o connected by a directed edge e with the label $\lambda(e)=p$. Such a representation for the example knowledge graph shown in Listing 1 is depicted in Figure 1a. This representation has a fatal flaw with respect to the task at hand: Neither properties nor triples can be ranked, as they do not belong to the nodes of the graph.

A way to alleviate these limitations is to assign one node to each resource and to each literal in the knowledge graph. This approach then models each triple (s,p,o) by adding edges (s,p) and (p,o) to the graph representing the input knowledge graph. While this representation has been used in the past, it is actually flawed (see [24] for details) as the knowledge graphs in the Listings 2 and 3 become indistinguishable (see Figure 1b, note that `:givenName` is assigned one node as per definition). Moreover, triples cannot be ranked using this approach. A similar representation assigns one node to each predicate for each triple in which it occurs. This representation however leads to an incorrect ranking of predicates as a single predicate is represented by several nodes, hence leading to incorrect transition probabilities. Moreover, this approach also does not cater for the ranking of triples.

```
:BarackObama :givenName "Barack"@en .
:MichelleObama :givenName "Michelle"@en.
```

Listing 2: Example of a knowledge graph K_1

[6]https://www.w3.org/TR/rdf11-concepts/

```
:BarackObama :givenName "Michelle"@en .
:MichelleObama :givenName "Barack"@en .
```

Listing 3: Example of a knowledge graph K_2

Previous works have suggested a representation which addresses the problems above by using hypergraphs [25]. Each triple $(s, p, o) \in K$ is represented as a directed edge which links three nodes s, p, and o. While this representation is correct by virtue of allowing the complete and correct reconstruction of K, it goes beyond the paradigm supported by the classical random surfer models. We hence first address the problem of representing RDF in graphs (1) that can be processed by random surfer approaches and (2) that are complete and correct, i.e., that are such that K can be reconstructed exactly from the graph representation. To this end, we use the representation of RDF using bi-partite graphs suggested in [13]. As shown in subsequent sections, it (1) allows for random walks to be executed and (2) is complete and correct.

B. Knowledge Graphs as Bi-Partite Graphs

Let K be an RDF knowledge graph. The bi-partite representation $\mathcal{B}(K)$ of K is an undirected unlabeled graph constructed as follows:

- $V(\mathcal{B}(K)) = \{x : x = (s, p, o) \in K \vee (x \in \{s, p, o\} \wedge (s, p, o) \in K)\}$,
- $E(\mathcal{B}(K)) = \{\{u, v\} : (u = (s, p, o) \wedge v \in \{s, p, o\})\}$.

For our running example (shown in Listing 1), we get the representation shown in Figure 1c. K can be reconstructed completely and correctly from $\mathcal{B}(K)$ by selecting only the nodes that stand for triples. Given this representation, one could simply apply classical random walk algorithms such as PageRank to derive stationary distribution and would be able to rank all nodes, i.e., triples, resources (including properties) and even literals, thus achieving the goal of this paper. However, using the classical random surfer approach would discard an important piece of information about the scalability of the algorithm, i.e., that $\mathcal{B}(K)$ is bi-partite. By making use of this piece of information, we can implement a significantly more time- and space-efficient random walk on $\mathcal{B}(K)$.

IV. HARE: RANDOM WALKS ON BI-PARTITE GRAPHS

A. Insights

HARE makes use of the two following intuitions:

1) Insight 1: Stationary distributions for ergodic transition matrices are idempotent.: Our first insight is that the stationary distribution vector $\mathbf{S}$ of a transition matrix $\mathbf{P}$ (which is given by $\mathbf{S} = \mathbf{P}^T \cdot \mathbf{S}$) is also the stationary distribution of $\mathbf{P}^2$ if $\mathbf{P}$ is ergodic.

Proof. $\mathbf{S} = \mathbf{P}^T \cdot \mathbf{S} \Rightarrow \mathbf{P}^T \cdot \mathbf{S} = \mathbf{P}^T \cdot \mathbf{P}^T \cdot \mathbf{S}$. Per definition, $\mathbf{S} = \mathbf{P}^T \cdot \mathbf{S}$. Given that $(\mathbf{P}^T)^2 = (\mathbf{P}^2)^T$, we can derive that $\mathbf{S} = (\mathbf{P}^2)^T \cdot \mathbf{S}$. Per definition of a stationary distribution, the last equation means that $\mathbf{S}$ is also a stationary distribution vector of $\mathbf{P}^2$. $\mathbf{P}$ and $\mathbf{P}^2$ being ergodic means that their stationary distributions are unique [19]. Hence, $\mathbf{S}$ is not only a stationary distribution of $\mathbf{P}$ and $\mathbf{P}^2$, it is *the unique* stationary distribution of $\mathbf{P}$ and $\mathbf{P}^2$. $\square$

This means that by finding the stationary distribution for $\mathbf{P}^2$, we also find the distribution for $\mathbf{P}$. We make use of this intuition by running our random walk on two-hop graphs.

2) Insight 2: All two-hop graphs of bi-partite graphs have at least 2 components.: Given a bi-partite undirected graph $\mathcal{G}$, one can generate an undirected weighted graph $\mathcal{G}^2$ with $\mathbf{A}(\mathcal{G}^2) = \mathbf{A}(\mathcal{G}) \cdot \mathbf{A}(\mathcal{G})$ whose nodes u and v are connected by an edge $\{u, v\}$ with a weight that is the number of paths of length 2 between the two nodes u and v (see example in Figure 2). We call this graph the *two-hop graph* of $\mathcal{G}$. If $\mathcal{G}$ is bi-partite, then $\mathcal{G}^2$ will only contain edges between nodes of the same color (i.e., nodes which belong to the same partition, see Figure 2). Hence, $\mathcal{G}^2$ will consist of at least two components, of which each will only contain nodes of different colors. A random walk can be applied to these two components independently as they do not share any nodes or edges.

We make use of these two insights to (1) devise efficient ways to compute ranks on large RDF graphs using the random surfer paradigm that (2) generate comparable ranks for triples and entities.

B. Transition Probabilities

Let $\mathcal{G}$ be a graph with $|V(\mathcal{G})| = n$. The goal of a random walk approach is to determine the stationary distribution of the probabilistic transition matrix $\mathbf{P}_{n \times n}(\mathcal{G})$ with

$$\mathbf{P}(\mathcal{G}) = \mathbf{D}(\mathcal{G})^{-1} \cdot \mathbf{A}(\mathcal{G}), \tag{1}$$

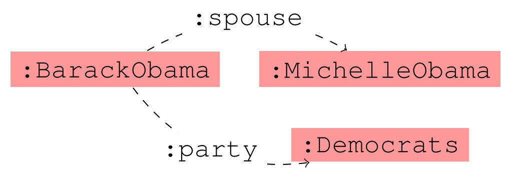
where $\mathbf{D}(\mathcal{G})$ stands for the diagonal matrix where d_{ii} is the degree of the ith node of $\mathcal{G}$ and $i \neq j \Rightarrow d_{ij} = 0$. Applying the classical random walk model to a bi-partite representation of knowledge graph $\mathcal{B}(K)$ would result in having to store and manipulate a transition matrix $\mathbf{P}(\mathcal{B}(K))_{\alpha+\beta,\alpha+\beta}$ where $\alpha = |R \cup B \cup L|$ and $\beta = |K| \in O(\alpha^3)$. At least $2\alpha\beta$ of these cells would be 0 due to the graph being bi-partite. We extend the surfer model to bi-partite graphs as follows: Instead of computing one transition matrix $\mathbf{P}(\mathcal{B}(K))$, we compute two transition matrices: one for the triples and one for all other entities in $\mathcal{B}(K)$.

Let $\mathcal{T}(K)$ be the graph derived from second-order hops in $\mathcal{B}(K)$ which only contains edges between triples. Formally, $\mathcal{T}(K)$ is defined as follows:

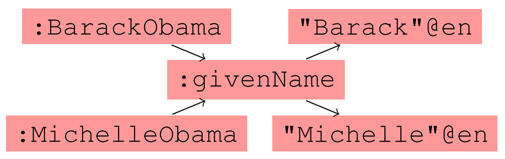
- $V(\mathcal{T}(K)) = \{x : (s, p, o) \in K\}$,
- $E(\mathcal{T}(K)) = \{(u, v) : u = (s, p, o) \wedge v = (s', p', o') \wedge \{s, p, o\} \cap \{s', p', o'\} \neq \emptyset\}$.

Conversely, let $\mathcal{N}(K)$ be the subgraph of the second-order hops in $\mathcal{B}(K)$ that only contains entities. $\mathcal{N}(K)$ is defined in a manner akin to $\mathcal{T}(K)$. In the following, we will write $\mathcal{T}$ and $\mathcal{N}$ to mean $\mathcal{T}(K)$ and $\mathcal{N}(K)$ for reasons of succinctness.

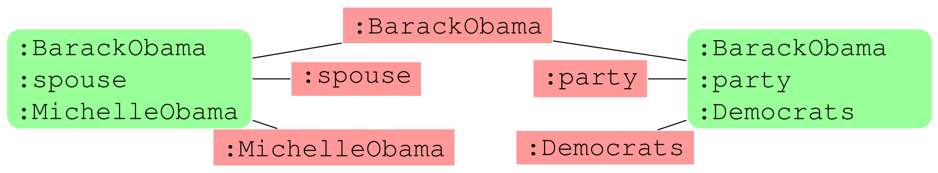
The idea behind HARE is that the probability of a random surfer going from a triple $t = (s, p, o)$ to triple $t' = (s', p', o')$



(a) Representation of our example using the RDF specification

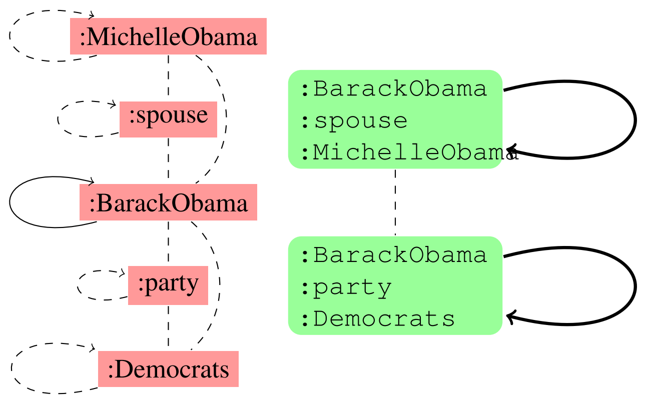


(b) Graph representation of K_1 and K_2.


:BarackObama
:BarackObama :spouse :MichelleObama
:spouse
:party
:BarackObama :party :Democrats
:MichelleObama
:Democrats

(c) Bi-partite graph representation of our example

Fig. 1: Representations of our example knowledge graph. Triples are rectangles with rounded borders and all other entities are rectangles.


:MichelleObama
:spouse
:BarackObama
:party
:Democrats
:BarackObama :spouse :MichelleObama
:BarackObama :party :Democrats

$$\begin{bmatrix} 1 & 1 & 2 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 3 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 3 \end{bmatrix}$$

Fig. 2: $\mathcal{G}^2$ to the example graph and corresponding adjacency matrix. Dashed lines have a weight of 1, thin lines stand for a weight of 2 while thick lines have a weight of 3.

(denoted $\pi(t \to t')$) can be regarded as the following sum of probabilities:

$$\pi(t \to s)\pi(s \to t') + \pi(t \to p)\pi(p \to t') + \pi(t \to o)\pi(o \to t'). \quad (2)$$

An equivalent proposition can be made for nodes of $\mathcal{B}(K)$ that are not triples. Formally, this means that we can compute $\mathbf{P}(\mathcal{T})_{(\beta,\beta)}$ independently from $\mathbf{P}(\mathcal{N})_{(\alpha,\alpha)}$ and vice-versa as follows: Let $\mathbf{W}_{(\beta,\alpha)}$ be the transition matrix from triples to entities. Given that the probability of transitioning from a triple to one of its elements is exactly $\frac{1}{3}$, we can write $w_{ij} = \frac{1}{3}$ iff the triple t_i contains the node j of $\mathcal{B}(K)$. Else $w_{ij} = 0$.

Now let $\mathbf{F}_{(\alpha,\beta)}$ be the matrix of which the entries are the transition probabilities from entities to triples. The entries f_{ij} of $\mathbf{F}$ are given by

$$f_{ij} = \frac{1}{|\{t = (s,p,o) : e_i \in \{s,p,o\}\}|} \text{ iff } e_i \in t_j. \text{ Else } f_{ij} = 0. \quad (3)$$

By virtue of $\mathcal{B}(K)$ being bi-partite, we now have

$$\mathbf{P}(\mathcal{T})_{(\beta,\beta)} = \mathbf{W}_{(\beta,\alpha)} \cdot \mathbf{F}_{(\alpha,\beta)} \text{ and } \mathbf{P}(\mathcal{N})_{(\alpha,\alpha)} = \mathbf{F}_{(\alpha,\beta)} \cdot \mathbf{W}_{(\beta,\alpha)}. \quad (4)$$

For our example (see Figure 1c), $\mathbf{W} = \begin{bmatrix} \frac{1}{3} & \frac{1}{3} & 0 & 0 & \frac{1}{3} \\ \frac{1}{3} & 0 & \frac{1}{3} & \frac{1}{3} & 0 \end{bmatrix}$, $\mathbf{F} = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 1 & 0 \end{bmatrix}$ and $\mathbf{P}(\mathcal{T}) = \begin{bmatrix} \frac{5}{6} & \frac{1}{6} \\ \frac{1}{6} & \frac{5}{6} \end{bmatrix}$. While a classical approach would necessitate $(\alpha+\beta)^3$ computations (i.e., multiplications and additions) to compute $\mathbf{A}(\mathcal{G}^2)$ using $\mathbf{A}(\mathcal{G})$, HARE only needs $\alpha\beta^2 + \beta\alpha^2$ computations to compute $\mathbf{P}(\mathcal{T})$ and $\mathbf{P}(\mathcal{N})$ (which fully describe $\mathbf{A}(\mathcal{G}^2)$ by virtue of $\mathcal{G}$ being bi-partite). Hence, HARE can save up to $\alpha^3 + 2\alpha^2\beta + 2\alpha\beta^2 + \beta^3$ computations during the computation of transition matrices.

C. Stationary Probability Distributions

Given the transition matrices computed in the precedent step, the goal of HARE is to compute the probability distribution row vector $\mathbf{S}_{\alpha+\beta}$, which contains the probability that a random surfer lands on the nodes of $\mathcal{B}(K)$. Based on our second insight, we can split $\mathbf{S}$ into two vectors: $\mathbf{S}(\mathcal{T})_\beta$ for triples and $\mathbf{S}(\mathcal{N})_\alpha$ for all other entities. A naïve implementation of HARE would now simply compute $\mathbf{S}(\mathcal{T})$ and $\mathbf{S}(\mathcal{N})$ independently by setting

$$\mathbf{S}(\mathcal{T}) = \mathbf{P}(\mathcal{T})^\top \cdot \mathbf{S}(\mathcal{T}) \text{ and } \mathbf{S}(\mathcal{N}) = \mathbf{P}(\mathcal{N})^\top \cdot \mathbf{S}(\mathcal{N}). \quad (5)$$

While this naïve implementation would already be clearly superior to a naïve ranking implementation of the random surfer model, it would still need to compute matrices of size β^2 (e.g., $\mathbf{P}(\mathcal{T})$) and α^2 (e.g., $\mathbf{P}(\mathcal{N})$). However, we can avoid a significant portion of this computation based on the following insight:

1) Insight 3: Dependency between Stationary Distributions in Bi-Partite Graphs.: In general, β is significantly larger than α (in the worst case, $\beta = \alpha^3$). Hence, limiting the number of computations in the order of β is bound to improve the runtime of HARE significantly. We do so by proving that $\mathbf{S}(\mathcal{T})$ can be derived from $\mathbf{S}(\mathcal{N})$ as follows:

$$\mathbf{S}(\mathcal{T}) = \mathbf{F}^T \cdot \mathbf{S}(\mathcal{N}). \quad (6)$$

Proof. Recall that $\mathbf{P}(\mathcal{N}) = \mathbf{F} \cdot \mathbf{W}$ and $\mathbf{P}(\mathcal{T}) = \mathbf{W} \cdot \mathbf{F}$. Now by definition $\mathbf{S}(\mathcal{N}) = \mathbf{P}(\mathcal{N})^T \cdot \mathbf{S}(\mathcal{N})$. Hence, $\mathbf{F}^T \cdot \mathbf{S}(\mathcal{N}) = \mathbf{F}^T \cdot \mathbf{P}(\mathcal{N})^T \cdot \mathbf{S}(\mathcal{N})$. Given that $\mathbf{P}(\mathcal{N})^T = \mathbf{W}^T \cdot \mathbf{F}^T$, we have $\mathbf{F}^T \cdot \mathbf{S}(\mathcal{N}) = \mathbf{F}^T \cdot \mathbf{W}^T \cdot \mathbf{F}^T \cdot \mathbf{S}(\mathcal{N})$. Given that $\mathbf{P}(\mathcal{T})^T = \mathbf{F}^T \cdot \mathbf{W}^T$, we get $(\mathbf{F}^T \cdot \mathbf{S}(\mathcal{N})) = \mathbf{P}(\mathcal{T})^T \cdot (\mathbf{F}^T \cdot \mathbf{S}(\mathcal{N}))$. By virtue of the definition of $\mathbf{S}(\mathcal{T})$, we obtain $\mathbf{S}(\mathcal{T}) = \mathbf{F}^T \cdot \mathbf{S}(\mathcal{N})$. $\square$

We hence only need to compute $\mathbf{S}(\mathcal{N})$ (and thus only $\mathbf{P}(\mathcal{N})$) explicitly and can derive $\mathbf{S}(\mathcal{T})$ from Equation 6, hence saving $O(\beta^2)$ in space and time in the best case (see Table I).

D. Ergodicity and Iterative Computation of Stationary Distributions

Recall that $\mathbf{S}(\mathcal{N})$ is the vector that satisfies

$$\mathbf{S}(\mathcal{N}) = \mathbf{P}(\mathcal{N})^\top \cdot \mathbf{S}(\mathcal{N}). \quad (7)$$

Like in previous works [3], [1], we apply an iterative approach that seeks to ensure the ergodicity of the random walk. The stationary probability distribution $\mathbf{S}(\mathcal{G})$ is commonly initialized by setting all entries to $1/|V(\mathcal{G})|$. However, doing so for $\mathbf{S}(\mathcal{N})$ (i.e., with $1/\alpha$) would lead to the total probability in $\mathbf{S}$ being larger than one. To ensure that $\mathbf{S}(\mathcal{T})$ and $\mathbf{S}(\mathcal{N})$ build a probability distribution, we set all the entries of the initial $\mathbf{S}(\mathcal{N})$ to

$$\frac{\beta}{\alpha(\beta+\alpha)}. \quad (8)$$

This value scales the vector down to the proportion it would have had if we had run the computation using a naïve version of the computation of stationary distributions.

We can now compute $\mathbf{S}(\mathcal{N})$ and derive $\mathbf{S}(\mathcal{T})$ by using Equation 6. To ensure the ergodicity of the matrices on which we operate, we iterate the PageRank equation [3]

$$\mathbf{S}(\mathcal{N}) = \gamma \mathbf{P}(\mathcal{N})^T \cdot \mathbf{S}(\mathcal{N}) + \frac{(1-\gamma)\mathbf{I}}{|\mathbf{S}(\mathcal{N})|}, \quad (9)$$

where $\gamma \in [0,1]$ is the damping factor, $|\mathbf{S}(\mathcal{N})|$ is the number of rows in $\mathbf{S}(\mathcal{N})$ and I is the 1-vector, i.e., the vector with entries of 1. The iteration is carried out until the euclidean distance of the vectors obtained in two successive iterations is under a given threshold ϵ or a maximal number of iterations has been reached. The entries of the resulting vector are the probability that a random surfer lands on a particular entity while surfing. Hence, the higher the value achieved by a node, the better the rank it should be assigned.

Once $\mathbf{S}(\mathcal{N})$ has been computed, we compute $\mathbf{S}(\mathcal{T})$ by using Equation 6. We can finally compute $\mathbf{S}$ by simply concatenating the two vectors.

V. EVALUATION

A. Experimental Setup

The goal of our evaluation was to elucidate the following three questions:

Q_1 Complexity: *What is the exact upper bound of the number of computations carried out by* HARE? How does it compare with the upper bound of the computations that PageRank would have to carry out to address the same task? We addressed the complexity question by computing the number of additions and multiplications that the two approaches carry out.

Q_2 Runtime: *How does the difference in complexity influence the runtime of the two approaches*? To address this question, we implemented both HARE and PageRank using the same libraries and compared their runtime on different datasets of up to 133.6 million triples.

Q_3 Ranking: *How well does* HARE *perform w.r.t. ranking quality*? We performed a comparative manual evaluation of the ranking of resources computed by HARE and PageRank on DBpedia.

All runtime experiments were carried out using a single core of an Intel(R) Xeon(R) CPU E5-2683 v4 @ 2.10GHZ Processor on a machine equipped with 503 GB of RAM. We measured the time to compute the transition matrix until convergence with $\epsilon = 10^{-3}$ and used a damping factor $\gamma = 0.85$ as chosen in [1]. We repeated each experiment 5 times and report average values.

B. Complexity

The runtime complexity of HARE and PageRank when applied to bi-partite graphs depends mostly on the complexity of the matrix multiplication implementation chosen. The exact numbers of computations shown in Table I are an upper bound of the runtime complexity for HARE and Pagerank. These numbers were derived for a naïve implementation of matrix multiplication, which has a cubic complexity. For this upper bound, the number of additions computed by the two approaches to determine the transition probability matrices is in $O(\alpha^3+\beta^3)$. If $\beta \in O(\alpha^3)$, then this difference is in $O(\alpha^9)$. Note that the table does not contain the last step carried out by HARE, which consists of computing $\mathbf{S}(\mathcal{T})$ using Equation 6. This step runs in $O(\alpha^2\beta)$, which is negligible when compared with $O(\alpha^3+\beta^3)$.

The lowest bound for the runtime complexity of any matrix multiplication is in $O(n^2)$ for any $n \times n$-matrix. This is due to all entries of the matrix needing to be read at least once during the computation. The worst-case complexity of PageRank would then be in $O((\alpha+\beta)^2)$. Given that $\beta = \alpha^3$ in the worst case, the complexity of PageRank would be $O(\alpha^6)$. This complexity is still higher than the complexity of HARE, which is $O(\alpha^5)$.

The answer to Q_1 is clear: HARE *has a lower theoretical complexity than PageRank on bi-partite graphs.*

This answer is an important result as it shows that HARE is guaranteed to be more scalable than PageRank.

C. Runtime Evaluation

We evaluated how the lower runtime complexity of HARE translates into real-world runtimes by measuring the runtime of HARE and PageRank on six real[7] and six synthetic knowledge graphs. The real datasets were as follows:

- *Airports*, which contains locations, names and codes for several airports, retrieved from openflights.org.[8]
- *UPTSO*, a knowledge graph about US Patent and Trademark Office data.[9]
- *SIDER*, which describes the side-effects of several marketed drugs.[10]
- *Semantic Dogfood* (also known as the Semantic Web Conference Corpus), a dataset on published papers, workshops and attendance of people or organizations.[11]
- *USSECCCO*, the U.S. Securities and Exchange Commission Corporate Ownership RDF dataset, which describes the ownership of corporations inside and outside the United States.[12]
- *DBpedia*, a large Linked Data set derived from Wikipedia.[13] We provide links to the underlying files and datasets in our project repository.

We used the Lehigh University Benchmark (LUBM) generator [11] for the generation of synthetic data. The number after the dataset in Table II indicates the number of universities created.

Table II gives an overview of our runtime results. HARE consistently outperforms PageRank on the same hardware by a factor between 6 (Airports) and 190 (LUBM1000) when executed on a single core. Our runtimes on large knowledge graphs such as LUBM1000 and DBpedia already show that our approach can be used in practical applications and that it only needs around 8 resp. 15 seconds to generate ranks for datasets of sizes in the order of 10^8 triples.

[7]We chose the knowledge graphs as representatives for sizes between $\approx 10^4$ triples and $\approx 10^8$ triples.

[8]http://hobbitdata.informatik.uni-leipzig.de/hare/airports.nt

[9]http://hobbitdata.informatik.uni-leipzig.de/hare/uspto.zip

[10]http://hobbitdata.informatik.uni-leipzig.de/hare/sider.nt

[11]http://hobbitdata.informatik.uni-leipzig.de/hare/dogfood.nt

[12]http://hobbitdata.informatik.uni-leipzig.de/hare/sec.nt

[13]http://wiki.dbpedia.org/downloads-2016-04

The almost linear scaling of the runtimes in the LUBM experiment can be explained by the structure of the data. Since several universities are generated separately, they are very sparsely connected by links and form partly independent blocks in the adjacency matrix.

Our experiments suggest that the answer to Q_2 is that *we outperform PageRank on both real and synthetic data and are up to two orders of magnitude faster.*

D. Ranking Evaluation

We aimed to determine how HARE performs w.r.t. its ranking quality when $\gamma = 0.85$. We hence chose a typical evaluation scenario to compare PageRank and HARE: We assumed an information retrieval setup within which a user provides the name of a class as input. The expected result is an ordered list of instances of the input class. We selected 19 classes from DBpedia (see Table III) as input classes. The classes were chosen to be such that most reviewers were guaranteed to be familiar with the instances of the classes.

We computed the top-10 resources of these classes according to PageRank and according to HARE. We then recorded the differences between the rankings as follows: Each ranking of resources $r_1, \ldots, r_k$ was represented as a set of pairs (r_i, r_j) with $i < j$. The difference in ranking was computed by computing the symmetric set difference between the sets of pairs generated by PageRank and HARE. Given that each ranking generates exactly $\frac{k(k-1)}{2}$ of such pairs, the highest possible number of differences between two rankings of size k is $k(k-1)$.

For each class, we selected 10 pairs from the differences recorded between PageRank and HARE randomly and showed them to 4 independent raters.[14] The raters were computer scientists and post-graduates. For each pair (r_i, r_j), the raters were to state whether they found the ranking to be correct or erroneous. They were also allowed to mark a ranking as unknown. To remove any bias, we ensured that the raters did not know which algorithm generated which pair.

We used Fleiss' kappa to quantify the inter-rater agreement in the experiment and achieved 0.3. According to [18], this value indicates a fair agreement between the raters. We considered a ranking to be agreed upon by the raters if at least 3 raters concurred. 66 of the 190 pairs could not be assigned a ranking. In 89 cases (49 for PageRank, 40 for HARE), the raters agreed with the rankings computed by the algorithms HARE and PageRank. In the remaining 35 (17 for PageRank, 18 for HARE) cases, they disagreed with the ranking (see Table III). We ran a one-tailed Wilcoxon Signed Rank test on the combined results of the evaluator (significance level: $p < 0.05$). The test showed that there is no statistically significant difference between the results achieved by the two approaches. This result is confirmed by an evaluation of the performance of the approaches at class level: HARE's ranking was deemed better than PageRank's

[14]The evaluation results can be found at https://goo.gl/ptLwRJ.

TABLE I: Worst-case analysis of HARE and PageRank for a naïve implementation of matrix multiplication. Recall that $\beta \in O(\alpha^3)$ in the worst case.

	Transition probability		
	Additions	Multiplications	Complexity
PageRank	$(\alpha+\beta)^2(\alpha+\beta-1)$	$(\alpha+\beta)^3$	$O(\alpha^3+\beta^3)=O(\alpha^9)$
HARE	$\alpha^2(\beta-1)$	$\alpha^2\beta$	$O(\alpha^2\beta)=O(\alpha^5)$
Gain (HARE)	$\beta^3-\beta^2+\alpha\beta(2\alpha+3\beta-2)+\alpha^3$	$\beta^3+3\alpha\beta^2+2\alpha^2\beta+\alpha^3$	$O(\alpha^3+\beta^3)=O(\alpha^9)$
	Stationary distribution (iteration)		
	Additions	Multiplications	Complexity
PageRank	$(\alpha+\beta)(\alpha+\beta-1)$	$(\alpha+\beta)^2$	$O(\alpha^2+\beta^2)=O(\alpha^6)$
HARE	$\alpha(\alpha-1)$	α^2	$O(\alpha^2)$
Gain (HARE)	$\beta^2+\beta(2\alpha-1)-2\alpha$	$\beta^2+2\alpha\beta$	$O(\alpha^2)$

TABLE II: Average runtimes of HARE and PageRank over five runs in seconds.

	#Triples	**#Entities**	**HARE**	**Pagerank**
LUBM20	2 688 046	663 661	0.13	7.75
LUBM50	6 654 562	1 639 709	0.34	24.08
LUBM100	13 405 381	3 301 732	0.71	50.48
LUBM200	26 696 578	6 574 874	1.52	103.84
LUBM500	66 731 200	16 429 334	4.26	425.12
LUBM1000	133 573 854	32 885 165	8.57	1631.39
Airports	67 267	47 772	0.02	0.12
SIDER	101 542	30 379	0.01	0.13
Dogfood	305 112	95 527	0.02	0.49
USSECCCO	1 813 135	866 611	0.12	4.90
UPSTO	10 438 358	4 114 136	1.01	27.2
DBpedia	96 222 324	31 176 843	15.45	937.15

on 9 classes and vice-versa. The approaches tied in 1 case (`dbo:HistoricPlace`).[15]

Overall, our results suggest there is no qualitative difference between the ranking results achieved with PageRank and HARE, hence making clear that HARE can retain the accuracy of PageRank while being significantly more time-efficient. Our current results suggest that HARE tends to work better on classes with a low- or average-size extensions (e.g., `dbo:Drug`, `dbo:Automobile`), while PageRank seems slightly better on classes that have large extensions (e.g., `dbo:Person`). These tendencies still have to be confirmed in larger experiments.

Our results answer Q_3 as follows: *Our experiments suggest that there is no statistically significant difference between the quality of the ranking results achieved by PageRank and* HARE *on the task of ranking resources.*

In contrast to classical approaches for ranking resources, HARE can however rank triples and properties while ranking resources. Table IV shows the top-10 properties, classes and resources of DBpedia according to HARE.

VI. RELATED WORK

We present the first holistic approach for ranking all components of an RDF graph and RDF triples at once. Ranking approaches for RDF split into three categories, namely ranking of RDF triples, RDF properties and RDF resources. Approaches to *ranking triples* have been developed in the area of entity summarization. For example, RELIN [5] uses a graph-based model to detect top-n triples based on the top-n relations of a given resource. Ranking RDF triples is also the focus of Franz et al. [9], which presents TripleRank, a tensor-decomposition-based approach emulating the HITS algorithm [17]. The resulting authority values were used as ranks and evaluated using human raters. The original datasets[16] exist but the evaluation data is not available. FRanCo is a novel ground truth corpus for Fact Ranking, i.e., given an entity, it poses the question: which triples are more important?[17] So

[15]`dbo` stands for `http://dbpedia.org/ontology`.

[16]http://west.uni-koblenz.de/de/forschung/datensaetze/triplerank-data-sets

[17]http://s16a.org/node/13

TABLE III: Comparison of HARE and PageRank

Class	HARE		PageRank	
	Agree(%)	Disagree(%)	Agree(%)	Disagree(%)
Drugs	40	0	20	20
SpaceStation	60	0	40	0
Automobile	80	0	40	20
EurovisionSongContestEntry	0	0	0	40
ChessPlayer	60	0	20	20
Album	40	20	40	40
Band	40	40	80	0
City	80	20	60	20
HistoricPlace	20	20	40	40
VideoGame	80	20	40	60
University	20	80	20	0
Mountain	40	0	80	20
Town	20	0	80	0
Comedian	20	20	40	20
MilitaryConflict	40	20	20	0
Country	40	60	60	20
ProgrammingLanguage	20	0	40	0
Person	20	20	80	20
Hotel	0	40	20	0

TABLE IV: Top-10 properties, classes and resources in DBpedia 2016-04 according to HARE. `rdfs` stands for `http://www.w3.org/2000/01/rdf-schema#`. `rdf` stands for `http://www.w3.org/1999/02/22-rdf-syntax-ns#`. `foaf` stands for `http://xmlns.com/foaf/0.1/`. `dbo` stands for `http://dbpedia.org/ontology/` and `dbr` stands for `http://dbpedia.org/resource/`.

Rank	**Property**	**Class**	**Resource**
1	`rdfs:label`	`dbo:CareerStation`	`dbr:Animal`
2	`dbo:wikiPageRedirects`	`foaf:Person`	`dbr:United_States`
3	`foaf:name`	`dbo:Person`	`dbr:Arthropod`
4	`rdf:type`	`dbo:SportsTeamMember`	`dbr:Insect`
5	`dbo:wikiPageDisambiguates`	`dbo:Settlement`	`dbr:Lepidoptera`
6	`dbo:team`	`dbo:PersonFunction`	`dbr:United_Kingdom`
7	`foaf:surname`	`dbo:Village`	`dbr:Poland`
8	`foaf:homepage`	`dbo:TimePeriod`	`dbr:India`
9	`dbo:isPartOf`	`dbo:Insect`	`dbr:Iran`
10	`foaf:givenName`	`dbo:Album`	`dbr:France`

far, FRanCo has not been used in other works to evaluate triple rankings. FranCo is not used to evaluate HARE since the underlying data was heavily processed and pruned, hence making it unfit for an evaluation of HARE over the whole DBpedia.

Ranking properties has also been regarded as important. For example, approaches such as RELIN [5] rank properties for entity summarization. The work of Dessi et al. [7] uses a machine learning approach to rank RDF properties based on a number of specifically designed numerical features that measure different aspects of each property. The authors apply existing learning-to-rank algorithms to a number of classified properties, thereby automatically constructing ranking models that reflect a given classification. DBtrends [20] is another framework designed for comparing and evaluating different ranking functions for RDF data. It allows the combination of these rankings by means of an extension of the Spearmans footrule estimation.

The largest body of work related to this work pertains to *ranking entities*. An extension to the PageRank algorithm using Linked Data knowledge has been described by Julia Stoyanovich [26]. This approach extracts semantic knowledge from a Web document and combines this with an underlying ontology which shows an improvement for the ranking quality. Ding et al. [8] present Swoogle, a first Semantic Web search engine. Although, Swoogle's creators aim to utilize more than just resources for ranking, e.g., RDF classes, the underlying OntoRank is based on the rational surfer model at the document level. However, the used dataset, namely DS-April 2005, is no longer available. In 2007, the Sindice [27] search engine was introduced which also includes a ranking function based on a fast metadata-using algorithm. However, Sindice does not make use of any RDF-based metadata and only covers information such as host rank etc. Blanco et al. [2] proposes an adaptation of the BM25F ranking function to the RDF data model that incorporates both field weights, document priors and a separate field for the subject URIs. The authors also go on to propose index structures in order to achieve efficient retrieval and ranking of results. Mirizzi et al. [21] propose a novel, hybrid ranking function for DBpedia using external sources such as search engines and bookmarking sites. The evaluation was done using a set of users and their rating towards the ranking. Unfortunately, the provided resources are not sufficiently available anymore. ReConRank [16] is another highly efficient algorithm based on PageRank. This algorithm is based on semantic sub-graphs whose size influences efficiency and precision of results. Other approaches such as Gupta et al. [12] or xhRank [14] also generate result list rankings. Graves et al. [10] present an approach to rank RDF nodes in result sets according to their centrality. Further overviews can be found in [22], [30].

VII. Conclusion

We presented HARE, the first holistic approach for ranking triples and entities in RDF knowledge graphs. Our approach relies on a bi-partite representation for RDF graphs. We combined this representation with the characteristics of ergodicity to derive a time-efficient approach for ranking triples and entities in a comparable way. We showed that the worst-case complexity of our approach is polynomial and lower than that of PageRank. We also evaluated the runtime of our approach on graphs of various sizes and showed that it scales better than the classical PageRank. Our results suggest that our approach achieves the same ranking quality as PageRank on DBpedia resources. In future work, we will study distributed implementations of HARE. Our work also revealed a lack of open benchmarks for ranking RDF resources. We will hence work on developing such resources in the future.

Acknowledgments

This work was supported by the H2020 project HOBBIT (GA no. 688227), the EuroStars projects DIESEL (E!9367) and QAMEL (E!9725) as well as the BMVI projects LIMBO (project no. 19F2029C) and OPAL (project no. 19F20284).

References

[1] Abdelghani Bellaachia and Mohammed Al-Dhelaan. Random walks in hypergraph. In *Proceedings of the 2013 international conference on applied mathematics and computational method*, pages 187–194, 2013.

[2] Roi Blanco, Peter Mika, and Sebastiano Vigna. Effective and efficient entity search in rdf data. In *ISWC*, pages 83–97. Springer, 2011.

[3] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. In *Computer Networks and ISDN Systems*, pages 107–117. Elsevier Science Publishers B. V., 1998.

[4] Gong Cheng, Weiyi Ge, and Yuzhong Qu. Falcons: searching and browsing entities on the semantic web. In *Proceedings of the 17th international conference on World Wide Web*, pages 1101–1102. ACM, 2008.

[5] Gong Cheng, Thanh Tran, and Yuzhong Qu. Relin: relatedness and informativeness-based centrality for entity summarization. In *International Semantic Web Conference*, pages 114–129. Springer, 2011.

[6] Philipp Cimiano, Vanessa Lopez, Christina Unger, Elena Cabrio, Axel-Cyrille Ngonga Ngomo, and Sebastian Walter. Multilingual question answering over linked data (qald-3): Lab overview. In *International Conference of the Cross-Language Evaluation Forum for European Languages*, pages 321–332. Springer, 2013.

[7] Andrea Dessi and Maurizio Atzori. A machine-learning approach to ranking rdf properties. *Future Generation Computer Systems*, 54:366–377, 2016.

[8] Li Ding, Rong Pan, Timothy W. Finin, Anupam Joshi, Yun Peng, and Pranam Kolari. Finding and ranking knowledge on the semantic web. In *ISWC*, volume 3729, pages 156–170. Springer, 2005.

[9] Thomas Franz, Antje Schultz, Sergej Sizov, and Steffen Staab. *TripleRank: Ranking Semantic Web Data by Tensor Decomposition*, pages 213–228. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[10] Alvaro Graves, Sibel Adali, and Jim Hendler. A method to rank nodes in an RDF graph. In *International Semantic Web Conference (Posters & Demos)*, volume 401 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2008.

[11] Yuanbo Guo, Zhengxiang Pan, and Jeff Heflin. LUBM: A benchmark for OWL knowledge base systems. *J. Web Sem.*, 3(2-3):158–182, 2005.

[12] Parul Gupta and AK Sharma. Ontology driven pre and post ranking based information retrieval in web search engines. *International Journal on Computer Science and Engineering*, 4(6):1241, 2012.

[13] Jonathan Hayes and Claudio Gutiérrez. Bipartite graphs as intermediate model for RDF. In *The Semantic Web - ISWC 2004: Third International Semantic Web Conference,Hiroshima, Japan, November 7-11, 2004. Proceedings*, pages 47–61, 2004.

[14] Xin He and Mark Baker. xhrank: Ranking entities on the semantic web. In *ISWC Posters&Demos*, CEUR Workshop Proceedings. CEUR-WS.org, 2010.

[15] Michiel Hildebrand, Jacco van Ossenbruggen, and Lynda Hardman. /facet: A browser for heterogeneous semantic web repositories. In *International Semantic Web Conference*, pages 272–285. Springer, 2006.

[16] Aidan Hogan, Andreas Harth, and Stefan Decker. Reconrank: A scalable ranking method for semantic web data with context. In *In 2nd Workshop on Scalable Semantic Web Knowledge Base Systems*, 2006.

[17] Jon M. Kleinberg. Authoritative sources in a hyperlinked environment. *J. ACM*, 46(5):604–632, September 1999.

[18] J Richard Landis and Gary G Koch. The measurement of observer agreement for categorical data. *Biometrics*, pages 159–174, 1977.

[19] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to information retrieval*. Cambridge University Press, 2008.

[20] Edgard Marx, Amrapali Zaveri, Mofeed Mohammed, Sandro Rautenberg, Jens Lehmann, Axel-Cyrille Ngonga Ngomo, and Gong Cheng. In *SUMPRE'16 at ESWC 2016*.

[21] Roberto Mirizzi, Azzurra Ragone, Tommaso Di Noia, and Eugenio Di Sciascio. Ranking the linked data: The case of dbpedia. In *ICWE*, volume 6189 of *Lecture Notes in Computer Science*, pages 337–354. Springer, 2010.

[22] Antonio J. Roa-Valverde and Miguel-Angel Sicilia. A survey of approaches for ranking on the web of data. *Inf. Retr.*, 17(4):295–325, August 2014.

[23] Muhammad Saleem, Maulik R Kamdar, Aftab Iqbal, Shanmukha Sampath, Helena F Deus, and Axel-Cyrille Ngonga Ngomo. Big linked cancer data: Integrating linked tcga and pubmed. *Web Semantics: Science, Services and Agents on the World Wide Web*, 27:34–41, 2014.

[24] Muhammad Saleem and Axel-Cyrille Ngonga Ngomo. Hibiscus: Hypergraph-based source selection for SPARQL endpoint federation. In *ESWC*, volume 8465 of *Lecture Notes in Computer Science*, pages 176–191. Springer, 2014.

[25] Muhammad Saleem and Axel-Cyrille Ngonga Ngomo. Hibiscus: Hypergraph-based source selection for sparql endpoint federation. In *European Semantic Web Conference*, pages 176–191. Springer, 2014.

[26] Julia Stoyanovich, Srikanta J. Bedathur, Klaus Berberich, and Gerhard Weikum. Entityauthority: Semantically enriched graph-based authority propagation. In *WebDB*, 2007.

[27] Giovanni Tummarello, Renaud Delbru, and Eyal Oren. Sindice.com: Weaving the open linked data. In *ISWC/ASWC*, volume 4825 of *Lecture Notes in Computer Science*, pages 552–565. Springer, 2007.

[28] Christina Unger, Lorenz Bühmann, Jens Lehmann, Axel-Cyrille Ngonga Ngomo, Daniel Gerber, and Philipp Cimiano. Template-based question answering over rdf data. In *Proceedings of the 21st international conference on World Wide Web*, pages 639–648. ACM, 2012.

[29] Christina Unger, Corina Forascu, Vanessa Lopez, Axel-Cyrille Ngonga Ngomo, Elena Cabrio, Philipp Cimiano, and Sebastian Walter. Question answering over linked data (qald-4). In *Working Notes for CLEF 2014 Conference*, 2014.

[30] Ricardo Usbeck. Combining linked data and statistical information retrieval - next generation information systems. In *ESWC*, volume 8465 of *Lecture Notes in Computer Science*, pages 845–854. Springer, 2014.