Novel Node Importance Measures to Improve Keyword Search over RDF Graphs

Elisa S. Menendez[1,3], Marco A. Casanova[1(✉)], Luiz A. P. Paes Leme[2], and Mohand Boughanem[3]

[1] Department of Informatics, PUC-Rio, Rio de Janeiro, RJ, Brazil
{emenendez, casanova}@inf.puc-rio.br
[2] Computer Science Institute, UFF, Niteroi, RJ, Brazil
lapaesleme@ic.uff.br
[3] Institut de Recherche en Informatique de Toulouse, IRIT, Toulouse, France
boughanem@irit.fr

Abstract. A key contributor to the success of keyword search systems is a ranking mechanism that considers the importance of the retrieved documents. The notion of importance in graphs is typically computed using centrality measures that highly depend on the degree of the nodes, such as PageRank. However, in RDF graphs, the notion of importance is not necessarily related to the node degree. Therefore, this paper addresses two problems: (1) how to define importance measures in RDF graphs; (2) how to use these measures to help compile and rank results of keyword queries over RDF graphs. To solve these problems, the paper proposes a novel family of measures, called InfoRank, and a keyword search system, called QUIRA, for RDF graphs. Finally, this paper concludes with experiments showing that the proposed solution improves the quality of results in two keyword search benchmarks.

Keywords: Keyword search · RDF · SPARQL · PageRank

1 Introduction

Keyword search is a well-known and convenient way for users to query large amounts of data, whether in Web pages or databases. The user simply types some terms, called *keywords*, and it is up to the system to retrieve the documents that best match the list of keywords. Search engines for Web pages popularized this kind of search. More recently, some of the Information Retrieval techniques used by Web search engines [17] were adapted to query databases to hide from users unfriendly SQL queries.

In the last decade, RDF emerged as a data model that represents data as a set of triples, which in turn induces a graph. This kind of modeling adds flexibility to describe resources and follows W3C standardized formats and ontologies. Considering that RDF graphs are interesting sources of knowledge that are also queried with unfriendly SPARQL queries, keyword search over RDF graphs (or briefly *RDF-KwS*) becomes a relevant research topic.

S. Hartmann et al. (Eds.): DEXA 2019, LNCS 11707, pp. 143–158, 2019.
https://doi.org/10.1007/978-3-030-27618-8_11

In Web Information Retrieval there are two main tasks: (1) matching keywords with indexed documents; (2) ranking the retrieved documents by order of relevance. RDF graphs present a further challenge, when compared to the Web, since the information that a user needs may not be in a single triple, but rather it is distributed over the graph. Hence, an answer for a keyword query over an RDF graph is better formalized as a minimal subgraph of the RDF graph that covers the keywords.

Summarizing, there are three main tasks in *RDF-KwS*: (1) finding pieces of information in the RDF graph; (2) assembling the retrieved pieces of information to compose complete answers; (3) ranking the complete answers. The main motivation of this work is how to construct an *RDF-KwS* system that covers these three tasks.

To achieve a good ranking mechanism, typical information retrieval systems rank the documents based not only on how well they match the keyword query, but also based on how important the documents are. The notion of importance for Web pages is typically computed using centrality measures for graphs created using the hyperlink structure of the Web. PageRank [6] and HITS [23] are some of the most popular centrality measures used in Web Information Retrieval. Their main idea is to assign high scores to pages that are referenced by many other important pages.

Returning to the RDF environment, the majority of the related work test their strategies using some RDF graph that reflects Web pages and their links [12, 15, 18, 21, 26], such as DBpedia[1], or using some dataset about co-authorship of research papers [3, 12, 33], such as DBLP[2]. We argue that PageRank or HITS variations work well for these types of RDF graphs because the incoming or outgoing edges indeed indicate the relevance of a resource. In the Web, it is reasonable that a Web page (or node) with several incoming edges is more important than a Web page with a few incoming edges. Likewise, in an RDF graph about research publications, the importance of an author is proportional to the number of accepted papers.

However, *RDF-KwS* operates over full RDF graphs, where the incoming or outgoing edges of a node do not necessarily indicate the node's importance with respect to any existing node relationship or, at least, it may be hard to detect which relationships would express the notion of importance. Thus, traditional measures may fail to compute the importance of a node. As an example, in an RDF graph representation of the Internet Movie Database – IMDb (www.imdb.com), instances of "common classes" (e.g. Genre, Language, Country, Company) have a high number of incoming edges. Hence, a traditional PageRank algorithm will assign scores to these common instances that are higher than the scores of popular movies and actors. Of course, we could manually assign weights to the object properties in order to capture their semantics, and use a Weighted PageRank or HITS Algorithm, as in [3, 10, 29]. However, one may argue that the manual assignment of weights is bothersome and subjective. Thus, other works focused on strategies to learn weights based on user feedback [1, 24, 27]. In addition to the difficulty of detecting relationships that express the importance of a graph node, it would be interesting to eliminate unwanted relationships that would distort traditional importance measures.

[1] http://dbpedia.org/sparql.

[2] http://dblp.uni-trier.de.

Summarizing, the problems addressed in this work are: (1) how to define importance measures in RDF graphs in which the importance of a node is not directly related to its degree; (2) how to use these measures to help compute and rank answers of keyword queries over RDF graphs.

To solve these problems, the first and key contribution of this paper is a novel family of importance measures for RDF graphs, collectively called *InfoRank*, that combine three intuitions: (I) *"important things have lots of information about them"*; (II) *"important things are surrounded by other important things"*; (III) *"few important relations (e.g. friends) are better than many unimportant relations (e.g. acquaintances)"*. They require neither the manual assignment of weights to object properties nor a training dataset to use as input to a learning algorithm.

The second contribution is an *RDF-KwS* system, called *QUIRA* (*QUerying with InfoRAnk*), which uses InfoRank: to narrow the retrieved pieces of information; to choose the best paths to connect the resources (nodes) in the graph; to rank the retrieved answers.

Finally, the third contribution of this paper consists of two enriched datasets, IMDb and MusicBrainz (http://musicbrainz.org), along with keyword search benchmarks adapted to the RDF environment. We use these datasets in our experiments to assess the correctness and the performance of InfoRank in the QUIRA system.

The rest of this paper is organized as follows. Section 2 summarizes related work. Section 3 defines the InfoRank measures. Section 4 describes the QUIRA keyword search system. Section 5 evaluates the performance of InfoRank in the QUIRA system. Finally, Sect. 6 contains the conclusions and suggestions for future work.

2 Related Work

Keyword Search over Structured Databases. Tools that implement keyword-based queries over relational databases [34] and RDF datasets have been investigated for some time. Since both fields have similar challenges, we discuss them together.

We may distinguish between tools that are *schema-based*, that use information about the conceptual schema to compile a keyword-based query into an SQL or SPARQL query, from those that are *graph-based*, which operate directly on the data.

BANKS [5] and BLINKS [16] are examples of relational graph-based tools, and Sindice [28] and *Structured LM* [11] are examples of RDF graph-based tools.

Relational schema-based tools explore the foreign keys declared in the relational schema to compile a keyword-based query into an SQL query with a minimal set of join clauses, based on the notion of *candidate networks* (CNs). This approach was first proposed in DISCOVER [19] and DBXplorer [2] and adopted in quite a few tools, including recent ones [9].

SPARK [37] offers an example of an early RDF schema-based tool. Tran et al. [31] combine the idea of generating summary graphs for the original RDF graph, using the class hierarchy, to generate and rank candidate SPARQL queries. QUICK [36] is a tool designed to translate keyword-based queries to SPARQL queries with the help of the users, who choose a set of intermediate queries, that the tool ranks and executes.

The *QUIOW* tool, our earlier implementation [13, 20], is schema-based and supports both the RDF and the relational environments by translating keyword queries into SPARQL or SQL queries. Although the tool proved efficient for an industrial dataset about petroleum, it had poor performance for an RDF graph representation of IMDb due to the large size and ambiguity of the domain. The importance measures introduced in this paper remediate these problems, as shown in Sect. 5.

Importance Measures for Structured Databases. ObjectRank [3] was one of the first proposals to compute a global importance score for database entities using PageRank. The authors transformed the structure of a relational database (RDB) into a graph, using foreign keys as links between entities, and then applied PageRank with manual weight assignment to different types of links. The authors evaluated their strategy using the DBLP dataset.

In RDF, other works that manually assign weights to use with PageRank are: Swoogle [10], which evaluated their strategy using documents crawled from the Web; Park et al. [29], which performs evaluation using their own small research dataset; and Beagle++ [7], which adapted ObjectRank to an RDF Graph about activity metadata in desktops.

TripleRank [12] represented an RDF graph as a tensor. Then, it used the PARAFAC decomposition of the tensor to induce groups of properties and resources, with authority and hub scores for the particular latent aspect (topic) the group represents. It showed how to use the result of the PARAFAC decomposition to guide a faceted browsing application. Finally, it tested the application in several experiments over RDF datasets with 5 to 55 thousand triples. PARAFAC decomposition proved interesting for faceted browsing exactly because it induces groups of properties and resources, together with authority and hub scores. However, it is not clear how to extend this strategy to the context of keyword search, not to mention the problem of computing the PARAFAC decomposition of tensors with 200+ million non-zero entries, as in the experiments described in Sect. 5.

More recently, FORK [24] adapted ObjectRank to Linked Data. The main contribution of the work is a learning algorithm for property weights based on user relevance feedback, instead of the manual assignment of weights. The authors evaluated their strategy using DBpedia and results showed that FORK achieves the best ranking method when compared to baseline approaches. Similarly, DBtrends [25] uses query logs to improve its ranking function.

As mentioned in the introduction, DBpedia and DBLP are highly influenced by link semantics: DBLP through authorship links, and DBpedia through links derived from Wikipedia, such as *wikiPageRedirects, wikiPageDisambiguates, primaryTopic, etc*. Furthermore, in the LOD cloud (http://lod-cloud.net), DBpedia has many incoming links from other RDF datasets.

For further references that focus on ranking strategies for degree-dependent datasets, such as DBpedia or DBLP, we refer the reader to [4, 30, 35]. We continue our discussion with some alternative strategies that do not highly depend on node degree.

Graves et al. [14] proposes the use of closeness centrality for undirected graphs and evaluates the strategy using three small datasets. The authors compare their strategy

with a ranking using the number of incoming edges. The problem with closeness centrality is that it is not efficient for large RDF graphs.

Although the work presented in [22] is not specific to RDF graphs, it proposes the *degree decoupled PageRank* technique that penalizes or boosts the importance of the node degree in recommendation graphs, depending on the domain characteristics. They argue that, in some contexts, the importance of the node can be inversely proportional to its degree. The authors performed an evaluation using graphs extracted from IMDb, Last.fm, DBLP and Epinions. From results for the IMDb dataset, they noticed that, for a movie recommendation graph, traditional PageRank performs better; however, for an actor recommendation graph, the node degree actually needs to be penalized. They argue that, when an actor plays in a large number of movies, he probably is a non-discriminating ("B movie") actor, whereas, when an actor is associated with relatively few movies, he may be a more discriminating ("A movie") actor.

3 The InfoRank Importance Measures

3.1 Background on Importance Measures

Importance measures have as goal to identify the most important or central node in a graph, depending on what importance means. A simple way to compute the importance of a node is just to analyze its degree. However, this returns a local measure of importance, whereas in some contexts a global analysis of the graph is preferable. For instance, the *Betweenness Centrality* counts the number of shortest paths going through a node; hence it is able to identify important connectors in a graph. The *Closeness Centrality* measures the average distance from a node to all other nodes, hence the more central a node is, the closer it is to all other nodes.

Other types of importance measures try to capture the idea that "it is not about what you know, but who you know". That is, the notion of importance is given by how well-connected a node is to other important nodes. PageRank [6] is the most popular importance measure of this type. Using the hyperlink structure of the Web, the basic idea is that, if a Web page has links from other high-quality Web pages, then that is an indication that it is likely to be worth looking at the page.

PageRank can be computed using an iterative method, called *Power Iteration*. Let $G = (V, E)$ be a directed graph and $PR(r, i)$ be the PageRank score of a node $r \in V$ calculated at iteration i. First, the method initializes all scores with the same value:

$$PR(r, 0) = 1/N \tag{1}$$

where N is the total number of nodes in G. Then, for $0 < i < x$, it iterates until the computation of the score converges or exceeds x, the maximum number of iterations:

$$PR(r, i) = \frac{1 - \alpha}{N} + \alpha \sum\nolimits_{s \in M_I(r)} \frac{PR(s, i - 1)}{d_O(s)} \tag{2}$$

where α is a dumping factor (usually set to 0.85), $M_I(r)$ is the set of nodes that have a link to r and $d_O(s)$ is the number of outgoing links from s.

One variant of PageRank uses link weights to give more importance for certain types of links. The *Weighted PageRank* PR_W is defined as:

$$PR_W(r, 0) = 1/N \tag{3}$$

$$PR_W(r, i) = \frac{1-\alpha}{N} + \alpha \sum_{s \in M_I(r)} \frac{PR_W(s, i-1)}{d_O(s)} * w(r, s) \tag{4}$$

where $w(r, s)$ is a weight between 0 and 1 of edge $(r, s) \in E$.

3.2 The Intuitions Behind InfoRank

Following the intuition that *"important things have lots of information about them"* and observing the way that RDF graphs are modeled, we notice that more important nodes are usually associated with more literals (information) through datatype properties than less important nodes. As an example, in IMDb, a movie with international projection, such as *Titanic* (1997), has 205 literals with trivia, 134 literals with quotes said by the characters, 180 triples with tags, and so on. In fact, there are a total of 1,297 literals describing the movie *Titanic*. By contrast, a movie with only national projection, such as the Brazilian movie *O Auto da Compadecida*, has only 70 literals. Furthermore, in a multilingual dataset, such as DBpedia, *Titanic* has the label translated in many languages (e.g. Japanese, Russian, French, Spanish, etc.), while the Brazilian movie has the label only in Portuguese and English.

The second intuition that we follow is inspired by PageRank and says that *"important things are surrounded by other important things"*. For instance, *Titanic* has links through object properties with actors *Kate Winslet* and *Leonardo Dicaprio*, which are also important nodes in the graph. As in [14], we agree that, in RDF graphs, the direction of an object property does not have the same meaning as a Web hyperlink since a property is often found in its inverse form (e.g. directedBy/hasDirector). Given that, we treat an RDF graph as undirected and consider all neighbors of a node (i.e. all other nodes that have an object property linked to it) when propagating the importance with PageRank.

We further improve this intuition by introducing a third one that says "few friends are better than many acquaintances". As discussed in the introduction, the typical centrality measures are highly dependent on the degree of the node. In our work, we do not want to boost (or penalize) the degree importance, but we focus on a strategy that favors the quality of relations, rather than their quantity, that is, we prefer an approach that captures the notion that *"few important relations (e.g. friends) are better than many unimportant relations (e.g. acquaintances)"*.

3.3 Ranking Resources with InfoRank

Let T be a set of RDF triples. Assume that T contains schema information and that it is possible to identify the set C of *classes* defined in T, the set P of *object properties* defined in T, the set L of *literals* defined in T, and the set R of *blank nodes* and *(class) instances* defined in T, i.e., $r \in R$ iff there is a triple $(r, rdf{:}type, c) \in T$ such that $c \in C$.

Instance Informativeness. The level of "informativeness" of a resource measures how informative the resource is. As discussed in the previous section, information is represented as literals in RDF graphs. However, data resources (instances) usually have more literals than metadata resources (classes and properties). Hence, we first focus our strategy on the informativeness of instances.

The *informativeness* of an instance $r \in R$, denoted $IW(r)$, is defined as the number of triples of the form $(r, p, v) \in T$, where $v \in L$.

Ranking Schema Elements. Continuing our strategy based on instance informativeness, we say that "important classes usually have informative instances" and "important properties are usually those connecting informative instances".

The *InfoRank* of a class $c \in C$, denoted $IR(c)$, is defined as the maximum value of $IW(r)$ of all instances of class c. We will rank classes by descending order of $IR(c)$.

Likewise, the *InfoRank* of an object property $p \in P$, denoted $IR(p)$, is defined as the maximum value of $IW(r) + IW(s)$ of all triples of the form $(r, p, s) \in T$. We will rank object properties by descending order of $IR(p)$.

Ranking Data. Note that we used only *Intuition I* in our strategies to rank metadata resources. However, we propose a combination of the three intuitions to rank data, that is, the instances and blank nodes.

Let $r, s \in R$ and $p \in P$. Assume that $(r, p, s) \in T$ or $(s, p, r) \in T$, that is, ignore the direction of the object property p. The *normalized weight* of (r, p), denoted $W(r, p)$, is defined as:

$$W(r,p) = IR(p) / \sum_{q \in P \text{ and } ((r,q,t) \in T \text{ or } (t,q,r) \in T)} IR(q) \tag{5}$$

Note that the normalized weight $W(r, p)$ does not depend on "who" the neighbors of v are, but it depends only on how they are connected to r, that is, it considers the InfoRank scores of properties p and q.

Then, we compute PageRank using $W(r, p)$ as the edge weights:

$$PR_W(r,i) = \frac{1-\alpha}{N} + \alpha \sum_{(r,p,s) \in T \text{ or } (s,p,r) \in T} PR_W(s, i-1) * W(r,p) \tag{6}$$

where, as in Eq. (2), N is the total number of nodes in G and α is a dumping factor.

The InfoRank score of an instance r, denoted $IR(r)$, is the final PageRank score of r after x iterations, $PR_W(r, x)$, weighted by the informativeness of r, $IW(r)$:

$$IR(r) = PR_W(r,x) * IW(r) \tag{7}$$

4 The QUIRA Keyword Search System

4.1 Overview

Recall that, given a graph G and a set M of nodes of G, a *Steiner tree* S for M is a tree whose nodes contain all nodes in M (and perhaps other nodes of G) and whose edges are edges of G. The Steiner tree S is *minimal* iff no other Steiner tree for M has fewer nodes than S.

As stated in the introduction, an answer of a keyword query over an RDF graph G is one or more minimal subgraphs that cover all keywords. Hence, a naïve approach to address the three main tasks would be: (1) find a set M of nodes of G that match all keywords; (2) find a minimal Steiner tree for M; (3) if there is more than one answer, rank the answers according to some criterion. Note that computing a Steiner tree avoids including unnecessary edges to connect the nodes.

There are two main problems with this approach that make it infeasible for most RDF graphs: (1) the set of nodes that match the keywords can be large; and (2) computing a minimal Steiner tree is an NP-complete problem.

Therefore, in previous work [13, 20], we described a tool, called *QUIOW*, that explores schema information to minimize these problems. The schema information is organized as a *schema graph*, as illustrated in Fig. 1. Without going into the details, in the first stage, *QUIOW* groups the keyword matches around classes, that is, *QUIOW* identities the properties whose values match keywords and creates groups of properties that have the same class as domain. In the second stage, *QUIOW* generates a Steiner tree for the set of classes found in the first stage over the schema graph (which is typically a small graph). In the third stage, *QUIOW* synthesizes a SPARQL query using the Steiner tree. Finally, the triplestore processes the SPARQL query synthesized to actually compute an answer to the keyword query.

In this work, we maintain the idea of grouping the matches in classes/properties to generate SPARQL templates. However, we completely reformulated the strategy to compute the templates to take advantage of InfoRank, as described in what follows.

4.2 Finding Pieces of Information in an RDF Graph

In this section, we present a greedy algorithm that takes keywords as input and returns the best set of class/property groups, as defined in Sect. 4.1.

Table 1 shows examples of groups in an IMDb dataset. The *count* column indicates that there are five movies named Titanic, one actress named *Kate Winslet* and four Episodes also named *Kate Winslet*. The *info_score* column is the aggregation of the InfoRank scores of all resources of a given group. For instance, all resources of group u_1 sum up to 0.0099 of InfoRank scores. Finally, group u_4 indicates that there is an *rdfs:Class* labeled Movie with score 1,468. We define a function *accum_score*(J, v) that simply counts the number of keywords from a set of keywords $J = \{j_1, j_2, \ldots, j_n\}$ that occurs in a literal value v. As an example, consider the keyword query $K = \{kate, winslet, titanic\}$ and the data in Table 1. The non-zero *accum* scores are:

$$accum\_score(\{kate, winslet, titanic\}, Kate\ Winslet) = 2$$
$$accum\_score(\{kate, winslet, titanic\}, Titanic) = 1$$

Table 1. Example of groups from IMDb.

Group	Class	Property	Value	info_score	Count
u_1	*imdb:Movie*	*rdfs:label*	Titanic	0.0099	5
u_2	*imdb:Actress*	*rdfs:label*	Kate Winslet	0.0010	1
u_3	*imdb:Episode*	*rdfs:label*	Kate Winslet	0.0000068	4
u_4	*rdfs:Class*	*rdfs:label*	Movie	1,468	1

Algorithm 1 presents an overview of a greedy strategy to obtain the best groups that satisfy a keyword query K. The strategy first gives priority to class matches. Then, it searches the groups looking for properties and data matches (e.g. *Titanic*, *Kate Winslet*).

Algorithm 1. Greedy Strategy to return the best set of groups that match a *Keyword Query*.

Input: A keyword query K and the set of groups U
Output: A subset of groups M
J = all keywords in K
M = empty list of groups
While J is not empty
  u = find in U a *class* group with the highest *accum_score* given J, use the highest *info_score* to disambiguate
  If a match is found
    add u to M, remove the keywords matched in u from J
  Else
    u = find in U a property or data group (i.e. *class* is *not rdfs:Class*) with the highest *accum_score* given J, use the highest *sum_score* to disambiguate
    If a match is found
      add u to M, remove the keywords matched in u from J
  If J did not change
    break

As an example of the algorithm, consider again K = {*kate*, *winslet*, *titanic*}. In the first iteration of the while loop, J = {*kate*, *winslet*, *titanic*} and the algorithm chooses group u_2. Although groups u_2 and u_3 have the same *accum_score* for J, the *info_score* is higher for u_2. In the second iteration, J = {*titanic*} and the algorithm chooses group u_1, and the loop ends. At the end of this step, we generate SPARQL templates that satisfy the groups retrieved in Algorithm 1. The resulting templates for K = {*kate*, *winslet*, *titanic*} are shown in Table 2.

Table 2. Templates generated for $K = \{kate, winslet, titanic\}$.

Template	Interpretation
?r1 rdf:type :Movie. ?r1 rdfs:label ?v1. **filter**(**contains**(?v1, 'titanic'))	All movies with label *titanic*
?r2 rdf:type :Actress. ?r2 rdfs:label ?v2. **filter**(**contains**(?v2, 'kate winslet'))	All actresses with label *kate winslet*

4.3 Connecting and Ranking

Connecting. The second task of the *RDF-KwS* process, i.e., connecting pieces of information, consists of finding a minimal Steiner tree between the classes of the groups retrieved in the first task. The Steiner tree is computed over the schema graph, a representation of the schema as in Fig. 1. Since the number of classes in an RDF Dataset is usually not large, it is feasible to compute a minimal Steiner tree.

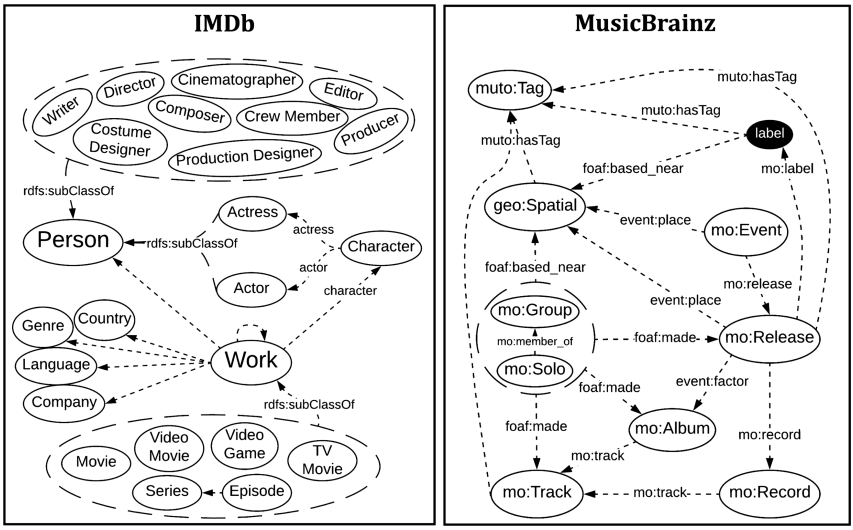

Fig. 1. IMDb schema.

Completing our templates example presented in Table 2, this step generates one more template (*?r1 ?p1 ?r2*), which says that a movie *?r1* and an actress *?r2* are connected through some property *?p1*.

Ranking. In the third task, i.e., ranking the results, we materialize triples together with the InfoRank score (e.g. *:Kate_Winslet :inforank "*0.0010*"*). Hence, we can generate templates for these triples (e.g. *?r1 :inforank ?s1*, *?r2 :inforank ?s2*), and synthesize a SPARQL query with an ORDER BY clause that aggregates the scores of all instances from the templates. Finally, the following SPARQL query is synthesized for $K = \{kate, winslet, titanic\}$.

```
select * where {
    ?r1 rdf:type :Movie . ?r1 rdfs:label ?v1 . filter(contains(?v1, 'titanic'))
    ?r2 rdf:type:Actress . ?r2 rdfs:label ?v2 . filter(contains(?v2, 'kate winslet'))
    ?r1 ?p1 ?r2 . ?r1 :inforank ?s1 . ?r2 :inforank ?s2 . }
order by desc (?s1 + ?s2)
```

5 Evaluation

5.1 Setup

In order to evaluate our strategy, we downloaded the relational IMDb dataset (https://sites.google.com/site/ontopiswc13/home/imdb-mo) in MySQL and used Oracle 12c to transform it to RDF via R2RML. We used an RDF dump of MusicBrainz as our second dataset; however, since the given dump was incomplete, we enriched it with DBpedia information. The IMDb and MusicBrainz datasets have around 200 million triples. Figure 1 shows an overview of the schemas.

All experiments were conducted using a RESTful Web application developed in Java. The app ran on a macOS Sierra, 1,7 GHz Intel Core i5 RAM 4 GB. To store and manage the RDF data, we used Oracle 12c, running on a 2x deca-core Intel(R) Xeon(R) CPU E5-2640 v4 @ 2.40 GHz, 128 GB RAM, 32 KB Cache L1.

The datasets, benchmarks, and a detailed description of the experiments are available at the QUIRA Web page (https://sites.google.com/view/quira/).

5.2 Ranking Experiments

This section presents experiments to assess the potential of InfoRank as an importance measure to be used in a keyword search system over RDF graphs.

Table 3 presents the InfoRank score and the node degree of several classes and properties (i.e., metadata) from IMDb. We argue that, in an IMDb dataset, the most important classes are those that represent works (movies, TV series, etc.) and people (actors, actresses, directors, etc.), which is the result that InfoRank gives. Note that if we ranked the results using the degree, the order of classes would be Character, Person, Work; however, a typical IMDb user is likely to be more interested in movies and other type of works rather than in characters. Furthermore, the top properties are those connecting movies, such as follows/followed_by, which indicates that a movie is a sequence of another.

Table 3. IMDb metadata ranking computed by InfoRank.

#	Class	Info	Degree	Property	Info	Degree
1	imdb:Work	1,619	2,410,207	imdb:follows	2,538	332,551
2	imdb:Person	1,482	3,913,018	imdb:followed_by	2,538	332,548
3	imdb:Character	3	19,419,994	imdb:edited_from	2,538	14,103
4	imdb:Company	3	224,971	imdb:edited_into	2,538	14,103
5	imdb:Language	2	364	imdb:referenced_in	2,509	223,535
6	imdb:Country	2	319	imdb:references	2,509	223,532
7	imdb:Genre	2	46	….		

Table 4 shows the top 10 instances induced by PageRank and InfoRank. With PageRank, the top instances are highly connected nodes, such as countries, language and genres. However, we argue that, when considering a movies dataset, we would expect as top instances popular movies, series, actors, actresses, etc.

To indicate popularity, Table 4 also shows the users' rating of works extracted from the IMDb Web site. In the case of a person, we extracted the most rated work that she stared, directed, produced, etc. InfoRank results show highly rated work/person, such as *Star Wars*, *The Wizard of Oz*, *Titanic* and *Morgan Freeman*. The results show some TV Series with lower rates because they have a considerable level of informativeness (*General Hospital* – 375 literals; *Days of Our Lives* – 232 literals), and also a high degree through property *:episode_of_series,* since they have been on the air for a long time. Likewise, the results show some hosts from TV Shows that also have been on the air for a long time. Although InfoRank results show a few less popular works/people, we argue that InfoRank results correspond better to what users would expect in an IMDb dataset.

A similar scenario happens with MusicBrainz, in which the PageRank top instances also include countries. However, the InfoRank top instances include famous musicians, such as *Elvis Presley*, *Mozart*, *Beethoven*, *Bob Dylan*, etc.

Table 4. IMDb top 10 instances induced by PageRank and InfoRank.

#	PageRank			InfoRank		
	Instance	Class	User rating	Instance	Class	User rating
1	English	Language	–	Star Wars	Movie	8.6
2	USA	Country	–	Dolly Parton	Actress	6.8
3	Short	Genre	–	Jay Leno	Actor	5.3
4	Drama	Genre	–	Morgan Freeman	Actor	8.6
5	Comedy	Genre	–	The Wizard of Oz	Movie	8.0
6	Documentary	Genre	–	General Hospital	Series	6.7
7	UK	Country	–	Days of Our Lives	Series	5.3
8	Spanish	Language	–	Bob Barker	Actor	7.7
9	German	Language	–	Titanic	Movie	7.8
10	France	Country	–	Around the World in 80 Days	Movie	6.8

5.3 Keyword Search Experiments

To evaluate the impact of using InfoRank in a keyword search system over IMDb, we used all 50 queries (adapted to the RDF schema) from Coffman's IMDb Benchmark [8]. We ran versions of *QUIRA* using a variety of ranking measures. Table 5 presents the Mean Average Precision (MAP) [32], the total elapsed time and the number of iterations needed to compute the measures.

The measures in Table 5 include InfoRank, a version of PageRank considering the graph as undirected, the HITS Authorities, which prioritizes nodes with high in-degree, and HITS Hubs, which prioritizes nodes with high out-degree. We also include the Degree-decoupled (DD) PageRank [22] with a penalization parameter of 0.5. Note that we compared InfoRank neither with any approach that uses manually weighted links due to their subjectivity nor with approaches that learn weights from user feedback since we face the cold start problem. Moreover, we eliminated measures that are not computed efficiently in large graphs, such as the closeness centrality.

Table 5. IMDb results.

	Time (min)	Iterations	MAP
InfoRank	28	24	0.82
PageRank	27	30	0.76
HITS Authorities	25	12	0.73
HITS Hubs	25	12	0.30
DD PageRank p = 0, 5	38	37	0.54

Analyzing the results (not shown here for brevity), we noted that PageRank and HITS Authorities fail when choosing class Character, instead of class Work, in queries where a Steiner tree needs to be computed. They also fail in the ranking step for some keyword queries due to the high dependency on the degree. For example, Fig. 2 shows the results for PageRank and InfoRank for the query "actor terminator", whose expected results are the movies stared by *Arnold Schwarzenegger*. PageRank ranks first the voice actor *Jim Cummings* because his node has a high degree, since voice actors are usually cast several times, whereas InfoRank correctly returns the movies *Terminator 2: Judgment Day* and *The Terminator* starred by *Arnold Schwarzenegger*.

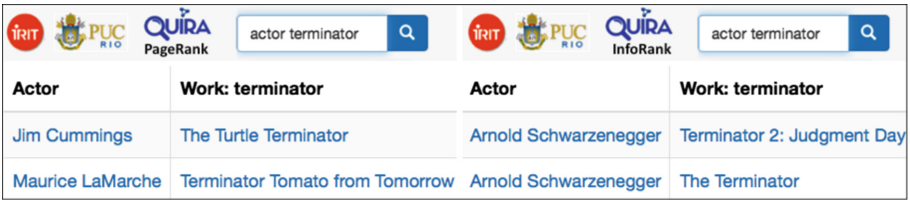

Fig. 2. Result for query $K = \{actor, terminator\}$ in PageRank and InfoRank.

The HITS Hubs fails in all queries that refer to a person (e.g. *Denzel Washington*) since instances of class Person do not have outgoing edges. Furthermore, the Degree Decoupled (DD) PageRank fails because it penalizes instances with a high degree, whereas many important instances (e.g. *Star Wars*) have a high degree.

To summarize, InfoRank achieves the best MAP result in Coffman's IMDb Benchmark queries, since it successfully finds a balance between degree and informativeness. Furthermore, Table 5 indicates that these type of centrality measures, based on the Power Iteration method, can be computed in a feasible time.

Finally, we used 25 queries from QALD-2 (https://github.com/ag-sc/QALD) to evaluate the impact of InfoRank in a keyword search system over MusicBrainz. InfoRank achieved a MAP of 0.80 and PageRank a MAP of 0.75. For instance, PageRank gives a priority to music albums that have a higher number of tracks, since more tracks imply more links. However, we argue that the number of tracks is not necessarily related to the importance of an album.

6 Conclusions and Future Work

In this paper, we addressed two problems: (1) how to define importance measures in RDF graphs; (2) how to use these measures to help compute and rank answers of keyword queries over RDF graphs. To solve these problems, we proposed a novel family of measures, called InfoRank, and a keyword search system, called *QUIRA*, for RDF graphs. *QUIRA* uses the proposed importance measures: to narrow the retrieved pieces of information; to choose the best paths to connect the resources (nodes) in a graph; and to rank the retrieved answers. We concluded with experiments that show that the proposed solution improves the quality of results in popular keyword search benchmarks.

As future work, we plan to use InfoRank to improve Entity Linking and Entity Summarization solutions, to evaluate *QUIRA* with larger schemas, and to test ranking functions that take advantage of domain knowledge.

Acknowledgments. This work was partly funded by CAPES under grant 88881.134081/2016-01, by CNPq under grants 153908/2015-7, 302303/2017-0 and by FAPERJ under grant E-26-202.818/2017.

References

1. Agarwal, A., et al.: Learning to rank networked entities. In: Proceedings 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD 2006, pp. 14–23 (2006)
2. Agrawal, S., et al.: DBXplorer: a system for keyword-based search over relational databases. In: Proceedings 18th International Conference Data Engineering, pp. 5–16 (2002)
3. Balmin, A., et al.: ObjectRank: authority-based keyword search in databases. In: Proceedings 13th International Conference on Very Large Data Bases - Volume 30, pp. 564–575 (2004)

4. Bast, H., et al.: Semantic Search on Text and Knowledge Bases. Foundation and Trends® in Information Retrieval, vol. 10, no. 2–3, pp. 119–271 (2016)
5. Bhalotia, G., et al.: Keyword searching and browsing in databases using BANKS. In: Proceedings 18th International Conference on Data Engineering, pp. 431–440. IEEE Computer Society (2002)
6. Brin, S., Page, L.: The anatomy of a large-scale hypertextual Web search engine. Comput. Netw. ISDN Syst. **30**(1–7), 107–117 (1998)
7. Chirita, P.A., et al.: Beagle ++: semantically enhanced searching and ranking on the desktop. In: The Semantic Web: Research and Applications - ESWC 2006, pp. 348–362 (2006)
8. Coffman, J., Weaver, A.C.: A framework for evaluating database keyword search strategies. In: Proceedings 19th ACM International Conference on Information and Knowledge Management, pp. 729–738 (2010)
9. De Oliveira, P., et al.: Ranking Candidate Networks of relations to improve keyword search over relational databases. In: Proceedings 31st International Conference on Data Engineering, pp. 399–410 (2015)
10. Ding, L., et al.: Swoogle: a search and metadata engine for the semantic web. In: Proceedings 13th ACM Conference on Information and Knowledge Management - CIKM 2004, pp. 652–659 (2004)
11. Elbassuoni, S., Blanco, R.: Keyword search over RDF graphs. In: Proceedings 20th ACM International Conference on Information and Knowledge Management - CIKM 2011, pp. 237–242 (2011)
12. Franz, T., Schultz, A., Sizov, S., Staab, S.: TripleRank: ranking semantic web data by tensor decomposition. In: Bernstein, A., et al. (eds.) ISWC 2009. LNCS, vol. 5823, pp. 213–228. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-04930-9_14
13. García, G.M., et al.: RDF Keyword-based query technology meets a real-world data set. In: Proceedings 20th International Conference on Extending Database Technology (EDBT), pp. 656–667 (2017)
14. Graves, A., et al.: A method to rank nodes in an RDF graph. In: Proceedings 7th International Semantic Web Conference, pp. 84–85 (2008)
15. Harth, A., Kinsella, S., Decker, S.: Using naming authority to rank data and ontologies for web search. In: Bernstein, A., et al. (eds.) ISWC 2009. LNCS, vol. 5823, pp. 277–292. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-04930-9_18
16. He, H., et al.: BLINKS: ranked keyword searches on graphs. In: Proceedings 2007 ACM International Conference on Management of Data - SIGMOD 2007, pp. 305–316 (2007)
17. Hiemstra, D.: Information retrieval models. In: Information Retrieval: Searching in the 21st Century, pp. 1–17 (2009)
18. Hogan, A., et al.: ReConRank: a scalable ranking method for semantic web data with context. In: Proceedings 2nd Workshop on Scalable Semantic Web Knowledge Base System (2006)
19. Hristidis, V., Papakonstantinou, Y.: Discover: keyword search in relational databases. In: Proceedings 28th International Conference on Very Large Databases, pp. 670–681. Elsevier (2002)
20. Izquierdo, Y.T., García, G.M., Menendez, E.S., Casanova, M.A., Dartayre, F., Levy, C.H.: *QUIOW*: a keyword-based query processing tool for RDF datasets and relational databases. In: Hartmann, S., Ma, H., Hameurlain, A., Pernul, G., Wagner, R.R. (eds.) DEXA 2018. LNCS, vol. 11030, pp. 259–269. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-98812-2_22
21. Kasneci, G., et al.: NAGA: searching and ranking knowledge. In: Proceedings 2008 IEEE 24th International Conference on Data Engineering, pp. 953–962 (2008)

22. Kim, J.H., et al.: PageRank revisited: on the relationship between node degrees and node significances in different applications. In: Proceedings 5th International Workshop on Querying Graph Structured Data at EDBT/ICDT, pp. 1–8 (2016)
23. Kleinberg, J.M.: Authoritative sources in a hyperlinked environment. J. ACM **46**(5), 604–632 (1999)
24. Komamizu, T., Okumura, S., Amagasa, T., Kitagawa, H.: FORK: feedback-aware objectrank-based keyword search over linked data. In: Sung, W.K., et al. (eds.) Information Retrieval Technology AIRS 2017. Lecture Notes in Computer Science, vol. 10648, pp. 58–70. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70145-5_5
25. Marx, E., et al.: DBtrends: exploring query logs for ranking RDF data. In: Proceedings 12th International ACM Conference on Semantic Systems, pp. 9–16 (2016)
26. Mirizzi, R., Ragone, A., Di Noia, T., Di Sciascio, E.: Ranking the linked data: the case of DBpedia. In: Benatallah, B., Casati, F., Kappel, G., Rossi, G. (eds.) ICWE 2010. LNCS, vol. 6189, pp. 337–354. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13911-6_23
27. Nie, Z., et al.: Object-level ranking. In: Proceedings 14th International Conference on World Wide Web - WWW 2005, pp. 567–674 (2005)
28. Oren, E., et al.: Sindice.com: a document-oriented lookup index for open linked data. Int. J. Metadata Semant. Ontol. **3**(1), 37–52 (2008)
29. Park, H., et al.: A link-based ranking algorithm for semantic web resources. J. Database Manag. **22**(1), 1–25 (2011)
30. Roa-Valverde, A.J., Sicilia, M.-A.: A survey of approaches for ranking on the web of data. Inf. Retr. **17**(4), 295–325 (2014)
31. Tran, T., et al.: Top-k exploration of query candidates for efficient keyword search on graph-shaped (RDF) data. In: Proceedings 25th International Conference on Data Engineering, pp. 405–416 (2009)
32. Turpin, A., Scholer, F.: User performance versus precision measures for simple search tasks. In: Proceedings 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 11–18 (2006)
33. Wei, W., et al.: Rational Research model for ranking semantic entities. Inf. Sci. **181**(13), 2823–2840 (2011)
34. Yu, J.X., et al.: Keyword Search in Databases. Morgan & Claypool, San Francisco (2010)
35. Yumusak, S., et al.: A short survey of linked data ranking. In: Proceedings 2014 ACM Southeast Regional Conference on - ACM SE 2014, pp. 1–4 (2014)
36. Zenz, G., et al.: From keywords to semantic queries - Incremental query construction on the semantic web. Web Semant. Sci. Serv. Agents W.W.W. **7**(3), 166–176 (2009)
37. Zhou, Q., Wang, C., Xiong, M., Wang, H., Yu, Y.: SPARK: adapting keyword query to semantic search. In: Aberer, K., et al. (eds.) ASWC/ISWC -2007. LNCS, vol. 4825, pp. 694–707. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-76298-0_50