Automatic Construction of Benchmarks for RDF Keyword Search Systems Evaluation

Angelo Batista Neves[1][a], Luiz André P. Paes Leme[2][b], Yenier Torres Izquierdo[1][c] and Marco Antonio Casanova[1][d]
[1]*Pontifical Catholic University of Rio de Janeiro, Rio de Janeiro, RJ, Brazil*
[2]*Universidade Federal Fluminense, Niterói, RJ, Brazil*

Keywords: Benchmark, Keyword Search, Resource Description Framework (RDF), Offline, Computation.

Abstract: Keyword search systems provide users with a friendly alternative to access Resource Description Framework (RDF) datasets. The evaluation of such systems requires adequate benchmarks, consisting of RDF datasets and keyword queries, with their correct answers. However, the sets of correct answers such benchmarks provide for each query are often incomplete, mostly because they are manually built with experts' help. The central contribution of this paper is an offline method that helps build RDF keyword search benchmarks automatically, leading to more complete sets of correct answers, called *solution generators*. The paper focuses on computing sets of generators and describes heuristics that circumvent the combinatorial nature of the problem. The paper then describes five benchmarks, constructed with the proposed method and based on three real datasets, DBpedia, IMDb, and Mondial, and two synthetic datasets, LUBM and BSBM. Finally, the paper compares the constructed benchmarks with keyword search benchmarks published in the literature.

1 INTRODUCTION

Keyword search is a very popular information discovery method because it allows naive users to retrieve information without any knowledge about schema details or query languages. The user specifies a few terms, called *keywords*, and it is up to the system to retrieve the documents, such as Web pages, that best match the keywords. Recently, keyword search applications designed for Resource Description Framework (RDF) datasets (or RDF graphs) have emerged.

Tools for keyword search over RDF datasets, or *RDF-KwS tools*, have three main tasks: (1) retrieve nodes in the RDF graph that the keywords specify; (2) discover how they are interrelated to compose complete answers; and (3) rank these answers (Menendez et al., 2019). Hence, answers for keyword searches over RDF datasets are not just sets of nodes but sets of nodes and paths between them, i.e., subgraphs of the dataset.

RDF-KwS tools are evaluated using benchmarks with sets of information needs and their respective lists of correct answers, possibly ordered, for a given dataset. Despite the many existing benchmarks for structured data, (Coffman and Weaver, 2010; Dosso and Silvello, 2020; Dubey et al., 2019; Trivedi et al., 2017) these benchmarks have at least three limitations when it comes to RDF-KwS: (1) they are frequently built for relational data; (2) they are incomplete in the sense that they do not cover many reasonable answers; and (3) they are not always publicly available.

To remedy the first limitation, some authors (García et al., 2017) adapted benchmarks originally developed for relational databases. However, the adaptation requires the triplification of relational databases and benchmark data, leading to problems while comparing different systems using different triplifications.

As an example of the incompleteness of existing benchmarks, consider the keyword query "Mauritius India", which is Query 43 for the Mondial dataset in Coffman's benchmark. The list of correct answers in the benchmark covers just the organizations that both countries belong to. However, there are other answers, which express the geographical relationship between Mauritius Islands and India, that should have been included in the list of correct answers. Incompleteness in this sense is a serious problem that is difficult to

[a] https://orcid.org/0000-0001-8043-1510
[b] https://orcid.org/0000-0001-6014-7256
[c] https://orcid.org/0000-0003-0971-8572
[d] https://orcid.org/0000-0003-0765-9636

Neves, A., Leme, L., Izquierdo, Y. and Casanova, M.
Automatic Construction of Benchmarks for RDF Keyword Search Systems Evaluation.
DOI: 10.5220/0010519401260137
In *Proceedings of the 23rd International Conference on Enterprise Information Systems (ICEIS 2021) - Volume 1*, pages 126-137
ISBN: 978-989-758-509-8

overcome in manually constructed benchmarks.

This paper overcomes these limitation by addressing the general problem of constructing RDF keyword search benchmarks in a holistic approach, i.e., from the definition of keyword queries to the computation of correct answers. This is a challenging problem for two fundamental and interrelated reasons: (1) RDF-KwS tools vary widely and compute different - albeit correct - answers for the same keyword query; (2) computing the set of all correct answers for a given keyword query over an RDF dataset leads to an explosive combinatorial problem.

In more detail, this paper proposes a method that, given an RDF dataset, automatically defines keyword queries and their respective correct answers. The method is designed as an offline process to benefit from less stringent response time constraints. It deals with the combinatorial nature of the problem by adopting heuristics based on the concepts of *seeds* and *solution generators*, which are additional contributions of the paper.

The rest of this paper is organized as follows. Section 2 covers related work. Section 3 contains the required definitions. Section 4 overviews the proposed method. Section 5 describes the automatic generation of keyword queries. Section 6 details the generation of answers. Section 7 evaluates the proposed benchmark generation method. Finally, Section 8 contains the conclusions and suggestions for future work.

2 RELATED WORK

A crucial aspect of keyword search systems is their evaluation. In recent years, the research community concentrated on evaluating keyword search systems over relational databases (Bast et al., 2016) and entity retrieval (Balog and Neumayer, 2013). Examples of relational benchmarks are the Coffman's benchmark (Coffman and Weaver, 2010), that uses samples of Mondial, IMDb, and Wikipedia, and the Oliveira's benchmark (Oliveira Filho, 2018), that uses Mondial, IMDb, DBLP, and Northwind.

Unfortunately, benchmarks to assess RDF keyword search systems are scarce (Dosso and Silvello, 2020). To remedy this situation, some authors (García et al., 2017), adapted relational benchmarks to RDF. However, this approach depends on the triplification of relational databases and does not easily induce complete sets of correct query answers (Izquierdo et al., 2018).

In fact, state-of-the-art RDF keyword search systems use different benchmarks, which are not always available, as shown in Table 1. For example, Dosso and Silvello (2020) described openly available benchmarks over three real datasets, LinkedMDB, IMDb, and a subset of DBpedia (Balog and Neumayer, 2013), and two synthetic databases, the Lehigh University Benchmark (LUBM) (Guo et al., 2005) and the Berlin SPARQL Benchmark (BSBM) (Bizer and Schultz, 2009). For IMDb, they defined 50 keyword queries and their correct translations to SPARQL queries. For DBpedia, the authors considered 50 topics from the classes `QALD2_te` and `QALD2_tr` of the *Question Answering over Linked Data* (QALD) campaigns (http://qald.aksw.org). For the synthetic databases, they used 14 SPARQL queries for LUBM and 13 SPARQL queries for BSBM. For all original `SELECT` queries from these datasets, Dosso and Silvello mapped these queries to SPARQL `CONSTRUCT` queries and produced their equivalent keyword queries.

3 DEFINITIONS

Recall that an *RDF dataset* is a set $\mathcal{T}$ of RDF triples (s,p,o). Furthermore, recall that $\mathcal{T}$ is equivalent to an edge-labeled direct graph $\mathcal{G}_{\mathcal{T}}$ such that the set of nodes of $\mathcal{G}_{\mathcal{T}}$ is the set of RDF terms that occur as subject or object of the triples in $\mathcal{T}$ and there is an edge (s,o) in $\mathcal{G}_{\mathcal{T}}$, labeled with p, iff the triple (s,p,o) occurs in $\mathcal{T}$. Figure 1.a shows an RDF graph.

The *resource graph* induced by a subset $\mathcal{T}' \subseteq \mathcal{T}$ is the subgraph $\mathcal{RG}_{\mathcal{T}'}$ of $\mathcal{G}_{\mathcal{T}'}$ obtained by dropping all literal nodes from $\mathcal{G}_{\mathcal{T}'}$. The nodes in $\mathcal{RG}_{\mathcal{T}'}$ are the *resources* of $\mathcal{T}'$. The *entity graph* induced by a subset $\mathcal{T}' \subseteq \mathcal{T}$ is the subgraph $\mathcal{EG}_{\mathcal{T}'}$ of $\mathcal{RG}_{\mathcal{T}'}$ obtained by dropping all class and property nodes from $\mathcal{RG}_{\mathcal{T}'}$. The nodes in $\mathcal{EG}_{\mathcal{T}'}$ are the *entities* of $\mathcal{T}'$.

A set of triples $\mathcal{T}' \subseteq \mathcal{T}$ *induces a path* $\pi = (s_0,p_0,s_1,p_1,s_2,\ldots,s_n,p_n,s_{n+1})$ in $\mathcal{G}_{\mathcal{T}}$ iff, for each $i \in [0,n]$, either $(s_i,p_i,s_{i+1}) \in \mathcal{T}'$ or $(s_{i+1},p_i,s_i) \in \mathcal{T}'$. We also say that (s_i,p_i,s_{i+1}) (or (s_{i+1},p_i,s_i)) *occurs* in π. Note that we assume that paths in $\mathcal{G}_{\mathcal{T}}$ may traverse arcs in both directions. A path $\pi = (s_0,p_0,s_1,p_1,s_2,\ldots,s_n,p_n,s_{n+1})$ *begins* (resp. *ends*) on a resource r iff $r = s_0$ or $r = p_0$ (resp. $r = s_{n+1}$ or $r = p_n$).

A *keyword query*, which represents an information need, is a set $\mathcal{K}$ of literals.

A triple $(s,p,o) \in \mathcal{T}$ is a *matching triple* for $\mathcal{K}$ iff o is a string literal that matches at least one keyword in $\mathcal{K}$. We also say that s is a *matching resource* for $\mathcal{K}$, the set $\mathcal{M}_s$ of all matching triples in $\mathcal{T}$ for $\mathcal{K}$ whose subject is s is the set of *matching triples* of s in $\mathcal{T}$, and the graph induced by $\mathcal{M}_s$ is the *matching subgraph* of s in $\mathcal{G}_{\mathcal{T}}$. Note that s may occur as the subject, property, or object of a triple in $\mathcal{T}$, but s is always the subject of

Table 1: Summary of the benchmarks used in some state-of-the-arts keyword search systems.

Tool	Ref.	Year	Description of Benchmark Used
SPARK	(Zhou et al., 2007)*	2007	Database and keyword queries from Mooney Natural Language Learning Data
QUICK	(Zenz et al., 2009)*	2009	An initial set of queries was extracted from a query log of the AOL search engine. Then, the queries were pruned based on the visited URLs, obtaining 3,000 sample keyword queries for IMDb and Lyrics Web pages. This process yielded 100 queries for IMDb, and 75 queries for Lyrics, consisting of 2–5 keywords.
	(Tran et al., 2009)*	2009	DBLP, TAP (http://tap.stanford.edu) and LUBM; 30 queries for DBLP, and 9 for TAP
	(Coffman and Weaver, 2010)†	2010	Samples of the Mondial, IMDb, and Wikipedia datasets; 50 queries for each dataset (not real user queries extracted from a search engine log).
	(Elbassuoni and Blanco, 2011)*	2011	Datasets derived from the LibraryThing community and IMDb; and 15 queries for each dataset.
	(Le et al., 2014)*	2014	Datasets: LUBM, Wordnet, BSBM, Barton and DBpedia Infobox. 12 Queries: 4 for LUBM, 2 for Wordnet, 2 for BSBM, 2 for Barton, 2 for DBpedia Infobox.
	(Zheng et al., 2016)	2016	DBpedia and Yago; queries derived from QALD-4
	(Han et al., 2017)	2017	DBpedia+QALD-6 and Freebase* + Free917: an open QA benchmark which consists of NL question and answer pairs over Freebase.
QUIOW	(Izquierdo et al., 2018)	2018	Full versions of the Mondial and IMDb datasets, and queries from Coffman's benchmark.
	(Lin et al., 2018)	2018	LUBM, Wordnet, BSBM, Barton and DBpedia Infobox; 4 queries for LUBM, and 10 queries for the other datasets.
KAT	(Wen et al., 2018)	2018	YAGO, DBLP and LUBM; 9 queries for YAGO, 3 queries for DBLP, and 6 queries for LUBM.
	(Rihany et al., 2018)	2018	AIFB and DBpedia; 10 queries for each dataset (the sizes of the queries were between 2 and 8 keywords).
QUIRA	(Menendez et al., 2019)	2019	Full versions of IMDb and MusicBrainz; 50 queries from Coffman's benchmark for IMDb, and 25 queries from QALD-2 for MusicBrainz. Details available at https://sites.google.com/view/quira/
TSA+BM25 and TSA+VDP	(Dosso and Silvello, 2020)	2020	**Real datasets:** LinkedMDB, IMDb, and a subset of DBPedia; 50 queries of Coffman's benchmark for each dataset. **Synthetic datasets:** LUBM and BSBM; with 14 and 13 queries, respectively.

* Datasets have no public link or are not available for download.
† Benchmark for evaluating keyword search systems over relational databases.

a matching triple.

A set of triples $\mathcal{A} \subseteq \mathcal{T}$ is an *answer for $\mathcal{K}$ over $\mathcal{T}$* iff $\mathcal{A}$ can be partitioned into two subsets $\mathcal{A}'$ and $\mathcal{A}''$ such that: (i) $\mathcal{A}'$ is the set of all matching triples for $\mathcal{K}$ in $\mathcal{A}$; let $\mathcal{R}_{\mathcal{A}}$ be the set of subjects of such triples; (ii) $\mathcal{RG}_{\mathcal{A}''}$, the resource graph induced by $\mathcal{A}''$, is connected and contains all resources in $\mathcal{R}_{\mathcal{A}}$. Condition (i) captures keyword matches and Condition (ii) indicates that an answer must connect the matching resources by paths in $\mathcal{RG}_{\mathcal{A}''}$.

We also say that $\mathcal{R}_{\mathcal{A}}$ is the set of *matching resources* of $\mathcal{A}$ and that $\mathcal{RG}_{\mathcal{A}''}$ is the *connectivity graph* of $\mathcal{A}$.

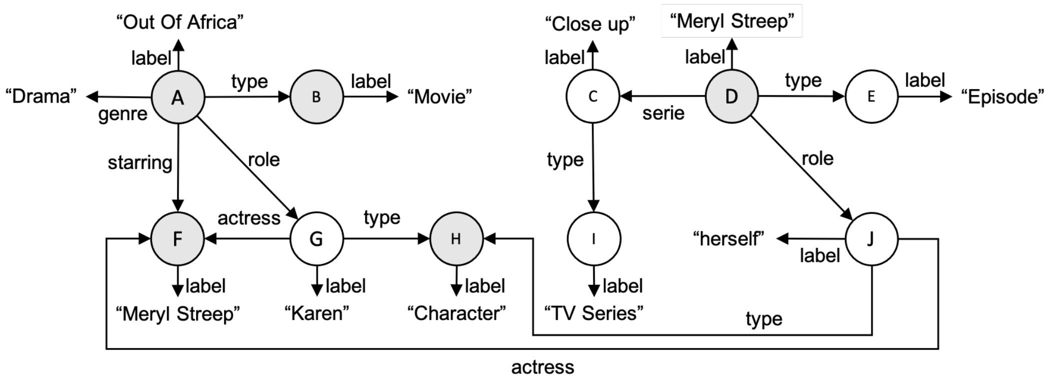
Figure 1 shows an RDF graph and three answers for the keyword query $\mathcal{MS}$ = {"character", "meryl", "streep", "movie", "out", "africa"}.

This definition of answer is less stringent than those introduced in (Bhalotia et al., 2002; Hristidis and Papakonstantinou, 2002; Kimelfeld and Sagiv, 2008) since it neither requires all keywords to be matched nor includes a minimality criterion. For later reference, we define that an answer $\mathcal{A}$ for $\mathcal{K}$ over $\mathcal{T}$ is *minimal* iff there is no proper subset $\mathcal{B}$ of $\mathcal{A}$ such that $\mathcal{B}$ is an answer for $\mathcal{K}$ and $\mathcal{B}$ and $\mathcal{A}$ have the same set of matching resources.

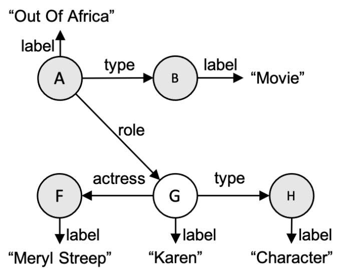
Finally, we can informally state the *RDF KwS-Problem* as: "Given an RDF dataset $\mathcal{T}$ and a keyword query $\mathcal{K}$, find an answer $\mathcal{A}$ for $\mathcal{K}$ over $\mathcal{T}$, preferably, with as many keyword matches as possible and with the smallest set of triples as possible".

4 OVERVIEW OF THE METHOD

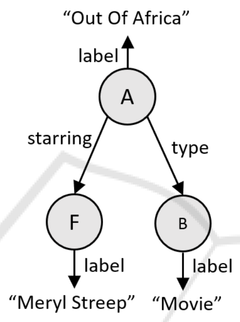
This section outlines the proposed method for the automatic construction of benchmarks for RDF datasets. Given an RDF dataset $\mathcal{T}$, the key problems are how to automatically generate sets of keyword queries over $\mathcal{T}$ and how to compute ranked lists of correct answers



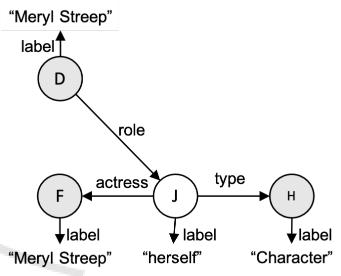
(a) The induced graph of an RDF dataset $\mathcal{T}$.



(b) Answer $\mathcal{A}_1$.



(c) Answer $\mathcal{A}_2$.



(d) Answer $\mathcal{A}_3$.

Figure 1: Example of an RDF graph and of three answers for the information need "*character of Meryl Streep in the movie Out of Africa*".

for these queries.

Let $\mathcal{T}$ be a dataset and $\mathcal{EG}_{\mathcal{T}}$ be the entity graph induced by $\mathcal{T}$.

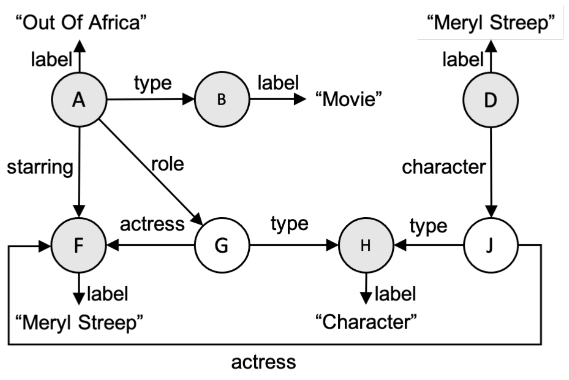
The method starts by computing keyword queries, that is, sets of keywords, based on a set I of *inducing entities* or *inducers*. First, it selects a set of inducers I (see Section 5 for examples) and computes a neighborhood graph $\mathcal{G}_{\mathcal{N}}$ from $\mathcal{EG}_{\mathcal{T}}$, for each $i \in I$. The neighborhood graph is composed of the nodes and edges visited by a breadth-first walk of distance d through the paths in $\mathcal{EG}_{\mathcal{T}}$, starting from i, where d is a user-defined parameter. After that, the method enriches $\mathcal{G}_{\mathcal{N}}$ with the nodes and edges of $\mathcal{G}_{\mathcal{T}}$ that denote the classes of the entities in $\mathcal{N}$. The enriched graph is denoted $\mathcal{G}_{\mathcal{N}'}$.

Then, it extracts from $\mathcal{T}$ keyword queries (again, sets of keywords) from the string literal values of datatype properties of resources and edges in $\mathcal{N}'$. Note that edges in $\mathcal{N}'$ can be resources in $\mathcal{T}$ with datatype properties. Since, by definition, answers are connected graphs, the neighborhood graph is an appropriate strategy to derive keywords.

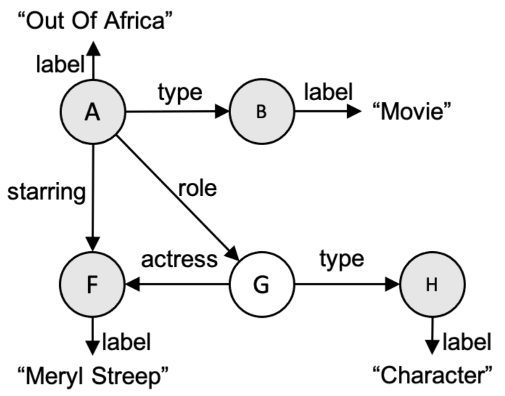
Let $\mathcal{K}$ be a keyword query thus generated. Note that, although $\mathcal{K}$ has been extracted for a particular inducer $i \in I$, $\mathcal{K}$ may also be present in other subgraphs of $\mathcal{T}$ that have no relationship with i, but that, according to the definition, can also be considered correct answers of $\mathcal{K}$. The inducers then have the only purpose of deriving keyword queries connected in at least $\mathcal{G}_{\mathcal{N}'}$. This approach allows one to compute as many keyword queries are needed to create a benchmark.

The next step is to compute possible answers for $\mathcal{K}$. The method computes the complete set of matching resources for $\mathcal{K}$, called the *set of seeds* for $\mathcal{K}$, denoted $\mathcal{S}_{\mathcal{K}}$. By the previous definitions, the set of matching resources of an answer $\mathcal{A}$ for $\mathcal{K}$ must be a subset of $\mathcal{S}_{\mathcal{K}}$. Preferably, answers should contain subsets of $\mathcal{S}_{\mathcal{K}}$ that match all keywords in $\mathcal{K}$.

Consider the following example. Let *"character of Meryl Streep in the movie Out of Africa"* be an information need over the dataset $\mathcal{T}$ of Figure 1.a. This information need can be translated to a keyword query $\mathcal{MS}$ = {"character", "meryl", "streep", "movie", "out", "africa"}. The set of seeds for $\mathcal{MS}$ is $\mathcal{S}_{\mathcal{MS}} = \{A, B, D, F, H\}$. Figures 1.b–d show the graphs induced by three possible answers for $\mathcal{MS}$ containing subsets of $\mathcal{S}_{\mathcal{MS}}$. Intuitively, answer $\mathcal{A}_1$ (Figure 1.b) is better than $\mathcal{A}_2$ (Figure 1.c) and $\mathcal{A}_3$ (Figure 1.d),



(a) The solution generator for $\mathcal{S}_{\mathcal{K}} = \{A,B,D,F,H\}$ from the dataset in Fig. 1a.



(b) The solution generator for $\mathcal{S}_{\mathcal{K}} = \{A,B,F,H\}$ from the dataset in Fig. 1a.

Figure 2: Examples of solution generators.

because $\mathcal{A}_1$ addresses what seems to be the query intention, which is to find the character played by the actress in the movie. Also, $\mathcal{A}_1$ matches 6 of the 6 keywords in $\mathcal{MS}$, while $\mathcal{A}_2$ and $\mathcal{A}_3$ match only 5 and 3 keywords, respectively. Note that, in $\mathcal{A}_1$ and $\mathcal{A}_2$, but not in $\mathcal{A}_3$, keywords are not matched by more than one literal. Also note that the keywords *"movie"* and *"character"* are associated with elements of the schema, nodes B and H.

This example illustrates two important characteristics of keyword queries and correct answers: *keyword queries name existing resources and answers correlate these resources*. In this example, "Meryl Streep", "Out of Africa", "character", and "movie" are informal resource identifiers that represent an actress, a movie, the `Character` class, and the `Movie` class, respectively.

Let $\mathcal{R} \subseteq \mathcal{S}_{\mathcal{K}}$ be a subset of seeds. Assume that all seeds in $\mathcal{R}$ belong to the same connected component of $\mathcal{RG}_{\mathcal{T}}$, the resources graph induced by $\mathcal{T}$. One can compute all possible answers whose matching resources are subsets of $\mathcal{R}$ by computing all acyclic paths in $\mathcal{RG}_{\mathcal{T}}$ between pairs of distinct nodes in $\mathcal{R}$, combining them in all distinct ways to construct connected graphs containing all nodes in $\mathcal{R}$, and adding the matching subgraphs for the seeds $r \in \mathcal{R}$.

Consider the set of seeds $\mathcal{R}'_{\mathcal{MS}} = \{A,B,F,H\}$, for example. If one selects the paths $(B \leftarrow A \rightarrow G \rightarrow F)$ and $(G \rightarrow H)$ and adds the matching subgraphs $(A \rightarrow$ "*Out of Africa*"), $(B \rightarrow$ "*Movie*"), $(F \rightarrow$ "*Meryl Streep*"), and $(H \rightarrow$ "*Character*"), one will obtain the answer in Figure 1.b. If one selects the path $(B \leftarrow A \rightarrow F \leftarrow G \rightarrow H)$ and adds the same matching subgraphs, one will obtain another valid answer (not shown in the figure).

Consider now the set of seeds $\mathcal{R}''_{\mathcal{MS}} = \{A,B,F\}$. If one selects the path $(B \leftarrow A \rightarrow F)$ and adds the matching subgraphs $(A \rightarrow$ "*Out of Africa*"), $(B \rightarrow$ "*Movie*"), and $(F \rightarrow$ "*Meryl Streep*"), one will obtain the answer in Figure 1.c. If one selects a different path $(B \leftarrow A \rightarrow G \rightarrow F)$, one will obtain yet another answer. In general, distinct path combinations may lead to distinct valid answers for the same set of seeds.

More precisely, let $\mathcal{K}$ again be a keyword query and $\mathcal{R} \subseteq \mathcal{S}_{\mathcal{K}}$. Assume that all seeds in $\mathcal{R}$ belong to the same connected component of $\mathcal{RG}_{\mathcal{T}}$. A set of triples $\mathcal{SG} \subseteq \mathcal{T}$ is a *solution generator* for $\mathcal{R}$ iff $\mathcal{SG}$ can be partitioned into two sets, $\mathcal{SG}'$ and $\mathcal{SG}''$, such that: (i) $\mathcal{SG}'$ is the set of all matching triples of the resources in $\mathcal{R}$; (ii) $\mathcal{SG}''$ is the set of all triples that occur in paths in $\mathcal{RG}_{\mathcal{T}}$ that begin and end on the seeds in $\mathcal{R}$. We say that $\mathcal{SG}$ *expresses* an answer $\mathcal{A}$ iff $\mathcal{A} \subseteq \mathcal{SG}$ and the set of matching resources of $\mathcal{A}$ is $\mathcal{R}$.

Figure 2 shows the solution generators for $\mathcal{R}_{\mathcal{MS}} = \{A,B,D,F,H\}$ and $\mathcal{R}'_{\mathcal{MS}} = \{A,B,F,H\}$. Note that, even though both $\mathcal{R}_{\mathcal{MS}}$ and $\mathcal{R}'_{\mathcal{MS}}$ cover all keywords, they are distinct and express distinct sets of answers.

A *synthetic benchmark* is then a triple $s = (\mathcal{T}, Q_s, A_s)$ such that $\mathcal{T}$ is an RDF dataset, Q_s is a list of keyword queries, and A_s contains, for each query in Q_s, a list of solution generators. Section 7 illustrates this concept.

However, computing all solution generators can be expensive, depending on the cardinality of $\mathcal{S}_{\mathcal{K}}$ and the number of paths between the seeds. We then propose to compute solution generators that capture only the most relevant answers. Fortunately, and unlike traditional Information Retrieval (IR) systems, the RDF KwS-Problem has some peculiarities that can be exploited to define optimization heuristics that can reduce the computational cost and also help rank solution generators.

The differences between traditional IR systems and

RDF-KwS systems stem from the fact that keywords which co-occur in a document may not convey the idea of keyword correlation. By contrast, an RDF subgraph connecting resources linked to these keywords is much more likely to be related to the intended meaning of the keyword query because resources are not connected by chance in an RDF graph, as it may be the case in a text document.

By assuming such differences, one can define some characteristics of good answers: the keywords must be connected as closely as possible, and answers must contain as many keywords as possible. Other features can be defined based on the number of resources in an answer and the co-occurrence of keywords among the literals. These features may guide the automatic computation of answers by helping prune the search space.

Section 6 discusses four heuristics to work around the complexity of the problem of computing solution generators, guided by five questions: (1) Are all seeds relevant?; (2) Are all paths between seeds relevant?; (3) Should we prefer answers that match more literals or answers that match fewer literals?; (4) Should we prefer answers in which literals in the keyword query occur in many seeds, or answers in which literals occur in only one seed?; (5) Should we prefer answers with many seeds or answers with fewer seeds?

5 GENERATING KEYWORD QUERIES

This section describes, with the help of examples, how to automatically generate sets of keyword queries for a given RDF dataset $\mathcal{T}$.

An *inducer function* for $\mathcal{T}$ is a function that maps $\mathcal{T}$ into a set of resources of $\mathcal{T}$, called *inducers*. Such function should follow requirements that are consistent with the benchmark's purpose, i.e, it should select entities from the information domain in question and with appropriate relevance scores.

The relevance scores typically express users' preferences and can be used to select entities and, consequently, induce relevant sets of keywords and their respective answers. They can also be used to challenge RDF-KwS systems by selecting less relevant resources and causing the opposite effect. For example, let $\mathcal{T}$ be the Mondial RDF dataset[1]. An inducers function for $\mathcal{T}$ would select the *top-k and bottom-k* countries according to their *infoRank* score, which is a metric defined in (Menendez et al., 2019) that reflects the relevance of the resources.

The next step is to compute the neighborhood graph for each inducer. Figure 3.a shows a fragment of the neighborhood graph $\mathcal{G}_{India'}$ for the Country *India*, which is the 8th country with the largest *infoRank* score in the Mondial dataset. The grayed nodes are the classes of the entities and the symbol "..." close to the edges indicates that the corresponding property is multivalued. For the sake of conciseness, some paths of length 2 starting from India were omitted.

The 43rd query of Coffman's benchmark for Mondial, $\mathcal{K}_{Coff\text{-}43} = \{\text{“}mauritius\text{”}, \text{“}india\text{”}\}$, can be created by extrating the keywords from the datatype property *name* of the entities *India* and *Mauritius* in $\mathcal{G}_{India'}$. Figure 3.b shows the ground truth for $\mathcal{K}_{Coff\text{-}43}$ in the aforementioned benchmark.

The query generation process then consists in creating sets of keywords from $\mathcal{G}_{India'}$ and $\mathcal{T}$. Let $\mathcal{G}_{India''}$ be the maximally subgraph of $\mathcal{T}$ containing all resources and properties in $India'$. Note that $\mathcal{G}_{India''}$ is equal to $\mathcal{G}_{India'}$ with additional property and literal nodes.

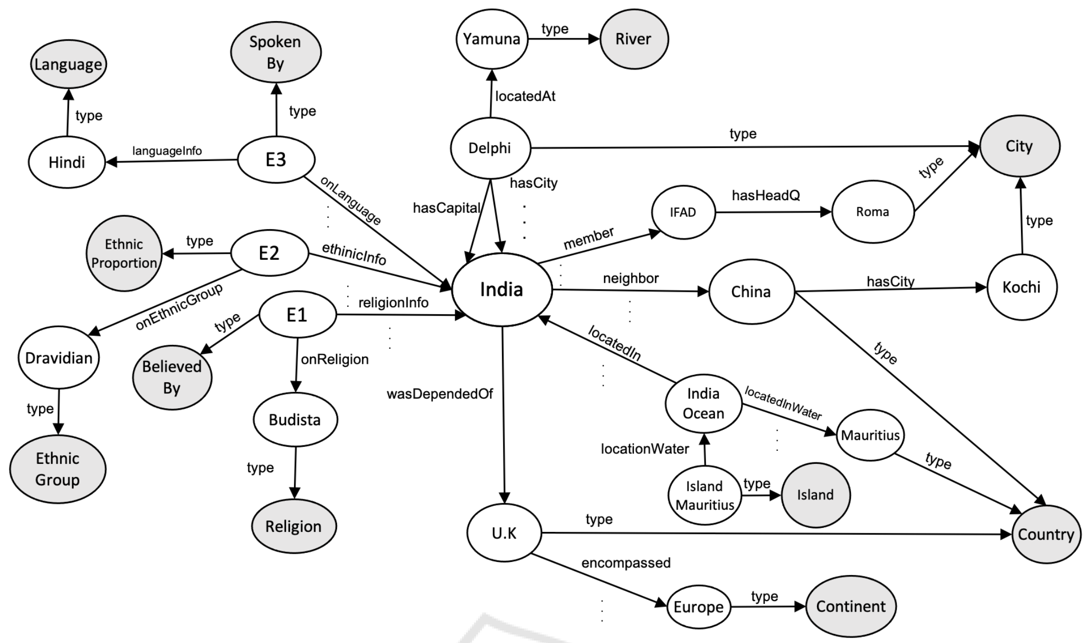
Let $\textit{TF-IDF}(k)$ be the term-frequency-inverse-document-frequency score for each non stop-word in the labels of literal nodes in $\mathcal{G}_{India''}$. One can then define sets of rules for generating keywords such as: (1) extract the top-k words with largest *TF-IDF* from literals of class resources; (2) extract the top-k words with largest *TF-IDF* from literals of property resources; (3) extract the top-k words with largest *TF-IDF* from literals of entity resources; (4) extract the top-k words with largest *TF-IDF* from literals of mixed resources. Note that *TF-IDF* can be replaced by any other convenient score, such as *BM25*, that the set of rules must be defined *ad hoc*, and that the relevance score *TF-IDF* can also be used likewise the *infoRank* to challenge the RDF-KwS systems by allowing one to select less relevant words.

6 COMPUTING SOLUTION GENERATORS

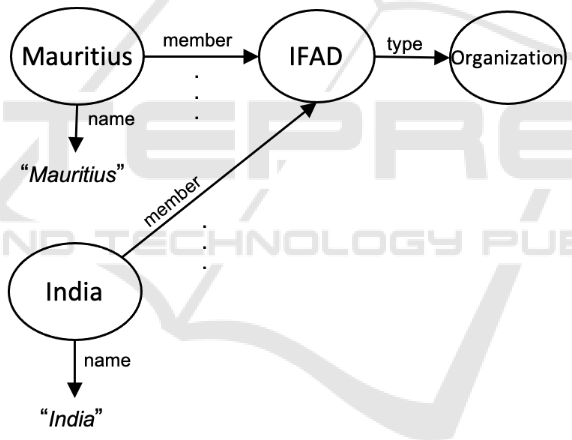
This section addresses the computation of solution generators for keyword queries, through four heuristics to circumvent the complexity of the problem.

The first heuristic refers to the selection of the most relevant seeds. Assume that Lucene for Apache Jena Fuseki is the text search engine over RDF adopted. The *Lucene score* is a TF-IDF-based score that captures the relevance of property values with respect to the keywords. The full set of seeds of a keyword query can be obtained from the Lucene inverted index, which can be a large set. One could then limit this set to the top-*k* resources according to the Lucene score, but the

[1]http://www.dbis.informatik.uni-goettingen.de/Mondial/



(a) Neighborhood graph of the entity denoting the country India.



(b) A correct answer for the 43rd query $\mathcal{K}_{Coff\text{-}43} = \{\text{"}mauritius\text{"}, \text{"}india\text{"}\}$ in the Coffman's benchmark.

Figure 3: Example neighborhood, keyword query and a possible answer.

resource labeled "Meryl Streep" would be the 7th entry in the ranked list of seeds and the resource labeled "Out of Africa" would be the 30th entry. If one took the top 30 resources in the ranking, the two seeds would be selected, but many other less relevant seeds would also be selected.

Let $\mathcal{K}$ be keyword query and $\mathcal{S}_{\mathcal{K}}$ be the set of seeds of $\mathcal{K}$. We define the *entity score* of a seed $s \in \mathcal{S}_{\mathcal{K}}$ with respect to $\mathcal{K}$ as:

$$es(s,\mathcal{K}) = \frac{1}{2}(max_{v_j}(lucene(s,v_j,\mathcal{K}))+ infoRank(s)) \quad (1)$$

where v_j is a string value such that there is a triple $(s,p,v_j) \in \mathcal{T}$. The *infoRank* score (Menendez et al., 2019) reflects the relevance of s to users. The entity score ranges in $[0,1]$, since we used normalized versions of $lucene(s,v_j,\mathcal{K})$ and $infoRank(s)$. If a text search engine other than Lucene is adopted or a resource relevance measure other than *infoRank* is used, Eq. 1 should be adjusted accordingly.

By ranking the set of seeds of $\mathcal{K}$ according to the entity score, the resource labeled "Meryl Streep" would appear in the 1st position and that labeled with "Out of Africa" would appear in the 18th position.

The *first heuristic* is then to refine the set of seeds $\mathcal{S}_\mathcal{K}$ to be the set $\Sigma_\mathcal{K}$ of the top-σ_2 elements of $\{s \in \mathcal{S}_\mathcal{K} | es(s, \mathcal{K}) \geq \sigma_1\}$ ordered in decreasing order of entity score, where σ_1 and σ_2 are thresholds empirically defined to optimize computing resources and match user's preferences.

The threshold σ_1 defines a minimum seed entity score to speed up the Lucene engine, for practical reasons, and σ_2 effectively limits the number of selected seeds. By defining $\sigma_1 = 0$ and $\sigma_2 = \infty$, one would select the full set of seeds. But, by defining $\sigma_1 > 0$ and $\sigma_2 < \infty$ one can restrict the cardinality of $2^{\Sigma_\mathcal{K}}$ and, consequently, the total number of paths to compute.

However, $\Sigma_\mathcal{K}$ may not cover all keywords in $\mathcal{K}$. We then compute another set $\mathcal{S}_{\mathcal{K}_i}$ of matching resources, where $\mathcal{K}_i$ is the subset of $\mathcal{K}$ not matched by resources in $\Sigma_\mathcal{K}$, and repeat this procedure to extend $\Sigma_\mathcal{K}$ until no further keyword can be matched or $\mathcal{K}_i = \{\}$. More precisely, for any $\mathcal{K}_i \subseteq \mathcal{K}$, let $\mathcal{S}_{\mathcal{K}_i}$ be the set of seeds in $\mathcal{S}_\mathcal{K}$ that match keywords in $\mathcal{K}_i$. Let $\mu[\sigma_1](\mathcal{K}_i, \mathcal{S}_{\mathcal{K}_i}) = \{s \in \mathcal{S}_{\mathcal{K}_i} | es(s, \mathcal{K}_i) \geq \sigma_1\}$, and $\tau[\sigma_1, \sigma_2](\mathcal{K}_i, \mathcal{S}_{\mathcal{K}_i})$ be the top-σ_2 elements of $\mu[\sigma_1](\mathcal{K}_i, \mathcal{S}_{\mathcal{K}_i})$, ordered by the entity score of the seeds.

Let $(\mathcal{K}_0, ..., \mathcal{K}_N)$ be the longest sequence such that $\mathcal{K}_0 = \mathcal{K}$ and, for each $i \in [1, N]$, $\mathcal{K}_i$ is the set of keywords in $\mathcal{K}$ not matched by seeds in $\tau[\sigma_1, \sigma_2](\mathcal{K}_{i-1}, \mathcal{S}_{\mathcal{K}_{i-1}})$ and $\mathcal{K}_i \neq \emptyset$ and $\mathcal{S}_{\mathcal{K}_i} \neq \emptyset$. Then, $\Sigma_\mathcal{K}$ is defined as

$$\Sigma_\mathcal{K} = \bigcup_{i=0}^{N} \tau[\sigma_1, \sigma_2](\mathcal{K}_i, \mathcal{S}_{\mathcal{K}_i}) \tag{2}$$

The *second heuristic* refers to the selection of sets in $2^{\Sigma_\mathcal{K}}$ for which one would compute the solution generators. This is done by scoring each $\mathcal{R} \in 2^{\mathcal{S}_\mathcal{K}}$ and selecting the top ranked ones based on four principles. First, the set of matching keywords of $\mathcal{R}$ is the union of the set of keywords matched by each resource in $\mathcal{R}$; the sets $\mathcal{R}$ with largest number of keyword matchings are preferable and potentially would generate better answers. Second, the keywords should preferably match just a few seeds of an answer, because keywords identify entities. We assume that answers where keywords do not match more than one resource, such as those in Figures 1.b and 1.c, are more relevant. Nevertheless, as detailed later in this section, this constraint can be relaxed to allow answers such as that in Figure 1d. Third, smaller answers, in terms of the number of seeds, are preferable over larger ones, since they are easier to understand. Lastly, not only small sets are preferable, but it is also necessary that their resources are the most relevant to users. Based on these principles, we define the following scores.

Let $\mathcal{E}$ be a set of resources. The *coverage score* of $\mathcal{E}$, denoted $cs(\mathcal{E}, \mathcal{K})$, measures the number of keywords in $\mathcal{K}$ matched by resources in $\mathcal{E}$; $cs(\mathcal{E}, \mathcal{K}) = 1$, if all keywords are matched, and $cs(\mathcal{E}, \mathcal{K}) = 0$, if no keyword is matched:

$$cs(\mathcal{E}, \mathcal{K}) = \frac{\sum_{k_i \in \mathcal{K}} occur(\mathcal{E}, k_i)}{|\mathcal{K}|} \tag{3}$$

where $occur(\mathcal{E}, k_i) = 1$, if some resource in $\mathcal{E}$ matches k_i, and $occur(\mathcal{E}, k_i) = 0$, otherwise.

The *co-occurrence score* of $\mathcal{E}$, denoted $os(\mathcal{E}, \mathcal{K})$, measures the repetition degree of keywords in $\mathcal{K}$ among resources in $\mathcal{E}$; $os(\mathcal{E}, \mathcal{K}) = 1$, if each keyword is matched by only one resource, and $os(\mathcal{E}) = 0$, if all keywords are matched by all resources:

$$os(\mathcal{E}, \mathcal{K}) = \frac{1 - \frac{f(\mathcal{E}, \mathcal{K})}{|\mathcal{E}|}}{1 - \frac{1}{|\mathcal{E}|}} \tag{4}$$

where $f(\mathcal{E}, \mathcal{K})$ is the average number of resources in $\mathcal{E}$ that match keywords in $\mathcal{K}$. Keywords not covered in $\mathcal{E}$ are not taken into account; $os(\mathcal{E}, \mathcal{K})$ is assumed to be 1, if the denominator is 0.

Let $\mathcal{C}$ be the collection of sets of resources considered. The *size score* of $\mathcal{E}$ w.r.t. $\mathcal{C}$, denoted $ss(\mathcal{E})$, measures the relative size of $\mathcal{E}$; $ss(\mathcal{E}) = 1$, if $\mathcal{E}$ is one of the smallest sets, and $ss(\mathcal{E}) = 0$, if $\mathcal{E}$ is one of the largest sets:

$$ss(\mathcal{E}) = \frac{\mathcal{N} - |\mathcal{E}|}{\mathcal{N} - 1} \tag{5}$$

where $\mathcal{N}$ is the cardinality of the largest set in $\mathcal{C}$; $ss(\mathcal{E}, \mathcal{K})$ is assumed to be 1, if the denominator is 0.

The *infoRank score* of $\mathcal{E}$, denoted $is(\mathcal{E})$, is the average infoRank value (Menendez et al., 2019) of the resources in $\mathcal{E}$:

$$is(\mathcal{E}) = average(\{infoRank(s_i) | s_i \in \mathcal{E}\}) \tag{6}$$

The second heuristic is then the refinement of the set $2^{\Sigma_\mathcal{K}}$ by choosing only those $\mathcal{R} \in 2^{\Sigma_\mathcal{K}}$ with better coverage, lower co-occurrence, fewer number of resources, and with more relevant nodes. Recall that a lower co-occurrence would favor answers such as those in Figures 1.b and 1.c, while allowing answers such as that in Figure 1.d. On the other hand, no co-occurrence ($os(\mathcal{R}, \mathcal{K}) = 1$) would allow answers such as those in Figures 1.b and 1.c only. The refinement is expressed by defining set Π_K as follows:

$$\Pi_K = \{\mathcal{R} \in 2^{\Sigma_\mathcal{K}} | cs(\mathcal{R}, \mathcal{K}) \geq \sigma_3 \wedge os(\mathcal{R}, \mathcal{K}) \geq \sigma_4 \wedge ss(\mathcal{R}) \geq \sigma_5 \wedge is(\mathcal{R}) \geq \sigma_6\} \tag{7}$$

where σ_3, σ_4, σ_5, and σ_6 are empirically defined according to the available computing resources and user preferences. If $\sigma_3 = \sigma_4 = \sigma_5 = \sigma_6 = 0$ then $\Pi_K = 2^{\Sigma_\mathcal{K}}$ and all possible solution generators with $\Sigma_\mathcal{K}$ would be computed. The above scores can be redefined for subsets of triples $\mathcal{U} \subseteq \mathcal{T}$ by taking $\mathcal{E} = \mathcal{E}_{\mathcal{U}}$, where $\mathcal{E}_{\mathcal{U}}$ is the set of all resources in $\mathcal{U}$.

The *third heuristic* is to consider only paths between seeds with length less than or equal to a given limit L, say $L = 4$, to compute solution generators. In fact, as argued in Nunes et al. (2014), paths longer than 4 would express unusual relationships, which might be misinterpreted by users.

The *fourth heuristic* is to rank the solution generators in Π_K according to their scores, following user needs. For example, if one ranks based only on the coverage and size then one may define a Boolean function "order" between solution generators as follows

$$order(\mathcal{SG}_1, \mathcal{SG}_2) = \begin{cases} cs(\mathcal{SG}_1, \mathcal{K}) \geq cs(\mathcal{SG}_2, \mathcal{K}), \text{ if } C \text{ holds} \\ ss(\mathcal{SG}_1, \mathcal{K}) \geq ss(\mathcal{SG}_2), \text{ otherwise} \end{cases} \tag{8}$$

where C is the condition $cs(\mathcal{SG}_1, \mathcal{K}) \neq cs(\mathcal{SG}_2, \mathcal{K})$. Alternatively, one could rank solution generators by their average scores as in Eq. 9 to balance the losses and gains of each individual score:

$$\begin{aligned} order(\mathcal{SG}_1, \mathcal{SG}_2) = \frac{1}{4}(cs(\mathcal{SG}_2, \mathcal{K}) + \\ os(\mathcal{SG}_2, \mathcal{K}) + ss(\mathcal{SG}_2) + is(\mathcal{SG}_2)) \end{aligned} \tag{9}$$

Eqs. 2, 7, 8, and 9 are in fact flexibilization points of the method. For example, instead of using the Lucene score in Eq. 1, one could use a keyword count, and instead of a selection in Eq. 7, one could use the top of the ranking of the sets $\mathcal{R} \in 2^{\Sigma_{\mathcal{MS}}}$ with the order function defined in Eq. 9. The method chosen will determine the set of solution generators that will be computed. The central contribution of this work lies, then, in using the peculiarities of RDF keyword search to define a process for optimizing the computation of solution generators.

Algorithm 1 embeds the proposed heuristics to reduce the cost of computing solution generators by disregarding the less relevant ones. It takes as input a keyword query $\mathcal{K}$ and an RDF dataset $\mathcal{T}$, and outputs a ranked list of solution generators according to the score function in Eq. 8. Lines 1 and 2 prepare the sets of nodes that will guide the computation of the solution generators according to Eqs. 2 and 7. Lines 4–12 compute solution generators for each set of seeds in Π_K. In lines 9–11, if the set of triples $\mathcal{SG}$ computed for a set of seeds $\mathcal{R} \in \Pi_K$ does not induce a connected graph $\mathcal{G}'_{\mathcal{SG}}$ (connectivity graph of $\mathcal{SG}$), then $\mathcal{SG}$ is discarded because, for each connected component $\mathcal{C}_i$ of $\mathcal{G}'_{\mathcal{SG}}$, there is a strict subset $\mathcal{R}_j$ of $\mathcal{R}$ such that the set of triples computed for $\mathcal{R}_j$ induces $\mathcal{C}_i$.

Algorithm 1: Computes a ranked list of solution generators for a keyword query $\mathcal{K}$ over an RDF dataset $\mathcal{T}$.

```
Require: a keyword query $\mathcal{K}$ and an RDF dataset $\mathcal{T}$
 1: $\Sigma_{\mathcal{K}} = \bigcup_{i=0}^{N} \tau[\sigma_1, \sigma_2](\mathcal{K}_i, \mathcal{S}_{\mathcal{K}_i})$
 2: $\Pi_K = \{\mathcal{R} \in 2^{\mathcal{S}_{\mathcal{K}}} | cs(\mathcal{R}, \mathcal{K}) \geq \sigma_3 \wedge os(\mathcal{R}, \mathcal{K}) \geq$
    $\sigma_4 \wedge ss(\mathcal{R}) \geq \sigma_5 \wedge is(\mathcal{R}, \mathcal{K}) \geq \sigma_6\}$
 3: $\mathcal{U} = \{\}$
 4: for all $\mathcal{R} \in \Pi_K$ do
 5:     $\mathcal{SG} = \{\}$
 6:     for all distinct unordered pairs of nodes $\{n_1,$
        $n_2\}$ such that $n_1, n_2 \in \Sigma_{\mathcal{K}}$ do
 7:         $\mathcal{SG} = \mathcal{SG} \cup \{(s,p,o) \in \mathcal{T} | (s,p,o)$ is in a
            path with length $\leq 4$ between $n_1$ and $n_2\}$
 8:     end for
 9:     if $\mathcal{G}'_{\mathcal{SG}}$ is connected then
10:         $\mathcal{U} = \mathcal{U} \cup \{\mathcal{SG}\}$
11:     end if
12: end for
13: Create $\mathcal{U}'$ by ordering $\mathcal{U}$ using the score function
    in Eq. 8
14: return $\mathcal{U}'$
```

7 EVALUATION OF THE BENCHMARK GENERATION METHOD

This section addresses the key question of evaluating the proposed benchmark generation method. The evaluation concentrates on assessing the quality of the solution generators. An evaluation of the strategy to generate keyword queries is omitted due to space limitations.

The evaluation strategy goes as follows. Let $b = (\mathcal{D}, Q_b, A_b)$ be a *baseline benchmark*, where $\mathcal{D}$ is an RDF dataset, Q_b is a set of keyword queries, and A_b defines the correct answers for the queries in Q_b. The strategy is to construct a synthetic benchmark $s = (\mathcal{D}, Q_s, A_s)$ for the same dataset $\mathcal{D}$, using the proposed method, where $Q_b \cap Q_s \neq \emptyset$ and A_s contains, for each keyword query in Q_s, a list of solution generators over $\mathcal{D}$, synthesized using an implementation of Algorithm 1. Then, for each keyword query in $Q_b \cap Q_s$, we compare the answers in A_b with the solution generators in A_s, as explained in what follows, and this is the key point of the evaluation. Note that we have to guarantee that $Q_b \cap Q_s \neq \emptyset$ as otherwise the benchmark comparison would have a vacuous effect. In fact, rather than using the keyword query generation method discussion in Section 5, we manually selected keyword queries from the baseline benchmarks to include in the synthetic benchmarks, as discussed in what follows.

As baselines, we adopted a benchmark for RDF-KwS based on Coffman's benchmark (Coffman and Weaver, 2010) and the Dosso's benchmarks described in (Dosso and Silvello, 2020). Coffman's benchmark was created to evaluate keyword search systems over relational databases, and is based on data and schemas of relational samples of IMDb, Mondial, and Wikipedia. For each database, it has 50 keyword queries and their correct answers. Dosso and Silvello (2020) used three real RDF datasets, LinkedMDB, IMDb, and DBpedia, and two synthetic RDF datasets, The Lehigh University Benchmark – LUBM, and The Berlin SPARQL Benchmark – BSBM.

As for the datasets, we chose triplifications of relational versions of the full Mondial and IMDb datasets, and not just samples as in Coffman's benchmark, and the RDF datasets BSBM, LUBM, and DBpedia from Dosso's benchmark. For each such RDF dataset, we computed the *infoRank* scores.

We selected the keyword queries of the synthetic benchmarks as follows. Quite a few keyword queries in Coffman's benchmark are *simple queries* in that their expected answers are single entities. Since these queries do not explore the complexity of the graph structure of the RDF datasets, we disregarded them for IMDb and Mondial. We used the keyword queries for BSBM, LUBM, and DBpedia as defined in Dosso's benchmark. In total, we used 35 keyword queries for IMDb and 24 for Mondial, from Coffman's benchmark, and 50 keyword queries for DBpedia, 14 for LUBM, and 13 for BSBM, from Dosso's benchmark.

Finally, we ran an implementation of Algorithm 1 to obtain a list of solution generators, for each of the selected keyword queries. The parameters described in Section 6 were set as $\sigma_1 = 0, \sigma_2 = 5, \sigma_3 = 1.0, \sigma_4 = 0.5, \sigma_5 = 0.2, \sigma_6 = 0.2$, for all datasets.

The above process resulted in 5 synthetic benchmarks, for IMDb, Mondial, BSBM, LUBM, and DBpedia. The RDF datasets are available at https://doi.org/10.6084/m9.figshare.11347676.v3, and the implementation of Algorithm 1, the keyword queries, the solution generators, and statistics are available at https://doi.org/10.6084/m9.figshare.9943655.v12.

For each of the five RDF datasets, we now compare the baseline benchmark with the corresponding synthetic benchmark. Consider the following questions (where $\mathcal{K}$ is a keyword query of both the baseline benchmark and the corresponding synthetic benchmark, as explained above):

Q1. What is the total number $s_{\mathcal{K}}$ of answers expressed by the solution generators for $\mathcal{K}$ in the synthetic benchmark?

Q2. What is the total number $bs_{\mathcal{K}}$ of answers of $\mathcal{K}$, defined in the baseline benchmark, that are expressed by the solution generators for $\mathcal{K}$ in the synthetic benchmark?

Q3. What is the total number $sn_{\mathcal{K}}$ of answers expressed by the solution generators for $\mathcal{K}$ in the synthetic benchmark that are not defined in the baseline benchmark?

Let $B_{\mathcal{K}}$ be the set of answers for $\mathcal{K}$ defined in the baseline benchmark and $b_{\mathcal{K}} = |B_{\mathcal{K}}|$.

To address these questions, one has to compute the set $S_{\mathcal{K}}$ of answers for $\mathcal{K}$ that the solution generators for $\mathcal{K}$ express. It depends on the exact notion of answer one is adopting. For example, the set of minimal answers can be estimated by counting the minimal Steiner Trees (Oliveira et al., 2020; Dourado and de Oliveira, 2009) of a solution generator $\mathcal{SG}$ whose terminal nodes are the set of seeds of $\mathcal{SG}$. To compute $|B_{\mathcal{K}} \cap S_{\mathcal{K}}|$, one has to test, for each answer $\mathcal{A} \in B_{\mathcal{K}}$, if there is some solution generator for $\mathcal{K}$ that expresses $\mathcal{A}$. Hence, we have that

- $s_{\mathcal{K}} = |S_{\mathcal{K}}|$
- $bs_{\mathcal{K}} = |B_{\mathcal{K}} \cap S_{\mathcal{K}}|$
- $sn_{\mathcal{K}} = |S_{\mathcal{K}} - B_{\mathcal{K}}| = |S_{\mathcal{K}}| - |B_{\mathcal{K}} \cap S_{\mathcal{K}}|$

Column **#$\mathcal{M}$s** of Table 2 shows the total number of minimal answers expressed by the solution generators for $\mathcal{K}$ that are not defined in the original benchmarks.

Q1 and Q2 lead to an interesting discussion. Consider that the baseline benchmarks are keyword search systems to be evaluated against the synthetic benchmarks. Then, one can compute the precision of the baseline benchmark for $\mathcal{K}$ against the equivalent synthetic benchmark as $p_{\mathcal{K}} = bs_{\mathcal{K}}/b_{\mathcal{K}}$. The larger $p_{\mathcal{K}}$ is, the larger will be the number of correct answers for $\mathcal{K}$, in the baseline benchmark, that the solution generators express. Likewise, one can compute the recall of the baseline benchmark for $\mathcal{K}$ against the equivalent synthetic benchmark as $r_{\mathcal{K}} = bs_{\mathcal{K}}/s_{\mathcal{K}}$.

Table 2 summarizes statistics for Mondial, IMDb, and DBpedia. For sample keyword queries (to save space), it shows the number of retrieved seeds, the precision values that the baseline benchmarks achieved, the number of solution generators obtained from the seeds, and the number of minimal answers expressed by the solution generators that are not defined in the baseline benchmarks. For example, for the keyword query $\mathcal{K} = \{$"*niger*","*country*"$\}$ from Mondial, the algorithm selected four seeds: the `country Niger`, the `province Niger`, the `river Niger`, and the class `Country`. Then, it computed four solution generators: 1) with all seeds; 2) with all seeds, except the class `Country`; 3) with two seeds, the class `Country` and the node `Niger`, which is an instance of class `Country`; and 4) with only the class `Country`.

Table 2: Benchmarks statistics obtained for Mondial, IMDb, and DBpedia.

Datasets	Sample Keyword Queries	#Seeds	Precision	#Sol. Generators	#$\mathcal{M}$s
Mondial	niger country	4	1.00	4	23
	haiti religion	2	1.00	1	2
	mongolia china	4	1.00	2	3
	lebanon syria	5	1.00	3	10
	poland cape verde organization	5	0.82	4	132
	rhein germany province	5	0.50	2	82
	OVERALL AVERAGE		**0.91**	**5.00**	**184.83**
IMDb	Johnny Depp Actor	5	1.00	12	46
	Will Smith Male	5	1.00	6	21
	Atticus Finch Movie	5	1.00	10	35
	russell crowe gladiator character	5	0.50	11	21
	sean connery ian fleming work	5	0.11	10	27
	OVERALL AVERAGE		**0.52**	**5.51**	**38.60**
DBpedia	Captain America creator notable works	5	1.00	5	47
	Canada Capital	5	1.00	3	57
	governor of Texas	5	1.00	5	58
	Francis Ford Coppola film director	5	1.00	11	61
	mayor of new york city	5	0.00	2	9
	NASA launchpad	5	0.00	3	5
	OVERALL AVERAGE		**0.58**	**8.22**	**91.86**

The Overall Averages can be interpreted as the percentage of the correct answers, defined in the baseline benchmarks, that the solution generators express - 91% for Mondial, 52% for IMDb, and 58% for DBpedia - which is quite reasonable for synthetic benchmarks.

Finally, we remark that, for the synthetic datasets (BSBM and LUBM), Algorithm 1 found exactly the correct answers listed in Dosso's benchmark.

8 CONCLUSIONS

One of the main issues in the development of RDF keyword search algorithms is the lack of appropriate benchmarks. This paper then proposed an offline method to construct benchmarks for RDF keyword search algorithms. The method automatically specifies sets of keyword queries and their correct answers over a given RDF dataset. It circumvents the combinatorial nature of generating correct answers by pruning the search space, following four heuristics based on the concepts of seeds and solution generators. The proposed heuristics introduce flexibilization points of the method that enable the construction of different benchmarks according to the purpose.

The paper proceeded to describe synthetic benchmarks for IMDb, Mondial, BSBM, LUBM, and DBpedia. The experiments compared the synthetic benchmarks with baseline benchmarks, and showed that the solution generators obtained express the majority of the correct answers defined in the baseline benchmarks, but express many more answers, for the datasets with real data, IMDb, Mondial, and DBpedia, and express exactly the answers defined for the synthetic datasets, BSBM and LUBM.

As future work, we plan to fine-tune Algorithm 1 to improve the results summarized in Table 2. We also plan to modify the proposed method to construct training datasets with Natural Language queries over complex RDF datasets.

ACKNOWLEDGEMENTS

This work was partly funded by FAPERJ under grants E-26/010.000794/2016 and E-26/202.818/2017; by CAPES under grant 88881.134081/2016-01 and 88882.164913/2010-01 and by CNPq under grant 302303/2017-0. We are grateful to João Guilherme Alves Martinez for helping with the experiments.

REFERENCES

Balog, K. and Neumayer, R. (2013). A Test Collection for Entity Search in DBpedia. In *Proceedings of the 36th International ACM SIGIR Conference*, pages 737–740.

Bast, H., Buchhold, B., and Haussmann, E. (2016). Semantic search on text and knowledge bases. *Foundations and Trends® in Information Retrieval*, 10(2-3):119–271.

Bhalotia, G., Hulgeri, A., Nakhe, C., Chakrabarti, S., and Sudarshan, S. (2002). Keyword searching and browsing in databases using BANKS. In *Proceedings of the 18th International Conference on Data Engineering (ICDE'02)*, pages 431–440.

Bizer, C. and Schultz, A. (2009). The berlin sparql benchmark. *International Journal on Semantic Web and Information Systems (IJSWIS)*, 5(2):1–24.

Coffman, J. and Weaver, A. C. (2010). A framework for evaluating database keyword search strategies. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*, pages 729–738.

Dosso, D. and Silvello, G. (2020). Search Text to Retrieve Graphs: a Scalable RDF Keyword-Based Search System. *IEEE Access*, 8:14089–14111.

Dourado, M. C. and de Oliveira, R. A. (2009). Generating all the Steiner trees and computing Steiner intervals for a fixed number of terminals. *Electronic Notes in Discrete Mathematics*, 35:323–328.

Dubey, M., Banerjee, D., Abdelkawi, A., and Lehmann, J. (2019). LC-QuAD 2.0: A Large Dataset for Complex Question Answering over Wikidata and DBpedia. In *Proceedings of the 18th International Semantic Web Conference (ISWC'19)*, pages 69–78.

Elbassuoni, S. and Blanco, R. (2011). Keyword search over RDF graphs. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management (CIKM'11)*, pages 237–242.

García, G. M., Izquierdo, Y. T., Menendez, E. S., Dartayre, F., and Casanova, M. A. (2017). RDF Keyword-based Query Technology Meets a Real-World Dataset. In *Proceedings of the 20th International Conference on Database Theory (ICDT'17)*, pages 656–667.

Guo, Y., Pan, Z., and Heflin, J. (2005). LUBM: A Benchmark for OWL Knowledge Base Systems. *Journal of Web Semantics*, 3(2-3):158–182.

Han, S., Zou, L., Yu, J. X., and Zhao, D. (2017). Keyword search on RDF graphs - A query graph assembly approach. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge (CIKM'17)*, pages 227–236.

Hristidis, V. and Papakonstantinou, Y. (2002). DISCOVER: Keyword Search in Relational Databases. In *Proceedings of the 28th VLDB (VLDB'02)*, pages 670–681.

Izquierdo, Y. T., García, G. M., Menendez, E. S., Casanova, M. A., Dartayre, F., and Levy, C. H. (2018). QUIOW: A keyword-based query processing tool for RDF datasets and relational databases. In *Proceedings of the 30th International Conference on Database and Expert Systems Applications (DEXA'18)*, volume 11030 LNCS, pages 259–269.

Kimelfeld, B. and Sagiv, Y. (2008). Efficiently enumerating results of keyword search over data graphs. *Information Systems*, 33(4-5):335–359.

Le, W., Li, F., Kementsietsidis, A., and Duan, S. (2014). Scalable keyword search on large RDF data. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 26(11):2774–2788.

Lin, X. Q., Ma, Z. M., and Yan, L. (2018). RDF keyword search using a type-based summary. *Journal of Information Science and Engineering*, 34(2):489–504.

Menendez, E. S., Casanova, M. A., Paes Leme, L. A. P., and Boughanem, M. (2019). Novel Node Importance Measures to Improve Keyword Search over RDF Graphs. In *Proceedings of the 31st International Conference on Database and Expert Systems Applications (DEXA'19)*, volume 11707, pages 143–158.

Nunes, B. P., Herrera, J., Taibi, D., Lopes, G. R., Casanova, M. A., and Dietze, S. (2014). SCS Connector - Quantifying and Visualising Semantic Paths Between Entity Pairs. In *Proceedings of the Satellite Events of the 11th European Semantic Web Conference (ESWC'14)*, pages 461–466.

Oliveira, P. S. d., Da Silva, A., Moura, E., and De Freitas, R. (2020). Efficient Match-Based Candidate Network Generation for Keyword Queries over Relational Databases. *IEEE Transactions on Knowledge and Data Engineering*, pages 1–1.

Oliveira Filho, A. d. C. (2018). Benchmark para métodos de consultas por palavras-chave a bancos de dados relacionais. Technical report.

Rihany, M., Kedad, Z., and Lopes, S. (2018). Keyword search over RDF graphs using wordnet. In *Proceedings of the 1st International Conference on Big Data and Cyber-Security Intelligence (BDCSIntell'18)*, volume 2343, pages 75–82.

Tran, T., Wang, H., Rudolph, S., and Cimiano, P. (2009). Top-k exploration of query candidates for efficient keyword search on graph-shaped (RDF) data. In *Proceedings of the 25th International Conference on Data Engineering (ICDE'09)*, pages 405–416.

Trivedi, P., Maheshwari, G., Dubey, M., and Lehmann, J. (2017). LC-QuAD: A Corpus for Complex Question Answering over Knowledge Graphs. In *Proceedings of the 16th International Semantic Web Conference (ISWC'17)*, pages 210–218.

Wen, Y., Jin, Y., and Yuan, X. (2018). KAT: Keywords-to-SPARQL translation over RDF graphs. In *Proceedings of the 23rd International Conference on Database Systems for Advanced Applications (DASFAA'18)*, volume 10827 LNCS, pages 802–810.

Zenz, G., Zhou, X., Minack, E., Siberski, W., and Nejdl, W. (2009). From keywords to semantic queries—Incremental query construction on the Semantic Web. *Web Semantics: Science, Services and Agents on the World Wide Web*, 7(3):166–176.

Zheng, W., Zou, L., Peng, W., Yan, X., Song, S., and Zhao, D. (2016). Semantic SPARQL similarity search over RDF knowledge graphs. In *Proceedings of the 42nd VLDB (VLDB'16)*, volume 9, pages 840–851.

Zhou, Q., Wang, C., Xiong, M., Wang, H., and Yu, Y. (2007). SPARK: Adapting keyword query to semantic search. In *Proceedings of the 6th International Semantic Web Conference (ISWC'07)*, volume 4825 LNCS, pages 694–707, Busan, Korea.