# Content-based Union and Complement Metrics for Dataset Search over RDF Knowledge Graphs

MICHALIS MOUNTANTONAKIS AND YANNIS TZITZIKAS, Institute of Computer Science, FORTH-ICS, Greece & Computer Science Department, University of Crete, Greece

RDF Knowledge Graphs (or Datasets) contain valuable information that can be exploited for a variety of real-world tasks. However, due to the enormous size of the available RDF datasets, it is difficult to discover the most valuable datasets for a given task. For improving dataset *Discoverability*, *Interlinking* and *Reusability*, there is a trend for *Dataset Search* systems. Such systems are mainly based on metadata and ignore the contents, however, in tasks related to data integration and enrichment, the contents of datasets have to be considered. This is important for data integration but also for data enrichment, for instance, quite often datasets' owners want to enrich the content of their dataset, by selecting datasets that provide complementary information for their dataset. The above tasks require content-based union and complement metrics between any subset of datasets, however, there is a lack of such approaches. For making feasible the computation of such metrics at very large scale, we propose an approach relying on a) a set of pre-constructed (and periodically refreshed) semantics-aware indexes , and b) "lattice-based" incremental algorithms that exploit the posting lists of such indexes, as well as set theory properties, for enabling efficient responses at query time. Finally, we discuss the efficiency of the proposed methods by presenting comparative results, and we report measurements for 400 real RDF datasets (containing over 2 billion triples), by exploiting the proposed metrics.

Additional Key Words and Phrases: Dataset Search, Dataset Quality, Interlinking, Enrichment, Reusability, Discoverability, Linked Data, Contextual Connectivity, Relevancy, Lattice of Measurements, Data Integration

**ACM Reference Format:**
Michalis Mountantonakis and Yannis Tzitzikas. 2019. Content-based Union and Complement Metrics for Dataset Search over RDF Knowledge Graphs. *ACM J. Data Inform. Quality* , (August 2019), 28 pages. https://doi.org/

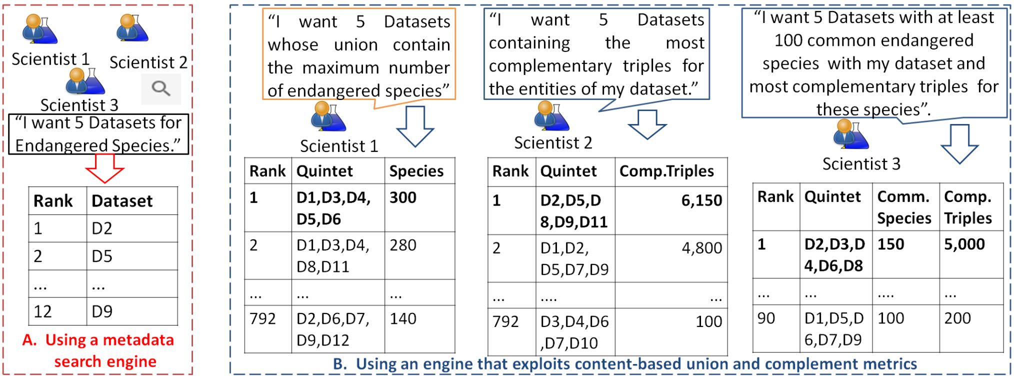
## 1 INTRODUCTION

RDF Knowledge Graphs (or Datasets) contain valuable information that can be exploited for a variety of real-world tasks and applications. Due to the enormous size and the heterogeneity of the available RDF datasets, it is not an easy task to identify which are the most valuable datasets for a given application. That problem becomes even more complicated, when one should select and integrate data from different sources, e.g., [11, 31]. For assisting dataset discoverability, interlinking and reusability, there is a trend for Dataset Search approaches [4], whose objective is to identify which datasets are relevant, valuable and reliable for fulfilling the requirements of a given task.

As a motivating example, suppose that in Fig. 1 three scientists want to perform an analysis for endangered species, i.e., for predicting the possibility those species to become extinct in the current century. For this reason, they desire to search and integrate at least $K$ (say five) different

Author's address: Michalis Mountantonakis and Yannis Tzitzikas, Institute of Computer Science, FORTH-ICS, Greece & Computer Science Department, University of Crete, Greece.

. 1936-1955/2019/8-ART $15.00
https://doi.org/


Scientist 1
Scientist 2
Scientist 3
"I want 5 Datasets for Endangered Species."
A. Using a metadata search engine
"I want 5 Datasets whose union contain the maximum number of endangered species"
Scientist 1
"I want 5 Datasets containing the most complementary triples for the entities of my dataset."
Scientist 2
"I want 5 Datasets with at least 100 common endangered species with my dataset and most complementary triples for these species".
Scientist 3
B. Using an engine that exploits content-based union and complement metrics

| Rank | Dataset |
|---|---|
| 1 | D2 |
| 2 | D5 |
| ... | ... |
| 12 | D9 |

| Rank | Quintet | Species |
|---|---|---|
| **1** | **D1,D3,D4, D5,D6** | **300** |
| 2 | D1,D3,D4, D8,D11 | 280 |
| ... | ... | ... |
| 792 | D2,D6,D7, D9,D12 | 140 |

| Rank | Quintet | Comp.Triples |
|---|---|---|
| **1** | **D2,D5,D 8,D9,D11** | **6,150** |
| 2 | D1,D2, D5,D7,D9 | 4,800 |
| ... | .... | ... |
| 792 | D3,D4,D6 ,D7,D10 | 100 |

| Rank | Quintet | Comm. Species | Comp. Triples |
|---|---|---|---|
| **1** | **D2,D3,D 4,D6,D8** | **150** | **5,000** |
| ... | ... | .... | ... |
| 90 | D1,D5,D 6,D7,D9 | 100 | 200 |

Fig. 1. Motivating Example. Three scientists want to find K datasets (say 5) from the 12 available ones, for performing an analysis for endangered species.

datasets containing information about such species. However, each of these scientists has different requirements (i.e., see the upper right side of Fig. 1). The first scientist has not collected any data, and he desires to discover the five datasets, whose union will produce an integrated dataset that will contain the maximum number of endangered species, i.e., comparing to any other combination of five datasets. In the second case, the scientist has already created a dataset, say $D_i$, containing information about several endangered species, and he desires to find five more datasets for enriching the information for the same species of $D_i$, i.e., by finding complementary information for those species. In the third case, the scientist wants also to enrich the content of his dataset (say $D_j$), however, his main requirement is the five desired datasets to contain at least 100 endangered species which are also covered in $D_j$. In that example, we suppose that there are available 12 different related datasets, which result to 792 possible quintets of datasets, thereby, it is time-consuming for the scientists to check all the possible quintets for discovering the best one for them. The above user needs should be taken into account, however, they are not covered from existing approaches.

Current methods for dataset search exploit metadata for aiding the discovery of datasets. Such metadata usually describe generic information about a dataset, like its description, a set of keywords, etc. By using services such as *Google Dataset search* (http://toolbox.google.com/datasetsearch) or *datahub.io*, one can find relevant datasets according to some keywords. In particular, they show to the users several datasets according to a ranking, however, the user does not know whether these datasets are worth to be combined, e.g., in Fig. 1 (left side) such methods return the same ranking list of single datasets for all three scientists. Another drawback of these approaches is that they contain several datasets with unstructured data, which makes that problem even harder, i.e., it is not easy to find all the datasets containing information about the same entities. The problem of unstructured data can be alleviated by adopting *Linked Data* [1], since linking can be achieved through common URIs or/and through equivalence relationships (e.g., `owl:sameAs` relationships). Existing Linked Data content-based approaches, such as LODStats [7] and LODLaundromat [29], offer services for searching and discovering datasets containing specific URIs or/and keywords, while SPARQLES [34] and SpEnD [37] exploit metadata for searching for SPARQL Endpoints. However, the main target of these approaches is to offer statistics, preservation and quality assessment services, i.e., they do not focus on assessing the *quality among combinations of datasets*. Thereby, it is impossible to get back hits each corresponding to a set (or "cluster") of datasets.

Another problem related to Discoverability and Reusability is that "the reuse and take-up of rich data sources is often limited and focused on a few well-known and established RDF datasets" [3]. Thereby, a key challenge is how to reveal the importance of high quality under-recognized datasets. For example, suppose that an under-recognized dataset, called $D_{Tiger}$, contains valuable information only for the species “Tiger". However, since the famous RDF datasets, e.g., DBpedia [14], Wikidata [35], YAGO [26], contain information about this species, in a global dataset ranking, the dataset $D_{Tiger}$ would be in a very low position comparing to the popular datasets, even if the desired task is to collect information about “Tiger".

For tackling the above problems, we propose content-based union and complement metrics among any subset of datasets, which can be exploited for several tasks i.e., a) for improving *Dataset Search and Discoverability*, b) for assessing and improving *Dataset Quality* in terms of *Interlinking and Reusability*, and c) for estimating *Data Integration Quality*. Concerning *Dataset Search and Discoverability*, a major challenge is to propose dataset-specific quality metrics, for returning ranking lists of multiple datasets, instead of single ones, since “Many tasks involving datasets require the stitching of several datasets to create a whole that is fit for purpose" [4]. Past works [16–18] have proposed content-based intersection metrics that measure the connectivity among several datasets and can be used by search engines for returning ranking lists of multiple datasets. However, by using only connectivity metrics, the following queries (and the queries of Fig. 1) are not answerable:
$Q_{coverage}$: "Give me the K datasets that maximize the information (i.e., triples) for an entity (or a set of entities)" (i.e., union metrics are required).
$Q_{enrichment}$: "Give me the K datasets that contain the most complementary information (e.g., maximum number of triples) for the entities of my dataset" (i.e., requires absolute complement metrics).
$Q_{uniqueness}$: "Give me the K datasets, that my dataset contributes the most unique content" (i.e., relative complement metrics are required).

For instance, in Fig. 1, an engine that computes union and complement metrics, returns a ranking of quintets of datasets (instead of a ranking of single datasets), while the ranking is different for each scientist, according to their requirements. In particular, for *Scientist 1* the best quintet of datasets contains $\{D_1, D_3, D_4, D_5, D_6\}$, while for *Scientist 2*, the five best datasets for his needs are $\{D_2, D_5, D_8, D_9, D_{11}\}$. Therefore, we can obviously see that for different users (and even for the same task), (i) different combinations of datasets and (ii) different “slices" of a specific dataset can have different quality and value, e.g., even a small “slice" of an under-recognized dataset can be of primary importance for being used in a specific application.

Regarding *Interlinking* and *Reusability*, a dataset’s publisher should be able to assess the *quality* of his dataset comparing to other datasets, e.g., to assess whether his dataset is worth to be reused. For assisting this task, union and complement metrics can be exploited, for answering queries like, i) “How unique/redundant is the content of my dataset comparing to others?, ii) “What percentage of the available information for a set of entities is offered only from my dataset?", and iii) “Does my dataset enrich the content of other datasets?". These queries can assist publishers to improve their dataset’s quality in a long run, however, they cannot be answered from current systems.

Concerning *Data Integration*, the integration process is quite challenging for a plenty of reasons e.g., see [19] for a recent survey, and it requires a big effort and possibly a monetary cost. It includes various time-consuming processes, e.g., data cleaning, transformation, and others [13]. Therefore, it is important to estimate the quality of a possible integration by using such metrics, e.g., for answering queries like “What is the gain of integrating dataset $D_j$ with datasets $D_k$ and $D_m$?".

**Novelty of this Paper.** In our previous work [16–18], we introduced algorithms and methods for constructing semantics-aware indexes and for performing “lattice-based" measurements based on intersection. In this paper, we extend that work for assisting and improving the tasks (a)-(c) (i.e., *Dataset Search*, *Interlinking*, *Reusability* and *Data Integration*). To this end, we exploit the mentioned indexes and we introduce novel content-based union and complement metrics over any subset of datasets. However, the computation of such metrics in a straightforward way is expensive, since the possible number of subsets of datasets is exponential, and the size of datasets is quite big [17]. For being able to compute the metrics at global scale, in this paper:

- we propose methods that rely on semantics-aware indexes, that are enriched with inferred equivalence relationships and their structure enables the discovery of under-recognized datasets,
- we show how to exploit the posting lists of semantics-aware indexes, for offering content-based metrics for any subset of datasets, which rely on union and complement, i.e., we introduce the metrics *coverage*, *information enrichment* and *uniqueness*,
- we introduce lattice-based incremental algorithms that exploit set theory properties and pruning and regrouping methods, for enabling efficient responses at query time,
- we measure the efficiency of the proposed methods through comparative results and we show indicative measurements for 400 RDF datasets (and 2 billion triples).

Finally, for the 400 datasets that we use, our website (http://www.ics.forth.gr/isl/LODsyndesis) offers (in the category *Dataset Discovery, Search and Selection*) several services related to the real use cases described in this section (see also a tutorial video in https://youtu.be/UdQDgod6XME).

The rest of this paper is organized as follows: §2 provides the background and discusses related work, §3 introduces the problem statement and describes the context, §4 shows how to compute the metrics by exploiting pre-constructed indexes, while §5 shows how to compute the metrics for any subset of datasets incrementally. §6 reports efficiency results and shows indicative measurements over hundreds of datasets, and finally, §7 concludes the paper.

## 2 BACKGROUND & RELATED WORK

### 2.1 Background:RDF

Resource Description Framework (RDF) [1] is a graph-based data model. It uses URIs (Uniform Resource Identifiers) or anonymous resources (blank nodes) for denoting resources, and constants (Literals), while it uses triples for creating statements between a) two resources, or b) a resource and a constant. A triple is a statement of the form subject-predicate-object ⟨s,p,o⟩, and it is any element of $\mathcal{T}$=$(\mathcal{U}\cup\mathcal{BN})\times(\mathcal{U})\times(\mathcal{U}\cup\mathcal{BN}\cup\mathcal{L})$, where $\mathcal{U}$, $\mathcal{BN}$ and $\mathcal{L}$ denote the sets of URIs, blank nodes and literals, respectively. Any finite subset of $\mathcal{T}$ can correspond to an RDF graph (or dataset) and $triples(D_i) \subseteq \mathcal{T}$ are the triples of a dataset $D_i$. For instance, in the upper left side of Fig. 2, the triple ⟨ex:Tiger_Species,d1:maxWeight,"388.7kg"⟩ of dataset $D_1$, contains an entity, a property and a literal. Moreover, URIs can be divided in three (pairwise disjoint) sets $\mathcal{U} = \mathcal{E}\cup\mathcal{P}\cup\mathcal{C}$, i.e., $\mathcal{E}$ refers to entities (e.g., ex:Tiger_Species in Fig. 2), $\mathcal{P}$ to properties (e.g., d1:taxonName in Fig. 2), and $C$ to classes (e.g., d2:Mammal in Fig. 2). For distinguishing these three sets, we first define the set of properties $\mathcal{P} = \{p \in \mathcal{U} \mid \langle s,p,o\rangle \in \mathcal{T}\}$, then the set of classes $C = \{o \in \mathcal{U} \mid \langle s, rdf{:}type, o\rangle \in \mathcal{T}\}$, and finally we define the entities as follows: $\mathcal{E} = \{s \in \mathcal{U} \mid \langle s,p,o\rangle \in \mathcal{T},\ s \notin (\mathcal{P} \cup C)\} \cup \{o \in \mathcal{U} \mid \langle s,p,o\rangle \in \mathcal{T},\ o \notin (\mathcal{P} \cup C)\}$.

### 2.2 Related Work

In the context of Linked Data, there are several approaches offering *Dataset Search*. We divide them in four categories, *Dataset Based*, *Entity Based*, *Keyword Based*, and *Query Based* approaches. Moreover, in Table 1, we also show the number of RDF datasets and triples (if such information was included in the corresponding papers) that each approach uses, while we categorize them according to whether they exploit metadata or the content of datasets. Finally, we divide the works according to the output that they offer, i.e., an unranked list of single datasets (*ULD*), a ranked list of single datasets (*RLD*), a ranked list of a set (or “cluster") of datasets (*RLS*), or a visualization (*VL*).

| Approach | RDF Datasets' Number | Number of RDF Triples | Uses Metadata | Uses Content | Entity-Based | Dataset-Based | Keyword-Based | Query-Based | Output |
|---|---|---|---|---|---|---|---|---|---|
| ***LODsyndesis*** [16–18] | 400 | 2 billion | | ✓ | ✓ | ✓ | | | RLS, V |
| Neves et. al. [15, 21] | 1,113 | Unknown | ✓ | | | ✓ | | | RL |
| Ellefi et. al. [6] | 90 | Unknown | ✓ | | | ✓ | | | RL |
| Nikolov et al. [22] | Unknown | Unknown | ✓ | | | ✓ | ✓ | | RL, V |
| *LODVader* [2] | 491 | 1 billion | ✓ | | | ✓ | | | RL, V |
| *LinkLion* [20] | 476 | 77 million | | ✓ | | ✓ | | | UL |
| *FluidOps* [36] | 81,000* | 8 billion | ✓ | ✓ | | ✓ | | | RL, V |
| *LODStats* [7] | 9,960 | 149 billion | ✓ | ✓ | | | ✓ | | UL |
| *LODLaundromat* [9, 29] | 658,000** | 38 billion | | ✓ | ✓ | | ✓ | | RL, V |
| *WIMU* [32] | 667,960*** | 187 billion** | ✓ | ✓ | ✓ | | | | RL |
| *Datahub* | 1,272 | Unknown | ✓ | | | | ✓ | | UL |
| *ExpLOD* [12] | 11 | 12 mil. | ✓ | | | | ✓ | | RL |
| *LODAtlas* [25] | 1,281 | Unknown | ✓ | | | ✓ | ✓ | | RL, V |
| *LODatio+* [8] | 43,598,858** | 4 billion | ✓ | ✓ | | | | ✓ | RL |
| *SPARQLES* [34] | 545 | Unknown | ✓ | | | | ✓ | | UL, V |
| *SpEnD* [37] | 658 | Unknown | ✓ | | | | ✓ | | UL, V |
| *LOV* [33] | 660 | 827 thousand | ✓ | | | | ✓ | | UL, V |
| *VOID RKBExplorer* [23] | 11,193** | Unknown | ✓ | | ✓ | | | ✓ | UL |

Table 1. Categorizing existing Linked Data approaches for Dataset Search,*documents from WorldBank and Eurostat, **documents instead of datasets, ***documents, datasets from LODLaundromat and LODstats.

**Dataset-Based Category.** The input is a dataset $D_i$. Concerning metadata-based approaches, in [15, 21], probabilistic methods are used for finding the most relevant datasets for a given one. *LODVader* [2] is a tool that takes as input the metadata of a dataset, for finding datasets that have commonalities with the given dataset, while in [6] dataset recommendation techniques are used, for retrieving datasets that are worth to be linked with a given one. Nikolov et al. [22] introduced a keyword based search service over semantic indexes, for ranking potentially relevant datasets to a source dataset. Regarding content-based approaches, in [16–18] content-based intersection metrics are used for discovering the K most connected dataset to a given one. *LinkLion* [20] provides mappings between pairs of datasets. *FluidOps Data Portal* [36] creates clusters of entities for finding how much complementary information (e.g., entities, properties) a source dataset contributes to a target one, and produces a ranked list with the most relevant datasets for a given dataset.

**Entity-Based Category.** The input is a URI. By using services like *LODLaundromat* [29], *LODsyndesis* [16–18], and *WIMU* [32], one can type a URI, and each such service returns to the user all the datasets where that URI occurs. Moreover, ExpLOD [12] creates interlink usage summaries, which are used for understanding the contribution of each dataset for a real world entity. Finally, *RapidMiner Link Explorer* [30] discovers all the URIs and datasets for a specific entity at query time.

**Keyword-Based Category.** The input is one or more keywords. For instance, by using portals based on metadata, such as *datahub.io* and *Google Dataset* Search, one can find the most relevant datasets for a given keyword. *LODLaundromat* [29] offers content-based analytics for thousands of documents, and those datasets are exploited through *Lotus* [9], which is an engine that returns datasets and triples containing a keyword. *LODStats* [7] has collected metadata for 9,960 RDF datasets, and has also analyzed the content for a subset of them. It offers statistics and keyword search services for discovering datasets containing specific namespaces, vocabularies and others. SPARQLES [34] and SpEnD [37] exploit metadata for searching for SPARQL Endpoints, while *LODAtlas* [25] is a metadata-based engine, where one can type one or more keywords for finding datasets containing those keywords in their metadata. Moreover, *LOV* [33] offers a keyword search engine for retrieving relevant vocabularies. Finally, approaches using information retrieval techniques for discovering datasets through keyword search, e.g., [5, 24], have also been proposed.

**Query-based Category.** The input is a query, e.g., a SPARQL query. *LODatio+* [8] is a service where one can type a SPARQL query containing only types or/and properties, and returns to the user a ranked list of documents that can answer a specific query. *VOID RKBexplorer* [23] exploits

| $F \in \mathcal{F}$ | Name | $F(D_i)$ |
|---|---|---|
| $RWE$ | Real World Entities | $RWE(D_i) = \{[s] \mid \langle s,p,o\rangle \in triples(D_i), s \in \mathcal{E}\} \cup \{[o] \mid \langle s,p,o\rangle \in triples(D_i), o \in \mathcal{E}\}$ |
| $RWP$ | Real World Properties | $RWP(D_i) = \{[p] \mid \langle s,p,o\rangle \in triples(D_i), p \in \mathcal{P}\}$ |
| $RWC$ | Real World Classes | $RWC(D_i) = \{[o] \mid \langle s,p,o\rangle \in triples(D_i), o \in C\}$ |
| $RWT$ | Real world Triples | $RWT(D_i) = \{ \langle [s],[p],[o]\rangle \mid \langle s,p,o\rangle \in triples(D_i), o \in \mathcal{U}\} \cup \{ \langle [s],[p],o\rangle \mid \langle s,p,o\rangle \in triples(D_i), o \in \mathcal{L}\}$ |
| $RWT_{E'}$ | Real World Triples of a set of entities $E'$ | $RWT_{E'}(D_i) = \{ \langle [s],[p],[o]\rangle \in RWT(D_i) \mid s \in E' \text{ or } o \in E', E' \subseteq \mathcal{E}\}$ |
| $LIT$ | Literals | $LIT(D_i) = \{o \mid \langle s,p,o\rangle \in triples(D_i), o \in \mathcal{L}\}$ |

Table 2. The Six Measurement Types

metadata of thousands of documents and one can send SPARQL queries for retrieving the relevant datasets to a given topic and documents containing one or more entities.

**Dataset Search for Other Formats.** A state-of-the-art survey for Dataset Search [4] contains a lot of approaches providing dataset search for several formats (e.g., web tables, web pages). In the context of web pages, in [27], the authors proposed a novel way for estimating the value of different combinations of web sources, by using a probabilistic approach and several quality metrics [28].

**Our Placement.** Our work, i.e., *LODsyndesis* in Table 1, belongs into dataset-based and entity-based categories. It also belongs to *Quality Assessment* field [38], mainly to contextual quality dimensions, i.e., we support "content-based" *Relevancy*, since our target is to find all the relevant datasets for a given task, and to *Interlinking* dimension, since we measure whether two or more datasets are worth to be interlinked. Our work is also connected to *Dataset Profiling* [3], i.e., it enables the retrieval of "Contextual Connectivity" features. To the best of our knowledge, this is the first work proposing content-based union and complement metrics for any combination of datasets (not only for a single, or pairs of datasets, as in [2, 15, 20, 29, 36]). Comparing to most approaches of Table 1, we exploit the contents of datasets (and not metadata, e.g., as in [25, 34, 37]). Comparing to content-based approaches [29, 32, 36], we use indexes enriched with inferred equivalence relationships. Finally, comparing to our past work [16–18], we propose several new content-based metrics (and their computation methods) that can be exploited for several tasks.

## 3 PROBLEM STATEMENT & CONTEXT

In §3.1 we introduce the problem statement, whereas in §3.2 we describe the context.

### 3.1 Problem Statement

Let $\mathcal{D} = \{D_1, ..., D_n\}$ be a set of datasets, $\mathcal{P}(\mathcal{D})$ denote the power set of $\mathcal{D}$, and let $B$ be any set of datasets $B \subseteq \mathcal{D}$. Let $\mathcal{F} = \{RWE, RWP, RWC, LIT, RWT, RWT_{E'}\}$ be the measurements types that we focus on. We shall use $F$ to denote a specific measurement type ($F \in \mathcal{F}$), and $F(D_i)$ to denote the measurement type $F$ applied to a dataset $D_i$.

For each element $F$ of $\mathcal{F}$, Table 2 introduces its name and how $F(D_i)$ is defined. The notation $[u]$ refers to the class of equivalence for a given URI $u$. That means that the first three measurement types of a dataset $D_i$ (i.e., $RWE(D_i)$, $RWP(D_i)$ and $RWC(D_i)$), correspond to all the entities, properties and classes that can be found in $D_i$, where each such URI has been replaced by a unique class of equivalence. Analogously $RWT(D_i)$ contains all the triples of $D_i$ where the URIs of each triple have been replaced by their corresponding class of equivalence. Now $RWT_{E'}(D_i)$ denotes all the "real world" triples of $D_i$ that contain at least one entity $e \in E'$, either as a subject or as an object. Finally, $LIT(D_i)$ is the set of literals occurring in $D_i$.

Our objective is to be able to answer $Q_{coverage}$, $Q_{enrichment}$ and $Q_{uniqueness}$ for any measurement type $F \in \mathcal{F}$. Each is a maximization problem that is defined formally below.

#### 3.1.1 *Coverage.*

The answer of $Q_{coverage}$ queries corresponds to $covBest(K, F)$ defined as:
**Input:** An integer $K$, where $1 \leq K < |\mathcal{D}|$ and a measurement type $F \in \mathcal{F}$.
**Output:** A subset $B \subseteq \mathcal{D}$, s.t. $|B| = K$, that maximizes the following criterion:

**Maximization Criterion:**

$$covBest(K, F) = arg_B \max\ |cov(B, F)| \text{ where } cov(B, F) = \cup_{D_i \in B} F(D_i) \quad (1)$$

*3.1.2 Information Enrichment.* The answer of $Q_{enrichment}$ queries corresponds to $enrichBest(K, F, D_m)$ defined as:
**Input:** An integer $K$ ($1 \leq K < |\mathcal{D}| - 1$), a measurement type $F \in \mathcal{F}$ and a dataset $D_m$ ($D_m \in \mathcal{D}$).
**Output:** A subset $B \subseteq \mathcal{D}$, s.t. $|B| = K$ and $D_m \notin B$, that maximizes the following criterion:
**Maximization Criterion:**

$$\begin{aligned} enrichBest(K, F, D_m) &= arg_B \max\ |enrich(B, F, D_m)| \text{ where} \\ enrich(B, F, D_m) &= cov(B, F) \setminus F(D_m) \end{aligned}$$

*3.1.3 Uniqueness.* The answer of $Q_{uniqueness}$ queries corresponds to $uniqBest(D_m, F, K)$, defined as:
**Input:** An integer $K$ ($1 \leq K < |\mathcal{D}| - 1$), a measurement type $F \in \mathcal{F}$, and a dataset $D_m$ ($D_m \in \mathcal{D}$).
**Output:** A subset $B \subseteq \mathcal{D}$, s.t. $|B| = K$ and $D_m \notin B$, that maximizes the following criterion:
**Maximization Criterion:**

$$\begin{aligned} uniqBest(D_m, F, K) &= arg_B \max\ |uniq(D_m, F, B)| \text{ where} \\ uniq(D_m, F, B) &= F(D_m) \setminus cov(B, F) \end{aligned}$$

*3.1.4 Difficulty and Challenge.* From the above definitions we can see that the computation of $cov(B, F)$, $enrich(B, F, D_m)$ and $uniq(D_m, F, B)$ is reduced to the computation of *union*, *absolute complement* and *relative complement*, respectively. However, even for a single subset of datasets $B$ and a measurement type $F$, these set operations are quite expensive if $B$ contains datasets whose $F(D_i)$ is very big. Moreover, the number of possible solutions (for finding the datasets that maximize the required formulas), can be exponential in number (specifically the possible solutions for a given $K$ is given by the binomial coefficient formula: ${}_nC_K \equiv \begin{pmatrix} n \\ K \end{pmatrix} \equiv \frac{n!}{(n-K)!K!}$), which is prohibitively expensive for such maximization problems.

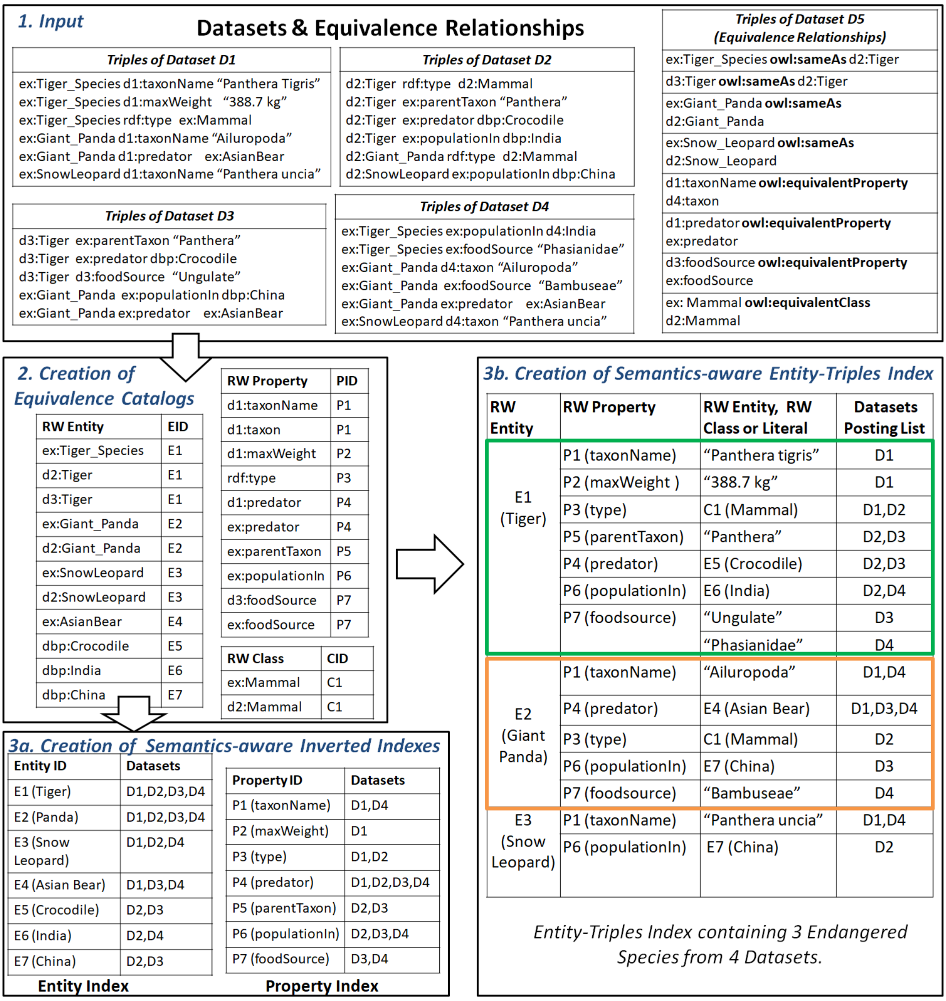
For tackling these issues, the objective is to solve the above maximization problems by reducing the number of set operations between different datasets for any measurement type $F$ and subset of datasets $B$. For this reason, we propose a solution based on the dedicated indexes $\mathcal{I}$ for each measurement type $F$, and we use set theory properties and pruning and regrouping methods for further reducing the number of set operations, i.e., we reuse the measurements $|cov(B, F)|$, $|enrich(B, F, D_m)|$ and $|uniq(D_m, F, B)|$, between two subsets of datasets $B$ and $B'$ (where $B \subset B'$).

## 3.2 Context:The Indexing Process

The computation of *coverage*, *information enrichment* and *uniqueness* will be based on five different indexes that have to be constructed once. For reasons of self-containedness, below we introduce in brief the steps of the indexing process (it is fully described in [17, 18]) through our running example (see Fig. 2).

**Step 1. Input.** The input is a set of datasets and equivalence relationships in schema and instance level (i.e., `owl:equivalentProperty`, `owl:equivalentClass` and `owl:sameAs`). Our running example (see Fig. 2) contains four datasets ($D_1 - D_4$), where each of them comprises triples for endangered species. Moreover, we use an additional dataset ($D_5$), which contains the equivalence relationships among the URIs of datasets $D_1 - D_4$. In the upper right side of Fig. 2, we can see the equivalence relationships among these datasets, e.g., ⟨ex:Giant_Panda,`owl:sameAs`,d3:Giant_Panda⟩.

**Step 2. Creation of Equivalence Catalogs.** We desire to keep a single representation for each real world entity, property and class. For this reason, we compute the transitive and symmetric



1. Input

Datasets & Equivalence Relationships

*Triples of Dataset D1*

ex:Tiger_Species d1:taxonName "Panthera Tigris"
ex:Tiger_Species d1:maxWeight "388.7 kg"
ex:Tiger_Species rdf:type ex:Mammal
ex:Giant_Panda d1:taxonName "Ailuropoda"
ex:Giant_Panda d1:predator ex:AsianBear
ex:SnowLeopard d1:taxonName "Panthera uncia"

*Triples of Dataset D2*

d2:Tiger rdf:type d2:Mammal
d2:Tiger ex:parentTaxon "Panthera"
d2:Tiger ex:predator dbp:Crocodile
d2:Tiger ex:populationIn dbp:India
d2:Giant_Panda rdf:type d2:Mammal
d2:SnowLeopard ex:populationIn dbp:China

*Triples of Dataset D3*

d3:Tiger ex:parentTaxon "Panthera"
d3:Tiger ex:predator dbp:Crocodile
d3:Tiger d3:foodSource "Ungulate"
ex:Giant_Panda ex:populationIn dbp:China
ex:Giant_Panda ex:predator ex:AsianBear

*Triples of Dataset D4*

ex:Tiger_Species ex:populationIn d4:India
ex:Tiger_Species ex:foodSource "Phasianidae"
ex:Giant_Panda d4:taxon "Ailuropoda"
ex:Giant_Panda ex:foodSource "Bambuseae"
ex:Giant_Panda ex:predator ex:AsianBear
ex:SnowLeopard d4:taxon "Panthera uncia"

*Triples of Dataset D5 (Equivalence Relationships)*

ex:Tiger_Species **owl:sameAs** d2:Tiger
d3:Tiger **owl:sameAs** d2:Tiger
ex:Giant_Panda **owl:sameAs** d2:Giant_Panda
ex:Snow_Leopard **owl:sameAs** d2:Snow_Leopard
d1:taxonName **owl:equivalentProperty** d4:taxon
d1:predator **owl:equivalentProperty** ex:predator
d3:foodSource **owl:equivalentProperty** ex:foodSource
ex: Mammal **owl:equivalentClass** d2:Mammal

2. Creation of Equivalence Catalogs

| RW Entity | EID |
|---|---|
| ex:Tiger_Species | E1 |
| d2:Tiger | E1 |
| d3:Tiger | E1 |
| ex:Giant_Panda | E2 |
| d2:Giant_Panda | E2 |
| ex:SnowLeopard | E3 |
| d2:SnowLeopard | E3 |
| ex:AsianBear | E4 |
| dbp:Crocodile | E5 |
| dbp:India | E6 |
| dbp:China | E7 |

| RW Property | PID |
|---|---|
| d1:taxonName | P1 |
| d1:taxon | P1 |
| d1:maxWeight | P2 |
| rdf:type | P3 |
| d1:predator | P4 |
| ex:predator | P4 |
| ex:parentTaxon | P5 |
| ex:populationIn | P6 |
| d3:foodSource | P7 |
| ex:foodSource | P7 |

| RW Class | CID |
|---|---|
| ex:Mammal | C1 |
| d2:Mammal | C1 |

3a. Creation of Semantics-aware Inverted Indexes

| Entity ID | Datasets |
|---|---|
| E1 (Tiger) | D1,D2,D3,D4 |
| E2 (Panda) | D1,D2,D3,D4 |
| E3 (Snow Leopard) | D1,D2,D4 |
| E4 (Asian Bear) | D1,D3,D4 |
| E5 (Crocodile) | D2,D3 |
| E6 (India) | D2,D4 |
| E7 (China) | D2,D3 |

Entity Index

| Property ID | Datasets |
|---|---|
| P1 (taxonName) | D1,D4 |
| P2 (maxWeight) | D1 |
| P3 (type) | D1,D2 |
| P4 (predator) | D1,D2,D3,D4 |
| P5 (parentTaxon) | D2,D3 |
| P6 (populationIn) | D2,D3,D4 |
| P7 (foodSource) | D3,D4 |

Property Index

3b. Creation of Semantics-aware Entity-Triples Index

| RW Entity | RW Property | RW Entity, RW Class or Literal | Datasets Posting List |
|---|---|---|---|
| E1 (Tiger) | P1 (taxonName) | "Panthera tigris" | D1 |
| | P2 (maxWeight ) | "388.7 kg" | D1 |
| | P3 (type) | C1 (Mammal) | D1,D2 |
| | P5 (parentTaxon) | "Panthera" | D2,D3 |
| | P4 (predator) | E5 (Crocodile) | D2,D3 |
| | P6 (populationIn) | E6 (India) | D2,D4 |
| | P7 (foodsource) | "Ungulate" | D3 |
| | | "Phasianidae" | D4 |
| E2 (Giant Panda) | P1 (taxonName) | "Ailuropoda" | D1,D4 |
| | P4 (predator) | E4 (Asian Bear) | D1,D3,D4 |
| | P3 (type) | C1 (Mammal) | D2 |
| | P6 (populationIn) | E7 (China) | D3 |
| | P7 (foodsource) | "Bambuseae" | D4 |
| E3 (Snow Leopard) | P1 (taxonName) | "Panthera uncia" | D1,D4 |
| | P6 (populationIn) | E7 (China) | D2 |

*Entity-Triples Index containing 3 Endangered Species from 4 Datasets.*

Fig. 2. Running example showing the indexing process for datasets $D_1 - D_4$.

closure of equivalence relationships (e.g., by using dataset $D_5$), and we create 3 equivalence catalogs, i.e., for entities, properties and classes. In this way, we create a single class of equivalence for each real world object, i.e., we replace the URIs referring to the same entity, property or class with a unique ID, e.g., in the left side of Fig. 2, the 3 URIs of "Tiger" replaced with the ID E1.

**Step 3. Creation of Semantically Enriched Indexes.** By using as input all the datasets (e.g., $D_1 - D_4$ in Fig. 2) and the equivalence catalogs (e.g., see Step 2 of Fig. 2), we construct the following set of indexes $\mathcal{I}$ = {*Entity Index*, *Property Index*, *Class Index*, *Entity-Triples Index*, *Literals Index*}. The first three indexes contain all the distinct "real world" entities ($RWE$), properties ($RWP$) and classes ($RWC$), respectively. For each category, we store in a posting list the datasets where each element (e.g., entity, property) occurs, e.g., see the produced *Entity Index* and *Property Index* in the lower left side of Fig. 2. Concerning *Literals Index* ($LIT$), it is an inverted index for the literals. Finally, in *Entity-Triples Index* we collect and store together all the available "real world" triples ($RWT$) and their provenance for each entity. In the right side of Fig. 2, we show the *Entity-Triples* Index, which contains all the triples and their provenance, for each of the endangered species that are included in datasets $D_1 - D_4$ (i.e., Tiger, Giant Panda, Snow Leopard). In the rest of this paper, we will use the *Entity-Triples* Index of Fig. 2, for providing examples for the proposed metrics.

**Step 4. Exploitation of Indexes for Supporting Content-based Metrics.** The structure of the indexes enables us to support the metrics introduced in §3.1 (*coverage*, *information enrichment* and *uniqueness*) for *Entity-Based Search*, i.e., we can read the part of the index containing information for a single entity (say "Tiger") or for a set of entities (say all the endangered species). This enables the discovery of under-recognized datasets that contain valuable information, but only for a specific domain or even a single entity, e.g., we can answer queries such as $Q_{coverage}$. We support also *Dataset-Based Search*, i.e., we can read the part of the index containing elements of a dataset $D_i$, for answering queries such as $Q_{enrichment}$ and $Q_{uniqueness}$. The indexes allow answering the intended questions fast, i.e., we use the posting lists of a stored index as input for special "lattice-based" algorithms, as we will explain in §4 and §5.

**Details about the Implementation.** In our implementation, we used 400 RDF datasets (in total two billion triples) collected manually through *datahub.io*, and 44 million of equivalence relationships collected from these 400 datasets and from the LinkLion webpage [20], i.e., we do not use any instance/schema matching tool. We used as input this set of equivalence relationships, and by using a cluster of machines and special algorithms (described in details in [18]), we computed their transitive and symmetric closure for inferring more relationships. However note that even a single incorrect relationship can produce several erroneous inferred ones [18], e.g., suppose that in Fig. 2 we had an erroneous relationship, e.g., ⟨ex:Tiger,`owl:sameAs`, d2:Giant_Panda⟩. Due to closure, we would infer that every URI referring to "Tiger" is equivalent to any URI of "Giant_Panda".

For tackling this issue, we checked the quality of the inferred relationships and we removed the erroneous ones in a semi-manually way. In particular, we supposed that an existing equivalence relationship is probably incorrect, if we inferred through this relationship that two or more URIs from the same dataset are equivalent. Indeed, we identified such cases programmatically (i.e., we found all the classes of equivalence that contain two or more URIs from the same dataset), we removed the incorrect relationships manually, and then we created the equivalence catalogs.

Afterwards, we used as input the triples of these 400 datasets and the equivalence catalogs for creating the set of indexes $I$. These indexes are used as input for the algorithms that are presented in this paper, for answering queries and providing services, based on *coverage*, *information enrichment* and *uniqueness* metrics. For the time being, our webpage (http://www.ics.forth.gr/isl/LODsyndesis) offers such services for the 400 RDF datasets which have been indexed by our approach. More statistics and details about the datasets and the implementation are given in [17, 18].

## 4 USING INDEXES FOR COMPUTING UNION AND COMPLEMENT METRICS

In §4.1 we describe how to exploit an index $ind_F \in \mathcal{I}$ (given a measurement type $F$) for computing *coverage*, *information enrichment* and *uniqueness*, in §4.2 we describe how to present and to visualize the results of the measurements, while in §4.3-§4.5, we describe how one can compute the three proposed content-based metrics, by using straightforward methods, which are not quite efficient for the maximization problems described in §3.1.

### 4.1 Direct Counts

Here, we show how to exploit an index $ind_F \in \mathcal{I}$ for computing the metrics for a measurement type $F$. Let $k \in Left(ind_F)$ be a single entry in the left part of the index $ind_F$, meaning that $k$ is a key while $ind_F(k)$ corresponds to the value of this key (i.e., it is a posting list). Below we define the number of times a subset $B$ occurs in the posting lists of an index $ind_F$.

*Definition 4.1.* $directCount(B, F) = |\{k \in Left(ind_F) \mid ind_F(k) = B\}|$ ◇

The value of $directCount(B, F)$ is computed by scanning the corresponding index ($ind_F$) once. The result of this process is a *directCount* list, where each entry contains at the left side a subset $B$ and in the right side the *directCount* score of $B$. If $m$ is the total number of entries of an index, then the time complexity for constructing that list is $O(m)$.

For instance, in Query $Q_{coverage}$ of Fig. 3, a scientist desires to find the triad of datasets having the maximum coverage for the species "Tiger". For creating the *directCount* list, we scanned only the triples of "Tiger" (i.e., we focus on $F = RWT_{\{E1\}}$, since E1 corresponds to "Tiger"), in the *Entity-Triples Index* (i.e., see the green box in Fig. 2). In the upper right side of Fig. 3, we can see the produced *directCount* list for Query $Q_{coverage}$. As we can observe, the $directCount(\{D_2, D_3\}, F) = 2$, since the subset $\{D_2, D_3\}$ occurs in total two times in the posting lists of "Tiger" triples (see Fig. 2). Particularly, the triples ⟨Tiger, parentTaxon, Panthera⟩ and ⟨Tiger, predator, Crocodile⟩ exist in both datasets $D_2$ and $D_3$, but not in any other dataset.

**How to Use Only a Part of the *directCount* List?** In many cases, especially for computing $|enrich(B, F, D_m)|$, $|uniq(D_m, F, B)|$ and for the incremental algorithms that will be introduced in §5, it is a prerequisite to exclude or to keep some posting lists containing specific datasets. For this reason, we introduce three definitions, while for each of them we show an example for the *directCount* list of Query $Q_{coverage}$ (see the upper right side of Fig. 3).

First, for a given measurement type $F$, we define the posting lists (i.e., subsets of datasets) which contain at least one dataset $D_i \in B$, i.e., we define the set $occur(B, F)$ as follows:

*Definition 4.2.* $occur(B, F) = \{B_i \in \mathcal{P}(\mathcal{D}) \mid directCount(B_i, F) > 0, B_i \cap B \neq \emptyset\}$

Obviously, $occur(B, F)$ is a finite set. In the extreme case when $B = \mathcal{D}$, $occur(\mathcal{D}, F)$ contains all the entries in the left side of the *directCount* list for a given $F$, e.g., in Query $Q_{coverage}$ of Fig. 3, $occur(\mathcal{D}, F) = \{\{D_1\}, \{D_1, D_2\}, \{D_2, D_3\}, \{D_2, D_4\},\{D_3\}, \{D_4\}\}$.

In many cases (especially for computing $|enrich(B, F, D_m)|$ and for the incremental algorithms of §5), we should keep only the posting lists containing at least one $D_j \in B'$, but not any $D_i \in B$ (i.e., we exclude such posting lists). For this reason, we define the set $occur(B', F)_{\setminus B}$ as follows:

*Definition 4.3.* $occur(B', F)_{\setminus B} = occur(B', F) \setminus occur(B, F)$ ⋄

Obviously, $occur(B', F)_{\setminus B}$ is a finite set. For instance, in the *directCount* list which is shown in Step B of Fig. 3, if we want to exclude the posting lists containing either $D_1$ or $D_3$ (and to keep all the remaining ones), we will construct the set $occur(\mathcal{D}, F)_{\setminus \{D_1, D_3\}} = \{\{D_2, D_4\}, \{D_4\}\}$.

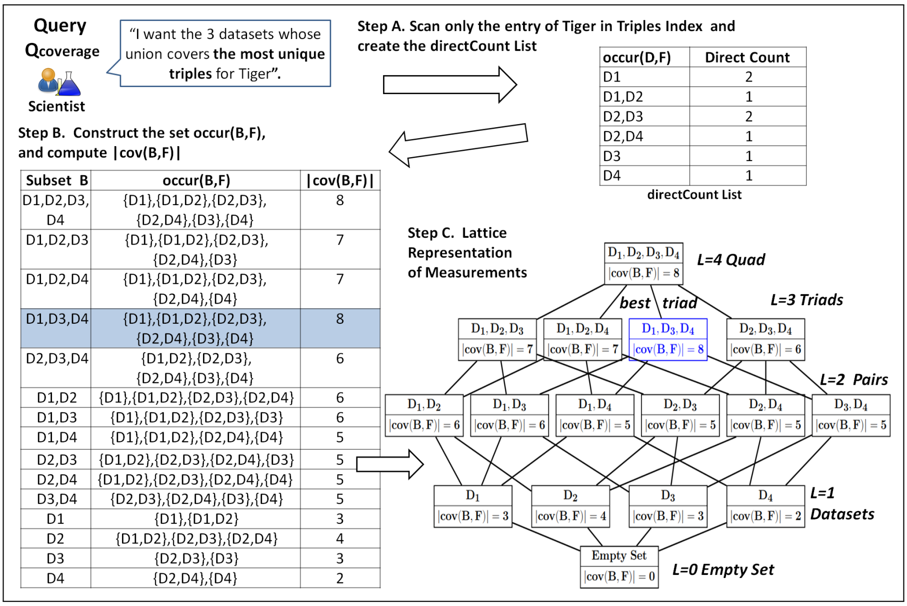
Finally, for computing $|uniq(D_m, F, B)|$ (in §4.5), we should keep all the posting lists containing the dataset $D_m$ and at least one dataset $D_i \in B$, i.e., we define the set $occur(B, F)_{D_m}$ as follows:

*Definition 4.4.* $occur(B, F)_{D_m} = occur(B, F) \cap occur(\{D_m\}, F)$ ⋄

Obviously, $occur(B, F)_{D_m}$ is a finite set. For the *directCount* list of Fig. 3, if $B = \{D_1, D_3\}$ and $D_m = D_2$, then $occur(\{D_1, D_3\}, F)_{D_2} = \{\{D_1, D_2\}, \{D_2, D_3\}\}$, i.e., we keep a posting list if it contains $D_2$ and at least one dataset of $B$, i.e., $D_1$ or $D_3$.

### 4.2 The Lattice of Measurements

Given a set of measurements one question is how to present and visualize them to aid understanding. Here we describe a lattice-based visualization of several measurements. Moreover, this lattice structure can be exploited for the incremental computation of the metrics (as we shall see in §5). We propose a Hasse Diagram-like structure, as shown in Step C of Fig. 3. Our approach is based on a lattice which is represented as a Directed Acyclic Graph $G = (V, E)$. The lattice of the power set of $\mathcal{D}$ (i.e., $\mathcal{P}(\mathcal{D})$) contains $|V| = 2^{|\mathcal{D}|}$ nodes, where each node corresponds to a subset of datasets


Query
Qcoverage
Scientist
"I want the 3 datasets whose union covers the most unique triples for Tiger".
L=4 Quad
best triad
L=3 Triads
L=2 Pairs
L=1 Datasets
L=0 Empty Set
$D_1,D_2,D_3,D_4$ $|cov(B,F)| = 8$
$D_1,D_2,D_3$ $|cov(B,F)| = 7$
$D_1,D_2,D_4$ $|cov(B,F)| = 7$
$D_1,D_3,D_4$ $|cov(B,F)| = 8$
$D_2,D_3,D_4$ $|cov(B,F)| = 6$
$D_1,D_2$ $|cov(B,F)| = 6$
$D_1,D_3$ $|cov(B,F)| = 6$
$D_1,D_4$ $|cov(B,F)| = 5$
$D_2,D_3$ $|cov(B,F)| = 5$
$D_2,D_4$ $|cov(B,F)| = 5$
$D_3,D_4$ $|cov(B,F)| = 5$
$D_1$ $|cov(B,F)| = 3$
$D_2$ $|cov(B,F)| = 4$
$D_3$ $|cov(B,F)| = 3$
$D_4$ $|cov(B,F)| = 2$
Empty Set $|cov(B,F)| = 0$

Step A. Scan only the entry of Tiger in Triples Index and create the directCount List

| occur(D,F) | Direct Count |
|---|---|
| D1 | 2 |
| D1,D2 | 1 |
| D2,D3 | 2 |
| D2,D4 | 1 |
| D3 | 1 |
| D4 | 1 |

directCount List

Step B. Construct the set occur(B,F), and compute |cov(B,F)|

| Subset B | occur(B,F) | \|cov(B,F)\| |
|---|---|---|
| D1,D2,D3, D4 | {D1},{D1,D2},{D2,D3}, {D2,D4},{D3},{D4} | 8 |
| D1,D2,D3 | {D1},{D1,D2},{D2,D3}, {D2,D4},{D3} | 7 |
| D1,D2,D4 | {D1},{D1,D2},{D2,D3}, {D2,D4},{D4} | 7 |
| D1,D3,D4 | {D1},{D1,D2},{D2,D3}, {D2,D4},{D3},{D4} | 8 |
| D2,D3,D4 | {D1,D2},{D2,D3}, {D2,D4},{D3},{D4} | 6 |
| D1,D2 | {D1},{D1,D2},{D2,D3},{D2,D4} | 6 |
| D1,D3 | {D1},{D1,D2},{D2,D3},{D3} | 6 |
| D1,D4 | {D1},{D1,D2},{D2,D4},{D4} | 5 |
| D2,D3 | {D1,D2},{D2,D3},{D2,D4},{D3} | 5 |
| D2,D4 | {D1,D2},{D2,D3},{D2,D4},{D4} | 5 |
| D3,D4 | {D2,D3},{D2,D4},{D3},{D4} | 5 |
| D1 | {D1},{D1,D2} | 3 |
| D2 | {D1,D2},{D2,D3},{D2,D4} | 4 |
| D3 | {D2,D3},{D3} | 3 |
| D4 | {D2,D4},{D4} | 2 |

Step C. Lattice Representation of Measurements

Fig. 3. Query $Q_{coverage}$. Computation of Coverage

$B \in \mathcal{P}(\mathcal{D})$, and $|E| = |\mathcal{D}| * 2^{|\mathcal{D}|-1}$ edges, where each edge points towards the direct supersets, i.e., an edge is created from a subset $B$ to a superset $B'$, where $B' = B \cup \{D_i\}, D_i \notin B$. We can divide the lattice in $|\mathcal{D}|$+1 levels, where each level $L$ ($0 \leq L \leq |\mathcal{D}|$) consists of subsets having exactly $L$ datasets, e.g., $L = 3$ corresponds to triads of datasets (see the lattice in Fig. 3).

## 4.3 Computation of Coverage

Here, we show how to exploit a pre-constructed *directCount* list (i.e., the set $occur(\mathcal{D}, F)$ and the corresponding *directCount* scores) for computing coverage, i.e., $|cov(B, F)|$.

Prop. 1. The sum of the *directCount* score of all $B_i \in occur(B, F)$ gives the cardinality of coverage for a subset $B$ and a given measurement type $F$:

$$|cov(B, F)| = \sum_{B_i \in occur(B,F)} directCount(B_i, F) \tag{2}$$

Proof. The proof is given in Appendix A.1. □

*4.3.1 Straightforward Computation of* $|cov(B, F)|$ *(*SF *method).* For each different subset $B$, we iterate over the set $occur(\mathcal{D}, F)$ once, for creating the set $occur(B, F)$. In particular, $\forall B_i \in occur(\mathcal{D}, F)$, we check if $B_i \cap B \neq \emptyset$, and if it holds, we add $B_i$ to $occur(B, F)$. Afterwards, we read the set $occur(B, F)$ once, for computing $|cov(B, F)|$ (see Prop. 1). For computing the $|cov(B, F)|$ for a finite set of subset of datasets $BV = \{B_i, ..., B_m\}$, the time complexity is $O(|BV| * |occur(\mathcal{D}, F)|)$, since we traverse once the whole set $occur(\mathcal{D}, F)$ for each $B \in BV$ (in the worst case $|BV| = 2^{|\mathcal{D}|}$). The space complexity is $O(|occur(\mathcal{D}, F)|)$ (i.e., we keep in memory the whole *directCount* list).

Concerning the check $B_i \cap B \neq \emptyset$ it has time complexity $O(m * log_n)$ where $m = \max(|B_i|, |B|)$ and $n = \min(|B_i|, |B|)$, since both $B_i$ and $B_i$ are ordered sets. We ignore this complexity in $O(2^{|\mathcal{D}|} * |occur(\mathcal{D}, F)|)$, since even in the worst case (when $m = n = |\mathcal{D}|$), the cost is quite smaller comparing to the number of possible subsets ($2^{|\mathcal{D}}|$) and to the size of $occur(\mathcal{D}, F)$, i.e., in the extreme case when the index contains any $B \in \mathcal{P}(\mathcal{D})$ as a posting list, its size equals $2^{|\mathcal{D}|}$.

**Example.** In Fig. 3, we show the steps for computing the coverage of triples for “Tiger" species (i.e., $F = RWT_{\{E1\}}$). For each subset of datasets $B$, we scan the whole set $occur(\mathcal{D}, F)$ (in Step A of Fig. 3), for creating the set $occur(B, F)$, and we compute $|cov(B, F)|$ (in Step B in Fig. 3). For example, for $B = \{D_1, D_2\}$, we constructed the set $occur(\{D_1, D_2\}, F) = \{\{D_1\}, \{D_1, D_2\}, \{D_2, D_3\}, \{D_2, D_4\}\}$. By taking the sum of their $directCount$ score (i.e., $directCount(\{D_1\}, F) = 2$, $directCount(\{D_1, D_2\}, F) = 1$, $directCount(\{D_2, D_3\}, F) = 2$, $directCount(\{D_2, D_4\}, F) = 1$) we computed $|cov(\{D_1, D_2\}, F)| = 6$. By seeing the results in Step C of Fig. 3, we observe that the best triad for the scientist is $covBest(3, F) = \{D_1, D_3, D_4\}$, since its union contains the maximum number of triples for “Tiger".

*4.3.2 Extra/Derived Coverage-related Metrics.* Here we describe some extra coverage-related metrics.

*Coverage Percentage.* We define the coverage percentage of a subset of datasets $B$ for a given $F$:

$$covPer(B, F, \mathcal{D}) = \frac{|cov(B, F)| * 100}{|cov(\mathcal{D}, F)|} \tag{3}$$

It can be used for answering questions, such as, “What percentage of all the available triples for “Tiger" species is covered from the union of DBpedia and YAGO datasets?". For example, in Fig. 3, the $covPer(\{D_1, D_3, D_4\}, RWT_{\{E1\}}, \mathcal{D}) = 100\%$, since the union of these datasets contain all the available triples for “Tiger" (in total 8 triples), in our running example of Fig. 2.

*Coverage Gain.* We define the coverage gain between a subset $B$ and its superset $B'$, ($B \subset B'$), for a given $F$ as follows:

$$covGain(B', F, B) = \frac{(|cov(B', F)| - |cov(B, F)|) * 100}{|cov(B, F)|} \tag{4}$$

We measure whether there is an increase in coverage, if we add (i.e., integrate) more datasets to a given subset of datasets. For instance, suppose that in Fig. 3 we want to measure the gain between $B = \{D_1, D_3, D_4\}$ and $B' = \{D_1, D_2, D_3, D_4\}$. Since $|cov(B, RWT_{\{E1\}})| = 8$ and $|cov(B', RWT_{\{E1\}})| = 8$, the coverage gain is $covGain(B', RWT_{\{E1\}}, B) = 0\%$. It means that for this task, the addition of $D_2$ to the subset $\{D_1, D_3, D_4\}$ is useless, i.e., $D_2$ does not offer additional unique triples for “Tiger".

## 4.4 Computation of Information Enrichment given a Dataset $D_m$

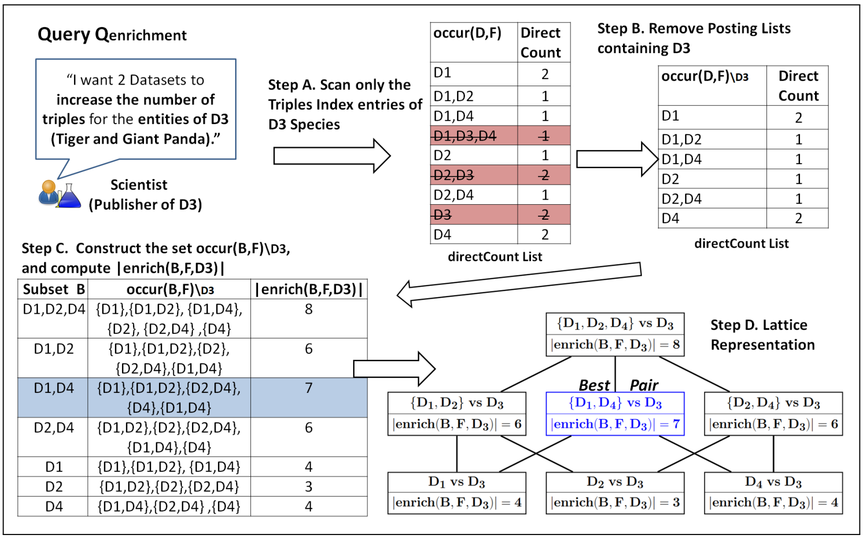
Here, we exclude any posting list containing $D_m$ (since we want to compute the complement to a dataset $D_m$), i.e., we use as input the set $occur(\mathcal{D}, F)_{\setminus\{D_m\}}$ instead of $occur(\mathcal{D}, F)$.

Prop. 2. For a subset $B$, a dataset $D_m$ and a given $F$, the sum of the $directCount$ score of all the $B_i \in occur(B, F)_{\setminus\{D_m\}}$ gives the cardinality of information enrichment:

$$|enrich(B, F, D_m)| = \sum_{B_i \in occur(B, F)_{\setminus\{D_m\}}} directCount(B_i, F) \tag{5}$$

Proof. The proof is given in Appendix A.2. □

*4.4.1 Straightforward Computation of $|enrich(B, F, D_m)|$ (SF method).* For a single subset $B$, we iterate over the set $occur(\mathcal{D}, F)_{\setminus\{D_m\}}$ once, for constructing the set $occur(B, F)_{\setminus\{D_m\}}$. In particular, for each $B_i \in occur(\mathcal{D}, F)_{\setminus\{D_m\}}$, we check if $B_i \cap B \neq \emptyset$, and if it holds, we add $B_i$ to $occur(B, F)_{\setminus\{D_m\}}$. In the end, we sum the $directCount(B_i, F)$ score of each $B_i \in occur(B, F)_{\setminus\{D_m\}}$, for computing $|enrich(B, F, D_m)|$ (see Prop. 2). For a finite set of subset of datasets $BV = \{B_i, ..., B_m\}$, the time complexity is $O(|BV| * |occur(\mathcal{D}, F)_{\setminus\{D_m\}}|)$, where in the worst case $|BV| = 2^{|\mathcal{D}|-1}$, i.e., we compute $|enrich(B, F, D_m)|$ for all the possible subsets of datasets that do not contain $D_m$. Moreover, the space complexity is $O(|occur(\mathcal{D}, F)_{\setminus\{D_m\}}|)$. For the complexity of checking non-null intersection, i.e., $B_i \cap B \neq \emptyset$, see §4.3.



"I want 2 Datasets to **increase the number of triples** for the **entities of D3 (Tiger and Giant Panda).**"

| occur(D,F) | Direct Count |
|---|---|
| D1 | 2 |
| D1,D2 | 1 |
| D1,D4 | 1 |
| ~~D1,D3,D4~~ | ~~1~~ |
| D2 | 1 |
| ~~D2,D3~~ | ~~2~~ |
| D2,D4 | 1 |
| ~~D3~~ | ~~2~~ |
| D4 | 2 |

| occur(D,F)\D3 | Direct Count |
|---|---|
| D1 | 2 |
| D1,D2 | 1 |
| D1,D4 | 1 |
| D2 | 1 |
| D2,D4 | 1 |
| D4 | 2 |

| Subset B | occur(B,F)\D3 | \|enrich(B,F,D3)\| |
|---|---|---|
| D1,D2,D4 | {D1},{D1,D2}, {D1,D4}, {D2}, {D2,D4} ,{D4} | 8 |
| D1,D2 | {D1},{D1,D2},{D2}, {D2,D4},{D1,D4} | 6 |
| D1,D4 | {D1},{D1,D2},{D2,D4}, {D4},{D1,D4} | 7 |
| D2,D4 | {D1,D2},{D2},{D2,D4}, {D1,D4},{D4} | 6 |
| D1 | {D1},{D1,D2}, {D1,D4} | 4 |
| D2 | {D1,D2},{D2},{D2,D4} | 3 |
| D4 | {D1,D4},{D2,D4} ,{D4} | 4 |

Fig. 4. Query $Q_{enrichment}$. Computation of Information Enrichment

**Example.** In Fig. 4, the publisher of $D_3$, desires to find two datasets containing additional information for $D_3$ endangered species (i.e., "Tiger" and "Giant Panda"). Therefore, we focus on $F = RWT_{\{E1,E2\}}$, since E1 corresponds to "Tiger" and E2 to "Giant Panda". For this reason, in Step A, we scan only their entries in the *Entity-Triples* index, i.e., green and orange boxes in Fig. 2, for constructing the set $occur(\mathcal{D}, F)$. However, since our target is to find the complementary triples for $D_3$ entities, in Step B we exclude all the posting lists containing $D_3$, i.e., we construct the set $occur(\mathcal{D}, F)_{\setminus\{D_3\}} = \{\{D_1\}, \{D_1, D_2\}, \{D_1, D_4\}, \{D_2\}, \{D_2, D_4\}, \{D_4\}\}$, and we store their corresponding *directCount* score. By traversing the set $occur(\mathcal{D}, F)_{\setminus\{D_3\}}$, for each subset $B$, we create the set $occur(B, F)_{\setminus\{D_3\}}$, and we compute the corresponding $|enrich(B, F, D_3)|$, e.g., see Step C of Fig. 4. For instance, for computing $|enrich(\{D_1, D_2\}, F, \{D_3\})|$, we constructed $occur(\{D_1, D_2\}, F)_{\setminus\{D_3\}} = \{\{D_1\}, \{D_1, D_2\}, \{D_1, D_4\}, \{D_2\}, \{D_2, D_4\}\}$ and we computed the sum of their *directCount* score, for finding $|enrich(\{D_1, D_2\}, F, \{D_3\})| = 6$. In Step D of Fig. 4, we observe that the best pair is $enrichBest(2, F, D_3) = \{D_1, D_4\}$, since it offers 7 additional triples for the species of $D_3$.

#### 4.4.2 Extra/Derived Information Enrichment related Metrics.

Here, we show one extra metric related to information enrichment.

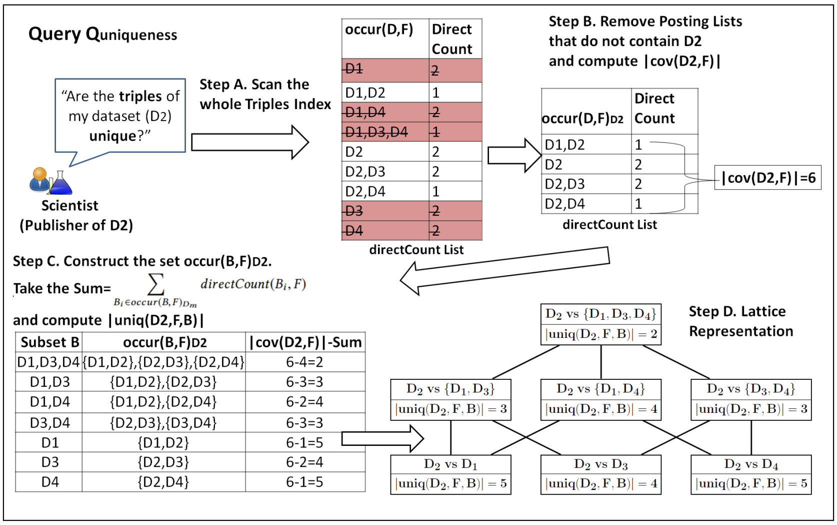
*Information Enrichment Percentage for a single dataset $D_m$.* We define the increase percentage of information enrichment for a dataset $D_m$ in a subset of datasets $B$, for a given $F$, as follows:

$$enrichPer(B, F, D_m) = \frac{|enrich(B, F, D_m)| * 100}{|cov(\{D_m\}, F)|} \qquad (6)$$

This metric shows the percentage that the information of a given dataset $D_m$ increases, in a possible integration with the datasets of subset $B$. For example, in Query $Q_{enrichment}$ of Fig. 4, by integrating $\{D_1, D_2\}$ with $D_3$, $enrichPer(\{D_1, D_2\}, RWT_{\{E1,E2\}}, D_3) = 120\%$, since $\{D_1, D_2\}$ offers 6 new triples for the entities of $D_3$, i.e., $|enrich(\{D_1, D_2\}, RWT_{\{E1,E2\}}, D_3)| = 6$, which is divided by the total number of $D_3$ triples for these two species, i.e., $|cov(\{D_3\}, RWT_{\{E1,E2\}})| = 5$.

### 4.5 Computation of Uniqueness given a Dataset $D_m$

Here, the input is only the set $occur(\{D_m\}, F)$ (and not $occur(\mathcal{D}, F)$), since we focus only on elements that can be found in $D_m$.


Query Quniqueness
"Are the triples of my dataset (D2) unique?"
Scientist (Publisher of D2)
Step A. Scan the whole Triples Index
Step B. Remove Posting Lists that do not contain D2 and compute |cov(D2,F)|
|cov(D2,F)|=6
directCount List
Step C. Construct the set occur(B,F)D2.
Take the Sum= $\sum_{B_i \in occur(B,F)_{D_m}} directCount(B_i, F)$
and compute |uniq(D2,F,B)|
Step D. Lattice Representation
$D_2$ vs $\{D_1, D_3, D_4\}$ $|uniq(D_2, F, B)| = 2$
$D_2$ vs $\{D_1, D_3\}$ $|uniq(D_2, F, B)| = 3$
$D_2$ vs $\{D_1, D_4\}$ $|uniq(D_2, F, B)| = 4$
$D_2$ vs $\{D_3, D_4\}$ $|uniq(D_2, F, B)| = 3$
$D_2$ vs $D_1$ $|uniq(D_2, F, B)| = 5$
$D_2$ vs $D_3$ $|uniq(D_2, F, B)| = 4$
$D_2$ vs $D_4$ $|uniq(D_2, F, B)| = 5$

| occur(D,F) | Direct Count |
|---|---|
| ~~D1~~ | ~~2~~ |
| D1,D2 | 1 |
| ~~D1,D4~~ | ~~2~~ |
| ~~D1,D3,D4~~ | ~~1~~ |
| D2 | 2 |
| D2,D3 | 2 |
| D2,D4 | 1 |
| ~~D3~~ | ~~2~~ |
| ~~D4~~ | ~~2~~ |

directCount List

| occur(D,F)D2 | Direct Count |
|---|---|
| D1,D2 | 1 |
| D2 | 2 |
| D2,D3 | 2 |
| D2,D4 | 1 |

directCount List

| Subset B | occur(B,F)D2 | \|cov(D2,F)\|-Sum |
|---|---|---|
| D1,D3,D4 | {D1,D2},{D2,D3},{D2,D4} | 6-4=2 |
| D1,D3 | {D1,D2},{D2,D3} | 6-3=3 |
| D1,D4 | {D1,D2},{D2,D4} | 6-2=4 |
| D3,D4 | {D2,D3},{D3,D4} | 6-3=3 |
| D1 | {D1,D2} | 6-1=5 |
| D3 | {D2,D3} | 6-2=4 |
| D4 | {D2,D4} | 6-1=5 |

Fig. 5. Query $Q_{uniqueness}$. Computation of Uniqueness

Prop. 3. For a subset $B$, a dataset $D_m$ and a given $F$, the uniqueness of a dataset $D_m$ to a subset $B$ can be given by subtracting from the cardinality of all the elements of $D_m$, the cardinality of all the elements that co occur in $D_m$ and in at least one $D_i \in B$.

$$|uniq(D_m, F, B)| = |cov(\{D_m\}, F)| - \sum_{B_i \in occur(B,F)_{D_m}} directCount(B_i, F) \tag{7}$$

Proof. The proof is given in Appendix A.3. □

*4.5.1 Straightforward Computation of $|uniq(D_m, F, B)|$ (SF method).* First, we should compute $|cov(\{D_m\}, F)|$ by scanning the $occur(\{D_m\}, F)$ once, since it is required for the computation of $|uniq(D_m, F, B)|$, for any subset of datasets $B$ (see Prop. 3). For a single subset $B$, we traverse the set $occur(\{D_m\}, F)$ once, for constructing the set $occur(B, F)_{D_m}$. For each $B_i \in occur(\{D_m\}, F)$, we check if $B_i \cap B \neq \emptyset$, and if it holds, we add $B_i$ to $occur(B, F)_{D_m}$. In the end, we compute the sum of the $directCount(B_i, F)$, for each $B_i \in occur(B, F)_{D_m}$ Finally, we subtract this sum from the value $|cov(\{D_m\}, F)|$ for computing the $|uniq(D_m, F, B)|$ (see Prop. 3). For a finite set of subset of datasets $BV = \{B_i, ..., B_m\}$, the time complexity is $O(|BV| * |occur(\{D_m\}, F)|)$, where in the worst case $|BV| = 2^{|\mathcal{D}|-1}$ (we exclude the subsets containing $D_m$), and the space complexity is $O(|occur(\{D_m\}, F)|)$. For the complexity of checking non-null intersection ($B_i \cap B \neq \emptyset$) see §4.3.

**Example.** In Fig. 5, the publisher of $D_2$ desires to know how unique are the triples of $D_2$ comparing to the other datasets. In Step A, we scan the whole *Entity-Triples* index for creating the *directCount* list, i.e., we focus on $F = RWT$. However, we keep only the subsets of the *directCount* list containing the dataset $D_2$, i.e., we create the set $occur(\{D_2\}, F)$. In Step B of Fig. 5, we compute the value $|cov(\{D_2\}, F)| = 6$, while in Step C, we construct the set $occur(B, F)_{D_2}$ and we compute $|uniq(D_2, F, B)|$ for each subset $B$. Finally, in Step D we show a visualization of the results. For example, for measuring the uniqueness of $D_2$ triples versus $\{D_1, D_3\}$, we created the set $occur(\{D_1, D_3\}, F)_{D_2} = \{\{D_1, D_2\}, \{D_2, D_3\}\}$, where the sum of the *directCount* score of the entries of $occur(\{D_1, D_3\}, F)_{D_2}$ equals 3. We subtracted that value from $|cov(\{D_2\}, F)|$ (which equals 6), for computing $|uniq(D_2, F, \{D_1, D_3\})| = 3$.

*4.5.2 Extra/Derived Uniqueness-related Metrics.* Here, we show two additional metrics related to uniqueness for a given dataset $D_m$.

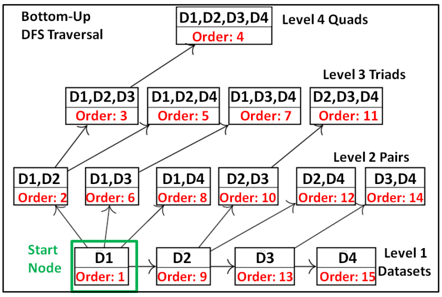


Fig. 6. The depth first search traversal that is followed

*Unique Content Percentage of a dataset $D_m$.* We define the uniqueness of a dataset $D_m$ for a given measurement type $F$, comparing to a subset of datasets $B$ as follows:

$$uniqPer(D_m, F, B) = \frac{|uniq(D_m, F, B)| * 100}{|cov(\{D_m\}, F)|} \tag{8}$$

It can be used for answering questions like "What it the percentage of the triples of DBpedia, that cannot be found in Wikipedia, YAGO, and Freebase?". For example, in Query $Q_{uniqueness}$of Fig. 5, for $B = \{D_1, D_3\}$ the $uniqPer(D_2, RWT, \{D_1, D_3\}) = 50\%$, since 3 out of 6 triples of $D_2$ cannot be found in any of the datasets of $B$, i.e., neither to $D_1$ nor to $D_3$.

*Unique Contribution of a Dataset $D_m$* We define the "unique contribution" percentage that a dataset $D_m$ offers to the "universe" for a measurement type $F$ as follows:

$$uniqCont(D_m, F, \mathcal{D}) = \frac{|uniq(D_m, F, \mathcal{D})| * 100}{|cov(\mathcal{D}, F)|} \tag{9}$$

It can be used for answering queries, such as "What percentage of triples of "Tiger" species is offered only by DBpedia?". For example, for the"Tiger" species of Fig. 2, dataset $D_3$ offers only one triple that cannot be found in any other dataset (i.e., the triple ⟨Tiger, foodSource, Ungulate⟩). In total, there are 8 available triples for "Tiger", thereby, $uniqCont(D_3, RWT_{\{E1\}}, \mathcal{D}) = 12.5\%$.

## 5 INCREMENTAL COMPUTATION OF UNION & COMPLEMENT METRICS

Since the possible subsets of datasets for solving the maximization problems (described in §3.1) can be exponential in number, it is very time-consuming (as we shall see experimentally in §6) to traverse the whole set $occur(\mathcal{D}, F)$ for each possible subset of datasets $B$, which is a requirement in the straightforward method (*SF*) of §4. For being able to solve such maximization problems, the target is to reduce the number of *directCount* entries that we read for each subset $B$. In this way, we propose incremental algorithms, based on set-theory properties and pruning methods, that reuse the measurements between two subsets of datasets $B$ and $B'$ (where $B \subset B'$). In particular, in §5.1, we propose an incremental algorithm for computing the coverage, in §5.2 we introduce pruning and regrouping methods, and in §5.3 we compare the mentioned approaches. Moreover, in §5.4 and §5.5, we show how to compute incrementally the information enrichment and uniqueness.

### 5.1 Lattice Based Incremental Algorithm for computing $covBest(K, F)$ (*LB* method).

**Rationale.** We present a lattice-based (*LB*) incremental algorithm, that can be used for solving $covBest(K, F)$. It reuses measurements between a subset $B$ and a superset $B'$ ($B \subset B'$), by following a depth-first search traversal and by exploiting set theory properties, as it is described below.

**Depth-First Search (DFS) Traversal.** Fig. 6 shows the exact visiting order for four datasets. For visiting each lattice node (subset of datasets) once, we create a directed edge from $B$ to $B'$, only if $B = prev(B')$. In particular, it holds that $B = prev(B')$, only if $B' = B \cup \{D_k\}$ ($D_k \notin B$), and

$\forall D_i \in B, k > i$, therefore, we follow a strict numerical ascending order. By following such a *DFS* traversal, we always go upwards (see Fig. 6), i.e., we compute first the metrics for all the supersets of $D_1$, then for all the supersets of $D_2$ that do not contain $D_1$ and so on, and we create $|E| = |V| - 1$ directed edges (instead of $|E| = |\mathcal{D}| * 2^{|\mathcal{D}|-1}$). Due to this traversal, we visit and compute the metrics for any $B \in \mathcal{P}(\mathcal{D})$ at run time, i.e., there is no need to pre-construct a data structure for the lattice, whereas it allows us to stop the computation at any desired level $L$, e.g., for computing the metrics for pairs of datasets, we can stop at level $L = 2$.

**How to Reuse the Measurements.** For reusing the measurements in an incremental way, we exploit the following set theory property:

Lemma 5.1. From the set theory, we know that for $n$ sets $\{A_1, A_2, ..., A_n\}$, $|A_1 \cup A_2 \cup ... \cup A_n| = |A_1 \cup A_2 \cup ... \cup A_{n-1}| + |A_n \setminus \{A_1 \cup A_2 \cup ... \cup A_{n-1}\}|$ *(Proof can be found in [10]).*

By exploiting the above property, for a subset $B$ and a superset $B'$ (where $B' = B \cup \{D_k\}$), we can add to $|cov(B', F)|$, the value $|cov(B, F)|$. The only requirement for computing $|cov(B', F)|$, is to find which are the *directCount* entries that contain the newly added dataset $D_k$, but not any $D_i \in B$, i.e., we should construct the set $occur(\{D_k\}, F)_{\setminus B}$, as it is stated below.

Prop. 4. If $B' = B \cup \{D_k\}$, then,

$$|cov(B', F)| = |cov(B, F)| + \sum_{B_i \in occur(\{D_k\}, F)_{\setminus B}} directCount(B_i, F) \tag{10}$$

Proof. The proof is given in Appendix A.4. □

Alg. 1 shows the exact steps for computing the coverage incrementally, while Fig. 7 depicts an example of Alg. 1, for the 5 “lattice nodes" in green color, of Query $Q_{coverage}$ (see Fig. 3) .

**Input.** The input is the set $occur(\mathcal{D}, F)$ and the corresponding $directCount(B_i, F)$ score, for each $B_i \in occur(\mathcal{D}, F)$, e.g., see the input in the lower right side of Fig. 7, and a parameter $L$, for stopping the computation of metrics, when level $L$ is reached, e.g., for finding $covBest(K, F)$, $L = K$ .

**Output.** It computes at query time the $|cov(B, F)|$ for all the subsets until a level $L$, e.g., if $L = |\mathcal{D}| + 1$ it computes the metrics for all the possible subsets $B \in \mathcal{P}(\mathcal{D})$. In particular, for finding the $covBest(K, F)$ it computes the metrics until the level $L = K$, and it keeps in memory the subset having the $arg_B \max\ |cov(B, F)|$.

**The Initial Phase.** We start from the bottom part of the lattice, thereby, we first explore the nodes containing a single dataset (lines 1-4 of Alg. 1), e.g., in Fig. 7, we start from the dataset $D_1$. For each single dataset the recursive function *covLB* is called, which takes as parameters the subset of datasets that will be visited, the dataset $D_k$ which was just added, the coverage of the previous visited node, i.e., $|cov(B, F)|$, the set $occur(\mathcal{D}, F)_{\setminus B}$, and the stop level $L$. For the case of single datasets, we suppose that the previous node is the empty set, therefore, $|cov(\{\emptyset\}, F)| = 0$, while $occur(\mathcal{D}, F)_{\setminus \{\emptyset\}}$ contains all the entries of the *directCount* list, i.e., $occur(\mathcal{D}, F)$ (lines 1-2).

**Computation of Measurements for the Current Node.** The target is to construct the set $occur(\{D_k\}, F)_{\setminus B}$ (see Lemma 5.1). For this reason, we read the entries of the input set $occur(\mathcal{D}, F)_{\setminus B}$, and $\forall\ B_i \in occur(\mathcal{D}, F)_{\setminus B}$ we check whether $D_k \in B_i$ (see lines 6-12 of Alg. 1). When $D_k \in B_i$, we add $B_i$ to $occur(\{D_k\}, F)_{\setminus B}$. On the contrary, if $D_k \notin B_i$, we store $B_i$ in another set, i.e., $occur(\mathcal{D}, F)_{\setminus B'}$, since $B_i$ should be scanned again when we reach the supersets of $B'$. Therefore, we divide the input set $occur(\mathcal{D}, F)_{\setminus B}$ in two disjoint sets, i.e., $occur(\mathcal{D}, F)_{\setminus B} = occur(\{D_k\}, F)_{\setminus B} \cup occur(\mathcal{D}, F)_{\setminus B'}$. Finally, Alg. 1 computes $|cov(B', F)|$ by taking the sum of $|cov(B, F)|$ and the sum of the *directCount* score of $occur(\{D_k\}, F)_{\setminus B}$, and prints (or stores) the result (see lines 13-14).

In Fig. 7, we first visit the dataset $D_1$, and we iterate over the set $occur(\mathcal{D}, F)$. Then, we create the sets $occur(\mathcal{D}, F)_{\setminus \{D_1\}} = \{\{D_2, D_3\}, \{D_2, D_4\}, \{D_3\}, \{D_4\}\}$ and $occur(\{D_1\}, F)_{\setminus B} = \{\{D_1\}, \{D_1, D_2\}\}$.

**ALGORITHM 1:** Computing $|cov(B,F)|$ incrementally for any possible subset of datasets $B$

```
Input: The directCount list including the set $occur(\mathcal{D},F)$ and the directCount values, the stop Level $L$
Output: The coverage $|cov(B,F)|$, for each possible subset $B \in \mathcal{P}(\mathcal{D})$
1  $occur(\mathcal{D},F)_{\setminus\{\emptyset\}} \leftarrow occur(\mathcal{D},F)$        // The starting node is the empty set
2  $|cov(\{\emptyset\},F)| = 0$
3  forall $D_k \in \mathcal{D}$ do                                    // Visit each single dataset once
4      $covLB(\{D_k\}, D_k, |cov(\{\emptyset\},F)|, occur(\mathcal{D},F)_{\setminus\{\emptyset\}}, L)$

   // The recursive function for computing the metrics incrementally
5  function $covLB(Subset\ B', Dataset\ D_k, |cov(B,F)|, occur(\mathcal{D},F)_{\setminus B}, level\ L\ )$      // Visit $B'$
6      $occur(\{D_k\},F)_{\setminus B} \leftarrow \emptyset$
7      $occur(\mathcal{D},F)_{\setminus B'} \leftarrow \emptyset$
8      forall $B_i \in occur(\mathcal{D},F)_{\setminus B}$ do        // Divide $occur(\mathcal{D},F)_{\setminus B}$ into two disjoint sets
9          if $D_k \in B_i$ then
10             $occur(\{D_k\},F)_{\setminus B} \leftarrow occur(\{D_k\},F)_{\setminus B} \cup \{B_i\}$
11         else if $D_k \notin B_i$ then
12             $occur(\mathcal{D},F)_{\setminus B'} \leftarrow occur(\mathcal{D},F)_{\setminus B'} \cup \{B_i\}$
13     $|cov(B',F)| \leftarrow |cov(B,F)| + \sum_{B_i \in occur(\{D_k\},F)_{\setminus B}} directCount(B_i,F)$
14     print $B'$, $|cov(B',F)|$
15     if $|B'| \leq L$ then                                    // Continue until the stop level
16         forall $D_j \in \mathcal{D}, j > k$ do            // A strict numerical dfs order is followed
17             $B'' = B' \cup \{D_j\}$                     // Add one more dataset to $B'(B' = prev(B''))$
18             $covLB(B'', D_j, |cov(B',F)|, occur(\mathcal{D},F)_{\setminus B'}, L)$                 // Visit $B''$
```

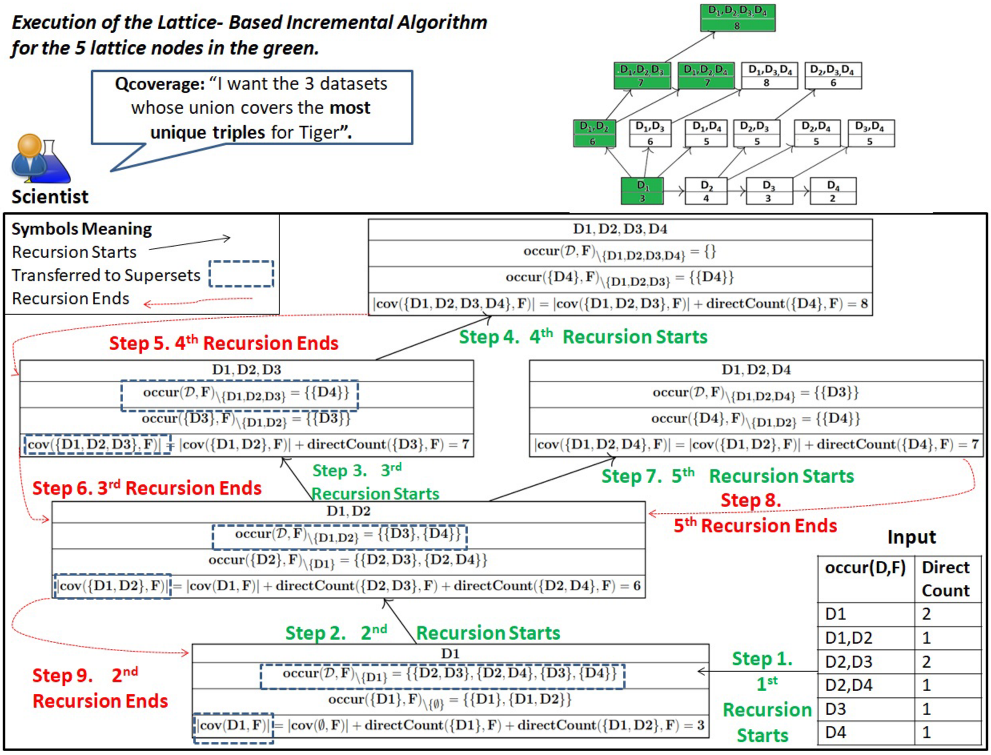
Finally, we sum the *directCount* score of $occur(\{D_1\},F)_{\setminus B}$ entries (which equals 3) and the $|cov(B,F)|$ of the previous subset, which is the empty set ($|cov(\{\emptyset\},F)| = 0$), for computing $|cov(\{D_1\},F)| = 3$.

**How to explore the Supersets of the Current Node.** By following the *dfs* traversal in a recursive way, we visit a superset $B''$ from a subset $B'$ (lines 16-18), e.g., see Fig. 7. Indeed, when we visit $B''$, we add a parameter for "highlighting" the dataset $D_j$ that was last added, we transfer the value $|cov(B',F)|$ and the set $occur(\mathcal{D},F)_{\setminus B'}$. Thereby, a new recursive call starts, where we perform exactly the same steps. It is obvious that since $occur(\mathcal{D},F)_{\setminus B'} \subseteq occur(\mathcal{D},F)_{\setminus B}$, the size of the input (which is scanned in lines 8-12) decreases as we continue upwards.

In Step 2 of Fig. 7, we "transferred" from $D_1$ to $\{D_1,D_2\}$ the value $|cov(\{D_1\},F)|$ and the set $occur(\mathcal{D},F)_{\setminus\{D_1\}}$. In the second recursion, we iterated over the set $occur(\mathcal{D},F)_{\setminus\{D_1\}}$ once, and we constructed the sets $occur(\{D_2\},F)_{\setminus\{D_1\}} = \{\{D_2,D_3\},\{D_2,D_4\}\}$ and $occur(\mathcal{D},F)_{\setminus\{D_1,D_2\}} = \{\{D_3\},\{D_4\}\}$. Finally, we took the sum of the *directCount* score of $occur(\{D_2\},F)_{\setminus\{D_1\}}$ entries (i.e., equals 3), and of $|cov(\{D_1\},F)|$ (i.e., 3), for computing $|cov(\{D_1,D_2\},F)| = 6$. Finally, the set $occur(\mathcal{D},F)_{\setminus\{D_1,D_2\}}$ will be transferred to the supersets of $\{D_1,D_2\}$, e.g., in Steps 3 and 7 of Fig. 7.

**When the Execution of the Algorithm Finishes.** We always continue upwards, until there are no other supersets to explore (e.g., in Step 4 of Fig. 7 we reached the top of the lattice), or we have reached the desired level $L$ (line 15 in Alg. 1), e.g., for finding the triad having $covBest(3,F)$, we stop at level $L = K = 3$. In any of the previous cases, the recursion ends, thereby, we return to the previous node and we check if there are other supersets that have not been visited, e.g., in Step 6 of Fig. 7, when we return to the node $\{D_1,D_2\}$, we visit also its superset $\{D_1,D_2,D_4\}$.

**LB versus SF.** For computing $|cov(B',F)|$ for a subset $B'$, we reuse $|cov(B,F)|$ and we scan the set $occur(\mathcal{D},F)_{\setminus B}$ instead of $occur(\mathcal{D},F)$ (in *SF* approach), where $occur(\mathcal{D},F)_{\setminus B} \subseteq occur(\mathcal{D},F)$.


Execution of the Lattice- Based Incremental Algorithm for the 5 lattice nodes in the green.
Qcoverage: "I want the 3 datasets whose union covers the most unique triples for Tiger".
Scientist
D1,D2,D3,D4 8
D1,D2,D3 7
D1,D2,D4 7
D1,D3,D4 8
D2,D3,D4 6
D1,D2 6
D1,D3 6
D1,D4 5
D2,D3 5
D2,D4 5
D3,D4 5
D1 3
D2 4
D3 3
D4 2
Symbols Meaning
Recursion Starts
Transferred to Supersets
Recursion Ends
D1, D2, D3, D4
occur(𝒟, F)\{D1,D2,D3,D4} = {}
occur({D4}, F)\{D1,D2,D3} = {{D4}}
|cov({D1, D2, D3, D4}, F)| = |cov({D1, D2, D3}, F)| + directCount({D4}, F) = 8
Step 4. 4th Recursion Starts
Step 5. 4th Recursion Ends
D1, D2, D3
occur(𝒟, F)\{D1,D2,D3} = {{D4}}
occur({D3}, F)\{D1,D2} = {{D3}}
|cov({D1, D2, D3}, F)| = |cov({D1, D2}, F)| + directCount({D3}, F) = 7
D1, D2, D4
occur(𝒟, F)\{D1,D2,D4} = {{D3}}
occur({D4}, F)\{D1,D2} = {{D4}}
|cov({D1, D2, D4}, F)| = |cov({D1, D2}, F)| + directCount({D4}, F) = 7
Step 3. 3rd Recursion Starts
Step 6. 3rd Recursion Ends
Step 7. 5th Recursion Starts
Step 8. 5th Recursion Ends
D1, D2
occur(𝒟, F)\{D1,D2} = {{D3}, {D4}}
occur({D2}, F)\{D1} = {{D2, D3}, {D2, D4}}
|cov({D1, D2}, F)| = |cov(D1, F)| + directCount({D2, D3}, F) + directCount({D2, D4}, F) = 6
Step 2. 2nd Recursion Starts
Step 9. 2nd Recursion Ends
D1
occur(𝒟, F)\{D1} = {{D2, D3}, {D2, D4}, {D3}, {D4}}
occur({D1}, F)\{∅} = {{D1}, {D1, D2}}
|cov(D1, F)| = |cov(∅, F)| + directCount({D1}, F) + directCount({D1, D2}, F) = 3
Step 1. 1st Recursion Starts
Input

| occur(D,F) | Direct Count |
|---|---|
| D1 | 2 |
| D1,D2 | 1 |
| D2,D3 | 2 |
| D2,D4 | 1 |
| D3 | 1 |
| D4 | 1 |

Fig. 7. Execution of Incremental Algorithm for the 5 lattice nodes in green color

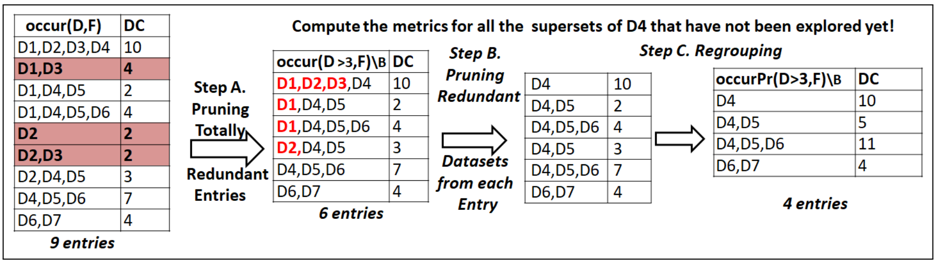
**Time & Space Complexity.** For each visited subset $B'$, we iterate over $|occur(\mathcal{D}, F)_{\setminus B}|$ entries (see line 8 of Alg. 1). Therefore, for a given set of subsets $BV$ ($BV = \{B_i, ..., B_n\}$), the average number of such iterations per subset $B'$ is: $avgIt_{LB}(BV) = \frac{\sum_{B' \in BV} |occur(\mathcal{D},F)_{\setminus B}|, prev(B')=B}{|BV|}$. The time complexity is $O(|BV| * avgIt_{LB}(BV))$, which is exponential in number when $|BV| = 2^{|\mathcal{D}|}$. Concerning space complexity, it is $O(|BV_d| * avgIt_{LB}(BV_d))$ where $BV_d = \{B_i, ..., B_n\}$ are the subsets that occur in the maximum depth $d$ of lattice (since $d$ is at most $|\mathcal{D}| + 1$, then $|BV_d| \leq |\mathcal{D}| + 1$).

## 5.2 Pruning & Regrouping the *directCount* List for computing $covBest(K, F)$

Here, the target is to decrease the size of the set $occur(\mathcal{D}, F)_{\setminus B}$, which is transferred in line 18 of Alg. 1 from a subset $B$ to a superset $B'$ ($B = prev(B')$), and is scanned in lines 8-12 of Alg. 1 when $B'$ is visited, for solving $covBest(K, F)$ by performing less set operations in the aforementioned lines. For this reason, we propose in §5.2.1 a process for pruning the totally redundant entries, and in §5.2.2 a process for pruning and regrouping all the entries of the *directCount* list.

*5.2.1 Pruning Totally Redundant Entries (LB+TPR).* First, we provide some definitions. Let $maxID(B)$ be the largest dataset ID in a subset of datasets $B$, i.e., $maxID(B) = \{k \mid D_k \in B \text{ and } \forall D_j \in B, k \geq j\}$ (e.g., $maxID(\{D_1, D_3, D_4\}) = 4$). We define as $D_{\leq k} = \{D_i \in \mathcal{D} \mid i \leq k, D_k \in \mathcal{D}\}$ all datasets whose ID is equal or smaller than the ID of $D_k$ (e.g., in Fig. 8, $D_{\leq 4} = \{D_1, D_2, D_3, D_4\}$. Finally, let $D_{>k} = \mathcal{D} \setminus D_{\leq k}$ be all datasets whose ID is larger than the ID of $D_k$ (e.g., in Fig. 8, $D_{>4} = \{D_5, D_6, D_7\}$).

**LB+TPR Approach.** The target is to remove the redundant entries from the set $occur(\mathcal{D}, F)_{\setminus B'}$, which is created in line 12 of Alg. 1. By using the *dfs* traversal of Fig. 6 (and the numerical ascending order), we visit a superset $B'$, where $B' = B \cup \{D_k\}$ and $k = maxID(B')$. Due to this traversal, in


Compute the metrics for all the supersets of D4 that have not been explored yet!
Step A. Pruning Totally Redundant Entries
Step B. Pruning Redundant Datasets from each Entry
Step C. Regrouping
9 entries
6 entries
4 entries

| occur(D,F) | DC |
|---|---|
| D1,D2,D3,D4 | 10 |
| **D1,D3** | **4** |
| D1,D4,D5 | 2 |
| D1,D4,D5,D6 | 4 |
| **D2** | **2** |
| **D2,D3** | **2** |
| D2,D4,D5 | 3 |
| D4,D5,D6 | 7 |
| D6,D7 | 4 |

| occur(D >3,F)\B | DC |
|---|---|
| **D1,D2,D3**,D4 | 10 |
| **D1**,D4,D5 | 2 |
| **D1**,D4,D5,D6 | 4 |
| **D2,**D4,D5 | 3 |
| D4,D5,D6 | 7 |
| D6,D7 | 4 |

| | |
|---|---|
| D4 | 10 |
| D4,D5 | 2 |
| D4,D5,D6 | 4 |
| D4,D5 | 3 |
| D4,D5,D6 | 7 |
| D6,D7 | 4 |

| occurPr(D>3,F)\B | DC |
|---|---|
| D4 | 10 |
| D4,D5 | 5 |
| D4,D5,D6 | 11 |
| D6,D7 | 4 |

Fig. 8. Pruning Example-Computing the metrics for all the supersets of $D_4$, that have not been explored yet

lines 11-12 of Alg. 1, it is redundant to add $B_i$ to $occur(\mathcal{D}, F)_{\setminus B'}$ if $maxID(B_i) < k$, i.e., in this case, all the IDs of $B_i$ datasets are smaller than $k$. Indeed, for such a $B_i$ and for any $B''$ (where $B'' \supset B'$) it holds that $B_i \cap B'' = \emptyset$, which means that $B_i$ will be "transferred" forever, as we go upwards (i.e., it will never pass the check in line 9). Therefore, in line 11 we check if $maxID(B_i) > k$ and only in case it holds, in line 12 we add $B_i$ to the set $occur(D_{>k}, F)_{\setminus B'}$ (instead of $occur(\mathcal{D}, F)_{\setminus B'}$), and then we transfer the set $occur(D_{>k}, F)_{\setminus B'}$ to the supersets of $B'$ (in line 18).

**Example.** In Fig. 8, we desire to compute the metrics for all the supersets of $D_4$ that have not visited yet, i.e., those that do not contain $D_1$, $D_2$ or $D_3$. Due to *dfs* traversal, we have already computed the measurements for any combination containing $D_1$, $D_2$ or $D_3$. Since each time we will add datasets belonging to $D_{>3} = \{D_4, D_5, D_6, D_7\}$, in Step A of Fig. 8 we prune the totally redundant entries, containing only datasets belonging to $D_{\leq 3}$, but not any $D_i \in D_{>3}$. In this way, we use 6 *directCount* entries instead of 9, i.e., we removed the entries $\{D_1, D_3\}$, $\{D_2\}$ and $\{D_2, D_3\}$.

**LB+TPR versus LB.** Since $D_{>k} \subseteq \mathcal{D}$, for any subset $B$ it holds that $occur(D_{>k}, F)_{\setminus B} \subseteq occur(\mathcal{D}, F)_{\setminus B}$, therefore, we transfer (line 18 of Alg. 1) and iterate (line 8 of Alg. 1) over smaller (or in the worst case the same) number of entries comparing to *LB* approach of §5.1 (see time and space complexity of *LB+TPR* in Table 3). However, we should perform an extra check (i.e., $maxID(B_i) > k$).

*5.2.2 Pruning & Regrouping (LB+PRGR).* The target is to further reduce the size of $occur(D_{>k}, F)_{\setminus B'}$, which is used for computing $|cov(B, F)|$ for the supersets of $B'$. In this way, we propose an approach, where we remove both the totally redundant *directCount* entries and the redundant datasets from each remaining *directCount* entry, and we perform a regrouping. In particular, for a specific $B'$ (where $maxID(B') = k$), we replace each $B_i \in occur(D_{>k}, F)_{\setminus B'}$ (constructed in lines 8-12 by using *LB+TPR*) with $B_i^k$, where $B_i^k = B_i \setminus D_{\leq k}$, and then we regroup the "pruned" entries, i.e., we group the same entries and we sum their *directCount* score. In this way, a new *directCount* list is created in each lattice node. This list contains in its left side the set $occurPr(D_{>k}, F)_{\setminus B'} = \bigcup_{B_i \in occur(D_{>k}, F)_{\setminus B'}} B_i \setminus D_{\leq k}$ (where $k = maxID(B')$), and in its right side the *directCount* score of each $B_i^k$, i.e., $directCountPr(B_i^k, F, B') = \sum_{B_i \in occur(D_{>k}, F)_{\setminus B'}, B_i^k = B_i \setminus D_{\leq k}} directCount(B_i, F)$.

**Example.** In Step B of Fig. 8, we prune in each entry the datasets belonging to $D_{\leq 3} = \{D_1, D_2, D_3\}$, i.e., we have already visited all the nodes containing any of these datasets. Due to pruning, some entries exist multiple times, e.g., $\{D_4, D_5\}$ exists twice, with *directCount* scores 2 and 3. For this reason, in Step C, we regroup the entries, e.g., we keep $\{D_4, D_5\}$ once, with a new *directCount* score, i.e., 5. Due to this process, we use only 4 *directCount* entries, instead of 6 (by using *LB+TPR*). Certainly, as we continue upwards, we create (and transfer) a new smaller *directCount* list.

**LB+PRGR versus LB+TPR.** It is obvious that $|occurPr(D_{>k}, F)_{\setminus B'}| \leq |occur(D_{>k}, F)_{\setminus B'}|$, therefore, we read and transfer smaller number of entries comparing to *LB+PRGR* in lines 8 and 18 of Alg. 1. However, we should perform additional operations for creating the set $occurPr(B', F)$ and the $directCountPr(B_i^k, F, B')$ of each $B_i^k$. Table 3 presents the time and space complexity of *LB+PRGR*.

| Approach | $avgIt$ per $B'$ in line 8 of Alg. 1 ($B = prev(B')$) | Time Complexity | Space Complexity |
|---|---|---|---|
| *SF* | $\|occur(\mathcal{D}, F)\|$ | $O(\|BV\| * \|occur(\mathcal{D}, F)\|)$ | $O(\|occur(\mathcal{D}, F)\|)$ |
| *LB* | $avgIt_{LB}(BV) = \frac{\sum_{B' \in BV} \|occur(\mathcal{D},F)_{\setminus B}\|}{\|BV\|}$ | $O(\|BV\| * avgIt_{LB}(BV))$ | $O(\|BV_d\| * avgIt_{LB}(BV_d))$ |
| *LB+TPR* | $avgIt_{LB+TPR}(BV) \frac{\sum_{B' \in BV} \|occur(D_{>k},F)_{\setminus B}\|}{\|BV\|}$ | $O(\|BV\| * avgIt_{LB+TPR}(BV))$ | $O(\|BV_d\| * avgIt_{LB+TPR}(BV_d))$ |
| *LB+PRGR* | $avgIt_{LB+PRGR}(BV) = \frac{\sum_{B' \in BV} \|occurPr(D_{>k},F)_{\setminus B}\|}{\|BV\|}$ | $O(\|BV\| * avgIt_{LB+PRGR}(BV))$ | $O(\|BV_d\| * avgIt_{LB+PRGR}(BV_d))$ |

Table 3. Comparison of different approaches for computing $|cov(B', F)|$, for a set of “visited" subsets $BV$.

### 5.3 Executive Summary

Table 3 compares the presented approaches in terms of the average iterations per subset $B$ for computing $|cov(B, F)|$, and of time and space complexity. As we have seen in this section, for a specific subset $B$ where $k = maxID(B)$, it holds that $|occurPr(D_{>k}, F)_{\setminus B}| \leq |occur(D_{>k}, F)_{\setminus B}| \leq |occur(\mathcal{D}, F)_{\setminus B}| \leq |occur(\mathcal{D}, F)|$. Therefore, for any given subset $B$, the number of iterations required for computing $|cov(B, F)|$ (and thus the number of set operations in lines 8-12 of Alg. 1), and the size of the set which is transferred to supersets (line 18 of Alg. 1), is reduced as we go downwards in Table 3. In this way, (in lines 8-12) less operations are required for finding the $covBest(K, F)$. However, we should note that for the case of *LB+TPR* we need to perform an extra check comparing to *LB*, i.e., $maxID(B_i) > k$. By using *LB+PRGR*, we perform the previous check, and we also replace each $B_i \in |occur(D_{>k}, F)_{\setminus B}|$, with $B_i^k = B_i \setminus D_{\leq k}$, and to compute the *directCount* score of $B_i^k$. However, as we shall see experimentally in §6, despite these extra operations, the two latter approaches (mainly *LB+PRGR*) are faster comparing to the other ones.

### 5.4 Lattice-Based Incremental Algorithm for computing $enrichBest(K, F, D_m)$

We use a variation of the incremental algorithm (i.e., Alg. 1) for *coverage*. The key difference is that we do not take into account the posting lists and lattice nodes containing $D_m$.

**Input.** We give as input the $occur(\mathcal{D}, F)_{\setminus\{D_m\}}$ instead of $occur(\mathcal{D}, F)$, and the lattice level $L$, e.g., for stopping the computation when $L = K$.

**Output.** It computes at query time the $|enrich(B, F, D_m)|$ for all the subsets of datasets (that do not contain $D_m$) until a level $L$. For finding the $enrichBest(K, F, D_m)$ it computes the metrics until the level $L = K$, and it keeps in memory the subset having the $arg_B \max\ |enrich(B, F, D_m)|$.

**Differences with Alg. 1.** Below, we analyze the differences comparing to Alg. 1.

- We transfer from a subset $B$ to a superset $B'$ ($B' = B \cup \{D_k\}$) the value $|enrich(B, F, D_m)|$ and the set $occur(\mathcal{D}, F)_{\setminus\{B \cup \{D_m\}\}}$ (instead of $|cov(B, F)|$ and $occur(\mathcal{D}, F)_{\setminus B}$).
- In each recursive call, we read the set $occur(\mathcal{D}, F)_{\setminus\{B \cup \{D_m\}\}}$ , and we divide that set into two disjoint sets, i.e., $occur(\{D_k\}, F)_{\setminus\{B \cup \{D_m\}\}}$ and $occur(\mathcal{D}, F)_{\setminus\{B' \cup \{D_m\}\}}$.
- We exploit the set theory property described in Lemma 5.1, for computing $|enrich(B', F, D_m)|$ as follows, $|enrich(B', F, D_m)| = |enrich(B, F, D_m)| + \sum_{B_i \in occur(\{D_k\}, F)_{\setminus\{B \cup \{D_m\}\}}} directCount(B_i, F)$.

**Time & Space Complexity.** Given a set of subsets $BV$ ($BV = \{B_i, ..., B_n\}$), the average number of iterations per $B' \in BV$ is: $avgIt_{enrichLB}(BV) = \frac{\sum_{B' \in BV} |occur(\mathcal{D}, F)_{\setminus\{B \cup \{D_m\}\}}|, B = prev(B')}{|BV|}$. The time complexity is $O(|BV| * avgIt_{enrichLB}(BV))$ (where $|BV| \leq 2^{|\mathcal{D}|-1}$), and the space complexity is $O(|BV_d| * avgIt_{enrichLB}(BV_d))$ where $BV_d = \{B_i, ..., B_n\}$ ($BV_d \leq |\mathcal{D}|$). For reducing the number of iterations, one can use the pruning and regrouping mechanisms of §5.2.

### 5.5 Lattice-Based Incremental Algorithm for computing $uniqBest(D_m, F, K)$

We use a variation of the incremental algorithm (i.e., Alg. 1) which is used for *coverage*. The key difference is that we compute the union of the elements of the datasets in $B$ that also occur in $D_m$.

**Input.** We give as input the $occur(\{D_m\}, F)$ instead of $occur(\mathcal{D}, F)$, and the level $L$, i.e., the level where we stop the computation.

Table 4. Statistics of indexes for 400 datasets, and for the 25 most popular datasets of each index

| Index for | Index Size | $\lvert occur(\mathcal{D}, F)\rvert$ |
|---|---|---|
| Entities ($\lvert\mathcal{D}\rvert = 400$) | 412 million | 23,357 |
| Literals ($\lvert\mathcal{D}\rvert = 400$) | 429 million | 336,570 |
| Triples ($\lvert\mathcal{D}\rvert = 400$) | 2 billion | 13,772 |
| Entities ($\lvert\mathcal{D}\rvert = 25$) | 359 million | 11,139 |
| Literals ($\lvert\mathcal{D}\rvert = 25$) | 403 million | 64,907 |
| Triples ($\lvert\mathcal{D}\rvert = 25$) | 1.6 billion | 5,250 |

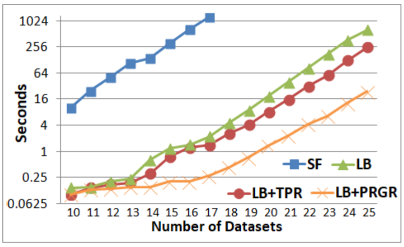


Fig. 9. Execution time of $|cov(B, RWE)|$ of Entities Index among any combination of 10-25 datasets

**Output.** It computes at query time the $|uniq(D_m, F, B)|$ for all the subsets of datasets (that do not contain $D_m$) until a level $L$. For finding the $uniqBest(D_m, F, K)$ it computes the metrics until the level $L = K$, and it keeps in memory the subset having the $arg_B$ max $|uniq(D_m, F, B)|$.

**Differences with Alg. 1.** Below, we analyze the differences comparing to Alg. 1.

- We initialize $|uniq(D_m, F, \{\emptyset\})| = |cov(\{D_m\}, F)|$ (all the elements of $D_m$ are unique comparing to the empty set), and we transfer from a subset $B$ to a superset $B'$ ($B' = B \cup \{D_k\}$) the value $|uniq(D_m, F, B)|$ and the set $occur(\{D_m\}, F)_{\setminus B}$ (instead of $|cov(B, F)|$ and $occur(\mathcal{D}, F)_{\setminus B}$).
- In each recursive call, we read $occur(\{D_m\}, F)_{\setminus B}$, and we divide that set into two disjoint sets, i.e., $occur(\{D_m\}, F)_{\setminus B'}$ and $occur(\{D_k\}, F)_{D_m, \setminus B}$, where the set $occur(\{D_k\}, F)_{D_m, \setminus B}$ is defined as $occur(\{D_k\}, F)_{D_m, \setminus B} = occur(\{D_k\}, F)_{D_m} \setminus occur(B, F)_{D_m}$.
- By exploiting the set theory property of Lemma 5.1, we compute the $|uniq(D_m, F, B')|$ as follows: $|uniq(D_m, F, B')| = |uniq(D_m, F, B)| - \sum_{B_i \in occur(\{D_k\}, F)_{D_m, \setminus B}} directCount(B_i, F)$.

**Time & Space Complexity.** Given a set of subsets $BV$ ($BV = \{B_i, ..., B_n\}$), the average number of iterations per $B' \in BV$ is: $avgIt_{uniqLB}(BV) = \frac{\sum_{B' \in BV} |occur(\{D_m\}, F)_{\setminus B}|, B = prev(B')}{|BV|}$. The time complexity is $O(|BV| * avgIt_{uniqLB}(BV))$ (where $|BV| \leq 2^{|\mathcal{D}|-1}$), and the space complexity is $O(|BV_d| * avgIt_{uniqLB}(BV_d))$ where $BV_d = \{B_i, ..., B_n\}$ ($|BV_d| \leq |\mathcal{D}|$). One can use the pruning and regrouping methods of §5.2, for reducing the number of iterations.

## 6 EXPERIMENTAL EVALUATION

In [17], we collected 400 RDF datasets from 9 domains, containing over 2 billion triples, and we created indexes for these 400 datasets in 81.5 minutes by using 96 machines. In this paper, in §6.1, we exploit the three largest of the aforementioned constructed indexes, i.e., *Entity*, *Literals* and *Entity-Triples* index, for evaluating the efficiency of the proposed methods by using a single machine, while in §6.2 we introduce indicative measurements about these 400 RDF Datasets.

### 6.1 Efficiency

**Datasets.** Table 4 contains statistics for the three indexes that we use in our experiments (see the first 3 rows). For each index, we provide efficiency measurements for the 25 most popular datasets (see the last 3 rows). We observe that in all the cases, the size of $|occur(\mathcal{D}, F)|$ (size of each *directCount* list) is extremely smaller than the number of total entries of each index.

**Models.** We compare the efficiency of four different models: a) a straightforward approach, i.e., *SF* (see §4), b) the lattice-based incremental approach, i.e., *LB* (see §5.1, §5.4 and §5.5), c) the lattice-based incremental approach by pruning the totally redundant entries, i.e., *LB+TPR* (see §5.2.1), and d) the lattice-based incremental approach with pruning and regrouping, i.e., *LB+PRGR* (see §5.2.2).

**The Selected Performance Metrics.** Here, we evaluate the speedup obtained by the proposed methods for computing the metrics for all the possible subsets of datasets. Certainly, for finding the $covBest(K, F)$ for a given query, e.g., "I desire to find the triad of datasets containing the most triples for entity $e$", we compute the metrics only for a small part of the lattice, i.e., we stop at level

Table 5. Statistics for measuring $|cov(B, F)|$ for each index incrementally, for $2^{20}$ subsets

| Category | LB | LB+TPR | LB+PRGR |
|---|---|---|---|
| $avgIt$ (Entities) | 110 | 44 | 6.1 |
| $avgIt$ (Triples) | 191 | 87 | 6.6 |
| $avgIt$ (Literals) | 473 | 263 | 12.9 |
| Exec. Time (Entities) | 18s | 9s | 1.3s |
| Exec. Time (Triples) | 55s | 19.8s | 1.4s |
| Exec. Time (Literals) | 151s | 64s | 3.2s |

Table 6. Max Speedup by using different models for measuring $|cov(B, F)|$, for each index

| Max Speedup of | Entities | Triples | Literals |
|---|---|---|---|
| LB vs SF | 565× | 136× | 1,099× |
| LB+TPR vs SF | 891× | 214× | 1,459× |
| LB+PRGR vs SF | 5,773× | 2,284× | 6,000× |
| LB+TPR vs LB | 3× | 5× | 2.37× |
| LB+PRGR vs LB | 28× | 68× | 97× |
| LB+PRGR vs LB+TPR | 10× | 14× | 33× |

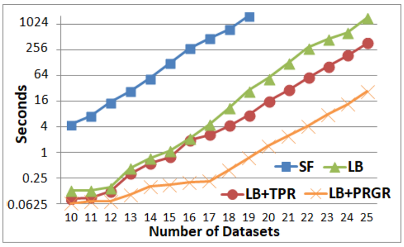


Fig. 10. Execution time of $|cov(B, RWT)|$ of Triples Index among any combination of 10-25 datasets ($2^{10} - 2^{25}$ subsets $B$)

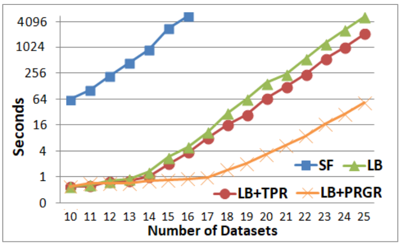


Fig. 11. Execution time of $|cov(B, LIT)|$ of Literals Index among any combination of 10-25 datasets ($2^{10} - 2^{25}$ subsets $B$)

$L = K = 3$. However, the objective is to evaluate the worst case, i.e., how fast are the proposed methods when the metrics for all the possible subsets of datasets should be computed. Indeed, we measure the execution time for $10 - 25$ datasets (i.e., the 25 most popular datasets in each index) by using pre-constructed *directCount* lists, i.e., we compute the metrics for 1,024 ($2^{10}$) to 33 million ($2^{25}$) subsets of datasets. Moreover, we compute the average number of entries that are transferred (in line 18 of Alg. 1) from a subset $B$ to its superset $B'$ ($B = prev(B')$) and scanned (in line 8 of Alg. 1) for computing the metrics of $B'$ (see a comparison of the approaches in §5.3).

**Hardware Setup & Code.** For all the experiments presented in this paper, we used a single machine having an *i5* core, 8 GB main memory and 1TB disk space. One can have access to all the datasets and the code (in JAVA programming language), in http://www.ics.forth.gr/isl/LODsyndesis.

*6.1.1 Efficiency of Coverage.* Figures 9, 10 and 11 show experiments for computing $|cov(B, F)|$, for all the three indexes. It is obvious that the incremental approaches are extremely faster comparing to the straightforward *SF* approach for all the three cases. More specifically, in the first three rows of Table 6 we show the maximum speedup obtained in the experiments of Figures 9, 10 and 11, by using each of the incremental approaches versus the *SF* one. For computing the coverage for literals, the *LB* was even $1,099\times$ faster than *SF*, while the *LB+TPR* and the *LB+PRGR* approaches were up to $1,459\times$ and $6,000\times$ faster, respectively. Indicatively, *SF* needs more than 10 minutes for computing the coverage of entities for 65,536 ($2^{16}$) subsets of datasets, while by using the best incremental approach (*LB+PRGR*), we computed the coverage, for millions of subsets (i.e., $2^{20} subsets$), for entities in 1.3 seconds, for triples in 1.4 seconds and for literals in 3.2 seconds.

**The Gain of Removing the Totally Redundant Entries.** In Figures 9, 10 and 11, we can see a clear speedup by using *LB+TPR* instead of *LB*, especially as the number of datasets grows. In particular, we observed up to 3×, 5× and 2.37× speedup comparing to *LB* (see Table 6) for entities, triples and literals, respectively. Indicatively, *LB* needs 55 seconds for computing the union of triples for $2^{20}$ subsets of datasets, while by using *LB+TPR* only 19.8 seconds were needed. This difference is rational according to the analysis of §5.3, and to the statistics presented in Table 5. Indeed, for each different index, we transfer to a superset $B'$ and iterate over smaller *directCount* entries for computing $|cov(B', F)|$, by using *LB+TPR* instead of *LB*. In particular, we performed on

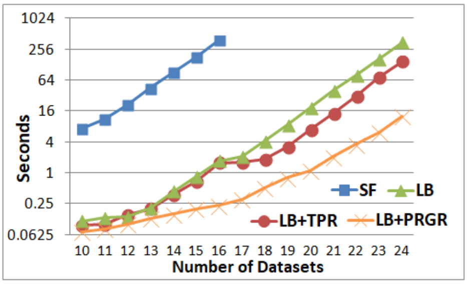


Fig. 12. Execution time of $|enrich(B, RWE, D_m)|$ among any combination of 10-24 datasets

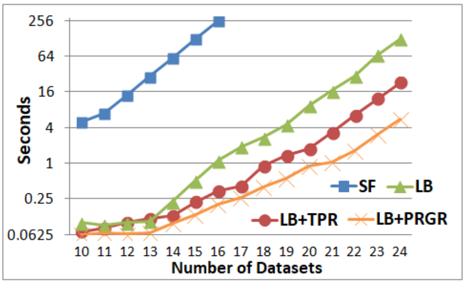


Fig. 13. Execution time of $|uniq(D_m, RWE, B)|$ among any combination of 10-24 datasets

average 44 iterations (in line 8 of Alg. 1) by using *LB+TPR* versus 110 by using *LB* for entities, 87 iterations versus 191 for triples, and 263 iterations versus 473 for literals.

**The Gain of Pruning and Regrouping.** Figures 9-11 show a clear speedup due to the pruning and regrouping approach (*LB+PRGR*). In particular, we identified even 33× speedup comparing to *LB+TPR* and 97× speedup comparing to *LB* (i.e., see Table 6). The difference between all the other models is big, especially as more datasets are added. This difference can be explained by measuring the number of iterations for computing the metric for a superset $B'$ (and by the analysis of §5.3). For instance, for computing the metrics for $2^{20}$ subsets of datasets, on average we iterate over only 6.1 *directCount* entries for entities, 6.6 for triples and 12.9 for literals, while even for the second best approach, the corresponding numbers are far higher (see Table 5). Indeed, for the case of literals, we performed on average 20× more iterations by using *LB+TPR* instead of *LB+PRGR*. Finally, *LB+PRGR* can compute the coverage **even for billions of subsets of datasets** ($2^{30}$) approximately in 12 minutes for entities, 14 minutes for triples and 22 minutes for literals.

*6.1.2 Efficiency of Information Enrichment and Uniqueness.* For computing $|enrich(B, F, D_m)|$ and $|uniq(D_m, F, B)|$, the algorithms are similar to $|cov(B, F)|$. The main difference is that for the $|enrich(B, F, D_m)|$, we use as input the set $occur(\mathcal{D}, F)_{\setminus\{D_m\}}$, and for the $|uniq(D_m, F, B)|$ the set $occur(\{D_m\}, F)$. Here, we show an indicative experiment for each case by using the *Entity Index*. We have selected a specific dataset $D_m$ whose $|occur(\{D_m\}, RWE)| = 4,031$ and its $|occur(\mathcal{D}, RWE)_{\setminus\{D_m\}}| = 7,108$. We expect that their execution time will be lower comparing to coverage, since its input ($|occur(\mathcal{D}, RWE)| = 11,139$) is larger comparing to $|occur(\mathcal{D}, RWE)_{\setminus\{D_m\}}|$ and $|occur(\{D_m\}, RWE)|$.

**Information Enrichment.** We can observe in Fig. 12, that the incremental approaches are very fast comparing to *SF*, e.g, *LB* is up to 935× faster. Moreover, the *LB+TPR* method offered up to 2.82× speedup versus *LB*, while we obtained up to 11.7× speedup by using *LB+PRGR* instead of *LB+TPR*. Since the input is smaller in this case, the computation of $|enrich(B, RWE, D_m)|$ was up to 1.23× faster comparing to $|cov(B, RWE)|$. Indicatively, by using *LB+PRGR* we needed 1.1 seconds to compute $|enrich(B, RWE, D_m)|$ for over 1 million subsets of datasets ($2^{20}$ subsets).

**Uniqueness.** We can observe in Fig. 13, that *LB*, *LB+TPR* and *LB+PRGR* approaches are far faster comparing to *SF*, e.g., *LB* is up to 635× faster than *SF*. Moreover, *LB+TPR* was even 5.48× faster than *LB* and *LB+PRGR* was up to 4.1× faster than *LB+TPR*. Comparing to $|cov(B, RWE)|$ and $|enrich(B, RWE, D_m)|$, the computation of $|uniq(D_m, RWE, B)|$ was up to 2.6× and 2.3× faster, respectively, which is rational since the input was much smaller. Indicatively, by using *LB+PRGR* we needed 0.8 seconds to compute $|uniq(D_m, RWE, B)|$ for $2^{20}$ subsets of datasets.

## 6.2 Indicative Measurements & Services for 400 Real Datasets

Here, we show indicative measurements, by exploiting the metrics introduced in §4-5. For each of the measurements below, the results retrieved approximately in 1 second by using *LB+PRGR*.

Table 7. Best subset of datasets of each level $L$ containing the most triples for **Seafood red list species**

| L | $covBest(K, F)$ | $\|cov(B, F)\|$ | $covPer$ | $covGain$ |
|---|---|---|---|---|
| 1 | Fishbase | 3,785 | 25.8% | - |
| 2 | Fishbase, Freebase | 7,036 | 47.9% | 85.0% |
| 3 | Fishbase, Freebase, DBpedia | 9,745 | 66.4% | 38.5% |
| 4 | Fishbase, Freebase, DBpedia, WoRMS | 11,032 | 75.2% | 13.2% |
| 5 | Fishbase, Freebase, DBpedia, WoRMS, Wikidata | 11,507 | 78.4% | 4.3% |

Table 8. Top-5 triads of datasets that cover the most triples for **EDGE species**

| Triad of Datasets | $\|cov(B, F)\|$ | $covPer$ |
|---|---|---|
| DBpedia, Freebase, Wikidata | 12,139 | 62.3% |
| DBpedia, Freebase, Taxonconcept | 11,771 | 60.4% |
| DBpedia, Freebase,Geospecies | 11,750 | 60.3% |
| Freebase, Taxonconcept, Wikidata | 11,531 | 59.1% |
| Freebase, Geospecies, Wikidata | 11,507 | 59.0% |

Table 9. Top-5 dataset pairs providing complementary triples for the **entities of DBpedia**

| Pair of Datasets | $\|enrich(B, F, D_m)\|$ | $enrichPer$ |
|---|---|---|
| Freebase, YAGO | 179,582,545 | 115.0% |
| Freebase, Wikidata | 176,053,104 | 112.8% |
| Wikidata, YAGO | 153,894,606 | 98.6% |
| Freebase, GeoNames | 126,031,309 | 80.7% |
| Freebase, VIAF | 112,948,225 | 72.3% |

Table 10. $|uniq|$ of Wikidata (WD) & YAGO (YG), vs DBpedia (DB) and Freebase (FR)

| $\|uniq\|$ **of** | **Versus** | $uniqPer$ |
|---|---|---|
| Wikidata Triples | DB,FR,YG | 92.5% |
| Wikidata Entities | DB,FR,YG | 77.5% |
| Wikidata Literals | DB,FR,YG | 68.3% |
| YAGO Triples | DB,FR,WD | 93.6% |
| YAGO Entities | DB,FR,WD | 73.3% |
| YAGO Literals | DB,FR,WD | 68.5% |

#### 6.2.1 *Coverage Measurements.*

Tables 7 and 8 show examples for selecting the most relevant datasets for a specific task. Regarding Table 7, suppose that we desire to find $K$ datasets having the most triples for the "Seafood red list species", i.e., a set of 55 fishes from unsustainable fisheries, for 5 different lattice levels. In total, there are 14,660 triples for these fishes in 16 (out of 400) datasets. Table 7 shows for each level $L$, the subset of datasets $B$ with the $arg_B$ max $|cov(B, F)|$, the $covPer(B, F)$, and the $covGain(B', F, B)$ as we go from a subset $B$ to a superset $B'$. The best single dataset is *FishBase*, while as we add more datasets, the *covGain* decreases, e.g., if we add *Wikidata* to the quad of datasets {Fishbase,Freebase,DBpedia,WoRMS}, the *covGain* is only 4.3%. Concerning Table 8, our target is to find the top-5 triads of datasets whose union contains the most triples for all the 115 entities belonging to EDGE ("Evolutionarily Distinct and Globally Endangered") species, and their corresponding $covPer(B, F)$. For these species, there exists 19,485 unique triples in 13 datasets (out of 400). The triad having the $arg_B$ max $|cov(B, F)|$ is {Wikidata, Freebase, DBpedia}, i.e., their union contains 12,139 triples for EDGE species (i.e, 62.3% of all the available triples).

#### 6.2.2 *Information Enrichment Measurements.*

Table 9 shows the top five pairs of datasets that contain the maximum number of complementary triples for the entities of DBpedia, and their corresponding $|enrich(B, F, D_m)|$ and *enrichPer*. In particular, the pair {Freebase, YAGO} enriches the content for DBpedia entities with over 179 millions of new triples, while all the top five pairs of datasets offer over 112 millions of new triples, i.e., triples that are not included in DBpedia.

#### 6.2.3 *Uniqueness Measurements.*

Table 10 shows the percentage of the unique entities, triples and literals of Wikidata (WK) and YAGO (YG), comparing to the other popular Knowledge Bases (KBs), i.e., i) Wikidata versus Freebase (FR), DBpedia (DB) and YAGO, and (ii) YAGO versus the three remaining ones. We can see that both KBs have a high percentage (over 90%) of unique triples comparing to the other KBs. Concerning entities, the percentage is 77.5% for Wikidata and 73.3% for YAGO, meaning that 22.5% of Wikidata entities and 26.7% of YAGO entities, occur at least in one of the three other popular KBs. As regards literals, we can see that the percentages are decreased, however, both KBs contain a high percentage of unique literals (i.e., over 68%).

#### 6.2.4 *More Queries & Services - Online Demo.*

Through http://www.ics.forth.gr/isl/LODsyndesis, one can access the services in the category *Dataset Discovery, Search and Selection*, which exploit the proposed metrics and methods, and offer fast responses for millions of queries, i.e., for queries like those introduced in §1, for 400 datasets, 412 million entities and 2 billion triples. A tutorial video presenting that services and several use cases is available in https://youtu.be/UdQDgod6XME.

We should note that most queries can be answered fast through *LB+PRGR* approach, i.e., for most queries we use as input 25 or less datasets (i.e., at most $2^{25}$ subsets of datasets). Indeed, in our past work [17, 18] we have identified a power-law distribution in the LOD Cloud, i.e., a) most elements (entities, literals, etc.) occur in a small number of other datasets (less than 15 datasets) and only a few number of them in a lot of datasets (i.e., over 15 datasets), and b) most datasets are connected with a small number of datasets. However, in the worst case where more datasets (i.e., even hundreds of datasets) should be pre-selected for a given query (e.g., for entities that occur in a large number of datasets), it is prohibitively expensive to compute the metrics even with *LB+PRGR* approach (e.g., for 40 datasets there exists one trillion possible combinations of datasets). One potential solution is to pre-select the top-$K$ datasets according to some heuristics (such as their cardinality value $|cov(D_i, F)|$, their domain, etc.) and to apply the algorithms for these $K$ datasets, however, it is a problem that requires further research.

## 7 CONCLUDING REMARKS

We proposed content-based union and complement metrics among any subset of RDF Knowledge Bases (or datasets). These metrics are applicable a) for dataset search and discoverability, since they can be exploited for returning back hits, each corresponding to a set (or "cluster") of datasets that satisfy certain criteria, and (b) for assessing and improving the quality of a single or a subset of datasets, mainly in terms of interlinking and reusability. The proposed methods are based on equivalence relationships among different datasets (whose quality has been evaluated and improved in a semi-automatic way) and pre-constructed semantics-aware indexes. For the computation of union and complement metrics, it is a requirement to solve three maximization problems, which are prohibitively expensive by using a straightforward way. For solving such maximization problems and for making feasible the computation of metrics at large scale, we proposed lattice-based incremental algorithms that are based on set-theory properties and pruning and regrouping methods.

Concerning the evaluation results, the lattice-based incremental approach was far faster than a straightforward one (even more than 1000× faster). By combining the incremental approach with pruning and regrouping methods, we observed up to 97× speedup, and we managed to compute the cardinality of the union for millions of subsets of datasets, for entities, triples, and literals in a few seconds (1.3, 1.4 and 3.2 seconds, respectively). Moreover, we computed the cardinality of the absolute and relative complement of entities for a single dataset, with respect to millions of subsets of datasets, in 1.1 and 0.8 seconds, respectively, whereas we reported indicative measurements for 400 RDF datasets. In the future, we plan to work on automatic ways for evaluating/improving the quality of equivalence relationships among different datasets (which are required for computing the metrics), since it is time-consuming to evaluate them in a semi-manually way.

## ACKNOWLEDGEMENTS

We would like to thank the anonymous reviewers for their comments and suggestions which helped us to improve this paper. The research work was supported by the Hellenic Foundation for Research and Innovation (HFRI) and the General Secretariat for Research and Technology (GSRT), under the HFRI PhD Fellowship grant (GA. No. 166).

## REFERENCES

[1] Grigoris Antoniou and Frank Van Harmelen. 2004. *A semantic web primer.* MIT press.

[2] Ciro Baron Neto, Kay Müller, Martin Brümmer, Dimitris Kontokostas, and Sebastian Hellmann. 2016. LODVader: An Interface to LOD Visualization, Analyticsand DiscovERy in Real-time. In *Proceedings of WWW*. 163–166.

[3] M. Ben Ellefi, Z. Bellahsene, J. G. Breslin, E. Demidova, S. Dietze, J. Szymański, and K. Todorov. 2018. RDF dataset profiling–a survey of features, methods, vocabularies and applications. *Semantic Web* Preprint (2018), 1–29.

[4] Adriane Chapman, Elena Simperl, Laura Koesten, George Konstantinidis, Luis-Daniel Ibáñez-Gonzalez, Emilia Kacprzak, and Paul Groth. 2019. Dataset search: a survey. *arXiv preprint arXiv:1901.00735* (2019).

[5] M. d'Aquin and E. Motta. 2011. Watson, more than a semantic web search engine. *Semantic Web* 2, 1 (2011), 55–63.

[6] Mohamed Ben Ellefi, Zohra Bellahsene, Stefan Dietze, and Konstantin Todorov. 2016. Dataset recommendation for data linking: an intensional approach. In *ISWC*. Springer, 36–51.

[7] I. Ermilov, J. Lehmann, M. Martin, and S. Auer. 2016. LODStats: The data web census dataset. In *ISWC*. 38–46.

[8] Thomas Gottron, Ansgar Scherp, Bastian Krayer, and Arne Peters. 2013. LODatio: A schema-based retrieval system for linked open data at web-scale. In *Extended Semantic Web Conference*. Springer, 142–146.

[9] Filip Ilievski, Wouter Beek, Marieke van Erp, Laurens Rietveld, and Stefan Schlobach. 2016. LOTUS: Adaptive Text Search for Big Linked Data. In *ISWC*. Springer, 470–485.

[10] Thomas Jech. 2013. *Set theory*. Springer Science & Business Media.

[11] Maulik Kamdar and Mark Musen. 2017. PhLeGrA: Graph analytics in pharmacology over the web of life sciences linked open data. In *WWW Proceedings*. 321–329.

[12] Shahan Khatchadourian and Mariano P Consens. 2010. Exploring RDF Usage and Interlinking in the Linked Open Data Cloud using ExpLOD.. In *LDOW*.

[13] S. Kruse, P. Papotti, and F. Naumann. 2015. Estimating Data Integration and Cleaning Effort.. In *EDBT*. 61–72.

[14] Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, et al. 2015. DBpedia–a large-scale, multilingual knowledge base extracted from Wikipedia. *Semantic Web* 6, 2 (2015), 167–195.

[15] Luiz André P Paes Leme, Giseli Rabello Lopes, Bernardo Pereira Nunes, Marco Antonio Casanova, and Stefan Dietze. 2013. Identifying candidate datasets for data interlinking. In *ICWE*. Springer, 354–366.

[16] Michalis Mountantonakis and Yannis Tzitzikas. 2016. On Measuring the Lattice of Commonalities Among Several Linked Datasets. *VLDB Endowment* 9, 12 (2016), 1101–1112.

[17] Michalis Mountantonakis and Yannis Tzitzikas. 2018. High Performance Methods for Linked Open Data Connectivity Analytics. *Information* 9, 6 (2018), 134.

[18] Michalis Mountantonakis and Yannis Tzitzikas. 2018. Scalable Methods for Measuring the Connectivity and Quality of Large Numbers of Linked Datasets. *JDIQ* 9, 3, Article 15 (2018), 49 pages.

[19] Michalis Mountantonakis and Yannis Tzitzikas. 2019. Large Scale Semantic Integration of Linked Data: A survey. *CSUR, (accepted for publication)* (2019).

[20] Markus Nentwig, Tommaso Soru, Axel-Cyrille Ngonga Ngomo, and Erhard Rahm. 2014. LinkLion: A Link Repository for the Web of Data. In *ESWC*. Springer, 439–443.

[21] A. B. Neves, R. GG de Oliveira, L. A. P. P. Leme, G. R. Lopes, B. P. Nunes, and M. A. Casanova. 2018. Empirical Analysis of Ranking Models for an Adaptable Dataset Search. In *European Semantic Web Conference*. Springer, 50–64.

[22] Andriy Nikolov, Mathieu d' Aquin, and Enrico Motta. 2011. What should I link to? Identifying relevant sources and classes for data linking. In *Joint International Semantic Technology Conference*. Springer, 284–299.

[23] Tope Omitola, Landong Zuo, Christopher Gutteridge, Ian C Millard, Hugh Glaser, Nicholas Gibbins, and Nigel Shadbolt. 2011. Tracing the provenance of linked data using void. In *Proceedings of WIMS*. ACM, 17.

[24] Eyal Oren, Renaud Delbru, Michele Catasta, Richard Cyganiak, Holger Stenzhorn, and Giovanni Tummarello. 2008. Sindice.com: a Document-Oriented Lookup Index for Open Linked Data. *IJMSO* 3, 1 (2008), 37–52.

[25] Emmanuel Pietriga, Hande Gözükan, Caroline Appert, Marie Destandau, Šejla Čebirić, François Goasdoué, and Ioana Manolescu. 2018. Browsing Linked Data Catalogs with LODAtlas. In *ISWC*. Springer, 137–153.

[26] Thomas Rebele, Fabian Suchanek, Johannes Hoffart, Joanna Biega, Erdal Kuzey, and Gerhard Weikum. 2016. YAGO: a multilingual knowledge base from Wikipedia, Wordnet, and Geonames. In *ISWC*. Springer, 177–185.

[27] Theodoros Rekatsinas, Xin Luna Dong, Lise Getoor, and Divesh Srivastava. 2015. Finding Quality in Quantity: The Challenge of Discovering Valuable Sources for Integration.. In *CIDR*.

[28] Theodoros Rekatsinas, Xin Luna Dong, and Divesh Srivastava. 2014. Characterizing and selecting fresh data sources. In *SIGMOD*. ACM, 919–930.

[29] L. Rietveld, W. Beek, and S. Schlobach. 2015. LOD lab: Experiments at LOD scale. In *ISWC*. Springer, 339–355.

[30] Petar Ristoski, Christian Bizer, and Heiko Paulheim. 2015. Mining the web of linked data with rapidminer. *Web Semantics: Science, Services and Agents on the World Wide Web* 35 (2015), 142–151.

[31] Yannis Tzitzikas et al. 2016. Unifying heterogeneous and distributed information about marine species through the top level ontology MarineTLO. *Program* 50, 1 (2016), 16–40.

[32] Andre Valdestilhas, Tommaso Soru, Markus Nentwig, Edgard Marx, Muhammad Saleem, and Axel-Cyrille Ngonga Ngomo. 2018. Where is my URI?. In *ESWC*. Springer, 671–681.

[33] Pierre-Yves Vandenbussche, Ghislain A Atemezing, María Poveda-Villalón, and Bernard Vatant. 2017. Linked Open Vocabularies (LOV): a gateway to reusable semantic vocabularies on the Web. *Semantic Web* 8, 3 (2017), 437–452.

[34] Pierre-Yves Vandenbussche, Jürgen Umbrich, Luca Matteis, Aidan Hogan, and Carlos Buil-Aranda. 2016. SPARQLES: Monitoring public SPARQL endpoints. *Semantic Web* (2016), 1–17.

[35] D. Vrandečić and M. Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *Commun. ACM* 57, 10 (2014), 78–85.

[36] Andreas Wagner, Peter Haase, Achim Rettinger, and Holger Lamm. 2014. Entity-based data source contextualization for searching the web of data. In *European Semantic Web Conference*. Springer, 25–41.

[37] S. Yumusak, E. Dogdu, H. Kodaz, A. Kamilaris, and P. Vandenbussche. 2017. SpEnD: Linked Data SPARQL Endpoints Discovery Using Search Engines. *IEICE TRANSACTIONS on Information and Systems* 100, 4 (2017), 758–767.

[38] Amrapali Zaveri, Anisa Rula, Andrea Maurino, Ricardo Pietrobon, Jens Lehmann, and Sören Auer. 2015. Quality assessment for linked data: A survey. *Semantic Web* 7, 1 (2015), 63–93.

## A PROOFS FOR THE PROPOSITIONS

At first, we provide some lemmas that are used for providing the proofs for Propositions 1-4.

LEMMA A.1. If $k \in F(D_i)$, then $k \in Left(ind_F), ind_F(k) = B_i, D_i \in B_i$. In particular, since we store any element $k$ and its provenance, $k$ occurs also in the left side of the corresponding index, and its posting list $B_i \subseteq \mathcal{D}$ contains $D_i$.

LEMMA A.2. If $B_i \neq B_j$ ($B_i \in \mathcal{P}(\mathcal{D}), B_j \in \mathcal{P}(\mathcal{D})$), then $\{k \in Left(ind_F) | ind_F(k) = B_i\} \cap \{k \in Left(ind_F) | ind_F(k) = B_j\} = \emptyset$, i.e., they are disjoint. It holds because each element $k \in Left(ind_F)$ has a unique posting list of dataset IDs.

LEMMA A.3. From the set theory, we know that for $n$ disjoint sets $\{A_1, A_2, ..., A_n\}$, $|A_1 \cup A_2 \cup ... \cup A_n| = |A_1| + |A_2| + ... + |A_n|$ *(Proof can be found in [10]).*

LEMMA A.4. From the set theory, we know that for $n$ sets $\{A_1, A_2, ..., A_n\}$, $|A_n \setminus \{A_1 \cup A_2 \cup ... \cup A_{n-1}\}| = |A_n| - |A_n \cap \{A_1 \cup A_2 \cup ... \cup A_{n-1}\}|$ *(Proof can be found in [10]).*

### A.1 Proof of Prop. 1

$$
\begin{aligned}
|cov(B,F)| &= | \cup_{D_i \in B} \ F(D_i)| = | \cup_{D_i \in B} \{k | k \in F(D_i)\}| \\
&\overset{(Lemma\ A.1)}{=} | \cup_{B_i \in \mathcal{P}(\mathcal{D})} \{k \in Left(ind_F) | ind_F(k) = B_i, B \cap B_i \neq \emptyset\}| \\
&\overset{(Def.\ 4.2)}{=} | \cup_{B_i \in occur(B,F)} \{k \in Left(ind_F) | ind_F(k) = B_i\}| \\
&\overset{(Lemma\ A.2)}{=} |\{k \in Left(ind_F) | ind_F(k) = B_1\} \cup ... \cup \{k \in Left(ind_F) | ind_F(k) = B_n\}| \\
&\overset{(Lemma\ A.3)}{=} |\{k \in Left(ind_F) | ind_F(k) = B_1\}| + ... + |\{k \in Left(ind_F) | ind_F(k) = B_n\}| \\
&\overset{(Def.\ 4.1)}{=} directCount(B_1, F) + ... + directCount(B_n, F) \\
&\overset{(Def.\ 4.2)}{=} \sum_{B_i \in occur(B,F)} directCount(B_i, F) \diamond
\end{aligned}
$$

### A.2 Proof of Prop. 2

$$\begin{aligned}
|enrich(B,F,D_m)| &= |(cov(B,F) \setminus F(D_m))| = |\{\cup_{D_j \in B} F(D_j)\} \setminus F(D_m)| \\
&= | \cup_{D_j \in B} \{k | k \in F(D_j), k \notin F(D_m)\}| \\
&\overset{(Lemma\ A.1)}{=} | \cup_{B_i \in \mathcal{P}(\mathcal{D})} \{k \in Left(ind_F) | ind_F(k) = B_i, B_i \cap B \neq \emptyset, D_m \notin B_i\}| \\
&\overset{(Def.\ 4.3)}{=} | \cup_{B_i \in occur(B,F)_{\setminus\{D_m\}}} \{k \in Left(ind_F) | ind_F(k) = B_i\}| \\
&\overset{(Similarly\ to\ Proof\ of\ Prop.1.)}{=} \sum_{B_i \in occur(B,F)_{\setminus\{D_m\}}} directCount(B_i, F) \diamond
\end{aligned}$$

### A.3 Proof of Prop. 3

$$\begin{aligned}
|uniq(D_m,F,B)| &= |(F(D_m) \setminus cov(B,F))| \overset{(Lemma\ A.4)}{=} |F(D_m)| - |(F(D_m) \cap cov(B,F))| \\
&= |cov(\{D_m\},F)| - | \cup_{D_j \in B} \{k | k \in F(D_j), k \in F(D_m)\}| \\
&= |cov(\{D_m\},F)| - | \cup_{B_i \in \mathcal{P}(\mathcal{D})} \{k \in Left(ind_F) | ind_F(k) = B_i, D_m \in B_i, B \cap B_i \neq \emptyset\}| \\
&\overset{(Def.\ 4.4)}{=} |cov(\{D_m\},F)| - | \cup_{B_i \in occur(B,F)_{D_m}} \{k \in Left(ind_F) | ind_F(k) = B_i\}| \\
&\overset{(Similarly\ to\ Proof\ of\ Prop.1.)}{=} |cov(\{D_m\},F)| - \sum_{B_i \in occur(B,F)_{D_m}} directCount(B_i, F)
\end{aligned}$$

### A.4 Proof of Prop. 4

$$\begin{aligned}
|cov(B',F)| &\overset{B'=B\cup\{D_m\}}{=} |\{\cup_{D_i \in B} F(D_i)\} \cup F(D_m)| \\
&\overset{(Lemma\ 5.1)}{=} | \cup_{D_i \in B}\ F(D_i)| + |F(D_m) \setminus \{\cup_{D_i \in B} F(D_i)\}| \\
&= |cov(B,F)| + |\{k | k \in F(D_m),\ and\ \forall D_i \in B, k \notin F(D_i)\}| \\
&\overset{(Lemma\ A.1)}{=} |cov(B,F)| + | \cup_{B'_i \in \mathcal{P}(\mathcal{D})} \{k \in Left(ind_F) | ind_F(k) = B'_i, D_m \in B'_i, B \cap B'_i = \emptyset\}| \\
&\overset{(Def.\ 4.3)}{=} |cov(B,F)| + | \cup_{B'_i \in occur(\{D_m\},F)_{\setminus B}} \{k \in Left(ind_F) | ind_F(k) = B'_i\}| \\
&\overset{(Similarly\ to\ Proof\ of\ Prop.1.)}{=} |cov(B,F)| + \sum_{B'_i \in occur(\{D_m\},F)_{\setminus B}} directCount(B'_i, F) \diamond
\end{aligned}$$