On Measuring the Lattice of Commonalities Among Several Linked Datasets

Michalis Mountantonakis and Yannis Tzitzikas

FORTH-ICS
Information Systems Laboratory

University of Crete
Computer Science Department

Very Large Data Bases
INDIA 2016

Outline

- Introduction (3m)
- Related Work (1m)
- The Proposed Indexes & Algorithms (12 m)
  - Prefix Index
  - SameAs Catalog
  - Element Index
  - Lattice Algorithms
- Experimental Evaluation (3 m)
  - Experiments on 300 LOD Cloud Datasets
  - Comparative results
- Publishing & Exchanging Measurements (0.5 m)
  - Datahub, Visualization, Answerable Queries
- Conclusion (0.5 m)

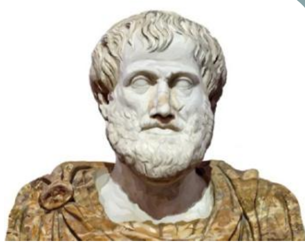
Introduction

Motivation

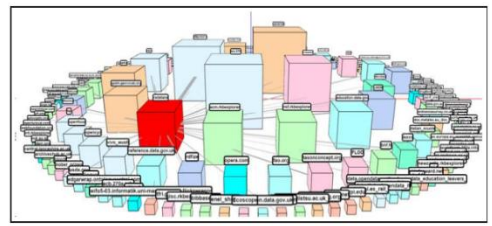
- Our approach tries to **aid** the **execution** of the **following tasks**:

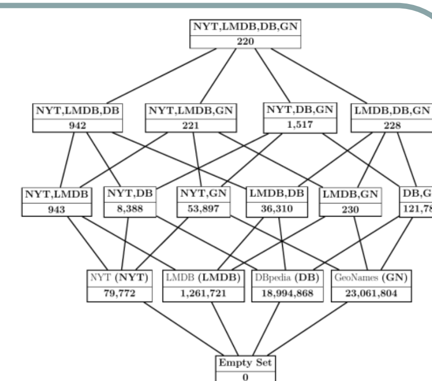
Object Coreference

Give me all the URIs of Aristotle

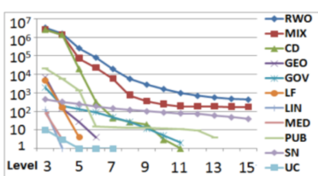


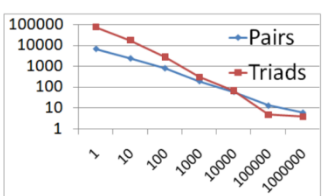
Visualizations





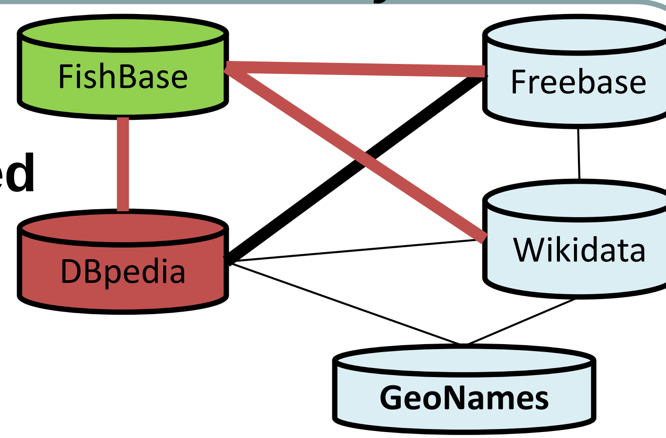
Connectivity Assessment and Monitoring





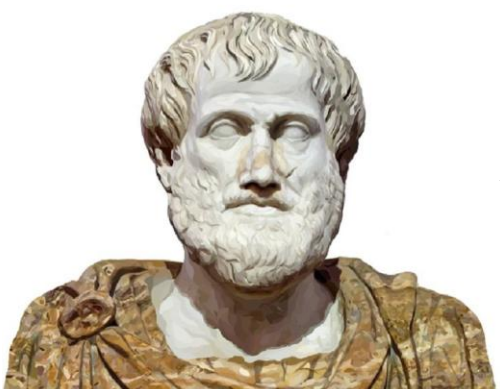
Dataset Discovery

Give me the K most connected Datasets to my Dataset



Motivation- **Object Coreference** (*Everything for a URI*)

- Suppose that one user wants to find **all the available information** (and URIs) **about an entity** (or URI).
  - We want also the URIs being **owl:sameAs** with the desired one.
- It is not trivial to achieve this, since
  - the symmetric and transitive closure of owl:sameAs relationships should be computed
  - it presupposes knowledge from all the datasets.



1 URI

Before Closure

http://dbpedia.org/resource/Aristotle

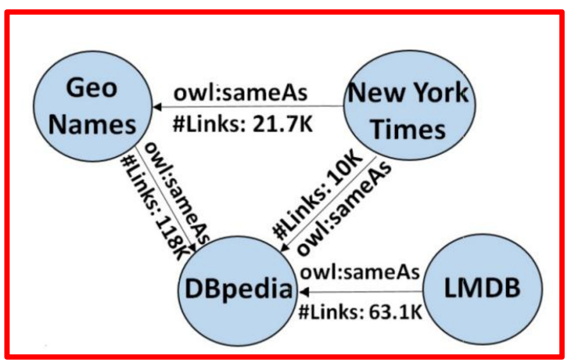
After Closure

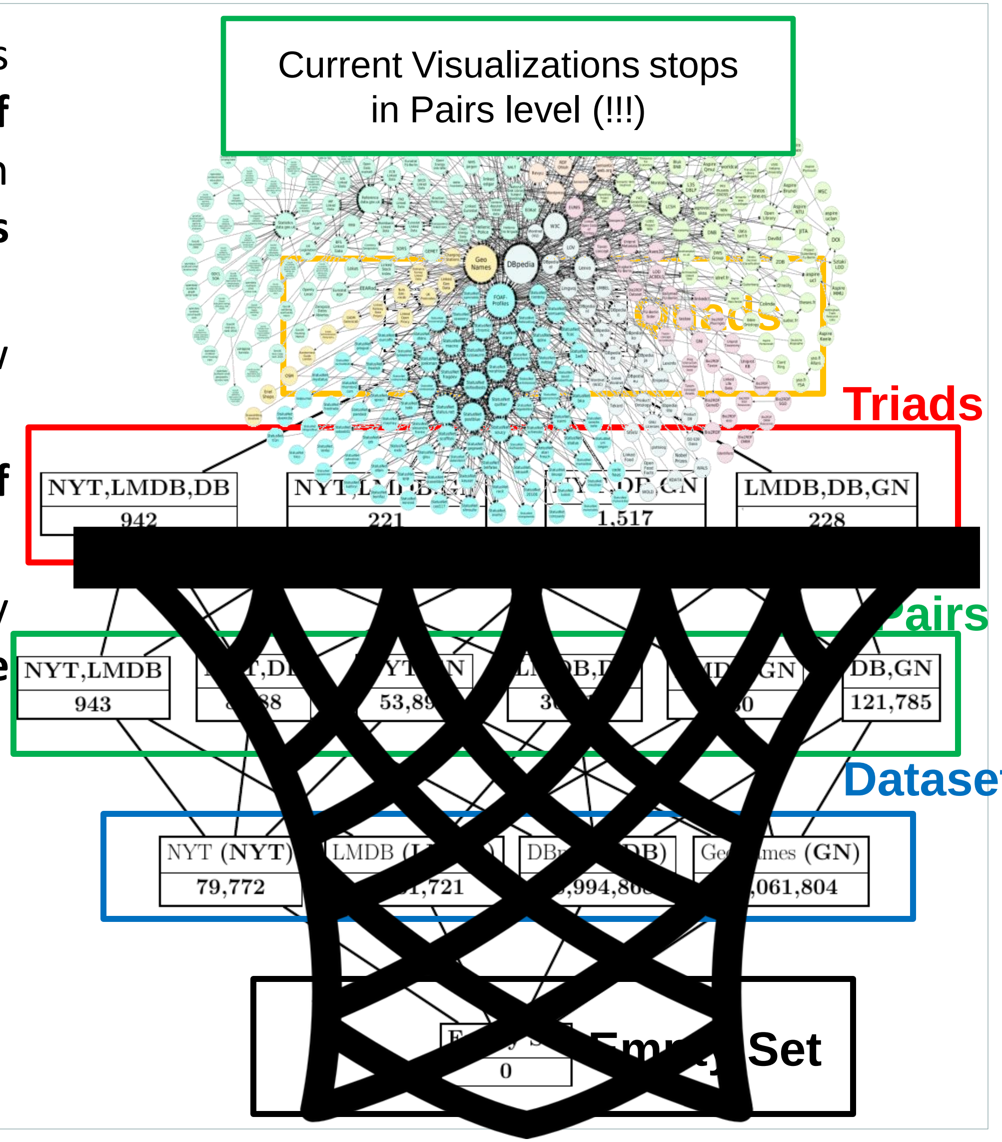
14 URIs!!!

equivalentURI
http://lemon-model.net/lexica/uby/wn/WN_Synset_58775
http://sw.cyc.com/concept/Mx4rvViJoJwpEbGdrcN5Y29ycA
http://ta.sandrart.net/-person-112
http://d-nb.info/gnd/118650130
http://wordnet-rdf.princeton.edu/wn31/110841942-n
http://datos.artium.org/id/collections/library/auth/E21_Person/f17bdd5d-9185-3871-ade2-3dc94fd7856f
http://data.bibsys.no/data/notrbib/authorityentry/x90052251
http://rdf.freebase.com/ns/m.0gz_
http://www.w3.org/2006/03/wn/wn20/instances/synset-Aristotle-noun-1
http://viaf.org/viaf/7524651
http://www.wikidata.org/entity/Q868
http://yago-knowledge.org/resource/Aristotle
http://sw.opencyc.org/concept/Mx4rvViJoJwpEbGdrcN5Y29ycA
http://dbpedia.org/resource/Aristotle

Motivation – **Connectivity & Visualizations (LOD Cloud)**

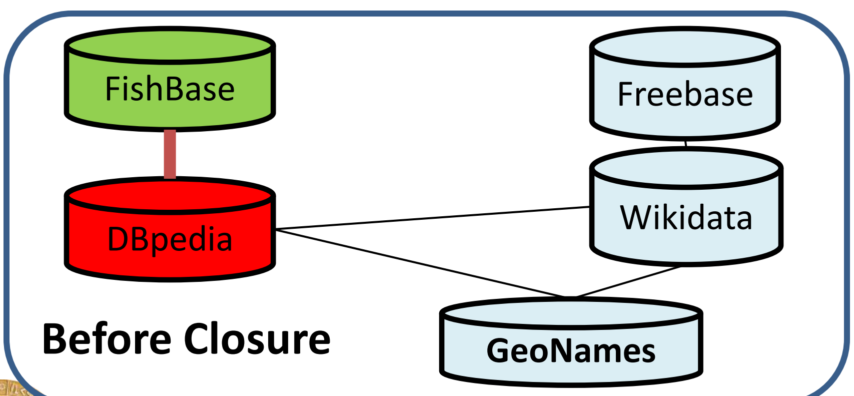
- The ultimate objective of Linked Data is linking and integration a **big number of RDF datasets** has already been **published** and this **number keeps increasing!**
- It is difficult to understand how **connected** the current **LOD cloud** is!
- **Only measurements** between **pairs of datasets** are **available!**
- It is **not possible** to see how many common **entities** exist between **three or more sources!** ☹

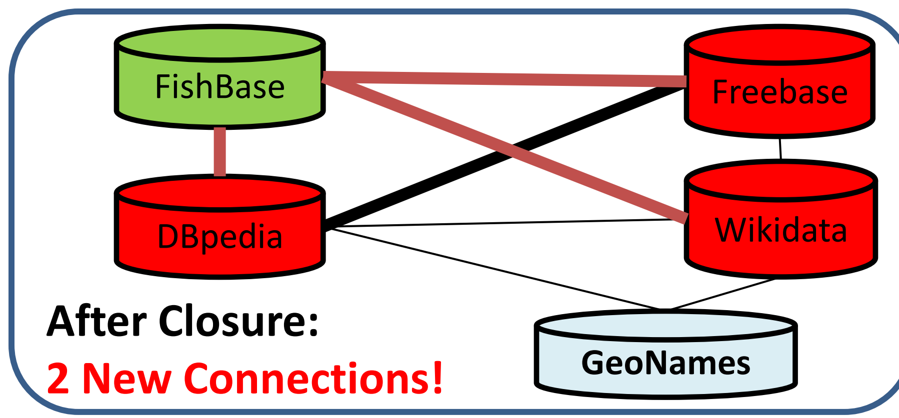




Motivation – **Dataset Discovery & Selection**

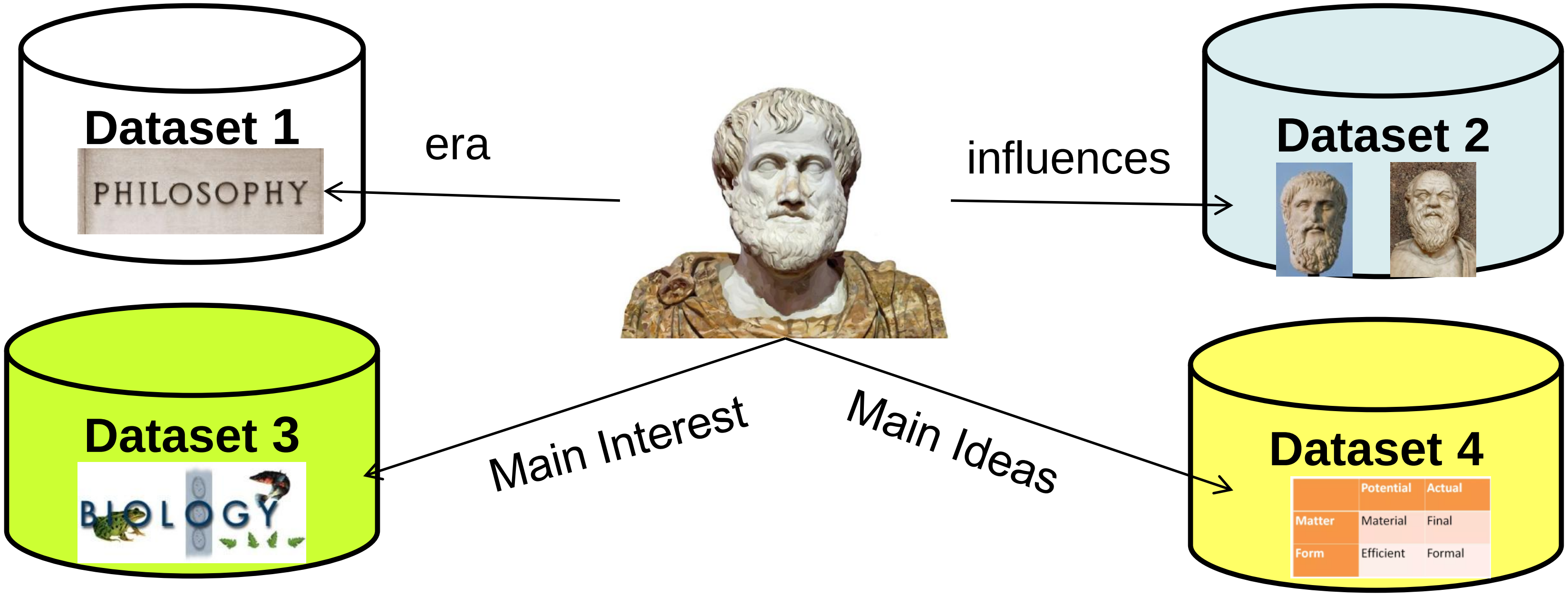
- Suppose that you **publish a dataset and you establish relationships with DBpedia**. Then, you would like to find the K more related datasets to our dataset :
  - (a) for constructing a semantic warehouse
  - (b) for mediator-based query answering.
- **With the proposed indexes and measurements** (including the computation of the transitive closure of **owl:sameAs**), **you could get much more datasets**!!!





Motivation- Gain of Integration

- **Collecting information** about the **same real world entity** from **several datasets**
  - can **verify or clean** that **information**
  - can **produce** a more **accurate** or **correct consolidated dataset**
  - can **“widen” the information** for a URI.



Contributions of our work

It is very expensive to perform all these measurements straightforwardly.

1. There are many datasets and some of them are very big.
2. The possible combinations of datasets is exponential in number!

Contributions

- We introduce a **namespace-based prefix index** for speeding up the computation of the metrics
- We introduce a **sameAs catalog** for computing the symmetric and transitive closure of the sameAs relationships encountered in the datasets
- We introduce a **semantics-aware element index** and **lattice-based incremental algorithms** for speeding up the computation of the intersection of URIs of any set of datasets
- We report **connectivity measurements** for a subset of the **current LOD Cloud** that comprises **300 datasets**.
- We **measure the speedup** obtained by the proposed indexes and algorithms by providing **comparative results**

Related Work

Related Work: Experiments in LOD Scale

- ***LOD Laundromat: LOD Lab Experiments in LOD Scale [1] (ISWC'2015)***
  - **38 billion triples** indexed from **657 thousand documents**

Key differences
1. We provide measurements concerning the **connectivity of various datasets.** They provide statistics about **validity or format of documents, number of triples etc.** 2. We take into account the semantics (e.g., sameAs relationships)

- ***LinkLion: A Link Repository for the Web of Data [2] (ESWC'2014)***
  - An open link repository containing mappings between pairs of datasets (e.g., owl:sameAs relationships)

Key differences
1.They take into account only mappings between **2 datasets**. We find common real world objects between **two or more datasets**. 2. We compute the **transitive** and **symmetric** closure. 3. We use **indexes** for **reusing** the **measurements** for **several tasks**.

Related Work: Indexes for search and queries

- **YARS2** [3] is a federated repository that queries linked data coming from different datasets.
  - **Why they use indexes**: for allowing direct lookups on multiple dimensions without requiring joins
- **Swoogle** [4] is a crawler-based indexing and retrieval system for the semantic web.
  - **Why they use indexes**: for answering user's queries
- **RDF-3X** [5] is an engine for scalable management of RDF data which is an implementation of SPARQL [6].
  - **Why they use indexes**: for faster query answering by maintaining six indexes for all possible permutations of an RDF triple members (s p o)
- **We use indexes** for finding how connected are the different subsets of any size of dataets and for performing faster such measurements

The Proposed Indexes & Algorithms

Definitions

- ***D={D$_1$,…, D$_n$}* :** a set of Datasets
- ***U={U$_1$,…, U$_n$}* :** a set of URIs
- ***P(D)*** : the powerset of ***D***.

- **Considering Equivalence Relationships**:
  - ***sm(D$_i$)***: the **owl:sameAs** relationships of D_i $sm(D_i) = \{(u, u') \mid (u, \texttt{sameAs}, u') \in triples(D_i)\}$
  - ***SM(B)*** : the union of the **owl:sameAs** relationships in B. $SM(B) = \cup_{D_i \in B}\ sm(D_i)$.
  - **C(SM(B)):** the transitive and symmetric closure of sameAs relationships for subset B

- **Classes of Equivalence**:
  - *Utemp*={u_1 , u_2 , u_3 , u_4 , u_5 } and 2 owl:sameAs relationships: $u_1 \sim u_3$ and $u_1 \sim u_4$.

  Derived classes of equivalence : *Utemp/~*={ {u_1 , u_3 , u_4 } , {u_2},{u_5}}

  ✓Equivalent URIs (considering all datasets in B) of a URI u (or all URIs U) :

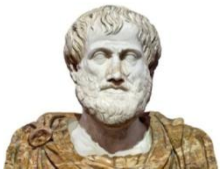
$$Equiv(u, B) = \{ u' \mid (u, u') \in \mathcal{C}(SM(B))\} \quad Equiv(U, B) = \cup_{u \in U} Equiv(u, B)$$

Problem Statement

- We ***focus** on how to compute efficient the following formulas for any* ***u* ∈ U** and ***B* ⊆ *D*:**

❑ **Datasets Containing a particular URI u or Equivalent URI:**

$$\boxed{dsets_{\sim}(u) = \{D_i \in \mathcal{D} \mid (\{u\} \cup Equiv(u, \mathcal{D})) \cap U_i \neq \emptyset\}}$$

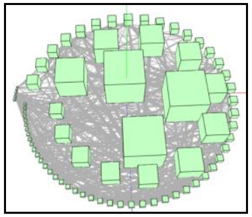


Object Coreference

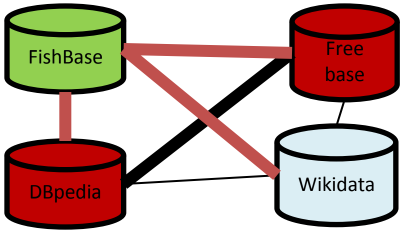
❑ **Number of common real world objects in a subset B:** the number of common classes of equivalence in a subset B where

$$cu_{\sim}(B) = \{\ u \in U \mid dsets_{\sim}(u) \supseteq B\}$$

$$\boxed{co_{\sim}(B) = |cu_{\sim}(B)/\sim|}$$

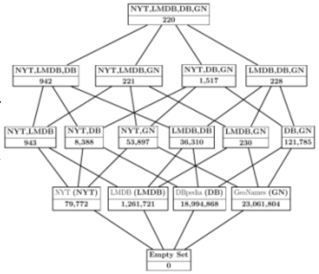


Visualizations



Dataset Discovery

Connectivity Assessment



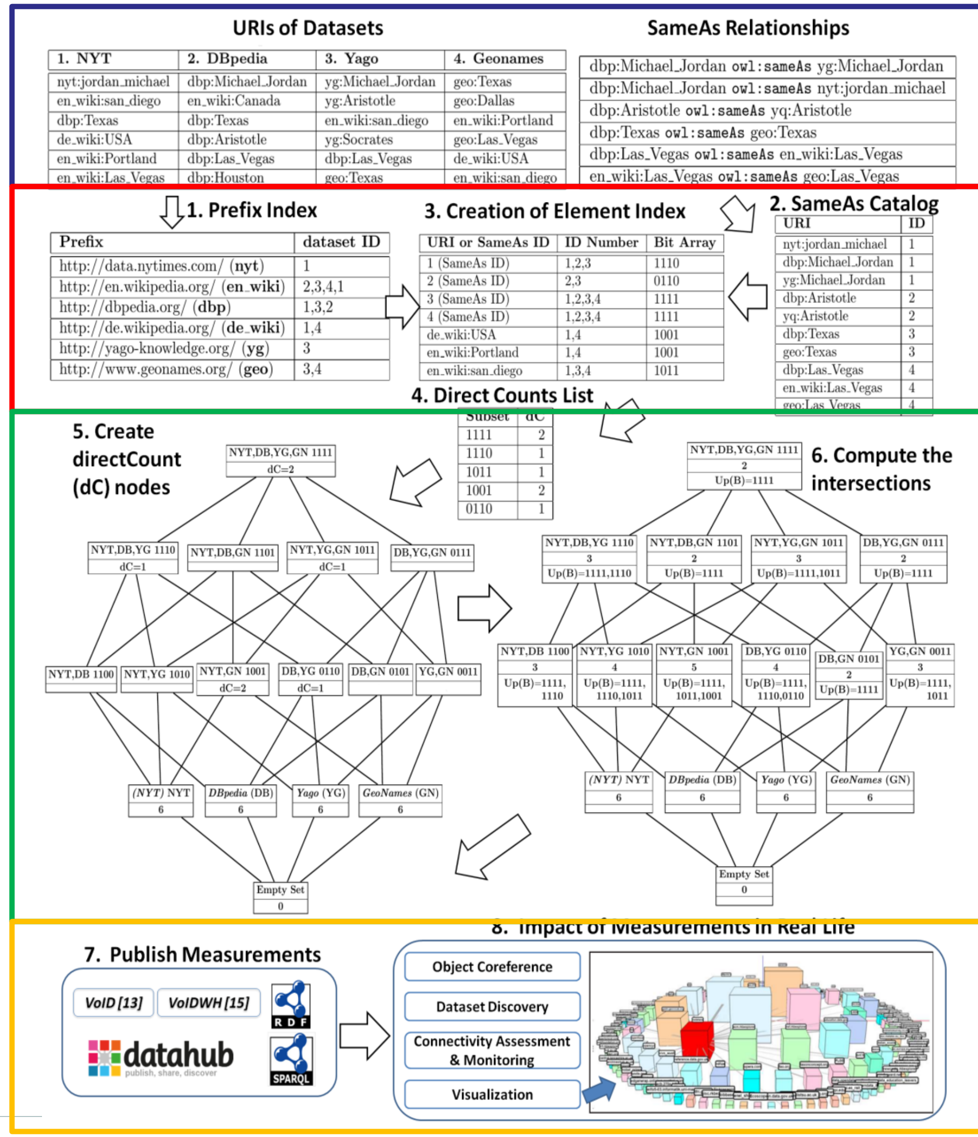
The Proposed Approach - Running example

Input

Indexes

Lattice Creation

Publishing & Exploitation



Prefix Index

- **What is it**: A prefix index lists all namespaces and for each one what datasets contain them.
- **Construction Method: Send a SPARQL query or Scan the URIs of each dataset once.**
  - Store for each prefix the datasets that is appears **(in ascending order w.r.t. frequency**)
- **Rationale**: For reducing the cost of finding common URIs since:
  - There is **no need to compare URIs having different Prefixes**
  - If **a prefix p exists in one dataset**, it is **impossible** for the **URIs starting with p to be found in an another dataset**
- **Efficiency**: This index is **usually small in size** (i.e., 212 prefixes per dataset)**.**

1. NYT	2. DBpedia	3. Yago	4. Geonames
nyt:jordan_michael	dbp:Michael_Jordan	yg:Michael_Jordan	geo:Texas
en_wiki:san_diego	en_wiki:Canada	yg:Aristotle	geo:Dallas
dbp:Texas	dbp:Texas	en_wiki:san_diego	en_wiki:Portland
de_wiki:USA	dbp:Aristotle	yg:Socrates	geo:Las_Vegas
en_wiki:Portland	dbp:Las_Vegas	dbp:Las_Vegas	de_wiki:USA
en_wiki:Las_Vegas	dbp:Houston	geo:Texas	en_wiki:san_diego

Prefix	dataset ID
http://data.nytimes.com/ (**nyt**)	1
http://en.wikipedia.org/ (**en_wiki**)	2,3,4,1
http://dbpedia.org/ (**dbp**)	1,3,2
http://de.wikipedia.org/ (**de_wiki**)	1,4
http://yago-knowledge.org/ (**yg**)	3
http://www.geonames.org/ (**geo**)	3,4

SameAs Catalog

- **What is it**: A catalog where all the URIs that belong to the same class of equivalence are getting the same signature.
- **Construction Method:** We introduce a signature-based algorithm that stores for each URI of the pair (u,u') ∈ SM(D) a unique ID (or signature) according to 5 rules.
- The 5 rules are the following for a pair of URIs u_1 *sameAs* u_2 *:*
  - ***Rule 1***. If both URIs have not a signature, a new signature is assigned in both of them.

Insert u_5 sameAs u_6

ID	URIs
1	u_1, u_2
2	u_3, u_4

Classes of Equivalence

ID	URIs
1	u_1, u_2
2	u_3, u_4
3	$\mathbf{u_5, u_6}$

SameAs Catalog Construction Rules

– ***Rules 2-3***. If u_1 has an signature while u_2 has not, u_2 gets the same signature as u_1. (or the opposite)

Insert u_3 sameAs u_7

ID	URIs
1	u_1, u_2
2	u_3, u_4
3	$\mathbf{u_5, u_6}$

⇒

ID	URIs
1	u_1, u_2
2	$u_3, u_4, \mathbf{u_7}$
3	u_5, u_6

– ***Rule 4***. If both URIs have the same signature nothing changes

– ***Rule 5***. If both URIs have a different signature, the URIs of these two signatures are concatenated and only the lower signature is kept.

Insert u_1 sameAs u_3

ID	URIs
1	u_1, u_2
2	$u_3, u_4, \mathbf{u_7}$
3	u_5, u_6

⇒

ID	URIs
1	$u_1, u_2, \mathbf{u_3}, \mathbf{u_4}, \mathbf{u_7}$
~~2~~	$\cancel{u_3, u_4, u_7}$
3	u_5, u_6

⇒

URI	ID
u_1	1
u_2	1
u_3	1
u_4	1
u_7	1
u_5	3
u_6	3

SameAs Catalog

SameAs Catalog- Running Example & Efficiency

SameAs Relationships

dbp:Michael_Jordan **owl:sameAs** yg:Michael_Jordan
dbp:Michael_Jordan **owl:sameAs** nyt:jordan_michael
dbp:Aristotle **owl:sameAs** yq:Aristotle
dbp:Texas **owl:sameAs** geo:Texas
dbp:Las_Vegas **owl:sameAs** en_wiki:Las_Vegas
en_wiki:Las_Vegas **owl:sameAs** geo:Las_Vegas

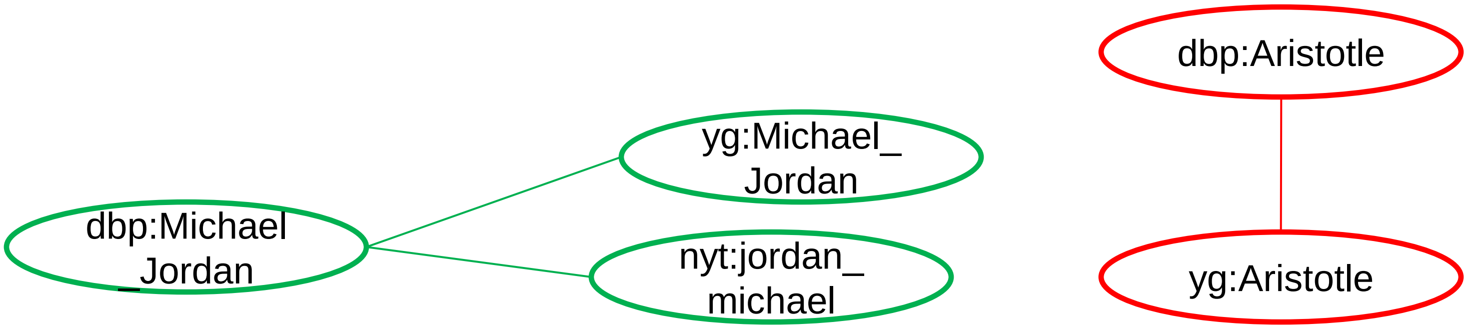
2. SameAs Catalog

URI	ID
nyt:jordan_michael	1
dbp:Michael_Jordan	1
yg:Michael_Jordan	1
dbp:Aristotle	2
yq:Aristotle	2
dbp:Texas	3
geo:Texas	3
dbp:Las_Vegas	4
en_wiki:Las_Vegas	4
geo:Las_Vegas	4

- **Efficiency:**
  - **(+)** It reads each sameAs pair only once i.e., O(n) time complexity
  - **(-)** It keeps in memory the catalog and the classes of equivalence, i.e., space complexity is O(m), where m is the number of distinct URIs.

SameAs Catalog – Alternative Approach

- Alternative Approach
  - Turn the **SameAs Relationships** to an undirected graph
  - Find the connected components (CC) be using Tarjan's Algorithm [7].
  - (+) The time complexity of CC algorithm is O(m) (m: distinct URIs)
  - (-) It requires the creation of the graph (O(n) for creating the graph)
  - (-) Total time complexity required is O(m+n)
  - (-) Total space needed is O(m+n)



Element Index

- **What it is**: For each URI or signature **appearing in two or more datasets,** it stores the datasets where it appears.

- There are two different ways to store the datasets in the Element Index:

  1. Store a **bit array** of length n (n=|D|) that indicates the datasets in which an element belongs
     - (+) Can be easily exploited for lattice representation
  2. Create an **inverted index** in which for each URI stores a posting list of dataset identifiers
     - (+) It reduces the size of the Element-Index especially for sparse indexes

Element Index (1st step)- SameAs Catalog Exploitation

- The algorithm reads each time the URIs of a specific dataset
- The first step is **always to look if the URI exists in the SameAs Catalog**
  - If a URI (of Di) belongs to SameAs Catalog add to the Element Index:
    - an entry comprising the identifier of the URI (in the SameAs Catalog)
    - the dataset ID (i.e., an arbitrary distinct number)

1. NYT	2. DBpedia	3. Yago	4. Geonames
nyt:jordan_michael	dbp:Michael_Jordan	yg:Michael_Jordan	geo:Texas
en_wiki:san_diego	en_wiki:Canada	yg:Aristotle	geo:Dallas
dbp:Texas	dbp:Texas	en_wiki:san_diego	en_wiki:Portland
de_wiki:USA	dbp:Aristotle	yg:Socrates	geo:Las_Vegas
en_wiki:Portland	dbp:Las_Vegas	dbp:Las_Vegas	de_wiki:USA
en_wiki:Las_Vegas	dbp:Houston	geo:Texas	en_wiki:san_diego

URIs of Datasets

URI	ID
nyt:jordan_michael	1
dbp:Michael_Jordan	1
yg:Michael_Jordan	1
dbp:Aristotle	2
yq:Aristotle	2
dbp:Texas	3
geo:Texas	3
dbp:Las_Vegas	4
en_wiki:Las_Vegas	4
geo:Las_Vegas	4

SameAs Catalog

URI or sameAsID	ID Number	Bit Array
1 (SameAsID)	1,2,3	1110

Element Index

Element Index - Remaining Steps

- ***2nd step*** : update the index entry of a URI if the URI already belongs to the Element-Index .
- ***3rd step***: If the prefix of the URI exists (by looking up the prefix index) in an another dataset, then there exist two possible approaches:
- **First approach (*Index*):**
  - 1. Store the URI in the element-index
    - → It is a **candidate** for existing in an another source
  - 2. In the end delete those URIs that belong only to one source
- **Second approach (*Index+ASK*) for reducing space:**
  - 1. Send an ASK Query **only** to the other datasets containing this prefix in order to discover if this URI exists also in an another dataset
    - *ASK { **graph <Dataset>** {{ URI ?p ?o} union {?o ?p URI}} }*
  - 2. Store the URI in the index if an ASK query returned a true answer
    - → It **surely exists** at least in two datasets

Element Index - Running Example

For the URI de_wiki:USA of NYT

1. Check SameAs Catalog→ No entry with this URI
2. Check Prefix in Prefix Index→ Exists in 2 Datasets
3. ASK Geonames for this URI→ True Answer
4. Add to element index the URI and the dataset IDs

For the URI yg:Socrates of Yago

1. Check SameAs Catalog→ No entry with this URI
2. Check Prefix in Prefix Index→ Ignore URI: It exists in 1 Dataset

URIs

1. NYT	2. DBpedia	3. Yago	4. Geonames
nyt:jordan_michael	dbp:Michael_Jordan	yg:Michael_Jordan	geo:Texas
en_wiki:san_diego	en_wiki:Canada	yg:Aristotle	geo:Dallas
dbp:Texas	dbp:Texas	en_wiki:san_diego	en_wiki:Portland
de_wiki:USA	dbp:Aristotle	yg:Socrates	geo:Las_Vegas
en_wiki:Portland	dbp:Las_Vegas	dbp:Las_Vegas	de_wiki:USA
en_wiki:Las_Vegas	dbp:Houston	geo:Texas	en_wiki:san_diego

SameAs Catalog

URI	ID
nyt:jordan_michael	1
dbp:Michael_Jordan	1
yg:Michael_Jordan	1
dbp:Aristotle	2
yq:Aristotle	2
dbp:Texas	3
geo:Texas	3
dbp:Las_Vegas	4
en_wiki:Las_Vegas	4
geo:Las_Vegas	4

Prefix Index

Prefix	dataset ID
http://data.nytimes.com/ (**nyt**)	1
http://en.wikipedia.org/ (**en_wiki**)	2,3,4,1
http://dbpedia.org/ (**dbp**)	1,3,2
http://de.wikipedia.org/ (**de_wiki**)	1,4
http://yago-knowledge.org/ (**yg**)	3
http://www.geonames.org/ (**geo**)	3,4

Element Index

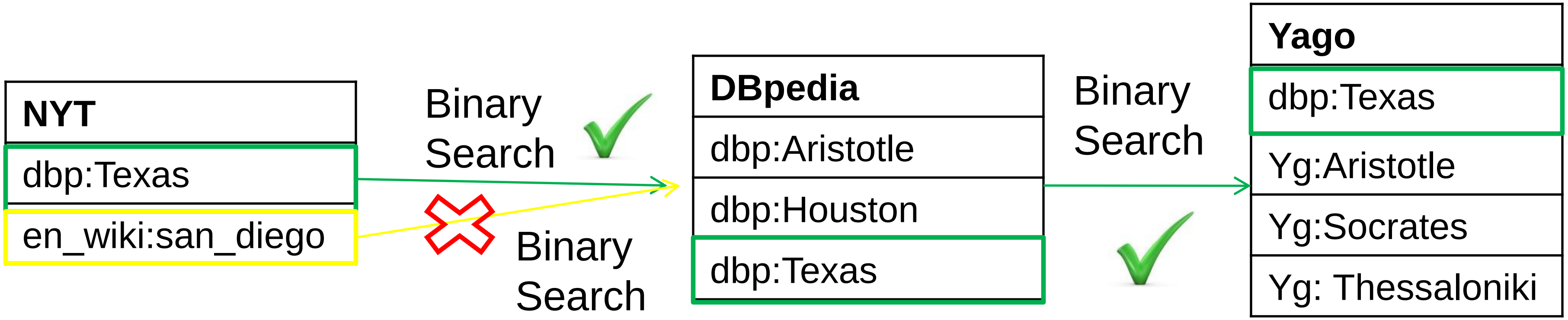
URI or SameAs ID	ID Number	Bit Array
1 (SameAs ID)	1,2,3	1110
2 (SameAs ID)	2,3	0110
3 (SameAs ID)	1,2,3,4	1111
4 (SameAs ID)	1,2,3,4	1111
de_wiki:USA	1,4	1001
en_wiki:Portland	1,4	1001
en_wiki:san_diego	1,3,4	1011

Element Index (cont.)

Efficiency: Time complexity is O(y) where y is the sum of all |Ui|.

An alternative straightforward approach can be the following:

- ❑ Sort the URIs of each dataset lexicographically
- ❑ For each B ∈ P(D) , read the URIs of the smallest dataset.
- ❑ Perform binary searches to the (n-1) remaining sources.
- ❑ Total time complexity is $O(2^{|D|} n\log n)$



Ordering the Prefix Index for Reducing the ASK Queries

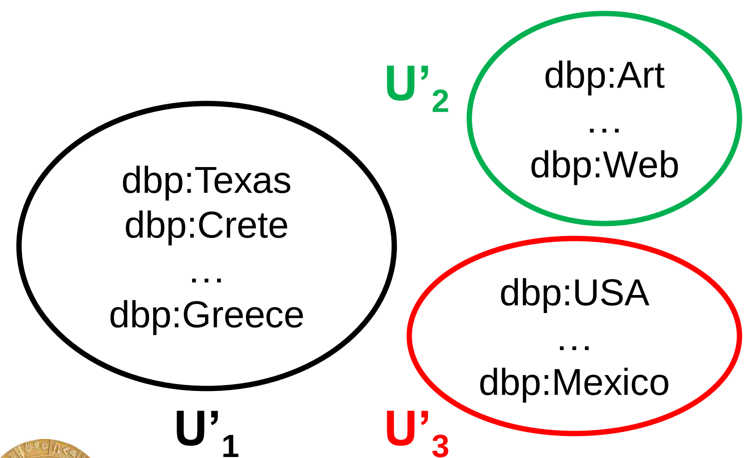
- Let see how the different combinations of the sequence dataset IDs for a prefix affect the number of ASK queries!

Prefix	dataset ID
http://data.nytimes.com/ (**nyt**)	1
http://en.wikipedia.org/ (**en_wiki**)	2,3,4,1
http://dbpedia.org/ (**dbp**)	1,3,2
http://de.wikipedia.org/ (**de_wiki**)	1,4
http://yago-knowledge.org/ (**yg**)	3
http://www.geonames.org/ (**geo**)	3,4

- U_p={ u ∈ U_i |namespace(u)=p } and U_i'=U ∩ U_p

- In the worst case, for each pair D_i, D_j, we should send an ASK query from D_i to D_j for all the U_i' in order to check if a URI exists in U_j'.

U_i'	**Freq. of** p
U_1'	1,000,000
U_2'	5,000
U_3'	10,000



Position 0,1,2	ASKs
D_1,D_2,D_3	2,005,000
D_1,D_3,D_2	2,010,000
D_2,D_1,D_3	1,010,000
D_2,D_3,D_1	20,000
D_3,D_1,D_2	1,020,000
D_3,D_2,D_1	25,000

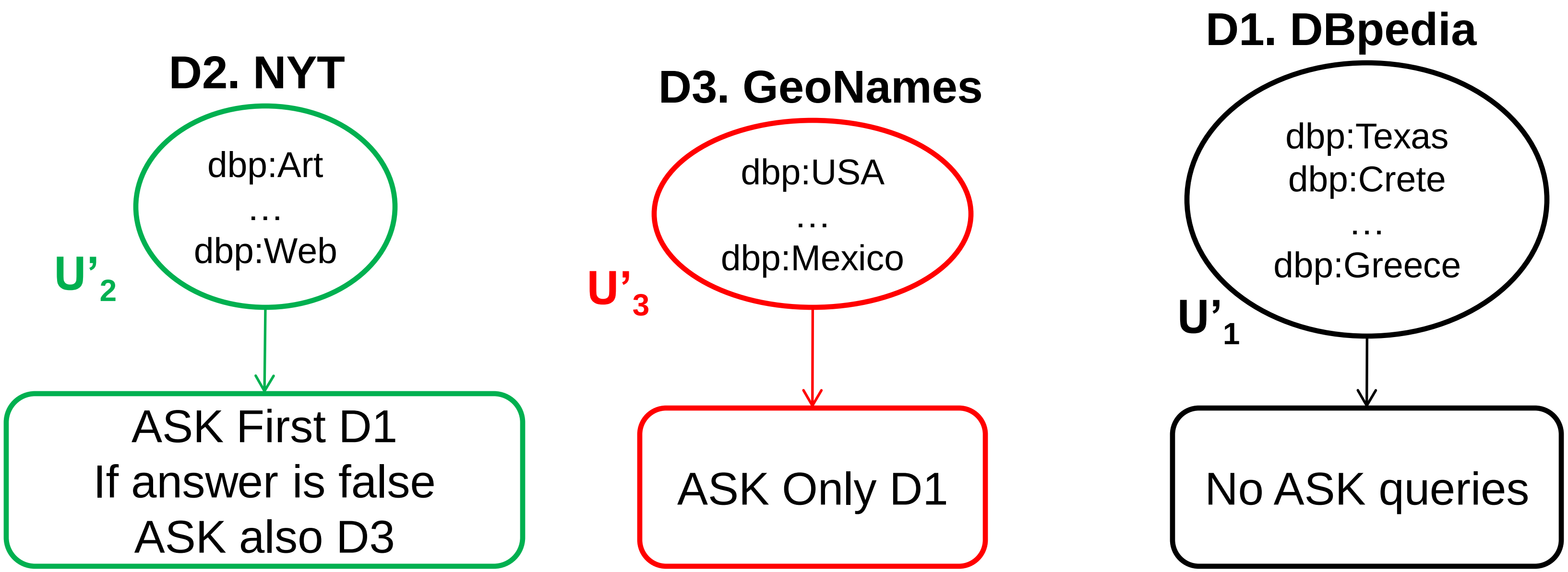
Asks=2*1,000,000+5,000

Asks=2*5,000+10,000

The order that we follow in Prefix Index!

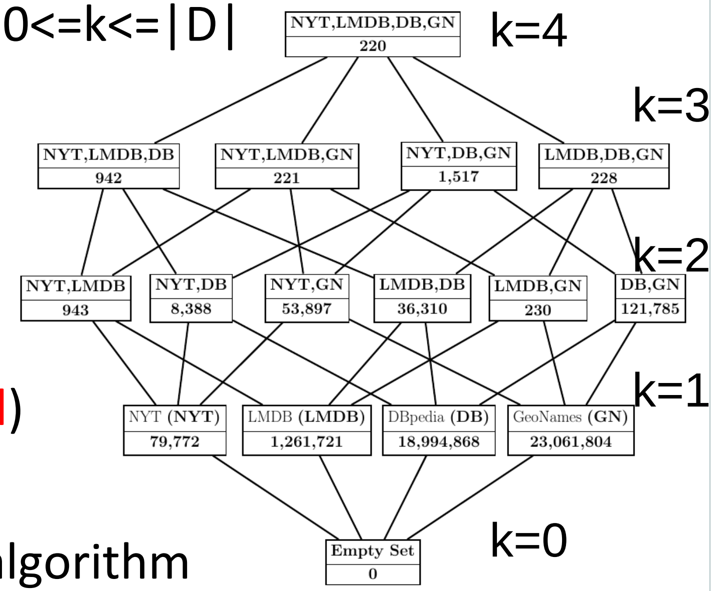
Ordering the Prefix Index for Reducing the ASK Queries (cont.)

- The proposed order reduces the number of ASK queries
  - In the worst case this order gives the minimum number of queries
  - We send queries to other datasets starting with the biggest one
    - This makes more possible to the answer of the first query to be true



The **Lattice of Measurements**

- **What it is:** A **lattice** is a partially ordered set which can be represented as a Directed Acyclic Graph (DAG) where the edges points towards the direct supersets.
- A lattice of |D| datasets contains k levels where 0<=k<=|D|



- **Rationale**: For speeding up the computation of the intersection of rwo of all the subsets.
  - We describe two more efficient (incremental) methods based on set theory properties:
    - A **top-down** lattice-based incremental algorithm
    - A **bottom-up** lattice-based incremental algorithm

Making the Measurements of the Lattice Incrementally-Notations

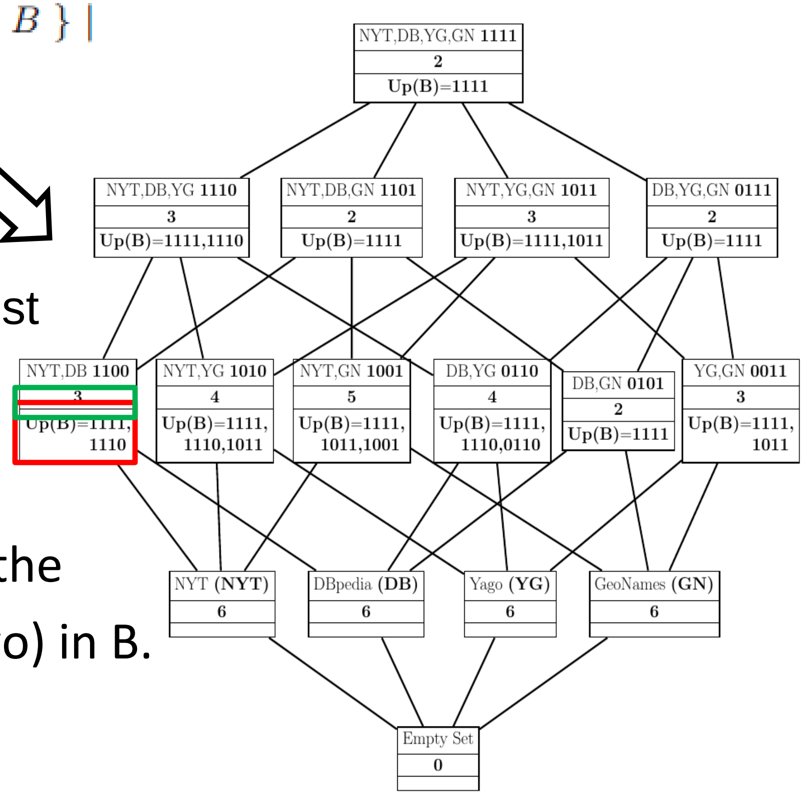
- ***directCount(B):*** frequency of subset B in the element index

$$directCount(B) = |\ \{\ u\ \in\ Left(ei)\ |\ ei(u) = B\ \}\ |$$

URI or SameAs ID	ID Number	Bit Array
1 (SameAs ID)	1,2,3	1110
2 (SameAs ID)	2,3	0110
3 (SameAs ID)	1,2,3,4	1111
4 (SameAs ID)	1,2,3,4	1111
de_wiki:USA	1,4	1001
en_wiki:Portland	1,4	1001
en_wiki:san_diego	1,3,4	1011

Subset	dC
1111	2
1110	1
1011	1
1001	2
0110	1

DirectCounts List



- ***Up(B):*** the supersets of B that can be found in directCount List.

$$Up(B) = \{B' \in \mathrm{P}(\mathcal{D})\ |\ B \subseteq B', directCount(B') > 0\}$$

- The sum of the directCount of Up(B) gives the number of common real world objects (rwo) in B.

$$co_{\sim}(B) = \sum_{B' \in Up(B)} directCount(B')$$

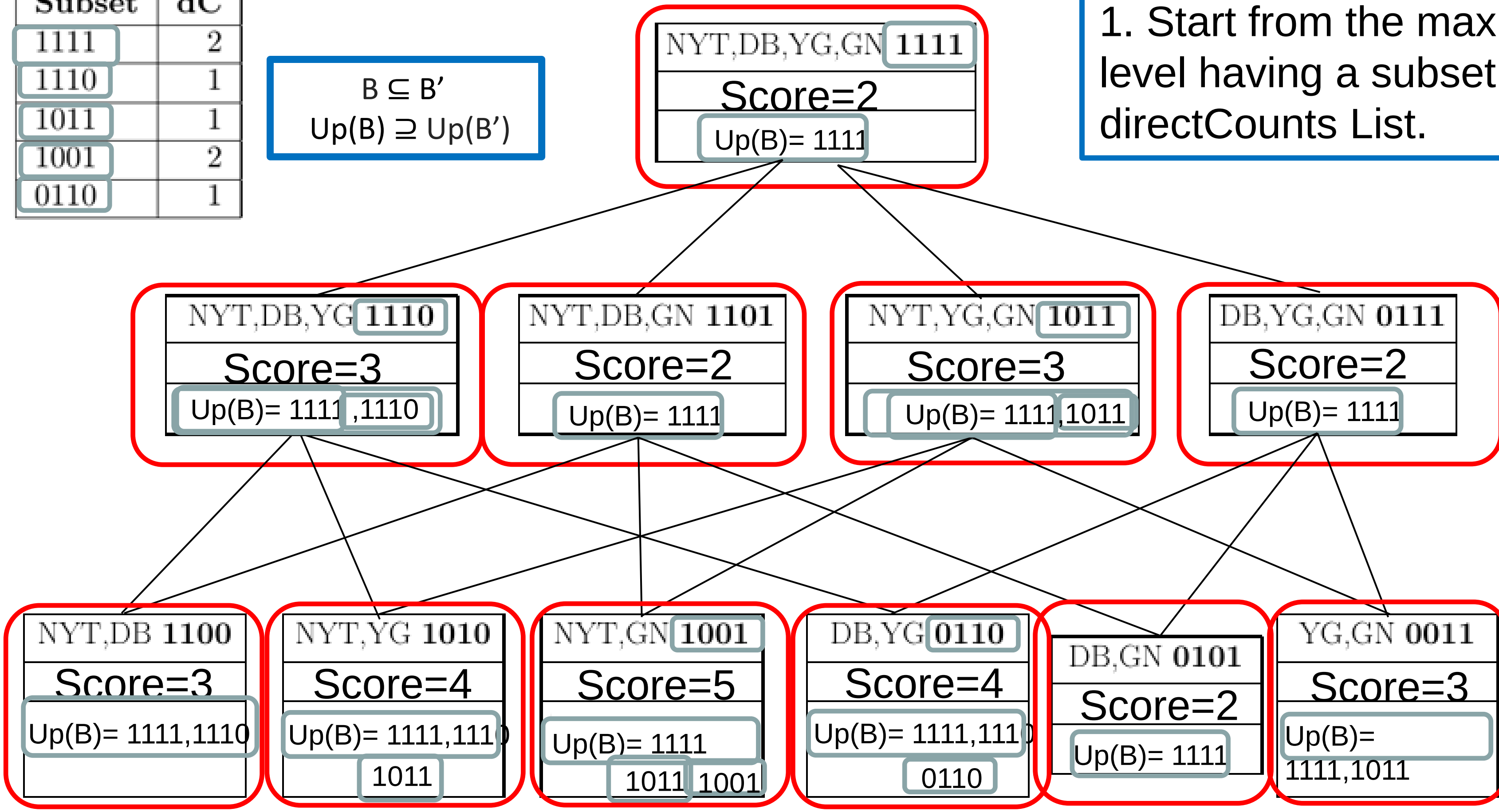
Proposition 2 *Let F and F' be two families of sets. If $F \subseteq F'$ then* $\underset{S \in F'}{\cap S} \subseteq \underset{S \in F}{\cap S}$.

Top-Down Algorithm (BFS Traversal)

Subset	dC
1111	2
1110	1
1011	1
1001	2
0110	1

B ⊆ B’
Up(B) ⊇ Up(B’)

1. Start from the maximum level having a subset in directCounts List.



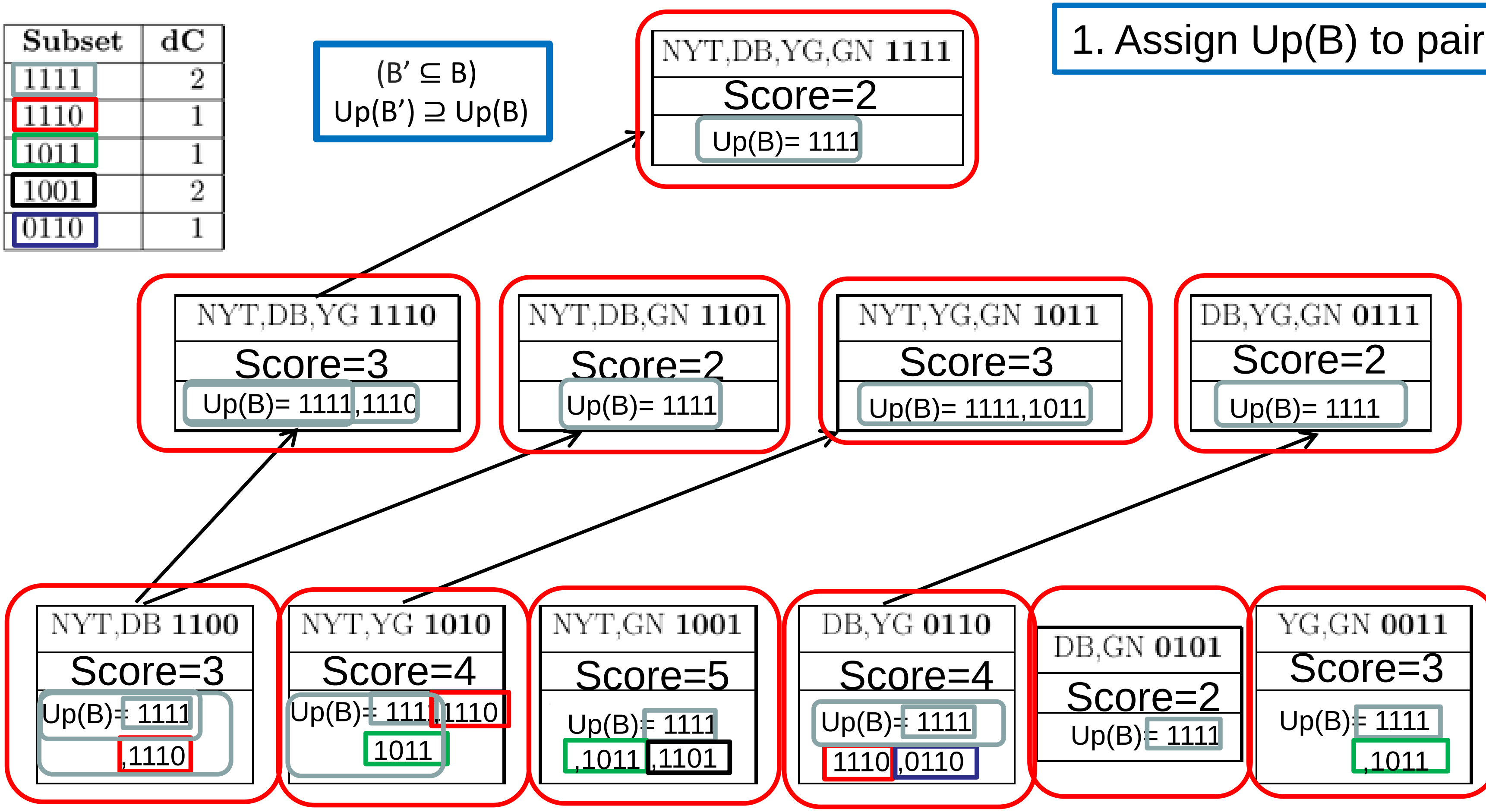
1. For each node check if directCount(B)>0 and Add B to Up(B)
2. Sum the values of directCount of Up(B)
3. Transfer Up(B) to all subsets of B of the previous level since Up(B) ⊇ Up(B’) (B ⊆ B’)

Bottom-Up Algorithm (DFS Traversal)

Subset	dC
1111	2
1110	1
1011	1
1001	2
0110	1

(B′ ⊆ B)
Up(B′) ⊇ Up(B)

1. Assign Up(B) to pairs



1. Sum the directCount of Up(B)
2. Assign the Up(B′) of each superset B′ of the next level if it has not visited yet and then visit B′.
3. Check which Up(B) goes to Up(B′) since Up(B′) ⊇ Up(B) (B′ ⊆ B)

Top-Down versus Bottom-Up Approach

	Top-Down	Bottom-up
Nodes	V	V
Edges	E	V
Time complexity	O(V+E)	O(V)
Space Complexity	$O(V_k)$ $V_k = \binom{\lvert D\rvert}{k} = \frac{\lvert D\rvert!}{k!(\lvert D\rvert-k)!}$	O(d) d:diameter of graph
Additional cost	**Extra edges**= $(\lvert D\rvert-2)*2^{(\lvert D\rvert-1)}$	**checkCost=** $\lvert \sum_{i=1}^{C} 2^{bits_1(B_i)} \rvert$
Its is faster when:	Extra edges<checkcost	Extra edges>checkcost

- ***Additional checkCost***: In the bottom-up approach, we check each time which of the Up(B) belong to Up(B'). For a subset B_i, that belongs in directCount list:

C:size of list 0.1 % of Element Index Size

Subset	dC
1111	2
1110	1
1011	1
1001	2
0110	1

Subset	$bits_1$	Checks
1111	4	2^4
1110	3	2^3
1011	3	2^3
1001	2	2^2
0110	2	2^2

checkCost

$$\lvert \sum_{i=1}^{C} 2^{bits_1(B_i)} \rvert$$

Power Law Distribution of nodes

- ***Proposition***: Group and order in descending order the directCount nodes according to their number of bits (i.e., categories)
  - Categories' frequency follow a power-law distribution ($2 \leq n \leq |D|$)
  - $f(n) = k * (m/2)^{n-2}$: the number of nodes of the n-th category
    - k: the number of such nodes having bits1(B) = 2
      - in our case ($k = |D^2|/4$)
    - m/2 is the reduction factor ($1 < m \leq 2$)
  - **The bottom-up approach is more efficient than top-down when:**

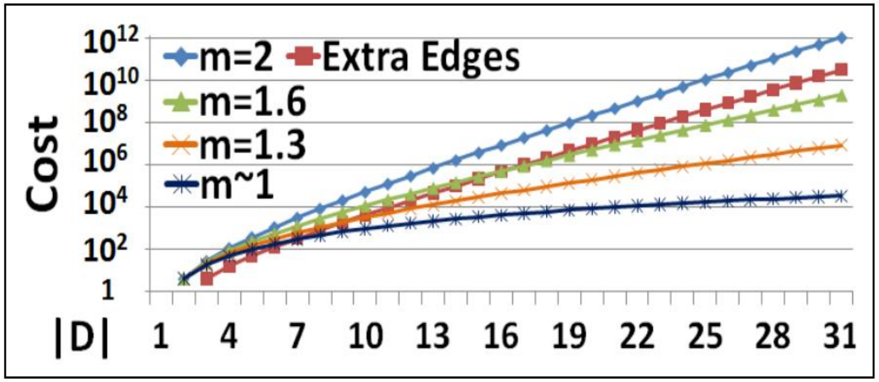
Subset	$bits_1$	Checks
1111	4	2^4
1110	3	2^3
1011	3	2^3
1001	2	2^2
0110	2	2^2

checkCost $\underbrace{|\mathcal{D}|^2 * \frac{m^{|\mathcal{D}|-1}-1}{m-1}}_{\text{checkCost}} < \underbrace{(|\mathcal{D}|-2) * 2^{|\mathcal{D}-1|}}_{\text{Extra Edges}}$ **Extra Edges**

m~1 *:* nodes are reduced by half as categories grow
Bottom-up is better for |D|>6

m=2 *:* each category has the same number of nodes → ***Top-Down is always better***

m=1.6 *:* nodes are reduced by 0.8 as categories grow
Bottom-up is better for |D|>16



Power Law Analysis

Experimental Evaluation

LOD Cloud Experiments- **Indexes**

- We collected **300 LOD Cloud Datasets** from **9 domains.**
  - 658 millions of triples, 172 millions of URIs, 13 millions of sameAs pairs!
  - Only 2.3% of rwo exists in three or more datasets (4% in two or more)
  - The impact of the **closure** was incredible!
    - 19 millions of newly discovered owl:sameAs pairs!
    - 2,393 of newly discovered connected pairs of datasets!
  - We calculated **billions of nodes** with the bottom-up approach in half-an-hour!

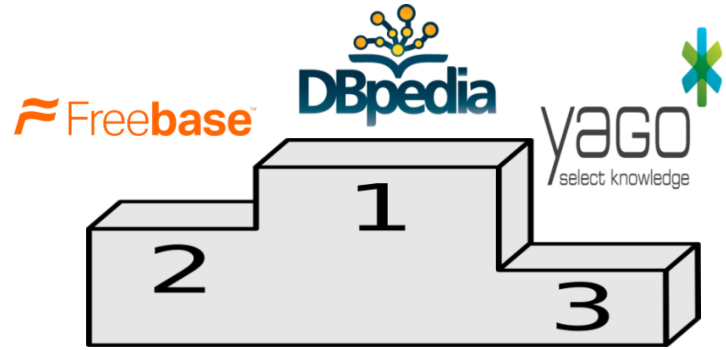
Domain	$\|\mathcal{D}\|$	|Triples|	|URIs|
Cross Domain (**CD**)	19	293,129,862	103,281,343
Geographical (**GEO**)	14	155,591,494	34,169,442
Life Sciences (**LF**)	17	66,684,349	9,725,521
Government (**GOV**)	45	61,189,128	6,896,850
Publications (**PUB**)	76	53,930,138	10,932,689
Media (**MED**)	9	15,267,271	4,434,038
Linguistics (**LIN**)	8	9,128,072	2,059,465
Social Networking (**SN**)	96	2,451,093	561,686
User Content (**UC**)	16	1,059,255	308,193
All	300	658,430,662	172,369,227

Category	Value
Prefix Index Size	63,803
Unique Real World Objects	141,269,960
Element Index Size (*rwo*)	6,242,344
Element Index Size (URIs)	17,840,499
Asks Number	6,684,242
rwo in 3 or more D_i	3,293,248
URIs corresponding to *rwo* in 3 or more D_i	12,296,650
Num. of Lattice Nodes (threshold $>$ 30)	130,525,631
Num. of Lattice Nodes (threshold $\geq$ 20)	1,541,968,012

Category	Value
SameAs Triples	13,158,621
SameAs Catalog Size	18,789,593
SameAs Triples Inferred	19,450,107
Pairs sharing at least 1 real world object	6,708
New Pairs discovered due to SameAs Alg.	2,393
Triads sharing at least 1 real world object	74,432
New Triads discovered due to SameAs Alg.	48,658
SameAs Unique IDs	6,218,958

LOD Cloud Experiments - **Most Connected Subsets**

- The **triad** of the popular Cross domain Datasets shares **2.7 millions of real world objects (rwo)**!
- The **quad** of the four popular cross domain datasets share **1.4 millions of rwo**.
- Most connected triads contain **cross domain and geographical datasets**!
- **Check** the **paper** for finding more experiments.



Datasets of subset B	$co_{\sim}(B)$
1: {DBpedia,Freebase,Yago}	2,709,171
2: {DBpedia,Freebase,Wikidata}	1,950,319
3: {DBpedia,Yago,Wikidata}	1,435,713
4: {Yago,Freebase,Wikidata}	1,434,407
5: {DBpedia,Yago,Freebase,Wikidata}	1,434,404
6: {DBpedia,GADM,Freebase}	107,968
7: {DBpedia,GeoNames,Freebase}	98,985
8: {DBpedia,GADM,Wikidata}	96,968
9: {GADM,Freebase,Wikidata}	96,968
10: {DBpedia,GADM,Freebase,Wikidata}	96,968

Top-10 Subsets ≥ 3 with the most common rwo

Dataset D_i		rwo in ≥ 3 D_i	(% of D_i rwo)
DBpedia		3,246,415	17.3%
Freebase	Cross-	3,237,604	11.3%
Yago	Domain	2,712,930	48.0%
Wikidata		1,952,222	7.3%
GADM Geovocab	Geo	108,503	9.4%
GeoNames		102,747	0.4%
d-nb.info	Pub	65,076	4.2%
LinkedGeoData (LGD)		43,265	0.6%
Opencyc		34,313	26%
LMDB	Media	30,225	2.3%

Top-10 datasets with the most rwo existing at least in 3 datasets

Comparative Results – ***Prefix Index***

- 89 % of distinct prefixes exist in 1 Dataset!
- However it concerns **only** the 10.8% of URIs
  - 16,689,866 URIs ignored
- 11 % of prefixes concern the 89.2% of URIS
  - Few prefixes are very popular!!
- We send 6.68 Million ASK queries
  - 1 ASK query per 19 URIs having a prefix that can be found in two or more datasets
  - **The optimized sequence was the key point for the above number!!**

http://dbpedia.org

http://yago-knowledge.org

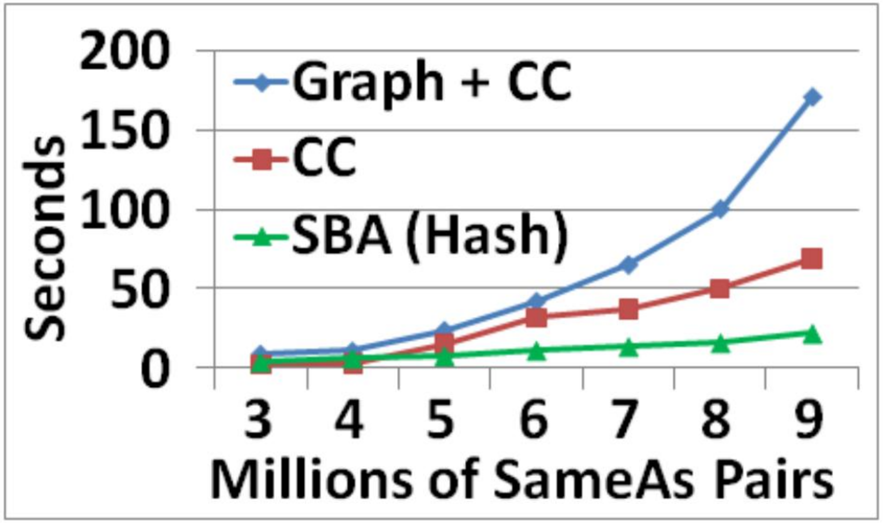
http://rdf.freebase.com

http://en.wikipedia.org

http://wikidata.org

Comparative Results – ***SameAs Catalog***

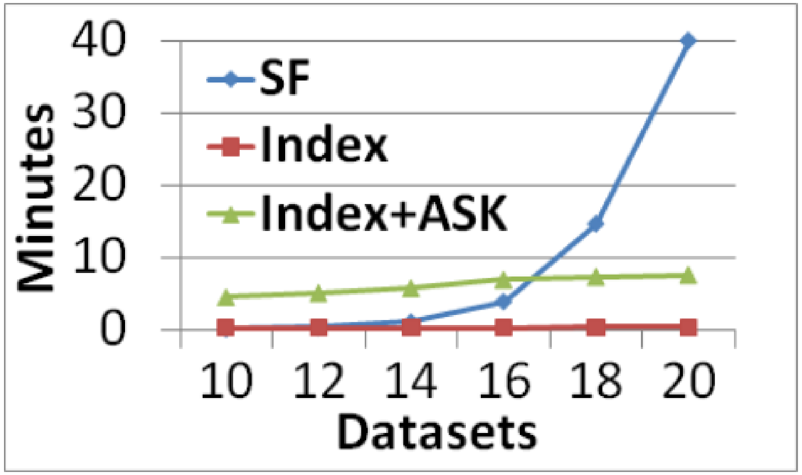
- The signature-based Algorithm is always faster than the Tarjan's algorithm[7] plus the creation of Graph!
- It was infeasible to load in memory the graph for more than 10 million pairs
- The SBA algorithm computed the closure of more than 13 million pairs in 45 seconds!!



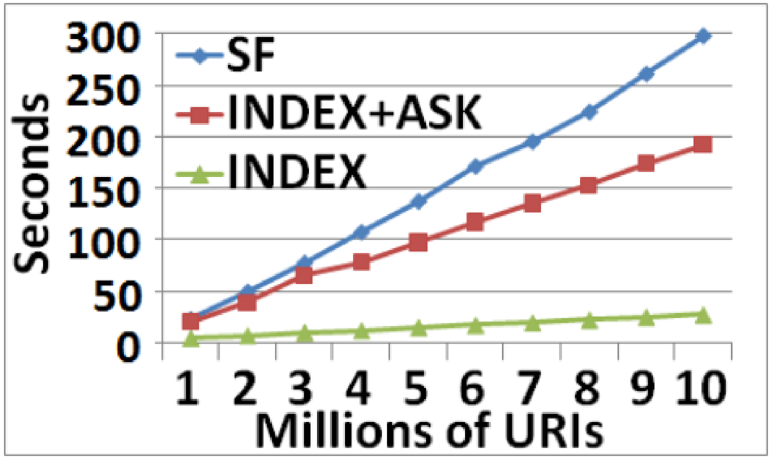
Comparative Results–*Index vs Straightforward method*

Here we compare the index approaches with the straightforward (**SF**) method with data that fit in memory!

- **Index Approach (without ASK queries)** is always faster.
- Time of **SF** method increases exponentially as dataset grows and linearly as URIs grows.
- **Index Approach with ASK Queries** increases linearly when URI grows.



Comparison of different approaches
Varying number of Datasets

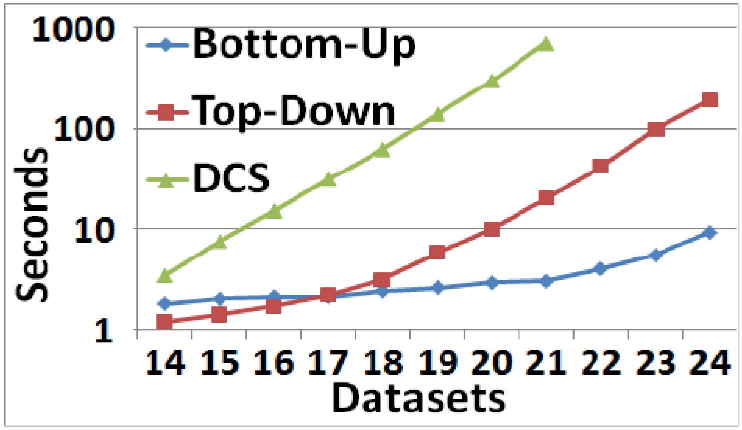


Comparison with stable |D| = 17
Varying number of URIs

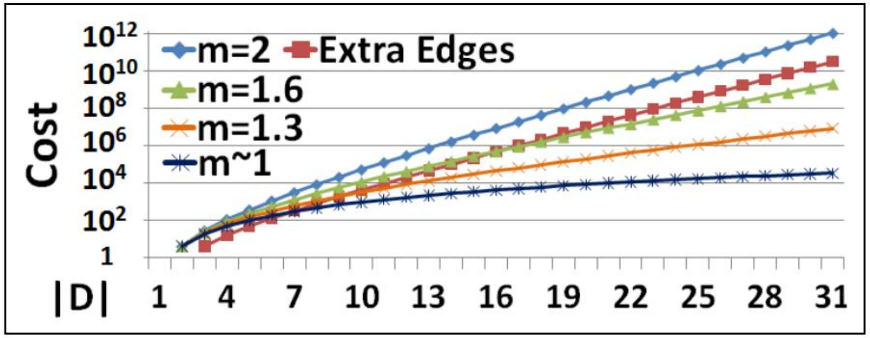
Comparative Results – ***Lattice Algorithms***

We compare the performance of the lattice incremental algorithms and the directCount scan approach (**dcs**).

- **Size of directCounts list**: 1,000 nodes (from 4,000,000 real world objects).
- Both incremental approaches are **always** faster than the **dcs** approach.
- For 17 or more datasets (**like in our power-law analysis**), bottom-up approach is faster than the top-down.
- For more than 24 datasets, it was infeasible to run top-down algorithm while with the bottom-up we can compute more than 1 billion nodes in 30 minutes!



Execution Time of Lattice creation

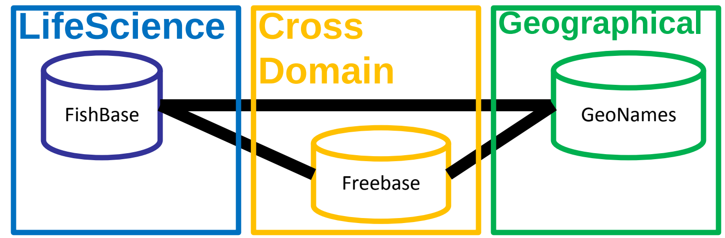


Power Law analysis

Our Website: ***LODsyndesis* -** Data Discovery & Entity Lookup

Number	Query
Query 1	Give me the top-K most connected datasets to my dataset
Query 2	Give me all the connected sources with FishBase, and how many URIs these datasets share with datasets from the geographical domain
Query 3	Give me all pairs of sources that were not connected, but now they are connected due to closure and the number of their common RWO
Query 4	Give me the increase of the commonalities of all pairs of sources due to closure in descending order
Query 5	Give me the K datasets that maximize the pluralism factor of the entities in my dataset
Query 6	Give me the connected triads from datasets coming from three different domains

Dataset Discovery



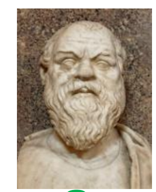
Give me Triads from three different domains

Number	Query
Query 7	Give me all the datasets that contain information about http://dbpedia.org/resource/Aristotle
Query 8	Give me all the equivalent URIs of http://dbpedia.org/resource/Aristotle
Query 9	Give me all the datasets from the publication domain containing information about yago:Socrates
Query 10	Give me all the URIs that are equivalent with the URIs of my dataset
Query 11	Give me the datasets that contain information for both Aristotle and Socrates
Query 12	Give me all the common RWO between Wikidata, DBpedia and Yago

Object Coreference

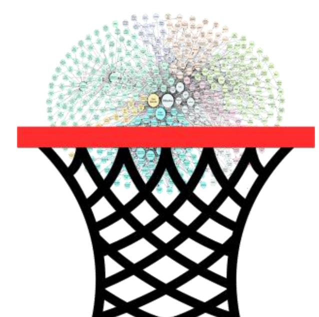
Give me Datasets

For



Yago:Socrates

- Prototypes that exploits the measurements can be found in
  - http://www.ics.forth.gr/isl/LODsyndesis/
- It contains:
  - A link to datahub (publish, share, discover) where we published the results
  - A link to a 3D Visualization: www.ics.forth.gr/isl/3DLod/
  - A list of Answerable Queries and a link to an active **SPARQL Endpoint**



Conclusion

- We introduced indexes and algorithms for
  - Assessing the **degree of connectivity of two or more sources**
  - **Obtaining information** about a **particular entity**
  - Discovering **relevant datasets**
  - **Visualizing the degree of connectivity** of two or more sources
- We reported **measurements that have never been carried out in the past**
- We discussed the **speedup** obtained by the proposed indexes & algorithms
- We introduced **novel systems** that **exploits** the proposed measurements.

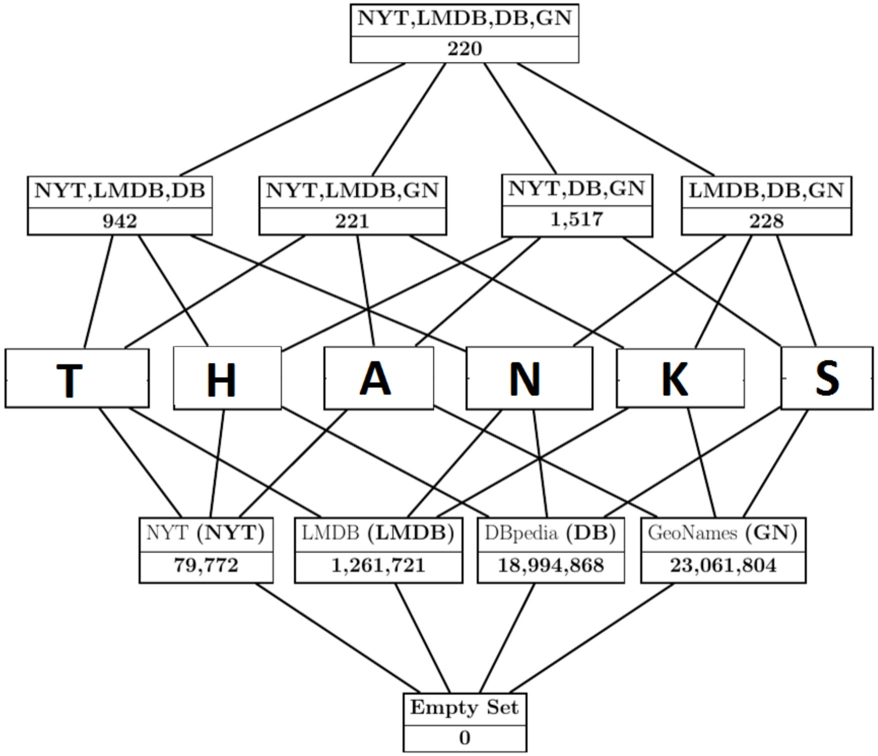
Future Work

- **Generalize** the **lattice based approach** for **other RDF Features**
  - E.g., Literals, Triples

- **Parallelize** the approach by using **Map Reduce Techniques** for
  - investigating the **speedup** that can be achieved
  - **running** the **experiments** for
    - **Billions** of Triples, URIs & Literals
    - More **sameAs** relationships
    - Even more **datasets** (identically for the whole **LOD Cloud**)

- Exploit the measurements for **better visualizations** and **monitoring services**

Thank you!


NYT,LMDB,DB,GN
220
NYT,LMDB,DB
942
NYT,LMDB,GN
221
NYT,DB,GN
1,517
LMDB,DB,GN
228
T
H
A
N
K
S
NYT (NYT)
79,772
LMDB (LMDB)
1,261,721
DBpedia (DB)
18,994,868
GeoNames (GN)
23,061,804
Empty Set
0


Questions

Acknowledgements

This work was partially supported by: BlueBRIDGE

References

[1] Laurens Rietveld, Wouter Beek, and Stefan Schlobach. Lod lab: Experiments at lod scale. In Proceedings of the International Semantic Web Conference (ISWC). Springer, 2015

[2] Markus Nentwig, Tommaso Soru, Axel-Cyrille Ngonga Ngomo, and Erhard Rahm. Linklion: A link repository for the web of data. In The Semantic Web: ESWC 2014 Satellite Events, pages 439–443. Springer, 2014.

[3] A. Harth, J. Umbrich, A. Hogan, and S. Decker. YARS2: A federated repository for querying graph structured data from the web. In ISWC, volume 4825, pages 211–224, 2007.

[4] L. Ding, T. Finin, A. Joshi, R. Pan, R. S. Cost, Y. Peng, P. Reddivari, V. Doshi, and J. Sachs. Swoogle: a search and metadata engine for the semantic web. In Proceedings of CIKM, pages 652–659, 2004.

[5] T. Neumann and G. Weikum. The RDF-3X engine for scalable management of RDF data. The VLDB Journal, 19(1):91–113, 2010.

[6]] E. Prud' Hommeaux, A. Seaborne, et al. SPARQL query language for RDF. W3C recommendation, 15, 2008.

[7] Robert Tarjan. Depth-first search and linear graph algorithms. In Twelfth Annual Symposium on Switching and Automata Theory, pages 114–121. IEEE, 1971.

Links To Our Tools & Services

- ***LODsyndesis***: http://www.ics.forth.gr/isl/LODsyndesis/
- ***Datahub***: https://datahub.io/dataset/connectivity-of-lod-datasets
- ***Demo Queries***: http://62.217.127.118:8890/fct/demo_queries.vsp
- ***SPARQL Endpoint for Metrics***: http://62.217.127.118:8890/sparql
- ***3DLod***: www.ics.forth.gr/isl/3DLod/

Top-Down Algorithm (BFS Traversal)-Analysis

- *We avoid passing from nodes having score=0.*
  - We can start from the maximum level having a node with score>0.
  - We can create nodes only when their |Up(B)|>0
- ***Time complexity is O(V+E)***
  - It passes from all nodes having intersection value bigger than zero
    - $|V|=2^{|D|}$
  - It creates all the edges E where
    - $|E|=|D|*2^{(|D|-1)}$
- ***Space Complexity is $O(V_k)$***
  - It keeps in memory all the nodes of a specific level since the traversal is BFS
  - k: a lattice level (e.g., pairs, triads, etc.)
  - $V_k = \binom{|D|}{k} = \frac{|D|!}{k!(|D|-k)!}$

How to Compute Specific Parts of Lattice

- Compute a Single node B
  - Scan directCount list once and see whether Bi ∈ Up(B)
- Threshold-based nodes
  - Use bottom up approach since fewer nodes will be created

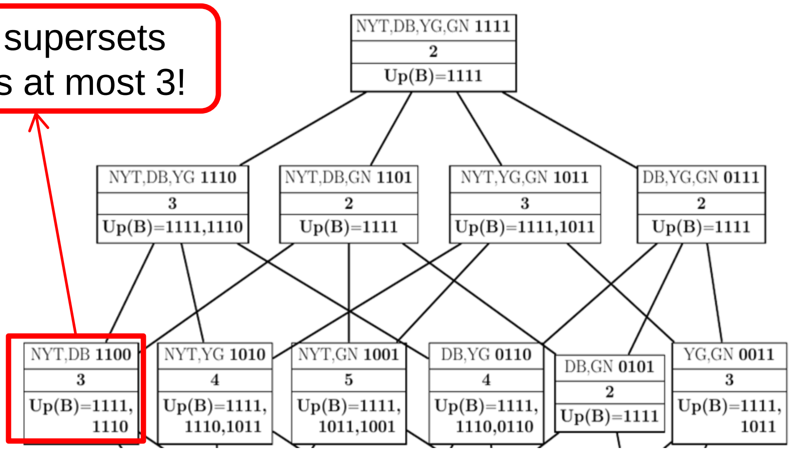
1001

Subset	dC
1111	2
1110	1
1011	1
1001	2
0110	1

Proposition 2 *Let F and F' be two families of sets. If $F \subseteq F'$ then* $\bigcap_{S \in F'} S \subseteq \bigcap_{S \in F} S$.

All its supersets score is at most 3!

- Lattice of a subset of datasets
  - Just scan the index and use the following directCount formula:



$$directCount(B, D') = |\{ u \in Left(ei) \mid B = ei(u) \cap D' \}|$$

- All nodes of a specific dataset
  - Again use an alternative version of directCount formula:

$$directCount(B, D_i) = |\{ u \in Left(ei) \mid ei(u) = B \text{ and } D_i \in B \}|.$$

Bottom Up Algorithm Analysis

- *We avoid passing from nodes having score=0.*
  - *When the score of a subset is 0, then the score of all its supersets is 0, too.*
- ***Time complexity is O(V)***
  - Passes from all nodes having intersection value bigger than zero: $|V|=2^{|D|}$
  - Creates one edge per node :$|E|=|V|$
- ***Space Complexity is O(d)***
  - Follows a Depth First Search (**DFS**) Traversal
  - d: the diameter of the lattice (d is at most |D|)
- ***Additional CheckCost***: However, we should check each time which of the Up(B) belong to Up(B'). For a subset B_i, that belongs in directCount list:

C:size of list 0.1 % of Element Index Size

Subset	dC
1111	2
1110	1
1011	1
1001	2
0110	1

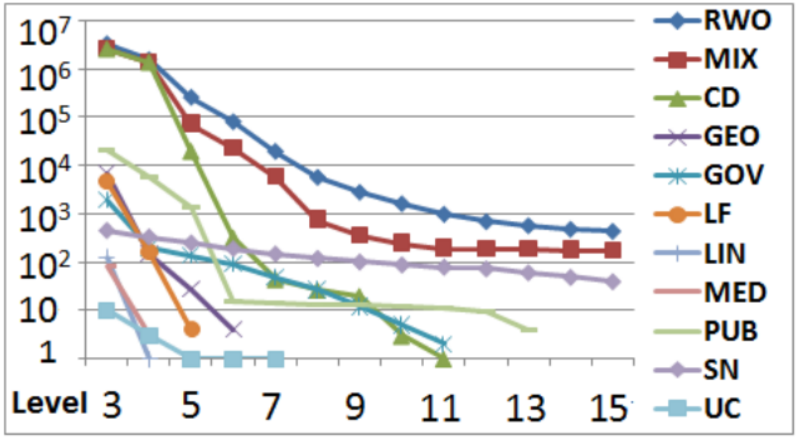
Subset	$bits_1$	Checks
1111	4	2^4
1110	3	2^3
1011	3	2^3
1001	2	2^2
0110	2	2^2

checkCost

$$\left|\sum_{i=1}^{C} 2^{bits_1(B_i)}\right|$$

LOD Cloud Experiments- **Biggest Hubs**

- Which are the biggest hubs in LOD Cloud?
  - Cross Domain Datasets are the most popular.
  - Geographical and Publications datasets follow.
  - **Most connected domains**:
    - Levels 3-6: cross-domain
    - Levels 7-15: social networking domain
  - **15 datasets** share more than **100 rwo!**



Max Subset per Domain for each level