Services for **Connecting** and **Integrating Big Number** of **Linked Datasets**

Michalis Mountantonakis

Computer Science Department,
University of Crete, Greece

PhD Defence

Heraklion, March 27, 2020

Funded by
H.F.R.I. Hellenic Foundation for Research & Innovation

Outline

- Motivation (10 min)

- Related Work (5 min)

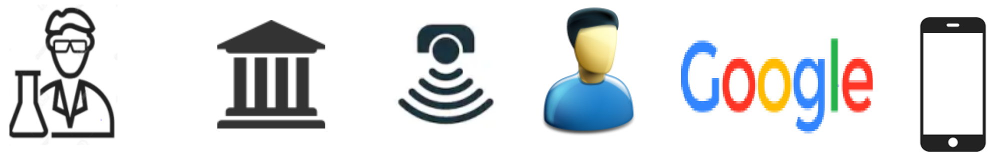
- Contributions (37 min)
  - Cross-dataset Identity Reasoning (6 min)
  - Semantics-aware Indexes at Global Scale (7 min)
  - Content-based Metrics for Dataset Discovery (20 min)
  - The LODsyndesis suite of Services (4 min)

- Conclusion (3 min)
  - Synopsis of Contributions
  - Directions for Future Research

Motivation

General Objective

- Almost everyone and everything produces and needs **data**



- Thousands of **RDF datasets** have been published (over 10,000)!

- The ultimate **objective** of Linked Data is **linking** and **integration**
  - Both are important for fulfilling the **requirements** of **e-science**
    - One of the **biggest challenges** in Computer Science

- The **processing** and the **analysis** of a large volume of integrated data is **crucial** for any scientific field
  - for providing **novel** and **accurate** scientific results

General Problems

- However data and information are not integrated

- *Michael Stonebraker* (a pioneer researcher in data management): "**Data integration at scale** is a very big deal and probably the **biggest problem** that many enterprises face, since the traditional approaches cannot scale easily to more than 25 sources."

- Mark Scrieber: "Data scientists spend even 95% of their time on **Data Discovery** and **Data Integration**"

- **Google** Research Group: "**Integration process** still requires a number of difficult and costly steps"

- But why is Data Integration so difficult?

Why Integration is difficult?

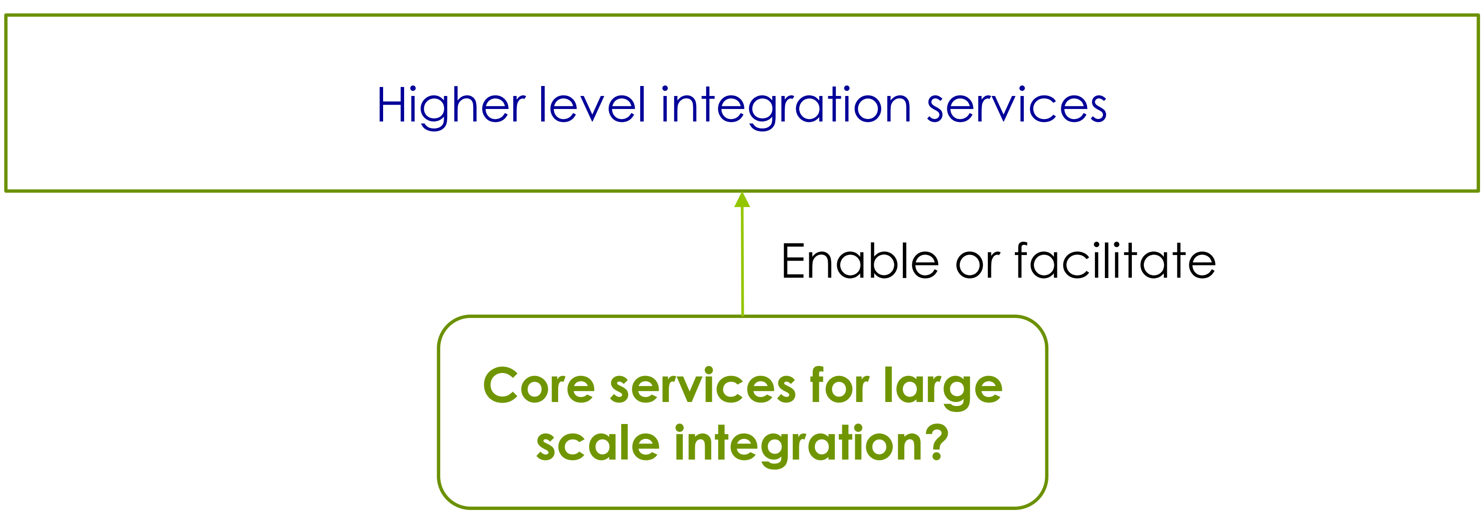
The main difficulties follow:

- **Different Authorities**: Datasets are produced by different organizations in different formats, schemas, models, and systems
- **Naming:** The same real world entities or relationships are referred with different URIs and names, and in different natural languages (and natural languages have synonyms and homonyms)
- **Complementarity**: Datasets contain complementary information
- **Errors/Conflicts:** Datasets contain erroneous, out-of-date or conflicting data
- **Different Conceptualizations:** Datasets may follow different conceptualizations of the same domain
- **Evolution:** Everything changes fast

Related Problems & Analysis

- Due to these difficulties, the **execution** of various tasks related to Data Integration at **large scale** is not so easy

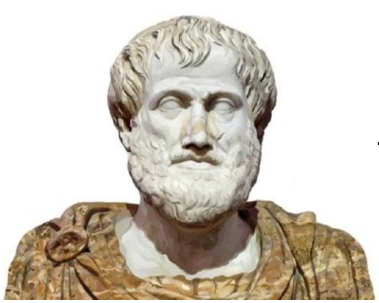
- Our **target** is to propose **advanced methods** for providing fast connectivity services, as core services
  - for enabling various higher level Data Integration services



Motivation

Core Services: Object Coreference & All Facts about an Entity

- Suppose that we want to find all the **available information** (and URIs) about an **entity**, but we know only one URI
  - owl:sameAs: a **symmetric** and **transitive** property connecting **two URIs** that refer to the **same entity**
  - http://dbpedia.org/resource/Aristotle **owl:sameAs** http://yago-knowledge.org/resource/Aristotle
- It is not trivial to find all the **equivalent URIs** with the desired URI
  - the symmetric and transitive closure of owl:sameAs relationships must be computed
  - it presupposes knowledge from all the datasets



Before Closure

1 URI

http://dbpedia.org/resource/Aristotle

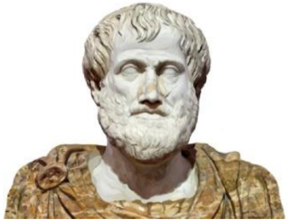
After Closure

31 URIs!!!

equivalentURI
http://es.dbpedia.org/resource/AristÃ³teles
http://nl.dbpedia.org/resource/Aristoteles
http://data.bnf.fr/ark:/12148/cb13091331s
http://bnb.data.bl.uk/id/person/Aristotle
http://yago-knowledge.org/resource/Aristotle
http://fr.dbpedia.org/resource/Aristote
http://d-nb.info/gnd/118650130

Core Services: All Facts about an Entity & Data Veracity

- Equivalence relationships also occur in Schema Level.
  - dbp:Aristotle | dbp:birthDate | "384 BC"
  - yago:Aristotle | yago:dateOfBirth | "384 BC"
  - test:Aristotle | test:birthDate | "383 BC"



Different URIs for the same **Entity**

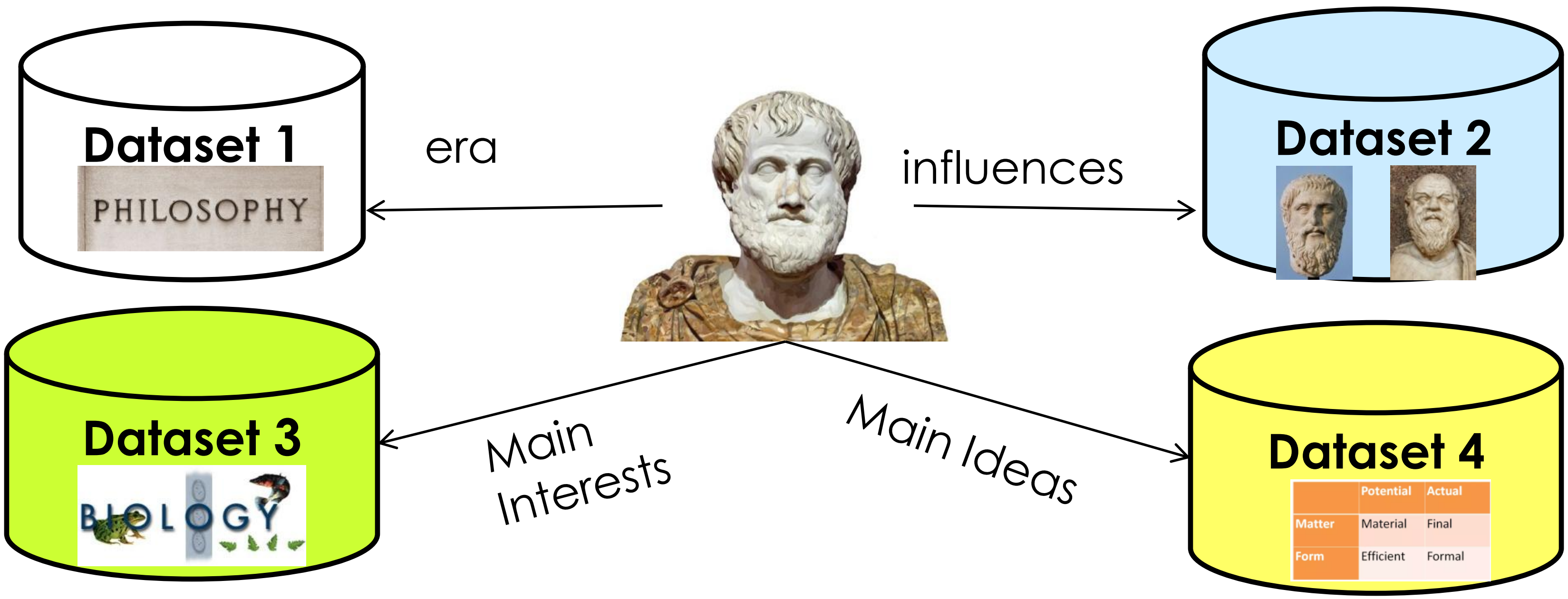
Different URIs for the same **Property**

- **Closure in schema level**: Crucial for collecting all the values for a fact
  - dbp:birthDate ≡ yago:dateOfBirth ≡ test:birthDate (**owl:equivalentProperty**)

- Now, we can easily **compare values** from different datasets
  - Aristotle birthDate "384 BC" → Provenance: Yago, DBpedia
  - Aristotle birthDate "383 BC" → Provenance: Test

Core Services: Data Enrichment & Quality

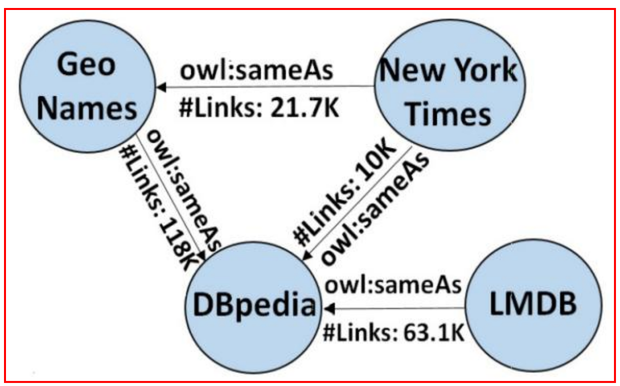
- Collecting information for the **same entity** from **many datasets**
  - offer **complementary** information for a URI
  - can **verify** or **clean** that information for producing a more accurate dataset
  - can improve **machine learning** based tasks

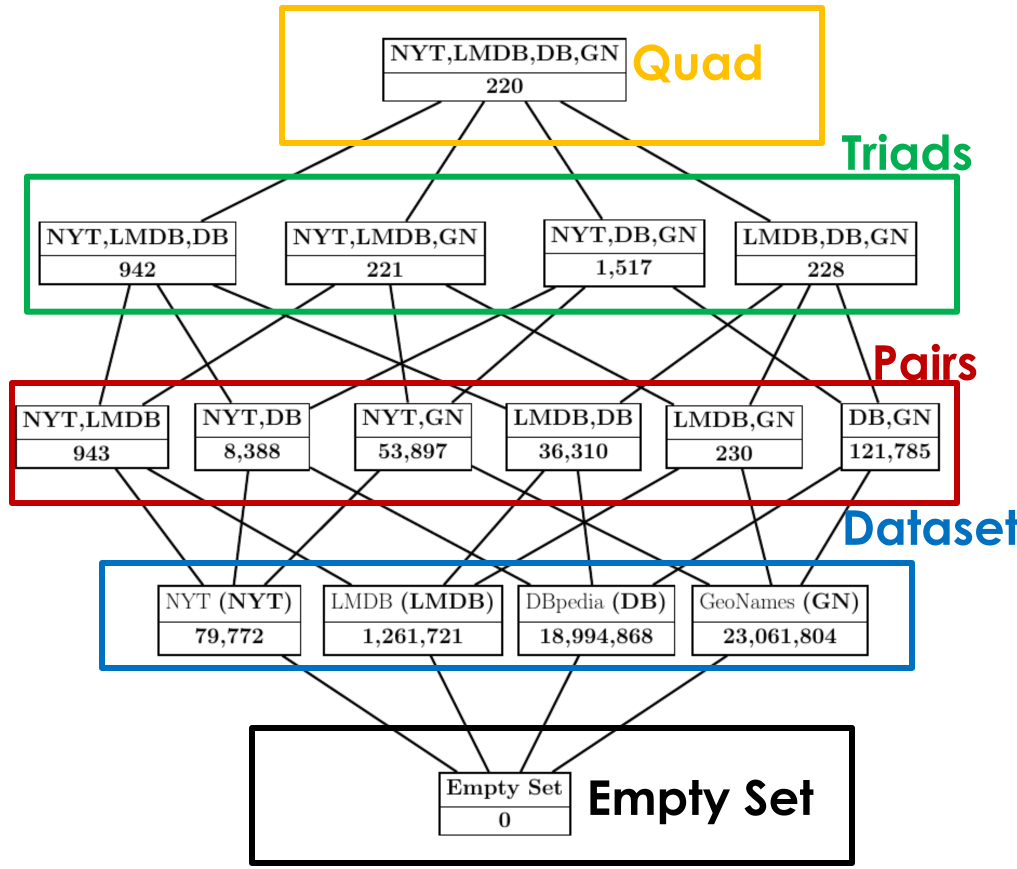


Core Services: Connectivity Analytics

- It is difficult to understand how **connected** the **LOD cloud** is!
- Only measurements between pairs of datasets are available!
- It is **not possible** to see how many **common entities** exist among three or more sources!

Current Visualizations stops in Pairs level (!!!)



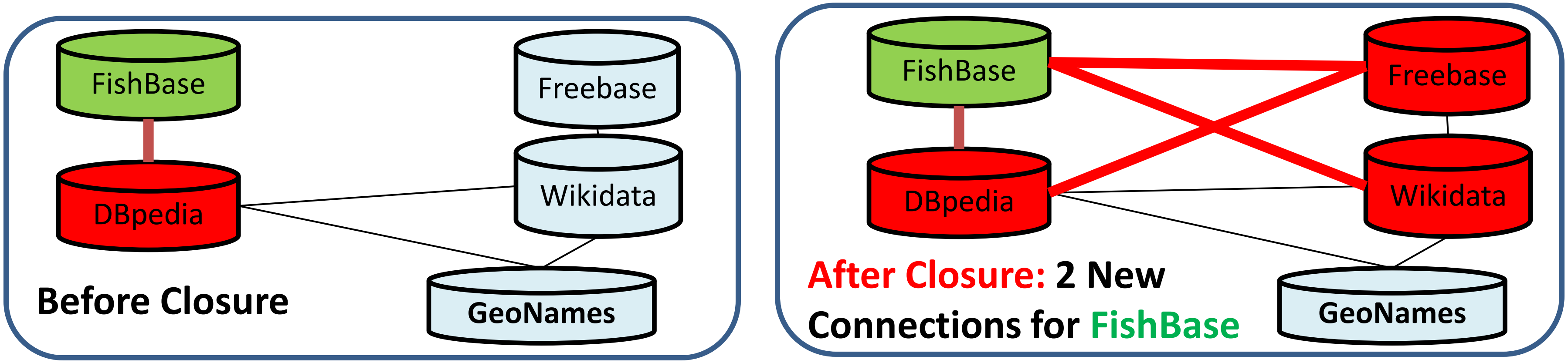


Core Services: Dataset Discovery - Impact of Closure

- Suppose that we publish a **dataset** and we create **relationships** with **DBpedia**.

- We want to find the **K most related datasets** to our dataset:
  - (a) for constructing a semantic warehouse
  - (b) for mediator-based query answering.

- With the **proposed approach** (including the computation of transitive closure), we could get **much more datasets**!!!



Core Services: Dataset Discovery

- Two scientists desire to find **5 datasets** (from 12 available ones) about endangered species
  - There are 792 possible quintets of datasets!
  - Time-consuming to check all these **possible quintets**
- Current Metadata Engines
  - do not use the contents of datasets
  - return the same ranking list of **single datasets** (e.g. for both scientists)

"I want **5 Datasets** having the **most common species** with **my dataset**"

Scientist 1

"I want **5 Datasets** whose **union** contains the **maximum number** of **endangered species**"

Scientist 2

Rank	Dataset
1	D2
2	D5
3	D10
4	D3
5	D12
…	…
11	D7
12	D9

But is this the best quintet for both scientists?

Core Services: Dataset Discovery (cont.)

- **Target**
  - Retrieve a **ranking list of quintets of datasets**, by using the **contents** of datasets
  - The ranking is **different** for each scientist according to their **requirements**
- **Different combinations** of datasets can have different quality and value for **different users** even for the same task!

"I want **5 Datasets** having the **most common species** with **my dataset**"

Scientist 1

Rank	Quintet of Datasets	Common Species
1	**D2,D3,D4,D6,D8**	**150**
2	D1,D2,D3,D6,D9	**145**
...	...	....
792	D1,D5,D6, D10,D12	2

Intersection Metrics

Different Ranking

"I want **5 Datasets** whose **union** contains the **maximum number** of **endangered species**"

Scientist 2

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

Union Metrics

Challenges & Research Questions

Challenge 1. Cross-Dataset Identity Reasoning

- Problem: It presupposes **knowledge** of all **datasets** and the computation of closure requires a lot of **RAM memory**
  - Research Question: How to compute in an **efficient way** the **transitive** and **symmetric closure** of equivalence relationships?

Challenge 2. Construction of Semantics-aware Indexes at Large Scale

- Problem: The **result of the closure** should be taken into account for **constructing** the indexes
  - Research Question: How to **apply the result** of the cross-dataset identity reasoning for constructing such **semantics-aware indexes**?

- Problem: There are **many datasets** (hundreds or thousands) and some of them are **very big**
  - Research Question: How to **parallelize** in an efficient way the **construction** of these indexes?

Challenges & Research Questions (cont.)

Challenge 3. Content-based Dataset Discovery among several datasets (maximization problems)

- Problem: The **possible combinations** of datasets is **exponential** in number (very expensive for maximization problems)
  - Research Question: Can a standard W3C **query language** (such as SPARQL) be used for **solving** such problems?

- Problem: **Set operations** (intersection, union, complement) between **large datasets** are quite expensive.
  - Research Question: How can we **reduce** the number of **set operations** between different datasets?
  - Research Question: Can these content-based measurements be **parallelized**?

Contributions

Overview of Semantic Data Integration at Large Scale

- a clear **landscape** of **large scale semantic integration** approaches for better **understanding** the problem and identifying the **open challenges** [ACM Computing Surveys '19]

Cross-Dataset Identity Reasoning and Construction of Indexes

- **scalable methods** and **algorithms** for performing **cross-dataset identity reasoning** and constructing **semantics-aware indexes** at large scale [VLDB '16, JDIQ '18, Information MDPI '18]

Content-Based Dataset Discovery

- **scalable methods** (based on indexes and set theory properties) **for content-based** intersection, union and complement **metrics** over large number of datasets [VLDB '16, JDIQ '18, Information MDPI '18, JDIQ '20]
  - formulated and tackled as **maximization problems**.
- **connectivity analytics** for a big subset of the **current LOD Cloud**

Global Scale Services

- ***LODsyndesis*** offers services for several real world tasks [Heritage MDPI '18]
- ***LODsyndesisML*** and ***LODVEC*** offer **Dataset enrichment** for **Machine Learning** tasks [TPDL '17,MTSR '19]

Related Work

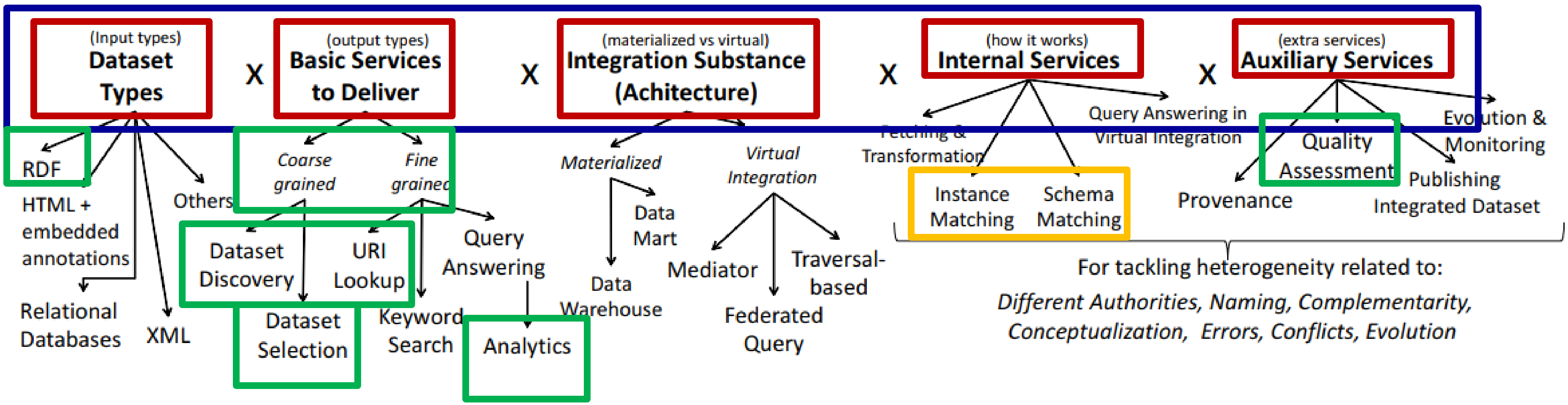
Large Scale Semantic Integration of Linked Data

For analyzing the problem of **Large Scale Semantic Integration of Linked Data** we analyzed the area according to the following aspects:

- Why **Integration** is **Difficult**
- **Data Integration Landscape**
- Traditional **materialized** and **virtual integration** approaches
  - Can these approaches scale to large number of datasets?
- Tools and approaches for each integration step
- **Evaluation** of Integration
- **Semantic Integration** on a **Large Scale**
  - The recent **trend** for **large scale RDF services**
  - Success stories

- Let's see the **key findings** that are related to this **thesis**.

Data Integration Landscape



- This work belongs to the following dimensions for RDF data:
  - ✓ **Mainly to:** Coarse-grained, Fine-grained & Auxiliary Services
  - ✓ **Secondarily to:** Instance and Schema Matching

Traditional Approaches for Data Integration

We analyzed **18 Data Integration tools** using traditional integration methods.

- **Key finding**: They have not been tested for large number of datasets (>20)
  - Materialized systems: Some steps require manual effort → **defining** and **configuring matching** and **transformation rules.**
  - Virtual integration systems: conceptualization, naming and conflicts issues are difficult to be tackled → rely on a **common schema** and do not offer **transformation** and **data fusion** mechanisms.

Tool/ Framework	Integration Substance	Dataset Types	Output Types	Transformations	Schema Matching	Instance Matching	VQA	Provenance Levels	Quality	Evolution	Tested |D|	Tested |T|
LDIF [218]	Materialized	RDF	Any	LT	PD+OMT	PD+IMT	✗	CL,UVL,TL	DF	S-Aut.	1-9	B
ODCleanstore [132]	Materialized	RDF	Any	LT	PD+OMT	PD+IMT	✗	CL,UVL,TL	DF	S-Aut.	1-9	M
MatWare [238]	Materialized	RDF+O	Any	LT, FT	PD+OMT	PD+IMT	✗	CL,UVL,TL	Con.	S-Aut.	1-9	M
KARMA [133]	Materialized	RDF+O	Any	LT, FT	PD+OMT	PD	✗	CL,UVL,TL	DC	S-Aut.	1-9	B
FuhSen [59]	Hybrid	RDF+O	KS	FT	PD	PD+IMT	✓	UVL,QL	DF	S-Aut.	1-9	M
TopFed [212]	Hybrid	RDF	QA	LT, FT	PD+OMT	PD+IMT	✓	UVL,QL	QP	S-Aut.	10-19	B
RapidMinerLOD [204]	Hybrid	RDF+O	Any	LT, FT	PD+OMT	PD+IMT	✓	UVL,QL	DF	Aut.	1-9*	M
SQUIN [113]	Traversal	RDF	QA	✗	PD	PD+C	✓	UVL,QL	QP	Aut.	1-9*	M
SWGET [96]	Traversal	RDF	QA	✗	PD	PD+C	✓	UVL,QL	QP	Aut.	1-9*	M
Linked-Data-Fu [110]	Traversal	RDF+O	Any	✗	PD	PD+C	✓	UVL,QL	QP	Aut.	10-19*	M
SEMLAV [156]	Mediator	RDF	QA	✗	PD+OMT	PD	✓	UVL,QL	QP	S-Aut.	1-9	M
DaRQ [197]	Federated	RDF	QA	✗	PD	PD	✓	UVL,QL	QP	S-Aut.	10-19	M
Splendid [103]	Federated	RDF	QA	✗	PD	PD	✓	UVL,QL	QP	S-Aut.	10-19	B
HiBISCuS [210]	Federated	RDF	QA	✗	PD	PD	✓	UVL,QL	QP	S-Aut.	10-19	B
FedX [219]	Federated	RDF	QA	✗	PD	PD	✓	UVL,QL	QP	Aut.	10-19	B
ANAPSID [28]	Federated	RDF	QA	✗	PD	PD	✓	UVL,QL	QP	Aut.	10-19	B
DAW [211]	Federated	RDF	QA	✗	PD	PD	✓	UVL,QL	QP	S-Aut.	1-9	M
MULDER [85]	Federated	RDF	QA	✗	PD	PD	✓	UVL,QL	QP	S-Aut.	10-19	M

The recent trend for Large Scale Services

Recent **trend** for services over **large number** of RDF datasets

- They can tackle **some integration difficulties** at Large Scale!

- **LODLaundromat [4]** offers fetching and transformation for over 650,000 RDF documents
  - It offers indexes and services for Object Coreference
    - (without cross-dataset identity closure)

- **LOD-a-lot [5]** provides advanced query answering services for the datasets of LODLaundromat
  - (without cross-dataset identity closure)

The recent trend for Large Scale Services (cont.)

- Services for URI Lookup
  - **WIMU [11]** shows all the triples and documents where a URI occurs
    - (without cross-dataset identity closure)
  - **SameAs.org [12]** shows the equivalent URIs of a given one
    - (but not the documents or triples).

- Services for Dataset Discovery & Connectivity
  - **Linklion [6]** provides mappings between pairs of 476 datasets.
  - **LODStats [7]** offers several basic metadata and statistics for over 9,000 datasets, such as the links between pairs of datasets.
  - **LODCloud [8]** diagram shows all the connections between pairs of over 1,200 datasets by exploiting metadata.
  - **Datahub.io** offers a keyword metadata search for thousands of datasets
  - **SPARQLES [9]** and **SpEnD [10]** monitors hundreds of SPARQL Endpoints for checking their healthiness.

Comparing RDF Services for Large in Number Datasets

We can identify **research gaps** in several tasks.

Tool/Service	Total Triples	Include > Datasets	Global URI Lookup	Dataset Discovery	Dataset Visualization	Connectivity	Fetching Transforming	Keyword Search	Dataset Analysis	Querying Datasets	Dataset Evolution
LODsyndesis [161]	2 Bil.	400	✓	✓	✓	✓					
LODLaundromat [203]	38 Bil.	>650,000*	✓	✓			✓	✓			
LOD-a-lot [93]	28 Bil.	>650,000*					✓		✓	✓	
LODStats [88]	130 Bil.	9,960		✓					✓		
Datahub.io	Unk.	>1,270		✓			✓		✓		
LinkLion [174]	77 Mil.	476		✓		✓	✓				
DyLDO [125]	Unk.	86,696*									✓
LODCache	4 Bil.	346						✓		✓	
LODCloud [215]	Unk.	1,239			✓	✓	✓		✓		
sameAs.org [102]	Unk.	>100	✓								
WIMU [241]	Unk.	>650,000*	✓				✓				
LOV [243]	Unk.	637**	✓					✓		✓	
Linghub [147]	Unk.	272		✓				✓		✓	
SPARQLES [244]	Unk.	557		✓	✓						✓
SpEnD [253]	Unk.	1,487		✓	✓						✓

Closure of equivalence relationships is not computed!

Measurements only among pairs of datasets.
Offer Metadata-based Dataset Discovery.

Novelty of Dissertation

Object Coreference & All Facts for an entity

- We offer probably the largest knowledge graph of Linked Data that includes all **inferred equivalence relationships**!

Dataset Discovery & Connectivity Analytics

- It is the first work offering **content-** based **measurements** among any **possible subset** of datasets (not only for pairs, by using metadata [6-10])
  - It returns **ranking lists of multiple datasets** (instead of single datasets)

Data Enrichment & Data Quality

- It is the first work offering **data enrichment** for **machine learning** tasks by using **hundreds of RDF datasets**, simultaneously.

Contributions of Dissertation

Contributions

- Cross-dataset Identity Reasoning
- Semantics-aware Indexes at Global Scale
- Content-based Metrics for Dataset Discovery
- The LODsyndesis suite of Services

Cross-dataset Identity Reasoning

Input & Output

owl:sameAs Relationships
ex:Aristotle ≡ d3:Artistotelis
d2:Aristotle ≡ d3:Artistotelis
ex:Socrates ≡ d3:Socrates
ex:Socrates ≡ d2:Socrates
d1:Immanuel_Kant ≡ d2:Kant
ex:Athens ≡ d4:Athens
d2:Kant ≡ d3:Kant
d2:Karl_Max ≡ ex:Marx

owl:equivalentProperty Relationships
d1:birthPlace ≡ d3:birthPlace
d3:birthPlace ≡ d4:wasBornIn
d2:birthPlace ≡ d3:birthPlace
d1:birthYear ≡ 3:birthYear
d1:birthYear ≡ d2:yearOfBirth
d1:influences ≡ d2:influences

owl:equivalentClass *Relationships*
d4:GR_Philosopher ≡ d2:Gre_Philosopher

Input

Entity	EID
ex:Aristotle	E1
d2:Aristotle	E1
d3:Aristotelis	E1
ex:Stagira	E2
ex:Imannuel_Kant	E3
d2:Kant	E3
ex:Athens	E4
d4:Athens	E4
ex:Socrates	E5
d2:Socrates	E5
d3:Socrates	E5
d4:Greece	E6
d2:Karl_Marx	E7
ex:Marx	E7

Entity Equiv. Catalog

Property	PID
d1:birthPlace	P1
d2:birthPlace	P1
d3:birthPlace	P1
d4:wasBornIn	P1
d1:birthYear	P2
d2:yearOfBirth	P2
d3:birthYear	P2
d4:yearOfBirth	P2
d1:influences	P3
d2:influences	P3
d4:capital	P4
rdf:type	P5
ex:lived	P6

Property Equiv. Catalog

Class	CID
d3:German_ Philosopher	C1
d4:GR_Philosopher	C2
d2:Gre_Philosopher	C2

Class Equiv. Catalog

Output

Challenges & Requirements

Challenges

- Computation of cross-dataset identity reasoning
  - presupposes **knowledge** of all **datasets**
  - requires a lot of **RAM memory**

Related Research Questions

How to compute in an **efficient way** the **transitive** and **symmetric closure** of **equivalence relationships**?

The Objective

- Create Catalogs where **all the URIs** that refer to the **same entity** (same class of equivalence) are getting the **same signature**
- Read each **owl:sameAs pair** only **once** (in an incremental way)

Signature Based Algorithm - Construction Rules

- We introduce an **incremental signature-based** algorithm which
  - requires **a single pass** for computing the closure, where each pair, e.g., u_1 sameAs u_2 is read only **once**
  - relies on **five rules**
  - assigns to each class of equivalence an **ID** (that we call **signature**)

- ***Rule 1***. If both URIs have not a signature, a new signature is assigned to 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	$\mathbf{u_5, u_6}$

Signature Based Algorithm - Construction Rules (cont.)

- ***Rules 2-3***. If u_1 has a 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	$\boxed{u_3}, u_4$
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, continue

- ***Rule 5***. If both URIs have a different signature, the URIs of these two signatures are concatenated

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~~	~~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

Equiv. Catalog

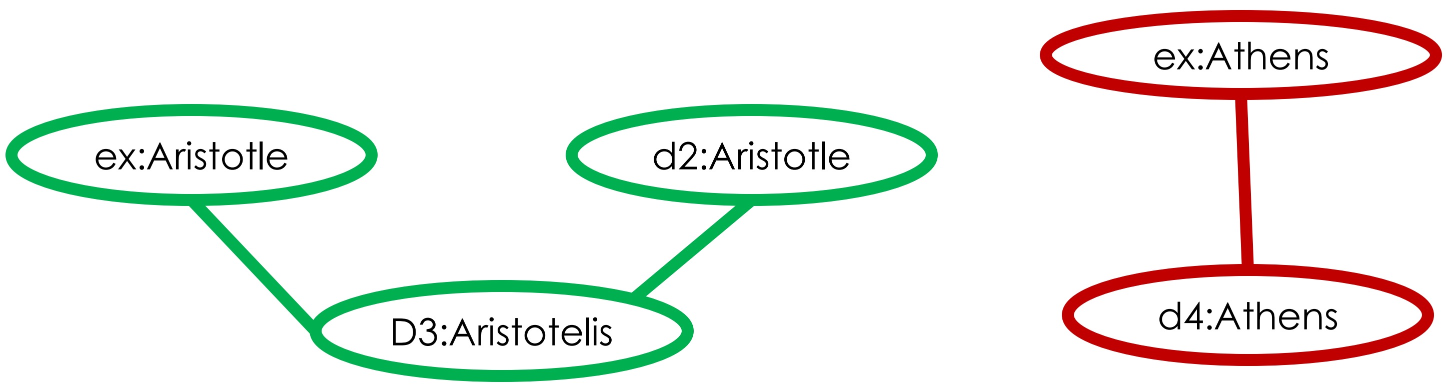
Signature Based Algorithm - Efficiency

Efficiency

- (+) Reads each equivalence pair **only once**
- (+) Keeps in memory **only** the catalog and the classes of equivalence

Alternative Approach

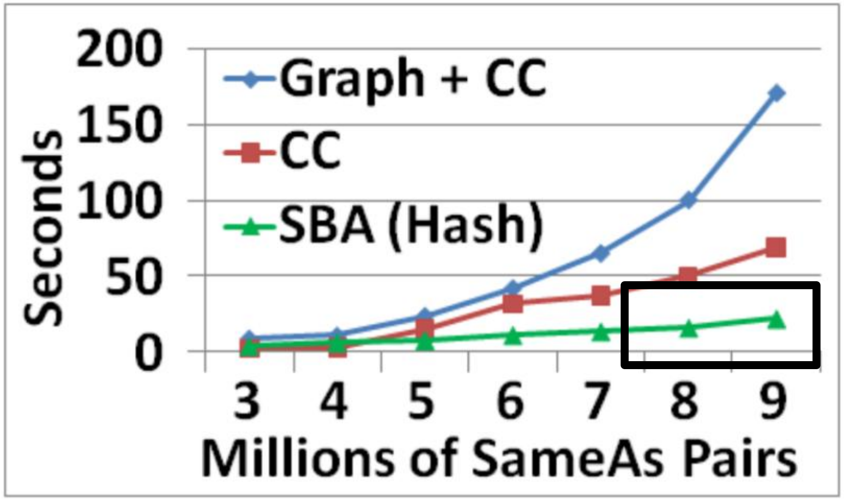
- Turn the equivalence Relationships to an undirected graph
- Find the connected components (CC) by using Tarjan's Algorithm.



Key Results – Closure in a Single Machine

We used a single computer with **8GB memory** and an **i7 core.**

- The **Signature-Based Algorithm** is always **faster** than a connected components algorithm
- We computed the closure of more than **13 million pairs** in **45 seconds!**



- The problem of these algorithms: unable to compute the closure for over 13 million relationships **due to main memory issues**

Parallel Algorithm for Computing the Closure

Challenge

- How to **break this task** to several Machines (e.g., MapReduce) with a **logarithmic** number of iterations and a **logarithmic** communication cost

Solution

- Use **Hash-to-min algorithm** [13] (proposed by Rastogi et al.)
  - Convert the equivalence relationships into an **undirected graph**
  - Compute the **Connected Components** in **parallel**
  - **Iterations number:** O(logV) V: number of nodes in the largest CC
  - **Communication cost** between iterations: O(logn|V|+|E|)

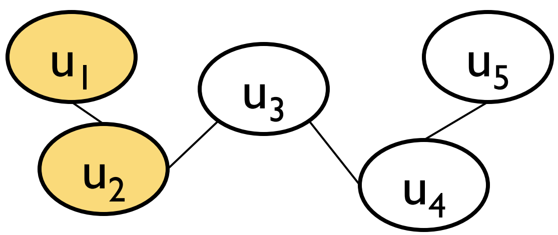
- We propose two **Heuristics,** applicable for our **domain**
  - for **decreasing** the number of iterations and communication cost

Hash-to-Min Algorithm

- Initial Job: For each **URI u** (or node) we find its **neighbors**
- Mapper: Find the u_{min} of the neighbours of each node wrt to a global ranking
  - Send **C_u to u_{min}** and inform other nodes about u_{min}
- Reducer: C_u is the union of all incoming clusters

$u_1 < u_2 < u_3 <, u_4 < u_5$

Iteration 1 Initial Job

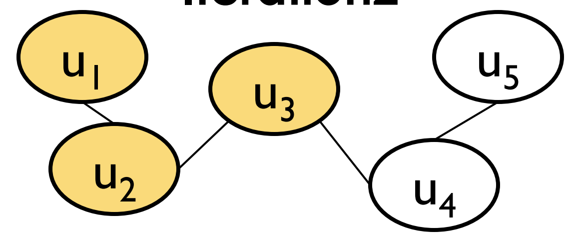


URI	C_u
U_1	**U_1**, U_2
U_2	**U_1**, U_2, U_3
U_3	**U_2**, U_3, U_4
U_4	**U_3**, U_4, U_5
U_5	**U_4**, U_5

Sent cluster to u_1

Informed u3 that $u_{min}=u_1$

Iteration2

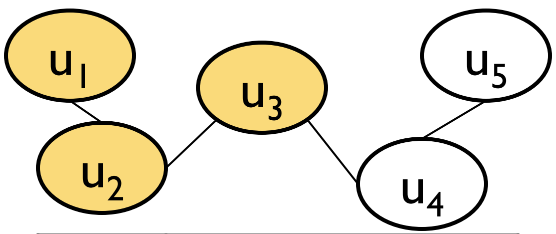


URI	C_u
U_1	**U_1, U_2, U_3**
U_2	**U_1**, U_2, U_3, U_4
U_3	**U_1**, U_3, U_4, U_5
U_4	**U_2**, U_4, U_5
U_5	**U_3**

Hash-to-Min Algorithm (cont.)

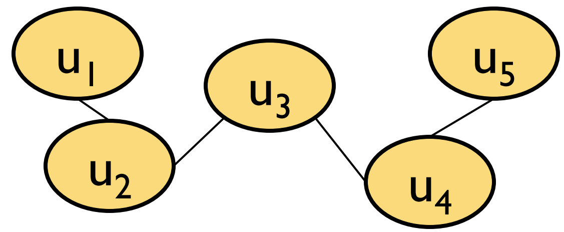
- The connected component (CC) has been **computed** when
  - The **cluster** of u_{min} contains the **entire connected component**
  - **All other nodes** in the connected component contain only u_{min}

Iteration 2



URI	C_u
U_1	**U_1, U_2, U_3**
U_2	**U_1**, U_2, U_3, U_4
U_3	**U_1**, U_3, U_4, U_5
U_4	**U_2**, U_4, U_5
U_5	**U_3**

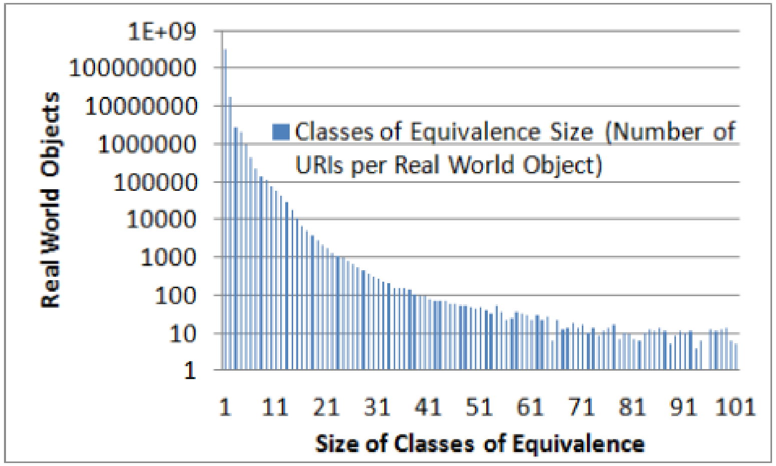
Iteration 3



URI	C_u
U_1	U_1, U_2, U_3, U_4, U_5
U_2	min=u_1
U_3	min=u_1
U_4	min=u_1
U_5	min=u_1

Hash-to-Min - Decrease Iterations Number

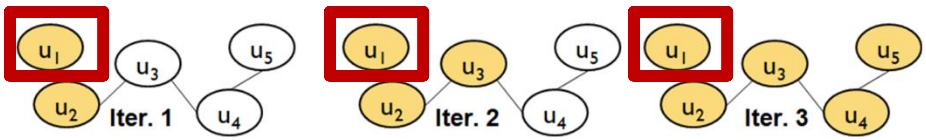
- **Power-Law Distribution:** In the datasets that we use, there exists
  - a small number of large connected components (many iterations)
  - a **large number** of **small** connected components (few iterations)



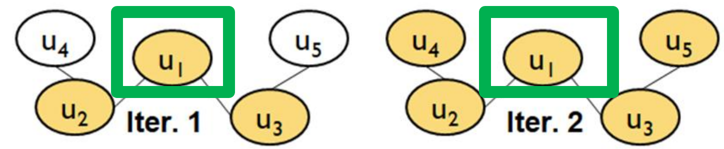
- **Target:** Avoid to perform **more iterations** for a **small number** of **large** Connected Components
- **Solution:** After an iteration, if the number of **remaining URIs** is **lower** than a **threshold t**
  - **Step 1.** Send the **remaining URIs** to one machine
  - **Step 2.** Use the **signature-based algorithm**

Hash-to-Min - Decrease Iterations Number(cont.)

Predefined Global Ranking: It can produce less or more MapReduce Jobs



umin is on the **edge of CC→3 Jobs**



umin is on the **centre of CC→2 Jobs**

How to **"Foresee"** the **centre** of the **CC**?

- Problem: **Expensive** to find the URI occurring as the center of a CC
- Solution: More **probable** a URI from a **popular dataset** to be **centre of a CC**
  - ✓ **Step 1.** Count the frequency of each prefix in the equivalence relationships
  - ✓ **Step 2.** Select as umin the URI of the most popular dataset

owl: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

SameAs Prefix Index

SameAs Prefix	Frequency
http://dbpedia.org/ (dbp)	5
http://yago-knowledge.org/ (yg)	2
http://data.nytimes.com/ (nyt)	2
http://en.wikipedia.org/ (en_wiki)	2
http://geonames.org/ (geo)	1

Result of Closure in Running Example

owl:sameAs Relationships
ex:Aristotle ≡ d3:Artistotelis
d2:Aristotle ≡ d3:Artistotelis
ex:Socrates ≡ d3:Socrates
ex:Socrates ≡ d2:Socrates
d1:Immanuel_Kant ≡ d2:Kant
ex:Athens ≡ d4:Athens
d2:Kant ≡ d3:Kant
d2:Karl_Max ≡ ex:Marx

Entity	EID
ex:Aristotle	E1
d2:Aristotle	E1
d3:Aristotelis	E1
ex:Stagira	E2
ex:Imannuel_Kant	E3
d2:Kant	E3
ex:Athens	E4
d4:Athens	E4
ex:Socrates	E5
d2:Socrates	E5
d3:Socrates	E5
d4:Greece	E6
d2:Karl_Marx	E7
ex:Marx	E7

Entity Equiv. Catalog

owl:equivalentProperty Relationships
d1:birthPlace ≡ d3:birthPlace
d3:birthPlace ≡ d4:wasBornIn
d2:birthPlace ≡ d3:birthPlace
d1:birthYear ≡ 3:birthYear
d1:birthYear ≡ d2:yearOfBirth
d1:influences ≡ d2:influences

Property	PID
d1:birthPlace	P1
d2:birthPlace	P1
d3:birthPlace	P1
d4:wasBornIn	P1
d1:birthYear	P2
d2:yearOfBirth	P2
d3:birthYear	P2
d4:yearOfBirth	P2
d1:influences	P3
d2:influences	P3
d4:capital	P4
rdf:type	P5
ex:lived	P6

Property Equiv. Catalog

owl:equivalentClass Relationships
d4:GR_Philosopher ≡ d2:Gre_Philosopher

Class	CID
d3:German_ Philosopher	C1
d4:GR_Philosopher	C2
d2:Gre_Philosopher	C2

Class Equiv. Catalog

Contributions - Next Task

- Cross-dataset Identity Reasoning
- Semantics-aware Indexes at Global Scale
- Content-based Metrics for Dataset Discovery
- The LODsyndesis suite of Services

Input

Datasets

Triples of Dataset D1		
ex:Aristotle	d1:birthPlace	**ex:Stagira**
ex:Aristotle	d1:birthYear	"384 bc"^^xsd:Year
ex:Aristotle	d1:influences	**ex:Immanuel_Kant**
ex:Aristotle	d1:influences	**ex:Marx**
ex:Socrates	d1:birthPlace	**ex:Athens**
ex:Socrates	d1:birthYear	"470 BC"^^xsd:Year

Triples of Dataset D2		
d2:Aristotle	d2:influences	**d2:Karl_Marx**
d2:Aristotle	d2:influences	**d2:Kant**
d2:Aristotle	d2:birthPlace	**ex:Stagira**
d2:Socrates	d2:yearOfBirth	"470 BC"
d2:Socrates	rdf:type	*d2:Gre_Philosopher*
d2:Aristotle	rdf:type	*d2:Gre_Philosopher*

Entities **Properties** **Literals** **Classes**

Triples of Dataset D3		
d3:Socrates	d3:birthYear	"471 BC"^^xsd:Date
d3:Socrates	d3:birthPlace	**ex:Athens**
d3:Aristotelis	d3:birthYear	"384 BC"^^xsd:Date
ex:Athens	ex:lived	**d3:Aristotelis**
ex:Marx	rdf:type	*d3:German_Philosopher*
d3:Aristotelis	d3:birthPlace	**ex:Stagira**

Triples of Dataset D4		
d4:Athens	ex:lived	**ex:Aristotle**
ex:Aristotle	d4:wasBornIn	**ex:Stagira**
ex:Socrates	d4:wasBornIn	**d4:Athens**
d4:Greece	d4:capital	**d4:Athens**
ex:Aristotle	rdf:type	*d4:GR_Philosopher*
ex:Socrates	rdf:type	*d4:GR_Philosopher*

Equivalence Catalogs

Entity	**EID**
ex:Aristotle	E1
d2:Aristotle	E1
d3:Aristotelis	E1
ex:Stagira	E2
ex:Imannuel_Kant	E3
d2:Kant	E3
ex:Athens	E4
d4:Athens	E4
ex:Socrates	E5
d2:Socrates	E5
d3:Socrates	E5
d4:Greece	E6
d2:Karl_Marx	E7
ex:Marx	E7

Entity Equiv. Catalog

Output

- A set of Semantically Enriched (Inverted) Indexes

Entity (EID)	Property (PID)	EID or Literal or CID	Datasets
E1 (Aristotle)	P1 (birthPlace)	E2 (Stagira)	D1,D2,D3,D4
	P2 (birthYear)	"384 bc"	D1,D3
	P3 (influences)	E3 (Kant)	D1,D2
		E7 (Marx)	D1,D2
	P6*(lived)	E6 (Athens)	D3,D4
	P5 (type)	C2 (GRE Philosopher)	D2,D4
E2 (Stagira)	P1* (birthPlace)	E1 (Aristotle)	D1,D2,D3,D4
E3 (Kant)	P3* (influences)	E1 (Aristotle)	D1,D2
E4 (Greece)	P4 (capital)	E6 (Athens)	D4
E5 (Socrates)	P1 (birthPlace)	E6 (Athens)	D1,D3,D4
	P2 (birthYear)	"470 bc"	D1,D2
		"471 bc"	D3
	P5 (type)	C2 (GRE Philosopher)	D2,D4
E6 (Athens)	P6(lived)	E1 (Aristotle)	D3,D4
	P1* (birthPlace)	E5 (Socrates)	D1,D3,D4
	P4* (capital)	E4 (Greece)	D4
E7 (Marx)	P5 (type)	C1 (GER Philosopher)	D3
	P3* (influences)	E1 (Aristotle)	D1,D2

Entity-Triples Index

RWE	Datasets
E1 (Aristotle)	D1,D2,D3,D4
E2 (Stagira)	D1,D2,D3,D4
E3 (Kant)	D1,D2
E4 (Greece)	D4
E5 (Socrates)	D1,D2,D3,D4
E6 (Athens)	D1,D3,D4
E7 (Marx)	D1,D2

Entity Index

RWC	Datasets
C1 (Greek Philosopher)	D2,D4
C2 (German Philosopher)	D3

Class Index

RWP	Datasets
P1 (birthPlace)	D1,D2,D3,D4
P2 (birthYear)	D1,D2,D3
P3 (influences)	D1,D2
P4 (capital)	D4
P5 (rdf:type)	D2,D3,D4
P6 (lived)	D3,D4

Property Index

Literal	Datasets
384 bc	D1,D3
470 bc	D1,D2
471 bc	D3

Literals Index

Challenges & Requirements

Challenges

- There are **many datasets** and some of them are **very big**
- The **result of the closure** should be taken into account for constructing the indexes.

Related Research Questions

How to **apply the result** of the cross-dataset identity reasoning for constructing such **semantics-aware indexes?**

How to **parallelize efficiently** the construction of indexes?

The Objective

- Apply the **result** of the **closure**
- Create **Entity-Based** Semantics-aware Indexes
- **Parallelize** the **construction** of indexes by reading each triple once
- Store the **Provenance**

Apply the Result of the Closure

- ❑Each machine reads **a subset of triples** and **a subset of entity equivalence catalog**
- ❑We keep **in memory** property and class equivalence catalogs (they are small in size)
- ❑We replace **each URI** with its **identifier**, and we perform **simple literals conversion**
- ❑We need **two MapReduce jobs** for converting all the triples

Step 1. Input

Datasets

Triples of Dataset D1		
ex:Aristotle	d1:birthPlace	**ex:Stagira**
ex:Aristotle	d1:birthYear	"384 bc"^^xsd:Year
ex:Aristotle	d1:influences	**ex:Immanuel_Kant**
ex:Aristotle	d1:influences	**ex:Marx**
ex:Socrates	d1:birthPlace	**ex:Athens**
ex:Socrates	d1:birthYear	"470 BC"^^xsd:Year

Triples of Dataset D2		
d2:Aristotle	d2:influences	**d2:Karl_Marx**
d2:Aristotle	d2:influences	**d2:Kant**
d2:Aristotle	d2:birthPlace	**ex:Stagira**
d2:Socrates	d2:yearOfBirth	"470 BC"
d2:Socrates	rdf:type	*d2:Gre_Philosopher*
d2:Aristotle	rdf:type	*d2:Gre_Philosopher*

Triples of Dataset D3		
d3:Socrates	d3:birthYear	"471 BC"^^xsd:Date
d3:Socrates	d3:birthPlace	**ex:Athens**
d3:Aristotelis	d3:birthYear	"384 BC"^^xsd:Date
ex:Athens	ex:lived	**d3:Aristotelis**
ex:Marx	rdf:type	*d3:German_Philosopher*
d3:Aristotelis	d3:birthPlace	**ex:Stagira**

Triples of Dataset D4		
d4:Athens	ex:lived	**ex:Aristotle**
ex:Aristotle	d4:wasBornIn	**ex:Stagira**
ex:Socrates	d4:wasBornIn	**d4:Athens**
d4:Greece	d4:capital	**d4:Athens**
ex:Aristotle	rdf:type	*d4:GR_Philosopher*
ex:Socrates	rdf:type	*d4:GR_Philosopher*

Entity	EID
ex:Aristotle	E1
d2:Aristotle	E1
d3:Aristotelis	E1
ex:Stagira	E2
ex:Imanuel_Kant	E3
d2:Kant	E3
ex:Athens	E4
d4:Athens	E4
ex:Socrates	E5
d2:Socrates	E5
d3:Socrates	E5
d4:Greece	E6
d2:Karl_Marx	E7
ex:Marx	E7

Entity Equiv. Catalog

Property	PID
d1:birthPlace	P1
d2:birthPlace	P1
d3:birthPlace	P1
d4:wasBornIn	P1
d1:birthYear	P2
d2:yearOfBirth	P2
d3:birthYear	P2
d4:yearOfBirth	P2
d1:influences	P3
d2:influences	P3
d4:capital	P4
rdf:type	P5
ex:lived	P6

Property Equiv. Catalog

→ RAM

Class	CID
d3:German_Philosopher	C1
d4:GR_Philosopher	C2
d2:Gre_Philosopher	C2

Class Equiv. Catalog

Output

RWT(D1)		
E1	P1	E2
E1	P2	"384 bc"
E1	P3	E3
E1	P3	E7
E5	P1	E6
E5	P2	"470 bc"

RWT(D2)		
E1	P3	E7
E1	P3	E3
E1	P1	E2
E5	P2	"470 bc"
E5	P5	C2
E1	P5	C2

RWT(D3)		
E5	P2	"471 bc"
E5	P1	E6
E1	P2	"384 bc"
E1	P6	E6
E7	P5	C1
E1	P1	E2

RWT(D4)		
E1	P6	E6
E1	P1	E2
E5	P1	E6
E4	P4	E6
E1	P5	C2
E5	P5	C2

Creation of Entity-Based Triples Index

The Objective

- ➢ Collect **all the available data** for a given entity
- ➢ Use a **single** MapReduce Job (read each triple once)

- ❑ For **not missing facts** for an entity, for the triples having **entities as objects**, we create two key-value pairs
- ❑ If the object is a **literal** or a **class**, we create one key-value pair

RWT(D1)	***RWT(D2)***	***RWT(D3)***	***RWT(D4)***
E1 P1 E2	E1 P3 E7	E5 P2 “471 bc”	E1 P6 E6
E1 P2 “384 bc”	E1 P3 E3	E5 P1 E6	E1 P1 E2
E1 P3 E3	E1 P1 E2	E1 P2 “384 bc”	E5 P1 E6
E1 P3 E7	E5 P2 “470 bc”	E1 P6 E6	E4 P4 E6
E5 P1 E6	E5 P5 C2	E7 P5 C1	E1 P5 C2
E5 P2 “470 bc”	E1 P5 C2	E1 P1 E2	E5 P5 C2

Input

Two key Value Pairs

Key	**Value**
E1	P1,E2, **D1**
E2	E1,P1*,**D1**

One key Value Pair

Key	**Value**
E5	P2, “471 BC”,**D3**

Creation of Entity-Based Triples Index (cont.)

- **Reducer**: collects **all the triples** for an entity.
- **Communication Cost**: O(|Triples|).

Entity (EID)	Property (PID)	EID or Literal or CID	Datasets
E1 (Aristotle)	P1 (birthPlace)	E2 (Stagira)	D1,D2,D3,D4
	P2 (birthYear)	"384 bc"	D1,D3
	P3 (influences)	E3 (Kant)	D1,D2
		E7 (Marx)	D1,D2
	P6*(lived)	E6 (Athens)	D3,D4
	P5 (type)	C2 (GRE Philosopher)	D2,D4
E2 (Stagira)	P1* (birthPlace)	E1 (Aristotle)	D1,D2,D3,D4
E3 (Kant)	P3* (influences)	E1 (Aristotle)	D1,D2
E4 (Greece)	P4 (capital)	E6 (Athens)	D4
E5 (Socrates)	P1 (birthPlace)	E6 (Athens)	D1,D3,D4
	P2 (birthYear)	"470 bc"	D1,D2
		"471 bc"	D3
	P5 (type)	C2 (GRE Philosopher)	D2,D4

Entity Triples Index

Some triples are stored twice.

We place together all the values for a specific entity-predicate pair for enabling their **comparison**!

Creation of Semantically Enriched Indexes

- Creation of other Semantics-aware Indexes for storing the provenance
  - ***Entity Index*** stores all the datasets where a **real world entity** occurs
  - ***Class Index*** stores the provenance of each **real world class**
  - ***Literals Index*** stores the provenance of each **literal**
  - ***Property Index*** stores the datasets of a **real world property**
- Construction: Read the **desired part** of the **triples** and use a **classical** inverted index **parallel algorithm** (require a single job)

RWE	Datasets
E1 (Aristotle)	D1,D2,D3,D4
E2 (Stagira)	D1,D2,D3,D4
E3 (Kant)	D1,D2
E4 (Greece)	D4
E5 (Socrates)	D1,D2,D3,D4
E6 (Athens)	D1,D3,D4
E7 (Marx)	D1,D2

Entity Index

RWC	Datasets
C1 (Greek Philosopher)	D2,D4
C2 (German Philosopher)	D3

Class Index

Literal	Datasets
384 bc	D1,D3
470 bc	D1,D2
471 bc	D3

Literals Index

RWT(D1)		
E1	P1	E2
E1	P2	"384 bc"
E1	P3	E3
E1	P3	E7
E5	P1	E6
E5	P2	"470 bc"

RWT(D2)		
E1	P3	E7
E1	P3	E3
E1	P1	E2
E5	P2	"470 bc"
E5	P5	C2
E1	P5	C2

RWT(D3)		
E5	P2	"471 bc"
E5	P1	E6
E1	P2	"384 bc"
E1	P6	E6
E7	P5	C1
E1	P1	E2

RWT(D4)		
E1	P6	E6
E1	P1	E2
E5	P1	E6
E4	P4	E6
E1	P5	C2
E5	P5	C2

Key	Value
P3	D1

Key	Value
P3	D2

RWP	Datasets
P1 (birthPlace)	D1,D2,D3,D4
P2 (birthYear)	D1,D2,D3
P3 (influences)	D1,D2
P4 (capital)	D4
P5 (rdf:type)	D2,D3,D4
P6 (lived)	D3,D4

Property Index

Key Results - Infrastructure & Datasets

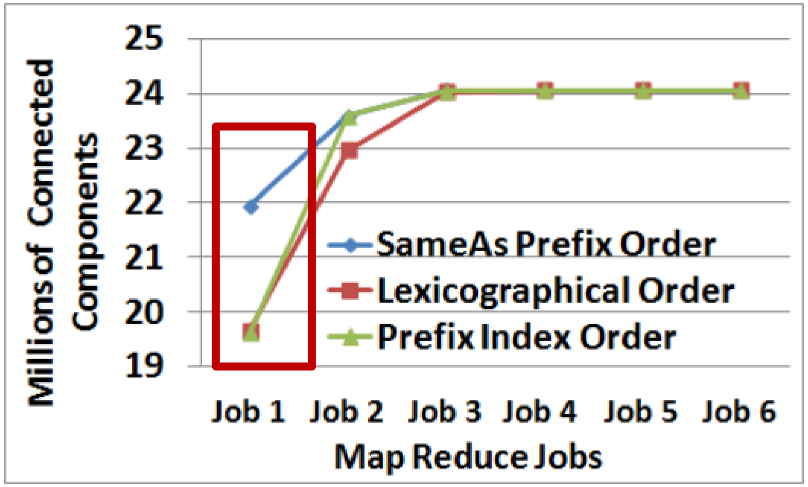
- We used a **cluster** in okeanos cloud computing service with
  - **12 real machines**
    - each one has 8 cores, 8 GB main memory and 60GB disk space.
- We created **96 virtual machines**
  - each one has 1 core and 1GB memory
- We used **Hadoop MapReduce 2.7.3**.
- We collected **400 Datasets** having over **2 billion triples**, **412 million URIs** and **44 million of equivalence relationships** (255 GB in total)

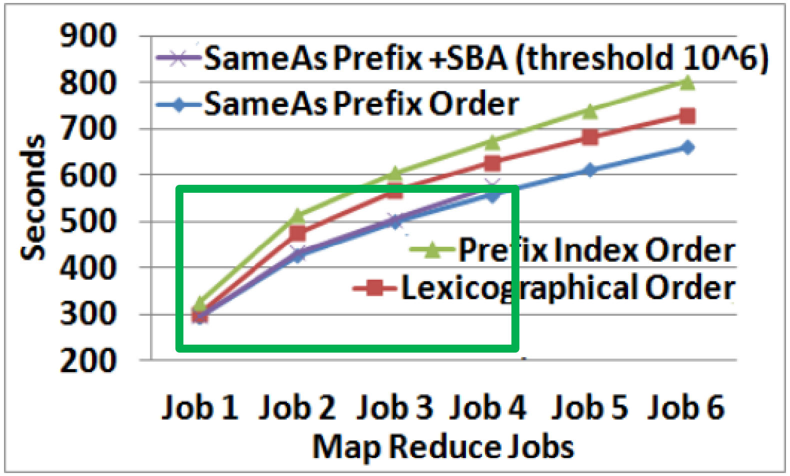
Domain	$\|D\|$	$\|$Triples$\|$	$\|$Entities$\|$	$\|$Literals$\|$
Cross-Domain (CD)	24	971,725,722	199,359,729	216,057,389
Publications (PUB)	94	666,580,552	127,624,700	155,052,015
Geographical (GEO)	15	134,972,105	40,185,923	25,572,791
Media (MED)	8	74,382,633	16,480,681	9,447,048
Life Sciences (LF)	18	74,304,529	10,050,139	10,844,398
Government (GOV)	45	59,659,817	6,657,014	7,467,560
Linguistics (LIN)	85	20,211,506	3,825,012	2,808,717
User Content (UC)	14	16,617,837	7,829,599	901,847
Social Networks (SN)	97	3,317,666	762,323	853,416
All	400	2,021,772,367	412,775,120	429,005,181

Key Results - Parallel Computation of Closure

- **Input**: 44 million owl:sameAs pairs
- **Output**: 24 million Connected Components

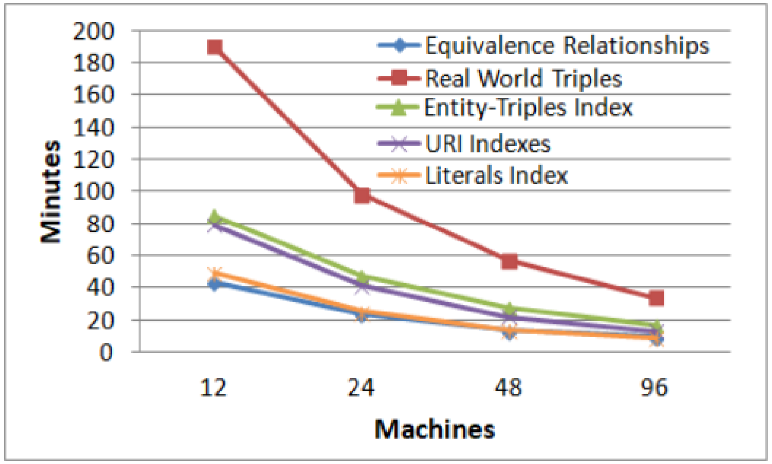
- By **predicting** the **centre** of the connected components (order of SameAsPrefixIndex) we computed in the **1st job** correctly
  - **2.5 million more** connected components comparing to any other order
- Best Variation**:** Using the order of **SameAsPrefixIndex**, and the **Signature-Based** algorithm (for a few large connected components)
  - Computation time of Best Variation: **9 minutes** in **4 jobs**
  - Computation time of other variations: over 11 minutes in 6 jobs





Key Results - Execution Time & Scalability

- All the catalogs and indexes (for 2 billion triples) constructed in **81.5 minutes**!
- All the algorithms & methods are **scalable**!
  - We identified **4.62x-6x speedup** (ideal is 8x) by using 96 VMs instead of 12.



- Indexes' size is **2.7x smaller** than the input datasets
  - Entity-Triples index disk size: 70.3 GB
  - Equivalence Catalogs disk size: 24 GB

Table 4.7: Construction time and size of catalogs and indexes.

Index/Catalog	Execution Time (96 Machines)	Size on Disk	Entries
Equivalence Catalogs	9.35 min	24 GB	413,567,083
Real World Triples	33.5 min	82.4 GB	1,826,224,504
Entity-Triples Index	17 min	70.3 GB	2,498,223,345
Entity Index	13.2 min	6 GB	368,295,245
Properties Index	5 sec	2.5 MB	247,713
Class Index	8 sec	6 MB	544,250
Literals Index	8.5 min	16 GB	379,043,131
All	81.55 min	198.7 GB	5,486,145,271

Contributions - Next Task

- Cross-dataset Identity Reasoning
- Semantics-aware Indexes at Global Scale
- Content-based Metrics for Dataset Discovery
- The LODsyndesis suite of Services

Input & Output

Query

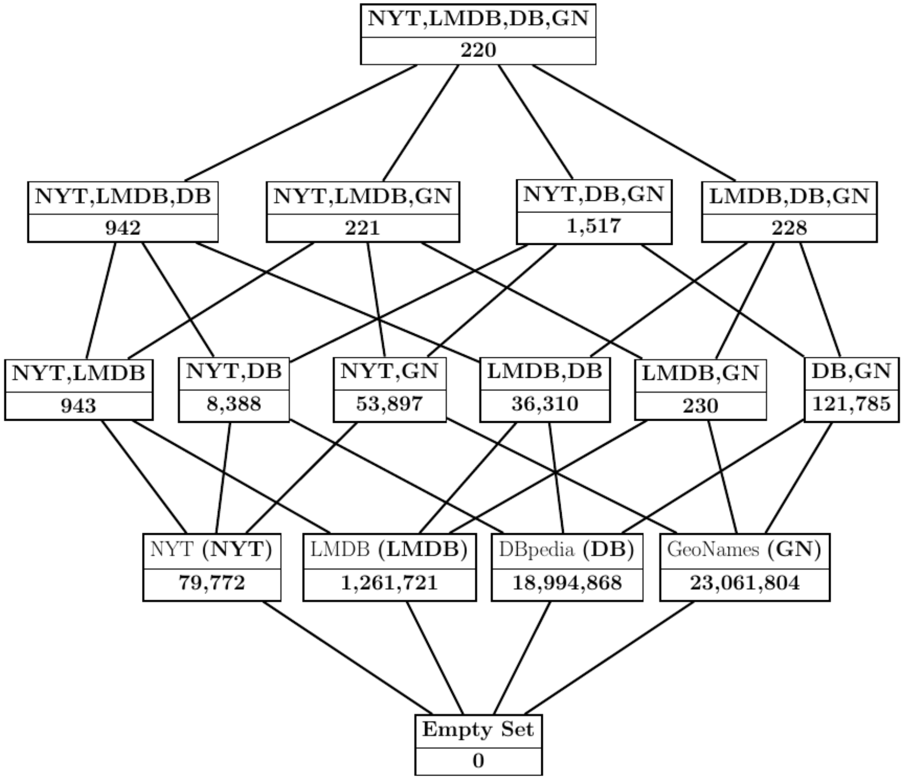
Qconnectivity

Scientist

"I want the 3 datasets having the most common species".

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 (Hyena)	D1,D2,D4
E7 (Cheetah)	D2,D3

Query & Index



Lattice of Measurements

Challenges

Challenges

- The **possible combinations** of datasets is **exponential** in number
- Set operations between **large datasets** are **quite expensive**

Related Research Questions

Whether a standard W3C **query language** (such as SPARQL) can be used for solving such **maximization problems**?

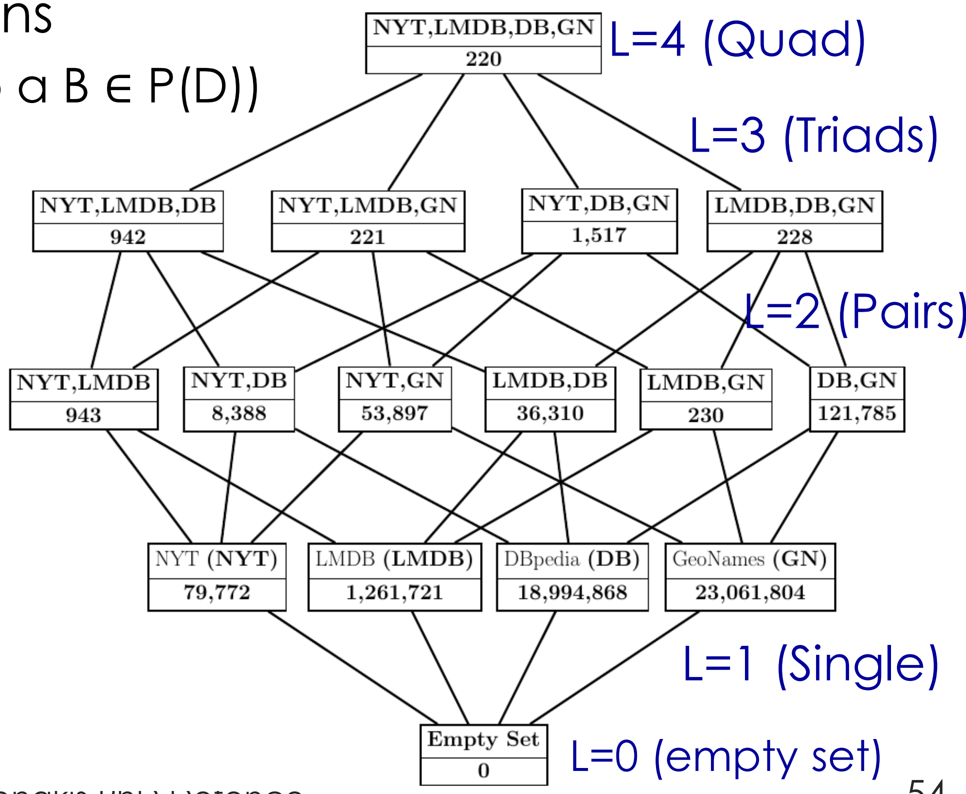
How we can **reduce** the number of **set operations** between different datasets?

Can these content-based measurements be **parallelized**?

The Lattice of Measurements

- **D = {D1, …, Dn}:** a set of datasets
- **P(D):** the power set of D
- A **lattice** is a **partially ordered set** that can be represented as a Directed Acyclic Graph (DAG) where the **edges points** towards **the direct supersets**.

- A **lattice** of |D| datasets contains
  - $2^{|D|}$ nodes (each corresponds to a $B \in P(D)$)
  - |D|+1 levels
    - Range 0<=L<=|D|



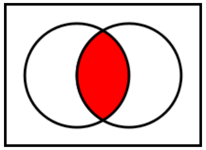
Dataset Discovery Metrics

- ***F* = {RWE,RWP,RWC, LIT,RWT,RWTE'}:** the measurement types
  - Measurements for Entities, Properties, Classes, Literals, Triples!
- **F(Di)** a measurement type applied to a dataset Di
  - RWE(Di) → entities of Di

To tackle the **requirements** we need to be able to solve some maximization problems

Commonalities: Find the **combination of datasets B of size K** having the **most common elements** (entities, literals, triples,...)

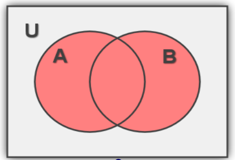
$$cmnBest(K, F) = arg_B \max |cmn(B, F)| \text{ where } cmn(B, F) = \cap_{D_i \in B} F(D_i)$$



Intersection

Coverage: Find the **subset of datasets B of size K whose union** has the **maximum number of elements**

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

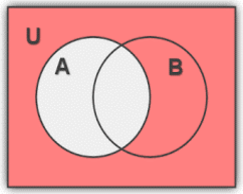


Union

Dataset Discovery Metrics (cont.)

Information Enrichment: Find the **subset of datasets B of size K** having the **most complementary information** (e.g., number of triples) to a dataset Dm

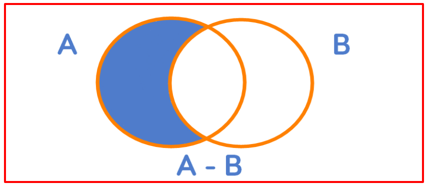
$$\begin{aligned} enrichBest(K, F, D_m) &= arg_B \max |enrich(B, F, D_m)| \text{ where} \\ enrich(B, F, D_m) &= cov(B, F) \setminus F(D_m) \end{aligned}$$



Absolute Complement

Uniqueness: Find the **subset of datasets B of size K** having the **most/less unique content** comparing to a dataset Dm

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



Relative Complement

SPARQL Queries for computing the metrics

- The **syntax** of **SPARQL** enables the computation of such metrics.

- Steps for the query for **common entities**
  - Finds the **URIs** occurring as a **subject or object** for each distinct **dataset**.
  - Performs **joins** among **different datasets** for finding the **common URIs.**
  - **Counts** the **distinct common URIs** of each group of datasets

Query for Computing the Common Entities between any subset of Level L

```
DEFINE input:same-As "yes"
select ?Di ?Dj ... ?Dn count (distinct ?u) as ?commonEntities where {
{graph ?Di {{?u ?p ?o} union {?o ?p ?u . filter(?p!=rdf:type)}}
. filter(isURI(?u))}.
{graph ?Dj {{?u ?p2 ?o2} union {?o2 ?p2 ?u .filter(?p2!=rdf:type)}}} .
...
{graph ?Dn {{?u ?pn ?on} union {?on ?pn ?u .filter(?pn!=rdf:type)}}} .
filter(?Di>?Dj && ... && ?Dn-1>?Dn)}
group by ?Di ?Dj ... ?Dn
```

Limitations of SPARQL Implementations

Computation of Closure

- *Virtuoso*: computes it on query time (time consuming)
- *Blazegraph*: Does not support inference in the quads mode
- Our Approach: closure has been **pre-computed once**

Indexes

- *Virtuoso & Blazegraph*: **fast response** to queries for a **given S,P,O**
- Our Approach: **fast access** to the **provenance** of distinct URIs, triples, etc.

Joins

- *Virtuoso & Blazegraph*: require a large number of joins (URIs, Literals)
- Our Approach: uses **distinct posting lists** of an index as input (very small comparing to the size of URIs, literals)

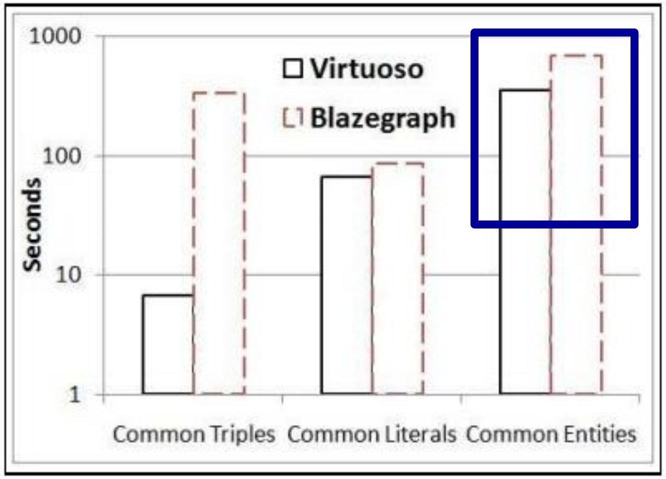
Set theory Properties

- *Virtuoso & Blazegraph*: do not reuse measurements among different subsets of datasets
- Our Approach: **reuses measurements** incrementally

Evaluation - SPARQL Implementations

Datasets and Experiments

- We used 10 datasets and 2 **million** triples
- We **ignored** the computation of **closure**
- Measurements for **45 pairs** of datasets



Key Results

- **Virtuoso** (v. 06.01.3127) was always faster than **Blazegraph** (v. 2.1.4)
- Both tools need over 5 minutes for computing the common entities
    - On average 7 seconds per pair of datasets
- By adding **more data** and computing the **closure** (Virtuoso)
    - the execution time increases (1 minute per pair of datasets)

Our Target

- Enable the computation of metrics for millions of subsets of datasets in a few seconds

How to use the Posting Lists - Direct Counts

- ***occur(D,F):*** all the subsets occurring as a posting list in an inverted index
- ***directCount(B,F):*** frequency of a posting list (i.e., subset of datasets B) in an inverted index
- Let's use the directCount List for computing the metrics!

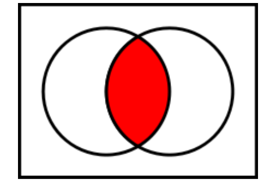
Entity ID	Datasets
E1 (Tiger)	D1,D2,D3,D4
E2 (Panda)	D1,D2,D3,D4
E3 (Snow Leopard)	D1,D2,D4
E4 (Asian Bear)	D1,D3,D4
E5 (Crocodile)	D2,D3
E6 (Hyena)	D1,D2,D4
E7 (Cheetah)	D2,D3

Entity Index containing 7 species

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

DirectCount List for Entity Index

Commonalities (Intersection)



- **Up(B,F):** the supersets of B that can be found in directCount List.
- The **sum** of the **directCount** of **Up(B,F)** gives the **cardinality** of **intersection** of B.

$$|cmn(B,F)| = \sum_{B' \in Up(B,F)} directCount(B',F)$$

- *Baseline Model (BM):* For each subset of datasets B, scan the directCount list once for finding the set Up(B,F).

Query

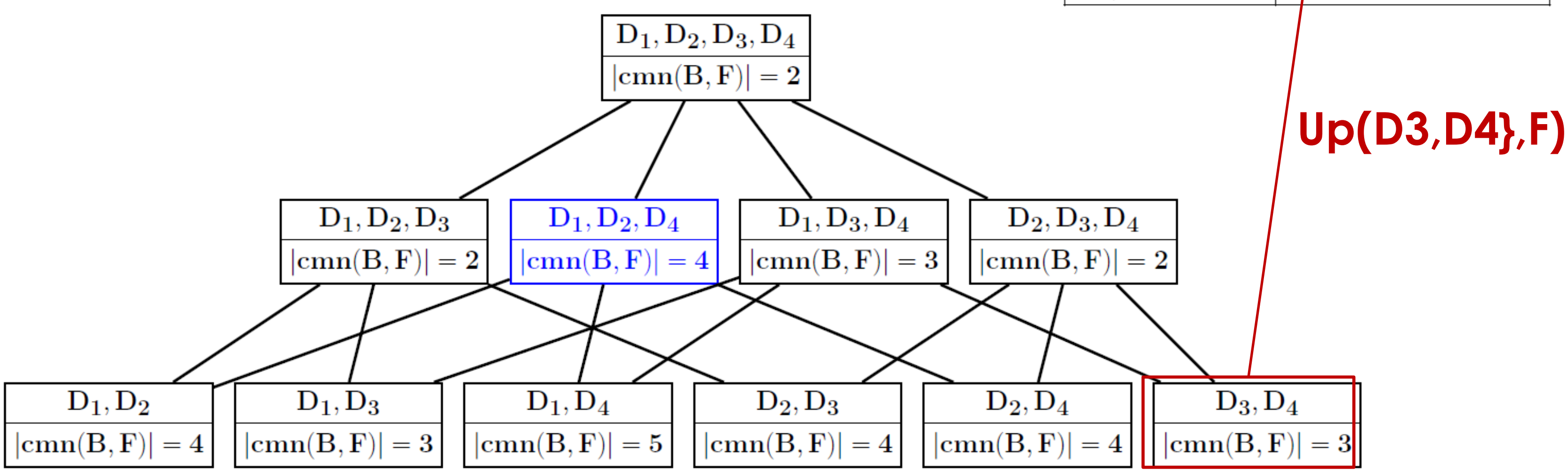
Qconnectivity

Scientist

"I want the 3 datasets having the most common species".

Scan the Corresponding Index

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



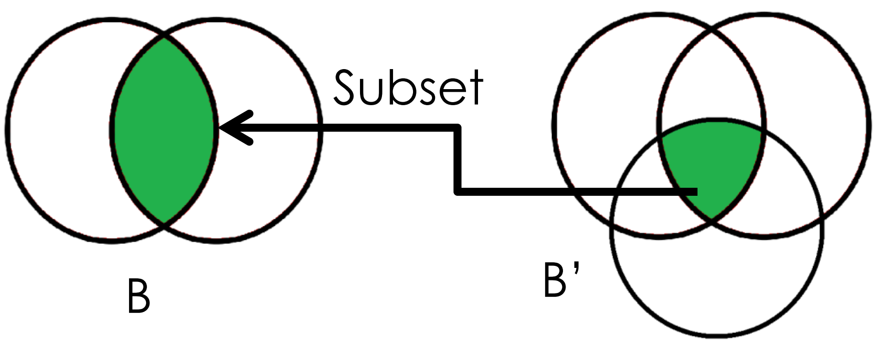
The Challenge

- *Baseline Model (BM):* It is very time-consuming to traverse **all** the **posting lists** for each possible **subset B**

- **Target**: Reduce the number of input posting lists for finding **cmnBest(K,F)**

$$cmnBest(K, F) = arg_B \max |cmn(B, F)| \text{ where } cmn(B, F) = \cap_{D_i \in B} F(D_i)$$

- **Solution**: We propose **two incremental algorithms**, that **reuse the measurements** between two subsets of datasets **B** and **B':**
  - We know that **if B' ⊃ B** then **Up(B',F) ⊆ Up(B,F)**

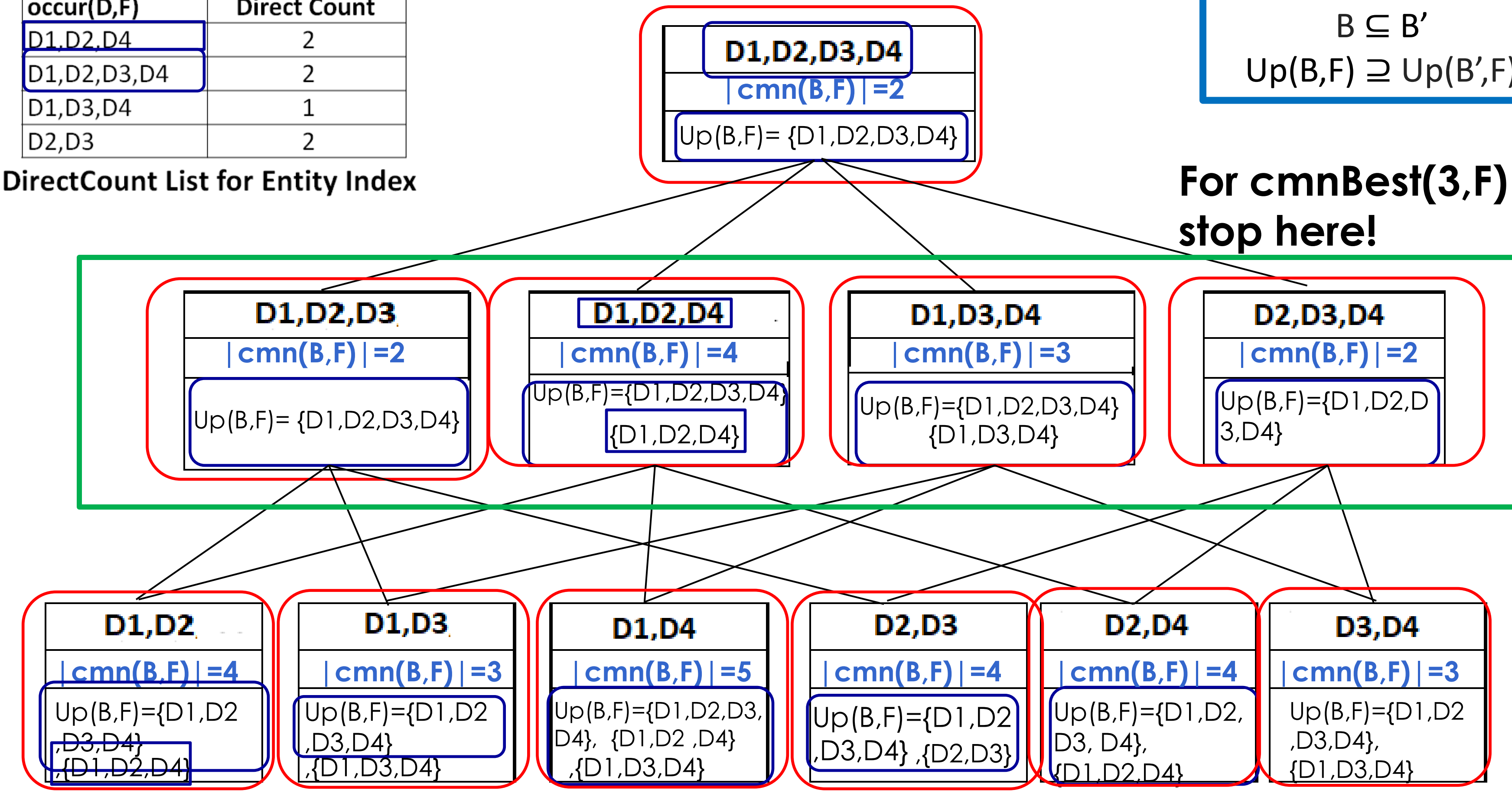


Top-Down Algorithm (BFS Traversal)

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

DirectCount List for Entity Index

B ⊆ B'
Up(B,F) ⊇ Up(B',F)



1. For each node B check if it exists in the directCount List and if it holds, add B to Up(B,F)
2. Sum the values of directCount of Up(B,F)
3. Transfer Up(B,F) to all subsets of B of the previous level since Up(B) ⊇ Up(B') (B ⊆ B')

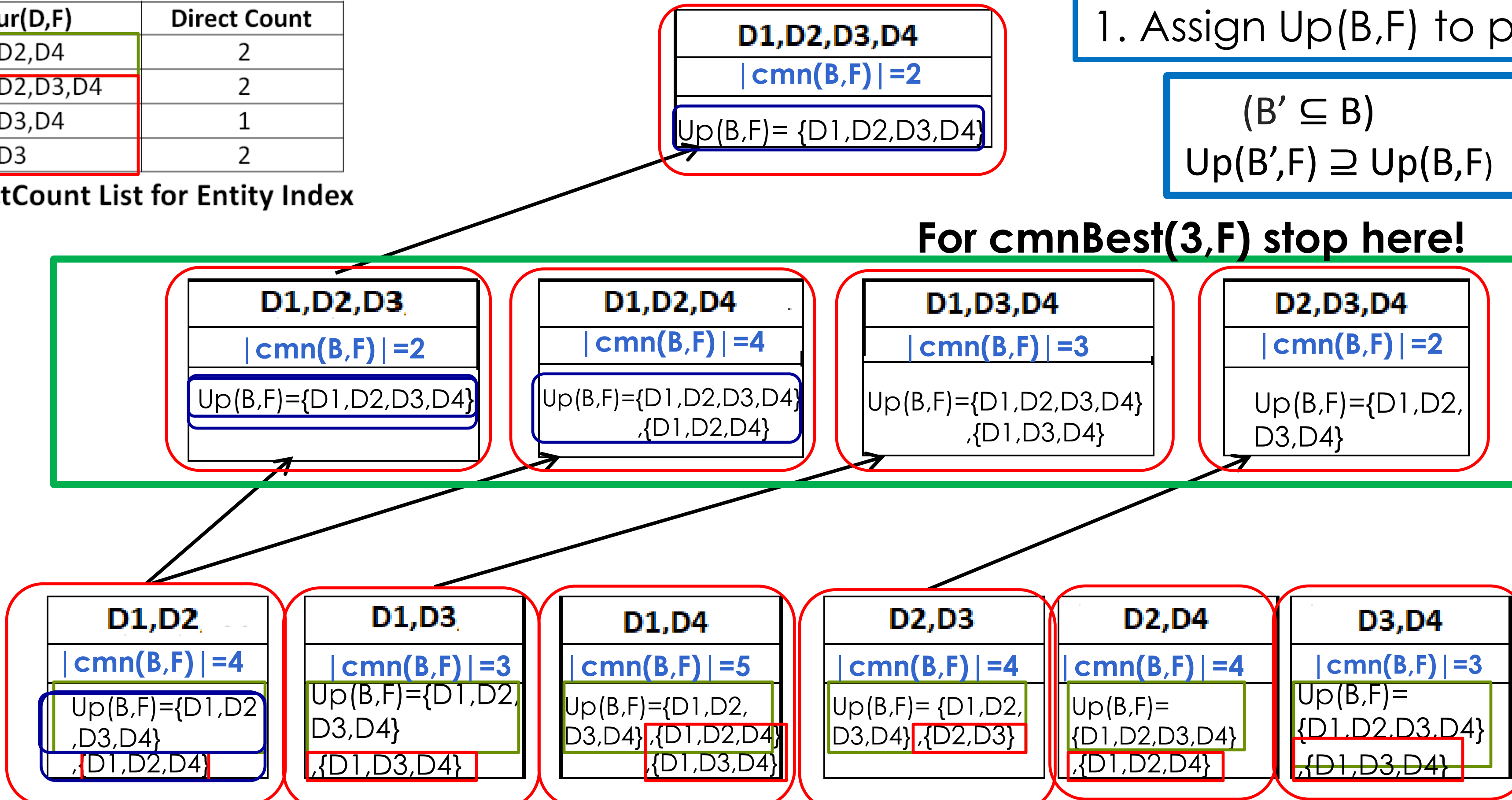
Bottom-Up Algorithm (DFS Traversal)

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

DirectCount List for Entity Index

1. Assign Up(B,F) to pairs

(B' ⊆ B)
Up(B',F) ⊇ Up(B,F)



1. Sum the directCount of Up(B,F)
2. Visit a superset B' of the next level if it has not visited yet .
3. Check which Up(B,F) goes to Up(B',F) since Up(B',F) ⊇ Up(B,F) (B' ⊆ B)

Baseline vs Top-Down vs Bottom-Up

- Both **incremental** methods read **less posting lists** than a **Baseline Model**
- **Top-Down** creates all the edges and has factorial space complexity
- **Bottom-up** creates **1 edge** per node and has **linear** space complexity
  - An extra check is required comparing to top-down approach
  - Can we **further improve** the bottom-up approach?

	Baseline	**Top-Down**	**Bottom-up**
Nodes	$\|V\|$ (worst: $2^{(\|D\|}$)	$\|V\|$ (worst: $2^{(\|D\|}$)	$\|V\|$ (worst: $2^{(\|D\|}$)
Edges	-	$\|E\|$ (worst: $\|D\| * 2^{(\|D\|-1)}$)	$\|V\|$ (worst: $2^{(\|D\|}$)
Time complexity	O(V* **|occur(D,F)|**) **(expensive)**	O(|V|+**|E|**) **(expensive)**	O(|V|*|Up(B,F)|avg)
Space Complexity	O(|occur(D,F)|)	$O(V_K)$ $V_k = \binom{\|D\|}{k} = \frac{\|D\|!}{k!(\|D\|-k)!}$	O(Vd) d:diameter of graph (d=|D|+1)
Biggest Disadvantage	Reads the whole directCount List for each node	•|D|/ 2 times more edges •Factorial space complexity	Check which entries of Up(B,F) can be transferred to B'

Removing Redundant Dataset IDs & Regrouping

Target: Further **decrease** the **number of posting lists** that we read

- Bottom-up DFS traversal follows a **strict numerical order.**
  - Each time we add a dataset **Dk** to a subset **B**, where
    - k is **larger** than the **ID** of all datasets **in B**.
- From <D1,D4>
  - We will visit <D1,D4,**D5**> → The ID of D5 is **larger** than the others **(5>4>1)**
  - We will not visit <D1,D2,D4> → The ID of D2 is smaller than D4 (2<4)

Solution: Remove the **redundant datasets** from the **posting lists** and **regroup** the "pruned" entries

Compute the commonalities for all the supersets of D1,D4 that have not been explored yet!

Up({D1,D4},F)	Direct Count
D1,**D2**,**D3**,D4,D5	5
D1,**D2**,D4,D5	7
D1,**D3**,D4,D5	4
D1,D4,D5	2
D1,D4,D5,D6	4
D1,**D2**,**D3**,D4,D5,D6	2
D1,D4,D7	2

Step A. Pruning Redundant

Datasets from each Entry

Pruned Entry	Direct Count
D1,D4,D5	5
D1,D4,D5	7
D1,D4,D5	4
D1,D4,D5	2
D1,D4,D5,D6	4
D1,D4,D5,D6	2
D1,D4,D7	2

Step B. Regrouping

UpPr({D1,D4},F)	Direct Count
D1,D4,D5	18
D1,D4,D5,D6	6
D1,D4,D7	2

3 entries only!!!

Evaluation - Datasets

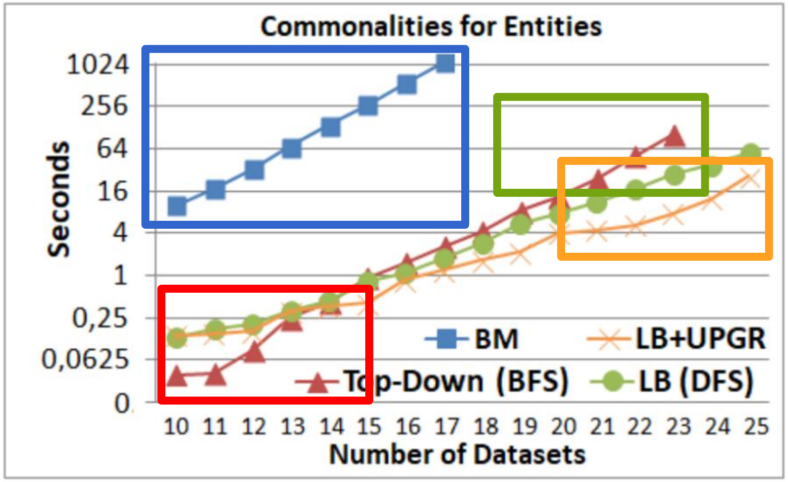
- We compute the metrics for **all the possible combination of datasets**, for 10-25 datasets
  - from 2^{10} (**1 thousand**) to 2^{25} (**33 million**) subsets of datasets
  - for Literals, Entity Index and Entity-triples Index
- We test the worst case
  - For finding **cmnBest(K,F),** there is no need to compute the metrics for all the possible combinations!

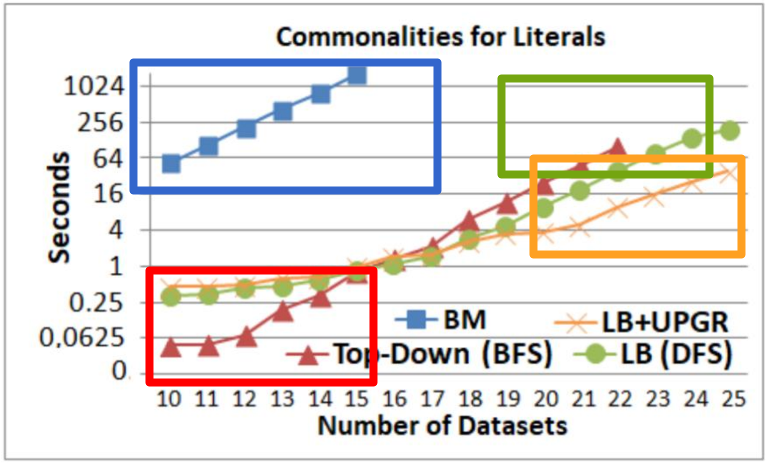
Index	Index Size	$\|occur(\mathcal{D}, F)\|$	Direct Count % of Index Size
Entities ($\|\mathcal{D}\| = 25$)	303 million	11,139	0.0036%
Literals ($\|\mathcal{D}\| = 25$)	353 million	64,907	0.0183%
Triples ($\|\mathcal{D}\| = 25$)	1.6 billion	5,250	0.0003%

- The size of our input (distinct posting lists) is extremely small!
  - **<0.02%** comparing to the **size of any index**

Key Results – Commonalities

- The **incremental** methods are far **faster** than the **Baseline Model (BM)**.
  - **BM** needs over 4 minutes for 2^{15} (**32,768**) subsets
  - **Incremental** approaches need **a few seconds** for **millions** of subsets
- **Top-Down** is faster for **a small number of datasets** (|D|<15)
  - It cannot be used for |D|>22 due to **memory issues.**
- **Bottom-up (LB DFS)** is faster than the **top-down** as we **add more datasets.**
- **Bottom-up with pruning and grouping** (LB+UPGR) is **faster** in most cases
  - For **1 million subsets** it needs ~**4 seconds** for Entity and Literals Index!
  - For **1 billion subsets (2^{30})** it needs **7.5 minutes** for Entity Index





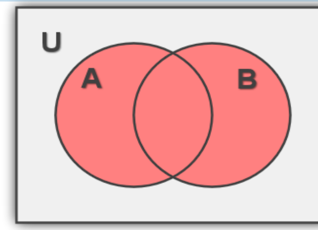
Key Results – Achieved Speedup

- Incremental Approaches versus SPARQL implementations
  - **Incremental** approaches: **4 seconds** for **1,000,000 subsets**
  - **SPARQL** implementations: 350 seconds for only 45 subsets

- Achieved Speedup
  - Even **4,921x speedup** by using a **lattice approach** vs a baseline model
  - Even **21x speedup** by using the **bottom-up** instead of top-down
  - Up to **5.61x speedup** by using **bottom-up with pruning and regrouping** versus the bottom-up approach

Max Speedup of	Entities	Triples	Literals
Top-Down vs BM	684×	446×	3,076×
LB (DFS) vs BM	1,409×	3,555×	4,350×
LB+UPGR vs BM	3,555×	1,785×	4,921×
LB (DFS) vs Top-Down	3.6×	21×	2.46×
LB+UPGR vs Top-Down	12.9×	11.7×	9.7×
LB+UPGR vs LB (DFS)	3.5×	-	5.61×

Coverage (Union)



- **occur(B,F):** the posting lists of an index containing at least one dataset Di that belongs to B
- The **sum** of the **directCount** of **occur(B,F)** gives the **cardinality** of **union** for a subset B

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

- *Baseline Model (BM):* For each subset B, scan all the posting lists once for finding occur(B,F), and sum their values

Query
Qcoverage

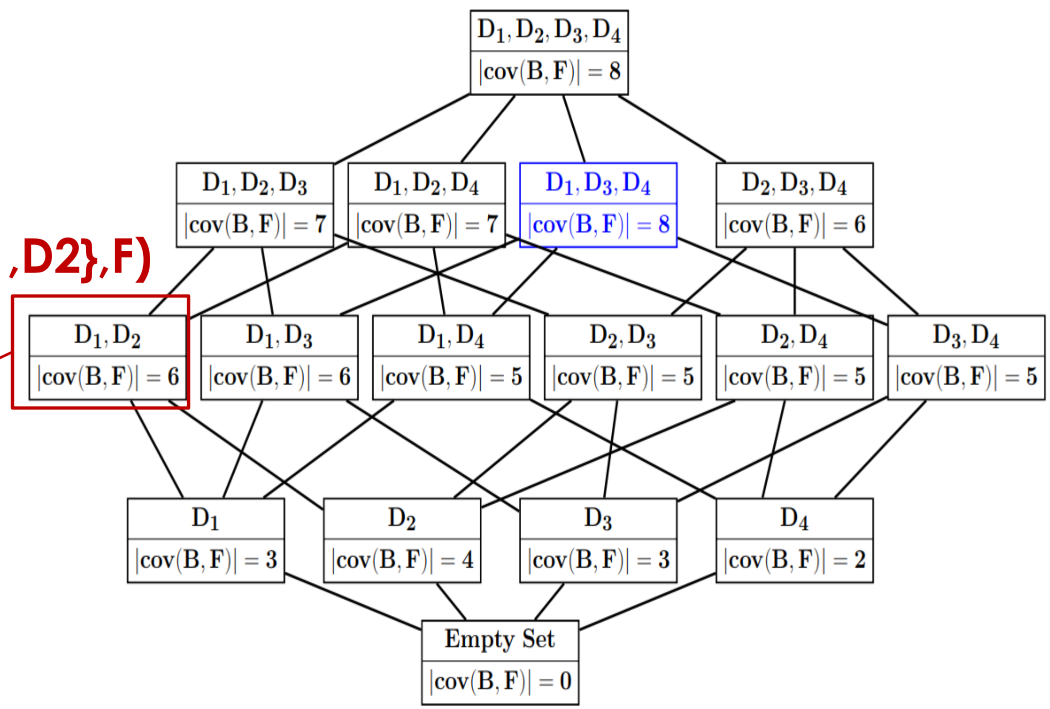
Scientist

"I want the 3 datasets whose union covers **the most unique triples** for Tiger**".**

Scan the Corresponding Index

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

occur({D1,D2},F)



The Challenge

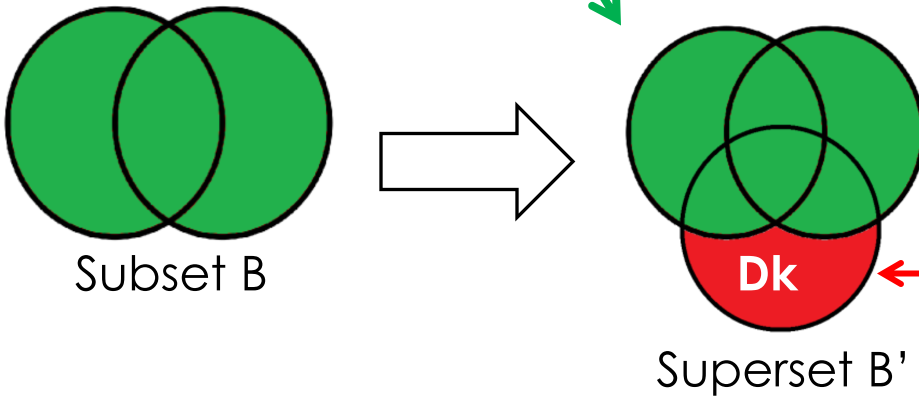
- *Baseline Model (BM):* Time-consuming to read all the **posting lists** for each possible **subset B.**
- **Target:** Read less posting lists for each B for finding **covBest(K,F)**

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

- **Solution**: Follow **a bottom-up DFS** traversal and use the following **set theory property**:
  - If **B' = B ∪ {Dk} →** |cov(B',F)|=|**cov(B,F)**|+|**F(Dk) \cov(B,F)**|

Already Computed

Just find this cardinality



Relative Complement of Dk to B

Bottom-Up Incremental Algorithm for Coverage

1. Find which posting lists contain the new dataset Dk
2. Take the sum of the directCount of these posting lists and of the coverage of the previous subset
3. Transfer |cov(B,F)| and posting lists that do not contain Dk

D1,D2,D3,**D4**	
D4 not included → {}	**D4 included→** {D4}
|cov({D1,D2,D3,D4},,F)|=|cov({D1,D2,D3},F)|+1=7	

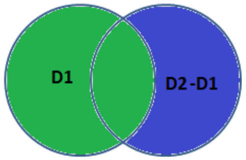
For covBest(3,F) stop here!

D1,D2,**D3**	
D3 not included → {D4}	**D3 included→** {D3}
|cov({D1,D2,D3},F)|=|cov({D1,D2},F)|+1=7	

D1,D2,**D4**	
D4 not included → {D3}	**D4 included→** {D4}
|cov({D1,D2,D4},F)|=|cov({D1,D2},F)|+1=7	

D1,**D2**	
D2 not included → {D3},{D4}	**D2 included→** {D2,D3},{D2,D4}
|cov({D1,D2},F)|=|cov(D1,F)|+3=6	

D1	
D1 not included {D2,D3},{D2,D4},{D3},{D4}	**D1 included** {D1},{D1,D2}
|cov(D1,F)|=3	



Input

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

Pruning and Regrouping for Coverage

- We managed to read **less posting lists** than the Baseline Model
- **Similarly to intersection**: Some datasets in the **posting lists** are redundant due to the **DFS order**
- Solution: Remove the redundant datasets from each posting list and regroup the remaining ones **(LB+PRGR approach)**

occur(D,F)	Direct Count
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

9 entries

Compute the coverage for all the supersets of D4 that have not been explored yet!

Step A. Pruning Totally Redundant Entries

occur(D >3,F)\B	Direct Count
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

6 entries

Step B. Pruning Redundant Datasets from each Entry

Pruned Entry	Direct Count
D4	10
D4,D5	2
D4,D5,D6	4
D4,D5	3
D4,D5,D6	7
D6,D7	4

Step C. Regrouping

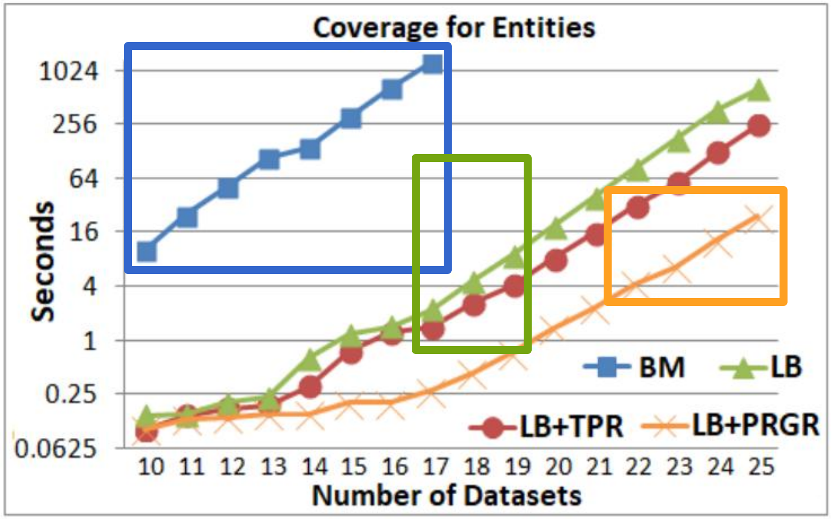
occurPr(D>3,F)\B	Direct Count
D4	10
D4,D5	5
D4,D5,D6	11
D6,D7	4

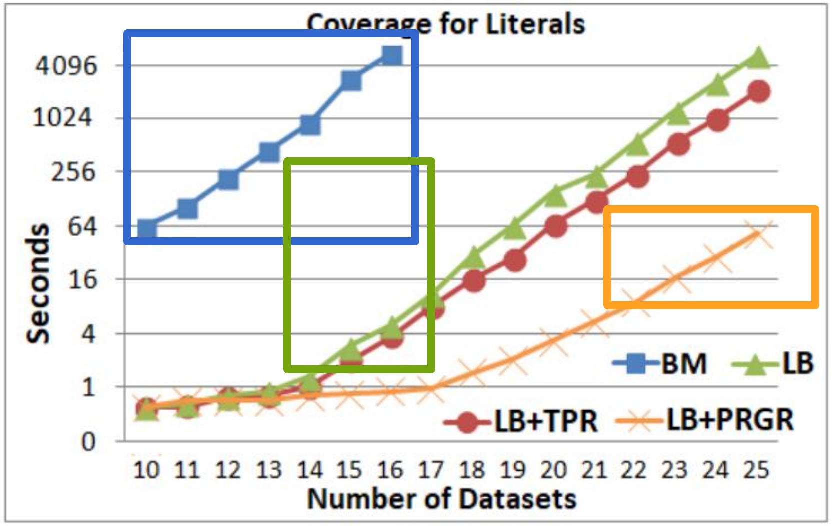
4 entries

Key Results for Coverage

Experiments for the **same datasets** as in **commonalities**

- The **incremental** models are far **faster** than a Baseline Model (BM)
- The **Bottom-up** approach (LB) is even **1,099x faster**
- Bottom-up with pruning and regrouping **(LB+PRGR)** is **faster** in all cases
  - **6,000x speedup** vs Baseline Model **97x speedup** vs the simple Bottom-up
  - 1 million subsets:**1.3 seconds** for **Entity Index** and **3.2 seconds** for **Literals Index**





Complement Metrics and More Experiments

Complement Metrics: Use almost the same algorithms as coverage.

- Information Enrichment (Absolute Complement)
  - We should remove the **posting lists** containing dataset Dm
- Uniqueness (Relative Complement)
  - We should **keep** the **posting lists** containing dataset Dm

- All the **details**, **proofs** and **more experiments** are included in **dissertation**

Query $Q_{enrichment}$

"I want 2 Datasets to **increase the number of triples** for the **entities of D3**"

Scientist (Publisher of D3)

Desired Dataset is D3

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

"Are the **triples** of my dataset (D2) **unique**?"

Scientist (Publisher of D2)

Desired Dataset is D2

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~~

Parallelization of Lattice Measurements

The Problem (Exponential Nature)

- The computation of measurements is time-consuming as the number of datasets **increases**.
  - More than **10 minutes** for **1 billion** subsets

The Challenge

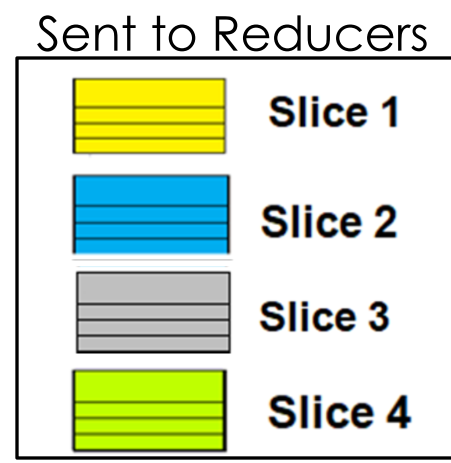
- With **m machines** and $2^{|D|}$ lattice nodes
  - each machine **mi** to compute $\mathbf{2^{|D|}/m}$ nodes

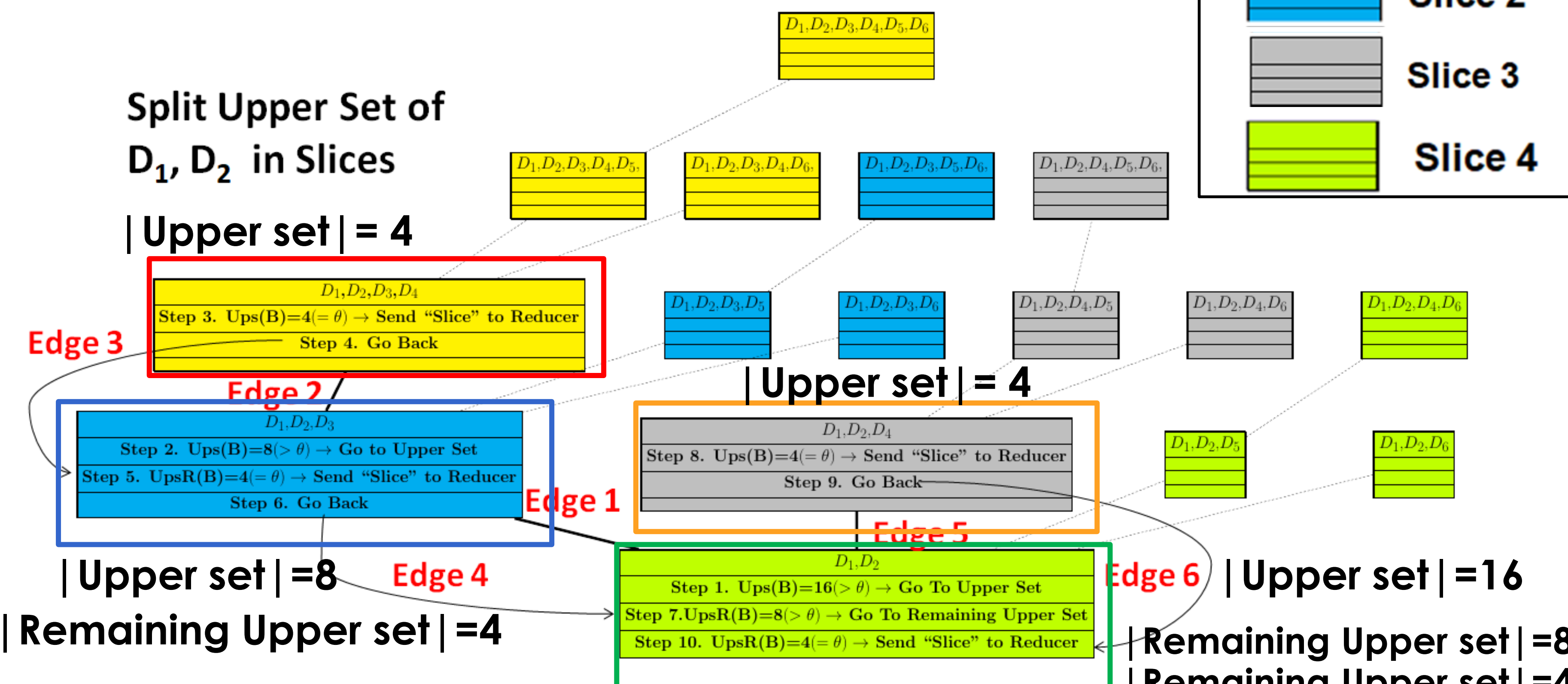
Solution

- We use a **parallel version** of **bottom-up** algorithm
  - Why a bottom-up approach?
    - It was **faster** comparing to top-down
    - it uses **depth-first traversal**
    - it computes the metrics for **the upper sets** of each node

How to Split Lattice Measurements in parts

- Datasets: 6
- Nodes: 64 (i.e., $2^{|6|}$)
- Machines: m=16
- **Threshold θ:** 64 /16=4
- Target: split the Lattice in **slices of 4 or less nodes**

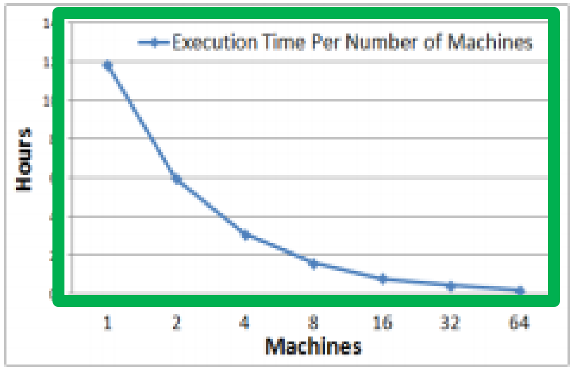




Key Results - Impact of Parallelization

- By splitting the lattice in very small pieces (by choosing a **small θ**)
  - We are very close to the **ideal case**!
  - Each machine computes the metrics for **almost the same number of nodes**
- We achieved over **55x speedup** by using **64 machines** instead of a single one
- We computed the metrics for
  - **1 billion** (2^{30}) subsets in ~1 minute (!)
  - **1 trillion** (2^{40}) subsets in ~6 hours (!)

Size of each "slice"	Number of "slices"	Maximum Nodes /all Nodes from one m_i	Distance from Ideal (Ideal is 1.56%)	Execution Time (Minutes)
≤ 1/4 of all Nodes	595	25.10%	23.54%	185.00
≤ 1/8 of all Nodes	596	18.80%	17.24%	147.00
≤ 1/16 of all Nodes	600	12.60%	11.04%	95.00
≤ 1/32 of all Nodes	611	7.90%	25.1%	59.00
≤ 1/64 of all Nodes	637	6.00%	6.34%	45.00
≤ 1/128 of all Nodes	694	4.10%	2.54%	31.50
≤ 1/256 of all Nodes	814	3.10%	1.54%	25.00
≤ 1/512 of all Nodes	1,061	2.85%	1.29%	22.50
≤ 1/1024 of all Nodes	1,563	2.79%	1.23%	20.20
≤ 1/2048 of all Nodes	2,576	1.64%	0.08%	12.10
≤ 1/4096 of all Nodes	4,612	1.62%	0.06%	12.30
≤ 1/8192 of all Nodes	8,695	1.60%	0.04%	12.50



Measurements for 35 datasets & 34.35 Billions of Nodes

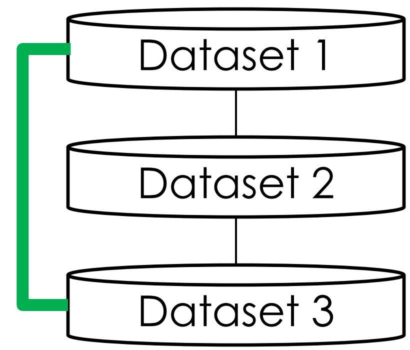
Contributions - Next Task

- Cross-dataset Identity Reasoning
- Semantics-aware Indexes at Global Scale
- Content-based Metrics for Dataset Discovery
  - Connectivity Analytics of LOD Cloud Datasets
- The LODsyndesis suite of Services

LOD Cloud Connectivity Measurements

Category	Value
owl:sameAs Triples	44,853,520
owl:sameAs Triples Inferred	73,146,062
RW Entities having at least two URIs	26,124,701
owl:equivalentProperty Triples	8,157
owl:equivalentProperty Triples Inferred	935
RW Properties having at least two URIs	4,121
owl:equivalentClass Triples	4,006
owl:equivalentClass Triples Inferred	1,164
RW Classes having at least two URIs	2,041

New Connections!



- (+) The **impact** of closure is promising for the entities.
  - **73 million inferred** owl:sameAs pairs (163% increase)
  - **2,700 newly** discovered connected pairs of datasets due to **closure!**
- (-) For properties and classes, the results are disappointing
  - Only a few inferred owl:equilaventProperty & owl:equivalentClass pairs
- **Key finding:** Publishers tend to connect more their entities than their schema elements with other datasets

LOD Cloud Connectivity Measurements (cont.)

Category	Connected Pairs	Connected Triads	Disconnected Datasets (of 400)
Real World Entities	9,075 (11.3%)	132,206 (1.24%)	87 (21.75%)
Literals	62,266 (78%)	4,917,216 (46.44%)	3 (0.75%)
Real World Triples	4,468 (5.59%)	35,972 (0.33%)	134 (33.5%)
Real Subject-Object Pairs	7,975 (10%)	107,083 (1%)	129 (32.2%)
Real World Properties	19,515 (24.45%)	569,708 (5.38%)	25 (6.25%)
Real World Classes	4,326 (5.42%)	53,225 (0.5%)	107 (26.7%)

Measurements for **pairs** of datasets

- Only 11.3% of pairs (9,075 in total) have at least **one entity** in common.
- 78% of them have common literals, only 5.59% share triples

Measurements for **triads** of datasets

- Only 1.2% of triads of datasets share **common entities**

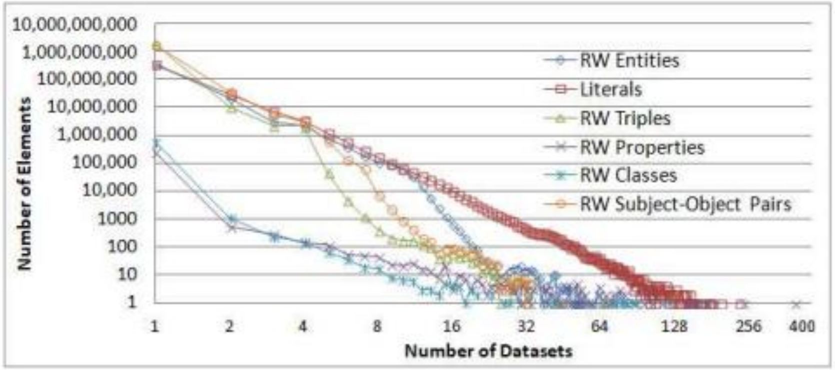
Key findings: Sparsity of LOD cloud

- A few connections for each dataset
- A large number of datasets are totally disconnected!

LOD Cloud Connectivity Measurements (cont.)

Other **Key Findings**

- Power law distribution
  - Most elements exist only in one dataset, only a few in many datasets
  - Most connected datasets share a few number of elements



- Most Connected Subset of Datasets
  - The **quad** of the **four popular** cross domain **datasets** (*Wikidata, DBpedia, YAGO and Freebase*) share
    - over 2.9 million entities, 3.4 million literals, 2.1 million triples

- Most Popular Datasets
  - Most datasets are **connected** with datasets from **cross-domain, publications** and **geographical domain**.
    - DBpedia, Wikidata, Freebase, YAGO, VIAF, GeoNames and others

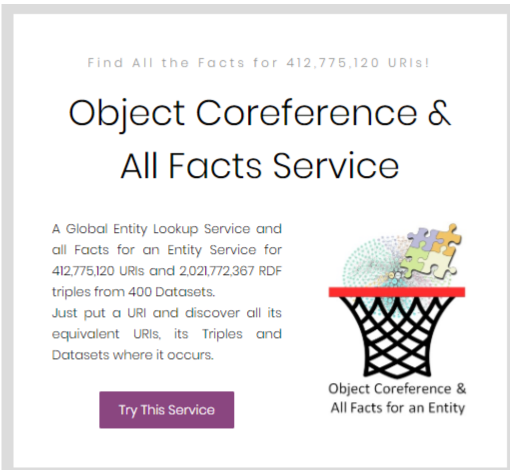
Check the thesis for finding much more experiments.

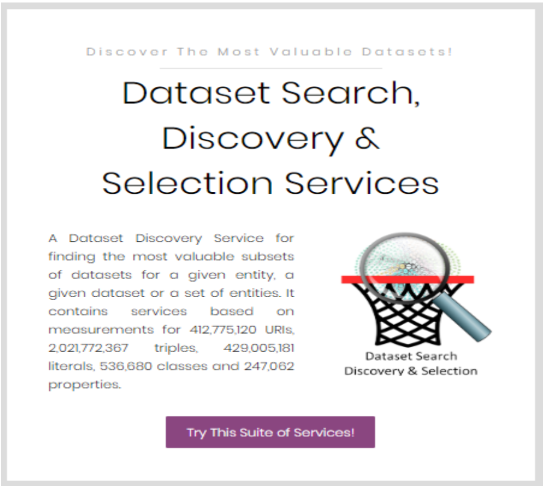
Contributions - Next Task

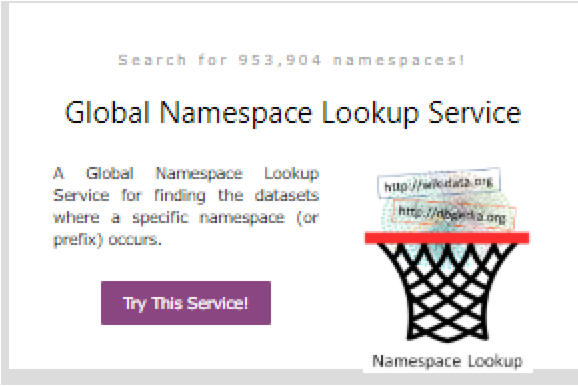
- Cross-dataset Identity Reasoning
- Semantics-aware Indexes at Global Scale
- Content-based Metrics for Dataset Discovery
- The LODsyndesis suite of Services

Services offered by LODsyndesis

- ***LODsyndesis*** offers **several online services** and **a REST API** which are based on the **indexes** and **measurements**, for 412 million URIs and 2 billion triples from 400 datasets.
- More details are given in thesis and in https://demos.isl.ics.forth.gr/lodsyndesis/







Research Prototypes

- Several **research prototypes** exploit ***LODsyndesis***!

Dataset Encrichment for Machine Learning

LODsyndesisML

LODsyndesisML is a research prototype that discovers and creates features for being used in any Machine Learning problem. It exploits Linked Data by connecting to LODsyndesis and can find features for entities of any domain.

LODsyndesisML

Learn More

Create Embeddings over Hundreds of Linked Datasets

LODVec

Services for creating and exploiting URI sequences and Embeddings for machine-learning and similarity tasks.

LODVec

Try the Service

Tools for Data Enrichment

Question Answering over Hundreds of Linked Datasets

LODQA Service

LODQA is a research prototype that exploits the semantics-aware indexes of LODsyndesis for answering questions expressed in Natural Language.

Learn More

Question Answering System

Interactive 3D Visualization of LOD Cloud

3DLOD Service

3DLOD is a research prototype that exploits the connectivity analytics of LODsyndesis and provides an interactive 3D visualization is already accessible.

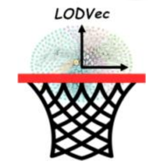
By clicking on a dataset the user can see the connected datasets.

Learn More

3D LOD Cloud Visualization

LODsyndesisML and LODVEC

- Applicable for Machine Learning tasks
  - **LODsyndesisML [14]** creates **features** from multiple datasets
  - **LODVec [15]** creates **embeddings** from multiple datasets



Running Example

- A user (**even non-familiar to RDF**) wants to
  - A. Predict the exact user rating for a set of movies
  - B. Find the top-K related movies for a given movie

Movies	**Rating**
Inception	85
Toy Story 2	?
…	….

- But we do not have any features ☹
  - We want to create **features** and **embeddings** for these movies by using multiple datasets
- We assume that **similar movies** will have **similar rating**

The Steps of these two Tools

A. Input & Configuration

Entities (URIs)
:Inception
:Toy Story 2
…

Input

Configuration

Datasets: DBpedia WIKIDATA VIAF

Features: Movies' Genre, Actors..

User Selections

B. Creation of Features and Embeddings from multiple datasets by using LODsyndesis

Movie	Has Award?	Is Thriller	Is Science Fiction Film?	Duration	Budget	# Actors	Actor Deg.	Political party	Y
Inception	1	1	1	148	1.6E8	2	1.5	Democratic	Good
Die Hard 2	0	1	0	124	7.0E7	2	1	Unknown	bad

SPARQL

LODsyndesisML

Entity e	Vector v(e)
:Inception	(0.123,0.287,…)
:Toy Story 2	(0.724,0.126,…)
…	…

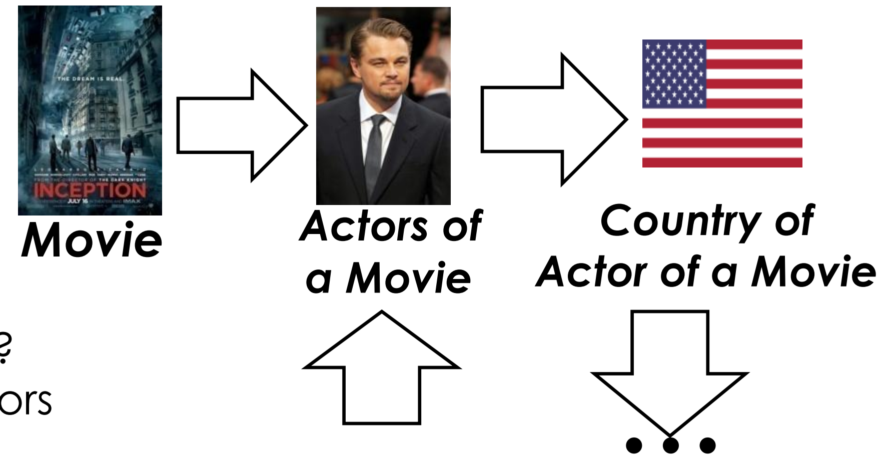
LODVec

C. Exploitation of Features and Embeddings

Machine Learning *Finding Similar Entities*

LODsyndesis$_{ML}$ – Features & GUI

- One can start from an entity of interest and **explore more "sub-entities"** by following **direct or undirected paths**!



- Features Categories
  - **Boolean:** Has a movie/actor won an Award?
  - **Count**: Number of Awards of a movie or actors
  - **Functional Features:** Movie Duration/Budget
  - **Most Frequent Value:** The nationality of most actors
  - **Degree:** The (graph) degree of a movie/actors

LODsyndesis-ML - Data Provenance:DBpedia

Property	Multiplicity	Property Range	Completeness	Boolean Instan...	Most Common Value	Average Nu...	Count for all ...
starring of	1-Many	URI	1		☐		☑
birthPlace	1-Many	URI	1		☐		☐
birthDate	1-Many	String	1		☐		☐
abstract	1-Many	String	1		☐		☐
wordnet_type	1 to 1	URI	0,933	☐	☐		
thumbnail	1 to 1	URI	0,867	☐	☐		
occupation	1-Many	URI	0,867	☐	☐		☐
birthYear	1 to 1	String	0,867	☐	☐		
birthName	1 to 1	String	0,867	☐	☐		
caption	1 to 1	String	0,8	☐	☐		
activeYearsStart...	1 to 1	String	0,667	☐	☐		
guests of	1-Many	URI	0,6	☐	☐		☐
children	1-Many	String	0,533	☐	☐		☐

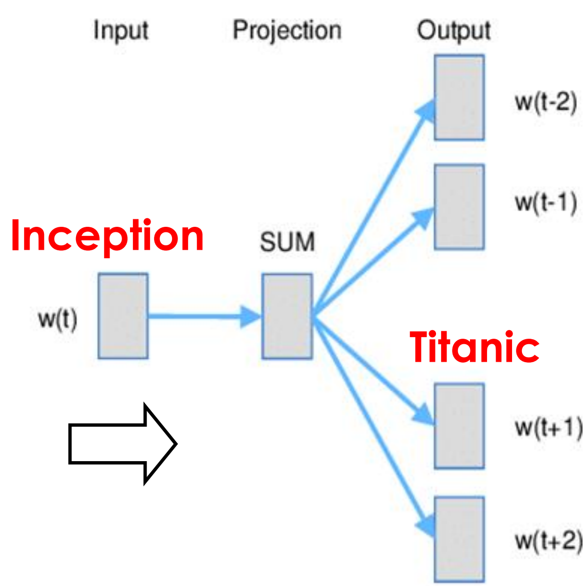
Characteristics *Metadata* *Feature Creation Operators*

LODVec – Creating Embeddings with Word2Vec

- We used indicatively the model **Word2vec**
  - a two-layer neural network that **converts text into vectors [10]**
  - we use the **skip-gram word2vec** model of DL4J library.
- Trains a **neural network** with **one hidden layer.**
- Guesses **potential neighboring entities**, based on the **entity** being analyzed.
  - **Example**: (Inception, Titanic) actor DiCaprio
    - *Inception, Titanic are expected to be close in the vector space*

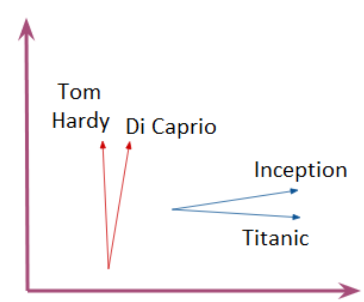
:Inception :genre :Thriller
:Inception :genre :Science Fiction
:Inception :actor :Di Caprio
:Toy Story 2 :genre :AnimatedFilm
:Toy Story 2 :genre :Comedy
......

Input: URI sequences



Skip-gram

Movie e	Vector v(e)
:Inception	(0.123,0.287,...)
:Toy Story 2	(0.724,0.126,...)
...	...



Output

Exploitation of Features & Embeddings

Movie e	Vector v(e)
:Inception	(0.123,0.287,...)
:Toy Story 2	(0.724,0.126,...)
...	...

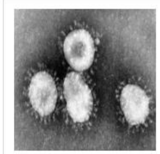
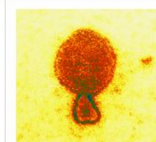
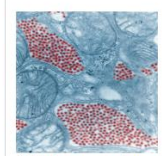
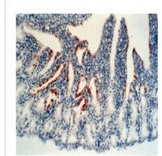
```
Entity,http://dbpedia.org/ontology/Work/runtime oneFeatureOneValue,Degree of ht
http://dbpedia.org/resource/Harry_Potter_(film_series),1179.0,416.666666666667
http://dbpedia.org/resource/Titanic_(1997_film),195.0,515.4,605,21,200.0,1
http://dbpedia.org/resource/The_Fast_and_the_Furious,0.0,358.0,63,3,759.0,1
http://dbpedia.org/resource/Shrek,90.0,487.0,96,3,60.0,1
http://dbpedia.org/resource/Transformers_(film_series),0.0,458.2,528,19,755.0,1
http://dbpedia.org/resource/Toy_Story,81.0,401.8,238,9,30.0,1
http://dbpedia.org/resource/The Karate Kid,127.0,226.6,152,12,8000000.0,1
```

- Machine Learning Tasks
  - LODVec exploits WEKA API [14]
  - Supports Classification & Regression tasks

Model	RMSE Value
Vote	13.971073114043271
Linear Regression	12.93772534818133
Support Vector Machine Regression	13.327412664437915

- Similarity Tasks
  - LODVec exploits DL4J Library
  - It can return the top-10 related entities to a given one

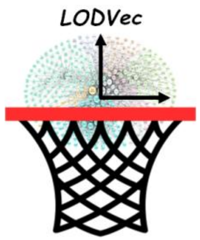
Give me 10 related **viruses** to the family of **Coronaviruses.**

Entity	Top-10 Related Entities
 Coronavirus	 1.Henipavirus
	 2.Eastern_equine_encephalitis_virus
	 3.Newcastle_disease

Key Results – LODsyndesisML & LODVec

- **Task**: Classify whether a movie is popular or not (**binary classification**)
  - Measure Accuracy: percentage of correct predictions
    - **Baseline Model**: 50% Accuracy

- **LODsyndesisML** (classified a set of 1,500 movies)
  - ✓ The accuracy of all the features was **87.1%**



- **LODVec** (classified a set of 2,000 movies)
  - ✓ The accuracy by creating embeddings
    - ✓ only from **DBpedia** was 71%
    - ✓ from **all the datasets** of **LODsyndesis** was **84.7%** (over 13% increase)

- **Key findings:** When we exploit **multiple datasets**
  - the number of possible features and embeddings **increases**
  - the accuracy of predictions **increases**.

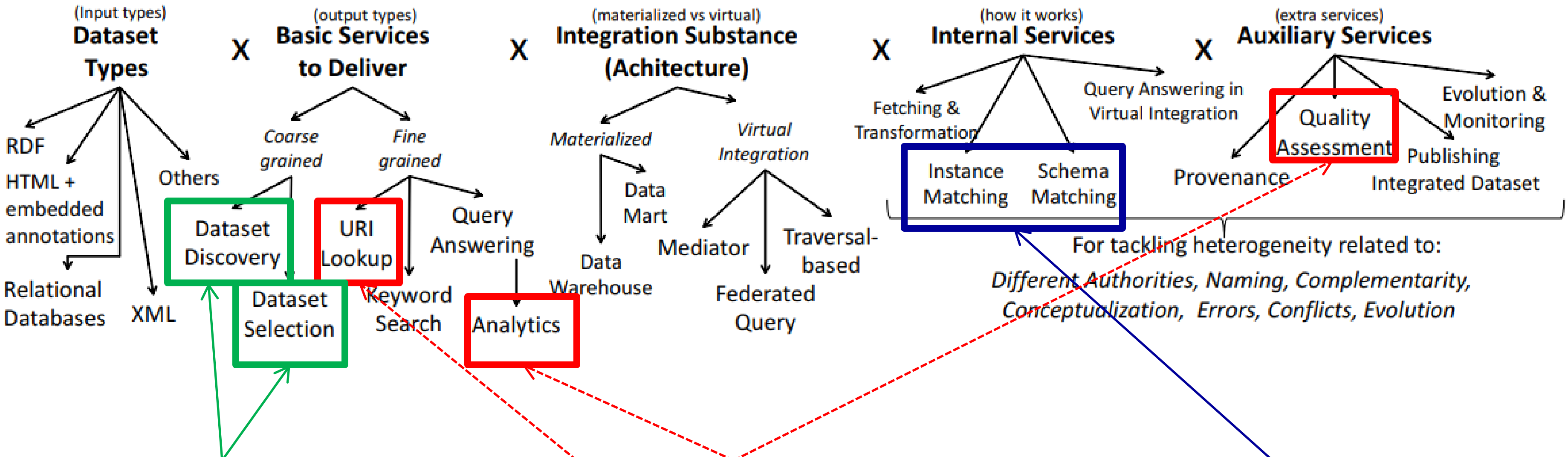
- Much more experiments are included in dissertation

Synopsis of Contribution and Future Work

Synopsis of Contributions

- We introduced a **survey** about **Large Scale Semantic Integration** of Linked Data
- We described **algorithms** for performing **cross-dataset identity reasoning** by using a **single** or a **cluster of machines**
- We introduced **MapReduce** methods for creating **five semantics-aware indexes**
- We proposed **content-based Dataset Discovery metrics** and **incremental algorithms** for their computation
  - We reported **connectivity analytics** over 400 Linked Datasets
- We presented the **LODsyndesis** suite of services
  - We gave emphasis on **LODSyndesisML** and **LODVEC**

Contributions wrt Data Integration Landscape



Content-based Dataset Discovery for millions of subsets in **less than 4 seconds**!

Fast Services based on indexes for 2 billion triples, constructed in **81 minutes**.

Computation of closure **in less than 10 minutes** for 44 million equivalence relationships!

We showed that the **proposed methods** can **scale** to **large number** of **Linked Datasets**!

Directions for Future Work

Data Integration

- Evaluation Collections and Reproducible Results: Propose **collections** and **challenges** for evaluating the **quality** of **automated methods** for fine-grained **data integration** and for providing comparative results
- Quality of Equivalence Relationships: Find **automatic ways** for improving the **quality of equivalence relationships**

Data Discovery

- Content-Based Metrics for Complex Queries: Answer queries requiring the **combination** of different **metrics.**
- Providing LOD Scale Analytics for a Dataset On-The-Fly: The proposed methods require that a dataset Di is **already indexed**.

Other tasks

- Exploitation of Indexes: Keyword Search, Instance and Schema Matching, and others.
- Embeddings over Large Number of Datasets: Create **longer URI sequences** and **vectors** through **other models** (e.g., GloVe).

Publications (2016-2020)

- (1) **M. Mountantonakis** and Y. Tzitzikas, On Measuring the Lattice of Commonalities Among Several Linked Datasets, Proceedings of the VLDB Endowment (PVLDB), 2016
- (2) **M. Mountantonakis** and Y. Tzitzikas, How Linked Data can aid Machine Learning based Tasks, 21st International Conference on Theory and Practice of Digital Libraries (TPDL), (pp. 155-168), September 2017, Thessaloniki, Greece
- (3) **M. Mountantonakis** and Y. Tzitzikas, Scalable Methods for Measuring the Connectivity and Quality of Large Numbers of Linked Datasets, ACM Journal of Data and Information Quality (JDIQ), 9(3), 15, 2018
- (4) **M. Mountantonakis** and Y. Tzitzikas, High Performance Methods for Linked Open Data Connectivity Analytics, Information MDPI 2018, 9, 134.(Special Issue Semantics for Big Data Integration)
- (5) **M. Mountantonakis** and Y. Tzitzikas, LODsyndesis: Global Scale Knowledge Services, Heritage, MDPI. Open Access Journal (ISSN 2571-9408), 1(2), 335-348.(Special Issue: On Provenance of Knowledge and Documentation: Select Papers from CIDOC 2018), 2018.
- (6) **M. Mountantonakis** and Y. Tzitzikas, Large Scale Semantic Integration of Linked Data: A survey, ACM Computing Surveys, 52(5), Sept. 2019
- (7) **M. Mountantonakis** and Y. Tzitzikas, Knowledge Graph Embeddings over Hundreds of Linked Datasets, 13th International Conference on Metadata and Semantics Research, Rome, Italy, October 2019
- (8) **M. Mountantonakis** and Y. Tzitzikas, Content-based Union and Complement Metrics for Dataset Search over RDF Knowledge Graphs, ACM Journal of Data and Information Quality (JDIQ), 2020

Other Publications (2016-2020)

- (9) **M. Mountantonakis** and Y. Tzitzikas, Services for Large Scale Semantic Integration of Data, ERCIM News 2017 (111), October 2017
- (10) **M. Mountantonakis** and Y. Tzitzikas, LODsyndesis: The Biggest Knowledge Graph of the Linked Open Data Cloud that Includes all Inferred Equivalence Relationships, ERCIM News 2018 (114), July 2018
- (11) **M. Mountantonakis**, N. Minadakis, Y. Marketakis, P. Fafalios, Y. Tzitzikas, Connectivity, Value, and Evolution of a Semantic Warehouse. In Innovations, Developments, and Applications of Semantic Web and Information Systems (pp. 1-31). IGI Global, 2018
- (12) ME Papadaki, P Papadakos, **M Mountantonakis** and Y Tzitzikas, An Interactive 3D Visualization for the LOD Cloud, EDBT/ICDT Workshops, 100-103 , 2018
- (13) E. Dimitrakis, K. Sgontzos, **M. Mountantonakis,** and Y. Tzitzikas, Enabling efficient question answering over hundreds of linked datasets, Proceedings of the ISIP Workshop, 2019

Systems & Tutorial Videos

- Web pages of Systems
  - LODsyndesis
    - http://www.ics.forth.gr/isl/LODsyndesis
  - LODsyndesisML
    - https://demos.isl.ics.forth.gr/lodsyndesis/LODsyndesisML
  - LODVec
    - https://demos.isl.ics.forth.gr/lodvec
  - LODQA:
    - https://demos.isl.ics.forth.gr/LODQA
- Videos of Systems
  - LODsyndesis: https://youtu.be/UdQDgod6XME
  - LODsyndesisML: https://youtu.be/S_ILRTZarjA
  - LODVec: https://youtu.be/qR9RFZVs4TY
  - LODQA: https://youtu.be/bSbKLlQBukk

Acknowledgements

H.F.R.I.
Hellenic Foundation for Research & Innovation

University of Crete
Computer Science Department

FORTH
INSTITUTE OF COMPUTER SCIENCE
FORTH-ICS

isl

okeanos

BlueBRIDGE

Thank You!

References

- [1] Maulik R Kamdar and Mark A Musen. 2017. PhLeGrA: Graph analytics in pharmacology over the web of life sciences linked open data. In Proceedings of World Wide Web Conference. 321–329
- [2] X. L. Dong, B. Saha, and D. Srivastava. Less is more: Selecting sources wisely for integration. In Proceedings of the VLDB Endowment, volume 6, pages 37–48. VLDB Endowment, 2012
- [3] Michalis Mountantonakis and Yannis Tzitzikas, Large Scale Semantic Integration of Data with emphasis on Linked Data: A Survey, ACM Computing Surveys, 2019
- [4] Laurens Rietveld, Wouter Beek, and Stefan Schlobach. 2015. LOD lab: Experiments at LOD scale. In ISWC. Springer, 339–355.
- [5] Javier D Fernández, Wouter Beek, Miguel A Martínez-Prieto, and Mario Arias. 2017. LOD-a-lot. In ISWC. Springer, 75–83
- [6] Markus Nentwig, Tommaso Soru, Axel-Cyrille Ngonga Ngomo, and Erhard Rahm. 2014. LinkLion: A Link Repository for the Web of Data. In The Semantic Web: ESWC 2014 Satellite Events. Springer, 439–443
- [7] Ivan Ermilov, Jens Lehmann, Michael Martin, and Sören Auer. 2016. LODStats: The Data Web Census Dataset. In International Semantic Web Conference. Springer, 38–46
- [8] Max Schmachtenberg, Christian Bizer, and Heiko Paulheim. 2014. Adoption of the linked data best practices in different topical domains. In The Semantic Web–ISWC 2014. Springer, 245–260

References (cont.)

- [9] Pierre-Yves Vandenbussche, Jürgen Umbrich, Luca Matteis, Aidan Hogan, and Carlos Buil-Aranda. 2016. SPARQLES: Monitoring public SPARQL endpoints. Semantic Web Preprint (2016), 1–17
- [10] Semih Yumusak, Erdogan Dogdu, Halife Kodaz, Andreas Kamilaris, and Pierre-Yves Vandenbussche. 2017. SpEnD: Linked Data SPARQL Endpoints Discovery Using Search Engines. IEICE TRANSACTIONS on Information and Systems 100, 4 (2017), 758–767.
- [11] Valdestilhas, A.; Soru, T.; Nentwig, M.; Marx, E.; Saleem, M.; Ngomo, A.C.N. Where is my URI?
- [12] Hugh Glaser, Afraz Jaffri, and Ian Millard. 2009. Managing co-reference on the semantic web. (2009).
- [13] Rastogi, Vibhor, et al. "Finding connected components in map-reduce in logarithmic rounds." 2013 IEEE 29th International Conference on Data Engineering (ICDE). IEEE, 2013.
- [14] Ian H Witten, Eibe Frank, Mark A Hall, and Christopher J Pal. Data Mining: Practical machine learning tools and techniques. Morgan Kaufmann, 2016
- [15] M. Mountantonakis, and Y. Tzitzikas, "How linked data can aid machine learning-based tasks." International Conference on Theory and Practice of Digital Libraries. Springer, Cham, 2017.
- [16] M. Mountantonakis, and Y. Tzitzikas, 2019, Knowledge Graph Embeddings over Hundreds of Linked Datasets. In Research Conference on Metadata and Semantics Research (pp. 150-162)