COST Action Hackathon Querying Federations of Knowledge Graphs

Report presentations

27 April 2023

EUROPEAN COOPERATION IN SCIENCE & TECHNOLOGY

Tool provider 1: Comunica

COMUNICA

Approach: Federation algorithm optimized for TPF interfaces

Splitting query up into triple patterns –> Bind Join

Trivially supports heterogeneous interfaces (incl SPARQL endpoints)

Our question for this workshop:

How well does this work for SPARQL endpoint federations?

Answer:

Better than expected, even for complex queries and many sources.

Will never perform as good algorithms dedicated to SPARQL federation.

Tool provider 1: Comunica

COMUNICA

Main changes:

- Bug: **cardinalities** were obtained through COUNT queries, but were **getting lost** -> Infinity was always used during **query planning.**
- Bug: Incorrect internal metadata -> **NLJ** was always picked.
- Optimization: Many (repeated) **COUNT** queries -> **caching** helps!
- When many virtual endpoints on 1 machine: **limiting number of parallel** queries, or enabling retries.

=> Changes will be included in upcoming Comunica release.

Tool provider 1: Comunica

Attempted optimizations (but no significant effect):

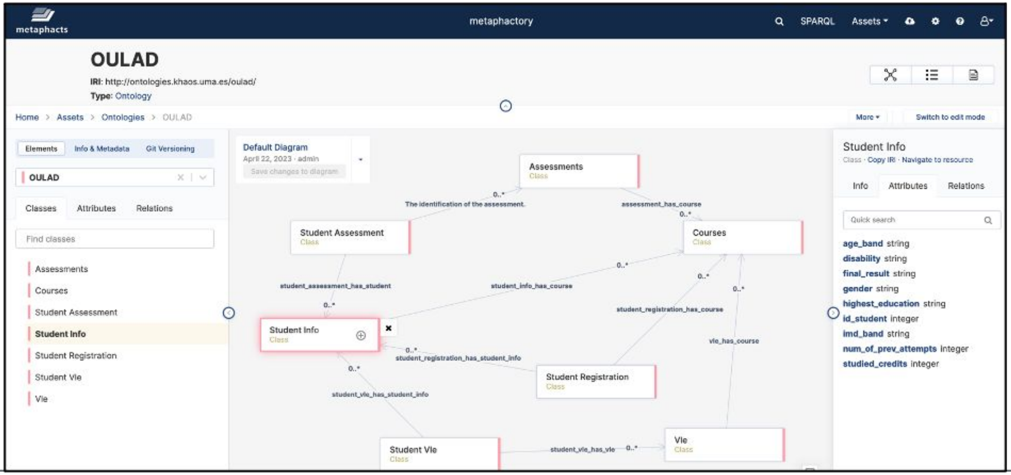
- Setting **timeouts** on COUNT queries.
- Derive cardinalities from **less selective** cached triple patterns.

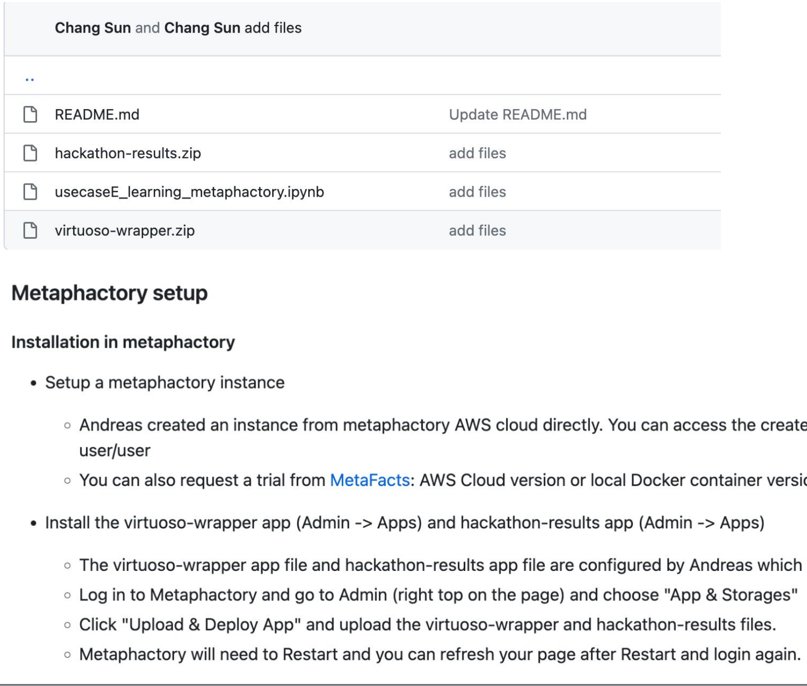
Possible future optimizations:

- Pushing down **FILTERs** into sources
- **Exclusive groups** (difficult in heterogeneous federations)

Tool provider 2: metaphactory

Find the report slides in https://drive.google.com/file/d/1BP5i2aSbae4xW6BdfD8Ot1LcnCi-6f0J/view?usp=share_link





Chang Sun and **Chang Sun** add files

..

README.md	Update README.md
hackathon-results.zip	add files
usecaseE_learning_metaphactory.ipynb	add files
virtuoso-wrapper.zip	add files

Metaphactory setup

Installation in metaphactory

- Setup a metaphactory instance
  - Andreas created an instance from metaphactory AWS cloud directly. You can access the create user/user
  - You can also request a trial from MetaFacts: AWS Cloud version or local Docker container versi
- Install the virtuoso-wrapper app (Admin -> Apps) and hackathon-results app (Admin -> Apps)
  - The virtuoso-wrapper app file and hackathon-results app file are configured by Andreas which
  - Log in to Metaphactory and go to Admin (right top on the page) and choose "App & Storages"
  - Click "Upload & Deploy App" and upload the virtuoso-wrapper and hackathon-results files.
  - Metaphactory will need to Restart and you can refresh your page after Restart and login again.

Tool provider 3: ColChain

Use case 3: Life Science

Goal for this use case

How well would a decentralized environment like ColChain fare for queries with very large intermediate results like Bio2RDF?

Network setup

Building a small P2P network: Each group member…

- …set up a node on their PC (guide in GitHub)
- …upload one of the datasets to their node
- …try to execute the query

Tool provider 3: ColChain

Use case 3: Life Science

Outcomes

```
SELECT DISTINCT ?drug ?go
{
   ?drug a dv:Drug .
   ?drug dv:target ?t .
   ?t dv:x-hgnc ?x .
   ?x bv:x-uniprot ?uniprot .
   ?uniprot ?p ?go .
   FILTER (?p = goa:process || ?p = goa:component
|| ?p = goa:function)
}
```

Main points:

- Uses predicates for source selection
- Predicates for variables mapped to all datasets (although most are pruned)

- More than 300,000 intermediate results!
  - => more than 10,000 requests!
- Not optimized for FILTER operations

Improvement ideas:

- Downloading data fragments before execution
- Optimize FILTER operations
- Compression of results
- ... more improvements outlined in Google Drive

Tool provider 3: ColChain

Use case 2: Federated Shop

Goal for this use case

Can ColChain efficiently simulate a federated network with a large number of sources?

Network setup

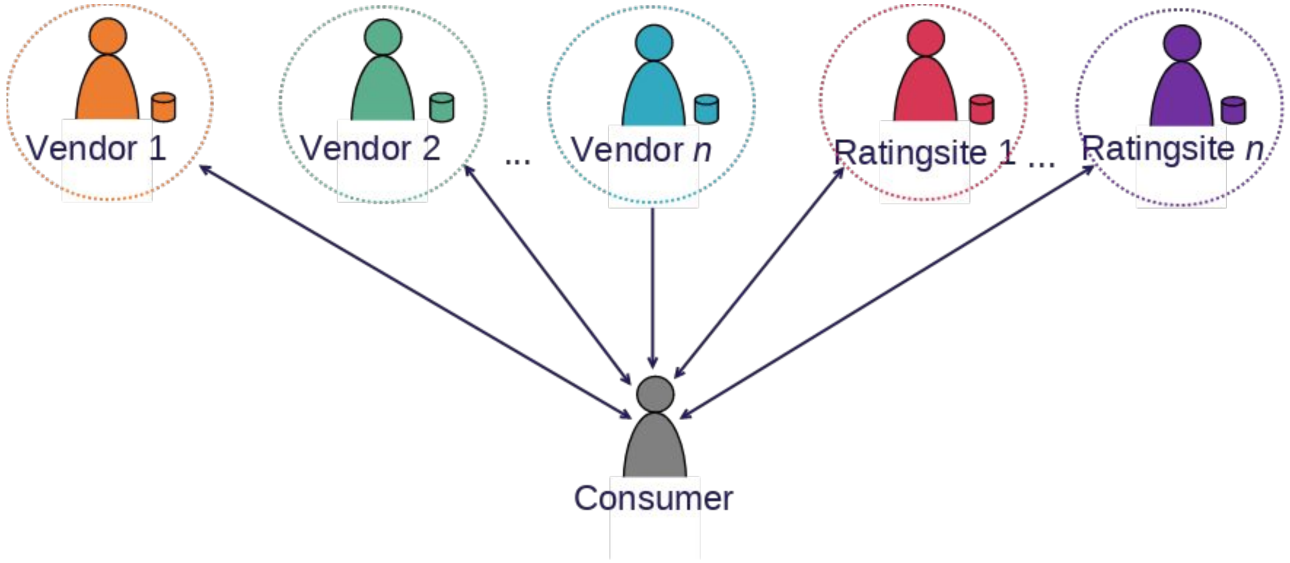
Building the P2P networks:

- We set up a network for each configuration (20 and 100 sources)
- Each shop/ratingsite has one node (simulating federated environment)
- Running queries using a local 'consumer' node

Tool provider 3: ColChain

Use case 2: Federated Shop

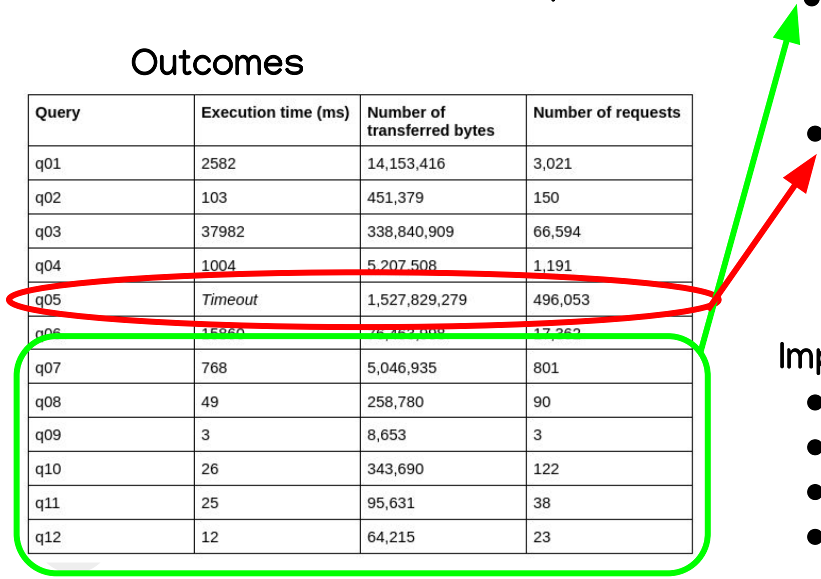
Setup:



Tool provider 3: ColChain

Use case 2: Federated Shop

Outcomes



Query	Execution time (ms)	Number of transferred bytes	Number of requests
q01	2582	14,153,416	3,021
q02	103	451,379	150
q03	37982	338,840,909	66,594
q04	1004	5,207,508	1,191
q05	*Timeout*	1,527,829,279	496,053
q06	15860	76,463,998	17,362
q07	768	5,046,935	801
q08	49	258,780	90
q09	3	8,653	3
q10	26	343,690	122
q11	25	95,631	38
q12	12	64,215	23

Main points:

- Most queries were very efficiently processed
  - ColChain generally scaled well with a large number of sources
- A few queries were challenging due to:
  - FILTER expressions
  - Large intermediate results (many requests)

Improvement ideas:

- Downloading data fragments before execution
- Optimize FILTER operations
- Compression of results
- … more improvements outlined in Google Drive

Tool provider 4 - HeFQUIN

Tool provider 5: Semagrow & KOBE

1) FedShop Use-Case

- **Activities:**
  - Provided a **Semagrow** setup for the use-case (100 “endpoints”)
  - Experimented with 2 types of summaries (only Void, Sevod w, URI prefixes)
  - Sevod medata improves our source selection, but execution plan is not efficient
  - FedShop team tested Sevod-Scraper (tool for providing medata) provided feedback
  - Worked on **KOBE** configuration files for the FedShop experiment
- **Future Work:**
  - Improve Sevod-scraper tool (the part that generates metadata from endpoint)
  - Make Semagrow able to query specific graph of a SPARQL endpoint
  - Further analyze the queries to Improve our source selection
  - Further integration of FedShop benchmark within KOBE

Tool provider 5: Semagrow & KOBE

2) LifeScience Use-Case

- **Activities:**
  - Provided a **Semagrow** setup for the use-case, focus on Q3
  - Deployed 3 federated Virtuoso endpoints from RDF dumps provided using **KOBE**
  - Tried to executed the query, but without the filter (the filter optimizer tried to optimize the OR operators of the filter but created a weird plan)
  - Documanted the overall process, realized that there is missing documentation for setting up the prerequisites of KOBE (Kuberentes environment)
- **Future Work:**
  - Update Sevod-scraper for automatically creating more refined
  - Further analyze the query provided by LifeScience to fix this bug
  - Improve the documentation of KOBE

Tool provider 6: CostFed

- E-learning Use-case
  - `SparqlTripleSource class`
  - `private boolean useASKQueries = false;`
  - Index creating took 45 minutes on server
  - Index size 0.3 MB
  - On my laptop with 4 GB, I was unable to compute index
  - Q1 196 Seconds, Q2 21.8 Seconds, Q3 Error due to non standard SPARQL function big:datedif, Q4: 7.4 Seconds
  - For all queries only a single relevant source per triple pattern and total 2 distinct sources required to get complete results
  - Mostly exclusive groups were created

Tool provider 6: CostFed

- **Federatedshop Use-case**
  - Virtual SPARQL endpoints: named graphs part of the SPARQL endpoint Url
  - On 4GB laptop, the index for 18 datasets were created in 2 minutes
  - 9.2 MB size for index
  - Error: single source query
- Lessons Learned
  - More clean documentation with step by step instructions on how to setup and run
  - Index creation can be time consuming on normal user laptops with less than 8 GB Ram
  - The implementation of the query planner can be improved
  - Dockerfile in order to be integrated in KOBE

Use case 1
E-learning

Managing Learning Data through Federated Queries
on Knowledge Graphs

María del Mar Roldán-García
Manuel Paneque
University of Málaga

EUROPEAN COOPERATION
IN SCIENCE & TECHNOLOGY

Description

- A known domain -> e-learning (Students, courses, assessments, submissions, etc.)
- Real use case with real data (not an academic use case)
- Five datasets
- Millions of triples
- Complex SPARQL queries (group by, count, functions)
- Each query needs only two datasets to be solved

Challenge:

Define federated SPARQL queries in the e-learning domain having only an ontology and a brief description of the content of the datasets.

Some statistics

SPARQL Endpoint	Nº of triples	Network usage
https://student-oulad.khaos.uma.es/sparql	20.238.494	12,4 GB
https://module-oulad.khaos.uma.es/sparql	34.279	3 GB
https://assignment-mud.khaos.uma.es/sparql	325.600	25,8 GB
https://user-mud.khaos.uma.es/sparql	59.634	14,6 GB
https://log-mud.khaos.uma.es/sparql	137.133.001	158 GB

Lessons learned

- **Scalability issues**
  - Virtuoso and Metaphactory & Fedx can deal with all queries
  - Other tools present problems with the size of the datasets
  - Query-intensive approaches seem to be unsuitable for this use case

- **Expressivity issues**
  - Standard SPARQL is not enough to solve the queries
    - Complex date handling is often necessary in real applications

Lessons learned

- **Non-expert users issues**
  - Having an ontology is not enough to solve queries.
  - Providing a good documentation of the ontology is crucial.
  - Extra metadata are necessary.
  - The use case requires advanced SPARQL knowledge
  - Graphical interfaces are needed to assist in the construction of the queries.
- **Expert users issues**
  - Users familiar with the ontologies and the datasets
    - Solve queries is easier with Meta than with Virtuoso
    - Focus on queries, not in the federation

Use case 2
The Federated Shop

Scalability benchmark for federated-query engines

Authors: Minh-Hoang DANG [1], Olaf HARTIG [2], Hala-Skaf MOLLI [1], Pascal MOLLI [1], Yotlan LE CROM [1], Julien AIMONIER-DAVAT [1]
[1] Nantes University (France), [2] Linköping University (Sweden)

cost
EUROPEAN COOPERATION IN SCIENCE & TECHNOLOGY

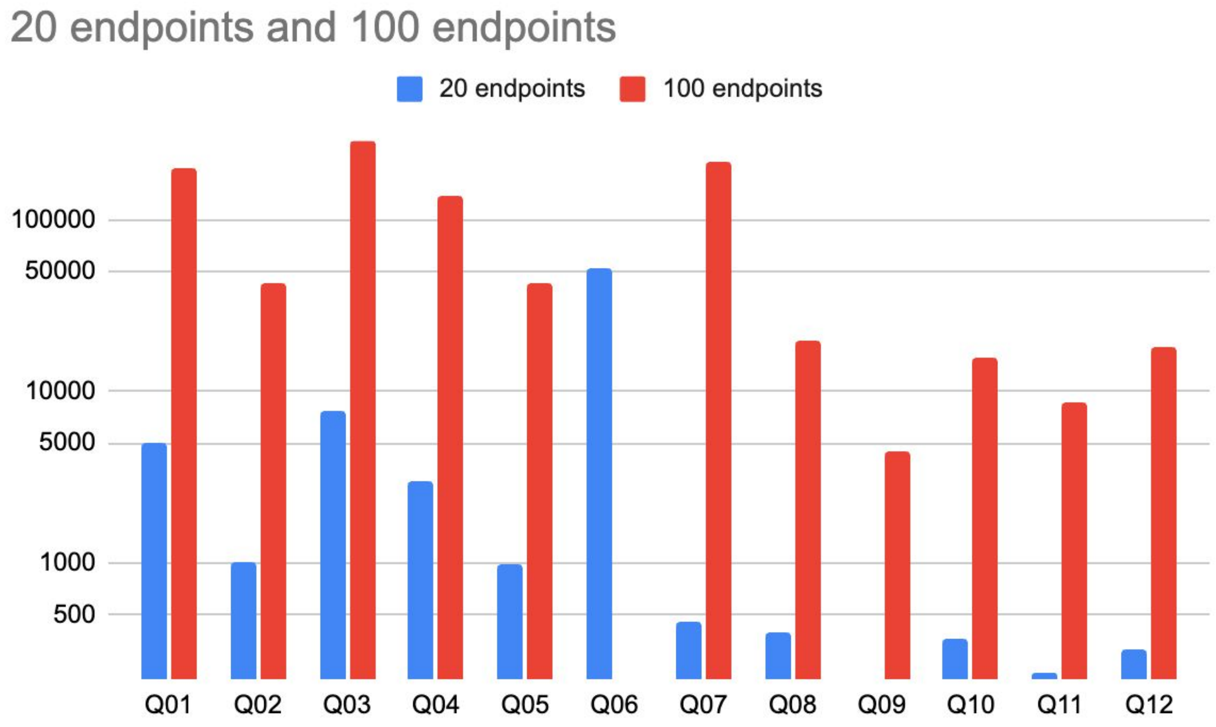
Recapitulation

- 2 configurations
  - 20 endpoints: 10 vendors + 10 rating-sites (~6M triples)
  - 100 endpoints : 50 vendors + 50 rating-sites (~28M triples)
- 12 queries simulating a user browsing a e-commerce virtual shop (as proposed in BSBM)
- Question:
  - Do existing federated-query engines scales?
  - Ie . we multiply number of source per 5, what is the effect on execution time with different engines ??

FedX results

Specs:
– MacOS M1
– 8GB RAM

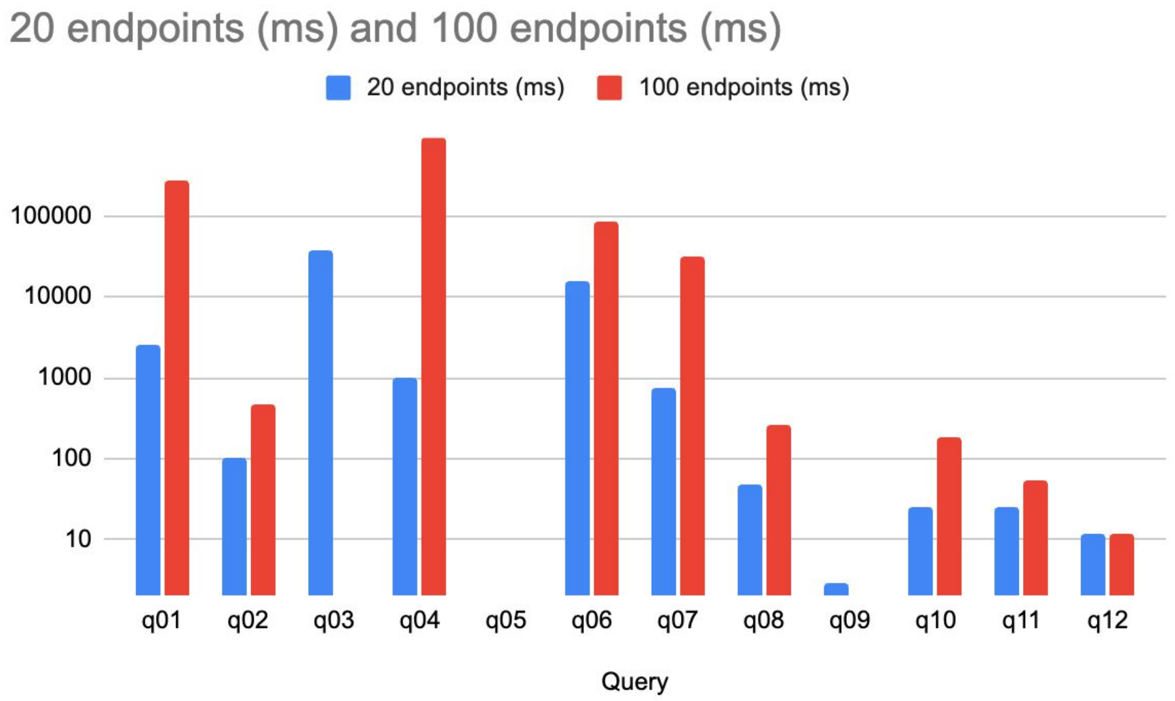
	20 endpoints (ms)	100 endpoints (ms)
Q01	5036	199200
Q02	1021	42307
Q03	7693	287680
Q04	3035	137462
Q05	981	42418
Q06	52238	*timeout*
Q07	450	215610
Q08	393	19671
Q09	208	4476
Q10	366	15953
Q11	231	8678
Q12	318	17916



EUROPEAN COOPERATION IN SCIENCE & TECHNOLOGY

Colchain results

Query	20 endpoints (ms)	100 endpoints (ms)
q01	2582	271592
q02	103	473
q03	37982	*Timeout*
q04	1004	917683
q05	*Timeout*	*Timeout*
q06	15860	86199
q07	768	31363
q08	49	256
q09	3	2
q10	26	185
q11	25	54
q12	12	12



EUROPEAN COOPERATION IN SCIENCE & TECHNOLOGY

Summary

- Ideal = Hand-Crafted SPARQL 1.1 queries with minimal source selection.
- Results are rounded up.
- Engines are NOT tested using the same resources (NOT comparable).
- Comunica provide partial results.
- CostFed/Semagrow could not deliver results during the Hackathon. (CostFed results were obtained offline).

	20 endpoints	**100 endpoints**
Ideal	51 ms	81ms
FedX	5 s	125 s
CostFed	86 s	155s
ColChain	6 s	126 s
Comunica	11 s	56 s (partial)
Semagrow	N/A	N/A

Conclusions

- Current federated-query engine **do not scale** (as expected)
- Increasing the number of endpoint **reveals different bugs** in federated-query engines.
  - FedX, CostFed, SemaGrow, ColChain, Comunica
- For index-assisted federated-query engines (CostFed,SemaGrow):
  - Need for **common practices** for producing/maintaining endpoints' summaries. (e.g, only require a list of URLs as FedX).
- Compared to the hand-crafted SPARQL-1.1 queries, the room of improvement is considerable.

cost
EUROPEAN COOPERATION
IN SCIENCE & TECHNOLOGY

Use case 3: Life Science

Authors: Remzi Celebi, Vincent Emonet, Maryam Mohammadi, Chang Sun and Michel Dumontier
Maastricht University

Tools:

ColChain

Semagrow & KOBE

Metaphactory & FedX

HeFQUIN

Challenges

Challenge 1: Same entities having different URIs in different resources

Example: matching of the IRIs of the genes:
http://identifiers.org/ensembl/ in the kg-covid-19 and
https://identifiers.org/ensembl/ in the WikiPathways.

Challenge 2: Would it be possible to split the datasets and run the query on different endpoints to achieve better load balancing?

Use case 3: Life Science

ColChain

Managed to execute the Bio2RDF query on distributed **nodes,** but slow

HeFQUIN

This tool can resolve challenge 1 as it requires users to provide vocabulary mapping as an input.

Example: For query 1, We added "http: gene Same as https: gene" as input.

Use case 3: Life Science

Metaphactory & **FedX**

Succeed to run the Bio2RDF query against local endpoints, quite fast. No time to test this query on the remote endpoints

Semagrow & KOBE

Managed to execute the alternation of the Bio2RDF query as the execution of the original query with filter could not be handled by the tool.

To use Semagrow it is required to provide an endpoint and data dump both as input for each resource.

Use case 3: Life Science

Next steps:

- **Follow-up discussion to deploy ColChain on the whole (a bigger subset) of Bio2RDF**
- **Defining more challenging queries**
- **On-the-fly solutions for identifier matching (finding undefined sameAs links)**
    - http://identifiers.org/ensembl/ENSG00000008710
    - https://identifiers.org/ensembl/ENSG00000008710
    - http://bio2rdf.org/ensembl/ENSG00000008710
    - “ENSG00000008710”^^xsd:string

Use case 3: Life Science

Identified challenges for use case providers:

ColChain: User interface

Semagrow & KOBE
Required input

Metaphactory & FedX
License

Achievements:

- Documentations to run and set up requirements for tools
- Identified potential of colchain to be used as future infrastructure for efficient query execution of Bio2RDF

EUROPEAN COOPERATION IN SCIENCE & TECHNOLOGY

THANK YOU