

Keywords-To-SPARQL translation for RDF data search and exploration

Katerina Gkirtzou¹ Kostis Karozos² Vasilis Vassalos² Theodore Dalamagas¹

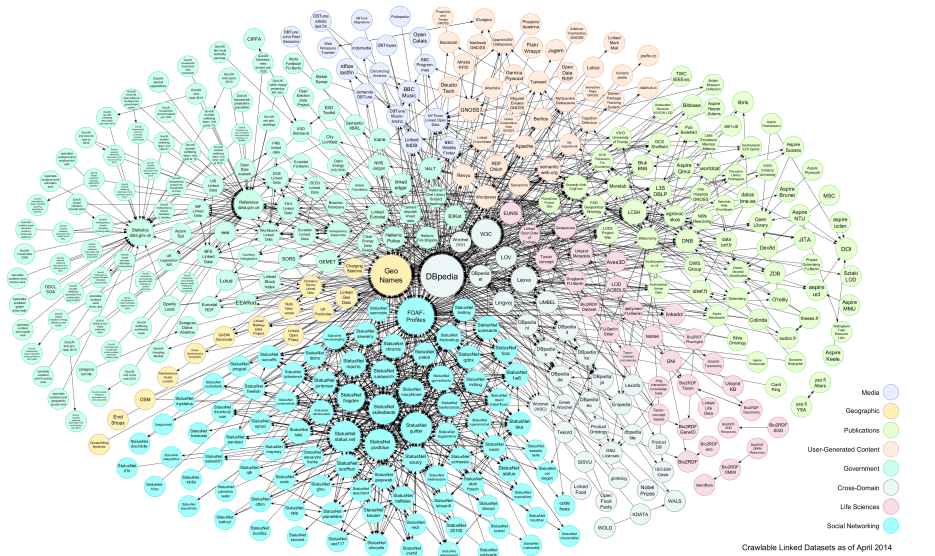


TPDL 2015 - 17th September, 2015

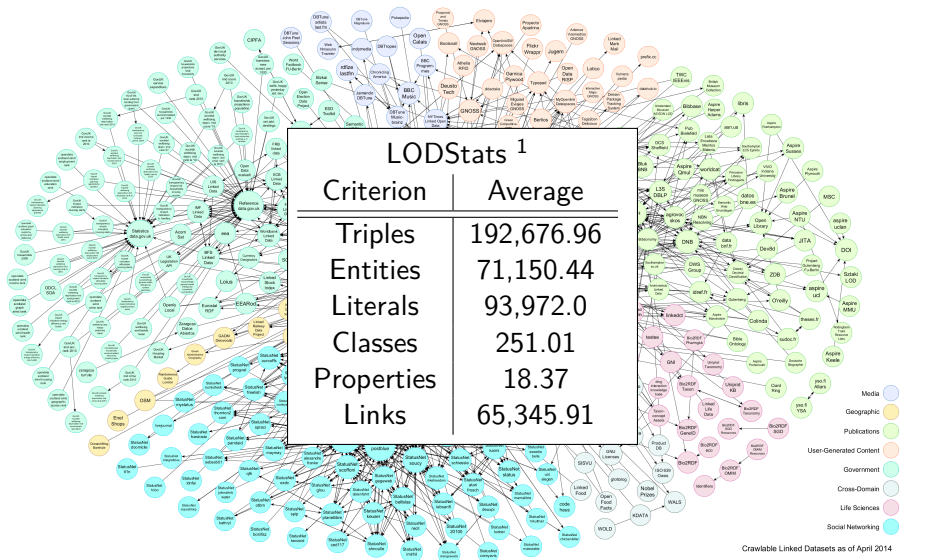
Table of Contents

- 1 Introduction
- 2 Keyword-To-SPARQL Algorithm
 - Indices
 - Generating SPARQL Queries
 - Method Overview
 - Temporal Operators
- 3 Preliminary Evaluation
- 4 Conclusion

Linked Data Cloud



Linked Data Cloud



¹Examined 3426 datasets - <http://stats.lod2.eu/>

Exploring LOD

Common methods

- RDF browsers
- SPARQL Queries
- Keyword Search

Exploring LOD

Common methods

- RDF browsers
 - Manual searching
 - Limited exploration usability
- SPARQL Queries
 - + Efficient
 - Knowledge of SPARQL syntax
 - Knowledge of the RDF schema used to model the data
- Keyword Search
 - + Intuitive
 - Provide answers directly from the RDF data graph

Exploring LOD

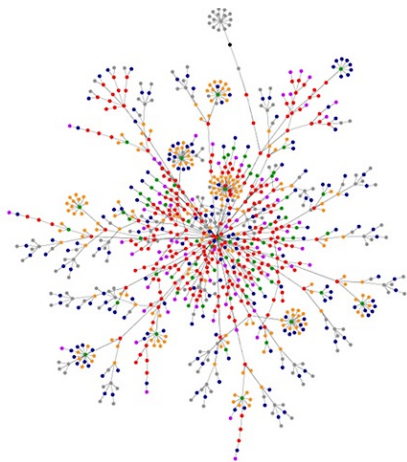
Common methods

- RDF browsers
 - Manual searching
 - Limited exploration usability
- SPARQL Queries
 - + Efficient
 - Knowledge of SPARQL syntax
 - Knowledge of the RDF schema used to model the data
- Keyword Search
 - + Intuitive
 - Provide answers directly from the RDF data graph

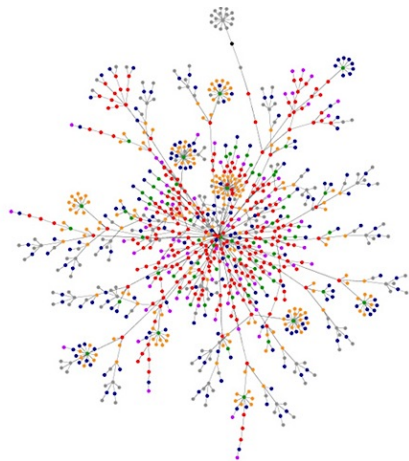
Our Approach

Keyword search that generates automatically a set of **candidate SPARQL queries**, specially designed for evolving linked data.

Indices



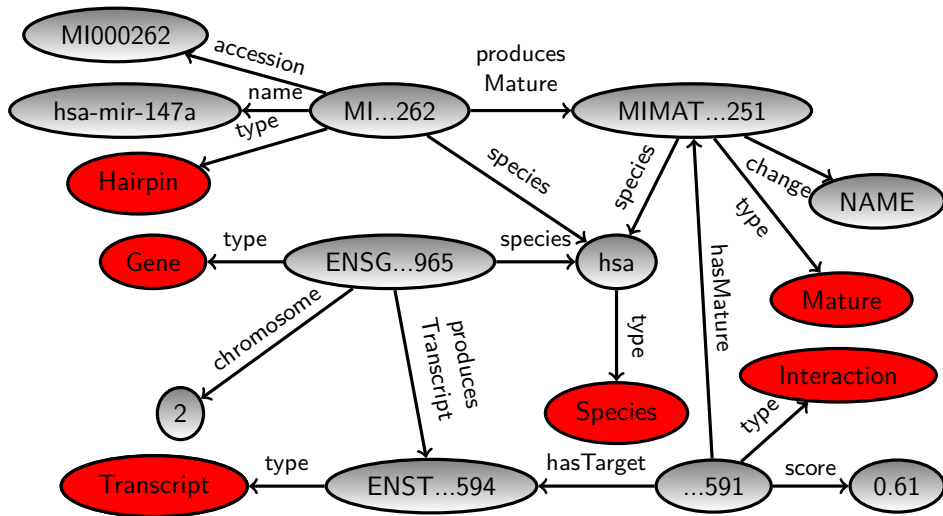
Indices



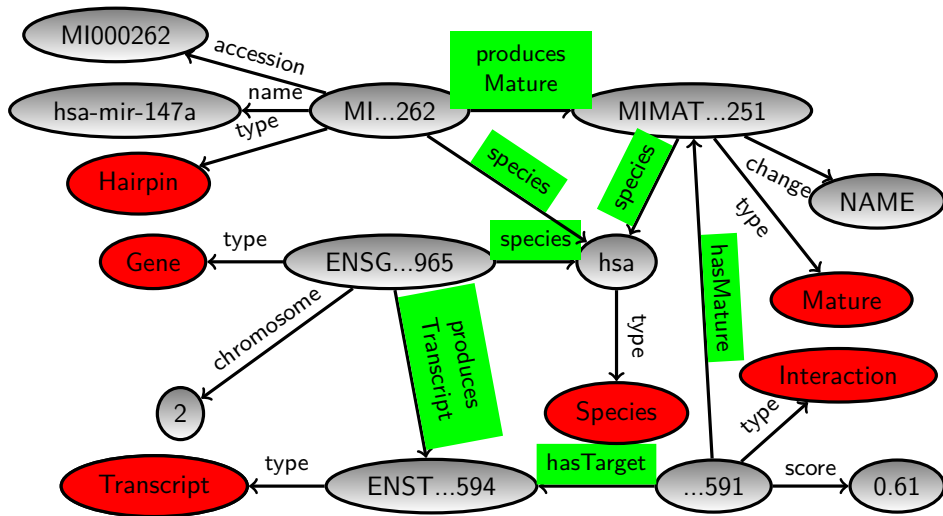
Indices

- **Schema-Guide Graph**
An aggregated representation of the RDF data graph, used as guide for the query computation process.
- **Term Index**
An index of all RDF elements of the RDF data graph, used to match keywords to RDF elements.

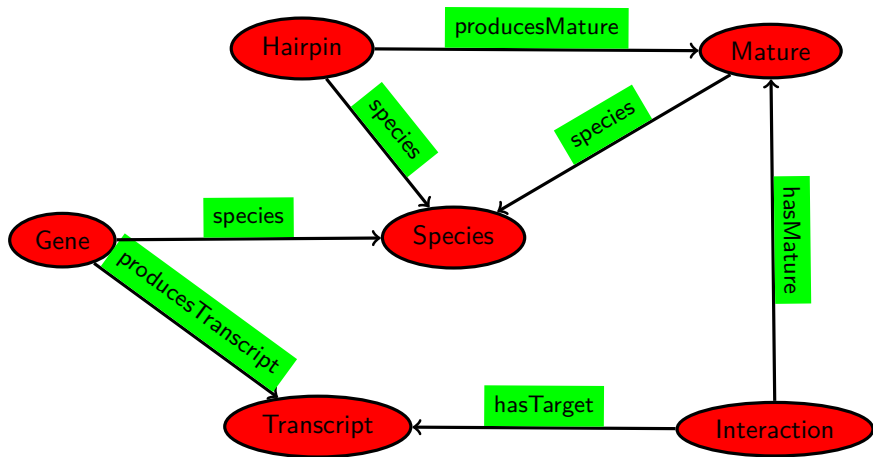
Schema-Guide Graph



Schema-Guide Graph



Schema-Guide Graph



Term Index

- Literal Value
- Entity-to-Attribute Property
- Inter-Entities Property
- Class Name

Term Index

- Literal Value



- Entity-to-Attribute Property

- Inter-Entities Property

- Class Name

Term Index

- Literal Value



- Entity-to-Attribute Property

- Inter-Entities Property

- Class Name

Term Index

- Literal Value



- Entity-to-Attribute Property



- Inter-Entities Property

- Class Name

Term Index

- Literal Value



- Entity-to-Attribute Property



- Inter-Entities Property

- Class Name

Term Index

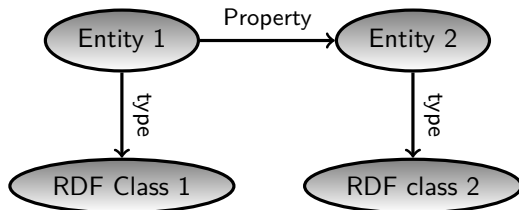
- Literal Value



- Entity-to-Attribute Property



- Inter-Entities Property



- Class Name

Term Index

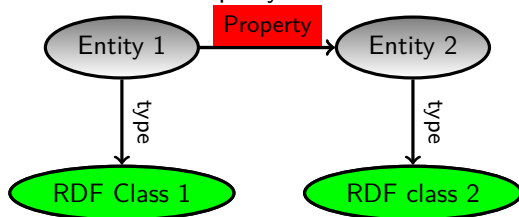
- Literal Value



- Entity-to-Attribute Property



- Inter-Entities Property



- Class Name

Term Index

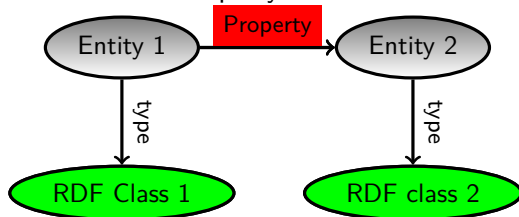
- Literal Value



- Entity-to-Attribute Property



- Inter-Entities Property



- Class Name



Term Index

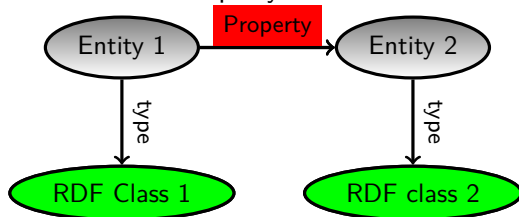
- Literal Value



- Entity-to-Attribute Property



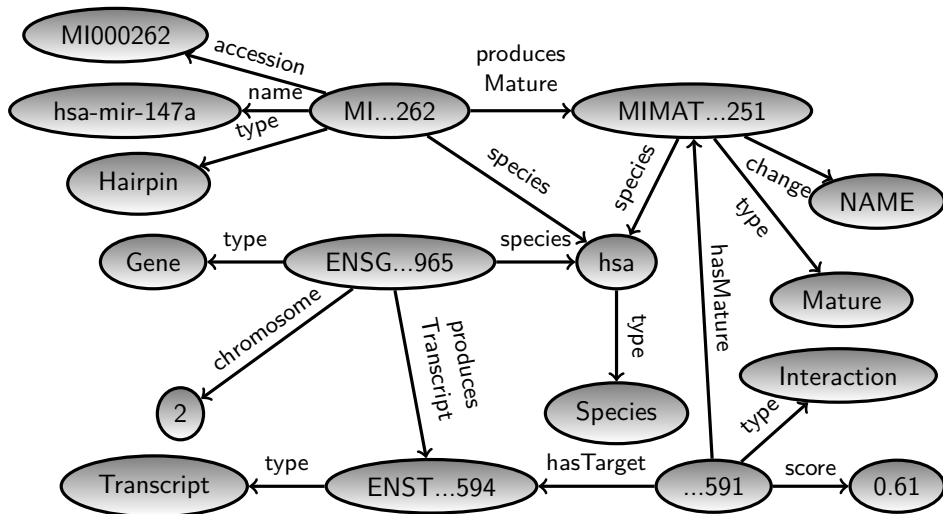
- Inter-Entities Property



- Class Name

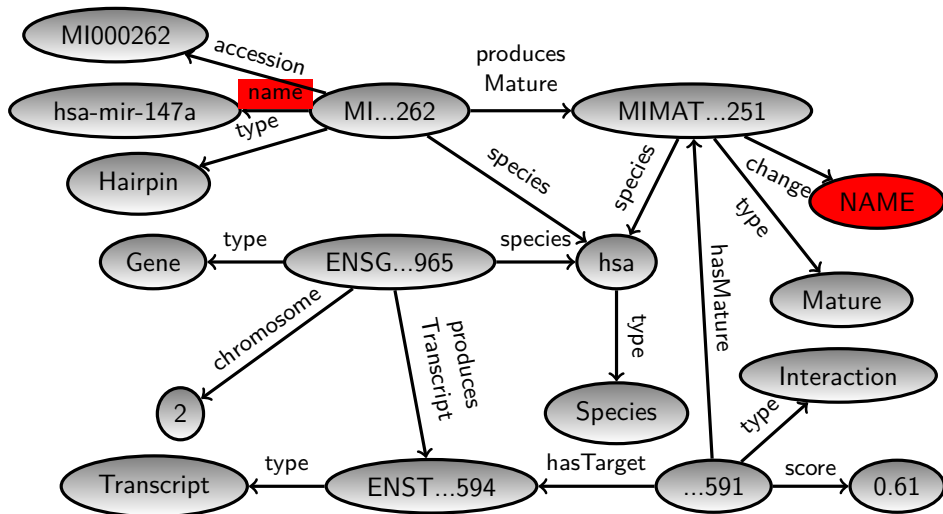


Matching Keywords to RDF Elements



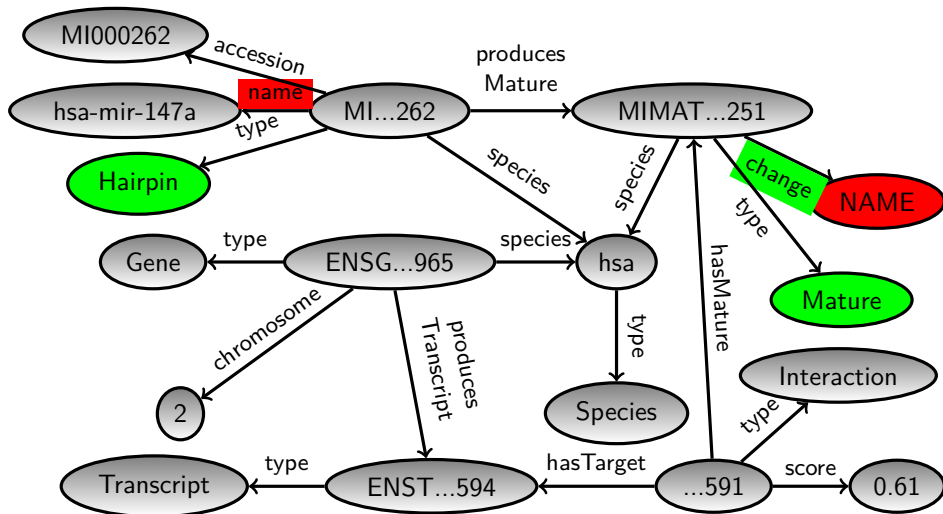
Keywords: name, target, MI000262

Matching Keywords to RDF Elements



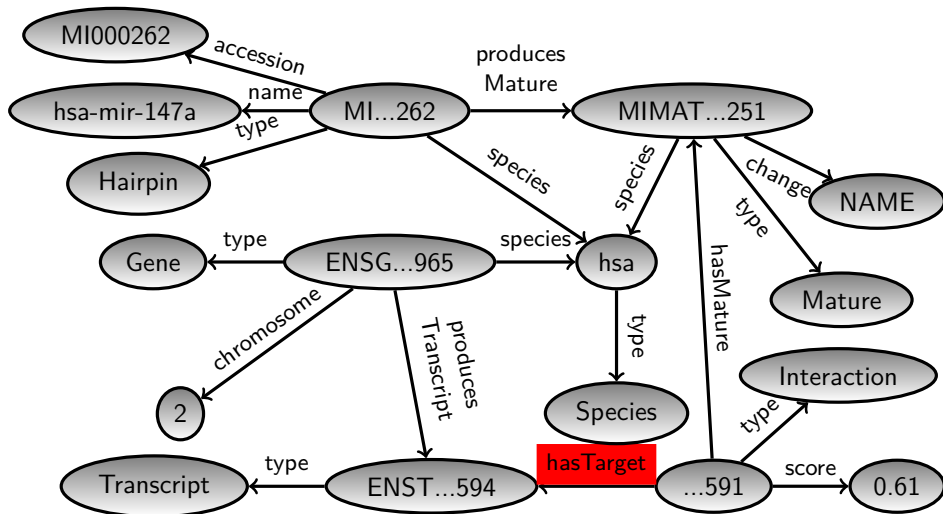
Keywords: **name**, target, MI000262 - 2

Matching Keywords to RDF Elements



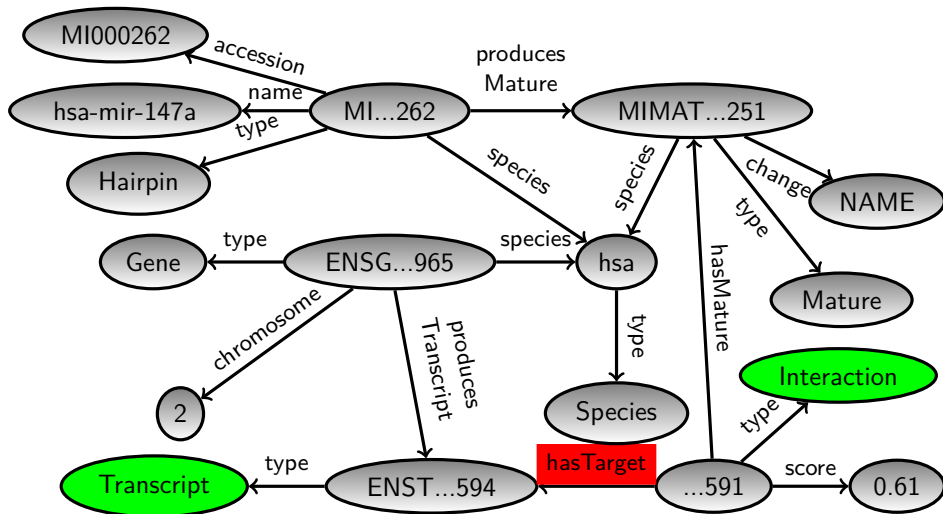
Keywords: **name**, **target**, **MI000262 - 2**

Matching Keywords to RDF Elements



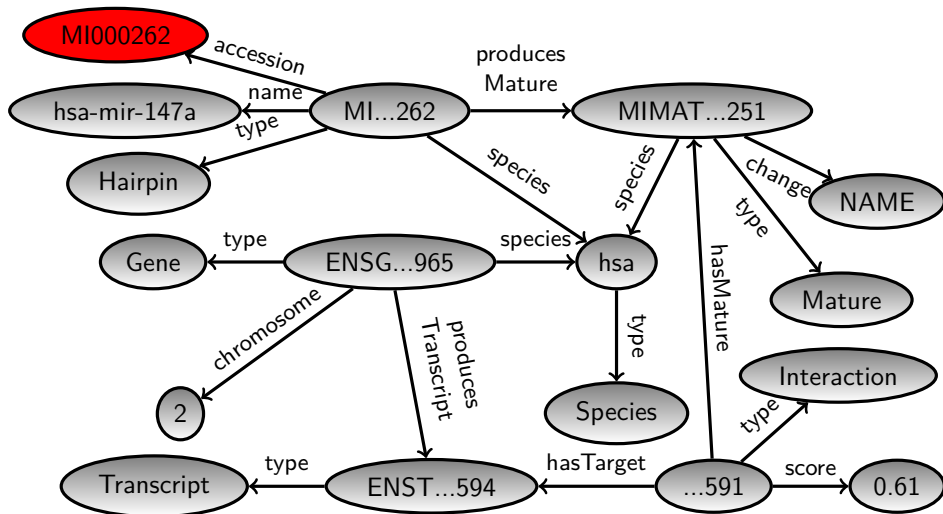
Keywords: name, **target**, MI000262 - 2 × 1

Matching Keywords to RDF Elements



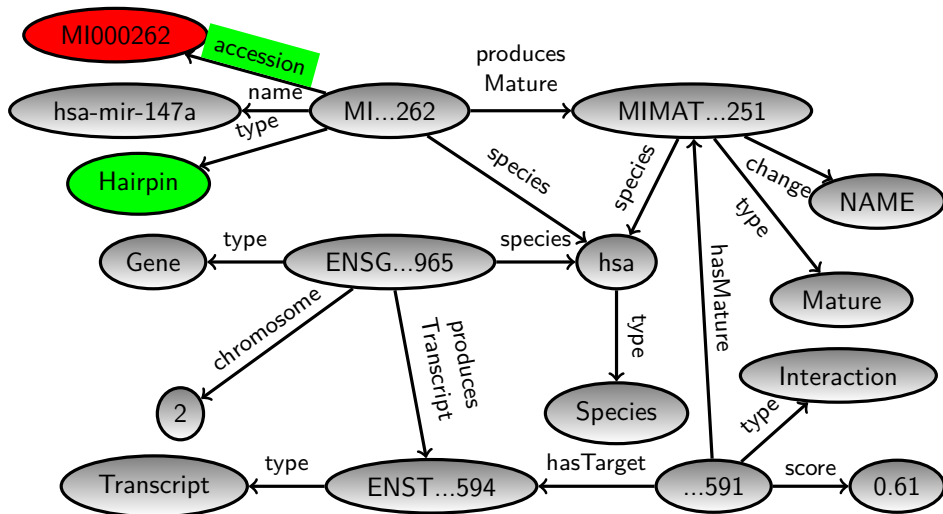
Keywords: name, **target**, MI000262 - 2×1

Matching Keywords to RDF Elements



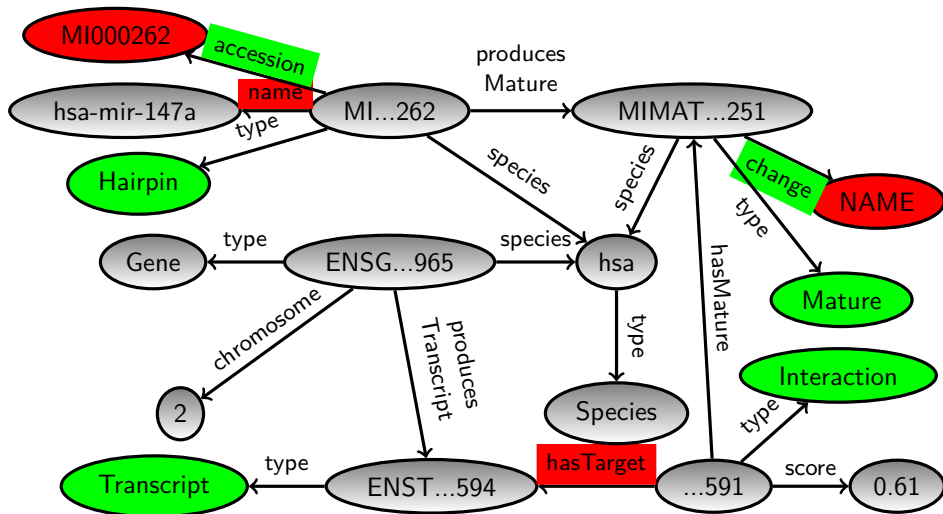
Keywords: name, target, **MI000262** - $2 \times 1 \times 1$

Matching Keywords to RDF Elements



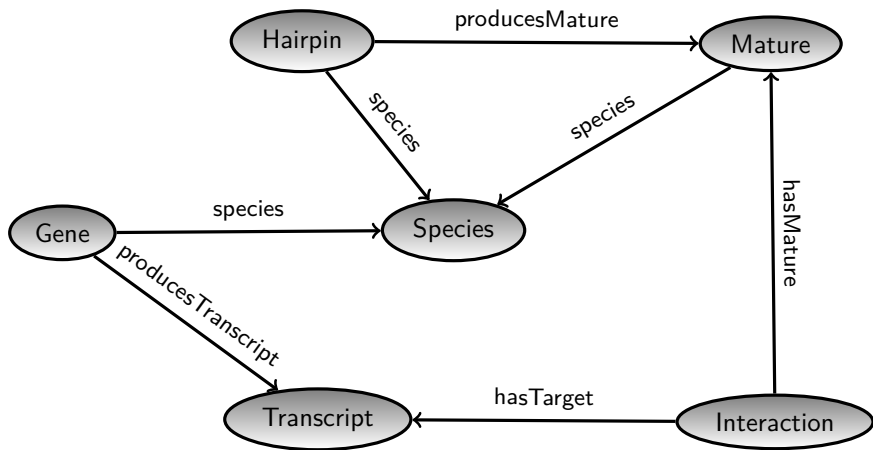
Keywords: name, target, **MI000262** - $2 \times 1 \times 1$

Matching Keywords to RDF Elements



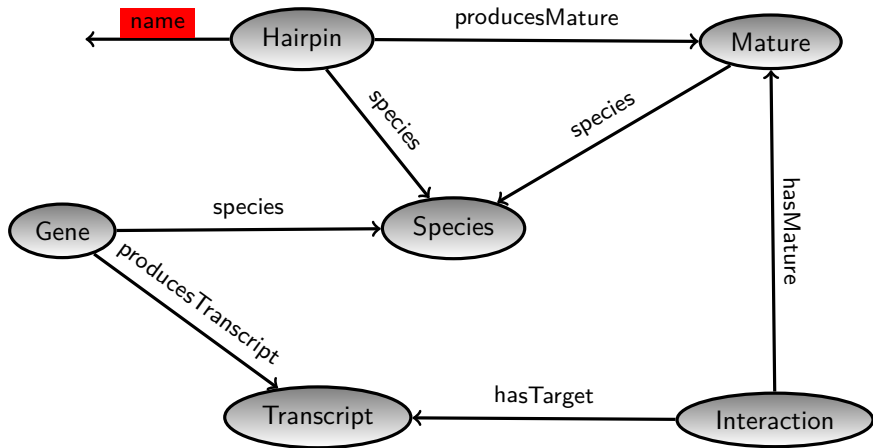
Keywords: **name, target, MI000262** - $2 \times 1 \times 1 = 2$

Augmented Schema-Guide Graph



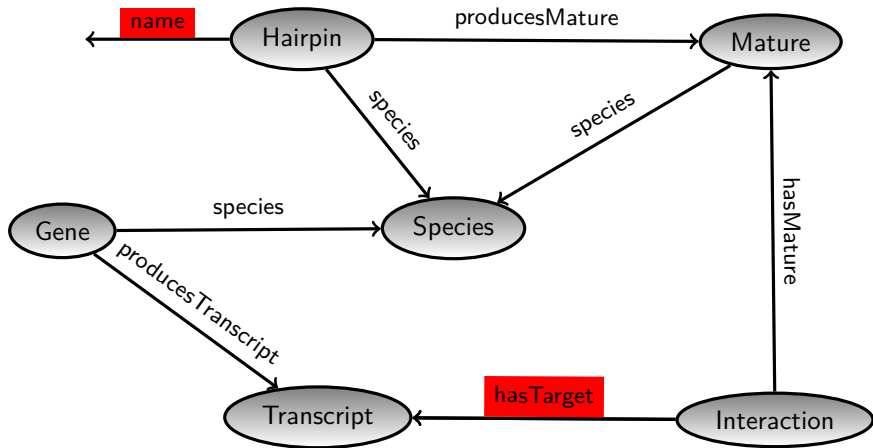
Keywords: name, target, MI000262

Augmented Schema-Guide Graph



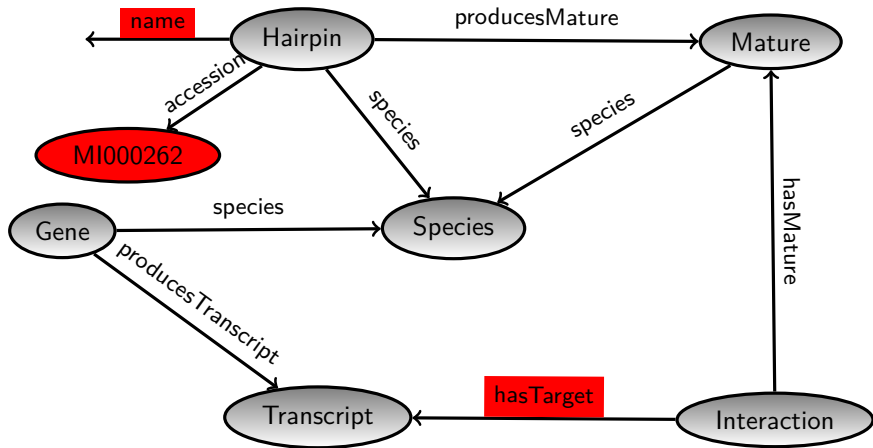
Keywords: **name**, target, MI000262

Augmented Schema-Guide Graph



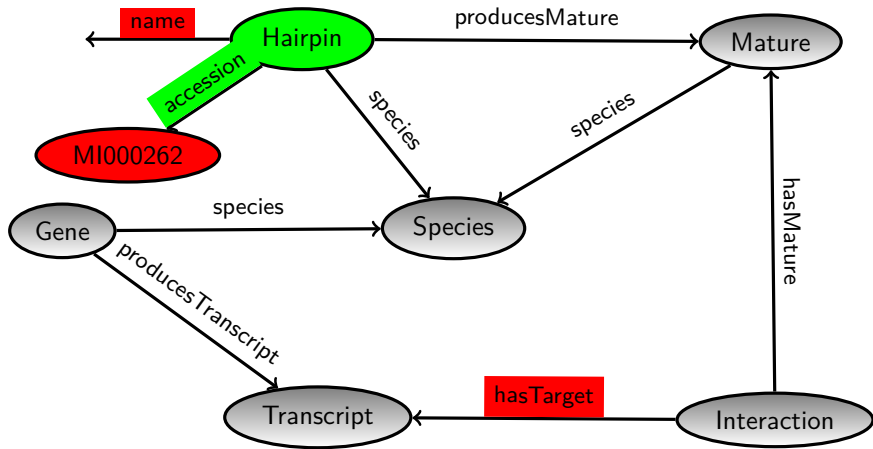
Keywords: name, target, MI000262

Augmented Schema-Guide Graph



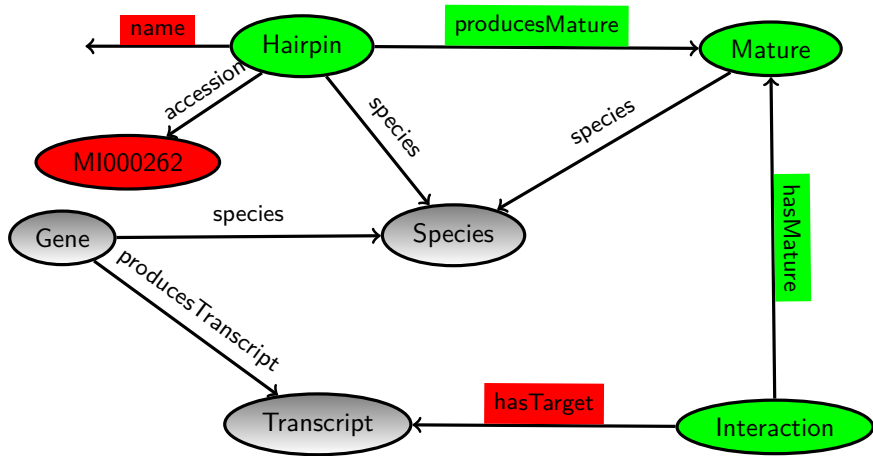
Keywords: name, target, **MI000262**

Augmented Schema-Guide Graph



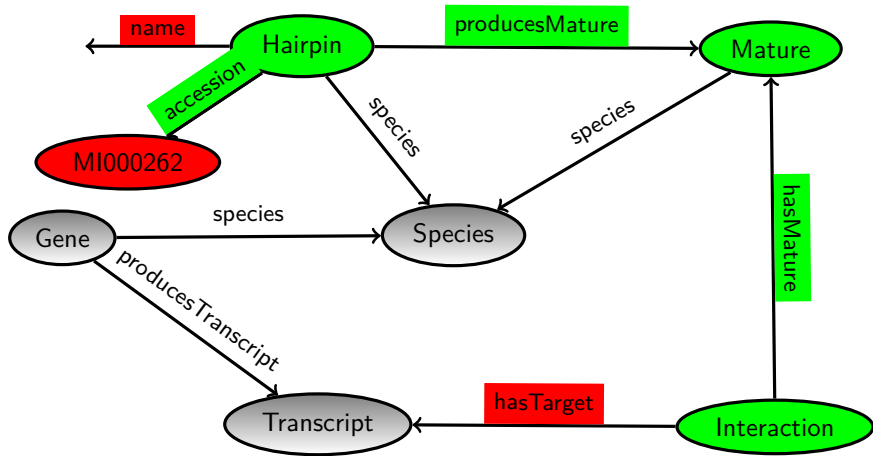
Keywords: name, target, MI000262

Augmented Schema-Guide Graph



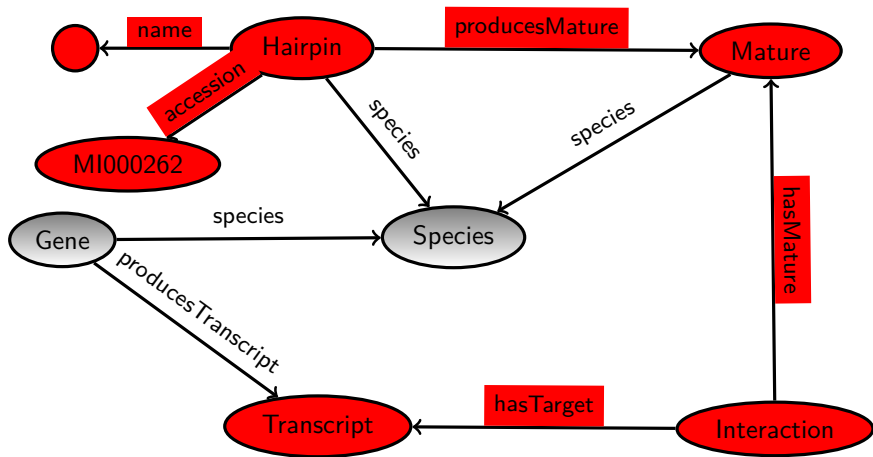
Keywords: name, target, MI000262

Augmented Schema-Guide Graph



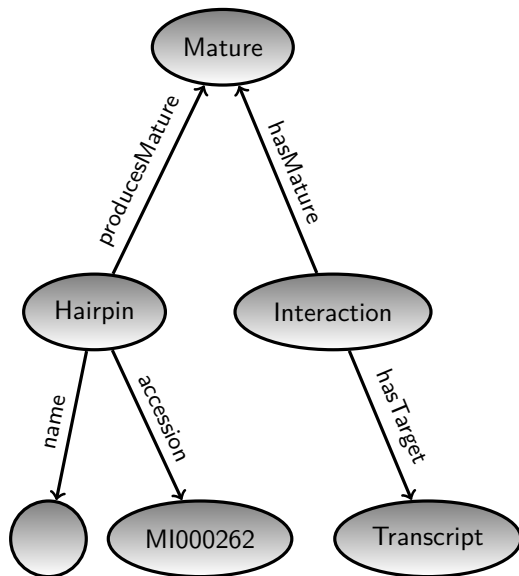
Keywords: name, target, MI000262

Augmented Schema-Guide Graph

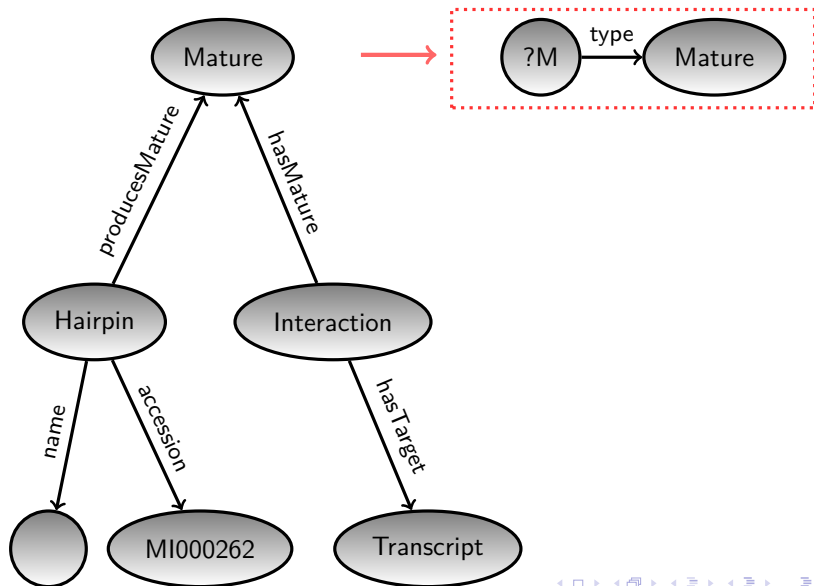


Keywords: name, target, MI000262

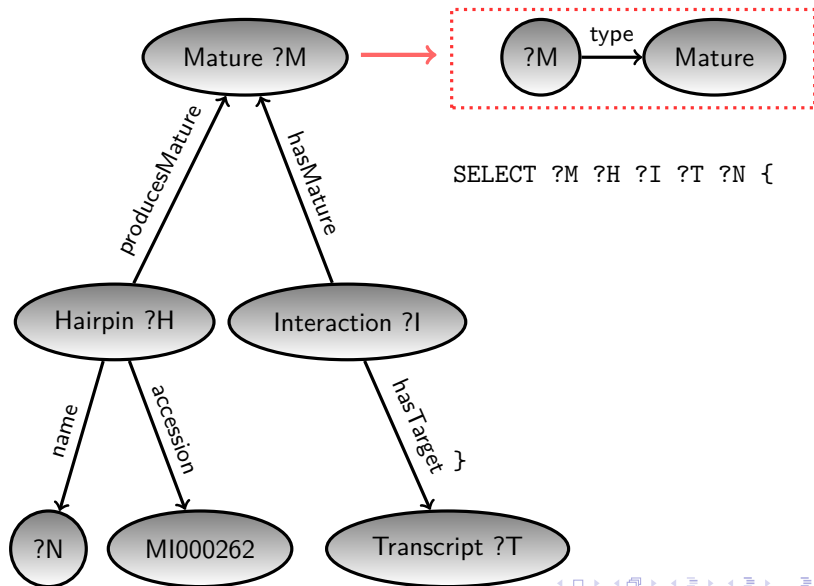
Query Pattern Graph to SPARQL



Query Pattern Graph to SPARQL

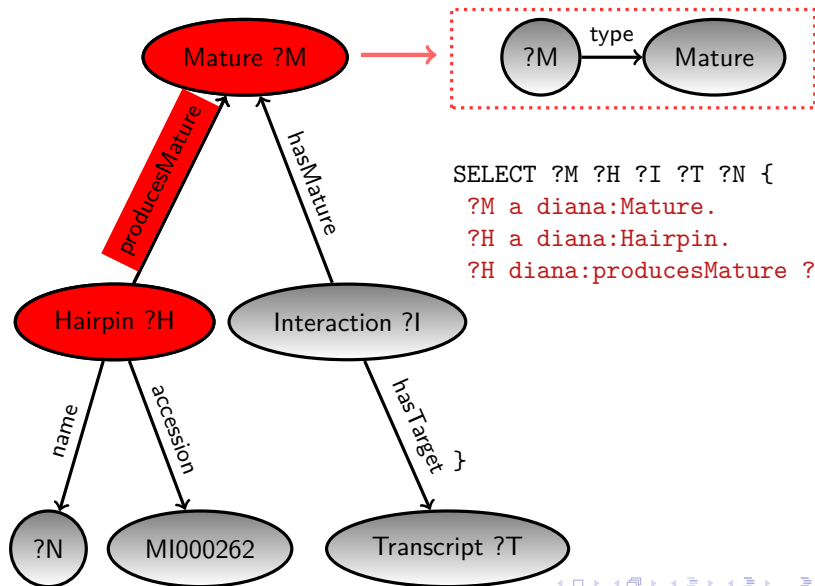


Query Pattern Graph to SPARQL



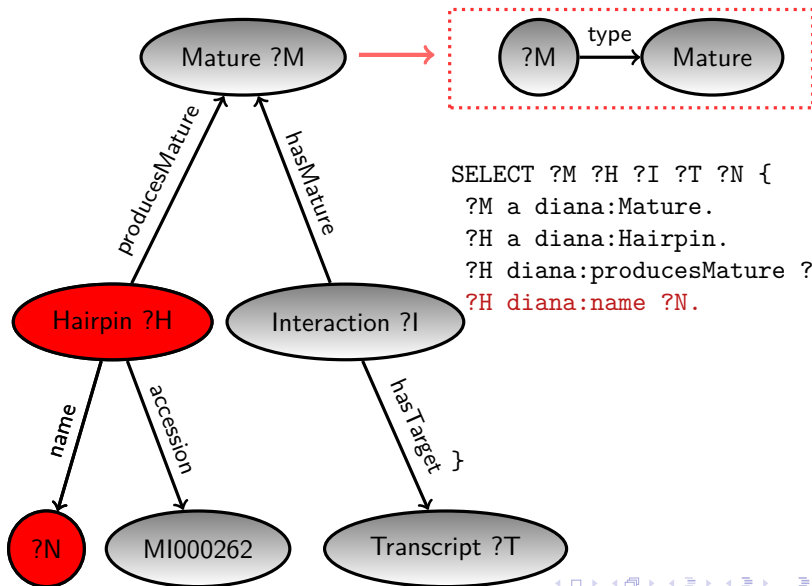
```
SELECT ?M ?H ?I ?T ?N {
```

Query Pattern Graph to SPARQL



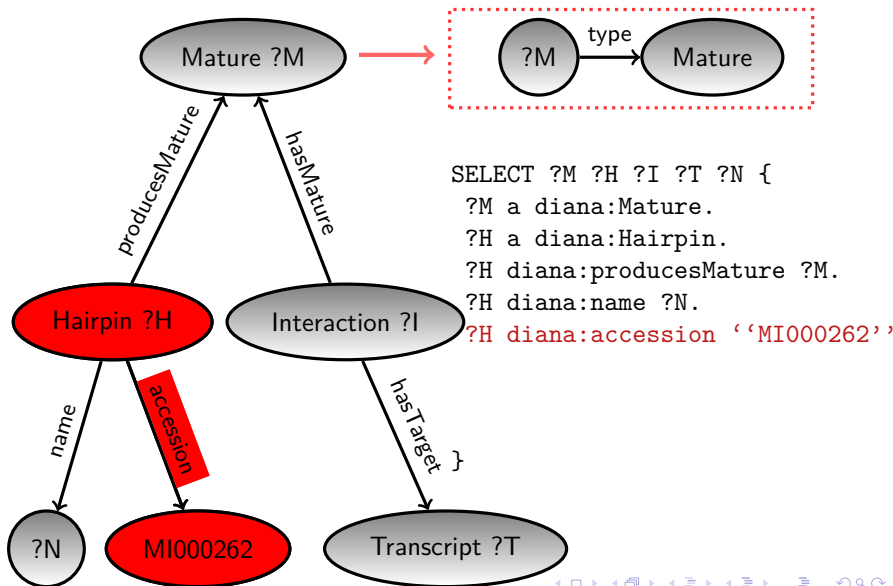
```
SELECT ?M ?H ?I ?T ?N {
  ?M a diana:Mature.
  ?H a diana:Hairpin.
  ?H diana:producesMature ?M.
}
```

Query Pattern Graph to SPARQL

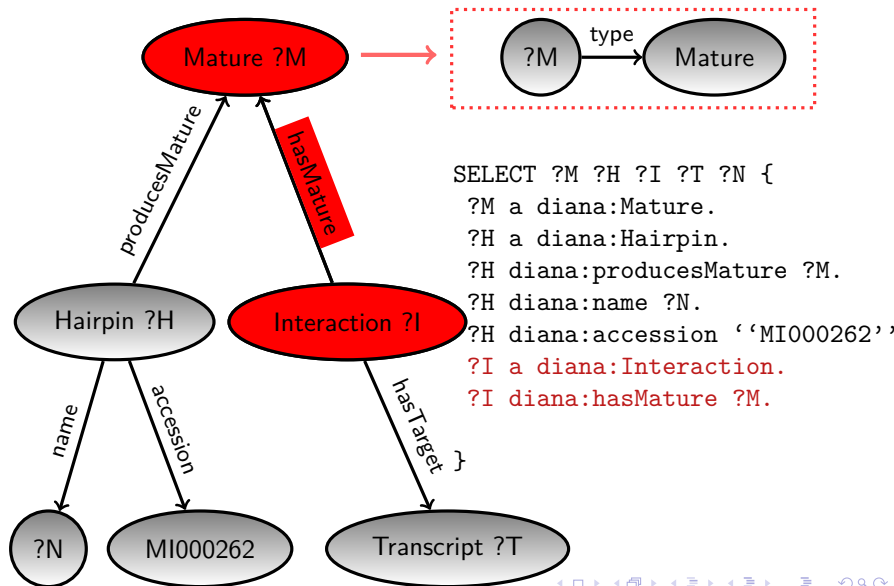


```
SELECT ?M ?H ?I ?T ?N {
  ?M a diana:Mature.
  ?H a diana:Hairpin.
  ?H diana:producesMature ?M.
  ?H diana:name ?N.
```

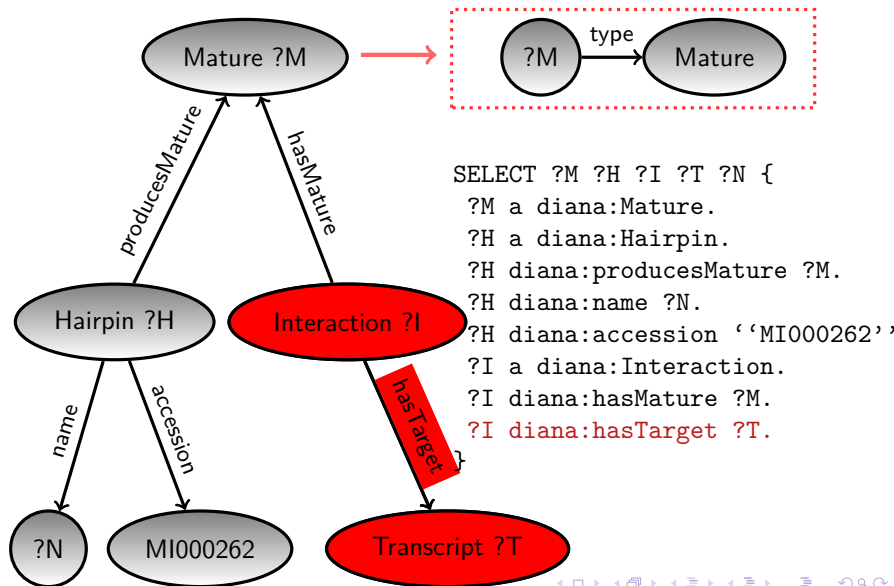
Query Pattern Graph to SPARQL



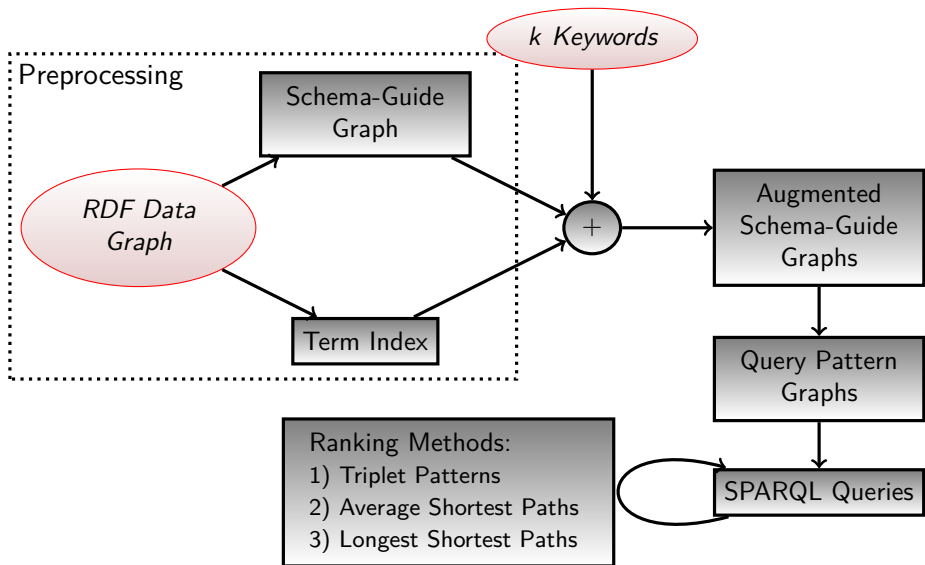
Query Pattern Graph to SPARQL



Query Pattern Graph to SPARQL



Keyword-To-SPARQL Method: Overview



Temporal Operators

Problem

Exploring data that evolve over time requires having access to specified time points or time ranges, i.e. temporal constraints.

Temporal Operators

Problem

Exploring data that evolve over time requires having access to specified time points or time ranges, i.e. temporal constraints.

Solution

Define three temporal operators with syntax `property operator:value`

- `at`
- `before`
- `after`

Temporal Operators

Problem

Exploring data that evolve over time requires having access to specified time points or time ranges, i.e. temporal constraints.

Solution

Define three temporal operators with syntax `property operator:value`

- `at`
`?entity property ?v.`
`FILTER(?v = value)`
- `before`
- `after`

Temporal Operators

Problem

Exploring data that evolve over time requires having access to specified time points or time ranges, i.e. temporal constraints.

Solution

Define three temporal operators with syntax `property operator:value`

- `at`
`?entity property ?v.`
`FILTER(?v = value)`
- `before`
`?entity property ?v.`
`FILTER(?v <= value)`
- `after`

Temporal Operators

Problem

Exploring data that evolve over time requires having access to specified time points or time ranges, i.e. temporal constraints.

Solution

Define three temporal operators with syntax `property operator:value`

- `at`
`?entity property ?v.`
`FILTER(?v = value)`
- `before`
`?entity property ?v.`
`FILTER(?v <= value)`
- `after`
`?entity property ?v.`
`FILTER(?v >= value)`

Evaluation Setting

Datasets

Dataset	# Triples	# Classes	# Properties	# Unique String Values
AI4B	$2,7 \times 10^6$	15	148	6.350
DIANA	$4,6 \times 10^9$	16	76	613.408

Evaluation Setting

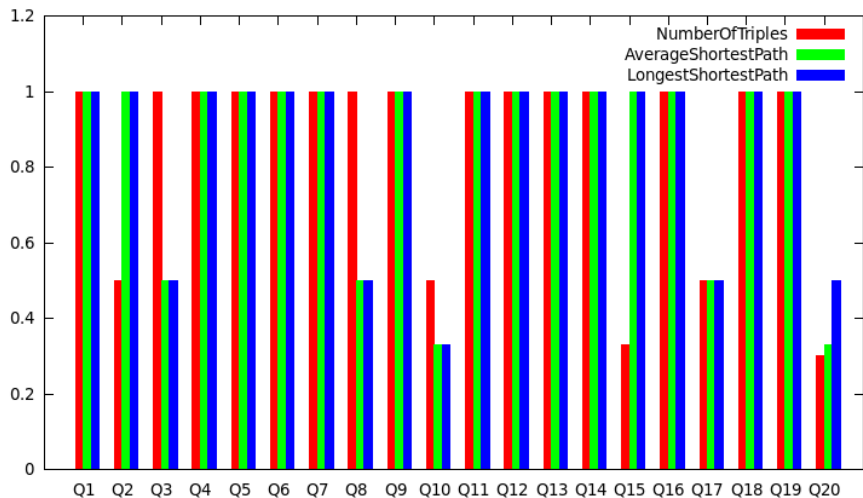
Datasets

Dataset	# Triples	# Classes	# Properties	# Unique String Values
AI4B	$2,7 \times 10^6$	15	148	6.350
DIANA	$4,6 \times 10^9$	16	76	613.408

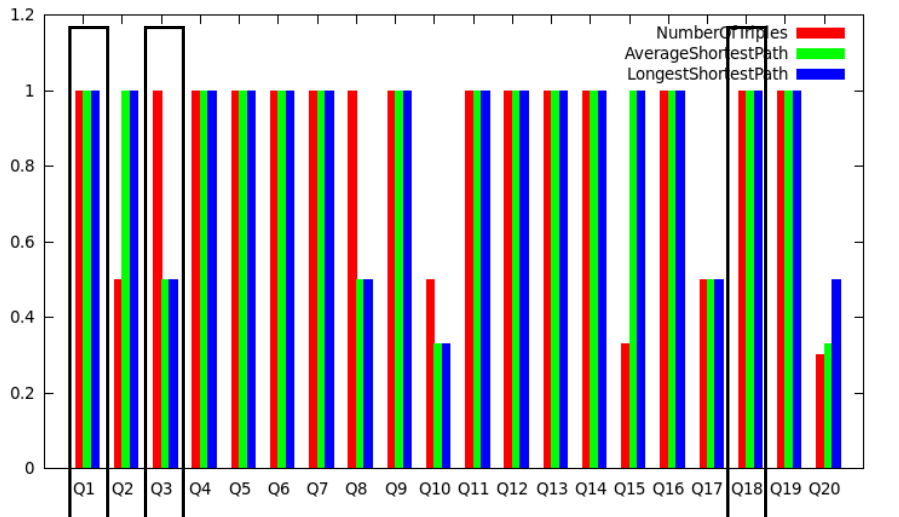
Effectiveness Study

- Pair of keyword query and a natural language (NL) description of the required information.
- 15 questions for the DIANA dataset (Q1-Q15)
- 5 for the AI4B dataset (Q16-Q20).
- Reciprocal Rank metric $RR = 1/r$, where r is the ranking position of the query that corresponds to the provided NL description.
- Use SPARQL2NL [Ngonga Ngomo 13] to provide NL description of generated SPARQL queries.

Quantitative Results



Quantitative Results



Qualitative Results - Q1

Keyword Query

hairpin version at:18

Information requested

Retrieve all hairpins of
miRBase version 18

Qualitative Results - Q1

Keyword Query

hairpin version at:18

Information requested

Retrieve all hairpins of miRBase version 18

Generated SRARQL Query

```
SELECT DISTINCT * WHERE {
  ?H diana:version ?v.
  ?H a diana:Hairpin.
  FILTER (str(?v) = "18")
}
```

Generated query (NL)

This query retrieves distinct values ?v and distinct hairpin miRNAs ?H such that ?v is ?H's miRBase version and the string of ?v is equal to "18".

Qualitative Results - Q18

Keyword Query

hasPrice electricity

Information requested

Retrieve all prices for the
product of electricity

Qualitative Results - Q18

Keyword Query

hasPrice electricity

Information requested

Retrieve all prices for the product of electricity

Generated SPARQL query

```
SELECT DISTINCT * WHERE {
  ?U sym:isCompany ?C.
  ?C sym:hasBusinessName ?v1.
  ?C a sym:Company.
  ?B a sym:BiomassOffer.
  ?B sym:hasPrice ?v2.
  ?U a sym:User.
  ?U sym:offersBiomass ?B.
  FILTER (str(?v1) = "ELECTRICITY").
}
```

Generated NL

This query retrieves distinct Biomass Offers ?B, distinct Users ?U and distinct Companies ?C distinct values ?v1, distinct values ?v2, such that ?U is company ?C, ?B has price ?v2, ?U's offers biomass ?B, ?C has business name ?v1 and the string of ?v1 is equal to "ELECTRICITY".

Qualitative Results - Q3

Ranking with Average Shortest Paths Distance

Keyword Query

hairpin change sequence

Information requested

Retrieve all hairpins that change their sequence at their lifetime

Qualitative Results - Q3

Ranking with Average Shortest Paths Distance

Keyword Query

hairpin change sequence

Information requested

Retrieve all hairpins that change their sequence at their lifetime

Generated SRARQL Query on 1st position

```
SELECT DISTINCT * WHERE {
  ?H diana:change ?v2.
  ?H a diana:Hairpin.
  ?H diana:sequence ?v1.
}
```

Generated NL query on 1st position

This query retrieves distinct values ?v1, distinct values ?v2 and distinct hairpin miRNAs ?H such that ?H's sequence is ?v1 and ?H's change is ?v2.

Qualitative Results - Q3

Ranking with Average Shortest Paths Distance

Keyword Query

hairpin change sequence

Information requested

Retrieve all hairpins that change their sequence at their lifetime

Generated SRARQL Query on 2st position

```
SELECT DISTINCT * WHERE {
  ?H a diana:Hairpin.
  ?H diana:change ?v1.
  FILTER (str(?v1) = "SEQUENCE").
}
```

Generated NL query on 2nd position

his query retrieves distinct values ?v1 and distinct hairpin miRNAs ?H such that ?H change is ?v1 and the string of ?v1 is equal to "SEQUENCE".

Conclusions

Keyword-To-SPARQL search

- A novel approach of RDF data exploration
- Ranked list of SPARQL queries
- Natural Language description of SPARQL query
- Intuitive way of search
- Efficient
- Real-time response
- Temporal Operators for evolving data exploration
- Evaluated on two Linked Open Datasets
- Example for DIANA LOD available at
<http://snf-624527.vm.oceanos.grnet.gr:8080/KeywordSearchDiana/web/index.jsp>.

Future Work - Acknowledgments

Future Work

- Explore pruning schemes
- Explore more complicated ranking schemes
- Further evaluation (scalability, etc)

Acknowledgments

Work has been supported by LODGOV project, Research Programme ARISTEIA (EXCELLENCE), General Secretariat for Research and Technology, Ministry of Education, Greece and the European Social Fund.



European Union
European Social Fund



MINISTRY OF EDUCATION & RELIGIOUS AFFAIRS, CULTURE & SPORTS
MANAGING AUTHORITY

Co-financed by Greece and the European Union





Axel-Cyrille Ngonga Ngomo, Lorenz Bühmann, Christina Unger, Jens Lehmann & Daniel Gerber.

Sorry, I Don'T Speak SPARQL: Translating SPARQL Queries into Natural Language.

In WWW, pages 977–988, 2013.