Keyword Search in RDF Databases

Charalampos S. Nikolaou

charnik@di.uoa.gr

Department of Informatics & Telecommunications
University of Athens

MSc Dissertation Presentation
April 15, 2011

Outline

Background

Keyword Search on (Semi-)Structured data

Data model

Graph-based data model

- Schema-aware
- Schema-agnostic (comes with specialized indexes)

Keyword Search on (Semi-)Structured data

Data model

Graph-based data model

- Schema-aware
- Schema-agnostic (comes with specialized indexes)

Exploration Algorithms

- Backward Expansion
- Bidirectional Expansion
- Variants of the above

Keyword Search on (Semi-)Structured data

Data model

Graph-based data model

- Schema-aware
- Schema-agnostic (comes with specialized indexes)

Structure of the Answer

- Trees
- Graphs
- Nodes/Entities
- (Multi-)Relations

Exploration Algorithms

- Backward Expansion
- Bidirectional Expansion
- Variants of the above

Keyword Search on (Semi-)Structured data

Data model

Graph-based data model

- Schema-aware
- Schema-agnostic (comes with specialized indexes)

Structure of the Answer

- Trees
- Graphs
- Nodes/Entities
- (Multi-)Relations

Exploration Algorithms

- Backward Expansion
- Bidirectional Expansion
- Variants of the above

Ranking/Scoring of Answers

- TF/IDF and other more complex measures from IR
- Node/Edge weights
- Page Rank like, Hub/Authority nodes
- Path length
- Number of nodes/edges

Keyword Search on (Semi-)Structured data

Data model

Graph-based data model

- Schema-aware ✓
- Schema-agnostic (comes with specialized indexes)

Exploration Algorithms

- Backward Expansion
- Bidirectional Expansion
- Variants of the above ✓

Structure of the Answer

- Trees
- Graphs
- Nodes/Entities ✓
- (Multi-)Relations

Ranking/Scoring of Answers

- TF/IDF and other more complex measures from IR✓
- Node/Edge weights ✓
- Page Rank like, Hub/Authority nodes ✓
- Path length ✓
- Number of nodes/edges ✓

User Requirements

Requirements

Use Case

Historians interested in the evolution of Biotechnology and Renewable Energy (see Papyrus Project)

Requirements

Use Case

Historians interested in the evolution of Biotechnology and Renewable Energy (see Papyrus Project)

1. Search for information about certain concepts, facts, events

   e.g., stem cells and public opinion

Requirements

Use Case

Historians interested in the evolution of Biotechnology and Renewable Energy (see Papyrus Project)

1. Search for information about certain concepts, facts, events
2. Search may involve temporal restrictions

   e.g., evolution of biotechnology in 20th century

Requirements

Use Case

Historians interested in the evolution of Biotechnology and Renewable Energy (see Papyrus Project)

1. Search for information about certain concepts, facts, events
2. Search may involve temporal restrictions
3. Source of information may contain indefinite temporal information
   e.g., "around 7000 BC biotechnology involved brewing beer, fermenting wine and baking bread with help of yeast"

Our Approach

Data Model

- Extension of the temporal RDF data model with indefinite time intervals for the validity of a triple: $(s, p, o)[i]$
- Validity of a resource r: $(r, subclass, Resource)[i]$

Data Model

- Extension of the temporal RDF data model with indefinite time intervals for the validity of a triple: $(s, p, o)[i]$
- Validity of a resource r: $(r, \mathit{subclass}, \mathit{Resource})[i]$
- Indefinite time intervals (set I): (s_1, s_2, e_1, e_2), where s_1, s_2, e_1, e_2 are natural numbers

Data Model

- Extension of the temporal RDF data model with indefinite time intervals for the validity of a triple: $(s, p, o)[i]$
- Validity of a resource r: $(r, subclass, Resource)[i]$
- Indefinite time intervals (set I): (s_1, s_2, e_1, e_2), where s_1, s_2, e_1, e_2 are natural numbers

- Indefinite time point: (s, e, s, e)

Query Language

- Keyword-based language extended with temporal constraints expressed in a controlled natural language

Query Language

- Keyword-based language extended with temporal constraints expressed in a controlled natural language
- Temporal constraints (set TR):
  - Allen's temporal relations: `before`, `after`, `meets`, `met by`, `overlaps`, `overlapped by`, `starts`, `started by`, `finishes`, `finished by`, `during`, `contains`
  - Other lexical forms (formal/informal) used in speech and writing

Query Language

- Keyword-based language extended with temporal constraints expressed in a controlled natural language
- Temporal constraints (set TR):
  - Allen's temporal relations: `before`, `after`, `meets`, `met by`, `overlaps`, `overlapped by`, `starts`, `started by`, `finishes`, `finished by`, `during`, `contains`
  - Other lexical forms (formal/informal) used in speech and writing
- Time intervals: $[s_1 - s_2, e_3 - e_4]$, with each s_i (e_i) being of the form `yyyy/mm/dd`

Example

Database

Berlin has been a city since some point in the 12th century

```
(ex:city, rdf:type, rdfs:Class).
(ex:city, rdfs:label, "City").
(ex:berlin, foaf:name, "Berlin").
(ex:berlin, rdf:type, ex:city)[1100 - 1190, 2011].
```

Example

Database

Berlin has been a city since some point in the 12th century

```
(ex:city, rdf:type, rdfs:Class).
(ex:city, rdfs:label, "City").
(ex:berlin, foaf:name, "Berlin").
(ex:berlin, rdf:type, ex:city)[1100 - 1190, 2011].
```

Information needs (1)

I would like information about cities during the period 1200 – 1210

```
city during [1200, 1210]
```

Example

Database

Berlin has been a city since some point in the 12th century

```
(ex:city, rdf:type, rdfs:Class).
(ex:city, rdfs:label, "City").
(ex:berlin, foaf:name, "Berlin").
(ex:berlin, rdf:type, ex:city)[1100 - 1190, 2011].
```

Information needs (1)

I would like information about cities during the period 1200 – 1210

```
city during [1200, 1210]
```

Answer

There is such a city named Berlin

Example

Database

Berlin has been a city since some point in the 12th century

```
(ex:city, rdf:type, rdfs:Class).
(ex:city, rdfs:label, "City").
(ex:berlin, foaf:name, "Berlin").
(ex:berlin, rdf:type, ex:city)[1100 - 1190, 2011].
```

Information needs (2)

I would like information about cities during the period 1100 – 1110

```
city during [1100, 1110]
```

Example

Database

Berlin has been a city since some point in the 12th century

```
(ex:city, rdf:type, rdfs:Class).
(ex:city, rdfs:label, "City").
(ex:berlin, foaf:name, "Berlin").
(ex:berlin, rdf:type, ex:city)[1100 - 1190, 2011].
```

Information needs (2)

I would like information about cities during the period 1100 – 1110

```
city during [1100, 1110]
```

Answer

There is *possibly* a city named Berlin

Example

The ideal answer

The answer to both questions should include the individual "Berlin" and the class "City", but in the second case these two entities should have lower score to convey the notion of possibility

Data Structures

Data graph

The underlying RDF graph

Data Structures

Data graph

The underlying RDF graph

Schema graph

A summary of the data graph, containing *data-driven* schema information

Data Structures

Data graph

The underlying RDF graph

Schema graph

A summary of the data graph, containing *data-driven* schema information

Query graph

Super graph of schema graph containing elements from the RDF graph

Data graph

Subject	Predicate	Object
pro_1	*type*	*Project*
pro_2	*type*	*Project*
pro_1	*name*	*Papyrus*
pub_1	*type*	*Publication*
pub_1	*author*	res_1
pub_1	*author*	res_2
pub_1	*year*	2010
pub_2	*type*	*Publication*
res_1	*type*	*Researcher*
res_2	*type*	*Researcher*
$univ_1$	*name*	*DI&T*
res_2	*name*	*Y. Ioannidis*
res_1	*name*	*M. Koubarakis*
res_1	*worksAt*	$univ_1$
$univ_1$	*type*	*University*
$univ_2$	*type*	*University*
University	*subclass*	*Agent*
Agent	*subclass*	*Resource*
pub_1	*hasProject*	pro_1

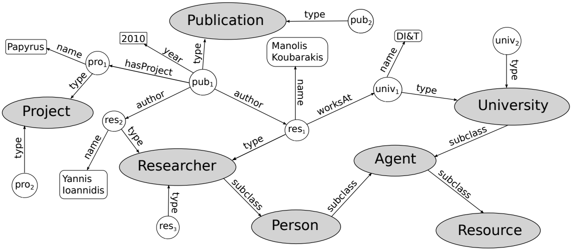


Figure: a) RDF triples b) Data graph

Schema and Query graphs

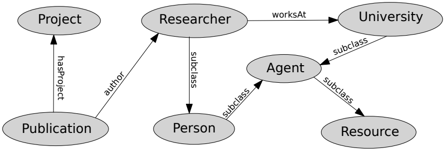


Figure: a) schema graph

Schema and Query graphs

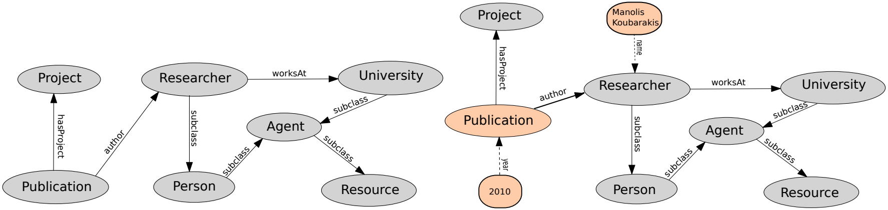


Figure: a) schema graph b) query graph for `koubarakis publications during 2010`

Keyword Search Algorithm

Query processing in 4 phases:

1. *Keyword Interpretation (KI)*: Interpret keywords as RDF graph elements (*keyword elements*) and construct the query graph

Keyword Search Algorithm

Query processing in 4 phases:

1. *Keyword Interpretation (KI)*: Interpret keywords as RDF graph elements (*keyword elements*) and construct the query graph
2. *Graph Exploration (GE)*: Explore the query graph starting from keyword elements for finding top-k subgraphs

Keyword Search Algorithm

Query processing in 4 phases:

1. *Keyword Interpretation (KI)*: Interpret keywords as RDF graph elements (*keyword elements*) and construct the query graph
2. *Graph Exploration (GE)*: Explore the query graph starting from keyword elements for finding top-k subgraphs
3. *Query Mapping (QM)*: Map top-k subgraphs to SPARQL queries and evaluate them

Keyword Search Algorithm

Query processing in 4 phases:

1. *Keyword Interpretation (KI)*: Interpret keywords as RDF graph elements (*keyword elements*) and construct the query graph
2. *Graph Exploration (GE)*: Explore the query graph starting from keyword elements for finding top-k subgraphs
3. *Query Mapping (QM)*: Map top-k subgraphs to SPARQL queries and evaluate them
4. *Entity Transformation (ET)*: Transform and rank entities appearing in subgraphs and results from query evaluation to entities

Keyword Search Algorithm

Query processing in 4 phases:

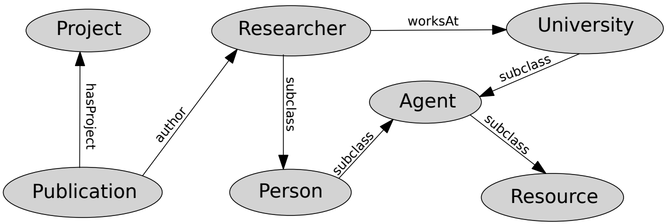
1. *Keyword Interpretation (KI)*: Interpret keywords as RDF graph elements (*keyword elements*) and construct the query graph
2. *Graph Exploration (GE)*: Explore the query graph starting from keyword elements for finding top-k subgraphs
3. *Query Mapping (QM)*: Map top-k subgraphs to SPARQL queries and evaluate them
4. *Entity Transformation (ET)*: Transform and rank entities appearing in subgraphs and results from query evaluation to entities

Running query

`koubarakis publications during 2010`

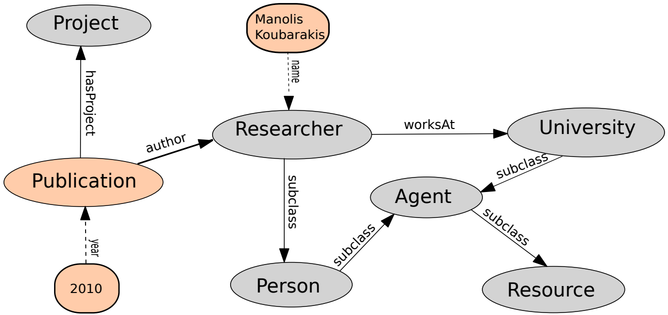
Keyword Interpretation

`koubarakis publications during 2010`



Keyword Interpretation

`koubarakis publications during 2010`

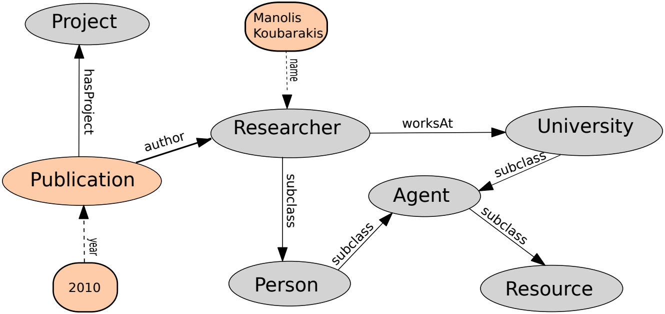


Graph Exploration

`koubarakis publications during 2010`

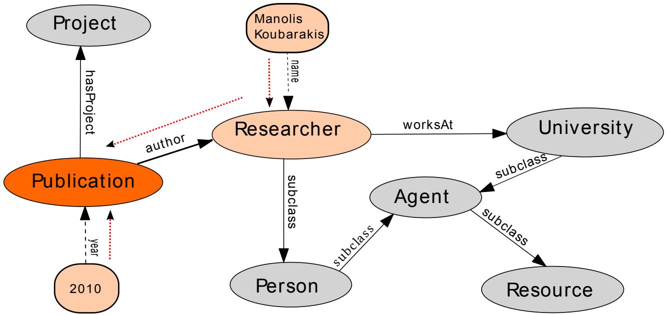
Graph Exploration

`koubarakis publications during 2010`



Graph Exploration

`koubarakis publications during 2010`

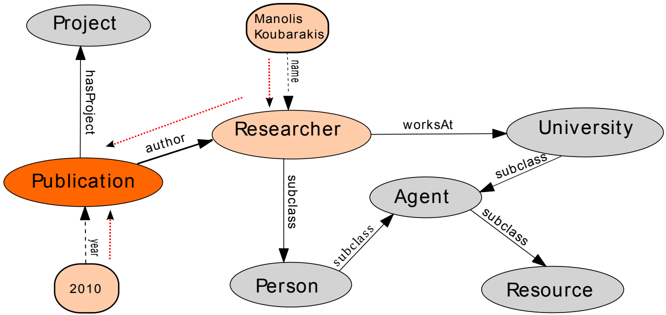


Query Mapping

`koubarakis publications during 2010`

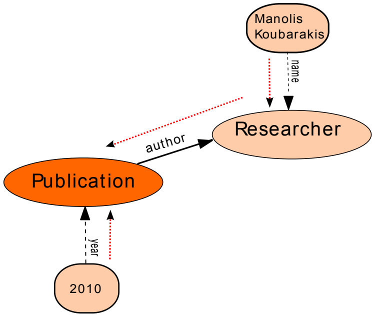
Query Mapping

`koubarakis publications during 2010`



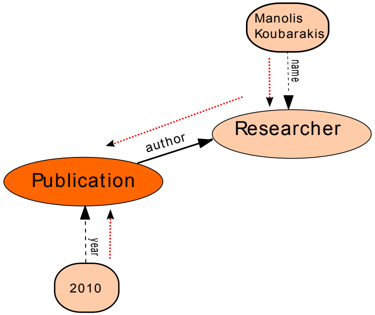
Query Mapping

`koubarakis publications during 2010`



Query Mapping

`koubarakis publications during 2010`



Conjunctive query

name(x,"Koubarakis") ∧ type(x,Researcher) ∧ author(y,x) ∧ year(y,"2010") ∧ type(y,Publication)

Query Mapping

koubarakis publications during 2010

SPARQL Query

```
SELECT ?x ?y ?xl ?yl
WHERE {
     ?x rdf:type ex:Researcher .
     ?x foaf:name "Manolis Koubarakis" .
     ?y ex:author ?x .
     ?x rdf:type ex:Publication .
     ?x ex:year "2010" .
     ?x rdfs:label ?xl .
     ?y rdfs:label ?yl .
}
```

Query Mapping

`koubarakis publications during 2010`

Table: SPARQL result

?x	**?y**	**?xl**	**?yl**
res_1	pub_1	"Manolis Koubarakis"	

Entity Transformation

`koubarakis publications during 2010`

Table: Answer

Rank	Entity
1	"Manolis Koubarakis"
2	Publication
3	pub_1
4	Researcher

Entity Transformation

`koubarakis publications during 2010`

Table: Answer

Rank	Entity
1	"Manolis Koubarakis"
2	Publication
3	pub_1
4	Researcher

How ranking is performed?

Scoring Functions

Scoring of a graph (cont'd)

For any cost function $c_f : V \cup E \to [0,1]$:

$$C_G = \sum_{p_i \in P} \sum_{n \in p_i} c_f(n)$$

Scoring Functions

Scoring of a graph (cont'd)

For any cost function $c_f : V \cup E \rightarrow [0,1]$:

$$C_G = \sum_{p_i \in P} \sum_{n \in p_i} c_f(n)$$

Path length

$$c_f(n) \equiv c_p(n) = 1/diam_G, \text{ for any element n}$$

Scoring Functions

Scoring of a graph (cont'd)

For any cost function $c_f : V \cup E \rightarrow [0,1]$:

$$C_G = \sum_{p_i \in P} \sum_{n \in p_i} c_f(n)$$

Path length

$$c_f(n) \equiv c_p(n) = 1/diam_G, \text{ for any element n}$$

Keyword Matching

$$c_f(n) \equiv c_{kw}(n) = \begin{cases} \varepsilon > 0 & \text{, if } 1 - sim(n) = 0 \\ 1 - sim(n) & \text{, otherwise} \end{cases}$$

Scoring Functions

Scoring of a graph

Popularity

$$c_f(n) \equiv c_{pop}(n) = \begin{cases} 1 - \frac{|n_{agg}|}{|V|} & \text{, if } n \in V_C \\ 1 - \frac{|n_{inc}|}{|V|} & \text{, if } n \in V_E \\ 1 - \frac{|n_{agg}|}{|E|} & \text{, if } n \in L_R \end{cases}$$

Scoring Functions

Scoring of a graph

Popularity

$$c_f(n) \equiv c_{pop}(n) = \begin{cases} 1 - \frac{|n_{agg}|}{|V|} & \text{, if } n \in V_C \\ 1 - \frac{|n_{inc}|}{|V|} & \text{, if } n \in V_E \\ 1 - \frac{|n_{agg}|}{|E|} & \text{, if } n \in L_R \end{cases}$$

Combine

$$c_f(n) \equiv c_{comb}(n) = c_p(n) * c_{kw}(n) * c_{pop}(n)$$

Scoring Functions

Scoring of an entity

Entity Cost

Represents the weighted average of the cost of an entity over the subgraphs it appears

$$Cost(e, S) = \frac{C_{cf}(e) * minSGCost}{|SG_S(e)|} \sum_{SG_i \in SG_S(e)} \frac{1}{C_{cf}(SG_i)}$$

Scoring Functions

Scoring of an entity

Entity Cost

Represents the weighted average of the cost of an entity over the subgraphs it appears

$$Cost(e,S) = \frac{C_{cf}(e) * minSGCost}{|SG_S(e)|} \sum_{SG_i \in SG_S(e)} \frac{1}{C_{cf}(SG_i)}$$

Final Entity Cost

Represents the average cost of an entity derived both during graph exploration and query mapping

$$Cost(e) = \begin{cases} \frac{Cost(e,SGE)+Cost(e,QE)}{2} & \text{, if } e \in SGE \cap QE \\ Cost(e,SGE) & \text{, if } e \in SGE \text{ and } e \notin QE \\ Cost(e,QE) & \text{, if } e \in QE \text{ and } e \notin SGE \end{cases}$$

Implementation

System Architecture

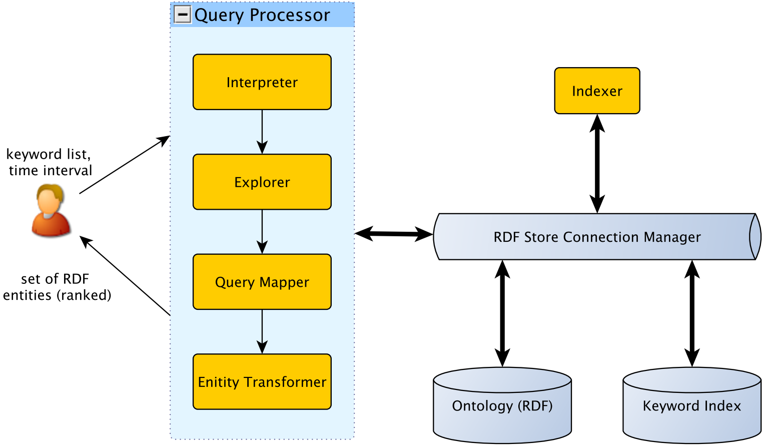


Figure: The architecture of the keyword querying system

System Implementation

RDF Store:

System Implementation

RDF Store: Sesame

System Implementation

RDF Store: [Sesame]

Keyword index and full-text search:

System Implementation

RDF Store: Sesame

Keyword index and full-text search: LuceneSail

System Implementation

RDF Store: Sesame

Keyword index and full-text search: LuceneSail

Query Processor:

System Implementation

RDF Store: Sesame

Keyword index and full-text search: LuceneSail

Query Processor: Java implementation

Experimental Evaluation

Experimental Setup

Machine

- CPU: Intel(R) Core(TM)2 Quad CPU Q9650 @ 3.00 GHz, L2 6144 KB
- Main Memory: 4 GB, 1033 MHz
- Hard Disk: 750 GB, 7200 rpm, 8 MB Buffer

Experimental Setup

Machine

- CPU: Intel(R) Core(TM)2 Quad CPU Q9650 @ 3.00 GHz, L2 6144 KB
- Main Memory: 4 GB, 1033 MHz
- Hard Disk: 750 GB, 7200 rpm, 8 MB Buffer

Datasets

- History Ontology (HO)
- Semantic Web Dog Food Ontology (SWDF)
- Digital Bibliography & Library Project Ontology (DBLP)

Experimental Setup

Machine

- CPU: Intel(R) Core(TM)2 Quad CPU Q9650 @ 3.00 GHz, L2 6144 KB
- Main Memory: 4 GB, 1033 MHz
- Hard Disk: 750 GB, 7200 rpm, 8 MB Buffer

Datasets

- History Ontology (HO)
- Semantic Web Dog Food Ontology (SWDF)
- Digital Bibliography & Library Project Ontology (DBLP)

Evaluated Dimensions

- Efficiency: Load Scalability, Query Answering Performance
- Effectiveness: Precision/Recall, F-measure, NDCG

Dataset Characteristics

	Triples	Classes	Properties	Instances	Avg. Inst./Class
HO	8,327	**367**	**727**	1,405	4
SWDF	88,996	96	369	8,580	89
DBLP	**55,364,046**	12	20	**3,609,294**	300,775

HO: High schema complexity and small number of instances

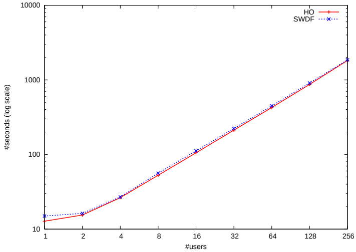
SWDF: Medium schema complexity and number of instances

DBLP: Low schema complexity and high number of instances

Efficiency

Load/Service Scalability

$\#$ users $\equiv$ $\#$ sessions $\equiv$ $\#$ queries



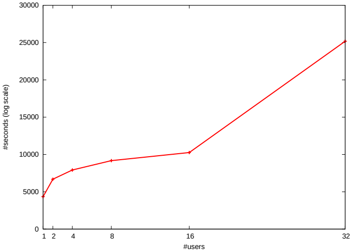
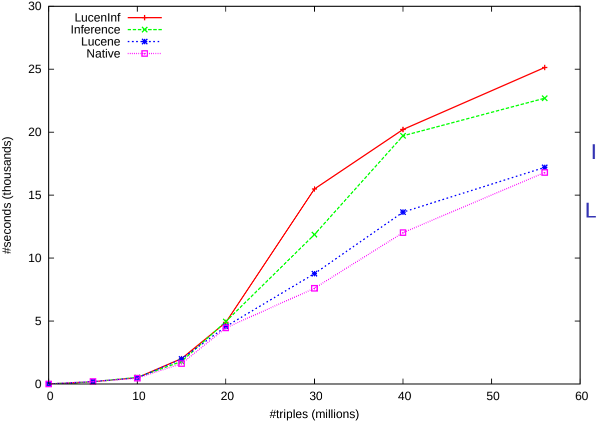


Figure: Load scalability for a) HO/SWDF and b) DBLP

Efficiency

Index Performance (cont'd)

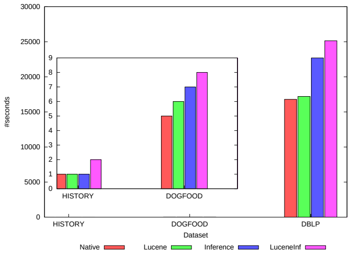


Native	Triple indexing on disk
Lucene	Native+Keyword indexing
Inference	Native+Schema graph inference
LuceneInf	Lucene+Schema graph inference

Figure: Index construction (DBLP)

Efficiency

Index Performance



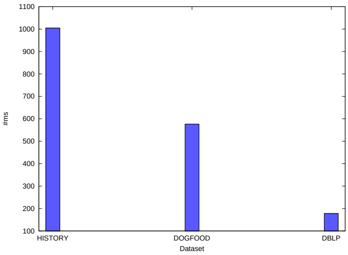


Figure: a) Index construction time, b) Load times of schema graph index

Efficiency

Scoring Function Performance (cont'd)

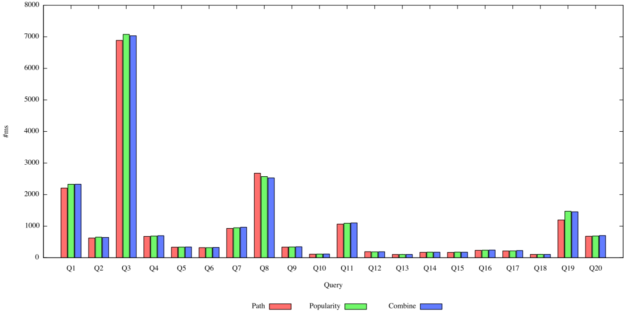


Figure: Tasks performance: HO

Efficiency

Scoring Function Performance (cont'd)

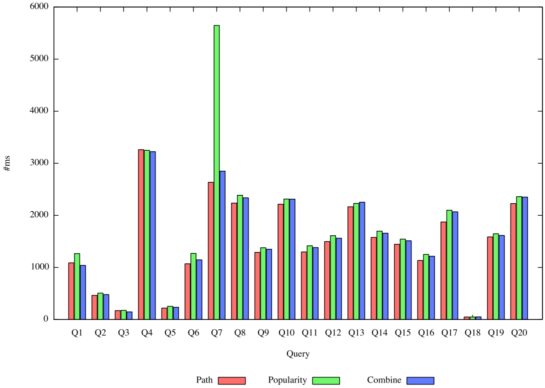


Figure: Tasks performance: SWDF

Efficiency

Scoring Function Performance

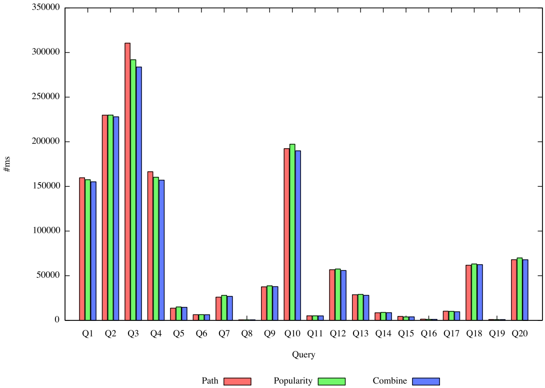


Figure: Tasks performance: DBLP

Conclusion

Scoring functions have similar computational characteristics

Efficiency

Query Performance (cont'd)

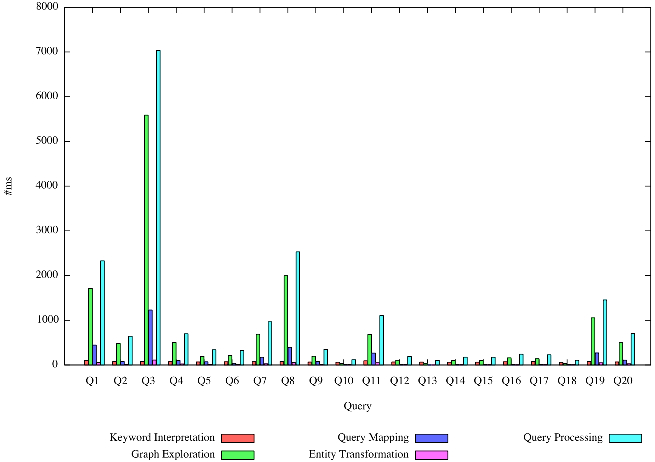


Figure: Tasks performance: HO

Efficiency

Query Performance (cont'd)

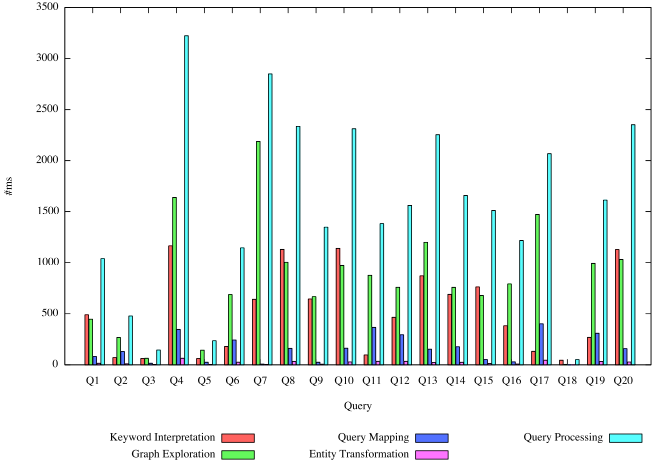


Figure: Tasks performance: SWDF

Efficiency

Query Performance (cont'd)

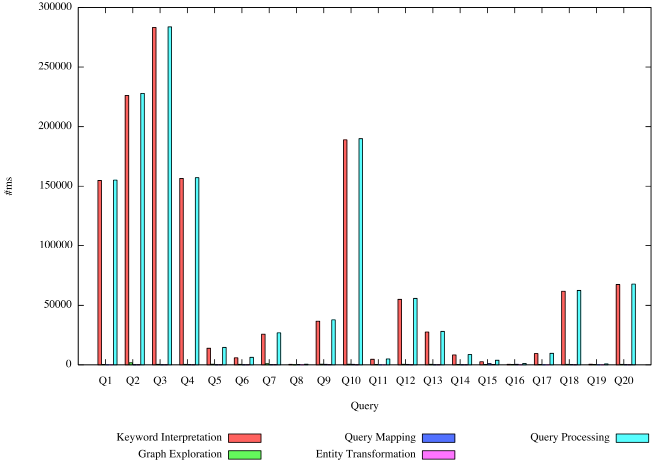


Figure: Tasks performance: DBLP

Efficiency

Query Performance

Table: Tasks performance: Overall contribution

	KI	GE	QM	ET
HO	0.23	**0.60**	0.13	0.02
SWDF	0.35	0.52	0.11	0.01
DBLP	**0.90**	0.05	0.04	0.01

Efficiency

Query Performance

Table: Tasks performance: Overall contribution

	KI	**GE**	**QM**	**ET**
HO	0.23	**0.60**	0.13	0.02
SWDF	0.35	0.52	0.11	0.01
DBLP	**0.90**	0.05	0.04	0.01

Conclusions

HO: QP dominated by graph exploration

SWDF: QP rather fairly distributed among querying tasks

DBLP: QP dominated by keyword indexing

Effectiveness

Measurement methodology (cont'd)

Precision (or how succinct is the answer)

$$P = \frac{\#(\text{relevant items retrieved})}{\#(\text{retrieved items})}$$

Effectiveness

Measurement methodology (cont'd)

Precision (or how succinct is the answer)

$$P = \frac{\#(\text{relevant items retrieved})}{\#(\text{retrieved items})}$$

Recall (or how much did it cover the question)

$$R = \frac{\#(\text{relevant items retrieved})}{\#(\text{relevant items})}$$

Effectiveness

Measurement methodology (cont'd)

Precision (or how succinct is the answer)

$$P = \frac{\#(\text{relevant items retrieved})}{\#(\text{retrieved items})}$$

Recall (or how much did it cover the question)

$$R = \frac{\#(\text{relevant items retrieved})}{\#(\text{relevant items})}$$

F1-measure (or how much it is to the point)

$$F_1 = \frac{2PR}{P+R}$$

Effectiveness

Measurement methodology

NDCG (or how good was the ranking)

$$NDCG_k = \frac{r_1 + \sum_{i=2}^{k} \frac{r_i}{log_2(i)}}{IDCG_k}, \quad k = 15$$

Effectiveness

Measurement methodology

NDCG (or how good was the ranking)

$$NDCG_k = \frac{r_1 + \sum_{i=2}^{k} \frac{r_i}{log_2(i)}}{IDCG_k}, \quad k = 15$$

Judgement

- 5 history experts
- 20 keyword queries (2-3 keywords each)
- Relevance judgements for relevant entities

Effectiveness

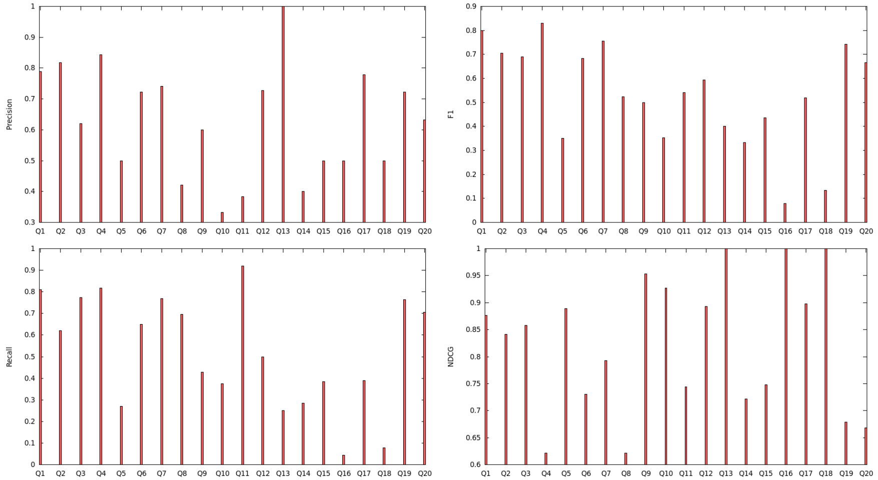


Figure: Effectiveness evaluation results (HO)

Other Directions to Keyword Search

Other Directions to Keyword Search

Browsing

- Rich interfaces for result exploration, discovery of hidden inter-connections, and query refinement
- Zero-effort web publishing

Other Directions to Keyword Search

Result Snippets

Generation of small passage descriptions for quick judgement of results

Other Directions to Keyword Search

Result Clustering

Clustering of results according to different interpretations of the semantics of the query

Other Directions to Keyword Search

Query Cleaning

- Semantic linkage and spelling corrections of database-relevant query keywords
- Segmentation of nearby query keywords so that each segment corresponds to a high quality data term

Conclusions & Future Work

Conclusions

- Design & implementation of a keyword-based system on RDF graphs with indefinite temporal information

Conclusions

- Design & implementation of a keyword-based system on RDF graphs with indefinite temporal information
- Evaluation results: rather scalable

Conclusions

- Design & implementation of a keyword-based system on RDF graphs with indefinite temporal information
- Evaluation results: rather scalable, modest performance

Conclusions

- Design & implementation of a keyword-based system on RDF graphs with indefinite temporal information
- Evaluation results: rather scalable, modest performance, modest effectiveness

Conclusions

- Design & implementation of a keyword-based system on RDF graphs with indefinite temporal information
- Evaluation results: rather scalable, modest performance, modest effectiveness
- Keyword interpretation and graph exploration call for improvement

Future Work

- Sesame/LuceneSail substitution by BigOWLIM

Future Work

- Sesame/LuceneSail substitution by BigOWLIM
- Keyword interpretation and graph exploration improvement

Future Work

- Sesame/LuceneSail substitution by BigOWLIM
- Keyword interpretation and graph exploration improvement
- Challenge: Integration and querying of semi-structured data and linked-data, stored in different formats (RDF, XML) and data sources (ontologies, knowledge bases, databases)

Future Work

- Sesame/LuceneSail substitution by BigOWLIM
- Keyword interpretation and graph exploration improvement
- Challenge: Integration and querying of semi-structured data and linked-data, stored in different formats (RDF, XML) and data sources (ontologies, knowledge bases, databases)
- Schema-agnostic or hybrid data model

This is the End. . .

References

Check my dissertation