

WWW 2017 Tutorial: Semantic Data Management in Practice

Part 4: Searching



Olaf Hartig

Linköping University

 olaf.hartig@liu.se

 @olafhartig

Olivier Curé

University of Paris-Est Marne la Vallée

 olivier.cure@u-pem.fr

 @oliviercure

Goal

- Query a dataset by using keywords
 - Full-text search
- Typical use cases are related to datasets that contain literals with (large) texts



Options

- Use a full-text search engine
(as a separate component in the software stack)
- Use full-text search features built into triple stores
 - Native full-text search functionality
 - Integration of external search engines

What is Full-Text Search?

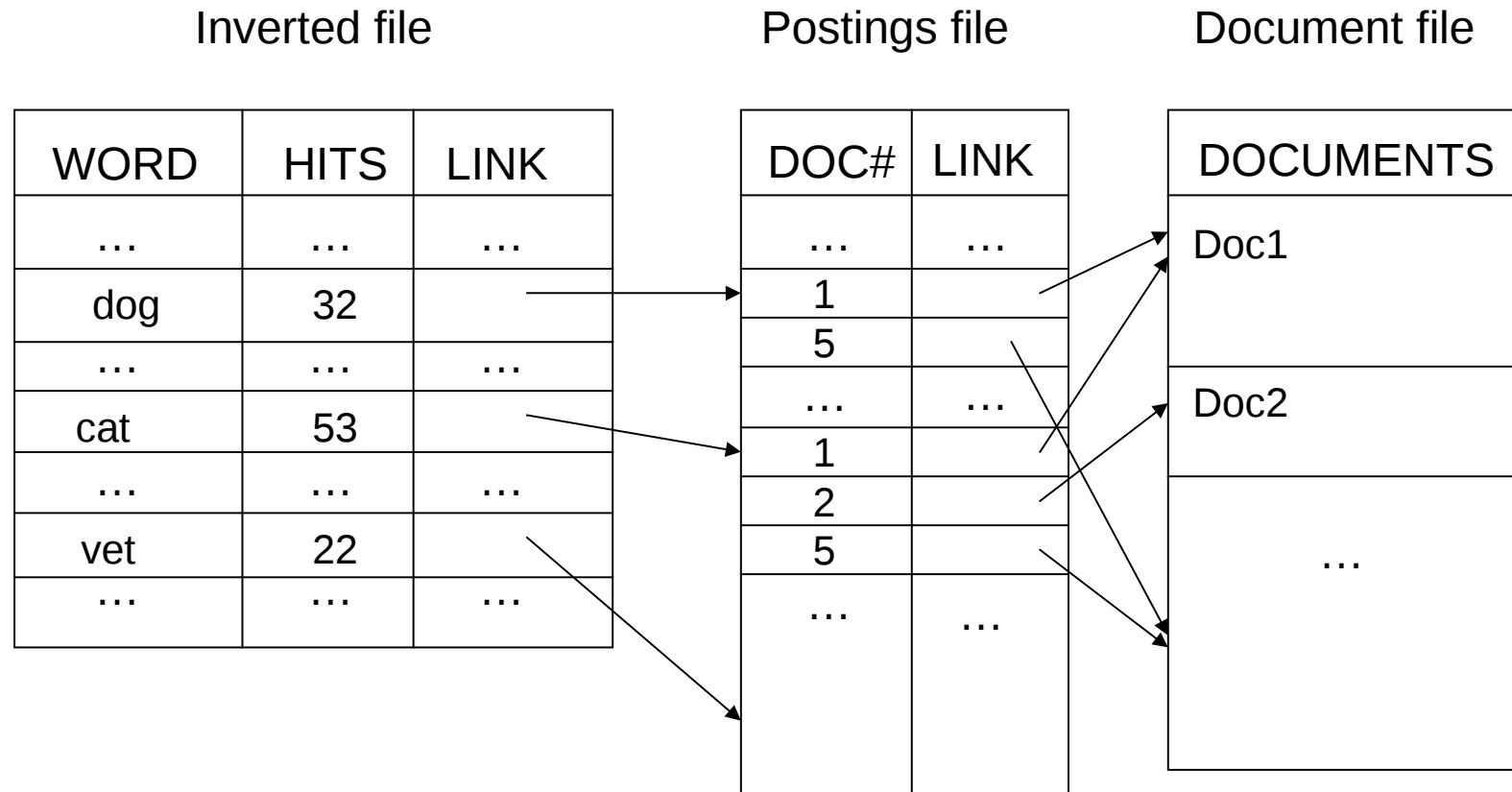
- Retrieve text documents out of a large collection
- **Query** is an unordered set of tokens (seq. of chars)
- **Result** is a set of documents relevant to the query
- **Relevance** may be *boolean*
 - i.e., document contains all tokens or not

or

degree-based

- relevance of a document usually measured by taking into account the frequency of tokens in it, normalized by frequency in all documents
- in this case, result set is ordered

Indexes to Support Full-Text Search



- Note that such an index must be maintained

Popular Full-Text Search Engines

- [Apache Lucene](#) is a Java-based full-text indexing and search library with a lot of features



- [ElasticSearch](#) is a distributed full-text search engine built on Lucene



- [Apache Solr](#) is another distributed full-text search engine, also built on Lucene



Options

- Use a full-text search engine
(as a separate component in the software stack)
- Use full-text search features built into triple stores
 - Native full-text search functionality
 - Integration of external search engines

Built-In Full-Text Search in Triple Stores

Triple store	Integrated	external
MarkLogic	x	
Virtuoso	x	
Allegrograph	x	Solr (1.5.2)
Stardog	x	
GraphDB SE	x	Lucene Solr, Elasticsearch (Enterprise)
BlazeGraph	x	SolR
Oracle 12c	x	

Native Full-Text Search in Blazegraph

- Built-in full-text search feature is custom-built
- Enabled by default in the configuration file
`com.bigdata.rdf.store.AbstractTripleStore.textIndex=true`
- B+Tree over tokens extracted from each RDF literal added to the database
 - Fast exact match on tokens
 - Fast prefix match
 - Fast match on multiple tokens
 - No performance gains for arbitrary regular expressions



Native Full-Text Search in Blazegraph

- Integration into SPARQL via the `bds:search` predicate

```
prefix bds: <http://www.bigdata.com/rdf/search#>
SELECT ?s ?p ?o WHERE {
  ?o bds:search "dog" .
  ?s ?p ?o .
}
```



Native Full-Text Search in Blazegraph

- Integration into SPARQL via the `bds:search` predicate and other related predicates

```
prefix bds: <http://www.bigdata.com/rdf/search#>
SELECT ?s ?p ?o ?score ?rank WHERE {
  ?o bds:search "dog cat" .
  ?o bds:matchAllTerms "true" .
  ?o bds:minRelevance "0.25" .
  ?o bds:relevance ?score .
  ?o bds:maxRank "1000" .
  ?o bds:rank ?rank .
  ?s ?p ?o .
}
```

Only literals that contain all of the specified search terms are to be considered



Using External Solr Services in Blazegraph

- Access to an external Solr service from within a SPARQL query is supported out of the box

```
prefix fts: <http://www.bigdata.com/rdf/fts#>
SELECT ?person ?kwDoc ?snippet WHERE {
  ?person rdf:type ex:Artist .
  ?person rdfs:label ?label .
  SERVICE <http://www.bigdata.com/rdf/fts#search> {
    ?kwDoc fts:search ?label .
    ?kwDoc fts:endpoint "http://my.solr.server/solr/select" .
    ?kwDoc fts:params "fl=id,score,snippet" .
    ?kwDoc fts:scoreField "score" .
    ?kwDoc fts:score ?score .
    ?kwDoc fts:snippetField "snippet" .
    ?kwDoc fts:snippet ?snippet .  }
} ORDER BY ?person ?score
```



Using External Solr Services in Blazegraph

- Use BIND to construct more complex search query

...

```
?person rdfs:label ?label .
```

```
BIND(CONCAT("\", ?label,  
            "\ " AND -\"expressionism\") AS ?search)
```

```
SERVICE <http://www.bigdata.com/rdf/fts#search> {  
  ?kwDoc fts:search ?search .
```

...



Native Full-Text Search in Virtuoso

- Objects of RDF triples with a given predicate or in a given graph can get indexed for full-text search
- Full-text index is in batch mode by default
 - Changes in triples are reflected in the index periodically (i.e., no strict synchronization)
 - Configuration option to enforce synchronization
- Powerful grammar for full-text queries – examples:

"dogs AND cats"

vet AND (dog OR cat)

dog AND NOT (dog NEAR cat)

"dog h*"



Native Full-Text Search in Virtuoso

- RDF triples whose object has been indexed can be found in SPARQL using the predicate `bif:contains`

```
- SELECT * WHERE {  
    ?s foaf:name ?name .  
    ?name bif:contains "rich*".  
}  
  
- SELECT * WHERE {  
    ?s ?p ?o .  
    ?o bif:contains "New AND York"  
    OPTION (score ?sc) .  
}  
ORDER BY DESC (?sc)  
LIMIT 10
```



Native Full-Text Search in AllegroGraph

- Full-text search via API and in SPARQL queries
- Syntax of full-text search queries:
 - Wildcards: `?` (single char.), `*` (multiple chars)
 - Boolean operators: `and`, and `or`
 - Double quotes around an exact phrase to match
- Multiple full-text indexes possible
- Each index works with one or more predicates, including an option to index all predicates
- Each index can be configured to include:
 - All literals, no literals, or specific types of literals
 - Full URI, just the local part, or ignore URIs entirely
 - Any combination of the four parts of a triple (incl. G)



AllegroGraph
Franz Inc.

Solr Integration in AllegroGraph

- External full-text search by using Apache Solr
 - Solr server must be installed and started separately
 - Inserts, updates, and deletes in the Solr database must be done in the application logic
- Solr features that the native solution does not have:
 - Faceted search
 - Finding words close together
 - Relevancy ranking and word boosting
 - Text clustering
 - Hit highlighting



AllegroGraph
Franz Inc.



Solr Integration in AllegroGraph (cont'd)

- Storage strategy for an RDF triple such as:

`ex:someSubj ex:somePred "text to index"`

- Tell Solr to associate "text to index" with a new id
- Then, add a new triple into AllegroGraph:

`ex:someSubj <http://www.franz.com/solrDocId> id`

- Now, you may write a SPARQL query such as:

```
PREFIX solr: <http://www.franz.com/ns/allegrograph/4.5/solr/>
```

```
PREFIX franz: <http://franz.com/ns/allegrograph/4.5/>
```

```
SELECT * WHERE {
```

```
  ?s solr:match 'medicate disastrous' .
```

```
  ?s franz:text ?text .
```

```
  ?s otherProperty ?other . }
```

- Solr can also be used from the API and the CLI

Native Full-Text Search in Stardog



- Based on Lucene
- Creation of a “search document” per RDF literal
- Disabled by default, must be enabled:

```
stardog-admin db create -o  
search.enabled=true -n myDb
```

- Three modes for rebuilding indexes, configured by setting `search.reindex.mode` to:
 - *sync* (synchronous rebuild with a transacted write, dflt.)
 - *async* (asynchronous rebuild “as soon as possible”), or
 - *scheduled* (cron expression specifies when to rebuild)

Native Full-Text Search in Stardog (cont'd)



- Search syntax as in Lucene:
 - e.g., wildcards `?` and `*`, fuzzy with similarity `~0.5`

- Use it on the command line:

```
stardog query search -q "html" -l 10 myDb
```

- Use it in SPARQL:

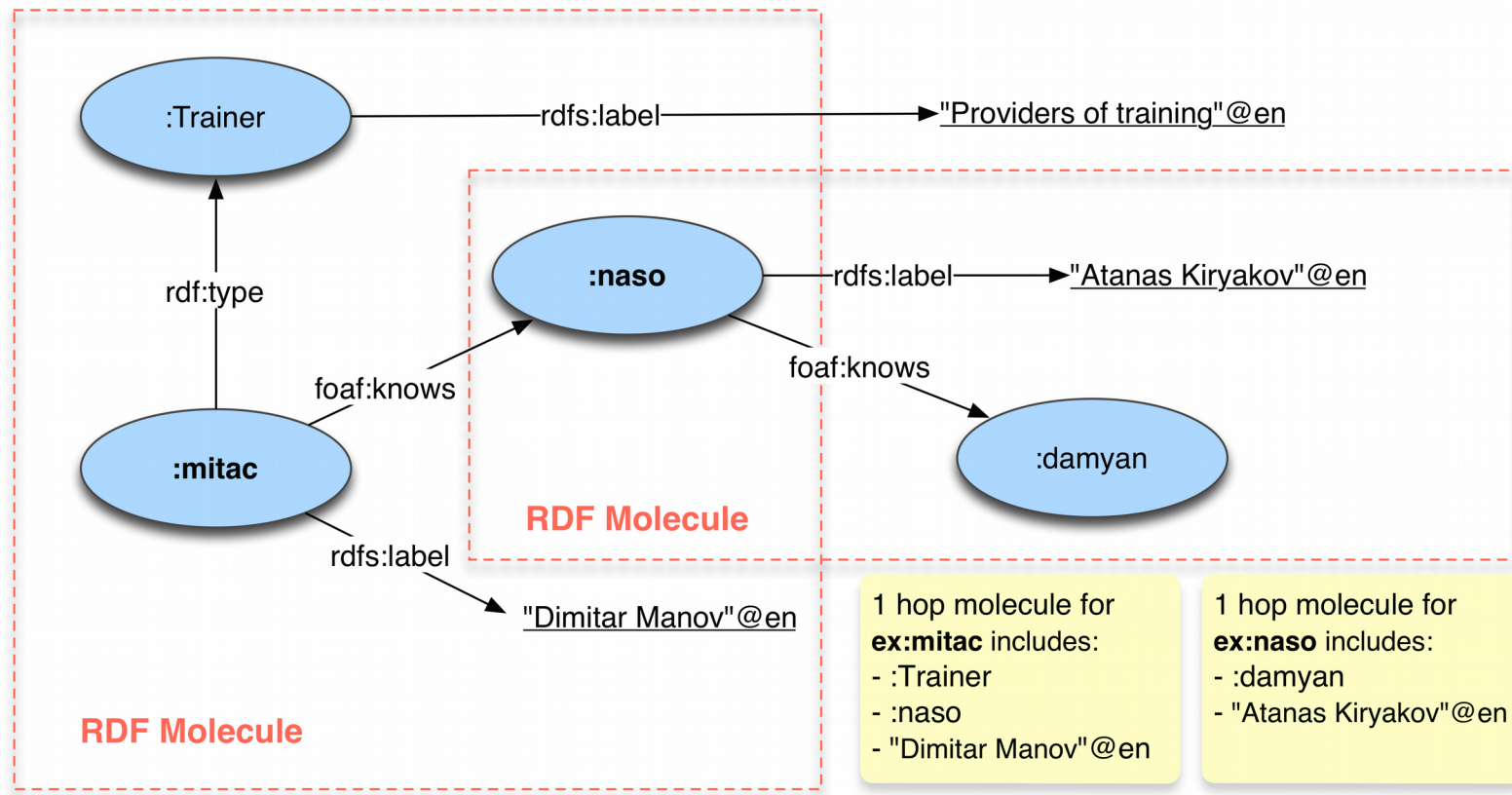
```
SELECT DISTINCT ?s ?score WHERE {  
  ?s ?p ?l.  
  (?l ?score) <tag:stardog:api:property:textMatch>  
                                                    "html" }
```

```
SELECT DISTINCT ?s ?score WHERE {  
  ?s ?p ?l.  
  (?l ?score) <tag:stardog:api:property:textMatch>  
                                                    ("html" 0.5 10) }
```

Native Full-Text Search in GraphDB



- Based on Lucene
- For each RDF node, text document that is made up of other nodes reachable from the node (“molecule”)



Native Full-Text Search in GraphDB



- Based on Lucene
- For each RDF node, text document that is made up of other nodes reachable from the node (“molecule”)
- Indexes can be parameterized
 - what kinds of nodes are indexed (URIs / literals)
 - literals with specific language tags only
 - what is included in the notion of “molecule”
 - size of the “molecule” to index
 - relevance of nodes boosted by RDF Rank values
 - alternative analyzers
 - alternative scorers
- Multiple, differently configured full-text indexes possible

Native Full-Text Search in GraphDB (cont'd)

- Setting up an (example) configuration for full-text indexes:

```
PREFIX luc: <http://www.ontotext.com/owlim/lucene#>
INSERT DATA {
    luc:index          luc:setParam  "uris" .
    luc:include        luc:setParam  "literals" .
    luc:moleculeSize  luc:setParam  "1" .
    luc:includePredicates luc:setParam
        "http://www.w3.org/2000/01/rdf-schema#label" .
}
```

- Creating a new index (uses the previous configuration):

```
PREFIX luc: <http://www.ontotext.com/owlim/lucene#>
INSERT DATA {
    luc:myTestIndex luc:createIndex "true" .
}
```

Native Full-Text Search in GraphDB (cont'd)

- Use the index in a query

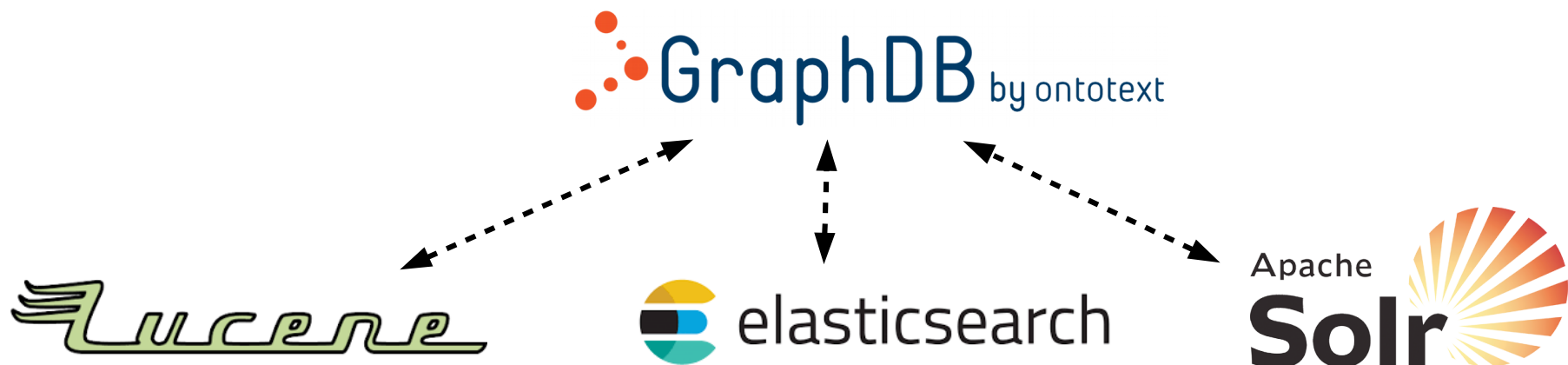
- ```
PREFIX luc: <http://www.ontotext.com/owlim/lucene#>
SELECT * { ?id luc:myTestIndex "ast*" }
```
- ```
PREFIX luc: <http://www.ontotext.com/owlim/lucene#>
SELECT * {
    ?id luc:myTestIndex "lucene query string" .
    ?node luc:score ?score .
} ORDER BY ( ?score )
```

- Incremental update

```
PREFIX luc: <http://www.ontotext.com/owlim/lucene#>
INSERT DATA {
    luc:myTestIndex luc:addToIndex ex:newURI .
}
```


Connecting External Search to GraphDB

- Connectors for Lucene, Solr, and Elasticsearch (the latter two only in the enterprise edition of GraphDB)
- Similar to how Blazegraph supports access to an external Solr service from within a SPARQL query



Native Full-Text Search in MarkLogic

- Built-in full-text search feature is custom-built
- Full-text indexes created when loading a document
- Powerful grammar for string queries
 - Examples:



`(cat OR dog) NEAR vet`

`dog NEAR/30 vet`

`cat -dog`

`"cats and dogs"`

`dog NOT_IN "dog house"`

`dog BOOST cat`

Built-In Full-Text Search in Triple Stores

Triple store	Integrated	external
MarkLogic	x	
Virtuoso	x	
Allegrograph	x	Solr (1.5.2)
Stardog	x	
GraphDB SE	x	Lucene Solr, Elasticsearch (Enterprise)
BlazeGraph	x	SolR
Oracle 12c	x	