

# Q2Semantic: A Lightweight Keyword Interface to Semantic Search

Zhenning Shangguan

For CSCI-6965 Class

Mar-03-2009

# Introduction

- Semantic search should support more expressive queries that addresses complex information needs
- However, current semantic query engines only support formal queries, e.g., SPARQL
  - Users must learn complex syntax first
  - Users also must know the underlying schema and vocabularies in the data set
- Keyword search is one possible solution
  - Simple syntax
  - Open vocabularies

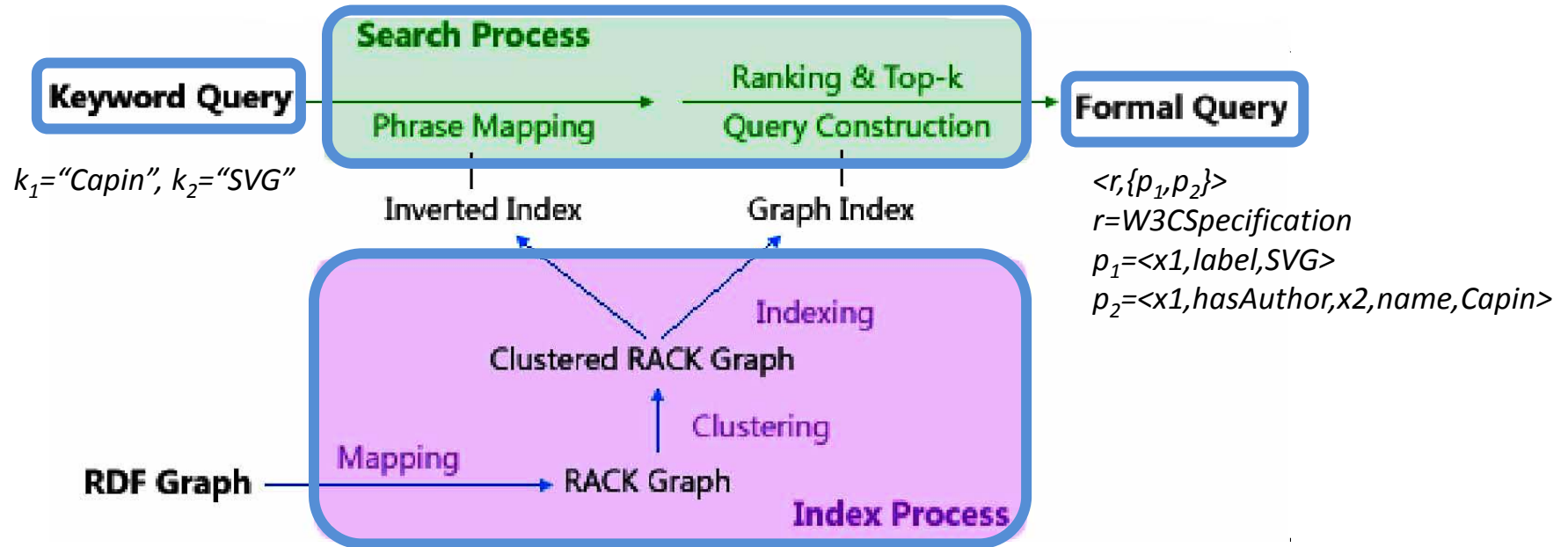
# Keyword Search

- How to bridge the gap keyword and formal queries
  - IR and DB community
  - SW community
    - SPARK (ISWC'07), Tran. T (ISWC'07), etc.
- Challenges
  - Open vocabulary
  - Ranking
  - Scalability

# Contributions

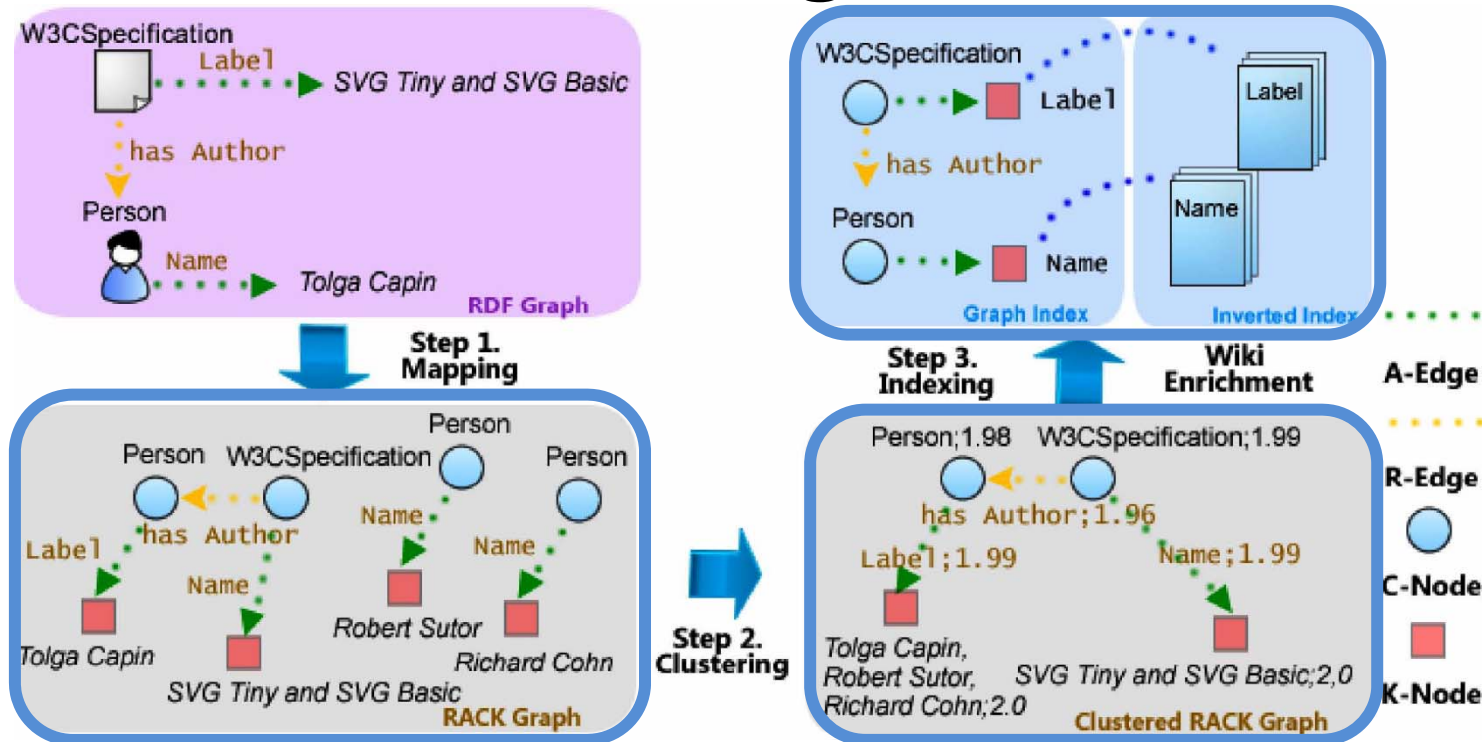
- Terms extracted from Wikipedia to enrich literals described in the original RDF data
- Mechanisms for query ranking, considering several relevant factors
- Novel graph data structure called clustered RACK graph and an exploration algorithm
  - Allows the construction of the top- $k$  queries

# Workflow of Q2Semantic



- Input: a keyword query  $K$  composed of keyword phrases  $\{k_1, k_2, \dots, k_n\}$ .
- Search Process
  - Phrase Mapping
  - Query Construction and Ranking
- Index Process
  - Mapping, Clustering and Indexing
- Output: a formal query  $F$  as a tree of the form  $\langle r, \{p_1, p_2, \dots, p_n\} \rangle$ , where  $r$  is the root node of  $F$  and  $p_i$  is a path in  $F$ .

# Data Pre-Processing in Q2Semantic



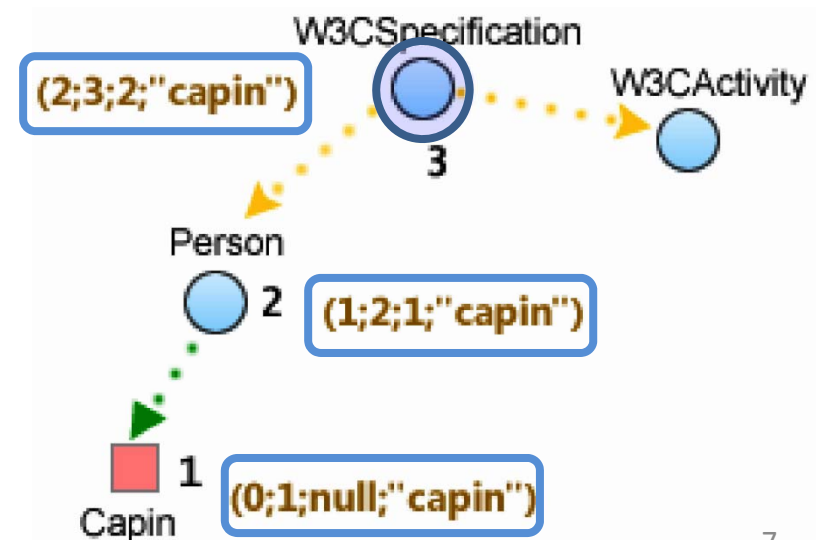
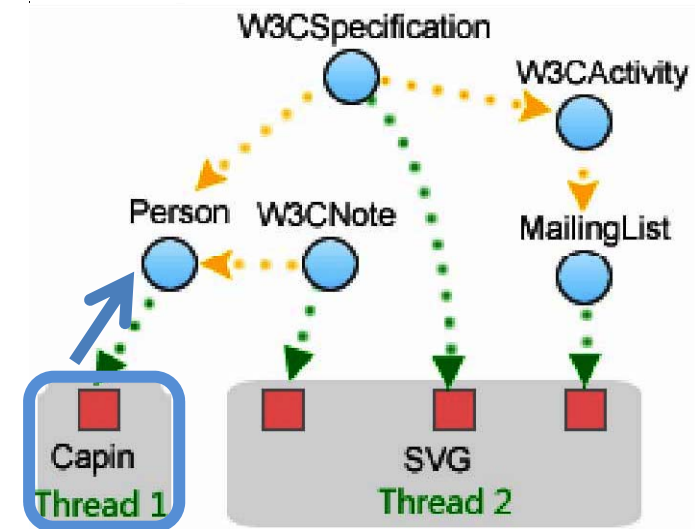
## Four rules for mapping from RDF graph to RACK graph

### Four rules for clustering RACK graph

- Every instance of the RDF graph is mapped to a **C-Node** labeled by the concept name that the instance belongs to.
- Two C-Nodes are clustered to one if they have the same label.
- Two R-Edges are clustered to one if they have the same label and connect the same pair of C-Nodes.
- Every attribute value is mapped to a **K-Node** labeled by the value literal.
- Two A-Edges are clustered to one if they have the same label and connected to the same C-Nodes.
- Every relation is mapped to a **R-Edge** that is labeled by the relation name and connects two C-Nodes.
- Two K-Nodes are clustered to one if they are connected to the same A-Edge. The resulting node inherits the labels of both these K-Nodes.
- Every attribute is mapped to an **A-Edge** that is labeled by the attribute name and connects a C-Node with a K-Node.

# Query Interpretation in Q2Semantic

- Phrase Mapping
- Query Construction
  - Thread Expansion (*T-Expansion*)
  - Cursor Expansion (*C-Expansion*)
  - Two strategies for expansion
    - Intra-Thread Strategy
    - Inter-Thread Strategy
  - Optimization for Top-*k* Termination
  - Optimization for Repeated Expansion



# Query Construction Algorithm

**Input:**  $K = \{k_1, k_2, \dots, k_n\}$ , where  $k_i$  hits the K-Nodes

$KL_i = \{k\text{-node}_{i1}, k\text{-node}_{i2}, \dots, k\text{-node}_{im_i}\}$  with the matching relevance  
as  $S_i = \{s_{i1}, s_{i2}, \dots, s_{im_i}\}$ ;

**Output:**  $A$ : result set, initially  $\emptyset$ ;

**Data:**  $\tau_{prune}$ : pruning threshold, initially  $\tau_0$ ;

```
1 for  $i \in [1, n]$  do
2   |  $t_i = \text{new Thread}()$ ;
3   | for  $j \in [1, m_i]$  do
4   |   |  $t_i.add(\text{new Cursor}(s_{ij}, k\text{-node}_{ij}, \text{NULL}, k_i))$ ;
5   |   end
6 end
7 while  $\exists i \in [1, n] : t_i.peekCost() \neq \infty$  do
8   |  $j \leftarrow$  pick from  $[1, n]$  in a round-robin fashion;
9   |  $c \leftarrow t_j.popMin()$ ;
10  |  $C\text{-Expansion}(c)$ ; //  $A$  and  $\tau_{prune}$  will be updated here;
11  | if  $t_j.peekCost() > \tau_{prune}$  then
12  |   | Output the top  $k$  answers in  $A$ ;
13  |   end
14 end
```

# Query Ranking in Q2Semantic

- Path only

$$R_1 = \sum_{1 \leq i \leq n} \left( \sum_{e \in p_i} 1 \right)$$

- Adding matching relevance

$$R_2 = \sum_{1 \leq i \leq n} \left( \frac{1}{D_i} \sum_{e \in p_i} 1 \right)$$

- Adding importance of edges and nodes

$$R_3 = cost_r \sum_{1 \leq i \leq n} \left( \frac{1}{D_i} \sum_{e \in p_i} cost_e \right)$$

$$cost_{node} = 2 - \log_2 \left( \frac{|node|}{N} + 1 \right)$$

$$cost_{edge} = 2 - \log_2 \left( \frac{|edge|}{M} + 1 \right)$$

# Experiment Setup

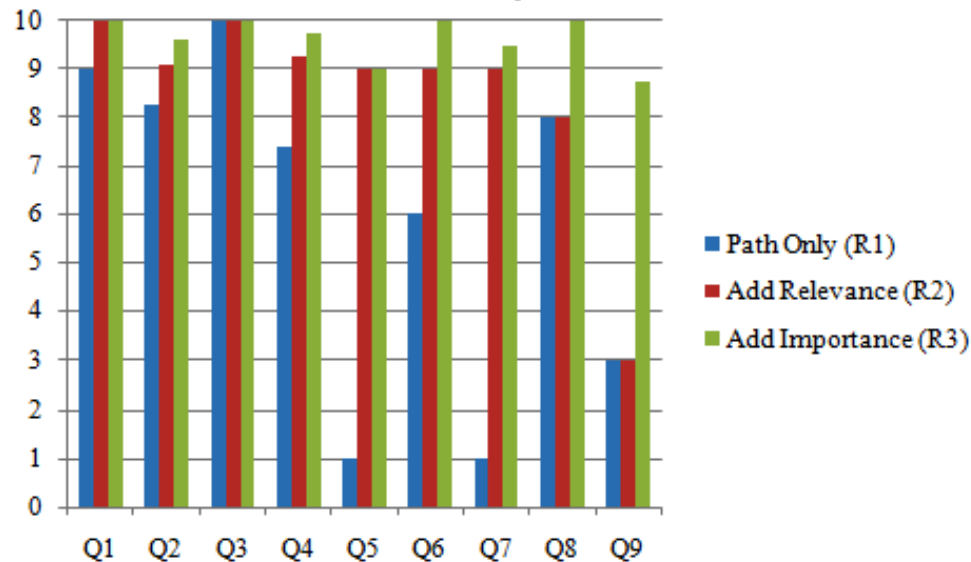
- TAP (220K triples)

| Query | Keywords                                      | Potential information need  |
|-------|---|---|
| Q3    | Supergirl                                     | Who is called "supergirl"   |
| Q5    | Strip, Las Vegas                              | What is the well-known "Strip" in Las Vegas   |
| Q9    | Web Accessibility Initiative, www-rdf-perllib | Find persons who work for Web Accessibility Initiative and involved in the activity with mailing list "www-rdf-perllib" |

- DBLP (26M triples)
  - 100 valid queries by combining literals from different attributes (from one to three keywords)
- LUBM(1,0), LUBM(20,0) and LUBM(50,0)
  - 8 queries from the LUBM Query Set (LQ) are used by removing 2 cyclic queries and 4 queries requiring reasoning support

# Effectiveness Evaluation

- A simple but effective metric *Target Query Position (TQP)*:  $TQP = 11 - P_{target}$
- TQPs of different ranking schemes on TAP

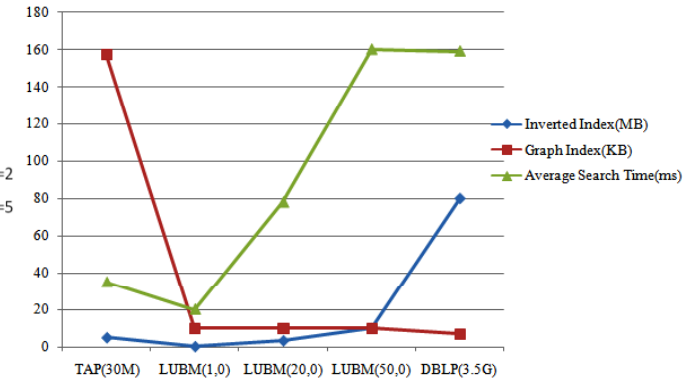
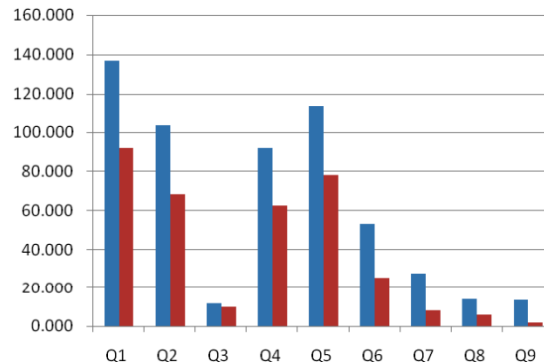
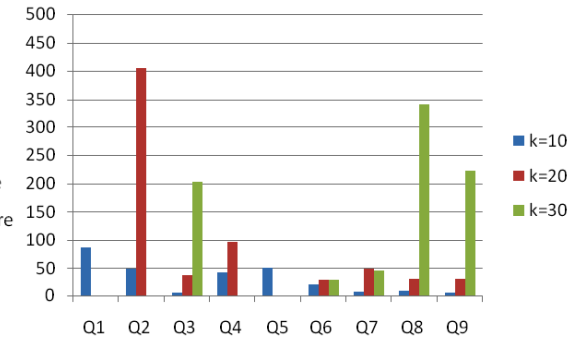
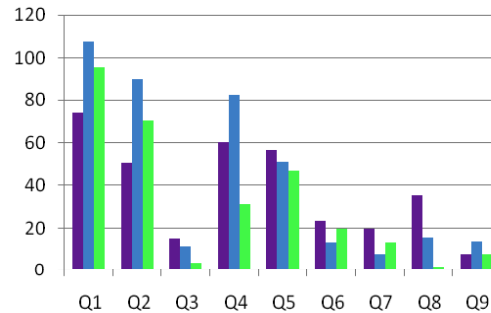


- TQPs on LUBM benchmark queries

|              |     |     |     |     |     |     |      |      |
|--------------|-----|-----|-----|-----|-----|-----|------|------|
| <b>TQP</b>   | 9   | 10  | 9   | 10  | 8   | 8   | 9    | 10   |
| <b>Query</b> | LQ1 | LQ3 | LQ4 | LQ6 | LQ7 | LQ8 | LQ10 | LQ14 |

# Efficiency Evaluation

- Search time under different ranking schemes
- Search time under different top- $k$
- Performance of penalty parameters
- Index size and search time on different datasets
- RACK graph vs. clustered RACK graph



|                   | R-Edge  |            | A-Edge   |            | C-Node  |            | K-Node   |            |
|-------------------|---------|------------|----------|------------|---------|------------|----------|------------|
| <b>TAP</b>        | 41914   | <b>158</b> | 87796    | <b>666</b> | 167656  | <b>314</b> | 87796    | <b>666</b> |
| <b>LUBM(1,0)</b>  | 41763   | <b>43</b>  | 30230    | <b>39</b>  | 16221   | <b>13</b>  | 30230    | <b>39</b>  |
| <b>LUBM(20,0)</b> | 1127823 | <b>43</b>  | 815511   | <b>39</b>  | 411815  | <b>13</b>  | 815511   | <b>39</b>  |
| <b>LUBM(50,0)</b> | 2788382 | <b>43</b>  | 2015672  | <b>39</b>  | 1018501 | <b>13</b>  | 2015672  | <b>39</b>  |
| <b>DBLP</b>       | 5619110 | <b>19</b>  | 12129200 | <b>23</b>  | 1366535 | <b>5</b>   | 12129200 | <b>23</b>  |

# Conclusions and Future Work

- For the efficiency purpose, we propose a new clustered graph index structure as a summary of the original RDF data and support top- $k$  formal query construction on it.
- For the effectiveness purpose, we design well-performed ranking schemes. Additionally, we leverage knowledge from Wikipedia to enrich and disambiguates the keyword queries.
- Future Work
  - Query Capability Extension
  - Clustering Method

# Backup Slides

- Clustered graph structure
  - Corresponds to the summary of the original ontology
  - Considered as “reduced data space”
  - Used when computing top-k queries
- Query ranking
  - Query length
  - Relevance of ontology elements to the query
  - Importance of ontology elements