Distributed Keyword Search over RDF via MapReduce

Roberto De Virgilio and Antonio Maccioni
Semantic Web is distributed

- The (Semantic) Web is distributed
- We have distributed and cloud-based Infrastructures for data processing
- The Linked (Open) Data are getting popular also to non-expert users
Distributed Keyword Search
MapReduce: a simple programming model

MapReduce

Map:
- Accepts *input* key/value pair
- Emits *intermediate* key/value pair

Reduce:
- Accepts *intermediate* key/value pair
- Emits *output* key/value pair
The problem
The problem

- **Query:**
  - Bernstein
  - 2008
  - SIGMOD
The problem

Query:
- Bernstein
- 2008
- SIGMOD
Existing Approach

Query:
- Bernstein
- 2008
- SIGMOD
Existing Approach

relevance
Existing Approach

relevance

S1
S2
S3
S4
...
Sn
Existing Approach

- drawbacks:
  - low specificity
  - high overlapping
  - high computational cost
  - centralized computation
Desired direction

relevance
Desired direction

relevance

S1

S2

Sk
**Desired** direction

- **strong points:**
  - linear computational cost
  - monotonic ranked result
  - low overlapping
  - distributed computation
From **Graph** parallel to **Data** parallel
Graph Data Indexing

Breadth First Search

Distributed Storage

PATH STORE 1:
- \( p_1 : \text{pub1-year-2008} \)
- \( p_2 : \text{pub1-author-aut1-name-Bernstein} \)
- \( p_4 : \text{pub2-year-2008} \)
  ...

PATH STORE 2:
- \( p_3 : \text{pub1-acceptedBy-conf1-name-SIGMOD} \)
- \( p_5 : \text{pub2-editedBy-conf1-name-SIGMOD} \)
  ...
Paths and Templates

Query = \{Bernstein, 2008, SIGMOD\}

Path Store 1 \rightarrow Path Store 2 \rightarrow Path Store j

\[
\text{Path} \quad \text{Template} \\
[pub1-acceptedBy-conf1-name-SIGMOD] \\
[#-acceptedBy- #-name- #]
\]
Paths Clustering

$CL_1 : (cl_1[#-year-#] : p_1, p_4) (cl_2[#-author-#-name-#] : p_2)$

$CL_2 : (cl_3[#-acceptedBy-#-name-#] : p_3) (cl_4[#-editedBy-#-name-#] : p_5)$

cl1

[1] p1  pub1  year  2008

cl2

[1] p2  pub1  author  aut1  name  Bernstein

cl3

[1] p3  pub1  acceptedBy  confi  name  SIGMOD

cl4

[1] p5  pub2  editedBy  confi  name  SIGMOD
Unlike all current approaches, we are independent from the scoring function: we do not impose a monotonic, aggregative nor an “ad-hoc for the case” scoring function.

\[
score(sg, Q) = \frac{\alpha(sg)}{\omega_{str}(sg)} \cdot \sum_{q \in Q} (\rho(q) \cdot \omega_{ct}(sg, q))
\]
Scoring

Unlike all current approaches, we are independent from the scoring function: we do not impose a monotonic, aggregative nor an “ad-hoc for the case” scoring function.

\[ \text{score}(sg, Q) = \frac{\alpha(sg)}{\omega_{str}(sg)} \cdot \sum_{q \in Q} (\rho(q) \cdot \omega_{ct}(sg, q)) \]

deploying:

- how keywords are strictly connected
- length of the paths

Roberto De Virgilio, Antonio Maccioni, Paolo Cappellari: A Linear and Monotonic Strategy to Keyword Search over RDF Data. ICWE 2013: 338-353
Scoring

Unlike all current approaches, we are independent from the scoring function: we do not impose a monotonic, aggregative nor an "ad-hoc for the case" scoring function.

\[
\text{score}(sg, Q) = \frac{\alpha(sg)}{\omega_{str}(sg)} \cdot \sum_{q \in Q} (\rho(q) \cdot \omega_{ct}(sg, q))
\]

- **topology:**
  - how keywords are strictly connected
  - length of the paths

- **relevance:**
  - information carried from nodes
  - implementation of TF/IDF
Building solutions

- different strategies:
  - linear time complexity in one round of MR
  - monotonically in a quadratic time and 2k rounds of MR
Linear Strategy (Map)

\[ \textbf{Map}: \text{Iterates over the clusters;} \]
- \textbf{key}: position of the path in the cluster
- \textbf{value}: path itself

\[ \mathcal{L}_1: \]
\[ (\text{cl}_1[\#-year-#]: p_1, p_4) \]
\[ (\text{cl}_2[\#-author-\#-name-#]: p_2) \]

\[ \mathcal{L}_2: \]
\[ (\text{cl}_3[\#-acceptedBy-\#-name-#]: p_3) \]
\[ (\text{cl}_4[\#-editedBy-\#-name-#]: p_5) \]

\[ c_{11} \]
\[ [1] \quad p_1 \quad \text{pub1} \quad \text{year} \quad 2008 \]
\[ [2] \quad p_4 \quad \text{pub2} \quad \text{year} \quad 2008 \]

\[ c_{12} \]
\[ [1] \quad p_2 \quad \text{pub1} \quad \text{author} \quad \text{autr} \quad \text{name} \quad \text{Bernstein} \]

\[ c_{13} \]
\[ [1] \quad p_3 \quad \text{pub1} \quad \text{acceptedBy} \quad \text{confi} \quad \text{name} \quad \text{SIGMOD} \]

\[ c_{14} \]
\[ [1] \quad p_5 \quad \text{pub2} \quad \text{editedBy} \quad \text{confi} \quad \text{name} \quad \text{SIGMOD} \]

\[ <1, p_1>, <2, p_4>, <1, p_2> \]

\[ <1, p_3>, <1, p_5> \]
Linear Strategy (Reduce)

- **Reduce**: Each machine receives a list of paths out of which it computes connected components;
  - **key**: position of the path in the cluster
  - **value**: path itself

\[
<1, p1>, <2, p4>, <1, p2> \quad <1, \{p1, p2, p3, p5\}>
\]

\[
<1, p3>, <1, p5> \quad <2, \{p4\}>
\]

- Each connected component is a final solution to output
  - **S1**: \{p1, p2, p3, p5\}
  - **S2**: \{p4\}
Linear Strategy: solutions

- Each connected component is a final solution to output

- S1: \{p1, p2, p3, p5\}
- S2: \{p4\}
Monotonic Strategy (Map round 1)

- **Map**: Iterates over the clusters (taking only the best ones);
  - **key**: position of the path in the cluster
  - **value**: path itself

\[
\mathcal{CL}_1 : \\
( cl_1[#-year-#] : p_1, p_4 ) \\
( cl_2[#-author-#-name-#] : p_2 )
\]

\[
\mathcal{CL}_2 : \\
( cl_3[#-acceptedBy-#-name-#] : p_3 ) \\
( cl_4[#-editedBy-#-name-#] : p_5 )
\]

c11

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>pub1</th>
<th>year</th>
<th>pub1</th>
<th>2008</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>p4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

c12

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>pub1</th>
<th>author</th>
<th>pub1</th>
<th>Bernstein</th>
</tr>
</thead>
</table>

\(<1, p1>, <1, p2>\)

\(<1, p3>, <1, p5>\)
Monotonic Strategy (Reduce round 1)

- **Reduce**: Each machine receives a list of paths out of which it computes global connected components;
  - **key**: position of the path in the cluster
  - **value**: path itself

\[
\begin{align*}
<1, p1>, &<1, p2> \\
<1, p3>, &<1, p5> \\
\end{align*}
\]

\[
<1, \{p1, p2, p3, p5\}>
\]

- Then it invokes a new round of MapReduce
Monotonic Strategy (Map round 2)

- **Map**: Dispatches the connected components to different machines;
  - **key**: occurrence of connected component
  - **value**: connected component itself

```
<1, {p1, p2, p3, p5}>
```
Monotonic Strategy (Reduce round 2)

- **Reduce**: Each machine receives a connected component and iterates on the paths;
  - **tau-test**: a variant of the TA algorithm
  - it determines if some path is **exceeding** in the connected component

\( <1, \{p_1, p_2, p_3, p_5\}> \)

\( S = \{p_1, p_2, p_3\} \)

- Discarded paths and discarded connected components are reinserted in the clusters
**Tau-test**

**Property 1** Given a query $Q$ and a path $p$, $\text{score}(p, Q) = \text{score}(\{p\}, Q)$.

**Property 2** Given a query $Q$, a set of paths $P$ in which $p_\beta$ is the more relevant path (i.e. $\forall p_j \in P$ we have that $\text{score}(p_\beta, Q) \geq \text{score}(p_j, Q)$) and $P^*$ is its power set, we have $\text{score}(S = P_i, Q) \leq \text{score}(S = \{p_\beta\}, Q) \quad \forall P_i \subseteq P^*$.

**Theorem 1.** Given a query $Q$, a scoring function satisfying **Property 1** and **Property 2**, a connected component $cc$, a subset $\text{opt}S \subset cc$ representing an optimum solution and a candidate path $p_x \in cc \setminus \text{opt}S$, $S = \text{opt}S \cup \{p_x\}$ is still optimum iff $\text{score}(S, Q) \geq \tau$.

Roberto De Virgilio, Antonio Maccioni, Paolo Cappellari: *A Linear and Monotonic Strategy to Keyword Search over RDF Data*. **ICWE 2013**: 338-353
Monotonic Strategy: solutions

- Each connected component is a final solution to output

- **S1**: \{p1, p2, p3\}
  - Bernstein
  - author
  - pub1
  - acceptedBy
  - aut1
  - confi
  - name
  - SIGMOD

- **S2**: \{p4, p5\}
  - 2008
  - editedBy
  - pub2
  - confi
  - name
  - SIGMOD
Experiments

- We deployed YaaniiMR on EC2 clusters.
  - cc1.4xlarge: 10, 50 and 100 nodes.

- YaaniiMR is provided with the Hadoop file system (HDFS) version 1.1.1 and the HBase data store version 0.94.3

- The performance of our systems has been measured with respect to data loading, memory footprint, and query execution.
300M triples

DBpedia

Data Loading

2008 edition: 1200M triples

Upload Time (sec)

DBPedia
Billion

10 nodes
50 nodes
100 nodes

45,49
50,49
56,43

314,26
348,84
389,88

A new application award
Semantic Web Challenge
http://challenge.semanticweb.org
Memory Consumption

300M triples

DBpedia

17k triples

size for node (MB)

10000
1000
100
10
1

overhead  space consumption

1.1  1.4
1.5

1.8

Mondial  DbPedia  Billion

10-nodes

2008 edition: 1200M triples
End-to-end job runtimes
[Coffman et al. Benchmark]

L = Linear strategy  M = Monotonic strategy

Job Runtime (sec)

<table>
<thead>
<tr>
<th></th>
<th>L</th>
<th>M</th>
<th>L</th>
<th>M</th>
<th>L</th>
<th>M</th>
<th>L</th>
<th>M</th>
<th>L</th>
<th>M</th>
<th>L</th>
<th>M</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mondial</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>Dbpedia 10-nodes</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>Billion</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>Dbpedia 50-nodes</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>Billion</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>Dbpedia 100-nodes</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>Billion</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
</tbody>
</table>

300M triples  1200M triples  17k triples
Questions?