USER GUIDE

Yan's Group (sqyang@cs.ucsb.edu)

Overview

The package was built for the work "Ontology-based Subgraph Querying" (in ICDE 2013). It includes the indexing and query processing. Refer to the paper for more details about the techniques. The package is implemented in C++ and compiled/tested in both Windows and Linux.

Package Manual

ONTOLOGY INDEXING

COMMAND
query –v [ontology graph file] [ontology index file]
PARAMETERS
[ontology graph file] ontology graph file (input)
[ontology index file] ontology index file (output)
EXAMPLE:
[user@server ~]$ ./query –v ont.graph ont.index Reading graph ... line 1000000 …… line 2000000 Time # 10s pivots # 12345 Graph View Released Graph Released

Input graph format

FORMAT
G # n_v n_e v id type ... e id1 id2 label

...

EXPLANATION

n_v: vertex size.

n_e: edge size.

v id type: vertex id and type

e id1 id2 type: edge source id, destination id and edge type

EXAMPLE

G # 4 3

v 1 1

v 2 1

v 3 2

v 4 3

e 1 4 4

e 2 4 4

e 4 3 5

Output index format

FORMAT

GV 1

v id1 pivot1

v id2 pivot2

...

EXPLANATION

id: a label in the ontology graph.

pivot: a label used to represent the label (id), e.g., “bird” can be used as a general label of “pelican”. Note here all text labels are denoted by numeric numbers.

EXAMPLE

GV 1

v 1 10

v 2 3

v 3 3

v 4 7

…

Note: here we randomly select pivot labels to represent their nearby labels wrt. the ontology graph. Another indexing method is to partition the ontology graph and then select a pivot from

each partition to represent the labels in the partition. Other ontology indexing techniques can also be applied here as long as the output index files follow the same format as described above.

GRAPH INDEXING

COMMAND

query –i [data graph file] [ontology index file] [data index file]

PARAMETERS

[data graph file] data graph file

[ontology index file] ontology index file (output from the ontology indexing step)

[data index file] data index file

EXAMPLE:

```
[user@server ~]$ ./query -i data.graph ont.index data.index
Reading graph ...
line 1000000
……
line 2000000
Time # 10s
line 1000000
Graph View Map # 120000
1000000
Super Graph # 33 0
Graph View Released
Time # 81s
Super Graph # 396021 1515244
Output to data.index
Graph View Released
Graph Released
```

The format of the data graph and the ontology index (refer to “Ontology Indexing”)

Output index format (A Super Graph)

FORMAT

```
SG n_sn n_se
N sid1 slabel1 size1
v id1
v id2
```

...

N sid2 slabel2 size2

EXPLANATION

n_sn, n_se: number of the super nodes and super edges

sid, slabel, size: the id, label and size (number of data nodes) of a super node

id: data node in the super node

EXAMPLE

SG 100 400

N 1 10 2

v 1

v 10

N 2 3 3

V 2

V 6

V 13

…

QUERY PROCESSING

COMMAND

query –i [data graph file] [ontology file] [ontology index file] [data index file] [query file] [result file]

PARAMETERS

[data graph file] data graph file

[ontology file] ontology graph file

[ontology index file] ontology index file (output from the ontology indexing step)

[data index file] data index file (output from the data indexing step)

[query file] query file

[result file] query result

EXAMPLE:

[user@server ~]$ ./query -i data.graph ont.graph ont.index data.index query.graph result

Reading graph ...

line 1000000

line 2000000

Time # 10s

Reading graph ...

```
line 1000000
line 2000000
Time # 20s
line 1000000
Graph View Map # 123456
Construct super graph from file...
Super Graph # 1000 2000
Reading graph ...
Time # 0s
Super Graph # 4 0
Time # 0s
Super Graph # 4 6
Query rewrite ...
Query rewrite ...
Start searching...
Result # 100 20
Time # 25.47s
Graph Released
Graph View Released
Graph Released
Graph Released
```

Output result format

FORMAT

```
id node_match value
```

EXPLANATION

id: the index of a result

node_match: the matching node in the graph corresponding to the query node

value: in terms of label matching

EXAMPLE

Query:

```
G # 4 6
v 1 10
v 2 3
v 3 9
v 4 20
```

```
e 1 2 12
e 1 3 14
e 3 2 4
e 4 1 19
e 4 2 193
e 4 3 405
1	2;9;4;3;	1
2	5;10;15;18;	3
…
```