Intuitive and Interactive Query Formulation to Improve the Usability of Query Systems for Heterogeneous Graphs

Nandish Jayaram
University of Texas at Arlington
PhD Advisors: Dr. Chengkai Li, Dr. Ramez Elmasri

VLDB 2015 Phd Workshop
August 31st 2015
Outline

- Motivation: Graph Data Usability
- Visual Interface for Recommendation Based Interactive Graph Query Formulation (Orion)
- Graph Query By Example (GQBE)
Large Heterogeneous Graphs

Large, complex and schema-less graphs capturing millions of entities and relationships between them!

Linking Open Data: 52 billion RDF triples
Freebase: 1.8 billion triples
DBpedia: 470 million triples
Yago: 120 million triples
Specifying Queries for Graphs

**SQL QUERY:**
SELECT Founder.subj, Founder.obj
FROM Founder, Nationality, HeadquarteredIn
WHERE
  Founder.property = 'founded' AND
  Founder.subj = Nationality.subj AND
  Nationality.property = 'nationality' AND
  Founder.obj = HeadquarteredIn.subj AND
  HeadquarteredIn.property = 'headquartered_in';

**SPARQL QUERY:**
SELECT ?company ?founder WHERE {
  :?founder dbo:founded :?company .
  :?founder dbo:nationality :USA .
  :?company dbprop:headquartered_in :Silicon Valley .
}
Simpler Querying Paradigms

- **Keyword Search**
  - Keyword search in Graphs [Kargar, VLDB’11], BLINKS [He, SIGMOD’07]
  - Limitation: Articulating keyword query for graphs is not simple

- **Approximate Query Specification and Answering**
  - NESS: uses neighborhood-based indexes to quickly find approximate matches to a query graph [Khan, SIGMOD’11]
  - TALE: approximate large graph matching [Tian, ICDE’08]
  - Limitation: Users still have to formulate the initial query graph
Visual Query Formulation Systems

- **Relational Databases**
  - CLIDE [Petropoulos, SIGMOD’06,07]

- **Graph Databases**
  - VOGUE, PRAGUE, Gblender, [Bhowmick, CIDR’13, ICDE’12, SIGMOD’11], GRAPHITE [Chau, ICDMW’08]

- **Single Large Graphs**
  - QUBLE [Bhowmick, VLDB’14]

- **Limitations:**
  - New relevant query components are not automatically recommended to users
  - Users require a good knowledge of the underlying schema
Desiderata of a User Friendly Query System

- **Usability**
  - An easy-to-use graphical interface for formulating query graphs
  - Easier paradigm to query complex heterogeneous graphs

- **Ability to express exact query intent**
  - Schema agnostic users assisted by an intelligent query system
Dissertation Research Outline

Possible Future Work

- Interactive Graph
  - Query Completion
    - (Orion)
  - Visual Query Formulation Interface
- <Jerry Yang, Yahoo!>
- Input Tuples
- Graph Query By Example (GQBE)

Query Formulation

Query Graph Discovery

Query Graph

Answer Space Modeling

Query Lattice

Query Processing

Ranked Top-k Answer Tuples
- <David Filo, Yahoo!>
- <Sergey Brin, Google>
- <Steve Wozniak, Apple Inc.>

User Feedback

Result Browsing

Possible Future Work
Visual Interface for Recommendation Based Interactive Query Formulation (Orion)

Ongoing work
Problem Statement

- Given a large heterogeneous graph, iteratively suggest edges to help build a query graph
- An interactive graphical user interface for building query components
- An edge recommendation system that ranks edges based on their relevance to the user’s query intent
Orion Interface (idir.uta.edu/orion)

Query Canvas

Dynamic help indicating possible actions at every moment

Useful tips for basic operations

Information Panel
Modes of Operation: Passive and Active

Grey edges and nodes automatically suggested in **passive mode**

A new node added in **active mode**

A suggested edge accepted by the user (with blue node) are **positive edges**. Grey edges ignored are **negative edges**.

A new edge added in **active mode**
Preliminaries

Edges in partial query graph (positive edges)
- e6, e7, e8, e9

Edges rejected by users (negative edges)
- e4, e11, e12

Candidate edges
- e1, e2, e3, e5, e10

Query Session:
<(e6,yes), (e7,yes), (e8,yes), (e9,yes), (e4,no),
  (e11,no), (e12,no)>

represented as
(e6, e7, e8, e9, -e4, -e11, -e12)
## Query Log

- **Collection of several user sessions**

<table>
<thead>
<tr>
<th>Session Id</th>
<th>Co-related Edges</th>
</tr>
</thead>
<tbody>
<tr>
<td>$w_1$</td>
<td>$(education, yes), (founder, yes), (nationality, no)$</td>
</tr>
<tr>
<td>$w_2$</td>
<td>$(starring, yes), (music, no), (director, yes)$</td>
</tr>
<tr>
<td>$w_3$</td>
<td>$(nationality, yes), (education, no), (music, yes), (starring, no)$</td>
</tr>
<tr>
<td>$w_4$</td>
<td>$(artist, no), (title, no), (writer, yes), (director, yes)$</td>
</tr>
<tr>
<td>$w_5$</td>
<td>$(director, no), (founder, yes), (producer, yes)$</td>
</tr>
<tr>
<td>$w_6$</td>
<td>$(writer, yes), (editor, no), (genre, yes)$</td>
</tr>
<tr>
<td>$w_7$</td>
<td>$(award, no), (movie, yes), (director, yes), (genre, no)$</td>
</tr>
<tr>
<td>$w_8$</td>
<td>$(education, yes), (founder, yes), (nationality, no)$</td>
</tr>
</tbody>
</table>
Algorithms to Rank Candidate Edges

Possible Solutions

- Order alphabetically
- Use standard machine learning methods
  - Recommendation system
  - Association rule mining based classification
  - Classification: naïve Bayesian classifier, random forests
- Query-specific random correlation paths based suggestion
Random Correlation Paths (RCPs) Based Ranking

- Choose edges from the query session randomly to form RCPs:

- Grow a path incrementally until its support in the query log drops below a threshold (t).

- For each RCP, use its corresponding query log subset to compute support for each candidate edge.

Each correlation path selects a subset of the query log, with no more than ‘t’ rows in it.

Final score of each candidate is its average score across all RCPs.
### Preliminary Results

<table>
<thead>
<tr>
<th>Target Query Graphs</th>
<th># of edges</th>
<th>Edge Ranking Algorithms</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>RCP</td>
</tr>
<tr>
<td>ForrestGump-directorType</td>
<td>3</td>
<td>12</td>
</tr>
<tr>
<td>FilmType-directorType</td>
<td>5</td>
<td>39</td>
</tr>
<tr>
<td>DirectorType-actorType</td>
<td>3</td>
<td>&gt;100</td>
</tr>
<tr>
<td>FilmType-DirectorType</td>
<td>4</td>
<td>28</td>
</tr>
<tr>
<td>FilmType-DirectorType</td>
<td>3</td>
<td>14</td>
</tr>
<tr>
<td>FounderType-SchoolType</td>
<td>5</td>
<td>34</td>
</tr>
<tr>
<td>FounderType-SchoolType</td>
<td>4</td>
<td>&gt;100</td>
</tr>
<tr>
<td>JerryYang-SchoolType</td>
<td>5</td>
<td>34</td>
</tr>
<tr>
<td>JerryYang-Yahoo-Stanford</td>
<td>4</td>
<td>14</td>
</tr>
</tbody>
</table>
Evaluation Plan for Orion

➢ Compare with other standard machine learning algorithms
➢ User studies to gauge the effectiveness of our system and compare with naïve approaches like listing suggestions alphabetically
➢ Study effectiveness (number of suggestions required) using several simulated target query graphs
➢ Experiments with other datasets (DBpedia, YAGO)

Publication

➢ VIIQ: Auto-suggestion Enabled Visual Interface for Interactive Query Formulation, Nandish Jayaram, Sidharth Goyal, Chengkai Li, VLDB 2015, Demonstration description
Graph Query By Example (GQBE)
GQBE Interface (idir.uta.edu/gqbe)

- Ranked similar answer tuples
- Keyword completion powered query interface
- Query graph automatically discovered by the system

An example answer graph

Maximum Query Graph
Query Graph Discovery

Neighborhood Graph

Query Graph
Query Processing

Every other node is a sub-graph of the MQG.
Experiments: Accuracy Comparison with NESS and EQ

Dataset:
Freebase (47 million edges, 27 million nodes, 5.4 K edge labels)
Experiments: User Study with Amazon MTurk

<table>
<thead>
<tr>
<th>Query</th>
<th>PCC</th>
<th>Query</th>
<th>PCC</th>
<th>Query</th>
<th>PCC</th>
<th>Query</th>
<th>PCC</th>
</tr>
</thead>
<tbody>
<tr>
<td>F_1</td>
<td>0.79</td>
<td>F_2</td>
<td>0.78</td>
<td>F_3</td>
<td>0.60</td>
<td>F_4</td>
<td>0.80</td>
</tr>
<tr>
<td>F_5</td>
<td>0.34</td>
<td>F_6</td>
<td>0.27</td>
<td>F_7</td>
<td>0.06</td>
<td>F_8</td>
<td>0.26</td>
</tr>
<tr>
<td>F_9</td>
<td>0.33</td>
<td>F_10</td>
<td>0.77</td>
<td>F_11</td>
<td>0.58</td>
<td>F_12</td>
<td>undefined</td>
</tr>
<tr>
<td>F_13</td>
<td>undefined</td>
<td>F_14</td>
<td>0.62</td>
<td>F_15</td>
<td>0.43</td>
<td>F_16</td>
<td>0.29</td>
</tr>
<tr>
<td>F_17</td>
<td>0.64</td>
<td>F_18</td>
<td>0.30</td>
<td>F_19</td>
<td>0.40</td>
<td>F_20</td>
<td>0.65</td>
</tr>
</tbody>
</table>

**Pearson Correlation Coefficient (PCC) between GQBE and Amazon MTurk Workers, k=30**

- [0.5, 1.0] : Strong positive correlation
- [0.3, 0.5) : Medium positive correlation
- [0.1, 0.3) : Small positive correlation
Publications

- Querying Knowledge Graphs by Example Entity Tuples, Nandish Jayaram, Arijit Khan, Chengkai Li, Xifeng Yan, Ramez Elmasri, TKDE (to appear)

- GQBE: Querying Knowledge Graphs by Example Entity Tuples, Nandish Jayaram, Mahesh Gupta, Arijit Khan, Chengkai Li, Xifeng Yan, Ramez Elmasri, ICDE’14, Demonstration description

- Towards a Query-by-Example System for Knowledge Graphs, Nandish Jayaram, Arijit Khan, Chengkai Li, Xifeng Yan, Ramez Elmasri, GRAPES’14
Orion Demonstration at VLDB 2015

- Demo Session 3 (Kona 4)
- VIIQ: Auto-Suggestion Enabled Visual Interface for Interactive Graph Query Formulation

September 3rd, Wednesday (10:30 am to 12:00 pm)
September 4th, Thursday (3:30 pm to 5:00 pm)
Thank You!
nandish.jayaram@mavs.uta.edu
https://sites.google.com/site/jnandish
Multiple Example Tuples
Experiments: Efficiency Results

Single Query Execution Times (in seconds)

![Bar chart showing query processing times for different queries and methods.](chart.png)
Future Work
Future Work

➢ Comprehensive experiments and evaluation of Orion

➢ Evaluate the partial query graph at every iteration of the query formulation process in Orion

➢ User feedback loop after browsing the results
Cleaning Neighborhood Graph

- Neighborhood graphs can be large even for a small $d$, hundreds of thousands of edges and vertices!
- Clean some clearly unimportant edges.
Reduced Neighborhood Graph
Query Processing
Query Processing (cont.)
Query Processing (cont.)
Query Processing (cont.)
Evaluation Plan for Orion (cont.)

➢ Study effectiveness (number of suggestions required) using simulated target query graphs
➢ Experiments with other datasets (DBpedia, YAGO)
➢ Experiments to study effectiveness of simulated query log