Tackling Usability Challenges in Querying Massive, Ultra-heterogeneous Graphs

Chengkai Li
Associate Professor, Department of Computer Science and Engineering
Director, Innovative Database and Information Systems Research (IDIR) Laboratory
University of Texas at Arlington

North Carolina State University, Feb. 9th, 2016
Ultra-heterogeneous Entity Graphs

Large, complex and schema-less graphs capturing millions of entities and billions of relationships between entities.

Linked Open Data: 52 billion RDF triples
Freebase: 1.8 billion triples
DBpedia: 470 million triples
Yago: 120 million triples
Linked Open Data

http://linkeddata.org/ (hundreds of datasets, billions of RDF triples)
Structured Queries are Difficult to Write

**SQL QUERY:**
```
SELECT Founder.subj, Founder.obj
FROM Founder,
    Nationality,
    HeadquarteredIn
WHERE
    Founder.subj = Nationality.subj AND
    Founder.obj = HeadquarteredIn.subj
```

**SPARQL QUERY:**
```
SELECT ?company ?founder WHERE {
    ?company dbprop:headquartered_in db:Silicon_Valley .
}
```

- Require knowledge on data model, query language, and schema.
- Well-known usability challenges [Jagadish+07]
Simpler Query Paradigms

Keyword Search

- [Kargar+11], BLINKS [He+07]
  - Challenging to articulate exact query intent by keywords

Approximate Query Answering

- NESS [Khan+11]: uses neighborhood-based indexes to quickly find approximate matches to a query graph;
- TALE [Tian+08]: approximate large graph matching
  - Users still have to formulate the initial query graph
Visual Query Builders

**Relational Databases:** CLIDE [Petropoulos+06]

**Graph Databases:** VOGUE [Bhowmick+13], PRAGUE [Jin+12], Gblender [Jin+10], GRAPHITE [Chau+08]

**Single Large Graphs:** QUBLE [Hung+13]

- Require a good knowledge of the underlying schema
- No automatic suggestions regarding what to include in the query graph
An Example: QUBLE
Orion: Auto-Suggestion Based Visual Interface for Interactive Query Construction
Orion

Prototype
http://idir.uta.edu/orion

Introduction Video
http://bit.ly/1O0GnNo
Objectives of Orion

- Interactive GUI for building query components
- Iteratively suggest edges based on their relevance to the user’s query intent, according to the partial query graph so far
Orion GUI

Dynamic list of all possible user actions at any given moment

Control panel for various settings and tips
Passive Mode

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

- Accepted by user (blue): *positive edges*
- Ignored by user: *negative edges*
Active Mode

A new node added in active mode

A new edge added in active mode
Rank Candidate Edges for Suggestions

Possible Solutions

- Order alphabetically
- Adapt standard machine learning algorithms
  - Naïve Bayes classifier
  - Random forests
  - Class association rules
  - Recommendation systems (based on SVD)

Query-specific Random Decision Paths (RDP)
Concepts

Edges in partial query graph (positive edges)
- starring, director, music

Edges rejected by users (negative edges)
- education, nationality

Candidate edges
- producer, writer, editor

Query Session:
<br><br>&lt;starring, director, music, education, nationality&gt;
Concepts

Query Log (W)

<table>
<thead>
<tr>
<th>Id</th>
<th>Query Session</th>
</tr>
</thead>
<tbody>
<tr>
<td>$w_1$</td>
<td>education, founder, nationality</td>
</tr>
<tr>
<td>$w_2$</td>
<td>starring, music, director</td>
</tr>
<tr>
<td>$w_3$</td>
<td>nationality, education, music, starring</td>
</tr>
<tr>
<td>$w_4$</td>
<td>artist, title, writer, director</td>
</tr>
<tr>
<td>$w_5$</td>
<td>director, founder, producer</td>
</tr>
<tr>
<td>$w_6$</td>
<td>writer, editor, genre</td>
</tr>
<tr>
<td>$w_7$</td>
<td>award, movie, director, genre</td>
</tr>
<tr>
<td>$w_8$</td>
<td>education, founder, nationality</td>
</tr>
</tbody>
</table>

Problem

Given a query log, a query session, and a set of candidate edges, rank the candidate edges by their relevance to the user’s query intent.
Random Decision Path (RDP)

- Choose edges from the query session randomly, to form RDPs

- Each decision path selects a subset of the query log, with no more than ‘τ’ rows

- Grow a path incrementally until its support in the query log drops below ‘τ’
Random Decision Path: Scoring

- For each RDP, use its corresponding query log subset to compute the support of each candidate edge.
- Final score of each candidate is its average score across all RDPs.
- If R is the set of all RDPs:

\[
\text{score}(e) = \frac{1}{|R|} \sum_{Q_i \in R} \text{sup}(e, Q_i, W)
\]

\[
\text{sup}(e, Q_i, W) = \frac{|\{w \mid w \in W, Q_i \cup \{e\} \subseteq w\}|}{|\{w \mid w \in W, Q_i \subseteq w\}|}
\]
Query Log

Nonexistent (almost)

Simulate and bootstrap

- Find positive edges
  - Wikipedia and data graph
  - Data graph only
- SPARQL query log [Morsey+11]
- Inject negative edges
Query Log Simulation: Wikipedia + Data Graph

Use Sentences in Wikipedia Articles to Identify Positive Edges

Yang was born in Taipei, Taiwan on November 6, 1968, and moved to San Jose, California at the age of ten with his mother and younger brother. He claimed that despite his mother being an English teacher, he only knew one English word (shoe) on his arrival. Becoming fluent in the language in three years, he was then placed into an Advanced Placement English class. Yang graduated from Sierramont Middle School and Piedmont Hills High School in San Jose and went on to earn a Bachelor of Science and a Master of Science in electrical engineering from Stanford University, where he was a member of Phi Kappa Psi fraternity.

\langle\text{degree}, \text{almaMater}, \text{frat\_member}\rangle
Query Log Simulation: Data Graph Only

Represent Each Node as an Itemset of Positive Edges

Inject Negative Edges Based on Edge Types

Generate Frequent Itemsets of Varying Sizes

- Each frequent itemset of edges forms positive edges
Query Log Simulation: Injecting Negative Edges

Positive Edges List

(1) writer, starring, producer
(2) starring, editor, education
(3) editor, nationality, music

Inject Negative Edges

writer, starring, producer, editor, education (starring appears in 2)

starring, editor, education, writer, producer, nationality, music (starring appears in 1, and editor appears in 3)

editor, nationality, music, starring, education (editor appears in 2)
Experiments

System Configurations

- Double quad-core 2.0 GHz Xeon server, 24 GB RAM
- TACC: 5 Dell PowerEdge R910 server nodes, with 4 Intel Xeon E7540 2.0 GHz 6-core processors, 1 TB RAM

Datasets

- Freebase (33 M edges, 30 M nodes, 5253 edge types)
- DBpedia (12 M edges, 4 M nodes, 647 edge types)
Experiments (cont.)

User Studies

- Orion (RDP) and Naïve

Edge Ranking Algorithms Compared

- Random Decision Paths (RDP)
- Naïve Bayes classifier (NB)
- Random forest classifier (RF)
- Class association rules (CAR)
- SVD based recommendation system (SVD)
Query Logs Compared

- Freebase: Wiki, Data
- DBpedia: Wiki, Data, QLog

<table>
<thead>
<tr>
<th>Query Log</th>
<th>Components Used in Query Log Simulation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Freebase</td>
</tr>
<tr>
<td>Wiki-FB</td>
<td>Yes</td>
</tr>
<tr>
<td>Data-FB</td>
<td>Yes</td>
</tr>
<tr>
<td>Wiki-DB</td>
<td>-</td>
</tr>
<tr>
<td>Data-DB</td>
<td>-</td>
</tr>
<tr>
<td>QLog-DB</td>
<td>-</td>
</tr>
</tbody>
</table>
User Studies: Setup

15 Users for Orion, 15 Users for Naïve (A/B testing)
105 Query Tasks per System
45 Easy, 30 Medium, and 30 Hard Query Tasks
4 Survey Questions per User

<table>
<thead>
<tr>
<th>Likert Scale Score</th>
<th>Q1: How well do you think the query graph formulated by you captures the required query intent?</th>
<th>Q2: How easy was it to use the interface for formulating this query?</th>
<th>Q3: How satisfactory was the overall experience?</th>
<th>Q4: The interface provide features necessary for easily formulating query graphs.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Very Poorly</td>
<td>Very Hard</td>
<td>Unacceptable</td>
<td>Strongly Disagree</td>
</tr>
<tr>
<td>2</td>
<td>Poorly</td>
<td>Hard</td>
<td>Poor</td>
<td>Disagree</td>
</tr>
<tr>
<td>3</td>
<td>Adequately</td>
<td>Neither Easy Nor Hard</td>
<td>Satisfactory</td>
<td>Uncertain</td>
</tr>
<tr>
<td>4</td>
<td>Well</td>
<td>Easy</td>
<td>Good</td>
<td>Agree</td>
</tr>
<tr>
<td>5</td>
<td>Very Well</td>
<td>Very Easy</td>
<td>Excellent</td>
<td>Strongly Agree</td>
</tr>
</tbody>
</table>
### User Studies: Conversion Rate

**Conversion Rate:**

- Percentage of query tasks completed successfully
- Successful completion measured using edge isomorphism

<table>
<thead>
<tr>
<th>System</th>
<th>Queries</th>
<th>Sample Size</th>
<th>Conversion Rate ($c$)</th>
<th>z-value</th>
<th>p-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Orion</td>
<td>All</td>
<td>105</td>
<td>$c_O=0.74$, $c_N=0.68$</td>
<td>0.92</td>
<td>0.1788</td>
</tr>
<tr>
<td>Naïve</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Orion</td>
<td>Medium + Hard</td>
<td>60</td>
<td>$c_O=0.70$, $c_N=0.58$</td>
<td>1.36</td>
<td>0.0869</td>
</tr>
<tr>
<td>Naïve</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Orion has a higher success rate than Naïve for complex queries!
Time required to construct query graphs in Orion is comparable to Naïve in most cases, despite the steeper learning curve of Orion due to more features.
User Studies: Efficiency by Number of Iterations

Time required to construct query graphs in Orion is comparable to Naïve in most cases, despite the steeper learning curve of Orion due to more features.
User Studies: User Experience Results

As the difficulty level of the query graph being constructed increases, the usability of Orion seems significantly better than Naïve’s.
Edge Ranking Algorithms

- Simulates only Passive Mode
- 43 target query graphs
  - 6 two-edged, 10 three-edged, 9 four-edged, 17 five-edged, 1 sixe-edged
- 167 Input Instances for Freebase
- 130 Input Instances for DBpedia
Edge Ranking Algorithms: Efficiency by Number of Suggestions

RDP requires only 40 suggestions, 1.5-4 times fewer than other methods

RDP requires fewer suggestions compared to all other methods
Edge Ranking Algorithms: Efficiency by Time

RDP better than RF and comparable to NB, despite RF and NB being light models

RDP significantly better than SVD and CAR, but worse than RF and NB
Query Logs Comparison

Positive edges better captured based on the context of human usage of relationships in Wikipedia.

DBpedia is created using info-boxes in Wikipedia, and is thus very clean. Wiki-DB is highly similar to Data-DB for DBpedia.
Tuning RDP Parameters

RDP performs better with more random decision paths and higher query log threshold

Considering negative edges in query session is important, as it results in better performance of RDP
Orion Contributions

Unique visual query builder with suggestions

Edge ranking algorithm: random decision paths

Query log simulation

Extensive user studies and experiments

Prototype available
GQBE: Graph Query by Example

Prototype
http://idir.uta.edu/gqbe

Introduction Video
http://bit.ly/1PLqLTD
GQBE GUI

Ranked similar answer tuples

Keyword completion powered query interface

Query graph automatically discovered by the system

An example answer graph
TableView: Generating Preview Tables for Entity Graphs
Figure 1: An Excerpt of an Entity Graph.

Figure 3: The Schema Graph for the Entity Graph in Fig. 1.
### Table View

<table>
<thead>
<tr>
<th>FILM</th>
<th>Director</th>
<th>Genres</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_1$</td>
<td>Men in Black</td>
<td>Barry Sonnenfeld</td>
</tr>
<tr>
<td></td>
<td></td>
<td>{Action Film, Science Fiction}</td>
</tr>
<tr>
<td>$t_2$</td>
<td>Men in Black II</td>
<td>Barry Sonnenfeld</td>
</tr>
<tr>
<td></td>
<td></td>
<td>{Action Film, Science Fiction}</td>
</tr>
<tr>
<td>$t_3$</td>
<td>Hancock</td>
<td>Peter Berg</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$t_4$</td>
<td>I, Robot</td>
<td>Alex Proyas</td>
</tr>
<tr>
<td></td>
<td></td>
<td>{Action Film}</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>FILM ACTOR</th>
<th>Award Winners</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_5$</td>
<td>Will Smith</td>
</tr>
<tr>
<td></td>
<td>Saturn Award</td>
</tr>
<tr>
<td>$t_6$</td>
<td>Tommy Lee Jones</td>
</tr>
<tr>
<td></td>
<td>Academy Award</td>
</tr>
</tbody>
</table>

**Figure 2:** A 2-Table Preview of the Entity Graph in Fig. 1. (Upper and lower tables for subgraphs #1 and #2 in Fig. 3, respectively.)
Publications

Overview

- Intuitive and Interactive Query Formulation to Improve the Usability of Query Systems for Heterogeneous Graphs. Nandish Jayaram. VLDB 2015 PhD Workshop.

Orion


GQBE

- Querying knowledge graphs by example entity tuples (Extended Abstract). Nandish Jayaram, Arijit Khan, Chengkai Li, Xifeng Yan, Ramez Elmasri. ICDE 2016 (TKDE Poster)
Publications

GQBE (cont’d)

- Querying knowledge graphs by example entity tuples. Nandish Jayaram, Arijit Khan, Chengkai Li, Xifeng Yan, Ramez Elmasri. TKDE, 27(10): 2797-2811, October 2015.

- GQBE: Querying knowledge graphs by example entity tuples. Nandish Jayaram, Mahesh Gupta, Arijit Khan, Chengkai Li, Xifeng Yan, Ramez Elmasri. ICDE, pages 1250-1253, 2014. Demonstration description.

- Towards a query-by-example system for knowledge graphs. Nandish Jayaram, Arijit Khan, Chengkai Li, Xifeng Yan, Ramez Elmasri. GRADES 2014.

TableView

Acknowledgment

Funding sponsors

Disclaimer: This material is based upon work partially supported by the National Science Foundation Grants 1018865, 1117369, 1408928 and 1565699, 2011 and 2012 HP Labs Innovation Research Awards, the National Natural Science Foundation of China Grant 61370019, and Army Research Laboratory under cooperative agreement W911NF-09-2-0053 (NSCTA). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the funding agencies.
Acknowledgment

UTA Students
- Nandish Jayaram
- Sona Hasani
- Rohit Bhoopalam
- Sidharth Goyal
- Mahesh Gupta

Collaborators
- Arijit Khan (NTU)
- Ramez Elmasri (UTA)
- Ning Yan (FutureWei, former student)
- Xifeng Yan (UCSB)
Thank You! Questions?

- http://ranger.uta.edu/~cli
  cli@uta.edu

- Prototypes
  http://idir.uta.edu/orion
  http://idir.uta.edu/gqbe