Orion: Enabling Suggestions in a Visual Query Builder for Ultra-Heterogeneous Graphs

Nandish Jayaram
Member of Technical Staff 3
Pivotal
(work done as a student at the University of Texas at Arlington)

Rohit Bhoopalam
Chengkai Li
Vassilis Athitsos

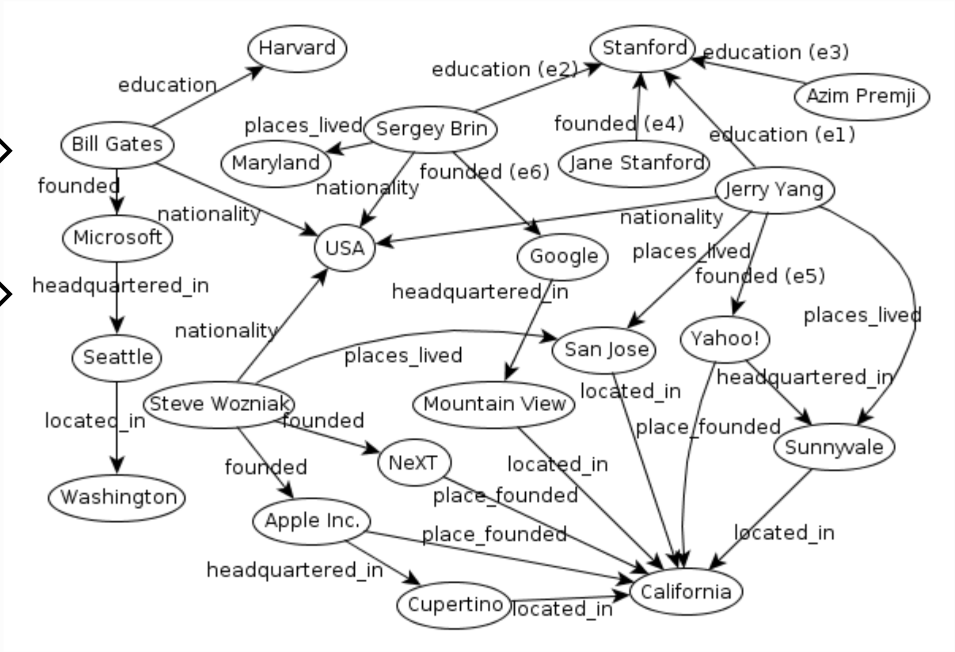
Eighth TUC Meeting , June 22-23, 2016

Ultra-heterogeneous Entity Graphs

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

entity

relationship



Linked Open Data : 52 billion RDF triples

Freebase : 1.8 billion triples

DBpedia : 470 million triples

Yago : 120 million triples

Structured Queries are Difficult to Write

SQL QUERY:

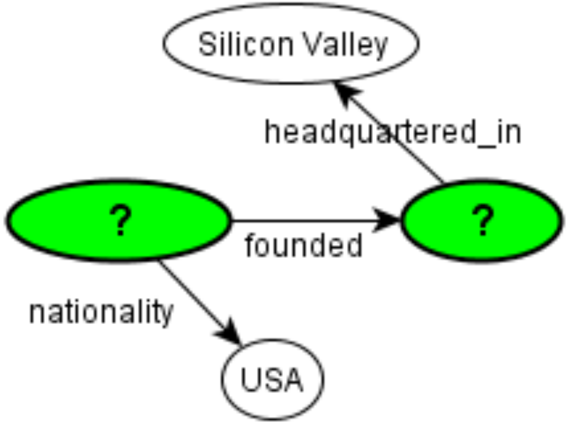
```
SELECT  Founder.subj, Founder.obj
FROM Founder,
   Nationality,
   HeadquarteredIn
WHERE
   Founder.subj = Nationality.subj AND
   Founder.obj = HeadquarteredIn.subj
```

- Require knowledge on data model, query language, and schema.

- Well-known usability challenges [Jagadish+07]

SPARQL QUERY:

```
SELECT   ?company   ?founder   WHERE {
   ?founder  dbo:founded  ?company .
   ?founder  dbo:nationality  db:United_States .
   ?company  dbprop:headquartered_in  db:Silicon_Valley .
}
```



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

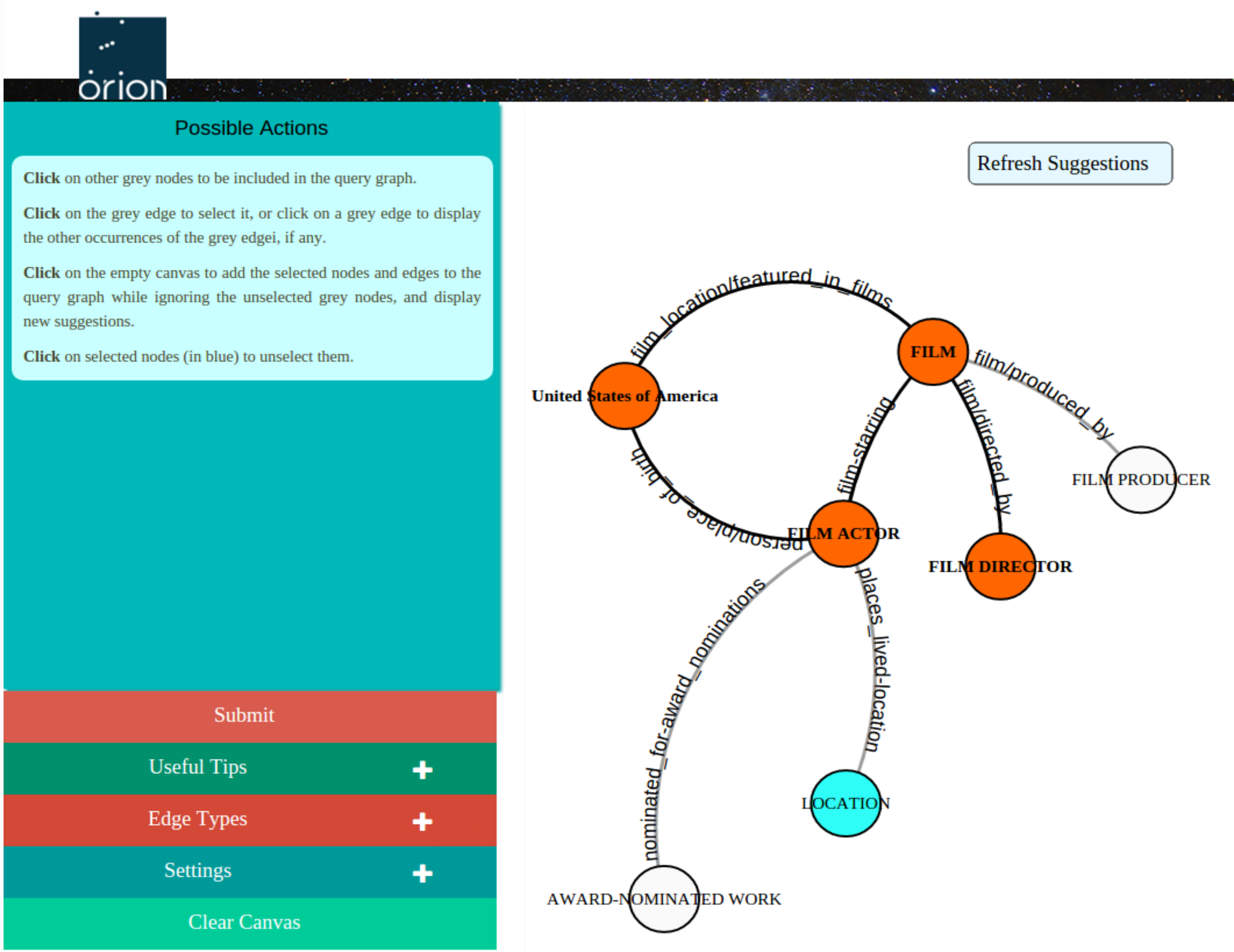
Orion

- Interactive GUI for building query components
- Iteratively suggests 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



Possible Actions

Click on other grey nodes to be included in the query graph.

Click on the grey edge to select it, or click on a grey edge to display the other occurrences of the grey edgei, if any.

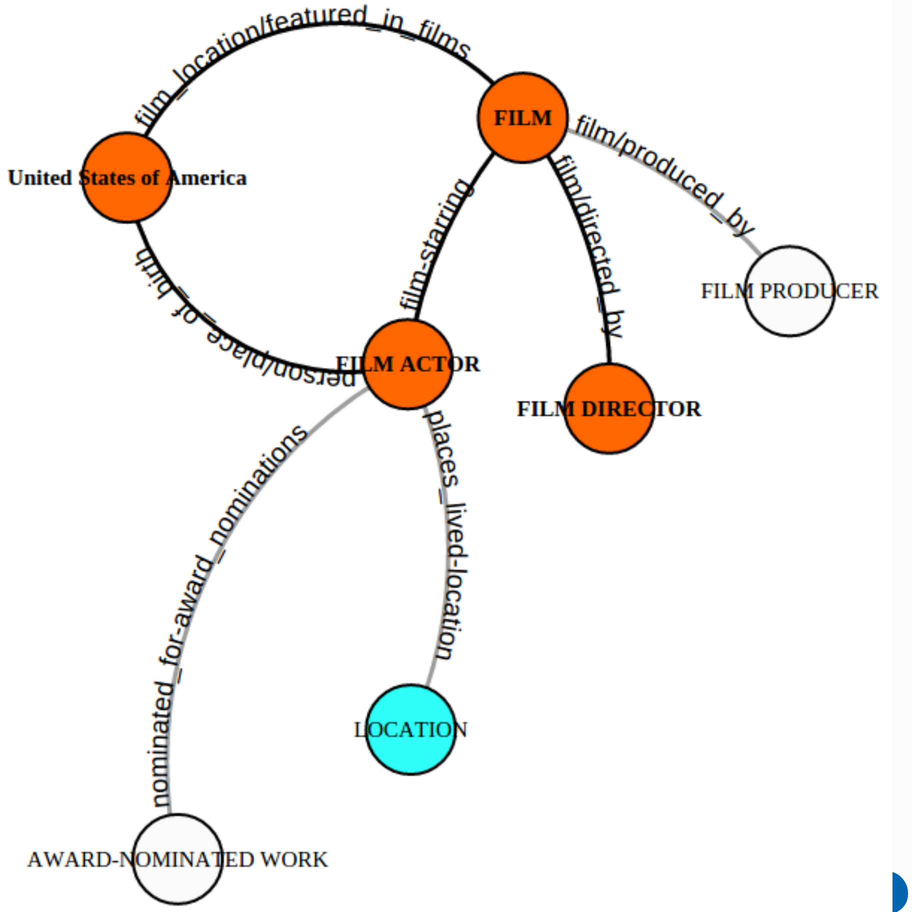
Click on the empty canvas to add the selected nodes and edges to the query graph while ignoring the unselected grey nodes, and display new suggestions.

Click on selected nodes (in blue) to unselect them.

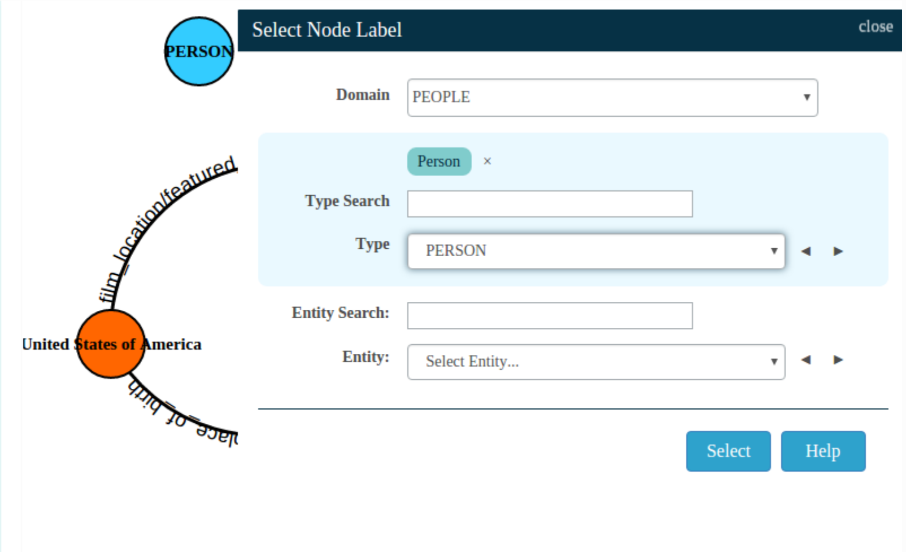
Active Mode

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

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



Passive Mode



Select Node Label

Domain: PEOPLE

Person ×

Type Search:

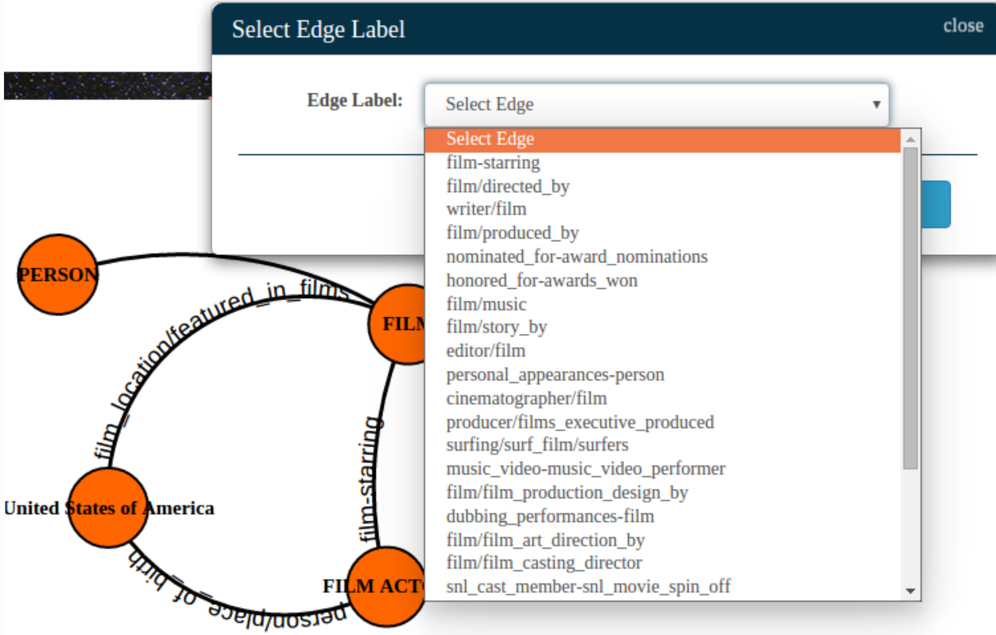
Type: PERSON

Entity Search:

Entity: Select Entity...

A new node added in **passive mode**

A new edge added in **passive mode**



Select Edge Label

Edge Label: Select Edge

close
Select Edge
film-starring
film/directed_by
writer/film
film/produced_by
nominated_for-award_nominations
honored_for-awards_won
film/music
film/story_by
editor/film
personal_appearances-person
cinematographer/film
producer/films_executive_produced
surfing/surf_film/surfers
music_video-music_video_performer
film/film_production_design_by
dubbing_performances-film
film/film_art_direction_by
film/film_casting_director
snl_cast_member-snl_movie_spin_off
PERSON
FILM
FILM ACT
United States of America
film_location/featured_in_films
film-starring
person/place_of_birth

Concepts

Edges in partial query graph (positive edges)

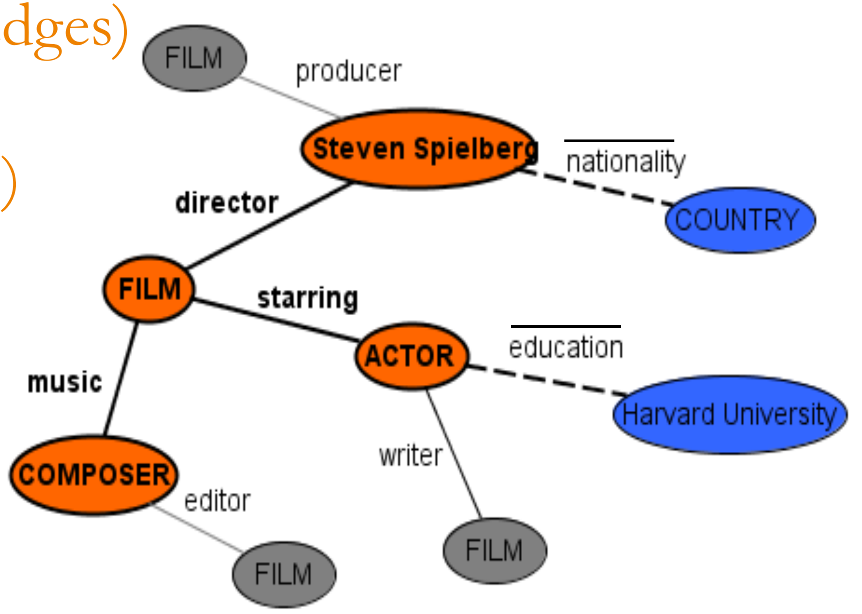
`starring, director, music`

Edges rejected by users (negative edges)

`education, nationality`

Candidate edges

`producer, writer, editor`

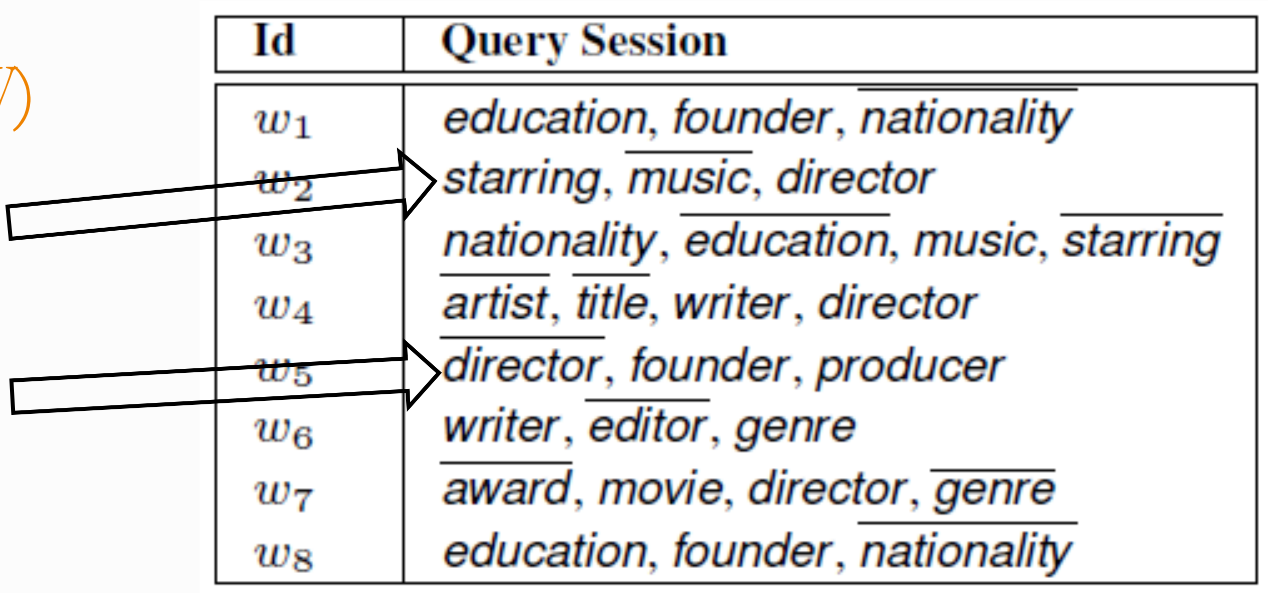


Query Session:

`<starring, director, music, education, nationality>`

Concepts

Query Log (W)



Positive Edge

Negative Edge

Id	Query Session
w_1	*education*, *founder*, $\overline{\textit{nationality}}$
w_2	*starring*, $\overline{\textit{music}}$, *director*
w_3	*nationality*, $\overline{\textit{education}}$, *music*, $\overline{\textit{starring}}$
w_4	$\overline{\textit{artist}}$, $\overline{\textit{title}}$, *writer*, *director*
w_5	$\overline{\textit{director}}$, *founder*, *producer*
w_6	*writer*, $\overline{\textit{editor}}$, *genre*
w_7	$\overline{\textit{award}}$, *movie*, *director*, $\overline{\textit{genre}}$
w_8	*education*, *founder*, $\overline{\textit{nationality}}$

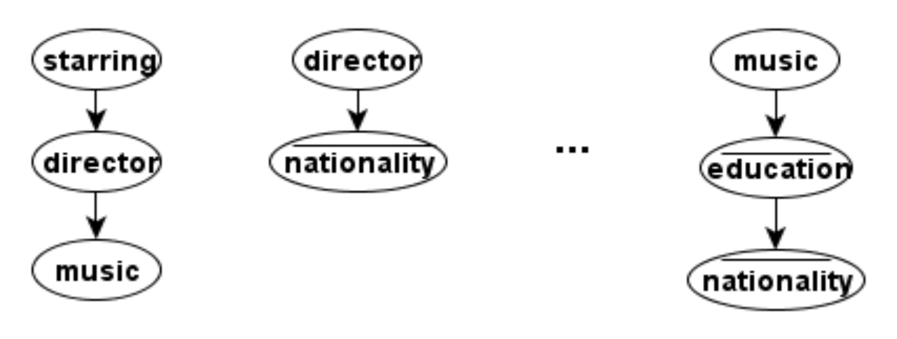
Problem

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)

<starring, director, music, $\overline{\text{education}}$, $\overline{\text{nationality}}$>

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


starring
director
music
director
$\overline{\text{nationality}}$
…
music
$\overline{\text{education}}$
$\overline{\text{nationality}}$

- 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:

$$score(e) = \frac{1}{|R|} * \sum_{Qi \in R} \sup(e, Q_i, W)$$

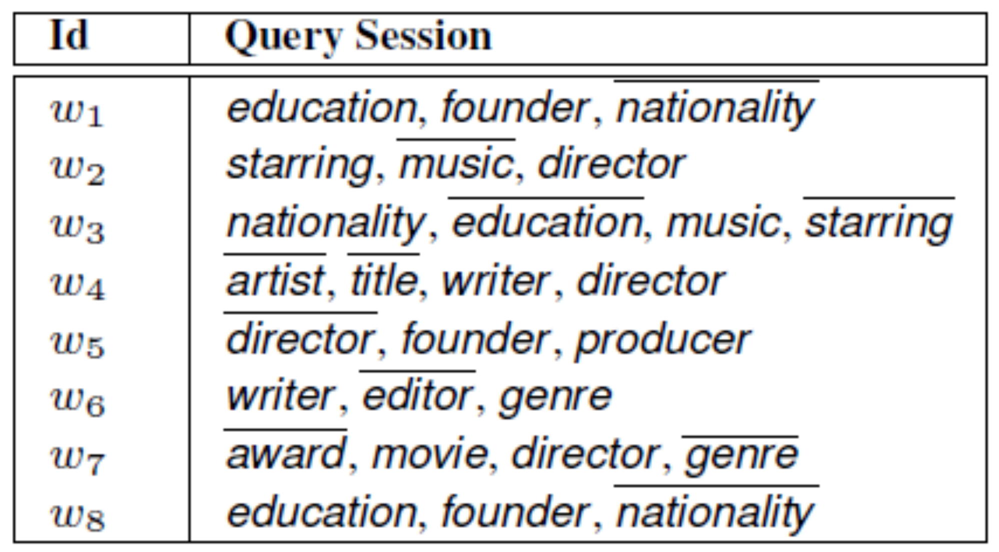
$$\sup(e, Q_i, W) = \frac{|\{w \mid w \in W, Q_i \cup \{e\} \subseteq w\}|}{|\{w \mid w \in W, Q_i \subseteq w\}|}$$

Query Log

Nonexistent (almost)

Simulate and bootstrap

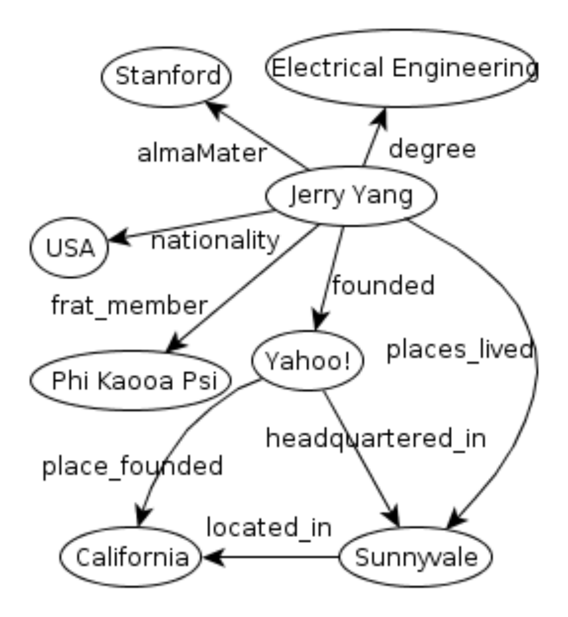
- Find positive edges
  - Wikipedia and data graph
  - Data graph only
  - SPARQL query log [Morsey+11]
- Inject negative edges

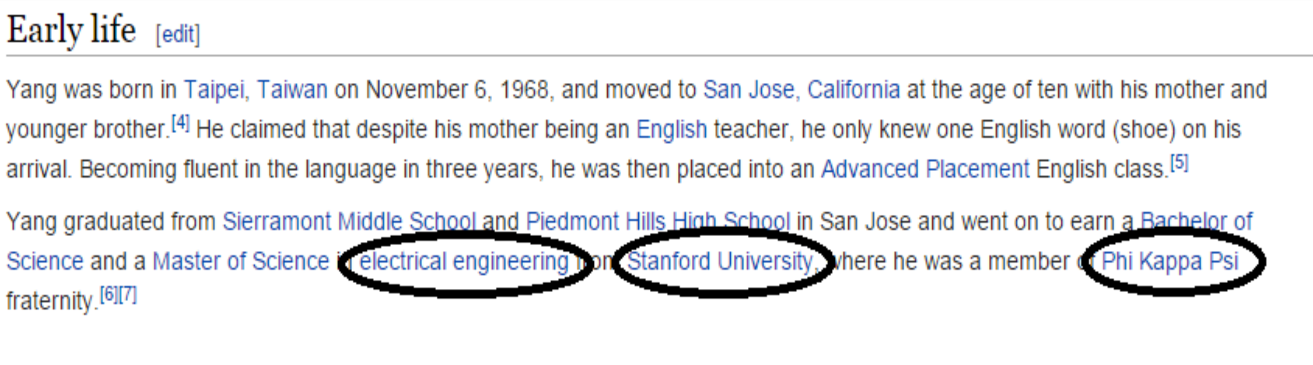


Id	Query Session
w_1	*education*, *founder*, $\overline{\textit{nationality}}$
w_2	*starring*, $\overline{\textit{music}}$, *director*
w_3	*nationality*, $\overline{\textit{education}}$, *music*, $\overline{\textit{starring}}$
w_4	$\overline{\textit{artist}}$, $\overline{\textit{title}}$, *writer*, *director*
w_5	$\overline{\textit{director}}$, *founder*, *producer*
w_6	*writer*, $\overline{\textit{editor}}$, *genre*
w_7	$\overline{\textit{award}}$, *movie*, *director*, $\overline{\textit{genre}}$
w_8	*education*, *founder*, $\overline{\textit{nationality}}$

Query Log Simulation: Wikipedia + Data Graph

Use Sentences in Wikipedia Articles to Identify Positive Edges





Early life [edit]

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.[4] 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.[5]

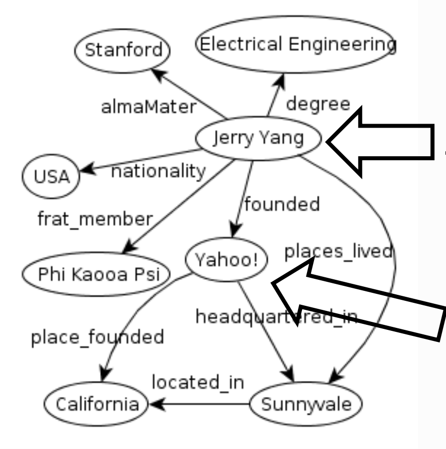
Yang graduated from Sierramont Middle School and Piedmont Hills High School in San Jose and went on to earn a Bachelor of Science and a Master of Science in electrical engineering from Stanford University, where he was a member of Phi Kappa Psi fraternity.[6][7]

Nodes Mapped: **Jerry Yang,** Electrical Engineering, Stanford University, Phi Kappa Psi

`<degree, almaMater, frat_member>`

Query Log Simulation: Data Graph Only

Represent Each Node as an Itemset of Positive Edges



<degree, almaMater, nationality, frat_member, founded, places_lived>

<founded, place_founded, headquartered_in>

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, $\overline{\texttt{editor}}$, $\overline{\texttt{education}}$ (starring appears in 2)

starring, editor, education, $\overline{\texttt{writer}}$, $\overline{\texttt{producer}}$, $\overline{\texttt{nationality}}$, $\overline{\texttt{music}}$ (starring appears in 1, and editor appears in 3)

editor, nationality, music, $\overline{\texttt{starring}}$, $\overline{\texttt{education}}$ (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

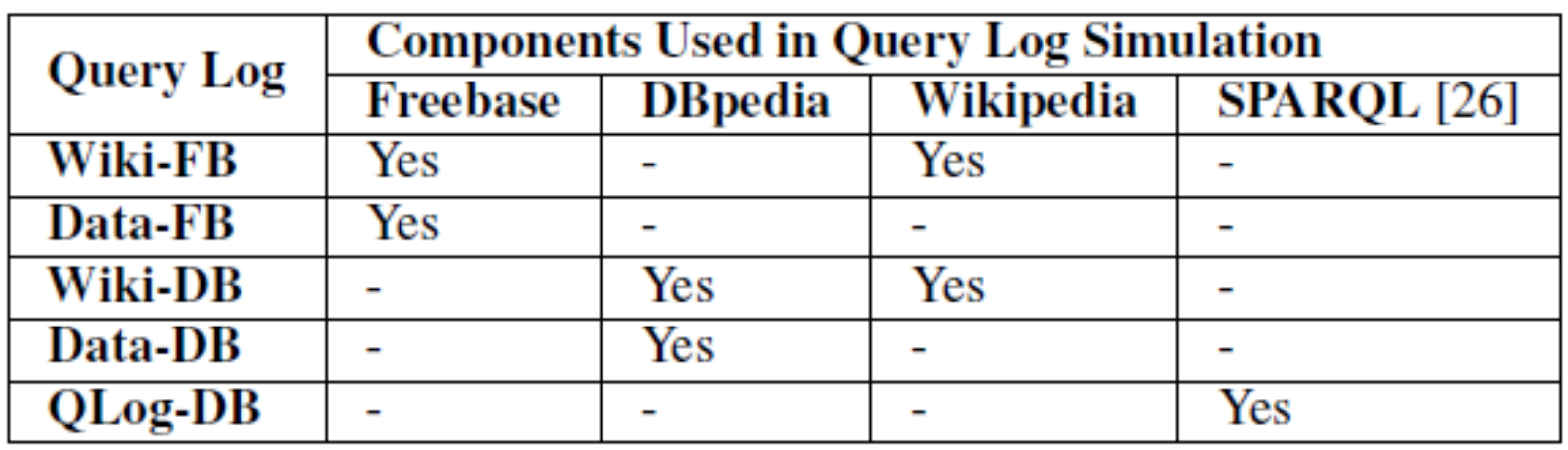
Datasets

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

User Studies with Freebase

Query Logs Compared

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



Query Log	Components Used in Query Log Simulation			
	Freebase	DBpedia	Wikipedia	SPARQL [26]
Wiki-FB	Yes	-	Yes	-
Data-FB	Yes	-	-	-
Wiki-DB	-	Yes	Yes	-
Data-DB	-	Yes	-	-
QLog-DB	-	-	-	Yes

User Studies: Setup

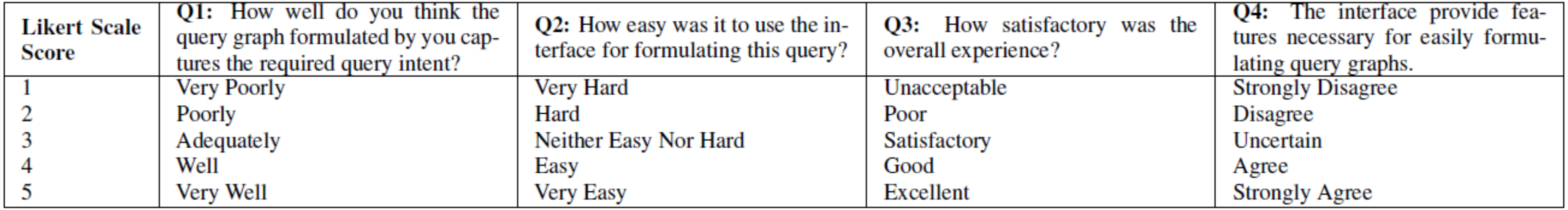
15 Users for Orion, 15 Users for Naïve (A/B testing)

45 Easy, 30 Medium, and 30 Hard Query Tasks Designed

3 Easy, 2 Medium, 2 Hard Queries Assigned per Query Task

105 Query Tasks per System in Total

4 Survey Questions per Query Task

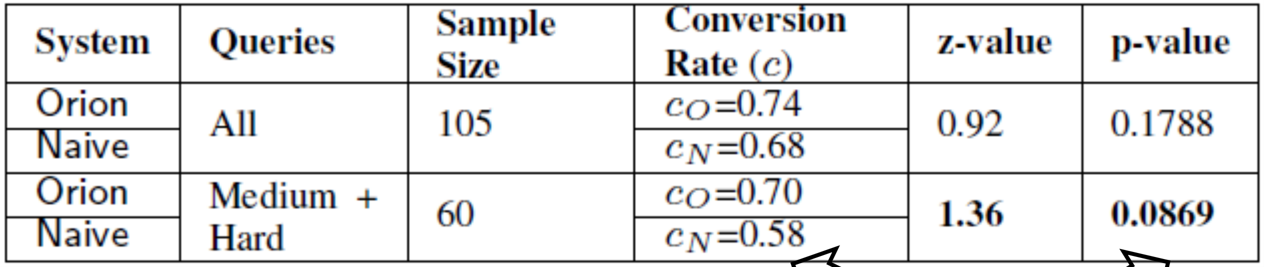


Likert Scale Score	**Q1:** How well do you think the query graph formulated by you captures the required query intent?	**Q2:** How easy was it to use the interface for formulating this query?	**Q3:** How satisfactory was the overall experience?	**Q4:** The interface provide features necessary for easily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

User Studies: Conversion Rate

Conversion Rate:

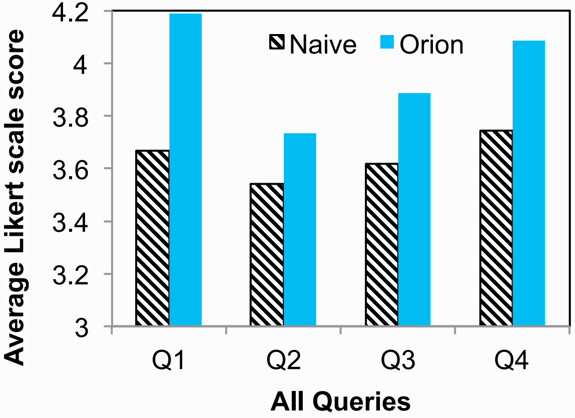
- Percentage of query tasks completed successfully
- Successful completion measured using edge isomorphism, and not a binary notion of matching

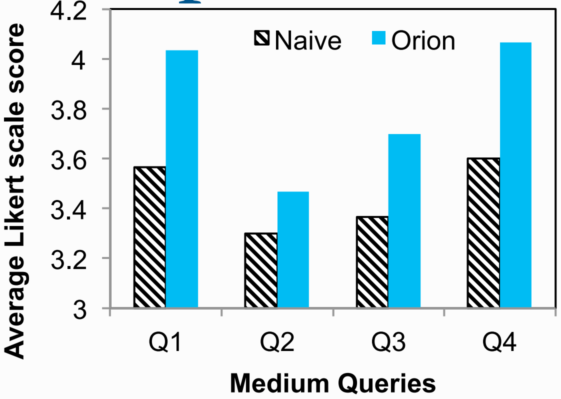


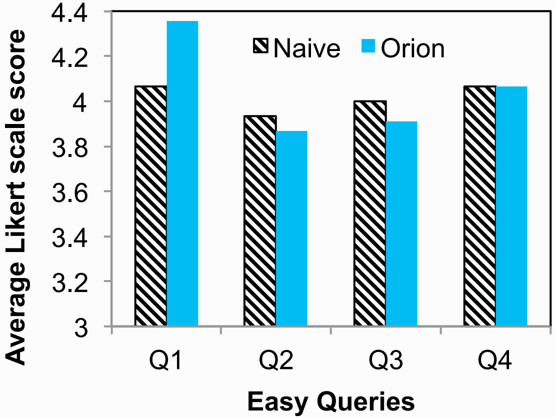
System	Queries	Sample Size	Conversion Rate (c)	z-value	p-value
Orion	All	105	c_O=0.74	0.92	0.1788
Naive			c_N=0.68		
Orion	Medium + Hard	60	c_O=0.70	**1.36**	**0.0869**
Naive			c_N=0.58		

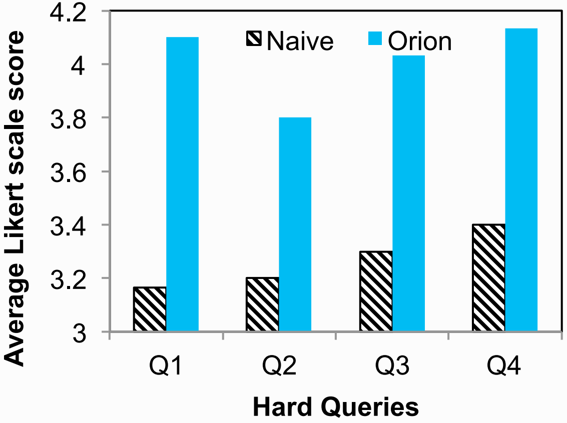
Orion has a higher conversion rate than Naïve for complex queries!

User Studies: User Experience Results



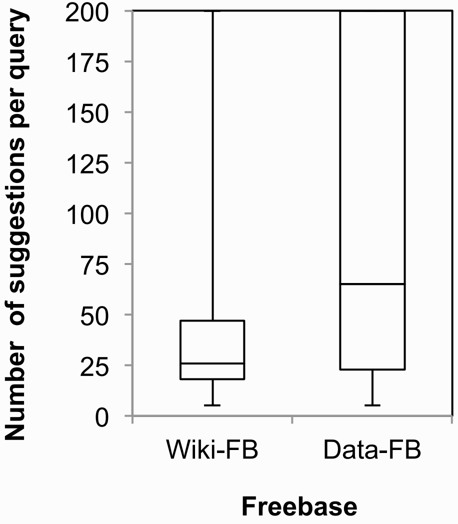


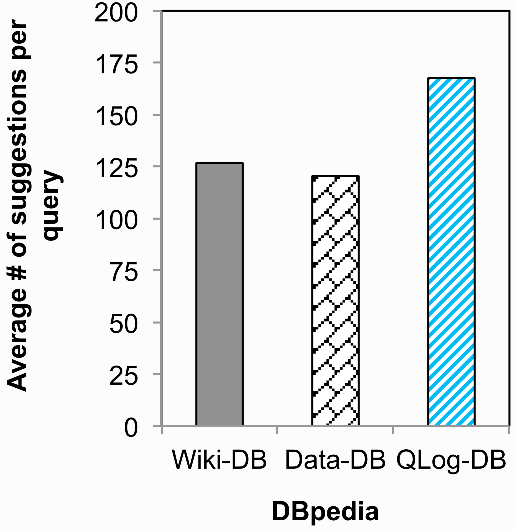




As the difficulty level of the query graph being constructed increases, the usability of Orion seems significantly better than Naïve's

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

Challenges for the LDBC Community

A benchmark query log to help improve the performance of systems such as Orion

A benchmark query set for visual query formulation, for better evaluation of systems

Orion

Prototype

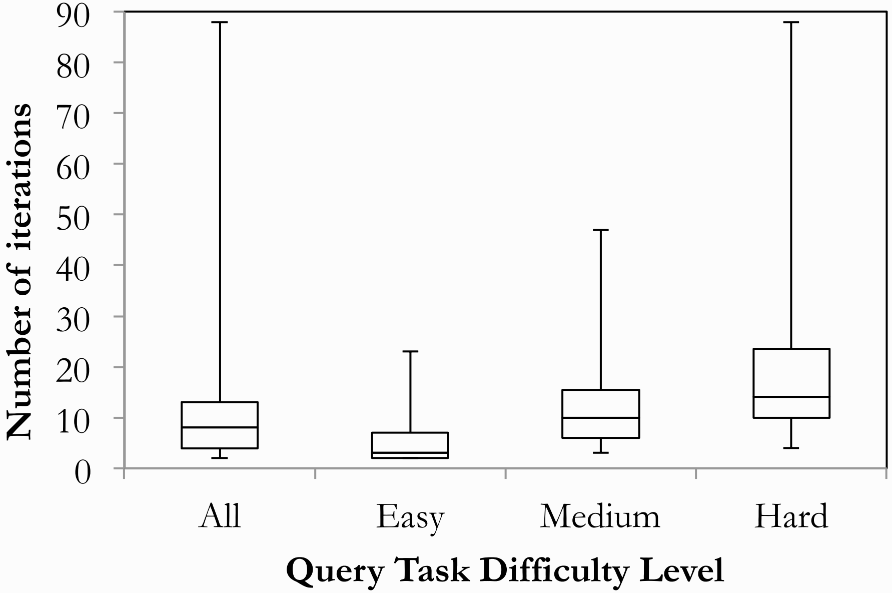
http://idir.uta.edu/orion

Introduction Video

http://bit.ly/1O0GnNo

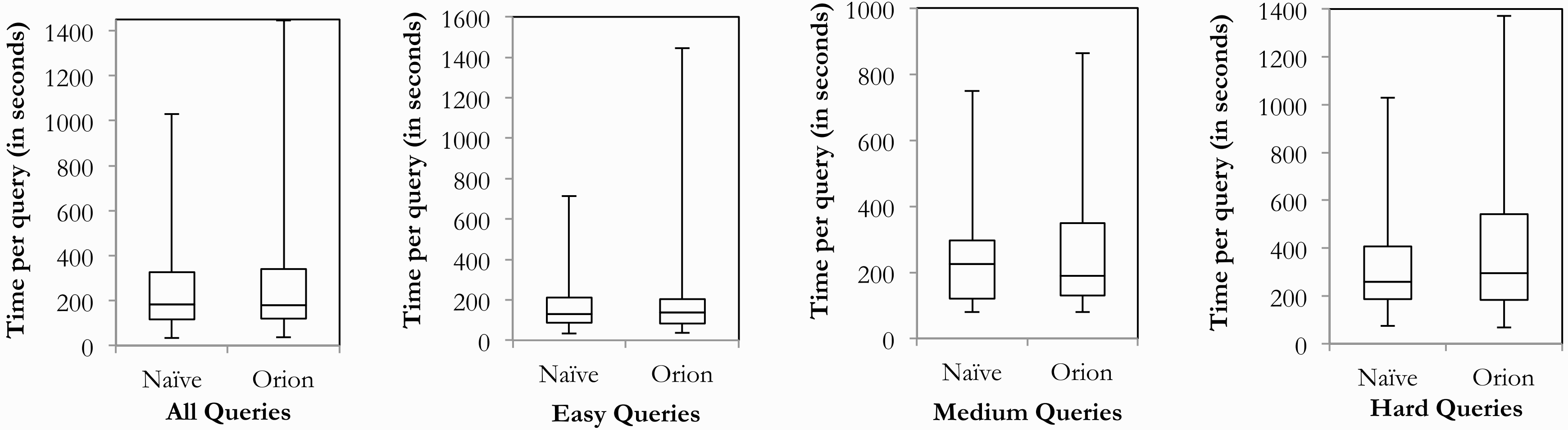
Thank You! Questions?

User Studies: Efficiency by Number of Iterations



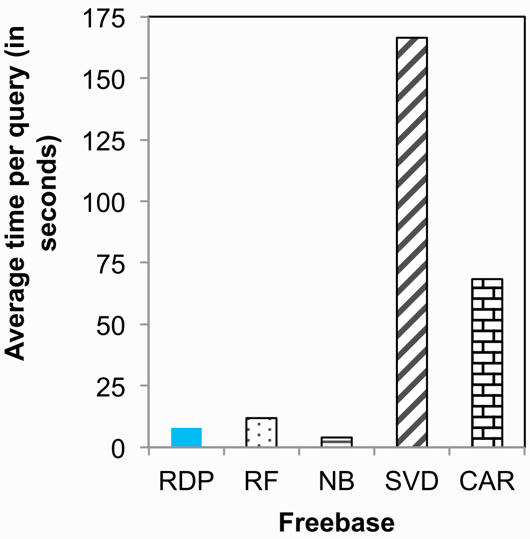
Time required to construct query graphs in Orion is comparable to Naïve in most cases, despite the steeper learning curve of Orion due to more features

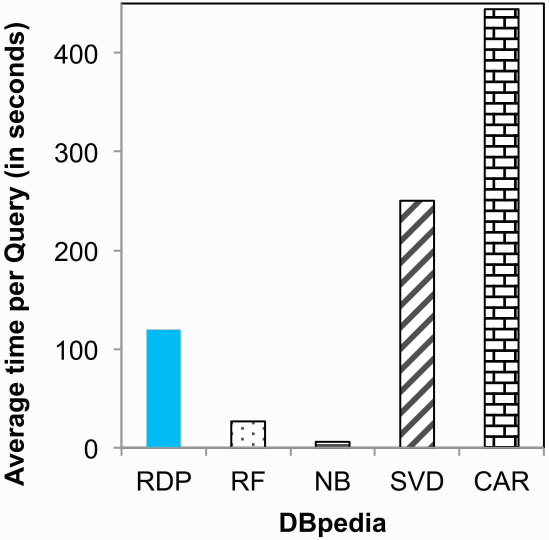
User Studies: Efficiency by Time



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

Edge Ranking Algorithms: Efficiency by Time





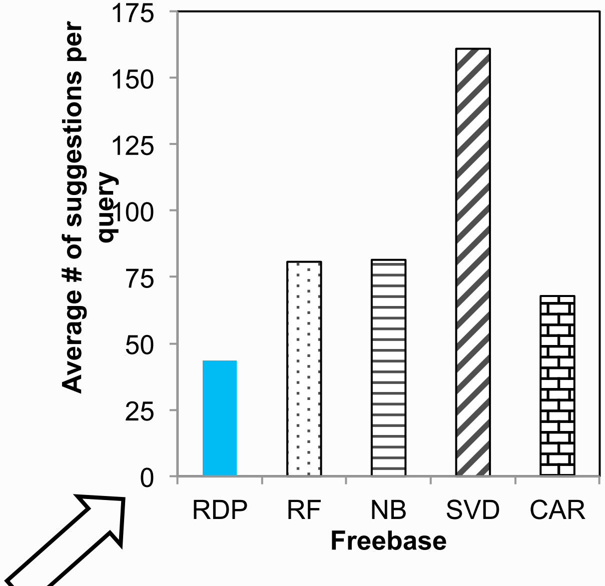
RDP better than RF and comparable to NB, despite RF and NB being light models

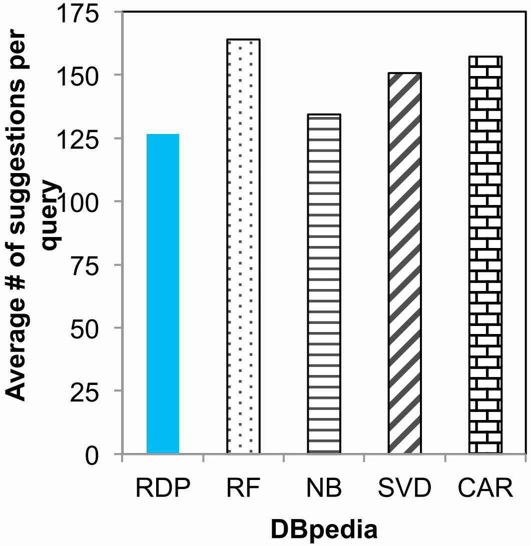
RDP significantly better than SVD and CAR, but worse than RF and NB

Edge Ranking Algorithms

- Simulates only Active Mode
- 43 target query graphs for Freebase
  - 6 two-edged, 10 three-edged, 9 four-edged, 17 five-edged, 1 six-edged (includes medium and hard queries from the user study)
  - 167 input instances
- 33 target query graphs for DBpedia
  - 2 three-edged, 29 four-edged, 2 five-edged
  - 130 input instances

Edge Ranking Algorithms: Efficiency by Number of Suggestions

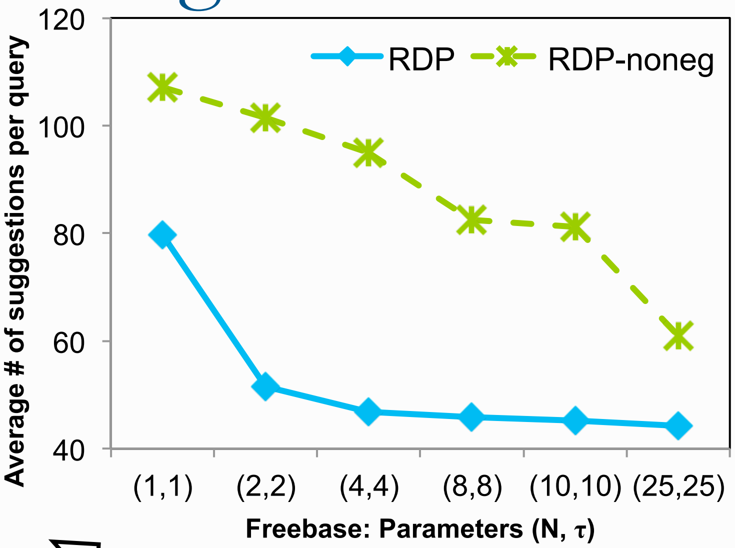


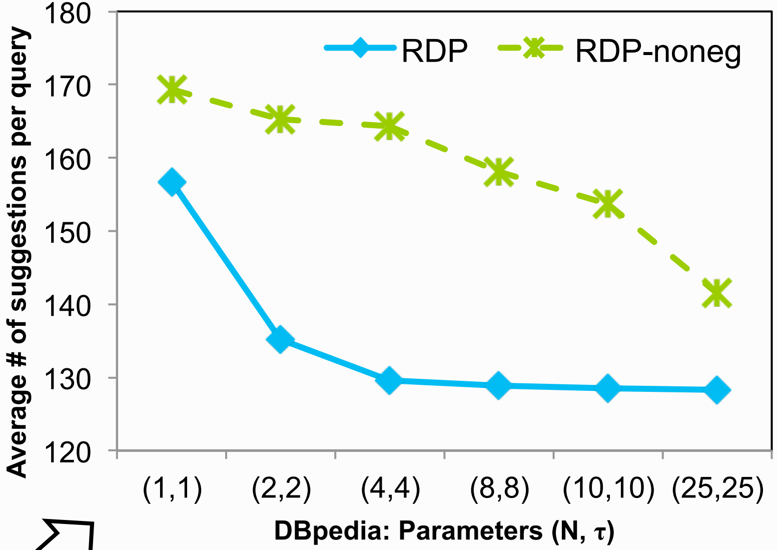


RDP requires only 40 suggestions, 1.5-4 times fewer than other methods

RDP requires fewer suggestions compared to all other methods

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