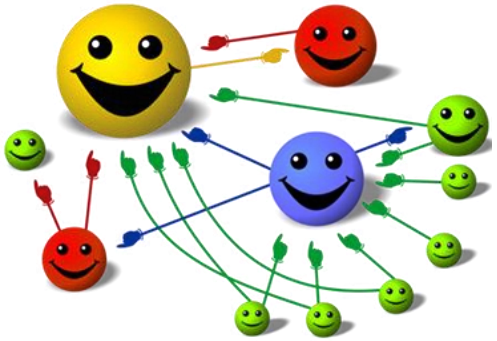
Pragmatics Aware Querying in Heterogeneous Knowledge Graphs

Amar Viswanathan (kannaa@rpi.edu)

RDF Graphs



- Data represented as graphs.
- An RDF Graph is composed of a lot of '**triples**'.
- A triple is a "Subject", "Predicate" and an "Object".

RDF Graphs

Simple RDF Movie Graph

?X rdf:type Person
?X hasName “Robert De Niro”
?X hasProfession Actor
?X actedIn Movie_1

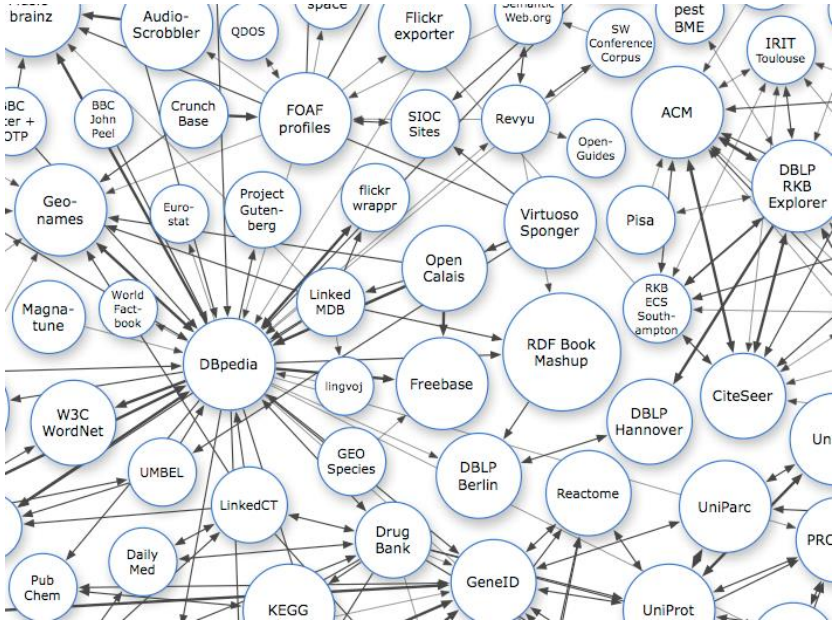
?X rdf:type Person
?X hasName “Joe Pesci”
?X hasProfession Actor
?X actedIn Movie_1

Movie_1 hasName “Goodfellas”
Movie_1 hasDirector Person

Possible Query : “Find all movies where Robert De Niro and Joe Pesci acted together”

Linked Data

- Lot of RDF Graphs interconnected to one another.
- DBPedia , YAGO and Freebase are examples.



The Semantic Web isn't just about putting data on the web. It is about making links, so that a person or machine can explore the web of data. With linked data, when you have some of it, you can find other, related, data.

Querying RDF Graphs

- By a graph language known as “SPARQL” (pronounced as **SPARKLE**…)
- Express your query conditions as “**graph patterns”** or “**subject predicate triple”** patterns.
- Patterns are matched against the graph and data is retrieved.
- Such queries are known as “Conjunctive Queries”, as in a conjunction of “query conditions”.

Conjunctive Query

- $\mathcal{T} \Leftarrow \mathcal{P}$ is a **Conjunctive query** $\mathcal{Q}$ [Hurtado et al., 2008].
- $\mathcal{T}$ is the head of $\mathcal{Q}$ and $\mathcal{P} = t_1 \wedge t_2 \wedge t_3 \ldots \wedge t_n$. is the body of $\mathcal{Q}$
- P is the conjunction *RDF triple patterns* t_k which is a $\{s, p, o\} \in (\Upsilon \cup \vartheta) \times (\Upsilon \cup \vartheta) \times (\Upsilon \cup \vartheta \cup \Lambda)$, where variables ϑ are disjoint from the sets Υ, β and Λ.

```
SELECT ?X ?Y
WHERE
{
?Y ub:subOrganizationOf 'University8' .   (t1)
?X ub:researchInterest 'Research28' .     (t2)
?X rdf:type ub:Lecturer .                 (t3)
?X ub:worksFor ?Y .                       (t4)
}
```

If any of the triples (t1,t2, t3 or t4) fail then our query also fails.

Query Reformulation

Reformulation

Query Reformulation takes in $\mathcal{Q}$ and rewrites it into an equivalent union of conjunctive triple patterns $\mathcal{Q}'$, based on *relaxation* or *inference rules* on $\mathcal{P}$

```
SELECT ?X ?Y
WHERE
{
?Y ub:subOrganizationOf 'University8' .   (t1)
?X ub:researchInterest 'Research28'   .   (t2)
?X rdf:type ub:Lecturer               .   (t3)
?X ub:worksFor  ?Y                    .   (t4)
}
```

```
?X rdf:type ub:FullProfessor          .   (t3)
```

Triples (t2) AND (t3) fail i.e. “No Lecturer researching on “Research28”

State of RDF Search

- One way communication between the user and the system.
- Onus on the user to understand the underlying system and then pose a question.
- If a user doesn't get results he tweaks the question again and sends it to the system.
- Process can be time consuming because :
  - User needs to understand the system.
  - System doesn't adapt itself to the user's needs.
- If a query does not give any results, the user has no clue as to why the system gave such results.

Ideal Search Process

Talk exchanges do not normally consist of a succession of disconnected remarks, but they are, to some degree, cooperative efforts and the participants recognize in them, a common set of goals..

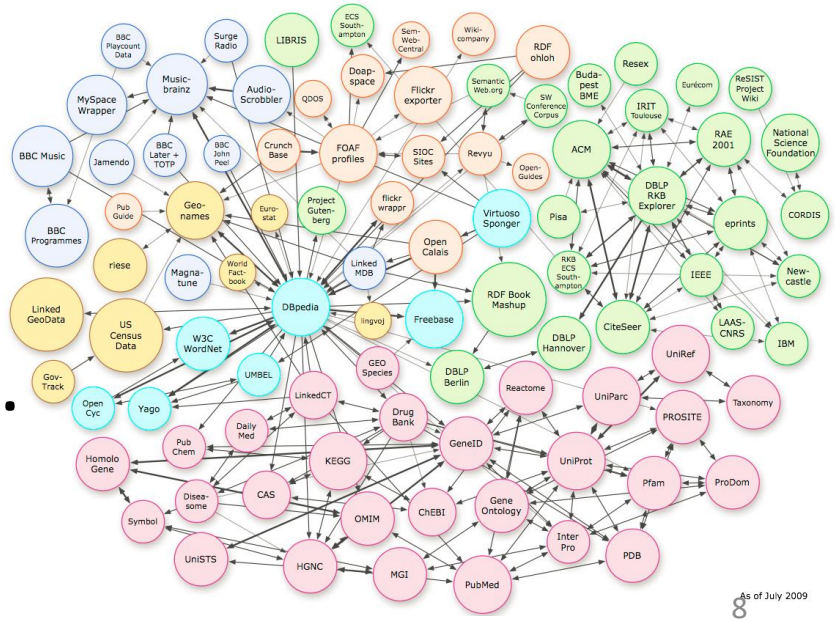
--H.P. Grice, "Logic and Conversation"

Motivation

- Handling SPARQL **Query Failure** is largely unexplored.
- No intuitive mechanism to aid user when a query fails i.e. **returns zero results.**
- User left to figure out the reason for query failure and system doesn't provide relevant clues.
- Cooperative Answering approach where both the participants engage to achieve mutual conversational ends.

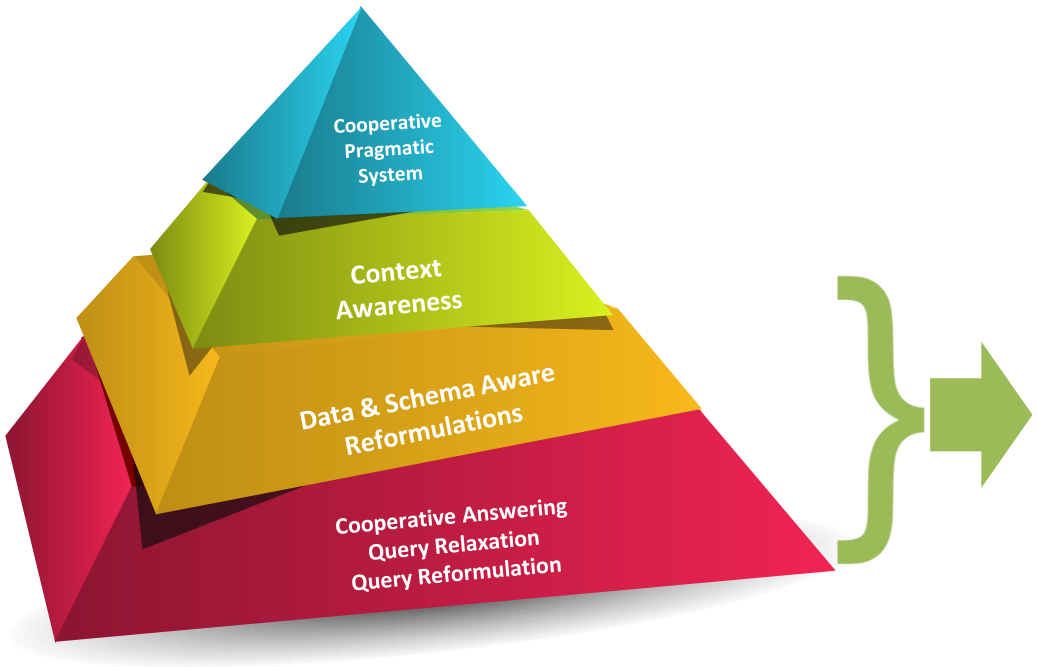
Why Linked Data?

- Proliferation of RDF based Linked Data
- Expressivity of RDF Graphs.
- Relationship based Graph Querying.
- Ability to express extracted **knowledge as relationships.**



Thesis Contributions

What was..



Pragmatic System Pyramid

What is..*

- A Query Reformulation methodology that takes into account:
  1. **RDF Entailment Rules**
  2. **Data Distribution**
  3. **Data Availability**
  4. **Schema Awareness**
- Finding **failing triples** – extending the Minimal Failing Subquery problem
- Using data distribution along with the **hierarchy** information to rank
- Using the generated reformulations as an input to further explore the addition of human preferences

Challenge

“It must be possible to express a query that doesn’t fail when some specified part of the original query fails to match.”

--W3C RDF Data Access Group

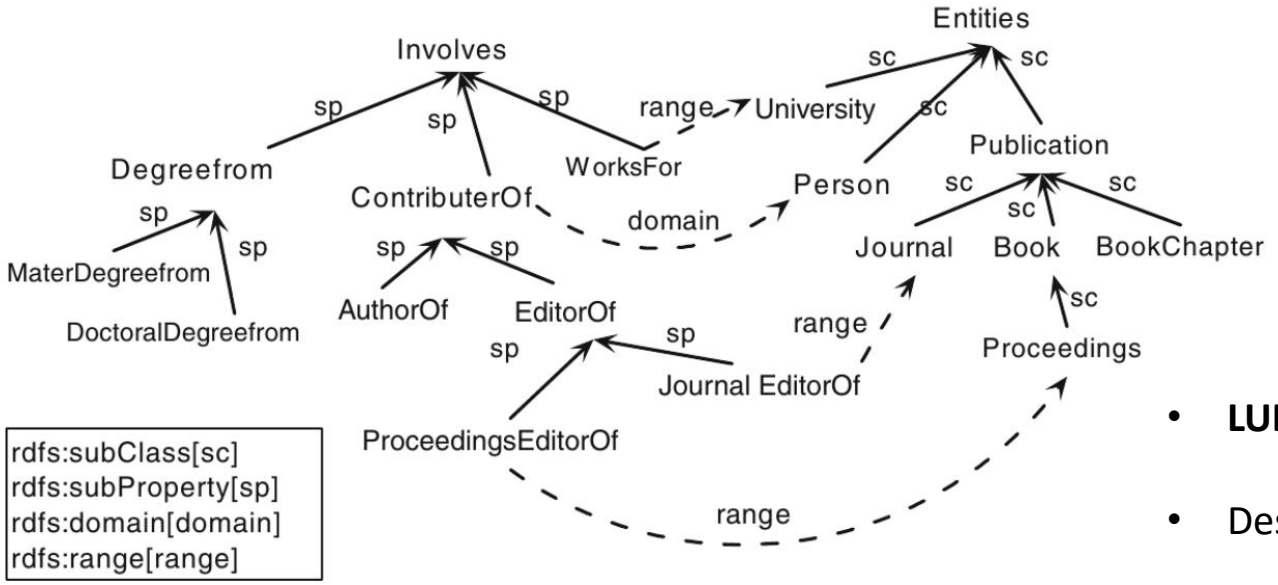
Questions

- Can “Query Reformulation” solve the **Query Failure** problem?
- If yes how many reformulations would you need?
- Are there rules to be followed? Where would you apply them?
- Is this process automatic or human aided?
- How do you identify parts of the query that need reformulation?
- Does the reformulated query **preserve the semantics** of the original query?

Background

- Idea of making queries more flexible
  - Borrows from Grice's Cooperative Answering Maxims.
  - Gaasterland, Godfrey and Minker proposed a mechanism for databases and termed it "query relaxation"
  - For RDF the relaxation rules were first proposed by Hurtado et.al.
- Basis for Data Awareness is based on the Information Entropy.
- Query Reformulation is an extension on Query Rewriting .
- Schema based similarity metrics owe their basis to hierarchical distance measures.

Setting



- LUBM100 with 10,115,551 triples, 43 classes, 32 properties as the synthetic dataset.

- **LUBM** rdf graph used.
- Describes a **University Ontology**.
- Interactions between Students, Professors, Employees and Lecturers.
- Expressive and easily “scalable”.

Example Query

All lecturers researching ‘Research28’ and part of University8 .

```
SELECT ?X ?Y
WHERE
{
?Y ub:subOrganizationOf 'University8' .   (t1)
?X ub:researchInterest 'Research28'   .   (t2)
?X rdf:type ub:Lecturer               .   (t3)
?X ub:worksFor  ?Y                    .   (t4)
}
```

- Query returns **zero results**.
- This is a failed query as no answers are retrieved.

When in doubt Reformulate

- One way to reformulate is to relax the triples
  - **rdf:type** to **FullProfessor,**
  - **‘Research28’** by **‘a variable’** .
- This uses the *subclass* and *subproperty* hierarchy in the schema and the sample reformulations would look like.

```
SELECT ?X ?Y
WHERE
{
?Y ub:subOrganizationOf 'University8' .   (t1)
?X ub:researchInterest ?Z                 .   (t2)
?X rdf:type ub:FullProfessor              .   (t3)
?X ub:worksFor  ?Y                        .   (t4)
}
```

This would result in 776 results on the current RDF graph

How many Reformulations?

- The system can actually reformulate using all the 43 classes and 32 predicates.
- This would actually result in **43 x 32 x 43 = 59,168 reformulations** ☺
- The number of reformulations increases as the expressivity of the Knowledge Graph Schema increases.

Schema and Data Awareness

Entailment Rules

	Triple Pattern	Relaxed Triple Pattern	Condition
1	t(s rdf:type c_1)	t(s rdf:type c_2)	c_1 rdfs subClassOf c_2
2	t(s p_1 o)	t(s p_2 o)	p_1 rdfs subPropertyOf p_2
3	t(s rdf:type X)	t(s rdf:type c_i) $\forall i$	$c_1, c_2, \ldots, c_i$ are classes in Schema S
4	t(s X o)	t(s rdf:type p_i) $\forall i$	$p_1, p_2, \ldots, p_i$ are properties in Schema S

- Using “class and property” hierarchies – **Ontological Relaxation**
- Replacing “constants with variables” - **Data Relaxation**
- Replacing “literals with one more triples by utilizing attributes of the schema and data” – Pragmatic Relaxation.

Information Content

- Given a concept (c1) and a relaxed concept (c2) the similarity is calculated by the **Information Content(IC)** present in them. That is :

$$Sim(c_1, c_2) = \frac{IC(c_2)}{IC(c_1)}$$

- However every triple is a combination of concepts so given a triple t(s,p,o) and its relaxed triple t’(s’,p’,o’), we define the similarity as :

$$Sim(t, t') = \frac{1}{3} * Sim(s, s') + \frac{1}{3} * Sim(p, p') + \frac{1}{3} * Sim(o, o')$$

Minimal Failing Triples

How do you pick the triples that you want to reformulate?

Intuition : Find all conjunction of triples in the query that would cause a failure.

Algorithm

1. Remove all the triples and then add them back to the query one by one.
2. If the addition of a triple causes a failure, then that triple is part of a Minimal Failing Subquery.
3. Iterate the process over until no triples remain.

- This is an extension of the Minimal Failing Subquery (Godfrey, 1998) problem to SPARQL queries.
- The algorithm is a greedy algorithm. (**Obviously we should do better**)
- Complexity increases with the number of triple patterns.

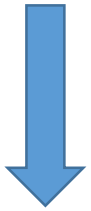
Advantages of finding “failing triples”

- Automatically gives us a pointer to the right triples that can be reformulated.
- The algorithm is simple enough and runs in fairly quick time.

```
SELECT ?X ?Y
WHERE
{
?Y ub:subOrganizationOf 'University8' .   (t1)
?X ub:researchInterest 'Research28'   .   (t2)
?X rdf:type ub:Lecturer               .   (t3)
?X ub:worksFor  ?Y                    .   (t4)
}
```

t2 and t3 are the failing triples i.e.

There is no lecturer who does “Research28”



Using Schema and Data Awareness

```
SELECT ?X ?Y
WHERE
{
?Y ub:subOrganizationOf 'University8' .   (t1)
?X ub:researchInterest ?Z             .   (t2)
?X rdf:type ub:FullProfessor          .   (t3)
?X ub:worksFor  ?Y                    .   (t4)
}
```

For a cutoff of k = 50 results,
just 2 reformulations are enough

Er..does the reformulation mean the same thing?

Does the reformulated query **preserve the semantics** of the original query?

- Use Taxonomy based similarity metrics (Resnik et.al, 1995)
- Based on the information content of the lowest common ancestor.

In simple words,

- Assistant Professor is closer to Lecturer than Associate Professor
- Full Professor is closer to Assistant Professor than Associate Professor

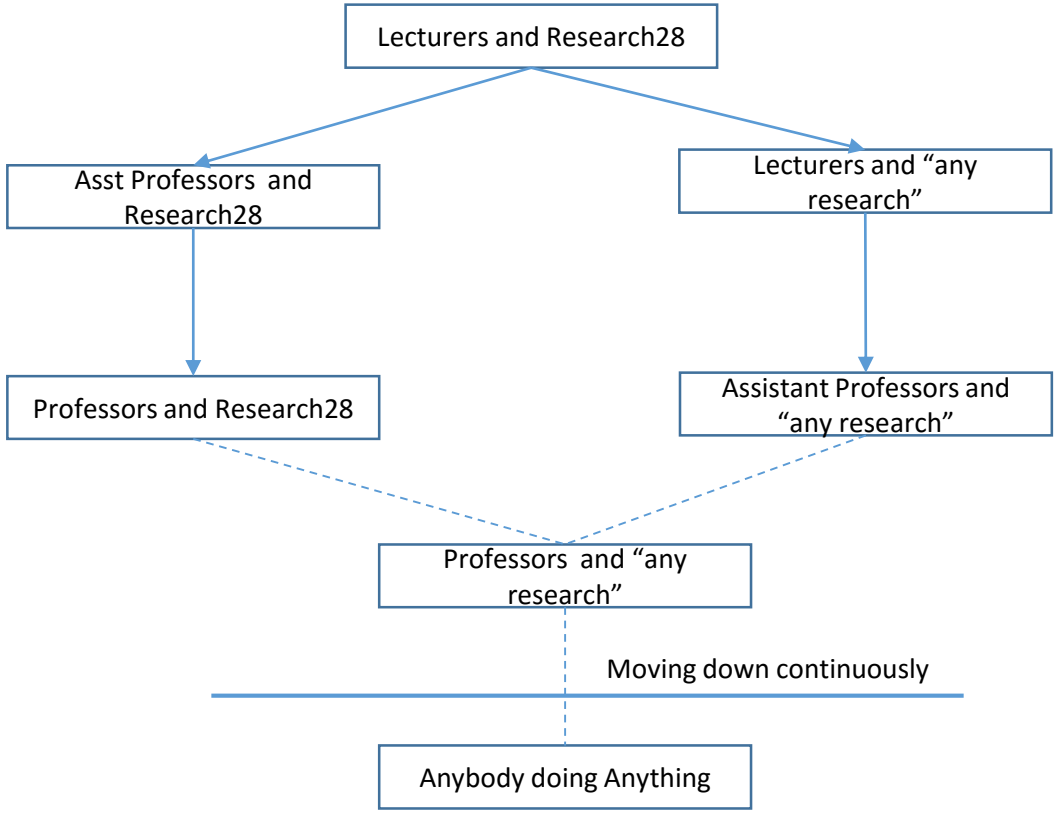
Lecturers Researching 'Research28' and part of Y that is subOrganization of University8	
Reformulated Query	**Similarity to Original Query**
AssistantProfessors researching ?Y	0.831
FullProfessors researching ?Y	0.80
AssociateProfessors researching ?Y	0.78
People researching ?Y	0.40
Entity researching ?Y	0.20

Similarity of reformulated queries to the original query

The Query Lattice

Which results do you display first?

- Build a query lattice of all possible paths.



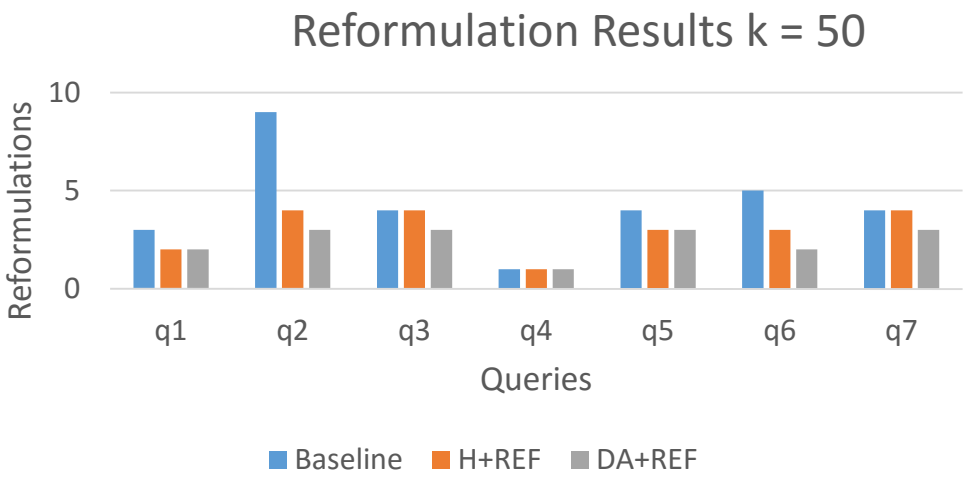
Evaluation Plans and Baseline

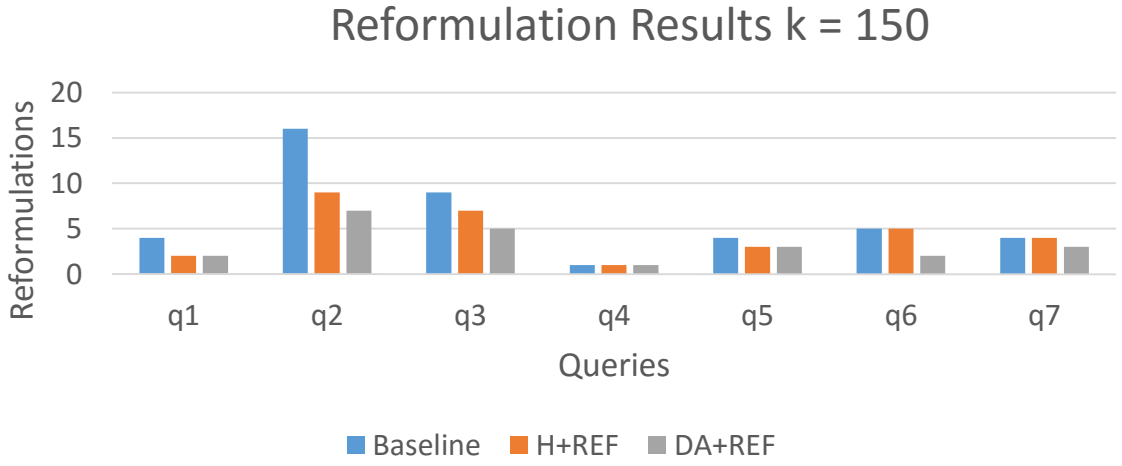
We evaluate our algorithms based on the number of reformulations required to generate a given **k results** when a query failure occurs.

Baseline : Baseline Algorithm by Huang et.al, which reformulates based on a heuristic and similarity tree.

H+REF : Our algorithm which uses Minimal Failing Subqueries and Hierarchical Similarity.

REF + DA : Uses Data Awareness along with REF





Quick note on the queries

```
Select ?X ?Y1 ?Y2 ?Y3 ?Y4  Where {
                ?X rdf:type ub:VisitingProfessor.
                ?X ub:memberOf <http://www.Department1.University1.edu>.
                ?X ub:name ?Y1. ?X ub:emailAddress ?Y2.
                ?X ub:telephone ?Y3.
                ?X ub:undergraduateDegreeFrom ?Y4.}
```

OR

```
Select ?X ?Y Where {
                ?X ub:advisor ?Y.
                ?Y ub:headOf University476.  }
```

Star Queries : Every triple shares the same subject

Chain Queries : Object of one triple is subject of another

```
Select ?X ?Y1?Y2 Where {
                ?X  rdf:type ub:FullProfessor.
                ?X ub:doctoralDegreeFrom <http://www.University8.edu>.
                ?X ub:researchInterest 'Research23'.
                ?X  ub:teacherOf ?Y1.
                ?Y2  ub:takesCourse ?Y1. }
```

Composite Queries : Combination of both **Star** and **Chain** Queries

Timeline of Future work

- Improve the process of finding “Failing Subqueries”.
- Can you use schema properties and filter out before using the MFS.
- How does presence/absence of linked concepts help find a better similarity measure.

Spring/Summer 2016

- Replacing “literals with more triples by utilizing attributes of the schema and data” – Pragmatic Relaxation or **Neighborhood Concept based Reformulation**
- Create the API and generalize it to take any RDF graph and ontology.

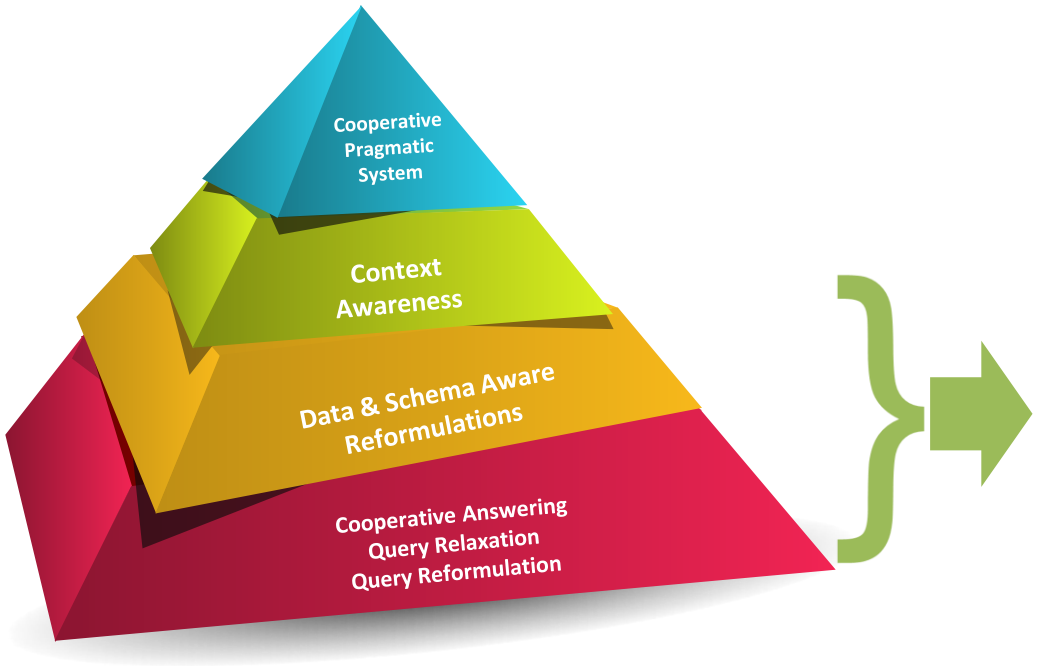
Summer/Fall 2016

Thesis Completion

Finalize the User Interface workflow
Use the user interface to define interactions and then look for “intent”
Use the generated reformulations along with the user preferences to slowly move towards “Pragmatics” in Grapy Querying.

Beyond Fall 2016…

Final Summary



Pragmatic System Pyramid

- To be able to truly build a “Pragmatic system” we need to address the issues of **interpretations**, **context** and finally **user preferences**.
- My thesis aims at solving the issue of **interpretations** by combining knowledge of both **data** and **schema** and hopefully aims to build some amount of context and user preferences.

For more technical details on the algorithms or methods, or perhaps criticisms, drop a mail at kannaa@rpi.edu