Discovering Informative Subgraphs in RDF Graphs

William H. Milnor
Cartic Ramakrishnan
Wright State University - Main Campus

Matthew Perry
Wright State University - Main Campus

Amit P. Sheth
Wright State University - Main Campus, amit@sc.edu

John A. Miller
Wright State University - Main Campus

See next page for additional authors

Follow this and additional works at: https://corescholar.libraries.wright.edu/knoesis

Part of the Bioinformatics Commons, Communication Technology and New Media Commons,
Databases and Information Systems Commons, OS and Networks Commons, and the Science and
Technology Studies Commons

Repository Citation

This Report is brought to you for free and open access by the The Ohio Center of Excellence in Knowledge-Enabled Computing (Kno.e.sis) at CORE Scholar. It has been accepted for inclusion in Kno.e.sis Publications by an authorized administrator of CORE Scholar. For more information, please contact library-corescholar@wright.edu.
Authors
William H. Milnor, Cartic Ramakrishnan, Matthew Perry, Amit P. Sheth, John A. Miller, and Krzysztof Kochut

This report is available at CORE Scholar: https://corescholar.libraries.wright.edu/knoesis/719
Discovering Informative Subgraphs in RDF Graphs

William H. Milnor, Cartic Ramakrishnan, Matthew Perry, Amit P. Sheth, John A. Miller and Krzysztof J. Kochut
LSDIS Lab, University of Georgia, Athens, Georgia
{milnor, cartic, mperry, amit, jam, kochut}@cs.uga.edu

Abstract. Discovering patterns in graphs has long been an area of interest. In most contemporary approaches to such pattern discovery either quantitative anomalies or frequency of substructure is used to measure the interestingness of a pattern. In this paper we address the issue of discovering informative subgraphs within RDF graphs. We motivate our work with an example related to Semantic Search. A user might pose a question of the form: “What are the most relevant ways in which entity X is related to entity Y?” the response to which is a subgraph connecting X to Y. Relevance of the discovered subgraph therefore will depend on the amount of useful information conveyed to the user. This in turn depends on the meaning of the edges in the subgraph. We introduce heuristics that guide a discovery algorithm away from banal paths towards more informative ones. This guidance is based on weighting mechanisms (driven by edge semantics) for the edges in the RDF graph. We present an analysis of the quality of the subgraphs generated with respect to path ranking metrics. We then conclude presenting intuitions about which of our weighting schemes and heuristics produce higher quality subgraphs.

1 Introduction

“I keep six honest serving-men (They taught me all I knew); Their names are What and Why and When And How and Where and Who.”—(Rudyard Kipling, from "The Elephant's Child" in Just So Stories 1902). The six questions in this quote by Rudyard Kipling are often tools we as humans use in an attempt to gain knowledge. In our opinion, Why and How two entities are related are the crucial questions that must be answered. Discovering relevant sequences of relationships between two entities answers these questions. A discovery process therefore requires investigation of relationships between entities. We envision a system, which supports its users in discovering ways in which a pair of entities are related. It is very likely that semantic search engines of the future will need to support such a discovery process. This perspective coincides with the Semantic Web vision [1]. To this end, we investigate techniques that provide users with a chain of relationships between entities in response to queries of the following kind: “What are the most relevant ways in which entity X is related to entity Y?” The notion of relevance is critical to the definition of such a query. This becomes clear when one considers the small-world phenomenon [4][5]. Given a knowledge base and any two entities X and Y there could be a myriad of relatively
short chains (i.e. six degrees) of relationships linking the two. Hence the need for some way of semantically constraining and discovering the possible ways in which X and Y could be related.

In [7] the authors address this issue by developing an algorithm to extract relatively small but most relevant subgraphs. They define the Connection Subgraph Problem as follows:

**Given**: an edge-weighted undirected graph \( G \), vertices \( s \) and \( t \) from \( G \) and an integer budget \( b \)

**Find**: a connected subgraph \( H \) containing \( s \) and \( t \) and at most \( b \) other vertices that maximizes a “goodness” function \( g(H) \).

We adapt this approach to recast the problem of finding complex relationships between RDF resources (Semantic Associations [2]) into that of finding informative and relevant subgraphs. The data set used in [7] is akin to a social network, and their weighting scheme is based on frequency of co-occurrence of names in Web pages. Clearly this weighting scheme will not work for finding relevant subgraphs in RDF graphs since the semantics of each property type in RDF is different. Therefore a systematic way of weighting edges based on the semantics conveyed by the ontology represented using RDF schema [8] is needed. To extend the approach in [7] to the more general case of an RDF graph, we propose heuristics for edge weighting that depend indirectly on the semantics of entity and property types in the ontology and on characteristics of the instance data. More specifically, we define class and property specificity, Instance Participation Selectivity and a Span Heuristic. We evaluate the generated subgraphs using the path ranking schemes suggested in [11],[22],[15].

2. Related work

Reasoning and knowledge discovery over graph data models has been studied in the Graph mining community and more recently in the context of the Semantic Web. The remainder of this section highlights work which is most relevant to ours.

The work most directly related to graph-based knowledge discovery and reasoning for the Semantic Web is that of Semantic Associations which was first introduced in [2]. Semantic Associations (termed ρ-operators) represent meaningful but directed paths in an RDF meta-base. To the best of our knowledge this is the only existing work of this type. Anyanwu and Sheth define the ρ-path operator among others. Two entities X and Y are said to be ρ-path associated if there exists a sequence of properties (relationships) starting at X connecting intermediate entities and ending at Y. The nature of web data [5] often leads to an overwhelming amount of associations between two entities. To combat this problem, [11][22] propose to rank Semantic Associations. As an alternate approach, the method in [10] filters the search space before
computing associations. They adapt Kleinberg’s hub and authority scores [14] to compute importance of Semantic Web resources and then only consider nodes with importance greater than some threshold when computing Semantic Associations. Their preprocessing step based on importance thresholds is likely to discount those paths that contain even a single unimportant node. Our approach to this problem is fundamentally different from these two. We try to find the ‘best’ set of associations which contain a visually comprehensible number of resources.

There has been a considerable amount of work done in the field of graph mining to detect patterns in graphs. Patterns discovered are characterized either by their anomalous nature or frequent occurrence, among other things. Efficient algorithms have been developed for many variations of the frequent subgraph discovery problem [16][17][18]. Community and group detection is another well-studied graph mining problem which attempts to discover communities and groups based on link analysis. The problem has been studied on both the web graph [19][20] and other data sets [21]. These graph mining problems focus on graphs with single node types and single edge types, however. For the Semantic Web we need algorithms which take into account the semantics of different node and edge types. Novel Link Discovery was introduced in [15] and involves finding novel paths between entities, novel loops, and significantly connected nodes. The methodology used in this work considers different node and edge types but differs from ours in that importance is determined purely from rarity. Also the paths examined are considerably shorter than the ones we examine.

3. Heuristics

RDFS vocabulary allows users to represent classes and properties thereby indirectly imposing meaning on resources. Hence we define three quantities indirectly based on semantics and RDF statement types and frequencies. Our aim in doing this is to use semantics to compute edge weights thereby guiding the algorithm in the subgraph discovery process. We define a schema $S$ as the union of the following sets:

$$C = \{ \{c, \text{rdfs:Class}\} \}$$

$$P = \{ p \{p, \text{rdf:type, rdf:Property}\} \land \exists c, c' \in C \land \text{rdfs:domain}(p) \land \text{rdfs:range}(p) \}.$$ 

Further, we define an RDF data store $R = (\Pi, I)$ where $\Pi = \bigcup S$ and $I$ is the set of corresponding instance triples. We assume a resource that is classified as an instance of classes belonging to different schemas in our data set is uniquely identified by its URI. In other words, no data integration operation is required.

**Class and Property Specificity (CS and PS)**

Intuitively more specific resources and properties convey more information than general ones. As a result of the $\text{rdfs:subClassOf}$ and $\text{rdfs:subPropertyOf}$ properties provided by RDF schema it is possible to impose a partial ordering of properties and classes in the schema resulting in a wellformed hierarchy of classes and properties. For a given property $p$, let $d(p)$ be the length of the longest path in the hierarchy tree...
that contains $p$, and for a given class $c$, let $d(c_H)$ be the length of the longest path in the hierarchy tree that contains $c$. Properties and classes at the root of their respective hierarchy trees in the schema are considered most general while those at the leaves of these trees are considered most specific. Therefore a measure of specificity can be associated with each class or property commensurate with its position in its hierarchy. Let the depth of an arbitrary property in its property hierarchy be $d(p_i)$ and the depth of an arbitrary class in its class hierarchy be $d(c_j)$. Therefore, the specificity of property $p_i$ and class $c_j$ are given by

$$\mu(p_i) = \frac{d(p_i)}{d(p_{ii})} \quad \mu(c_j) = \frac{d(c_j)}{d(c_{jj})}$$

(1)

Every resource that is an instance of the class $c_j$ is assigned the weight $\mu(c_j)$. If a resource $r$ is an instance of $k$ distinct classes it is assigned the value $\mu(r) = \max \{ \mu(c_j) \}$. To convert this node weight into an edge weight, the value is equally distributed among all of the edges incident on the resource $r$. This weighting scheme favors nodes with lower degree since the node specificity is divided equally among its incident edges, therefore edges incident on nodes with high degree will get a lower weight.

**Fig. 1.** Illustrative example for Instance Participation Distribution

**Instance Participation Selectivity (IPS)**

Another rule-of-thumb is that rarer facts are typically more informative that frequently occurring ones [15]. Consider the example shown in Fig.1. The example shows two relationships `lives_in` and `council_member_of` defined on the classes `Person` and `City`. The instances $p_1, p_2, ..., p_m$ of the class `Person` are members of the council of `City c_1`, hence the relationship `council_member_of` between each $p_1, p_2, ..., p_m$ to $c_1$. Instances of class `Person` $p_{m+1}, p_{m+2}, ..., p_k$, $p_k$ represent people who live in `City c_1` and therefore are related to $c_1$ by the relationship `lives_in`. From the perspective of the node $c_1$, following an edge labeled `lives_in` will lead to one node among $k-m$ possible nodes. In contrast, following an edge labeled `council_member_of` will lead to one node among $m$ nodes. Given that rarer paths are considered more informative, the amount of information gained by choosing to traverse the `council_member_of` relationship to a node in the set $\{p_1, p_2, ..., p_m\}$ is more than the gain achieved by choosing to traverse the `lives_in` relationship to a node in the set $\{p_{m+1}, p_{m+2}, ..., p_k\}$. This
is akin to choosing the hop with maximum information gain. To define this heuristic formally, we define the notion of the type of an RDF statement. The type of an RDF statement $\langle s,p,o \rangle$ is defined as the triple $\pi = \langle C_i, p, C_j \rangle$ where $\text{typeOf}(s) = C_i$ and $\text{typeOf}(o) = C_j$. Further, $|\pi|$ is thus the number of statements of type $\pi$ in a given RDF instance base. We therefore define Instance Participation Selectivity for each RDF statement as $\sigma_\pi = 1/|\pi|$. Going back to Figure 1, let $\pi = \langle \text{Person}, \text{lives_in}, \text{City} \rangle$ and $\pi' = \langle \text{Person}, \text{council_member_of}, \text{City} \rangle$. According to this example, $\sigma_\pi = 1/(k-m)$ and $\sigma_{\pi'} = 1/m$ and if $k>m$ then $\sigma_{\pi'} > \sigma_\pi$.

The Span Heuristic (SPAN)

In [11] the authors define a ranking metric known as Refraction. Given a path of the form $v_1, e_1, v_2, e_2, ..., e_n, v_n$ from $v_1$ to $v_n$, where $v_i \in \text{Resources}$ and $e_i \in \text{Properties}$, this path is said to refract if there exists at least a pair of statements $\langle v_i, e_i, v_{i+1} \rangle, \langle v_{i+1}, e_{i+1}, v_{i+2} \rangle$ such that $\not\exists \mathcal{S} \ni v_i, e_i, v_{i+1} \in \mathcal{S}$. In other words a path passes through more than one schema. This measures the extent to which a given path conforms to a schema. As mentioned earlier, one of the characteristics of a discovery process is the detection of anomalous information. We consider resources that are instances of classes belonging to different schemas as being indicative of anomalous paths between the given entities, since they tie different domains together. What makes such paths anomalous and therefore interesting is the fact that these paths represent a deviation from the expected paths suggested by the schemas. For example, in our scenario in Figure 4 an instance of the class $\text{Person}$ may be classified as both an instance of class $\text{Actor}$ in the $\text{Entertainment}$ domain and an instance of class $\text{SpokesPerson}$ in the $\text{Business}$ domain. Such an instance serves to link different schemas.

We therefore need a heuristic that favors the addition of such refracting paths to our subgraph. Let us consider the example in Figure 2. For every node $v$ in a given RDF graph we can define a set called $\text{SchemaCover} = \{ C \mid C \in S \wedge \text{typeOf}(v) = C \}$. The $\text{SchemaCover}$ for each of the nodes in the set $\{ u', u, v_1, v_2, v_3, v_4, v_5 \}$ is shown adjacent to the respective node in Figure 2. To favor paths that span as many schemas as possible the search algorithm favors nodes that are classified under as many “new” schemas as possible at each step. By “new” we mean schemas that have been least recently encountered along a particular path. Let $\text{SDiff}(u,v)$ represent the number of new schemas seen as a result of traversing the edge $(u,v)$, where the value of $\text{SDiff}(u,v) = |\text{SchemaCover}(v) - \text{SchemaCover}(u)|$. The idea behind $\text{SDiff}$ is to ensure that the discovery algorithm chooses a node that is in a “new” schema. However $\text{SDiff}$ alone does not ensure that search will continue through the “new” schema. To combat this problem we define the Cumulative Schema Difference $\text{CSDiff}(u,u';v_j) = 1 + \text{SDiff}(u, v_j) + \text{SDiff}(u', v_j)$ for $v_i \in \text{adj}[u] - \{ u' \}$. We normalize this Cumulative Schema Difference ($\text{CSDiff}$) measure to compute a factor $\beta_{u' \rightarrow u \rightarrow v_i}$:

$$\beta_{u' \rightarrow u \rightarrow v_i} = \frac{\text{CSDiff}}{1 + 2(m-1)}$$

where $m$ is the number of schemas. \hspace{1cm} (2)
We then obtain the adjusted weight given by:

\[ w'(u,v) = \beta_{u \rightarrow v} \ast w(u,v) \]  

(3)

Fig. 2. Example of Span metric computation

The effect of the factor \( \beta \) is to bias edge weights in the following way. Successor nodes that are instances of classes belonging to schemas other than those of the current and previous node are more likely to be visited, quantified by the two SDiff terms of CSDiff. More specifically, in the case of the example in Figure 2, a partial ordering is induced by the adjusted weights \( w'(u,v) \), on the nodes as follows \( v_1 \succ v_2, v_1 \succ v_3 \succ v_5 \). The node \( v_1 \) is therefore visited next. However, the measure \( \beta \) is not sufficient to distinguish between nodes in all cases. Consider the example in Figure 3. Nodes \( v_1 \) and \( v_2 \) have the same value of \( \beta_{u \rightarrow v} \); but \( v_1 \) should be more desirable than \( v_2 \) because it has a larger SchemaCover value. For such cases, we define a factor called SchemaCoverFactor \( \alpha(u,v) \):

\[ \alpha(u,v) = \frac{1}{2} \left( \frac{\text{SchemaCover}(u)}{m} + \frac{\text{SchemaCover}(v)}{m} \right) \]

(4)

As per the calculations shown in Figure 3 this factor treats the node \( v_1 \) preferentially over node \( v_2 \) i.e. \( v_1 \succ v_2 \) thus resolving the ambiguity. The value of \( \beta_{u \rightarrow v} \) is computed at each step during the discovery process in contrast with the \( a \) priori values of edge weights computed using \( \mu, \sigma \) and \( \alpha \) as shown in equation 5. \( \beta_{u \rightarrow v} \) is then used to adjust the weights \( w(u,v) \) as shown in equation 3.

\[ w(u,v) = \frac{\mu(p_{u,v})}{\text{degree}(u)} \left( \frac{\mu(u)}{\text{degree}(u)} + \frac{\mu(v)}{\text{degree}(v)} \right) + \sigma + \alpha(u,v) \]

(5)

where \( p_{u,v} \) is the property connecting the resource node \( u \) and \( v \), and \( \pi \) is the type of the statement \( (u, p_{u,v}, v) \).
Since
\[
\alpha(u, v) = \frac{1}{2} \left( \text{SchemaCover}(u) + \text{SchemaCover}(v) \right) = 0.75
\]
\[
\alpha(u', v) = \frac{1}{2} \left( \text{SchemaCover}(u') + \text{SchemaCover}(v) \right) = 0.625
\]
And,
\[
w(u, v) = \beta_{u',v_1} \ast w(u, v_1)
\]
Because \(w(u, v)\) is calculated using \(\alpha(u, v)\)
\(w(u,v) \neq w(u,v_2)\)
\(\therefore w'(u,v_1) \neq w'(u,v_2)\)
even if \(\beta_{u',v_1} = \beta_{u,v_2}\)

Fig. 3. Influence of the Schema Cover Factor \(\alpha\)

4. Algorithms

After obtaining a weighted RDF graph using computations described in the previous section, we employ the algorithms from [7] to obtain a display graph connecting two resources. The authors present an algorithm for extracting a so-called candidate graph from an input graph. They also propose an algorithm based on electrical circuits to extract a display graph from the candidate for a given budget \(b\). For our purposes we refer to these as Candidate \(\rho\)-graph and Display \(\rho\)-graph. We assume that the properties (edges) in the RDF graph are undirected. Consider a query which asks to find the relevant ways in which entity \(Y\) is related to entity \(X\). We make this assumption to prevent the exclusion of a path of the form \(Y \xrightarrow{\beta} \cdots \xrightarrow{\beta} X\).

Candidate \(\rho\)-graph generation algorithm

The candidate \(\rho\)-graph generation algorithm is based on a notion of distance between two nodes. The algorithm grows a set \(S\) around the source node \(s\) and a set \(T\) around the sink node \(t\) (\(s\) and \(t\) are referred to as the roots of their respective sets) until a
certain threshold is met: a maximum number of total nodes or maximum number of cut edges between $S$ and $T$. At each iteration, a pending list is maintained for each of these sets which consists of those nodes $n \not\in S$ and $n \not\in T$ and adjacent to some node $n' \in S$ and $n' \in T$. The sets $S$ and $T$ are expanded by choosing from the pending list the node with shortest distance to either $s$ or $t$. Let $u'$ be the predecessor of $u$ (the node adjacent to $u$ on the shortest path to its root). For an edge $(u, v)$ the distance between $u$ and $v$ is given by the following formula, and the length of a path is the sum of the distances between its edges.

$$
\text{distance}(u, v) = \log \left( \frac{\text{degree}(u) + \text{degree}(v)}{w(u, v) \beta_{u', v}} \right)
$$

(6)

The aim of our initial experiments is to determine the quality of the Candidate $\rho$-graph in terms of its ability to capture the best paths between the query endpoints.

**Display $\rho$-graph generation algorithm**

The display generation algorithm prunes the generated candidate graph down to a smaller size while ensuring that the resultant pruned graph conveys maximum information. In [7] the authors present a rather elegant solution to this by modeling the graph as an electrical circuit where the edge weights represent the conductance values in the circuit. They use the fact that current flows from high voltage to low voltage, to impose direction on an otherwise undirected graph. Using Ohm’s law and Kirchhoff’s law, a system of linear equations is created with voltages at each node as a variable in these equations. Solving this system of equations gives voltages at each node. This step takes $O(\rho^3)$ time, which motivates the need for the Candidate graph generation process. The greedy display generation algorithm attempts to find a display graph of at most $b$ (set to 100 in our experiments) nodes which maximize the amount of total current delivered from the start node to the end node. Starting with an empty subgraph, this algorithm iteratively adds paths until meeting the budget $b$. At each of the iterations, a dynamic programming algorithm is used to make the greedy choice of which path to add to the subgraph. The greedy choice is the path which has the maximum ratio of delivered current to number of new nodes added to the subgraph. In our experiments we test the model based on current flow used to compute these display graphs.

**5. Dataset and Scenario**

We used a synthetic dataset for our experiments since we needed control over characteristics of the data. This helps us ensure that our results are not unduly affected by unknown aspects i.e. connectivity, relative instance distribution etc. of the dataset. Collection of real world data follows an almost opportunistic approach since availability often dictates design. As a result there is room for skew in instance data population. This skew may not always reflect real-world distributions, as was observed in our experience with SWETO [23]. Consequently we developed an algorithm that takes as input a set of ontology schemas and a properties file specifying relative distributions of instances of classes and properties that would be expected in
tributions of instances of classes and properties that would be expected in the real world. For example, consider two classes in the Business ontology (Appendix Fig. A2.): Trustee and Employee. It would be reasonable to assume that if there are 5000 instances of the class Employee then there are unlikely to be 1000 instances of the class Trustee. Instances of the class Trustee are more likely to number between 10 and 100. These numbers are domain specific. Our method for assigning values to these relative distributions is empirical and a discussion of this issue is beyond the scope of this paper. The result of running this algorithm is an RDF graph that contains nodes and edges that are instances of classes and property types belonging to any or all of the classes in the given schemas. The graph for our experiments contains 30,000 nodes and 45,000 edges.

Fig. 4. Example snippet of a subgraph returned for the query $p(\text{Actor}_5567, \text{Captain}_8262)$ on our synthetic dataset– Nodes in the above graph are color-coded according to the ontology their class belongs to.

As a motivation for the domains used in our dataset consider the following example. A fraud investigator with the Securities and Exchange Commission (SEC) receives the following piece of information about a week after the stock prices for EntertainmentCompany_9982 plummet. Actor_5567 sold 70% of his shares of EntertainmentCompany_9982 one week after Capt_8262 sold all of his shares in the same company. Both transactions took place two weeks before the prices plummeted. The example subgraph shown in Fig. 4, might help an investigator visualize the connections between the resources Actor_5567 and Captain_8262.

6. Results and Evaluation

We recognize the fact that the notion “best” subgraph is very subjective and dependent on the user’s perspective. It is however desirable to have an objective measure that could be used to quantify the quality of a generated subgraph. The issue of judging relevance of paths i.e. path ranking has been addressed in [11] and [22]. In [15] the authors use rarity of the path as a measure of its interestingness. To the best of our knowledge these are the only three efforts that measure path relevance. We therefore use these path ranking mechanisms to evaluate the quality of both the Candidate $p$-graph and the Display $p$-graph. In our experiments the Candidate $p$-graphs gener-
ated contained 30001 nodes and the Display $\rho$-graphs were restricted to a maximum of 100 nodes making them easy to visualize.

6.1 Evaluation using Path Ranks

In our data set there are over 60 million paths of length 13 between the two endpoints used in Fig. 4. Paths of this length are unlikely to be of much interest to the user. To evaluate our subgraphs, we run an exhaustive $k$-hop limited Depth-First Search (DFS) on the input graph between the two entities. We use a depth limit of 9 hops for our experiments for feasibility of path enumeration for ranking. Note that both the Candidate $\rho$-graph and Display $\rho$-graph generated do contain arbitrary length paths, but we only consider paths of length at most 9 for fairness of comparison. We represent the paths returned by the $k$-hop DFS as the set $\text{FGPaths}_9$ (paths of up to 9 hops in the full graph). There are therefore 30 distinct $\text{FGPaths}_9$ sets, one for each query in our experiments. We rank the paths in each of the $\text{FGPaths}_9$ sets using the ranking mechanisms proposed in [11] and [22] in addition to what we call Rarity Rank based on the method suggested in [15]. The rank of a path $p$ based on the Rarity Rank scheme is given by the inverse of the number of paths that share the same type as path $p$. Each of the ranking mechanisms applied to the set $\text{FGPaths}_9$ results in a list of ranked paths. Let us assume that this leads to ranking from 1→$M$ where $M$ is the rank of the least relevant path. Let this set of ranked paths be represented as $\text{FGRankedPaths}_9$. We therefore have three distinct scales ($\text{FGRankedPaths}_9$ sets) against which the quality of both Candidate $\rho$-graph and the Display $\rho$-graph can be measured. In all of the graphs shown below the x-axis represents the 16 possible combinations of the 4 heuristics we use viz. class and property specificity (CS and PS), Instance Participation Selectivity (IPS) and The Span Heuristic (SPAN).

Measuring Candidate $\rho$-graph quality

To measure Candidate $\rho$-graph we compare the best paths in the entire graph to those in the Candidate $\rho$-graph. Let $\text{CGPaths}_9$ represent the set of paths in the Candidate $\rho$-graph with maximum length 9. For each path $p_{\text{candidate}} \in \text{CGPaths}_9$ we count the number of paths $p \in \text{FGRankedPaths}_9$ such that $\text{rank}(p) > \text{rank}(p_{\text{candidate}})$. This gives us the rank of each path in the Candidate $\rho$-graph with respect to all paths in the set $\text{FGRankedPaths}_9$. The score of a path is given by:

$$\text{score}(p_{\text{candidate}}) = |\text{FGRankedPaths}_9| - \text{rank}(p_{\text{candidate}})$$

(7)

The quality of the Candidate $\rho$-graph is therefore given by:

---

1 This was the observed number of nodes in the Candidate $\rho$-graph for all the 30 queries used in our experiments. Further investigation revealed that this was an artifact of the connectivity of our dataset.
Figure 5 shows that the Candidate $\rho$-graph containing $k$ paths obtained using our edge weighting schemes achieves between 80—90% of the score that can be achieved by choosing the top-$k$ ranked paths from the full graph (entire dataset of 30000 nodes and 45,000 edges). The Candidate $\rho$-graphs in our results typically contain 30-40% of the paths in the entire graph between the endpoints yet are 80-90% as “good” as the top paths in the entire graph between the two endpoints. It takes approximately a few hundred milliseconds to compute Candidate $\rho$-graphs. This satisfies our requirement of an interactive subgraph generator, but further timing comparisons with other algorithms would be interesting.

\[
Q(CGPaths_j) = \frac{\sum_{\text{score}(p_{\text{candidate}}))} \text{score}(p_{\text{candidate}})}{\sum_{FGRankedPaths} [\text{score}(p_{\text{candidate}})]} \tag{8}
\]

Measuring Display $\rho$-graph quality

Similar to Candidate $\rho$-graph quality, we compare the paths in the Display $\rho$-graph to the best paths in the entire graph. Let the set $\text{DPaths}$ represent the paths in the Display $\rho$-graph. The rank of a path in the Display $\rho$-graph is computed exactly the same way the rank of a path in the Candidate $\rho$-graph is computed, as is the score.

\[
\text{score}(p_{\text{display}}) = |\text{FGRankedPaths}| - \text{rank}(p_{\text{display}}) \tag{9}
\]

The quality of a display graph is computed by comparing its cumulative score to the best possible display that could be obtained from the ranked set of paths in the full graph. We refer to this best possible display as Pseudo-Display. In our experiments we use a budget of 100 nodes for our Display $\rho$-graphs. Starting with an empty...
Pseudo-Display graph and the path with rank 1 in the set $\text{FGRankedPaths}_\theta$ we add paths to the Pseudo-Display until 100 nodes have been added. The cumulative score of the Pseudo-Display is then computed as the sum of the scores of the paths. The quality of a Display $\rho$-graph is therefore given by:

$$Q(\text{DGPaths}) = \frac{\sum_{p_{\text{display}}} \text{score}(p_{\text{display}})}{\sum_{p_{\text{pseudo}}} \text{score}(p_{\text{pseudo}})}$$

Figure 6 shows that starting with the Candidate $\rho$-graphs with 80—90% quality the Display $\rho$-graphs computed capture a maximum of 84% of the score that can be obtained by taking the best paths in the full graph. Our results show the quality of Display $\rho$-graphs with respect to SemRank [11] to be surprisingly low – 43%. Further investigation of the methods used revealed that the difference between the ranking scheme in [22] and that in [11] is that in the former instance node degrees affect the rank of a path (nodes of lower degree being favored) whereas in the latter rank of path is determined purely by properties in the path. Our heuristics favor lower degree nodes and hence the observed trend. A personal communication with the authors of [11] revealed that extending SemRank to include the effect of nodes is an intended follow up to this work.

**Successive Display $\rho$-graph quality**

With the intention of validating the current flow model for subgraph relevance [7] we conducted the following experiment. We computed what we term as Successive Display $\rho$-graphs. To construct these displays we successively run the Display $\rho$-graph generation algorithm on the candidate graph. At each successive run we discount the paths used in previous displays. This results in the next best Display $\rho$-graph at every successive run. This process is repeated five times in our experiments to obtain five Display $\rho$-graphs. The current flow in each of these Display $\rho$-graphs is plotted rela-
tive to the current flow in the first Display \( \rho \)-graphs on a \textit{log} scale in Figure 7. The quality of these Display \( \rho \)-graphs is plotted relative to the quality of the first Display \( \rho \)-graph in Figure 8. There is a large difference both in the current flow and the display quality between the first display and the next display. This confirms that there is a correspondence between current flow in the Display \( \rho \)-graphs and their quality. This in turn supports the electricity based model for RDF graph relevance. Note that the plots below are averages of the relative differences of successive displays over all ranking schemes.

![Fig. 7. Current Flow in 5 Successive Display \( \rho \)-graphs relative to the best](image)

![Fig. 8. Quality of 5 Successive Displays relative to the best](image)

### 7. Conclusions and Future Work

Our results suggest that using edge weights generated by our weighting scheme results in highly relevant \textit{Candidate} \( \rho \)-graphs, where relevance is judged using estab-
lished path ranking metrics. Further evidence supporting this claim can be seen from quality of the Display $\rho$-graphs. The ranking metrics proposed by Aleman-Meza et.al. [22] in our experiments show that the quality of the Display $\rho$-graphs are best when using Class Specificity (CS), Instance Participation Selectivity (IPS) and Span together. Results for the Successive Displays serve to support the electricity flow based model for RDF subgraph relevance, besides validating our edge weighting schemes. Results presented in this paper seem very promising for application domains like Ontology based Scientific Discovery where the ability to visualize relevant relationships between metadata entities is crucial. As a follow up to this work we plan to apply our techniques to develop tools for finding correlations between Glycosylation patterns and patterns of gene expression within a cell line in the Glycomics [24] domain. We further propose to develop algorithms to support queries involving n endpoints for RDF graphs. Another interesting direction involves formalizing the notion of Context and investigating Context-Aware Subgraph Discovery algorithms.

8. Acknowledgements

We thank all SemDIS project members especially Boanerges Aleman-Meza for his insightful comments and revision suggestions. This project is funded by NSF-ITR-IDM Award#0325464 (SemDIS: Discovering Complex Relationships in the Semantic Web) and NSF-ITR-IDM Award#0219649 (Semantic Association Identification and Knowledge Discovery for National Security Applications).

9. References


[12] Deepayan Chakrabarti, Yiping Zhan, Christos Faloutsos: R-MAT: A Recursive Model for Graph Mining. SDM 2004


[16] Xifeng Yan, Jiawei Han. CloseGraph: Mining Closed Frequent Graph Patterns. In Proceedings of the 2003 Conference on Knowledge Discovery and Data Mining (SIGKDD2003), 2003.


Appendix

Comparison of Different query types

We differentiate our queries into two types: Inter-domain and Intra-domain. As the names suggest Inter-domain queries are those that seek paths that pass through instances that are classified as instances of classes belonging to different schemas. Intra-domain queries seek to find paths between instances of classes belonging to the same schema. All results presented thus far have been values averaged over 30 queries composed of 15 of each of the abovementioned types. In Figures A4 we present the Candidate ρ-graph quality Q(CG-Paths) for the Intra-domain and Inter-domain queries, in an attempt to gain insight into which of our weighting schemes work best for each query type.

Fig. A1. Candidate Graph Quality for Intra-Domain and Inter-Domain queries
Figure A1 shows peaks at the 2\textsuperscript{nd} and the 10\textsuperscript{th} combinations of settings of the four heuristics \textit{viz.} Instance Participation Selectivity (IPS the 2\textsuperscript{nd}) and Span + Instance Participation Selectivity (the 10\textsuperscript{th}). IPS favors rarer paths and since paths passing through multi-classified nodes are rarer, the 2\textsuperscript{nd} combination results in better quality Candidate ρ-graphs than other settings. Combining SPAN with IPS (10\textsuperscript{th} combination) results in better Candidate ρ-graphs. IPS and SPAN therefore are better settings to discover Inter-domain Display ρ-graphs.
Fig. A2. Schema for the Business Ontology

Fig. A3. Schema for the Entertainment Ontology

Fig. A4. Schema for the Sports Ontology
Fig. A5. Example of Display $\rho$-graph generated