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 knowledgebase 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]. Besides confirming that our edge weighting schemes work, we present results that confirm the electricity based [7] model for RDF graph relevance. Section 2 presents related work. In sections 3 and 4 we discuss our heuristics and algorithms respectively. This is followed by a discussion of the dataset for our experiments in Section 5. Section 6 presents our results and evaluations thereof. We finally conclude in Section 7 with a look at future research directions.

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 comprehendible 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: $C = \left\{ c \middle| \left\langle c, rdf\text{:}type, rdfs\text{:}Class \right\rangle \right\}$

$P = \left\{ p \middle| \left\langle p, rdf\text{:}type, rdf\text{:}Property \right\rangle \wedge \exists c, c' \in C \middle| c \in rdfs\text{:}domain(p) \wedge c' \in rdfs\text{:}range(p) \right\}$.

Further, we define an RDF data store $R = \langle \Pi, I \rangle$ 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 *rdfs:subClassOf* and *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 $d(p_H)$ 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

$$\mu(p_i) = \frac{d(p_i)}{d(p_{iH})} \qquad \mu(c_j) = \frac{d(c_j)}{d(c_{jH'})} \tag{1}$$

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 $\mu(r) = \max_{1 \le x \le k} \{\mu(c_x)\}$. 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.

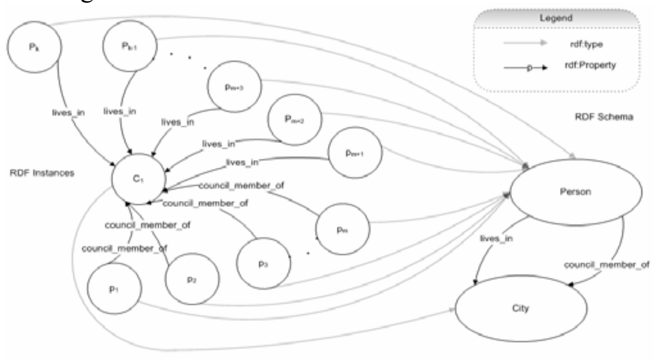


Fig. 1. Illustrative example for Instance Participation Distribution

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* $p_{m+1}, p_{m+2}, ... p_{k-2}, p_{k-1}, p_k$ 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 $\{p_{m+1}, p_{m+2}, ... p_{k-2}, p_{k-1}, p_k\}$. 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 *typeOf(s)*= C_i and *typeOf(o)*= C_j. Further, $|\pi|$ is thus the number of statements of type π 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 $\pi=\langle Person, lives\_in, City \rangle$ and $\pi' = \langle Person, council\_member\_of, City \rangle$. 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 $v_1, e_1, v_2, e_2 \ldots e_{n-2}, v_{n-1}, e_{n-1}, v_n$ from v_1 to v_n, where $v_i \in$ *Resources* and $e_i \in$ *Properties* $\forall i\ 1 \le i \le n$, 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 $\neg\exists(S|e_i,e_{i+1} \in S)$. 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 *Person* may be classified as both an instance of class *Actor* in the *Entertainment* domain and an instance of class *SpokesPerson* in the *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 *SchemaCover* = $\{S|\exists C \in S \wedge typeOf(v) = C\}$. The *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 *SDiff(u,v)* represent the number of new schemas seen as a result of traversing the edge *(u,v)*, where the value of *SDiff(u,v)* = *|SchemaCover(v)-SchemaCover(u)|*. The idea behind *SDiff* is to ensure that the discovery algorithm chooses a node that is in a "new" schema. However *SDiff* alone does not ensure that search will continues through the "new" schema. To combat this problem we define the *Cumulative Schema Difference* $CSDiff(u,u',v_i) = 1+SDiff(u, v_i) + SDiff(u', v_i)$ *for* $v_i \in adj[u] - \{u'\}$. We normalize this *Cumulative Schema Difference* (*CSDiff*) measure to compute a factor $\beta_{u' \to u \to v_i}$;

$$\beta_{u' \to u \to v_i} = \frac{CSDiff}{1+2(m-1)}, \text{ where } m \text{ is the number of schemas} \qquad \mathbf{(2)}$$

We then obtain the adjusted weight given by;

$$w'(u,v_i) = \beta_{u'\to u\to v_i} * w(u,v_i) \qquad \textbf{(3)}$$

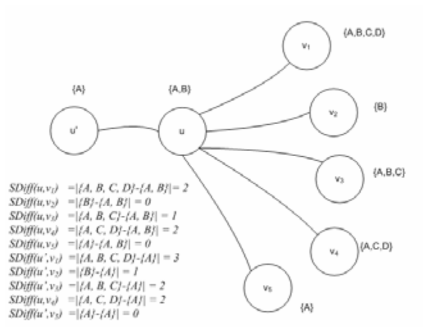

{A,B,C,D}
v1
{A}
{A,B}
v2
{B}
u'
u
v3
{A,B,C}
SDiff(u,v1) =|{A, B, C, D}-{A, B}|= 2
SDiff(u,v2) =|{B}-{A, B}| = 0
SDiff(u,v3) =|{A, B, C}-{A, B}| = 1
SDiff(u,v4) =|{A, C, D}-{A, B}| = 2
SDiff(u,v5) =|{A}-{A, B}| = 0
SDiff(u',v1) =|{A, B, C, D}-{A}| = 3
SDiff(u',v2) =|{B}-{A}| = 1
SDiff(u',v3) =|{A, B, C}-{A}| = 2
SDiff(u,v4) =|{A, C, D}-{A}| = 2
SDiff(u',v5) =|{A}-{A}| = 0
v4
{A,C,D}
v5
{A}

Fig. 2. Example of Span metric computation

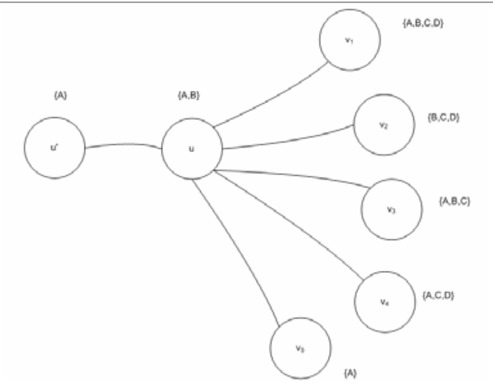
The effect of the factor β is to bias edge weights in the following way. Successor nodes that are instances of classes belonging to schemas other than those of the current and previous node are more likely to be visited, quantified by the two *SDiff* terms of *CSDiff*. More specifically, in the case of the example in Figure 2, a partial ordering is induced by the adjusted weights *w'(u, v*$_i$*)*, on the nodes as follows $v_1 \succ v_4 \succ v_3 \succ v_2 \succ v_5$. The node v_1 is therefore visited next. However, the measure β 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 $\beta_{u'\to u\to v_i}$; 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* $\alpha(u, v)$:

$$\alpha(u,v) = \frac{1}{2}\left(\frac{|SchemaCover(u)| + |SchemaCover(v)|}{m}\right), \text{ where } m \text{ is the number of schemas} \qquad \textbf{(4)}$$

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 $\beta_{u'\to u\to v_i}$ is computed at each step during the discovery process in contrast with the *a priori* values of edge weights computed using μ, σ and α as shown in equation 5. $\beta_{u'\to u\to v_i}$ is then used to adjust the weights $w(u, v_i)$ $\forall i\ 1 \le i \le n$ as shown in equation 3.

$$w(u,v) = \frac{\mu(p_{u\to v}) + \frac{1}{2}\left(\frac{\mu(u)}{degree(u)} + \frac{\mu(v)}{degree(v)}\right) + \sigma_\pi + \alpha(u,v)}{4} \qquad \textbf{(5)}$$

where $p_{u\to v}$ is the property connecting the resource node u and v, and π is the type of the statement $\langle u, p_{u\to v}, v\rangle$.



SDiff(u,v₁) $=|\{A, B, C, D\}-\{A, B\}|= 2$
SDiff(u',v₁) $=|\{A, B, C, D\}-\{A\}| = 3$
$CSDiff = 1+SDiff(u, v_1) + SDiff(u', v_1)$

$$\beta_{u'\to u\to v_1} = \frac{\text{CSDiff}}{1+2(m-1)}$$

$$\beta_{u'\to u\to v_1} = \frac{1+3+2}{1+2(4-1)} = 0.8571$$

SDiff(u,v₂) $=|\{B,C,D\}-\{A, B\}| = 2$
SDiff(u',v₂) $=|\{B,C,D\}-\{A\}| = 3$
$CSDiff = 1+SDiff(u, v_2) + SDiff(u', v_2)$

$$\beta_{u'\to u\to v_2} = \frac{\text{CSDiff}}{1+2(m-1)}$$

$$\beta_{u'\to u\to v_1} = \frac{1+3+2}{1+2(4-1)} = 0.8571$$

Since

$$\alpha(u,v_1) = \frac{1}{2}\left(\frac{|SchemaCover(u)| + |SchemaCover(v_1)|}{m}\right) = 0.75$$

$$\alpha(u,v_2) = \frac{1}{2}\left(\frac{|SchemaCover(u)| + |SchemaCover(v_2)|}{m}\right) = 0.625$$

And,

$$w'(u,v_i) = \beta_{u'\to u\to v_i} * w(u,v_i)$$

Because $w(u,v_i)$ is calculated using $\alpha(u, v_i)$

$w(u,v_1) \neq w(u,v_2)$

$$\therefore w'(u,v_1) \neq w'(u,v_2)$$

even if $\beta_{u'\to u\to v_1} = \beta_{u'\to u\to v_2}$

Fig. 3. Influence of the Schema Cover Factor α

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 ρ-graph* and *Display ρ-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 $X \xrightarrow{p_1} \ldots \xrightarrow{p_n} Y$.

Candidate ρ-graph generation algorithm

The *candidate ρ-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 $n \notin S$ and $n \notin T$ 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.

$$distance(u,v) = log\left(\frac{(degree(u) + degree(v))^2}{w(u,v) * \beta_{u' \to u \to v}}\right) \qquad \textbf{(6)}$$

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

Display ρ-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 Kirchoff'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 is step takes $\Theta(n^3)$ 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.

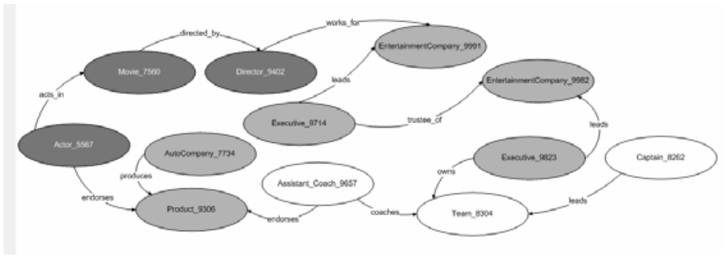


Fig. 4. Example snippet of a subgraph returned for the query ρ(Actor_5567, Captain_8262) on our synthetic dataset– Nodes in the above graph are color-coded according to the ontology their class belongs to

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 ρ-graph* and the *Display ρ-graph*. In our experiments the *Candidate ρ-graphs* generated contained 3000[1] nodes and the *Display ρ-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 end-points 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 ρ-graph* and *Display ρ-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 $FGPaths_9$ *(paths of up to 9 hops in the full graph).* There are therefore 30 distinct $FGPaths_9$ sets, one for each query in our experiments. We rank the paths in each of the $FGPaths_9$ 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 $FGPaths_9$ 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 $FGRankedPaths_9$. We therefore have three distinct scales ($FGRankedPaths_9$ sets) against which the quality of both *Candidate ρ-graph* and the *Display ρ-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 ρ-graph quality

To measure *Candidate ρ-graph* we compare the best paths in the entire graph to those in the *Candidate ρ-graph.* Let $CGPaths_9$ represent the set of paths in the *Candidate ρ-graph* with maximum length 9. For each path $p_{candidate} \in CGPaths_9$ we count the number of paths $p \in FGRankedPaths_9$ such that $rank(p) > rank(p_{candidate})$. This gives us the rank of each path in the *Candidate ρ-graph* with respect to all paths in the set $FGRankedPaths_9$. The score of a path is given by:

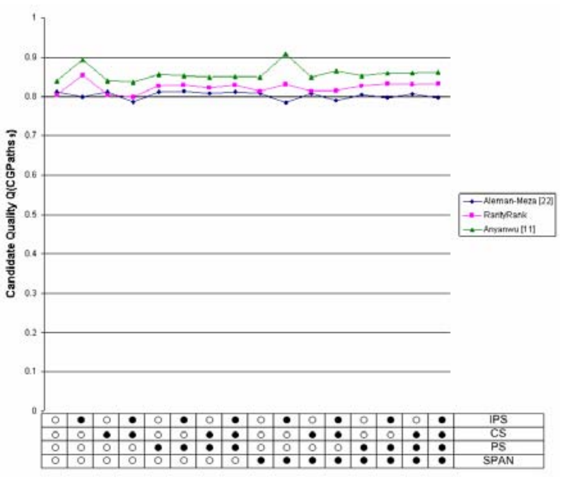
$$score(p_{candidate}) = |FGRankedPaths_9| - rank(p_{candidate}) \tag{7}$$

The quality of the *Candidate ρ-graph* is therefore given by:

[1] This was the observed number of nodes in the *Candidate ρ-graph* for all the 30 queries used in our experiments. Further investigation revealed that this was an artifact of the connectivity of our dataset.

$$Q(CGPaths_9) = \frac{\sum_{p_{candidate} \in CGPaths_9} \left(score(p_{candidate}) \right)}{\sum_{r=1}^{|CGPaths_9|} \left(|FGRankedPaths_9| - r \right)} \qquad \textbf{(8)}$$

Figure 5 shows that the Candidate ρ-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 ρ-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 ρ-graphs. This satisfies our requirement of an interactive subgraph generator, but further timing comparisons with other algorithms would be interesting.



○	●	○	●	○	●	○	●	○	●	○	●	○	●	○	●	IPS
○	○	●	●	○	○	●	●	○	○	●	●	○	○	●	●	CS
○	○	○	○	●	●	●	●	○	○	○	○	●	●	●	●	PS
○	○	○	○	○	○	○	○	●	●	●	●	●	●	●	●	SPAN

Fig. 5. Quality of the Candidate ρ-graph

Measuring Display ρ-graph quality

Similar to *Candidate ρ-graph* quality, we compare the paths in the *Display ρ-graph* to the best paths in the entire graph. Let the set *DGPaths* represent the paths in the *Display ρ-graph*. The rank of a path in the *Display ρ-graph* is computed exactly the same way the rank of a path in the *Candidate ρ-graph* is computed, as is the score.

$$score(p_{display}) = |FGRankedPaths_9| - rank(p_{display}) \qquad \textbf{(9)}$$

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 ρ-graphs*. Starting with an empty

Pseudo-Display graph and the path with rank 1 in the set *FGRankedPaths*$_9$ 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 ρ-graph* is therefore given by:

$$Q(DGPaths) = \frac{\sum_{p_{display} \in DGPaths} score(p_{display})}{\sum_{p_{pseudo} \in Pseudo-Display} score(p_{pseudo})} \qquad \textbf{(10)}$$

Figure 6 shows that starting with the Candidate ρ-graphs with 80—90% quality the Display ρ-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 ρ-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.

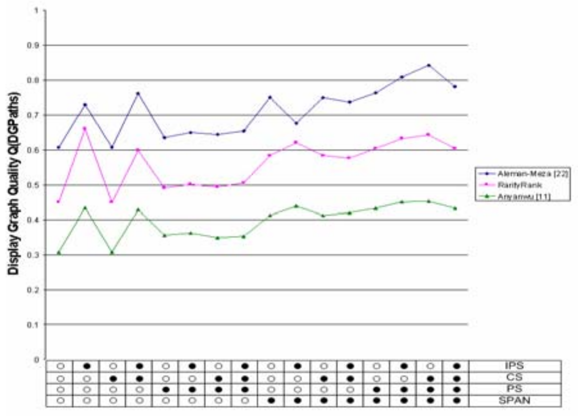


Fig. 6. Quality of the Display ρ-graph – Note that all weighting heuristics turned off results in poor graph quality in contrast with all heuristics turned on

Successive Display ρ-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 ρ-graphs*. To construct these displays we successively run the *Display ρ-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 ρ-graph at every successive run. This process is repeated five times in our experiments to obtain five Display ρ-graphs. The current flow in each of these Display ρ-graphs is plotted relative to the current flow in the first Display ρ-graphs on a ***log*** scale in Figure 7. The quality of these Display ρ-graphs is plotted relative to the quality of the first Display ρ-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 ρ-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.

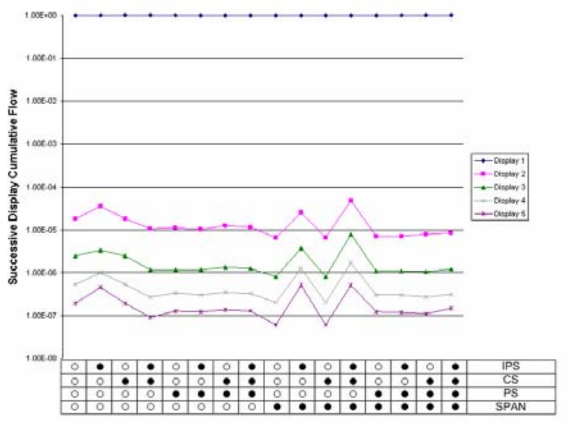


Fig. 7. Current Flow in 5 Successive Display ρ-graphs relative to the best

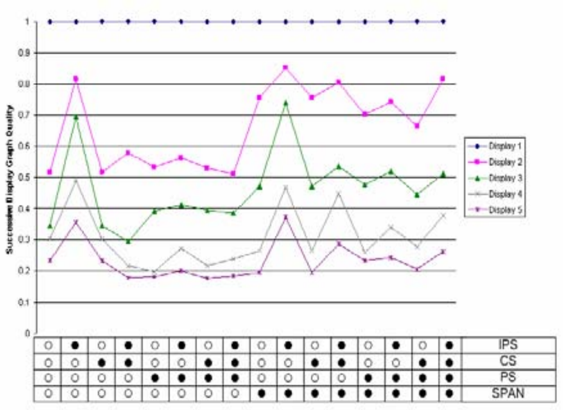


Fig. 8. Quality of 5 Successive Displays relative to the best

7. Conclusions and Future Work

Our results suggest that using edge weights generated by our weighting scheme results in highly relevant *Candidate ρ-graphs*, where relevance is judged using established path ranking metrics. Further evidence supporting this claim can be seen from quality of the *Display ρ-graphs*. The ranking metrics proposed by Aleman-Meza et.al. [22] in our experiments show that the quality of the *Display ρ-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

[1] T. Berners-Lee, J. Hendler, and O. Lassila, “The Semantic Web: A new form of Web content that is meaningful to computers will unleash a revolution of new possibilities”, Scientific American, May 2001.

[2] Kemafor Anyanwu, Amit P. Sheth: ρ-Queries: enabling querying for semantic associations on the semantic web. WWW 2003: 690-699.

[3] Ramanathan V. Guha, Rob McCool, Eric Miller: Semantic search. WWW 2003: 700-709.

[4] Stanley Milgram, "The Small World Problem", Psychology Today, May 1967. pp 60 - 67.

[5] Albert, R., and Barabási, A.-L., "Statistical mechanics of complex networks", Reviews of Modern Physics 7J (January 2002), 47-97.

[6] O. Lassila and R. Swick. Resource Description Framework (RDF) Model and Syntax Specification, W3C Recommendation. 1999.

[7] Christos Faloutsos, Kevin S. McCurley, Andrew Tomkins: Fast discovery of connection subgraphs. KDD 2004: 118-127.

[8] http://www.w3.org/TR/rdf-schema/

[9] G. Karvounarakis, S. Alexaki, V. Christophides, D. Plexousakis, M. Scholl, RQL: A Declarative Query Language for RDF, WWW2002, May 7-11, 2002, Honolulu,Hawaii, USA.

[10] Sougata Mukherjea, Bhuvan Bamba: BioPatentMiner: An Information Retrieval System for BioMedical Patents. VLDB 2004: 1066-1077

[11] Kemafor Anyanwu, Angela Maduko, Amit Sheth, SemRank: Ranking Complex Relationship Search Results on the Semantic Web. The 14th International World Wide Web Conference, (WWW2005), Chiba, Japan, May 10-14, 2005

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

[13] L. Page, S. Brin, R. Motwani, T. Winograd. "The PageRank Citation Ranking: Bringing Order to the Web", Stanford Digital Libraries Working Paper, 1998.

[14] Jon M. Kleinberg: Authoritative Sources in a Hyperlinked Environment. J. ACM 46(5): 604-632 (1999)

[15] Shou-de Lin, Hans Chalupsky: Unsupervised Link Discovery in Multi-relational Data via Rarity Analysis. ICDM 2003: 171-178

[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.

[17] Luke Huan, Wei Wang, Jan Prins, SPIN: Mining Maximal Frequent Subgraphs from Graph Databases. In Proceedings of the 2004 Conference on Knowledge Discovery and Data Mining (SIGKDD2004), 2004.

[18] Michihiro Kuramochi, George Karypis. Grew – A Scalable Frequent Subgraph Discovery Algorithm. In Proceedings of 2002 IEEE International Conference on Data Mining (ICDM), 2002.

[19] Gary Flake, Steve Lawrence, C. Lee Giles, Frans Coetzee. Self-Organization of the Web and Identification of Communities. IEEE Computer, 35(3), 66-71, 2002.

[20] David Gibson, Jon Kleinberg, Prabhakar Raghavan. Inferring Web Communities from Link Topology. In Proceedings of Ninth ACM Conference on Hypertext and Hypermedia, pages 225-234, New York, 1998.

[21] Jafar Adibi, Hans Chalupsky, Eric Melz and Andre Valente. The KOJAK Group Finder: Connecting the Dots via Integrated Knowledge-Based and Statistical Reasoning. In Proceedings of the Sixteenth Innovative Applications of Artificial Intelligence Conference (IAAI-04), 2004.

[22] Boanerges Aleman-Meza, Christian Halaschek-Wiener, I. Budak Arpinar, Cartic Ramakrishnan, and Amit Sheth. Ranking Complex Relationships on the Semantic Web. To Appear in *IEEE Internet Computing, Special Issue - Information Discovery: Needles & Haystacks May-June 2005.*

[23] B. Aleman-Meza, C. Halaschek, A. Sheth, I. B. Arpinar, and G. Sannapareddy, “SWETO: Large-Scale Semantic Web Test-bed”, In Proceedings of the 16th International Conference on Software Engineering & Knowledge Engineering (SEKE2004): Workshop on Ontology in Action, Banff, Canada, June 21-24, 2004, pp. 490-493.

[24] Amit Sheth, William York, Christopher Thomas, Meenakshi Nagarajan, John A. Miller, Krys Kochut, Satya S. Sahoo, Xiaochuan Yi, "Semantic Web technology in support of Bioinformatics for Glycan Expression," W3C Workshop on Semantic Web for Life Sciences, 27-28 October 2004, Cambridge, Massachusetts USA.

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$_9$) 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.

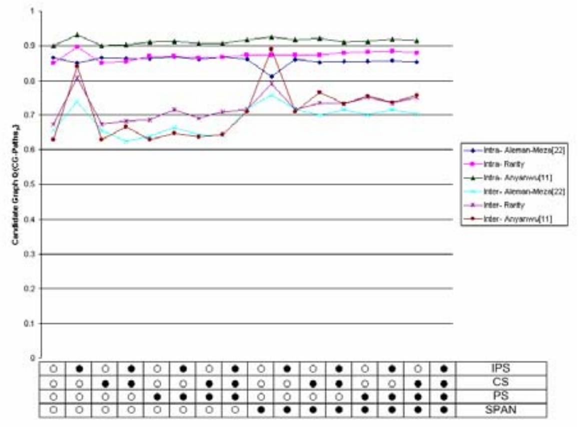


Fig. A1. Candidate Graph Quality for Intra-Domain and Inter-Domain queries

Figure A1 shows peaks at the 2nd and the 10th combinations of settings of the four heuristics *viz.* Instance Participation Selectivity (IPS the 2nd) and Span + Instance Participation Selectivity (the 10th). IPS favors rarer paths and since paths passing through multi-classified nodes are rarer, the 2nd combination results in better quality *Candidate ρ-graphs* than other settings. Combining SPAN with IPS (10th combination) results in better *Candidate ρ-graphs*. IPS and SPAN therefore are better settings to discover Inter-domain *Display ρ-graphs*.

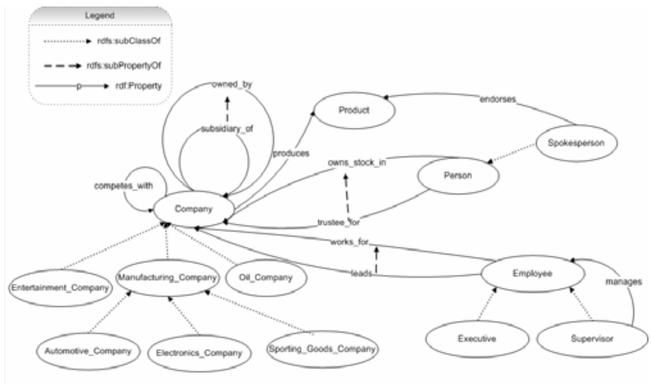


Fig. A2. Schema for the Business Ontology

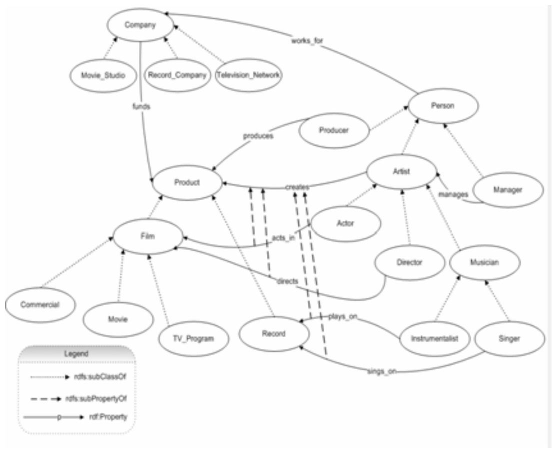


Fig. A3. Schema for the Entertainment Ontology

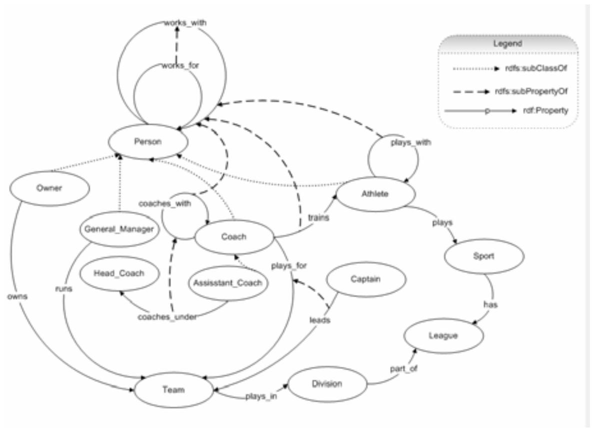


Fig. A4. Schema for the Sports Ontology

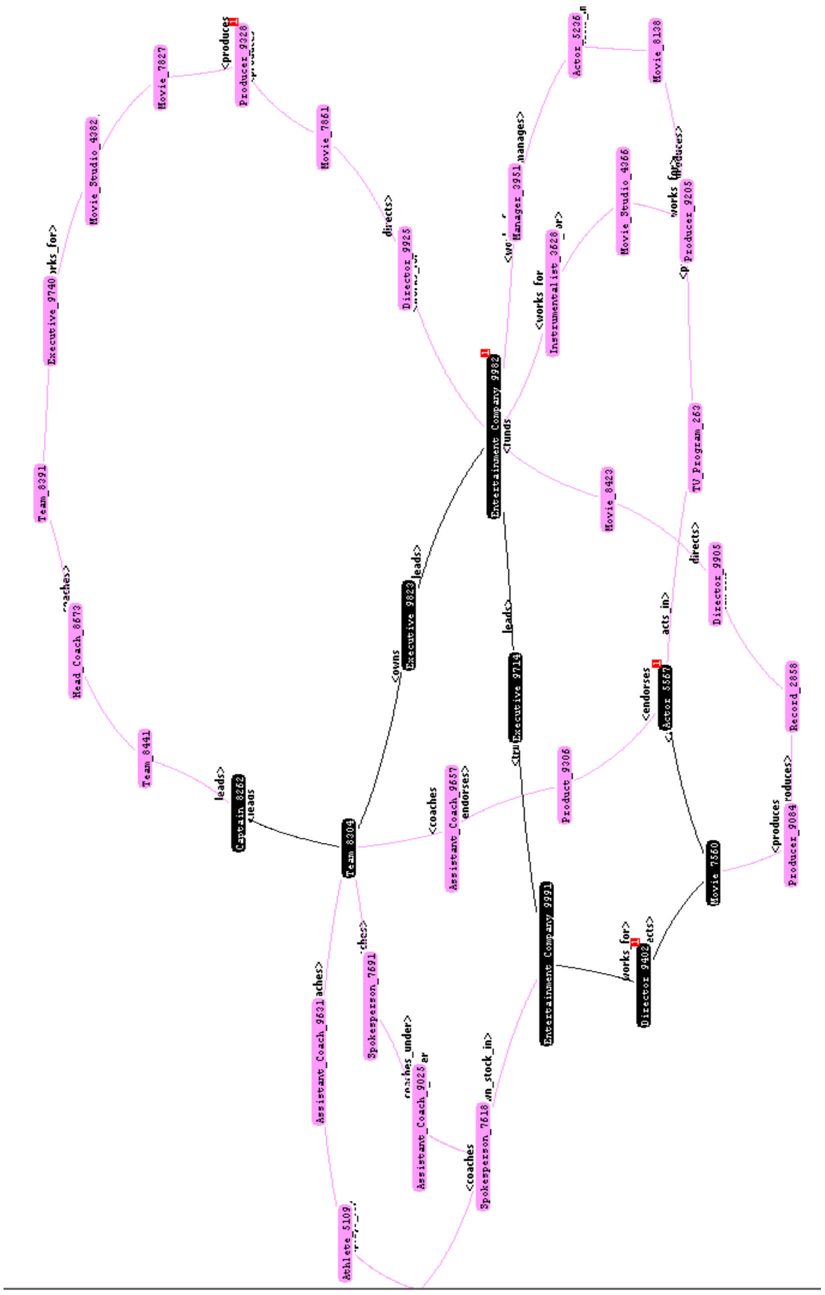


Fig. A5. Example of *Display ρ-graph* generated