Towards query model integration: topology-aware, IR-inspired metrics for declarative graph querying

Luiz Gomes-Jr
Institute of Computing – State University of Campinas (UNICAMP)
Campinas – SP – Brazil
gomesjr@ic.unicamp.br

Rodrigo Jensen
Faculty of Nursing – State University of Campinas (UNICAMP)
Campinas – SP – Brazil
rodjen@fcm.unicamp.br

André Santanchè
Institute of Computing – State University of Campinas (UNICAMP)
Campinas – SP – Brazil
santanche@ic.unicamp.br

ABSTRACT

Accompanying the growth of the internet and the consequent diversification of applications and data processing needs, there has been a rapid proliferation of data and query models. While graph models such as RDF have been successfully used to integrate data from diverse origins, interaction with the integrated data is still limited by inflexible query models that cannot express concepts from multiple paradigms.

In this paper we analyze data and query models typical of modern data-driven applications. We then propose an integrated query model aimed at covering a broad range of applications, allowing expressive queries that capture elements from diverse data models and querying paradigms.

We employ graphs models to integrate data from structured and unstructured sources. We also reinterpret as graph analysis tasks several ranking metrics typical of information retrieval (IR) systems. The metrics allow flexible correlation of data elements based on topological properties of the underlying graph. The new query model is materialized in a query language named in* (in star). We present experiments with real data that demonstrate the expressiveness and practicability of our approach.

Categories and Subject Descriptors

H.3 [**Information Storage and Retrieval**]: General; H.2 [**Database Management**]: Languages

General Terms

Languages

Keywords

Query model integration, graph data models, graph query languages

1. INTRODUCTION

Data integration has been a priority in the research agenda for many decades, producing a range of successful technologies. Data integration enables the development of applications that can explore and correlate a wider range of information. Most of the research was, however, developed in a scenario of low technological diversity. Therefore, the integration strategies could afford to rely on a single data model, such as the relational or multidimensional (for OLAP).

In modern technological environments, applications have to deal with different, often incompatible data and query models, which demand costly and time consuming ad-hoc integration solutions. For example, search engines must personalize search results based on structured data in user's profiles, decision support systems must analyze text from reviews to track user satisfaction over time, etc.

There has been initiatives to augment existing data and query models to support elements from other models, for example, keyword queries over relational data. These initiatives, however, typically have to compromise by choosing one query model among the integrated models, which constrains the interaction with the underlying data.

Here we tackle data integration from a query model perspective. We analyze modern data-driven applications and their requirements to develop an approach that covers a wide range of data models and, more importantly, query models. This not only allows applications to incorporate more relevant information, but also allows more expressive queries that combine elements from different querying paradigms. For example, consider the following queries:

- retrieve documents related to the keyword query "US elections" and the topic *politics*, written by democrat journalists, ranked by relevance to the keyword query and reputation of the author;
- retrieve employees relevant to a given project ranked by their reputation among peers;
- retrieve profiles of people over 30 years old, ranked by similarity of hobbies on their profiles to hobbies on mine;
- retrieve products not yet purchased by the client Bob that are relevant to him.

These queries cover a broad range of data models (e.g. unstructured documents, relational, graph) and applications (CMSs, social networks, recommendation systems). The queries also combine concepts from diverse query models, such as relational predicates, keywords, ranking, and metrics of relevance and reputation. These and similar queries appear in many situations in typical modern applications. Solutions for this type of information need in current infrastructures typically demand a substantial amount of resources and engineering to design ad-hoc subsystems.

To materialize our novel integration model, we adopt graph models as basis for data and query integration. For query model integration, we reinterpret as graph analysis tasks several querying concepts that are missing in current integration approaches. We implement this model in a new query language called in* (in star), which is an extension grammar for existing languages such as SPARQL[1] and Cypher[2].

Our approach allows flexible correlation of data in a simple and intuitive declarative query model. A key element of our model is the definition and integration of correlation metrics that rank data elements according to topological properties of the underlying graph. As we show along the text, our model can be employed in such diverse applications as CMSs, recommendation systems, social networks, diagnosis suggestion, lightweight decision support, etc.

This paper is organized as follows: Section 2 discusses aspects related to the use of graphs for data and query model integration. Section 3 presents the main technical components of our approach, describing our novel ranking metrics and their integration in a declarative querying setting. Section 4 describes experiments applied on real data that show the expressiveness and practicability of our approach. Section 5 describes and contextualizes related work. Finally, Section 6 concludes the paper.

2. EMPLOYING GRAPHS FOR DATA AND QUERY INTEGRATION

The first step towards a more expressive query model, that covers a wider range of applications, is to analyze the typical data our applications are dealing with today. We focus on data-driven applications for our analysis, which are the ones that would have the most benefit from an integrated query model. Other types of applications and data could also be integrated in our model, but we will not address them here.

Figure 1 shows the structuring spectrum of typical data in current data-driven applications. Highly structured data, on the top right of the figure, was the first type of data tackled by database technology. As we increased computational power and made technological advances, systems started to incorporate more data from the spectrum.

Graph models, due to their simplicity and flexibility, can represent data all along the structuring spectrum. Notably, graphs have been used to model relationships in text [15, 3] and there are several initiatives to model relational databases as graphs [1, 2]. Furthermore, graphs are often the most natural and widely adopted model for several applications, such as social, semantic or spatial networks.

Here we adopt the graph model for data integration. We employ typical property graphs [16] to represent the data as in Figure 1. The following section provides mode details on the mapping between source models to graphs.

[1]http://www.w3.org/TR/sparql11-query/
[2]http://docs.neo4j.org/

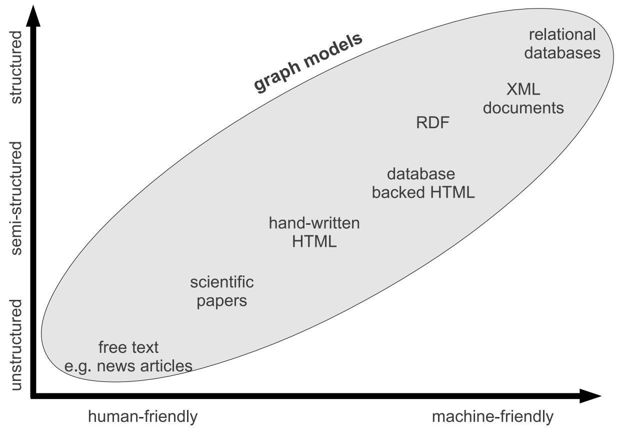

Figure 1: Spectrum of structuring of data

2.1 Data model integration

There are several alternatives for mapping a given data model into graphs. Although our query model works independently of the strategy adopted, we provide guidelines on basic transformations of typical models. Here we focus on text documents and the relational model. The mapping for other models, such as semi-structured or NoSQL variations, can be derived by similar approaches.

There are several alternatives for mapping a relational scheme to graphs [1, 2]. There is even a W3C working group[3] to define standards for relation to RDF mapping languages. Here, to simplify the discussion, we assume that (i) table descriptions become nodes, (ii) rows also become nodes, with their primary keys as identifiers, and are linked to their respective tables, (iii) columns become properties of the instances, with values corresponding to literals and foreign keys becoming explicit links to other instances.

Graph representation of documents for IR purposes is also possible. An inverted index (in the bag of words model) can be readily mapped into a graph that connects terms and documents. More modern schemes to index documents such as topic models [4] and explicit semantic analysis [14] also fit nicely into this strategy, bringing the benefits of reduced dimensionality (i.e. avoiding creating an unnecessarily large graph containing entire postings list), less semantic ambiguity, and more cognitive appeal.

Figure 2 shows a simplified example to illustrate all these elements represented as a unified graph. News articles about products are mapped into entities according to a given IR indexing/annotation technique (e.g. topic modeling, named entity recognition, etc). A keyword query is likewise mapped into these entities. Relational data from tables (Project, Employee) are also mapped into nodes in the graph and also connected to the entities. In the remaining of the paper we stop distinguishing between structured and unstructured data, assuming the data models are integrated in the unifying graph.

2.2 Need for language model integration

Data model integration is certainly an important step towards supporting applications that reuse and combine information from multiple sources. The benefits are, however,

[3]http://www.w3.org/2001/sw/rdb2rdf/

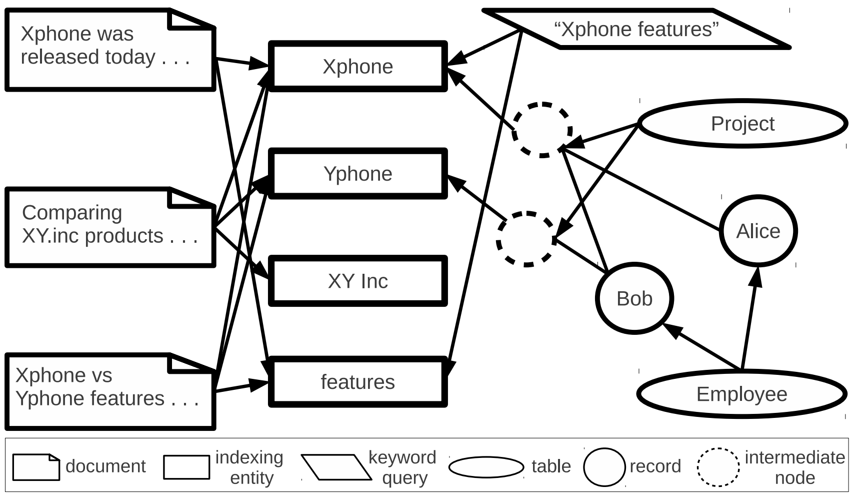

Figure 2: Data elements represented as a unified graph

limited by the capabilities of the query model adopted. For example, if the query language is more adequate for structured data, it would be hard to express information needs related to unstructured information.

In our proposal, we make a top-down analysis of the data models, language elements and applications we intend to support in our query model. We depart from the observation that graph models are powerful means to integrate data across the structuring spectrum. Our goal is to propose a query model that increases expressiveness in the interactions with the integrated data, supporting a wider range of applications.

Query integration enables more expressive querying because (i) it allows users to correlated data according to concepts that are consistent with the specific model of the data sources, (ii) it provides a new range of querying concepts that can be used over data from previously incompatible sources, and finally, (iii) it supports a mix of the previous points, allowing combinations of data from different original models to be queried based on combinations of diverse query concepts that would otherwise be restricted to specific models.

For example, considering the data on Figure 2, an integrated query model would allow relevance-based ranking of document nodes (i). It would also allow these concepts, which are usually associated with IR applications, to be used to query the structured data about employees and projects (ii). A query could also take advantage of a combination of data from documents and relations, and query concepts from IR and structured databases (iii).

In the next section we discuss the requirements for such integrated query model.

2.3 Query model requirements

Here we highlight the requirements for a query model that would cover a broad range of data and interaction models. First, we start with general high-level (conceptual) requirements for the query model.

Expressiveness: this is the main goal for our query integration model. To offer meaningful ways to correlate the integrated data, the model must offer means to express concepts from such distinct fields as databases and information retrieval. Our model offers means to express querying concepts from structured and graph databases, as well as concepts from IR systems.

Simplicity: an integrated query model should not resemble ad-hoc solutions. The integrated concepts from a given query model (e.g. relational) should be applied to data integrated from other models (e.g. unstructured documents) in a natural and coherent manner. It is also important to focus on the most relevant query models and most important concepts in them. Here we emphasize concepts from relational and graph databases and information retrieval systems. We do not integrate concepts from other important models such as inference, because we think they are still not as prevalent in current applications.

Non-modal interaction: the integrated model should blend concepts from each area to a point where there is no modal user interaction – there is no need to differentiate tasks, data formats or query strategies according to the original model of the data.

Gentle learning curve: to simplify learning and adaptation, our approach extends existing query languages, as opposed to creating an entirely new one. This has the advantage of leveraging grammars and processing technologies that are already mature and also simplifies employment in current applications.

We now analyze specific requirements for the elements of the language.

Declarative Querying: declarative queries empower users to express their information needs precisely, and the results are returned in a predictable format. The processing in-between is just as important: declarative queries enable the system to transparently optimize data access and computation strategies. Therefore, declarative querying should be a key element in an integration model.

Selections/projections/aggregations: these are basic components of structured query languages. Any proposal for query model integration has to provide this type of interaction. Our strategy inherits these constructors from the original language to which our extension is applied.

Graph patterns: since we integrate our underlying data in a unified graph, it is important to provide means to express topological constraints on queries. Graph patterns are adopted by most graph query languages. Like previously, our strategy is to leverage graph querying features from existing languages, like SPARQL or Cypher.

Keyword querying: keywords are the prevalent mean to query unstructured data in typical applications. In our framework, a keyword query is also represented as a (temporary) node in the graph. The same indexing strategy used for the stored documents is applied to generate the relationships of the query node (Figure 2). This graph representation of keyword queries allows them to be expressed alongside structured predicates in the queries (Section 3.3). This is equivalent to typical IR systems with the added benefit of query-time specification of ranking metrics (defined next), and direct access to other constructs of our model.

Ranking: to enable the flexibility required by many modern applications, such as document retrieval and recommendation systems, an integrated query model should support inexact results, providing a ranking mechanism that orders returned items according to conformity to the user's information needs. In our approach, the ranking scores are calculated based on combinations of our ranking metrics.

Flexible correlation of data elements: The IR field has been very successful in offering simple but efficient means for users to input their information needs and to get sensible results back. The key to the success of such systems and what makes a search engine or a recommendation system a market leader is the profound ways in which the systems correlate the underlying data. Abstract ranking metrics such as relevance and reputation are powerful, flexible and recurrent in several tasks, but at the same time are made intuitive to use and easy to understand. In our proposal, we introduce a novel interpretation of several ranking metrics inspired by current applications. These metrics are mapped into graph analysis tasks to adapt to our data integration framework (Section 3.1). These metrics are then integrated in our extensional query language (Section 3.3).

Here we propose a new query model that takes all the discussed characteristics into account: providing a declarative query language that can express concepts form traditional IR and Database systems, and compose results (optionally) as ranked lists. The challenge is to enable all these features over the unified graph model presented.

Many of the listed requirements are met by query languages such as SQL and, especially, modern graph query languages such as SPARQL and Cypher. The remaining issues are related to enabling IR-like ranking metrics that now have to be reinterpreted in a graph setting. In the next section we define our interpretation of the ranking metrics which represent a central piece towards bridging the gap between declarative, structured queries and flexible correlation of data.

3. RANKING METRICS AND LANGUAGE INTEGRATION

Correlating data is an important and defining characteristic of many applications. To enable a high level of flexibility for correlations, we specify a set of ranking metrics which are influenced by IR applications. The selection of the specific metrics aims at covering a wide range of applications while also being simple to use and understand. In the process of defining these metrics, we started with some popular metrics used in IR and then expanded the set according to the applications we wanted to cover. The set of metrics we ended up with can be organized in the taxonomy presented in Figure 3.

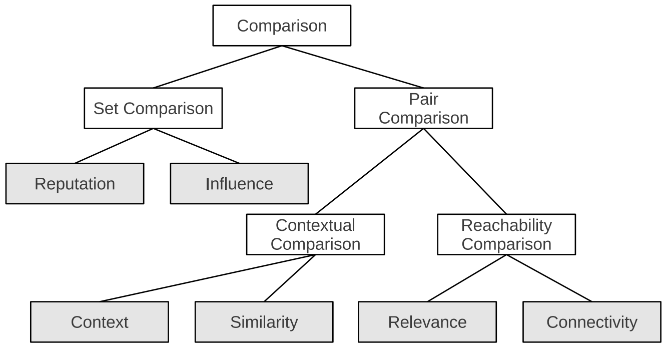

Figure 3: Taxonomy for the adopted ranking metrics

The basis of our taxonomy is the concept of comparison. Our metrics are meant to compare elements in the graph and generate a score that represents the strength of the association. The peculiar aspect about our metrics is that the scores are generated based on analysis of the topology of the graph, in contrast to most ranking approaches that are based on attributes of the elements.

There are two main groups of comparisons. Set comparisons corresponds to comparisons among elements from a finite set. Reputation and Influence are the metrics in this category. They assess, using different strategies, how well a node performs as a hub for information. The definitions of the metrics, as well as details on their interpretation, are presented in the next section.

Pair comparisons are applied to individual pairs of nodes. They assess properties of the topology surrounding or connecting the two nodes. The similarity and context metrics, classified under contextual comparison, assess the commonalities in respect to elements (nodes or relationships) surrounding the comparing nodes. Relevance and connectivity, classified under reachability comparison, assess properties of the paths interconnecting the comparing nodes.

As far as we know, this is the first time that these metrics are considered and defined under the same conceptual framework. Although these metrics are often associated with IR, they express cognitive processes or patterns that we use to assess correlation of entities in the real world, and which are the basis of many data-driven applications, as we intend to portray along the text. We now describe our metrics and define them from a graph analysis perspective.

3.1 Graph interpretation of the metrics

The translation of the ranking metrics to the unified graph strategy is a challenging task. Here we adopt a Spreading Activation (SA) [6] model for our novel interpretation of the metrics.

The Spreading Activation model

Spreading Activation methods were developed to infer relationships among nodes in associative networks. The mechanism is based on traversing the network from a initial set of nodes, activating new nodes until certain stop conditions are reached. By controlling several aspects related to this activation flow, it is possible to infer and quantify the rela-

notation	description
$SA(N)$	a set of activated nodes after the execution of the spread activation process defined by parameters G, N, I, O, t, d, c, l; parameters other than N are omitted for brevity
$SA(N)_n$	$n \in SA(N)$
G	unified data graph
N or M	set of initially activated nodes. n or m represent nodes from the respective sets
$I(n)$	function that calculates the input potential of a node. $I(n) = \sum_{i \in in(n)} O(i)$ in the general case
$O(n)$	function that calculates the output potential of a node. $O(n) = I(n) * d$ in the general case
$in(n)$	set of nodes with outbound edges linked to n
$out(n)$	set of nodes linked by outbound edges from n
a, t, d, c	respectively, initial activation potential, firing threshold, decay factor, maximum number of iterations (depth)
l	set of labels that determine valid nodes for traversal
$v(n)$	final potential value for node n
$p(N)$	set of activation paths (for each node in N)

Table 1: Notation used in the definitions

tionships of the initial nodes to the reached ones.

The SA model used here is defined by the parameters G, N, I, O, a, t, d, c, and l described, alongside other definitions, in Table 1. A SA process starts with the N nodes initially activated with potential a. Output potentials for each node are calculated by the function O. The output potential is spread through all edges whose labels are in l. The potential for the reached nodes is calculated by function I. For the next iteration, the potential is spread, restarting the process, as long as the current potential for reached nodes is higher than t and the number of iterations is lower than c.

IR metrics according to the SA model

In the SA model, to assess the rank of the relationship of nodes according to a metric, an activation potential is placed at the target elements defined in the query. The potential is spread across the topology of the graph, losing or gaining strength based on the IR metric, length of the path, or properties of the traversed elements. The metric-specific definitions of the SA processes are presented below.

Def. 1. $relevance(n, m) = v(SA(\{n\})_m)$, with $O(n) = \frac{I(n) * d}{|out(n)|}$

Relevance between two nodes is a measure that encompasses correlation and specificity. Correlation is proportional to the number of paths linking the two nodes and inversely proportional to the length of the paths. Specificity favors paths with less ramifications. It is easy to observe that traditional tf*idf weighting over data as in Figure 2 is an instance of this definition (for trivial paths of length one).

Def. 2. $connectivity(n, m) = v(SA(\{n\})_m)$

Connectivity between two nodes is a measure that assesses how interconnected two nodes are. The score is proportional to the number of paths linking the nodes in the network activated by the SA algorithm.

Def. 3. $reputation(n, N) = v(SA(N)_n)$

Reputation of a node measures how effective it is as a hub for information flow. Here the nodes of interest are activated at the beginning and the ranking scheme favors nodes that are revisited in the sequence of the SA process. This is a simple but convenient interpretations in scenarios where the reputation cannot be pre-calculated due to high update rates, variability in the types of relationships used for the queries, or need to bias the scores based on a set of initial nodes (as in [21]).

Def. 4. $influence(n) = |(SA(\{n\}))|$

Influence is a specialization of reputation where the only concern is the number of nodes reached from the origin. The topology of the graph – in/outdegree or cycles – do not influence the metric.

Def. 5. $similarity(m, n) = \frac{|p(SA(\{n\})) \cap p(SA(\{m\}))|}{|p(SA(\{n\})) \cup p(SA(\{m\}))|}$

Similarity measures the ratio of common relationships (same edge label linking common nodes) between two nodes.

Def. 6. $context(m, n) = \frac{|SA(\{n\}) \cap SA(\{m\})|}{|SA(\{n\}) \cup SA(\{m\})|}$

Context is a specialization of similarity where edge labels do not matter.

3.2 Semantics of ranking metrics in queries

Having the ranking metrics interpreted as graph analysis tasks, there is now the need of integrating these metrics in a declarative language. As opposed to creating an entirely new query language, we decided to leverage existing languages by defining an extension language that can be integrated into other languages. To that extent, we first define the semantics of the intended integration.

In our model, the proposed ranking metrics are intended to be used with graph query languages that offer: (i) means to reference individual nodes in the graph, (ii) selection of match variables, and (iii) query results as a set of tuples (or a graph representation of). These are basic components of graph languages like SPARQL and Cypher. A ranking metric can refer to:

- a single match variable (set of vertices), e.g."rank papers from EDBT 2012 according to first author *reputation*", where first author is the match variable in question (e.g. "SELECT ?firstAuthor ..." in SPARQL);
- a given vertex[4] and a match variable, e.g. "rank papers according to *relevance* of their first author (match variable) to the topic data integration (vertex)";
- two match variables, e.g."rank papers according to *relevance* of the first author to the topic in the first keyword of the paper".

Conceptually, the ranking metrics are applied to query results, generating a ranking value for each returned tuple. In practice, to speed up query processing, results would be approximate and the rank would be generated for some of the nodes based on access pattern heuristics.

3.3 Extending Declarative Queries

A convenient way to integrate the ranking metrics into existing query languages is to add a "RANK BY" clause. The

[4]as defined previously, a keyword query would also be a node in the graph

```
1  ExtendedQuery::=RegularQuery RankClause
2  RankClause::='RANK BY' RankMetric ( ',' RankMetric )*
3  RankMetric::= Weight? ( UnaryRMetricDesc | BinaryRMetricDesc )
4  UnaryRMetricDesc::=UnaryRankMetric  'OF' MatchVariable
5          Modifiers*
6  BinaryRMetricDesc::=BinaryRankMetric 'OF' MatchVariable 'TO'
7          ( MatchVariable | Vertex | KWQuery) Modifiers*
8  UnaryRankMetric::=Reputation | Influence
9  BinaryRankMetric::=Relevance | Connectivity | Similarity | Context
10 Modifiers::=Follow | Depth | Direction | Weighted
11 Follow::='FOLLOW' EdgeSet
12 Depth::='DEPTH' INTEGER
13 Direction:=='DIRECTION' ( 'INBOUND' | 'OUTBOUND' | 'BOTH' )
14 Weighted:=='WEIGHTED'
15 KWQuery:=='KWQUERY' '(' String ')'
16 EdgeSet:=='(' Edge ( ',' Edge )* ')' . . .
```

Figure 4: Simplified BNF grammar for the proposed extension (terminators omitted)

clause should enable an arbitrary combination of metrics that expresses the global ranking condition defined by the user. We encode the clause in the extension query language that we denominated in* (or in star). in* can then be used to extend other languages, for example, extended SPARQL becomes inSPARQL. This strategy is a good fit for graph languages with SQL-inspired syntaxes, such as SPARQL and Cypher. A similar strategy could be developed to other types of languages.

Note that the extension causes query semantics and result interpretation to change, therefore, any extended language would be more adequately described as new language based on the syntax of the original language. This suggests an incidental meaning for an acronym like inSPARQL: recursively, "inSPARQL is Not SPARQL".

Figure 4 shows a simplified BNF grammar of the proposed extension. A ranking can be specified as mix of weighted ranking metrics (lines 2 and 3). Weights capture the relative importance of each metric. The scores generated by the metrics are normalized before the calculation of the final weighted score.

Ranking metrics are unary or binary. Unary ranking metrics are applied to a single match variable (lines 4 and 5). Binary ranking metrics can be applied to a match variable and a named vertex or between two match variables.

The language allows for modifiers (lines 10 to 14) to be applied to the ranking definitions. These modifiers define the parameters for the execution of the SA algorithm. FOLLOW specifies valid edges for the algorithm to traverse. DEPTH defines the maximum length for the traversal paths. DIRECTION sets the direction of traversal as outbound, inbound or both (default) edges. WEIGHTED makes edge weights influence the degradation of the activation potential (the potential is multiplied by the weight).

3.4 Use case: extended SPARQL

This section presents examples of queries in the extended SPARQL language. These queries are meant to demonstrate the expressiveness of the approach in a wide range of applications.

Figure 5a shows a product recommendation query that finds products that the client Bob (with uri :bob) has not purchased. The query traverses Bob's friendship network to find products purchased by his friends that might be relevant to him. The spreading activation interpretation of this query evaluation also implies that products purchased by Bob, even though they do not appear in the results, will be traversed on the way to customers that have co-purchased these products, which in turn will activate other products from these customers.

Figure 5b shows a query that could be used on an online dating application. It ranks the top 5 persons over a given age based on the similarity of hobbies and movie preferences of user Alice.

Figure 5c ranks species that play an important role in the food web and are related to the biome of coral reefs. This type of query would identify species that should be main targets for monitoring and preservation efforts.

Figure 5d shows a possible implementation for a document retrieval query using topic modeling. The keyword query is expressed by the function KWQUERY and the relevance is assessed as if the query was a node in the graph. The query also takes into account the relevance of documents to the topic :Politics and the reputation of the authors.

```
PREFIX movie: <http://data.linkedmdb.org/resource/movie/>
PREFIX director: <http://data.linkedmdb.org/resource/director/>
SELECT DISTINCT ?actor WHERE {
  ?film movie:director director:8501 .
  ?film movie:actor ?actor .
  ?film movie:initial_release_date ?date .
  FILTER ( fn:starts-with(?date, "199") ) }
RANK BY RELEVANCE OF ?actor TO director:8501
```

Figure 6: Baseline query for the experiments

4. EXPERIMENTS

Here we present experiments that demonstrate (i) the adequacy of the results in a query that correlates non-trivial elements of a movies database and (ii) the interplay of ranking metrics in a real scenario based on a nursing diagnose query. The first set of experiments were implemented over a Virtuoso Triplestore[5] server and used extended SPARQL queries. The second set of experiments employed a Neo4j[6] graph database and extended Cypher queries.

4.1 Relevance metric

Here we present analysis of execution for a basic query containing our relevance metric. The database used in the experiments is the Linked Movie Data Base (LinkedMDB) [8]. The database integrates data from several sources (FreeBase, OMDB, DBpedia, Geonames, etc). The database contains 3,579,616 triples.

The query used for our analysis is shown in Figure 6. The query returns actors that acted in films directed by Woody Allen in the 90's. The results are ranked by relevance of the actors to Woody Allen (director). This query should be interpreted as ranking actors according to how linked to the director their careers are – a common pattern throughout Allen's idiosyncratic production.

The top-10 and bottom-10 ranked performers are shown in Table 2. The analysis of the results reaffirms that the graph interpretation of relevance proposed here is indeed strongly correlated to the typical interpretation of relevance in IR applications. The top ranked actor is Woody Allen himself[7]. Allen is well known for interpreting roles in his

[5]http://virtuoso.openlinksw.com/

[6]http://www.neo4j.org/

[7]LinkedMDB uses distinct descriptors for the actor and the

```
SELECT DISTINCT ?product                                        a
 WHERE { ?product :type :Product .
   FILTER NOT EXISTS (:bob :purchased ?product) }
RANK BY RELEVANCE OF ?product TO :bob
 FOLLOW (:friendsWith, :purchased)
 DEPTH 3 DIRECTION BOTH
```

```
SELECT ?person
 WHERE { ?person :hasGender 'M' .                               b
        ?person :hasAge ?age .
        FILTER (?age > 30)}
LIMIT 5
RANK BY SIMILARITY OF ?person TO :alice
 FOLLOW (:hasHobby, :favoriteMovie)
```

```
SELECT ?species
 WHERE { ?species :type :MarineSpecies }                        c
RANK BY
 3 INFLUENCE OF ?species FOLLOW :preysOn DIRECTION OUTBOUND,
 1 RELEVANCE OF ?species TO :CoralReefBiome
```

```
SELECT ?document, ?author
 WHERE { ?document :type :BlogPost }                            d
RANK BY
 2 RELEVANCE OF ?document TO KWQUERY(``US elections''),
 1 RELEVANCE OF ?document TO :Politics,
 3 REPUTATION OF ?author
```

Figure 5: Examples of extended SPARQL queries (namespaces have been omitted)

films, and he rarely performs in films from other directors. Mia Farrow, the second highest rank, has her career strongly linked to the director[8], acting in 13 of Allen's films, out of her total of 39 films registered in the database.

Less known actors also appear in the top-10 list. Hazelle Goodman, for example, has only one performance recorded in the database, which would make her career highly linked to Woody Allen (she is credited in IMBD as the first person of Black origin to have a major role in a Woody Allen film). The database does not have complete castings for the movies, especially for small roles, which should not affect ranking for most practical applications.

Low ranking actors are usually actors that participated in many films but few of them were directed by Woody Allen. This is the case for Robin Williams and Sean Penn, for example, which perform in only one of Allen's films. The interpretation is that low ranked performers would by no standards have their careers linked to Woody Allen, despite having been cast in their movies.

top 10	name	bottom 10	name
1.89	Woody Allen	0.03	Stanley Tucci
0.79	Mia Farrow	0.03	William Hurt
0.59	Tony Darrow	0.03	Dom DeLuise
0.53	Julie Kavner	0.03	Kathy Bates
0.50	Brian Markinson	0.03	John Malkovich
0.40	Hazelle Goodman	0.03	Anthony LaPaglia
0.39	Diane Keaton	0.03	Sean Penn
0.29	Mayim Bialik	0.02	Uma Thurman
0.29	Ted Bessell	0.02	Donald Pleasence
0.28	Tracey Ullman	0.02	Robin Williams

Table 2: Top-10 and bottom-10 ranked results for the baseline query (total of 98 returned actors)

The experiments indicate that our approach is practical in terms of performance, a major concern with the type of graph analysis involved. The parameters in our SA model (t, d, c, and a) ultimately determine how far the activation process would go in its exploration of the graph. This has consequences in terms of performance and completeness of query execution. Relaxed values for the parameters, which would allow bigger portions of the graph to be included in the query, have expensive computational requirements, but render more contextualized ranking that might include

```
START diag=node(*)
  WHERE has(diag.Type) and diag.Type = "Diagnose"
  RETURN diag
RANK BY
  %r RELEVANCE OF diag TO node( %p ) WEIGHTED,
  %c CONNECTIVITY OF diag TO node( %p ) WEIGHTED
```

Figure 7: Query template for the diagnosis experiments (in Cypher)

non-obvious aspects of the correlation of the elements in the query.

Execution times for the metric in the baseline query took under 3 seconds. We think this would already be an interesting achievement for the non-trivial query, database, and computations we are dealing with, but there is room for radical improvements. A deeper analysis of these experiments as well as ongoing efforts for performance improvements are presented in [7].

4.2 Combining metrics

We tested our approach over real diagnose data compiled for a previously unrelated project with the faculty of nursing at our university [10]. The data consists of matrices SD (for Symptoms × Diagnoses), PS (for Patients × Symptoms) and PD (for Patients × Diagnoses). SD correlates 62 symptoms to 28 diagnoses, all defined in a domain-specific protocol. The strength of the correlations $sd_{i,j} \in \{0, 0.25, 0.5, 0.75, 1\}$ were determined based on consensus in committees of experts. Similarly, PS correlates 6 patients (clinical cases) and their symptoms. The strength of the correlations $ps_{i,j} \in \{0, 0.25, 0.5, 0.75, 1\}$ were also determined by experts. Experts also provided most likely diagnoses for each patient, with strength $pd_{i,j} \in \{0, 0.25, 0.5, 0.75, 1\}$ (these were not included in our test dataset).

The matrices were consolidated in a graph with the strength of the correlations becoming weights for the edges. Figure 8 shows a sample of the graph with the weights of the associations represented as the thickness of the edges. As an example, the patient represented by the node 'Case 1' presents the symptom 'Fatigue' that was classified as having a 1.0 pertinence to the diagnosis 'Caregiver Role Strain' and 0.5 for 'Risk for Falls'. The final graph has 96 nodes and 324 relationships. Our goal was to suggest diagnoses based on the assessment of our metrics in the graph. The template for the queries issued for each patient is shown in Figure 7. $\%r$ and $\%c$ are weights for the relevance and connectivity metrics, $\%p$ is the id of the node of the patient. To repre-

director, implying that they are separate entities

[8]note that non-professional interactions are absent from the database

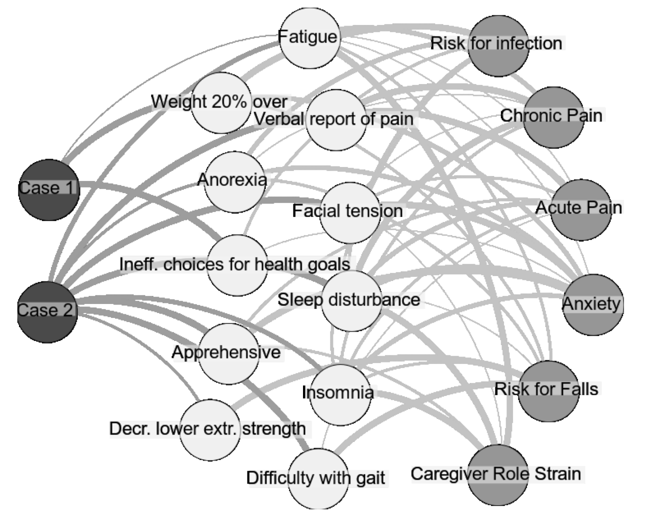

Figure 8: Sample subgraph of the dataset for the diagnosis experiments

sent an instance of the query template we use the notation Rr:cC, meaning that the query was specified with weight r for the relevance metric and weight c for connectivity.

For an assessment of the general rankings produced, we asked an expert to evaluate the entire ranks for each patient and assign a pertinence value in $\{0, 0.25, 0.5, 0.75, 1\}$. To produce the rankings, we used the metrics for relevance and connectivity in isolation, i.e. runs R1:0C and R0:1C. On average, the rankings from relevance received a 0.79 quality score while the connectivity metric received 0.67.

We also analyzed the rankings based on the diagnoses suggested by the committees of experts. For that, we used the diagnoses with total correlation (1) to each patient. We run the queries to assess, for the sets of diagnoses for each patient, how many ranking places were needed to cover all diagnoses. The patients had 1 to 3 diagnoses matching our criteria. In this more specific analysis, we observed unexpected performance results. The connectivity metric required on average 1.5 extra ranking positions to cover the diagnosis, outperforming relevance which required 2. We then ran the query with different combinations of weights for the metrics, as shown in Figure 9. Combining the metrics improved the performance, eventually reducing the required extra positions to 1.17.

The analysis of these experiments suggest that characteristics from both metrics are important to produce good performing rankings, a trait we conjecture as being pervasive across applications. In fact, this aspect motivated the support for arbitrary combination of metrics in the query language in the first place.

The accuracy of our diagnosis suggestions is also very practical, especially considering that we only analyzed a fraction of the information available for the experts – they also had access to patient's clinical report and a list of patient-specific risk factors. Moreover, we were able to produce the suggestions from a very simple declarative query, in a development effort of a few minutes (including the light parameter tuning that could have been skipped).

5. RELATED WORK

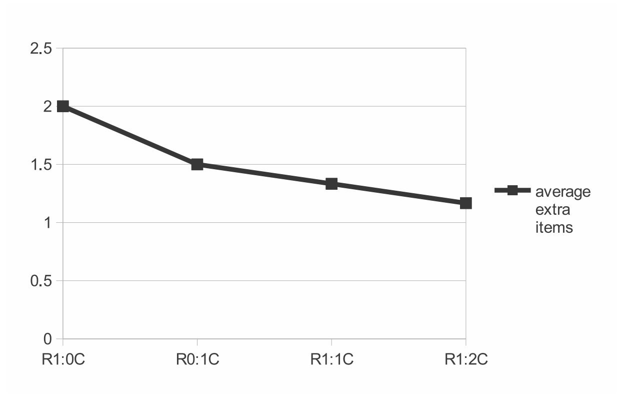

Figure 9: Average number of extra ranking items to cover the diagnosis from the expert

Research on integration of the IR and DB areas been an important effort towards query models that cover wider ranges of applications. Following the initial identification of challenges and applications, several successful approaches were proposed and implemented [20]. Most prominent research focuses on keyword queries over structured data and documents, top-k ranking strategies and extraction of structured information from documents.

Keyword query research draws from the simple yet effective keyword query model to allow integrated querying over documents and structured data. Most of the frameworks match keywords to documents, schema and data integrated in a graph structure. The connected matches form trees that are ranked based on variations of IR metrics such as tf*idf and PageRank. Some of the research focus on optimizing the top-k query processing [12] while others implement more effective variations of the ranking metrics [13].

Keyword queries over structured data are intended for tasks where the schema is unknown to the user. The techniques are effective for data exploration, but there is no support for more principled interactions. There are conceptual and structural mismatches among queries, data and results that make returned matches hard to predict and interpret.

The research on Top-k queries focus on enabling efficient processing of ranked queries on structured and semi-structured data. Ranking is based on scores derived from multiple predicates specified in the query. The main challenge is to compute results avoiding full computation of the expensive joins. The proposals vary on adopted query model, data access methods, implementation strategy, and assumptions on data and scoring functions (see [9] for a contextualized survey).

Scoring functions enable ranking based on properties of data elements. There is, however, no simple means to rank results based on the context of elements or how they are correlated, typical requirements for IR-like applications.

Information Extraction (IE) refers to the automatic extraction from unstructured sources of structured information such as entities, relationships between entities, and attributes describing entities [18]. Loading the extracted facts on a DBMS allows declarative querying over the data. This is a one-way, data-centric type of integration of DB and IR. The integration proposed here focuses on unified querying and data models.

We argue that the mentioned IR+DB approaches tend to focus on infrastructure issues related to extremes of enabling the type interaction present in one area over the data model of the other. In this paper we take a top-down approach to modeling query integration, questioning what are the main and defining properties of each area, and how to offer a unified, non-modal interaction over data and query models.

Less specific proposals, that also aim at increasing query expressiveness, have been developed, especially in the context of knowledge bases. Kasneci et al. [11] propose a new querying model for knowledge bases generated from IE. The model allows patterns matching and *relatedness* queries that allow flexible correlation of elements in the base. Rodriguez et al. [17] propose a query model that allows the context of user's knowledge base to influence query results. White and Smyth [21] present several algorithms to assess *importance* of nodes in a network. The calculation of the scores can be biased based on a set of initial nodes, which makes the approach more flexible than alternatives such as PageRank [5]. The work described in Varadarajan et al. [19] enable flexible graph queries that can express path patterns and ranking based on pre-defined metrics of importance. The framework requires the specification of a schema for the data and assumes that users know the schema and topology of the dataset to write path queries.

All these proposals aim at addressing the problem of inflexible query models that are inadequate to current application needs. However, the basic concepts of relatedness, context or importance are defined by the model and there are no means for users to express application-specific interpretations of the concepts. In our proposal, we offer declarative means for users to express combinations of metrics that can be adapted to specific information needs. We believe that our (i) declarative strategy, (ii) expressive metrics, and (iii) focus on data and query model integration offers a new level of flexibility and applicability in modern information processing scenarios.

6. CONCLUSION

We presented a query model that integrates querying concepts from prominent data management fields, aiming at covering a broader range of application scenarios. An important aspect to achieve more expressiveness at the query level is the combination of IR concepts in a declarative model. Keyword queries, ranking, and especially, effective metrics are important aspects in the integration. Our query model redefines IR metrics that rank entities based on the topology of their correlations. To the best of our knowledge, this is the first time the metrics presented are considered and formalized under the same model. Similarly, we are not aware of other ranking strategies that enable the level of expressiveness offered by the combination of our metrics and a declarative language. This combination allows data correlation queries that cover a wide range of applications.

As suggested by the query examples presented (Figure 5), it is possible to represent information needs that would require a level of data analysis that is beyond current implementations of typical systems. In fact, answering the type of queries introduced here in a typical technological environment nowadays would require substantial engineering for the implementation of *ad-hoc* solutions.

Our experiments show that our approach is practical in terms of performance and that our language can express complex concepts in real data and application scenarios. A deeper analysis of our integration scenarios can be found in [7].

We expect query-level integration to become increasingly important as our technological landscape continues to diversify. We showed how our model can cover a broad range of models and applications. Our experiments indicate the practicability of our approach. We are now working on architectural and query optimization strategies to enable streamlined and efficient deployment of our framework.

7. ACKNOWLEDGMENTS

The authors would like to thank the Faculty of Nursing, UNICAMP. This work was partially financed by the Microsoft Research FAPESP Virtual Institute (NavScales project), CNPq (MuZOO Project and PRONEX-FAPESP), INCT in Web Science (CNPq 557.128/2009-9) and CAPES, with individual grants from CAPES and FAPESP (2010/10158-2).

8. REFERENCES

[1] S. Auer, S. Dietzold, J. Lehmann, S. Hellmann, and D. Aumueller. Triplify: light-weight linked data publication from relational databases. In *Proceedings of the 18th international conference on World wide web*, WWW '09, 2009.

[2] C. Bizer. D2rq - treating non-rdf databases as virtual rdf graphs. In *Proceedings of the 3rd International Semantic Web Conference (ISWC2004)*, 2004.

[3] R. Blanco and C. Lioma. Graph-based term weighting for information retrieval. *Inf. Retr*, 15(1):54–92, 2012.

[4] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent Dirichlet Allocation. *Journal of Machine Learning Research*, 3(4-5):993–1022, 2003.

[5] S. Brin and L. Page. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1-7):107–117, 1998.

[6] F. Crestani. Application of spreading activation techniques in information retrieval. *Artif. Intell. Rev*, 11(6):453–482, 1997.

[7] L. Gomes-Jr and A. Santanchè. The Web Within: leveraging Web standards and graph analysis to enable application-level integration of institutional data. Technical Report IC-13-01, Institute of Computing, University of Campinas, January 2013.

[8] O. Hassanzadeh and M. Consens. Linked movie data base. In *Proceedings of the 2nd Workshop on Linked Data on the Web (LDOW2009)*, 2009.

[9] I. F. Ilyas, G. Beskales, and M. A. Soliman. A survey of top-k query processing techniques in relational database systems. *ACM Computing Surveys*, 40(4):11:1–11:58, Oct. 2008.

[10] R. Jensen, P. S. P. Silveira, N. R. S. Ortega, and M. H. B. de Moraes Lopes. Software application that evaluates the diagnostic accuracy of nursing students. *Intl Journal of Nursing Knowledge*, 23:163–171, 2012.

[11] G. Kasneci, F. M. Suchanek, G. Ifrim, M. Ramanath, and G. Weikum. NAGA: Searching and Ranking Knowledge. In *2008 IEEE 24th International Conference on Data Engineering*, pages 953–962. IEEE, Apr. 2008.

[12] B. Kimelfeld and Y. Sagiv. Finding and approximating top-k answers in keyword proximity search. In *PODS*, 2006.

[13] Y. Luo, W. Wang, X. Lin, X. Zhou, J. Wang, and K. Li. SPARK2: Top-k keyword query in relational databases. *TKDE*, 23(12):1763–1780, 2011.

[14] S. Markovitch and E. Gabrilovich. Computing semantic relatedness using wikipedia-based explicit semantic analysis. In *IJCAI*, 2007.

[15] R. Mihalcea and D. Radev. *Graph-based natural language processing and information retrieval*, volume 26. Cambridge University Press, 2011.

[16] M. A. Rodriguez and P. Neubauer. The graph traversal pattern. *CoRR*, abs/1004.1001, 2010.

[17] M. A. Rodriguez, A. Pepe, and J. Shinavier. The Dilated Triple. In *Emergent Web Intelligence: Advanced Semantic Technologies*, pages 3–16. Springer London, June 2010.

[18] S. Sarawagi. Information extraction. *Foundations and Trends in Databases*, 1(3):261–377, 2008.

[19] R. Varadarajan, V. Hristidis, L. Raschid, M.-E. Vidal, L. D. Ibáñez, and H. Rodríguez-Drumond. Flexible and efficient querying and ranking on hyperlinked data sources. In *EDBT*, pages 553–564, 2009.

[20] G. Weikum, G. Kasneci, M. Ramanath, and F. Suchanek. Database and information-retrieval methods for knowledge discovery. *Communications of the ACM, apr 2009*, 52(4):56–64, Apr. 2009.

[21] S. White and P. Smyth. Algorithms for estimating relative importance in networks. In *Proc of SIGKDD*, 2003.