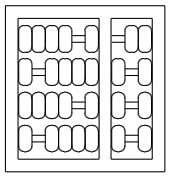

INSTITUTO DE COMPUTAÇÃO
UNIVERSIDADE ESTADUAL DE CAMPINAS

The Web Within: leveraging Web standards and graph analysis to enable application-level integration of institutional data

L. Gomes-Jr *A. Santanchè*

Technical Report - IC-13-01 - Relatório Técnico

January - 2013 - Janeiro

The Web Within: leveraging Web standards and graph analysis to enable application-level integration of institutional data

Luiz Gomes-Jr* André Santanchè†

Abstract

The expansion of the Web and of our capacity of producing and storing information have had a profound impact on the way we organize, manipulate and share data. We have seen an increased specialization of database back-ends and data models to respond to modern application needs: text indexing engines organize unstructured data, standards and models were created to support the Semantic Web, BigData requirements stimulated an explosion of data representation and manipulation models. This complex and heterogeneous environment demands unified strategies that enable data integration and, especially, cross-application, expressive querying.

Here we present a new approach for the integration of structured and unstructured data within organizations. In our framework, diverse data models are integrated in a unifying graph. A novel query model allows the combination of concepts from Information Retrieval and Database Management Systems into a declarative query language. This query language enables flexible correlation queries over the unified data that can be used to support a wide range of applications.

We show how Web technologies such as RDF can be leveraged to integrate institutional data. In our approach, the SPARQL query language is extended to enable the new query model. We also present examples of application of the model in several areas, such as CMSs, recommendation systems, social networks, etc. Experimental results demonstrate the viability of our approach in real scenarios.

1 Introduction

Digital data availability has grown to unprecedented levels and surpassed our capacity of storage and analysis. This new scenario has a profound impact on the way we organize and manipulate data. We have seen an increased specialization of database back-ends and data models to respond to modern application needs: text indexing engines organize data

*Instituto de Computa cão, Universidade Estadual de Campinas, 13081-970 Campinas, SP. Work 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

†Instituto de Computa cão, Universidade Estadual de Campinas, 13081-970 Campinas, SP. Work 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

on the Web, standards and models were created to support the Semantic Web, BigData requirements stimulated an explosion of data representation and manipulation models labeled under the NoSQL umbrella. This complex and heterogeneous environment demands unified strategies that enable data integration and, more importantly, cross-application, expressive querying.

Although data integration has been an active research topic for many decades, most proposals depart from environments that do not take into account the modern diversity of technological infrastructures. Federated databases, for example, usually adopt the relational model to integrate data sources, with limited capabilities when dealing with semi or unstructured data. Similarly, in typical OLAP implementations, the benefits of integration are restricted by the adopted query model: data analysts may able to answer complex questions, but there is no direct benefit to other applications inside the institution. For example, Web developers cannot leverage the potential of the integration in their implementations of recommendation systems because they typically work on very different query models. Similar issues also appear in other contexts, such as the Semantic Web, which brings great benefits for data integration but querying capabilities do not match the diversity of Web applications.

A level of integration that covers a wide range of data models and, more importantly, data query models would not only allow applications to incorporate more relevant information, but would also allow 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 my own;
- 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 show up in many situations in typical institutions, both for internal, administrative purposes or for Web applications developed for external use. The solutions in current infrastructures typically demand a substantial amount of resources and engineering to design ad-hoc subsystems.

To provide a richer approach for querying, enabling the correlation of data from diverse sources and applying components from diverse query paradigms, the integration must be tackled in the levels of data access integration, data model integration, and especially query model integration. To tackle the integration of data models, we employ an RDF graph that

interconnects data from diverse sources and models. The flexibility of graph models allows easy mapping from otherwise incompatible models (e.g. unstructured text and structured databases).

As for query model integration, we aim at bridging conceptual and cognitive gaps between query models and applications. We acknowledge the importance of the Information Retrieval (IR) and Databases (DB) fields – which dominate data-driven applications in current settings – to propose a new query model that unifies concepts from both areas. To enable our query model over this unifying graph, we reinterpret querying concepts from IR into graph analysis tasks. We implement this model in a new query language called in* (in star), which is an extension grammar for existing languages such as SPARQL.

This paper is organized as follows: Section 2 describes the requirements for data access and model integration in our framework. Section 3 details our integrated query model. Section 4 discusses usage scenarios. Section 5 presents experimental results. Section 6 contextualizes related work in respect to our proposal. Finally, Section 7 concludes the paper.

2 Data model integration

The level of integration that we aim at requires solutions to three main issues: (i) unified data access, so that queries have access to all data, (ii) unified data model, so that queries can reference data from diverse formats; and (iii) unified query model, so that applications can have a single interface for interaction with data. Our framework focuses on issues more related to (iii), but we also discuss the other aspects in this section.

2.1 The local unified graph

Data integration brings many benefits to institutions. Correlating data produced by diverse groups in distinct contexts allows the development of more capable applications and better data analysis. Although several researches and products were developed to address integration issues, we argue that revisiting this problem through the perspective of the new developments in applications and standards of the Web would allow for a better response to modern challenges. One aspect of this reassessment regards taking advantage of the new developments for data integration in the context of the Semantic Web.

The Semantic Web initiative has advertised the benefits of treating the Web as an integrated Giant Global Graph (GGG) [4]. Similar benefits could be achieved by institutions by integrating all their data in a Large Local Graph (LLG). A LLG lacks the diversity and magnitude of the GGG, but it allows higher levels of control over data and local processing power, enabling better semantic integration among distinct data sources and more expressive querying. Another advantage of creating LLGs is that it facilitates transference of information to and from the GGG.

The framework proposed here assumes an underlying LLG. Although our solutions have interesting applications also in the context of the Semantic Web, we require levels of integration and processing power that are not available for the GGG in the present. We,

therefore, focus on institutional data but expect that in the future technological advances would allow similar interactions in a broader context.

A LLG is meant to integrate a broad range of data from an institution. Aggregation of external data from the GGG would also be important in many scenarios. Integrating data across domains and models is important to allow rich correlation queries between diverse data elements. The graph model is suiting for this scenario. Its simplicity and flexibility allows the representation of most of the popular data models [3, 5]. Figure 1 shows a graph containing data derived from documents and relational databases (more details on the mapping in Section 2.3).

Here we employ the RDF(S) model for the LLG for several reasons: it is a stable and popular model, it implements a flexible graph model, classes facilitate the mapping of other models (e.g. object, relational), integration with other standards (e.g. URI, XML), standardized query language (SPARQL), simplified data sharing, etc.

It is important to emphasize that the strategy to create the unified graph is environment-specific. Although we provide general guidelines on how data should be represented as nodes and edges, our framework assumes the data are converted and interlinked in a coherent graph. What we want to show in this paper, and our main contribution, is that popular query models can also be translated into graph concepts, employing graph analysis in query processing. To take full advantage of the model, users should be aware of the semantics of the elements composing the graph. In that regard, our strategy is similar to an OLAP environment, in which the query model assumes data are integrated in a multidimensional schema – according to whichever strategy is adequate for the specific environment.

2.2 Data access integration

Integration of data access is a complex issue that requires considerable planning and engineering. A typical solution has to balance data redundancy with freshness. Redundant, localized repositories have better reading performance and allow for more complex data transformation workflows, while distributed federated databases allow access of up-to-date data and facilitate writes. The ultimate decision on which approach or combination of technologies is more adequate depends on the data and queries the institution plan to integrate.

In our RDF model of the LLG, we adopt the quadruple interpretation, in order to allow the specification of distinct graphs (a graph provenance URI complements the triple). In this setting, an institution could create separate graphs based on departments or data domains. The main challenges in creating these graphs are similar to the ones faced by any data integration strategy, covering aspects of syntactic and semantic integration.

Semantic integration between graphs (i.e. record linkage and deduplication) must employ traditional techniques of data cleaning. The difference is that here the result of these processes is the creation of *sameAs* properties between elements of the graphs, as typically done in Semantic Web applications. An important distinction regarding our query model (Section 3) is that edge traversals through *sameAs* properties must be cost-free.

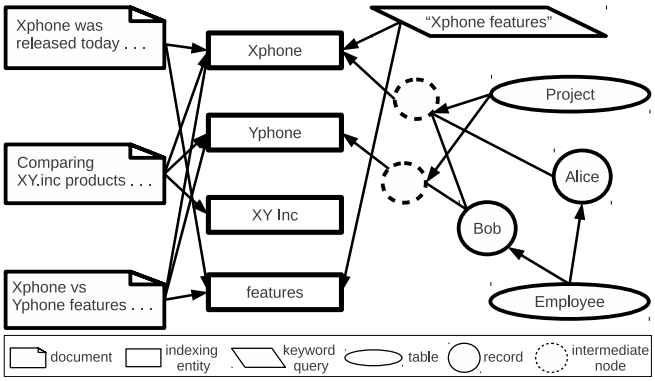

Figure 1: Data elements represented as a unified graph

2.3 Data model integration

There are several alternatives for mapping a given data model into graphs. Although our framework 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 an RDF graph [3, 5]. There is even a W3C working group[1] to define standards for these mapping languages. Here, to simplify the discussion, we assume that (i) table descriptions become RDF classes, (ii) rows become instances of their respective tables, with their primary keys as identifiers, (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 [7] and explicit semantic analysis [19] 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.

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 1). This graph representation of keyword queries allows them to be expressed alongside structured predicates in the queries (Section 3.4).

Figure 1 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

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

data, assuming the data models are integrated in the unifying graph.

3 Query model integration

Data access and model integration brings many benefits to institutions, providing a unified path for interaction with data. This interaction is, however, usually constrained by the data model and the query language employed for the integration. For example, in a typical OLAP setting, data are integrated in a data warehouse, but no direct benefit is gained by applications such as institutional search engines. The problem is that there is a conceptual gap between the interaction language in the integration infrastructure (OLAP) and the languages used by the applications (keyword queries, SQL, etc.).

Here we adopt a top down analysis to specify a query language that can be employed in a broader range of applications. We start from the analysis of the main query models used by application and then select the strongest characteristics of the models as components of our integrated language. The goal is to specify a language that can express concepts from diverse interaction models in a unified and intuitive way.

3.1 Information Retrieval and Databases

The two main groups of models for data driven applications today are those associated with Information Retrieval and Database Systems. It is natural that these two areas attained such distinction over the last decades. They together cover a broad range of the data structuring spectrum – from unstructured data in documents to structured data in relations. Typical applications in IR include search engines, recommendation systems, social networks etc. Applications taking advantage of DBMSs are ubiquitous, being through traditional relational databases or the more recent models for document databases, XML and semistructured databases, graph databases and the NoSQL movement. To specify a query language that could be used in such a diverse scenario it is important to identify the main characteristics of these areas.

Keyword queries and ranking are important concepts from IR, as other integration approaches have identified [26, 8, 2]. Significant research efforts have been dedicated to enable efficient ranking and keyword queries in a wider range of data model (e.g. relational, XML). Although central piece of an integrated approach, we think that efficient ranking and keywords are far from being everything that IR can offer in terms of data manipulation strategies.

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. User interaction in a IR system typically starts with keyword queries and ends with ranked results. However, 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. The complex algorithms that make it all possible are hidden behind the query interface. An integration approach should enable this type of complex correlation queries in a unified manner across diverse data models and tasks.

IR metrics like relevance and reputation are recurrent to many applications, and there is no reason why such concepts should be restricted to document retrieval. The same cognitive interpretation of these metrics would hold for typically relational data (e.g. the reputation of managers or the relevance of employees to a given product). This type of expressive querying should also cross the boundaries between unstructured and structured data, allowing an integrated analysis (e.g. what is the employee most relevant to a given news article). Other examples of this type of integrated querying are widespread across applications, as we intend to portray here.

The interaction with typical DB systems is also remarkably powerful. Declarative queries empowers 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 also be a key element in an integration framework.

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 (LLG) presented.

Declarative querying and traditional database concepts like selections, projections and aggregations are already provided by RDF query languages such as SPARQL. The remaining issues are related to enabling IR-like ranking metrics that now have to be reinterpreted in an RDF graph setting.

3.2 Graph representation of IR ranking metrics

The IR ranking metrics addressed in this paper are: relevance, reputation, influence, similarity, and context. 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. The selection of the specific metrics aims at covering a wide range of applications while also being simple to use and understand.

The translation of the ranking metrics to the unified graph strategy is a challenging task. Here we adopt a Spreading Activation (SA) [9] model for our novel interpretation of the IR 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 relationships 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 $O(n) = \dfrac{I(n) * d}{|out(n)|}$

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 1 is an instance of this definition (for trivial paths of length one).

Def. 2. $reputation(n, N) = v(SA(N)_n)$

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 or variability in the types of relationships used for the queries.

Def. 3. $influence(n) = |(SA(\{n\}))|$

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. 4. $similarity(m, n) = \dfrac{|p(SA(\{n\})) \cap p(SA(\{m\}))|}{|p(SA(\{n\})) \cup p(SA(\{m\}))|}$

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

Def. 5. $context(m, n) = \dfrac{|SA(\{n\}) \cap SA(\{m\})|}{|SA(\{n\}) \cup SA(\{m\})|}$

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

3.3 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 SPARQL and other graph query languages. A ranking metric can refer to:

- a single match variable (set of vertices), e.g. "rank papers from WWW 2012 according to first author *reputation*", where first author is the match variable in question (e.g. "SELECT ?firstAuthor ..." in SPARQL);
- a given vertex[2] 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.

3.4 Extending Declarative Queries

A convenient way to integrate the ranking metrics into existing query languages is to add a "RANK BY" clause. The clause should enable an arbitrary combination of metrics that expresses the global raking 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[3]. 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 another meaning for an acronym like inSPARQL: recursively, "inSPARQL is Not SPARQL".

Figure 2 shows a simplified BNF grammar of the proposed extension. A ranking can be specified as mix of (labeled) weighted ranking metrics (line 3). Weights capture the relative importance of each labeled metric. Ranking metrics are unary or binary. Unary ranking metrics are applied to a single match variable (lines 5 and 6). 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 11 to 14) to be applied to the ranking definitions. 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.

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

[3] http://docs.neo4j.org/

```
 1 ExtendedQuery::=RegularQuery RankClause
 2 RankClause::='RANK BY' ( RankMix | RankMetricDesc )
 3 RankMix::='(' Weight RankMetricLabel ( ',' Weight RankMetricLabel )+ ')'
 4 RankMetricDesc::=UnaryRankMetricDesc | BinaryRankMetricDesc
 5 UnaryRankMetricDesc::=UnaryRankMetric  'OF' MatchVariable
 6         Modifiers* RankMetricNaming?
 7 BinaryRankMetricDesc::=BinaryRankMetric 'OF' MatchVariable 'TO'
 8         ( MatchVariable | Vertex | KWQuery) Modifiers* RankMetricNaming?
 9 RankMetricNaming::='AS' RankMetricLabel
10 UnaryRankMetric::=Reputation | Influence | Popularity
11 BinaryRankMetric::=Relevance | Similarity | Context
12 Modifiers::=Follow | Depth | Direction
13 Follow::='FOLLOW' EdgeSet
14 Depth::='DEPTH' INTEGER
15 Direction:=='DIRECTION' ( 'INBOUND' | 'OUTBOUND' | 'BOTH' )
16 KWQuery:=='KWQUERY' '(' String ')'
17 EdgeSet:=='(' Edge ( ',' Edge )* ')' . . .
```

Figure 2: Simplified BNF grammar for the proposed extension (terminators omitted)

4 Usage scenarios

In this section elaborate on usage scenarios for the framework, from architectural aspects to applications.

4.1 GIRDB

We have so far discussed our framework in general terms, with little focus on implementation or architectural aspects. The proposed query model implies new requirements for user interaction, query processing and data management. We concentrate these new requirements in a database system architecture that we denominated GIRDB (Graph Information Retrieval Database system).

A GIRDB (Graph Information Retrieval Database system) is a system for which (i) there is a unified query mechanism with a single declarative query language that seamlessly expresses information needs that mix typical concepts from DB and IR areas; and (ii) the underlying data are transparently modeled as an integrated graph. The architecture of such system and data managements aspects are discussed in another paper [10].

4.2 Applications

The combination of the IR-inspired metrics in a declarative querying setting enables a high level of flexibility and expressiveness for the applications to explore. In the next paragraph we show and discuss some examples of queries that can be used for practical applications.

Search engines/CMSs: Figure 3a 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 reputation of the authors and the relevance of documents to the topic :Politics (assessed based on the connections between the query node and documents that are created by a Topic Modeling algorithm such as LDA).

Other aspects of a GIRDB not discussed in this paper would be interesting matches to implement novel CMS architectures like in Ngomo et al. [20]. Our metrics would also allow query answering based on the context of the user or a context defined by the user, implementing a query model such as the one proposed by [21]. Graph-based term weighting [6] could also be simulated in our query model.

Recommendation systems: Figure 3b 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.

Social Networks: Figure 3c shows a query that could be used for friend suggestion on a social network application. It ranks the top 5 persons over a given age based on the similarity of hobbies and movie preferences of user Alice.

Collaborative filtering: Figure 3d shows a query that filters posts from pages that friends of user Carol follow. The posts are ranked based on their influence in the network.

Decision support: Figure 3e shows a query that can be used to prospect for employees that would be good candidates to replace a manager (Charlie) in his post. The query favors employees strongly related to a (presumably important) product (yPhone) and also those that have professional contexts similar to the current manager.

Other applications: Similar queries could be used in several other scenarios, especially the ones with richly interconnected data and that require complex analysis of the correlations. Some examples are Semantic Web inference applications, were assessing correlations between classes and candidate instances can be complex [1]. The scientific domain is another interesting application field. For example, in a database with food network relationships, a query could identify relevant species or areas for conservation efforts.

5 Experiments

Here we show results for experiments in the proposed query model. We focus on reports for the relevance metric, which we consider a good representative of the model because of its (i) applicability in many areas, (ii) cognitive appeal, and (iii) challenges for query processing. The experiments are meant to show the impacts of parameter tuning of the SA model in the execution time and accuracy.

The database used in the experiments is the Linked Movie Data Base (LinkedMDB) [12], which we think is a good representative for the type of unified graph we aim at. The database integrates data from several sources (FreeBase, OMDB, DBpedia, Geonames, etc). The process used to semantically integrate the distinct sources is similar to what is done in

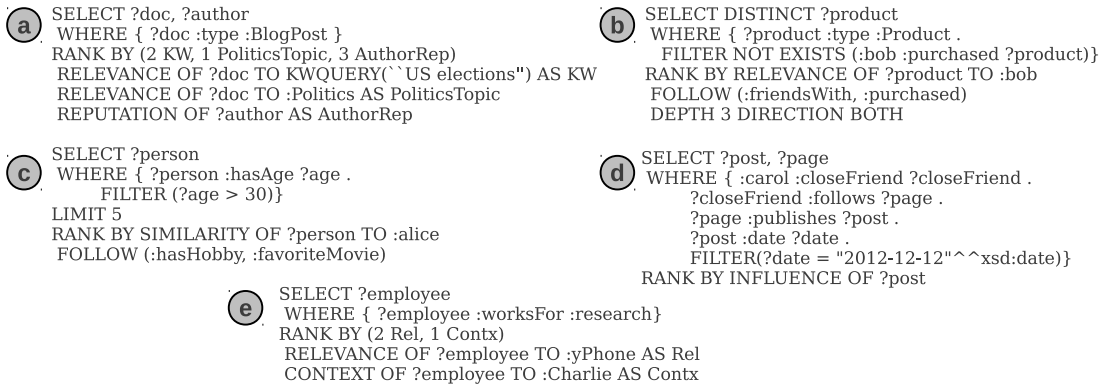

Figure 3: Examples of extended SPARQL queries (namespaces have been omitted)

```
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 4: Baseline query for the experiments

a typical Data Warehouse and precisely what we envision to be the workflow for the usage scenarios of our framework. The database contains 233,103 entities and 3,579,616 triples. The database represents the bulk of the relevant production in a real and important area of human activity, demonstrating that our framework can be applied to real scenarios.

The query used in the experiments is shown in Figure 4. The query combines graph pattern matching and structured filtering (equivalent to selection in relational algebra) predicates and our new RANK BY clause. 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 raking actors according to how linked to the director their careers are – a common pattern throughout Allen’s idiosyncratic production.

Although it would make sense in a real scenario, we do not restrict which properties (edges) are traversed in query processing (i.e. using the FOLLOW modifier in the query). Doing so would reduce execution time and perhaps make result interpretation more straightforward, but we think the unrestricted query better probes the robustness of our model.

5.1 Baseline

As the baseline for the performance and accuracy experiments, we executed the query in Figure 4 with parameters [t=0.1, d=0.9, c=3, a=100]. These are conservative parameters that showed good results in our analysis. Parameter c, which has the biggest impact in performance predictability, was set at 3 because this is the length of the path for most

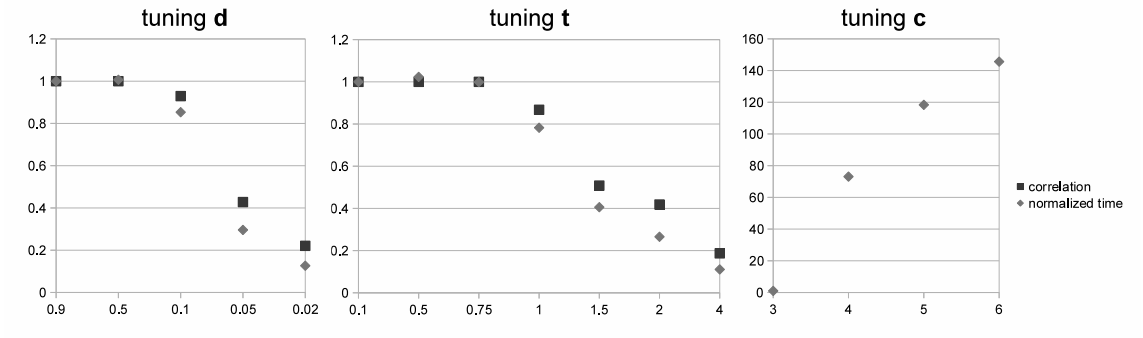

Figure 5: Correlation and normalized execution time for variation of parameters d, t, and c (in respect to the baseline)

important relationships between actors and directors (performances are linked to films by an intermediate *:performance* resource, and directors are directly linked to films). The next section shows how varying the parameters affect performance.

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[4]. Allen is well known for interpreting roles in his films, and he rarely performs in films from other directors. Mia Farrow, the second highest rank, has her career strongly linked to the director[5], 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, have only one performance recorded in the database, which would make her career highly linked to Woody Allen[6]. 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.

5.2 Parameter tuning

The parameters in our SA model 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 require-

[4]LinkedMDB uses distinct descriptors for the actor and the director, implying that they are separate entities

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

[6]she is credited in IMBD as the first person of Black origin to have a major role in a Woody Allen film

ments, but render more contextualized ranking that might include non-obvious aspects of the correlation of the elements in the query. The best balance between performance and completeness is application-specific. Here we want to demonstrate how parameter tuning affect these aspects.

Decay factor (d): Figure 5 (left) shows the values for normalized execution time and correlation for decay factors varying from 0.9 to 0.02. To assess correlation between the produced ranks and the baseline, we used the Kendall tau rank correlation coefficient [16]. The graph shows how both execution time and correlation drops as we use more aggressive values for d. Lower values for d mean faster degrading of the activation potential, which implies more effective pruning of the expanding activated network.

Activation threshold (t): Figure 5 (middle) shows the values for normalized execution time and correlation for decay factors varying from 0.1 to 4. As with the decay factor, correlations falls at rates proportional to performance gains. Appropriate values might be set according to query-specific tolerance for inaccuracies and performance requirements.

Depth (c): c is the parameter whose impact in performance is most predictable, since it directly limits the growth of the diameter of the activated network. Figure 5 (right) shows how execution time increases sharply as c grows. The correlation between the ranking, however, stays stable: correlation between c=3 and c=4 is 0.99, and values higher than that show total correlation and convergence of scoring. Therefore, the results show that close to optimum scoring can be achieved with low values of c.

The graph shows that the sharp increase in execution time is slowed down by the self-containing characteristic of the SA algorithm. More aggressive parameter tuning would of course increase this effect. For example, setting t=1 reduces execution time to less than one third for c=6.

5.3 Discussion

The experiments indicate that our approach is practical in terms of performance, a major concern with the type of graph analysis involved. 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.

We ran the experiments in a desktop machine that was querying a remote database server. This means that, in practice, we were issuing one remote procedure call (RPC) per step of the computation, which made network communication the bottleneck as processors and memory for the server and client were underused. As mentioned before, the goal is to have the computation of the metrics integrated inside the query processor.

Moreover, to simplify implementation and facilitate incremental design of our query processor, we are employing a tall stack of APIs[7] that can be reduced for further performance gains.

The LinkedMDB data has many hub nodes (nodes with high degree), notably the nodes representing classes (e.g. class actor has 50603 instances) which could affect the performance

[7] our implementation stack includes the graph traversal language Gremlin, the Blueprints and OpenRDF frameworks, and the Virtuoso triple store

of the algorithm. We implemented an artificial mechanism to filter out these nodes, but tests showed that filtering out hubs had little performance impact (approximately 8%) and no influence in the rankings (average difference of score is $2.71 * 10^{-5}$). This shows how the SA model does well in degrading the activation potential of the hubs, avoiding computations that would contribute little to the final score.

Appropriate tuning of the parameters is related to diverse properties of the traversed graph. The most important variables are (i) the expected average path length between the source and the destination nodes, and the (ii) expected branching factor in the paths. These variables can be calculated or estimated by the query processor to allow automatic tuning of parameters. We are working on defining the important statistics to maintain and on designing heuristics for this automation.

6 Related work

We now discuss related work on data integration in various levels: from data access integration, through syntactic/semantic integration, and up to application or query model integration. Integration at any level is highly dependent on the lower levels.

6.1 Data access integration

The first level of integration must provide a unique access point for the data. This can be accomplished by basically two approaches: centralizing the data or connecting the data sources in an infrastructure that simulates a centralized repository. Centralized integration of institutional data is typically related to the deployment of data warehouses or data marts [14]. Data centralization approaches have also been proposed in the context of the Semantic Web [12], and the DBpedia project[8] is a notable example of this type of approach.

The research on Federated Databases aims at providing a unified view of the data while maintaining the autonomy of the data sources [25]. In the context of the Semantic Web, Schwarte et al. [24] have proposed a federation layer for Linked Open Data. Schenk and Staab [23] have proposed a mechanism for the specification of views over Linked Data, enabling declarative federation of data sources.

Our framework is independent of the specific strategy chosen for data access integration. The requirement is that all interaction is done as if the data was integrated in a unified graph. Whether this integration is done through federation or physically integrating the data is an architectural decision based on expectations of performance and requirements for preserving the autonomy of data sources.

6.2 Data model integration

The next step towards data integration regards enabling data manipulation under a unified model. Federated databases frequently employ the relational model (common among data sources) for the integration. Data minig, which has application-specific requirements, favors the multidimensional model [15].

[8] http://dbpedia.org/

In the Semantic Web, the adopted unifying model is the RDF graph. The Resource Description Framework (RDF)[9] is a general-purpose language created for representing information about resources in the Web. The basic unit of information is a statement triple, which contains a subject, a predicate, and an object. All elements in a triple are identified by URIs (except for objects that can also be literal values). Triples can refer to each other, forming a graph. The advantage of the RDF model comes from its simplicity, enabling the representation of data from a wide range of domains.

There has been a substantial amount of research in mapping other data models into RDF [3, 5]. The W3C RDB2RDF Working Group[10] is defining languages and standards for mapping relational databases into RDF.

Besides having the data in a unified representation model, it is important to correlate data from the diverse sources into unified concepts. In the relational world, this process is know as record deduplication or linkage and is part of the ETL (Extraction Transformation Loading) workflow [11]. In the Semantic Web, the usual way to represent these correlations is the creation of *sameAs* relationships between entities. These relationships can be created manually or by automated processes. Hassanzadeh and Consens [12] employ several string matching techniques to correlate Linked Open Data from diverse sources to create an interlinked version of a movies database.

In this proposal, we assume that the institutional data is integrated in an RDF graph. This allows us to take advantage of other standardized technologies developed in the context of the WWW and the Semantic Web, such as universal identification through URIs, semantic integration through *sameAs* relationships, and the SPARQL query language.

6.3 Query model integration

Once data is integrated, it becomes possible to pose queries that could not be answered before, producing more valuable information for institutions and the public. The integration approaches, however, typically focus on integrating data under a specific query model, such as the relational or OLAP. This usually constrains the range of data models that can be integrate and, foremost, restricts direct querying of the integrated data from applications that use other query models.

Recently, there has been initiatives aimed at tackling integration at the application/query level. The research community has identified the interplay between the fields of Databases (DB) and Information Retrieval (IR) as a means to improve data integration and query expressiveness across applications [8, 2]. The drive to integrate the areas stems from the fact that they represent the bulk of data stored and processed across institutions. Furthermore, either field has been very successful by their own but still faces challenges when dealing with interactions typical to the other field.

The integration of the IR and DB areas has been an important topic in the agenda of the research community for many years. Following the initial identification of challenges and applications, several successful approaches were proposed and implemented [26]. Most

[9] http://www.w3.org/RDF/
[10] http://www.w3.org/2001/sw/rdb2rdf/

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 [17] while others implement more effective variations of the ranking metrics [18].

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 [13] 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 refers to the automatic extraction from unstructured sources of structured information such as entities, relationships between entities, and attributes describing entities [22]. 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 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 the 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.

7 Conclusion

We showed how modern standards and technologies developed to solve integration issues on the Web can be applied in a unifying framework for institutional data. Representing the integrated data as a graph is a good strategy for data model integration. Our main contribution is on extending this type of integration to a higher level of abstraction, tackling integration of query models.

In our approach, the key 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 IR 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 3), it is possible to represent information needs that would require a level of data analysis that is beyond current implementations of typical DB or IR 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.

Higher levels of expressiveness, however, always imply challenges to meet performance requirements. Our experiments demonstrate how the self-containing characteristics of the SA model can establish boundaries for query processing. SA has so far proved adequate to our setting, but since we adopt declarative queries, other processing models could be used. Simpler algorithms could be envisioned for specific cases, but we feel that the flexibility of the SA model, accommodating diverse topologies and path lengths, would be hard to replicate in other models.

Our experiments show that our processing model is adequate for practical applications, but to match the expanding availability and complexity of data, improvements in performance become a priority. We are exploring several strategies in that regard, from heuristics for query optimization and parameter tuning to a hybrid graph analysis model that combines SA with random walks.

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. Further query processing optimizations, in terms of architecture and heuristics are viable and will improve the applicability of the framework.

References

[1] H. Alves and A. Santanchè. Abstract framework for social ontologies and folksonomized ontologies. In *SWIM*. ACM, 2012.

[2] S. Amer-Yahia, P. Case, T. Rölleke, J. Shanmugasundaram, and G. Weikum. Report on the DB/IR panel. *SIGMOD Record*, 34(4):71–74, Dec. 2005.

[3] 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.

[4] T. Berners-Lee. Giant global graph. online posting, 2007. http://dig.csail.mit.edu/breadcrumbs/node/215.

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

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

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

[8] S. Chaudhuri, R. Ramakrishnan, and G. Weikum. Integrating DB and IR technologies: What is the sound of one hand clapping? In *CIDR*, pages 1–12, 2005.

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

[10] L. Gomes-Jr and A. Santanchè. Revisiting db/ir integration: graph-based data and query model unification. In *submitted to the International Conference on Extending Database Technology (EDBT 2013)*, 2013.

[11] J. Han and M. Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann, 2006.

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

[13] 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.

[14] C. Imhoff, N. Galemmo, and J. G. Geiger. *Mastering Data Warehouse Design: Relational and Dimensional Techniques*. Wiley, 2003.

[15] M. Jarke, M. Lenzerini, Y. Vassiliou, and P. Vassiliadis. *Fundamentals of Data Warehouses*. Springer, 2003.

[16] M. G. Kendall. A new measure of rank correlation. *Biometrika*, 30:81–93, 1938.

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

[18] 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.

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

[20] A.-c. N. Ngomo, N. Heino, and M. Kaltenb. SCMS - Semantifying Content Management Systems. In *The Semantic Web - ISWC 2011*, pages 189–204, 2011.

[21] 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.

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

[23] S. Schenk and S. Staab. Networked graphs: a declarative mechanism for SPARQL rules, SPARQL views and RDF data integration on the web. In *WWW*, 2008.

[24] A. Schwarte, P. Haase, K. Hose, R. Schenkel, and M. Schmidt. FedX: A federation layer for distributed query processing on linked open data. In *ESWC (2)*, volume 6644 of *Lecture Notes in Computer Science*, pages 481–486. Springer, 2011.

[25] A. Sheth and J. Larson. Federated database systems for managing distributed, heterogeneous, and autonomous databases. *ACM Computing Surveys*, 22(3):183–236, 1990.

[26] 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.

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

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)