SocialGQ: Towards Semantically Approximated and User-aware Querying of Social-Graph Data

Riccardo Martoglia
FIM Department, University of Modena and Reggio Emilia, I-41125 Modena, Italy,
E-mail: riccardo.martoglia@unimore.it

***Abstract* – The proliferation of social and collaborative sites makes users increasingly active in the generation of social-graph data; however, such sea of data often hinders them from finding the information they need. In this paper, we present SocialGQ ("Social-Graph Querying"), a novel approach for the effective and efficient querying of social-graph data overcoming the limitations of typical search approaches proposed in the literature. SocialGQ allows users to compose complex queries in a simple way, and is able to retrieve useful knowledge (top-k answers) by jointly exploiting: (a) the structure of the graph, semantically approximating the user's requests with meaningful answers; (b) the unstructured textual resources of the graph; (c) its social and user-aware dimension. An experimental evaluation comparing SocialGQ to leading approaches shows strong gains on a real social-graph data scenario.**

Keywords* – *social-graph, knowledge management, approximate querying, user-aware techniques, semantic retrieval.

1. Introduction

In recent years, the web has evolved from a static web, where users consume information, to a "social web" where they are also able to produce them. The proliferation of social networks (e.g., Facebook, LinkedIn, ...), social bookmarking and collaborative tagging sites (e.g., BibSonomy, CiteULike and Delicious), social question-answering sites (e.g., Stack Overflow), microblogging sites (e.g., Twitter) and social components on "traditional" websites makes users increasingly active in the generation of content. This ever increasing amount of "social-graph" data is quite peculiar in its composition. First of all, it is typically characterized by a dual nature, both *graph-structured* (users/resources as nodes and relations, such as friendships, as arcs), and *unstructured* (the large amount of textual resources, e.g., documents and comments). Moreover, the *social/user-aware* component is obviously quite prominent, where the available nodes and resources are possibly created/modified by different users. Consider, for instance, the small excerpt from the BibSonomy graph data [9] depicted in Fig. 1, showing two content resources (nodes) with their associated annotations. In particular, the BibTeX instance "C19837" (node n_{13}[1]) has been annotated with the "semantics" tag by user "U150" (n_4), while Bookmark instance "C45829" (n_{14}) has been tagged as "semantic web" by user "U881" (n_{12}). The social nature of the data is highlighted in figure by small symbols (circle, triangle) depicting the nodes modified by the two users. Note that entity nodes (including instances) are depicted with rounded corners, differently from simple value nodes, representing textual content (e.g., abstracts, descriptions) and other attributes (e.g., publication years). Finally, consider the strong semantic characterization of such data, given by type hierarchies (both BibTeXs and Bookmarks are defined as Contents) and also user-specified tag hierarchies (for instance, user "U881" defined the "semantic web" tag as a specialization of the "semantics" tag).

In this context, a crucial problem is to extract useful knowledge from this wealth of information. Users should be able to compose complex queries (beyond the classic keyword model) in a simple way, allowing them to quickly find all the relevant information with respect to the their needs. Consider for instance the following three examples of information need:

Q1. Find (not already known by me) content having a summary about "semantic" and "text";
Q2. Find BibTeX entries semantically related to BibTeX titled "Versatile...";
Q3. Find documents that have been tagged as "semantics" (also considering sub-tags defined by me).

Such requests go well beyond what typical websites and standard search tools allow: they are very difficult or even impossible to express with simple keywords; they require semantic approximation in order to be effectively solved on the graph data, both in terms of labels (e.g., terms like "summary" and "document" are not defined in the data) and structure (e.g., Q1 asking for "content" should retrieve all its subclasses, i.e., both BibTeX and Bookmark entries; Q2 asks for BibTeXs generically related to the named one; Q3 also asks for content tagged with sub-topics); they also require unstructured content full-text search capabilities going beyond exact search (e.g., Q1), as well

(DOI reference number: 10.18293/SEKE2018-052)

[1] For clarity's sake, nodes are univocally identified by the node ids i shown on the left upper corner and will be referenced as n_i; moreover, some type relationships (e.g., users, blank nodes) are omitted from the figure.

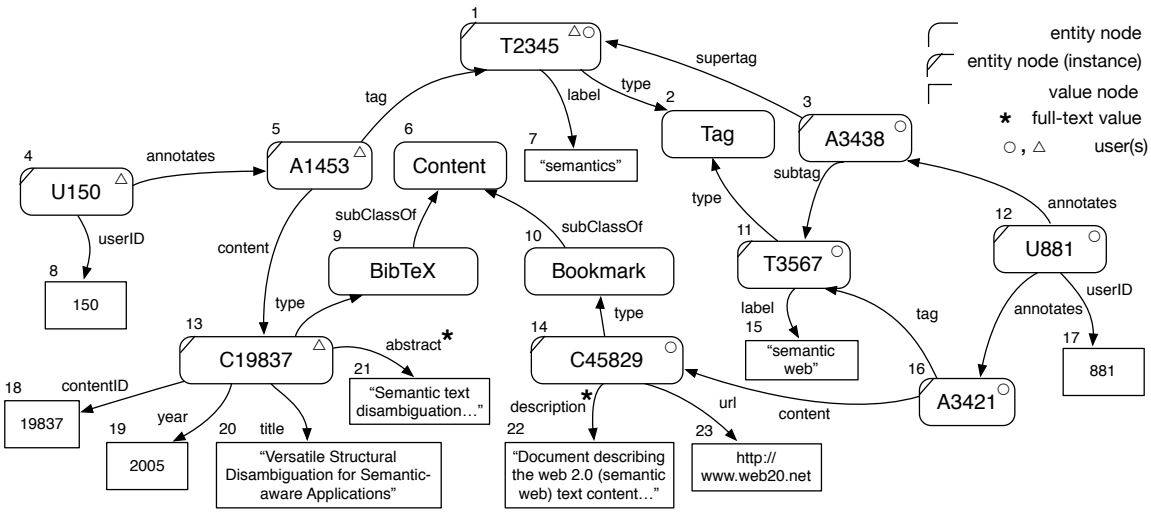

Figure 1. An excerpt from the BibSonomy graph data

as user-awareness (e.g., Q1, Q3); finally, they require an effective ranking in order to provide first the most relevant results.

Generally speaking, currently available search tools do not support all the above-mentioned requirements, typically making such complex requests impossible to be solved automatically, i.e., without requiring a lot of manual effort from the user. Typical social sites search capabilities are keyword-only, limited to exact search in specific fields, syntactic-only (absence of semantics) and typically disregard the social nature of the data (i.e., who created what). Similarly, looking at the current research state of the art, there are several (approximate) graph querying [4, 8, 14], user-aware textual search [12, 15] and social search [6, 7] proposals; however, no one combines all the required features in a single solution applicable to the context of the social-graphs.

In this paper, building on the acquired know-how on graph data management [11], we focus on defining the foundations of SocialGQ ("Social-Graph Querying"), a novel approach for the effective and efficient querying of social-graph data. SocialGQ is aimed to overcome the limitations of current approaches, by jointly exploiting: (a) the structure of the graph, by means of approximation techniques capable of semantically approximating the user's request with meaningful answers; (b) the unstructured textual resources present in the contents of the graph; (c) the social and user-aware dimension. The capacity for semantic approximation also responds to the need for simplicity in the composition of the query itself, intelligently adapting the request to the graph. The final aim is to retrieve the most useful (top-k) answers in an efficient and automated way.

Fig. 2 provides an overview of the SocialGQ architecture: the Data Manager module (described in Sect. 3) organizes the social-graph data into ad-hoc data structures efficiently supporting the semantic approximation, full-text and user-aware requirements. The Query Processing and Ranking modules (Sect. 4) actually provide the top-k answers to the user query in input, avoiding to build useless solutions. The modules are based on specific data, query and ranking models (briefly sketched in Sect. 2). An experimental evaluation comparing SocialGQ to existing approaches shows strong gains on a real social-graph data scenario (Sect. 5). Finally, Sect. 6 concludes the paper also by briefly analyzing related works.

2. Social-graph data, queries and answers

The aim of the SocialGQ data model is to have a flexible social-graph model (a) capturing key social-graphs' features and (b) not bound to specific graph standards (e.g., RDF), even if easily supporting them. Data is generically represented as a connected multigraph (i.e., a graph with parallel edges) with node and edge labels. A social-graph essentially represents a portion of the real world through *entities* (concepts and their instances), *values*, and *relationships* between them. Considering our reference example (Fig. 1), scientific publications and annotations are described by concepts such as `BiBTeX`, `Bookmark`, `Tag`; n_1, n_{11}, n_{13} and n_{14} are some instances. Instances are characterized by a *type* and a *userlist*; the latter is represented in figure with small symbols in the upper right part and denotes users that created/modified them.

As to queries, in order to support complex requests such as those discussed in Sect. 1, we go beyond the keyword-based approach. A SocialGQ query is expressed as a labeled multigraph connecting entity nodes and conditions on values. Fig. 3 shows Q1, Q2 and Q3 expressed in our model. Note that users can annotate any node or edge with the wildcard "any label", '#'. For instance, n_2, n_3 and edge n_2-n_3 in Q2 denote the user's absence of knowledge about the specific nodes and edges connecting them. Conditions c can be defined on query nodes, specific cases characterizing a social-graph query are:

- the *full-text condition* (e.g., see n_3 in Q1), supporting full text search in value nodes and returning a normalized TF-IDF score [13] cs. This allows SocialGQ to exploit the unstructured part of the social-graph;

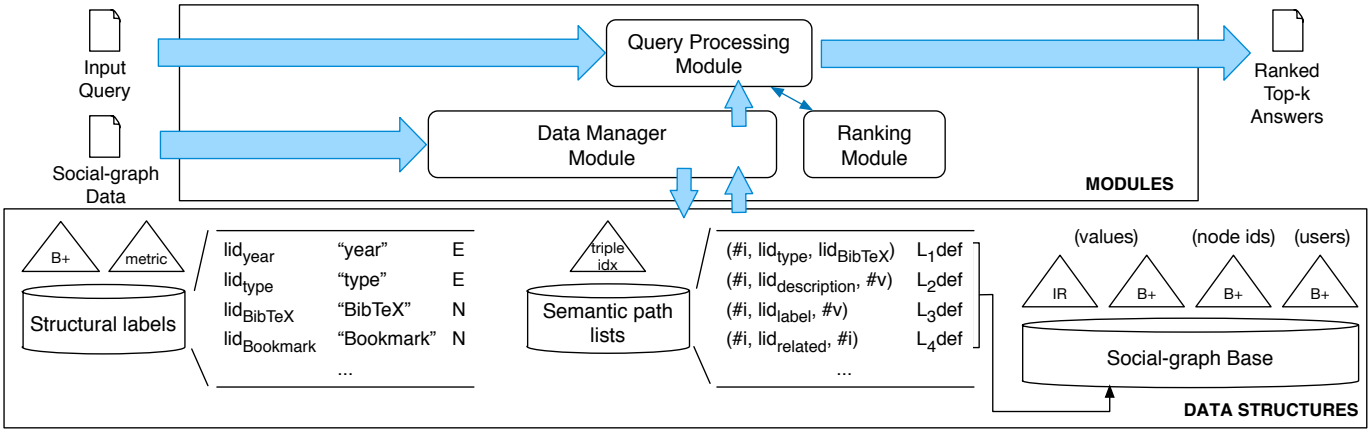

Figure 2. An overview of Social-GQ architecture

- the *user-aware condition* on query nodes (represented as +/- in figure), allowing users to restrict data matches to nodes not modified/known by them (-, as in Q1 for n_2), or the opposite (+, as in Q3 for n_6, requesting tags related to "semantics" and defined by who submitted the query).

SocialGQ answers are portions of the data graph that *semantically approximate* the query. Two kinds of approximations are tackled: node/edge label and structural mismatch. To this regard, note that none of the sample queries finds an exact match on the reference graph. For instance, in Q1 (Fig. 3) the edge label `summary` is used instead of `abstract`, moreover no data edges are directly typed as `Content` (however, there are possible `BibTeX` and `Bookmark` matches, which are subclasses of `Content`). In SocialGQ, the degree of mismatch between labels is quantified by means of a user-definable semantic distance function d_L that, for any pair of labels, returns a value ranging from exact match (0) to total mismatch (1). As to structural mismatches, a purely topological approach which relaxes adjacency constraints by allowing arbitrary node/edge insertions in the data graph would not be able to produce meaningful answers. Instead, we consider meaningful sequences of consecutive edges (i.e., *paths*), named *semantic paths*, that match query edges with an approximation cost ac. For instance, the path n_2-n_1 in Q1 (Fig. 3) could be approximated with the semantic path n_{13}-n_9-n_6 (Fig. 1), which has the same "meaning".

The goodness of each answer a to a query Q is quantified through a scoring function $\mathcal{S}$:

$$\mathcal{S}(a) = \alpha_n \cdot (1 - \underset{n \in Q}{avg}(d_L(n, \overline{n}))) +$$
$$+ \alpha_e \cdot (1 - \underset{e \in Q}{avg}(d_L(e, \overline{e}) + ac(\overline{e}))) + \alpha_c \cdot \underset{c \in Q}{avg}(cs(c)), \quad (1)$$

where α_n, α_e and α_c, α_n+α_e+α_c=1, are customizable coefficients (default=1/3) combining the average of: (a) label approximations (d_L) occurring with each query node n and edge e ($\overline{n}$ and $\overline{e}$ denote matching data nodes/edges); (b) structural approximation costs (ac) on each edge e; (c) cs scores of each full-text query condition c (note that user-aware conditions prune out incompatible answers and do not affect ranking). The higher the returned score $\mathcal{S}(a)$, in $[0, 1]$, the better the answer a.

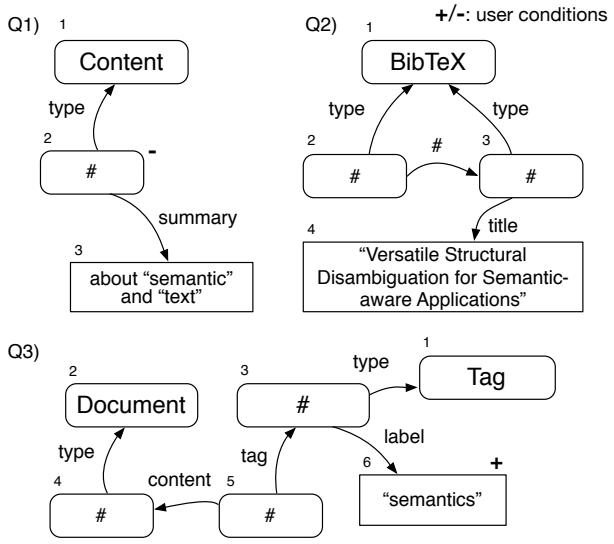

Figure 3. Three sample queries

More details on the data structures supporting SocialGQ query processing and how query processing itself is managed are presented in Sects. 3 and 4, respectively.

3 Data Manager

SocialGQ data manager organizes data in a "core" social-graph base (lower right part of Fig. 2), which is managed via the graph management system Neo4j [1], and in a series of additional ad-hoc auxiliary structures, which are the focus of this section. These provide advanced indexing and support for the innovative query features, ultimately helping the query processor in building the best answers as soon as possible.

First of all, the "Structural labels" table (lower left part of Fig. 2) stores structural (i.e., edges' and entity nodes') labels, associating the label to a short identifier (lid) and a kind ("E" for edges and "N" for nodes). Label data are indexed by means of B+ trees (allowing for exact search) and metric indexes (allowing for approximate search). As we will see, in query pro-

cessing this allows to quickly check if the structure of the given query is solvable on the social-graph data.

Besides dealing with labels, one of the most expensive operations for solving a query is to find the paths in the social-graph data which match the given query edges. To this end, the idea behind SocialGQ "semantic path lists" (lower central part of Fig. 2) is to organize repetitive data paths by means of identifying triples summarizing their structural role, ignoring specific instance and value information. Let us start the discussion by considering single edges. A single edge e connecting two nodes n and n' is straightforwardly identified by the involved label ids (i.e., (lid_n,lid_e,$lid_{n'}$)). Value and instance nodes labels are generically represented with "#v" and "#i", respectively. For instance, the `label` edges n_1-n_7 and n_{11}-n_{15} (Fig. 1), connecting tag instances to their label value, are both associated to the triple (#i,lid_{label},#v). Triples are indexed and used in query processing to match with query edges (as we will see in next section). Associated to each triple, is a definition $Ldef$ of a list pointing to the actual involved portions of the graph base (for instance, in our example, list L_3def points to the two above mentioned edges, among others). One interesting feature of SocialGQ is that such lists, which can be potentially quite large, do not need to be pre-computed or materialized. Being the core graph base managed in Neo4j, list definitions are (sets of) Cypher [1] scripts that can be easily customized by the data administrator. This flexibility is indeed particularly useful since such lists are not limited to identify relevant data edges, but also semantic paths. In this way, custom semantic rules capturing the specific meaning of the social-graph can be easily coded into the definitions so to allow meaningful structural approximations to be easily identified in the query processing phase.

Let us consider for instance the above discussed case of (#i,lid_{label},#v) paths: L_3def can be defined to match single edges by means of the following Cypher MATCH clause:

```
(t:Tag)-[l:label]->(v:Value)
```

but can also include a first level of approximation matching paths involving subtags (i.e., specializations of a tag label):

```
(t:Tag)<-[s:subtag]-n -[s:supertag]->
  (t2:Tag)-[l:label]->(v:Value)
```

In this way, for instance, Q3 edge n_3-n_6 (Fig. 3) will match with data edge n_1-n_7 ("semantics" label) but also with path n_{11}-n_3-n_1-n_7 ("semantic web" label, a specialization).

Approximation levels (i.e., scripts) in list definitions are stored in order of increasing approximation cost ac; the cost is automatically defined on the basis of the path length (in case, that can be freely customized). Other useful rules that could be easily incorporated in our social-graph example are for managing `type` paths (i.e., subclasses) or even for enriching the graph with useful relations not explicitly present in the data: for instance, a `related` relation between contents can be defined for contents sharing a common tag. This will match, for instance, with Q2 edge n_2-n_3 (Fig. 3).

A number of indices on the graph base (lower right of Fig. 2) complete SocialGQ graph structures. These allow for efficient filtering of the semantic path lists on: (a) node values (both B+trees for exact search and inverted indices for full-text search); (b) node ids (useful for joining edges in query processing); (c) users (for user-aware filtering). Such indices are directly managed in Neo4j, while inverted indices are externally coded in order to provide full TF-IDF score support.

4 Query processing and ranking

The goal of the query processor and ranking modules is to exploit the data structures made available by the data manager to generate the *top-k answers* approximating the query. Since social-graph are typically large and repetitive in their structure, a large number of approximate answers are typically available in the graph. Instead of generating the whole answer space and then applying the ranking formula (Eq.1), SocialGQ generates the top-k answers in an order that is already correlated with the ranking measure, avoiding to generate many useless results. The algorithm builds on the foundations of the Threshold Algorithm (TA) [3] and follows the steps summarized below:

1. search for (approximate) structural query node label matches in the "structural labels" tables, possibly pruning out unanswerable queries;
2. for each query edge, search associated triple in "semantic path lists" and associate the relevant list definition(s) $Ldef$;
3. perform sorted access in parallel to each of the lists. For each access: (i) build answers involving the extracted data path, computing the score $\mathcal{S}(a)$ of each answer a, and remembering it if one of the k highest; (ii) update *uBound*, the score of the set of the next data items under sorted access to the lists;
4. stop whenever at least k answers have been built whose grade is higher than *uBound*.

Please note that, as described in Sect. 3, each list $Ldef$ returns data paths which are already sorted by approximation cost; this allows the algorithm to: (a) be aware of the goodness of upcoming answers by means of *uBound*; (b) avoid unnecessary relaxations to the query (and, therefore, optimize data accesses).

To get a very simplified intuition of its working, consider Q1 in Fig. 3 submitted by user "U881". Since the specified structural labels (i.e., `Content`, `type`, `summary`) are available in the graph base (`summary` is approximated by `abstract` and `description`), list definitions are retrieved for the two edges' triples, (#i,lid_{type},$lid_{Content}$) and (#i,$lid_{summary}$,#v). Lists access is restricted to values matching the full-text condition (e.g., node n_{20}) and instances not modified by user "U881" (e.g., node n_{14}). The semantic paths extracted from the two lists are then accessed and joined to build such final answers as (n_{13}-n_{20})-(n_{13}-n_9-n_6).

5. Experimental Evaluation

We will now present the preliminary results we obtained from an exploratory evaluation on a real social-graph data scenario by means of a first prototype of SocialGQ. This paper is

Query	#n	#e	#any	Description	Struct relax	(# edge)	(mult types)	Label appr	Full text	User aware	#exp ans
Q1	6	5	3	Users who tagged a BibTeX entry having author "Klaus Reuter"							1
Q2	7	7	4	Tags of BibTeX entries (-) having same year as BibTeX titled "Features of Similarity"						✓	48
Q3	6	5	3	Users who tagged a Content (-) having description about "Writing techniques"	✓		✓		✓	✓	2
Q4	5	4	3	Bookmarks (-) tagged "apple" (+)	✓					✓	18
Q5	4	4	3	Users (-) connected (# edge) to bookmark with URL "http://moodle.org/"	✓	✓				✓	1
Q6	5	4	3	BibTeX (-) connected (# edge) to BibTeX titled "Conceptual Knowledge Processing"	✓	✓				✓	17
Q7	6	5	3	Documents (-) tagged by profile with id 12	✓		✓	✓		✓	147
Q8	3	2	1	Content (-) having summary about "programming" and "database"	✓		✓	✓	✓	✓	23

Table 1. Features of the reference queries

focused on evaluating effectiveness, also in comparison to leading literature approaches. This requires a graph not particularly big in size but with a sufficiently complex structure, including different annotations and relation types. To this end, we consider as our reference collection a portion of the Bibsonomy graph dump [9], containing information about 139551 tag annotations, 28611 bookmarks, 11378 BibTeX entries, 8127 tags, 9566 tag specializations and 347 users. Moreover, a small hint regarding efficiency will be presented at the end of the section; in this case, we will also consider a larger graph involving over 1 million documents and 4 million annotations. The prototype incorporates the described advanced data indexes (including full-text ones) and query processing techniques, written ad-hoc in Python. It also exploits Neo4j graph management system, benefiting from node type management optimizations allowed by its built-in type management. User information is stored in Neo4j using node array properties. The chosen label distance d_L is a WordNet-based one we already used for disambiguation purposes [10]. All parameters are kept at their default values.

We consider a set of significant queries, named Q1-Q8, representative of a full-range of possible user information needs. Tab. 1 shows their features, including number of nodes, edges, "any label" wildcards, textual description, required features and number of expected answers. Q1 and Q2 are examples of exact queries which, even if not requiring special approximations, would be quite difficult to express using simple keywords. Queries Q3-Q8, instead, require several kinds of approximations; all queries except Q1 also require user-aware processing. For instance, Q4 asks for bookmarks not already modified/known by the user tagged as "apple": this should include, if available, documents tagged with subtags by the same user (e.g., "mac"). Structural relaxation is required to manage node subclasses (as for "content" in Q3 including types "BibTeX" and "bookmark"). Q5 and Q6 contain generic connection edges ('#'). Q7 and Q8 require label approximations (e.g., "document", "profile", "summary" used instead of "content", "user", "description"/"abstract", respectively). Finally, Q3 and Q8 also contain full-text conditions.

Tab. 2 shows the results of the effectiveness evaluation performed on Q1-Q8 in terms of: precision P (i.e. percentage of relevant retrieved answers w.r.t. the retrieved ones) and recall (i.e. percentage of relevant retrieved answers w.r.t. existing relevant ones). The results achieved by SocialGQ are also compared with the ones achievable through alternative approaches. Let us start with SocialGQ (left part of table): our approach is able to retrieve all results (recall is 1) for all queries, and all the retrieved results are relevant. This is achieved thanks to its combined semantic approximation, full-text and user-aware features. In particular, the high repetitiveness of the structure of a social-graph makes the employed structural and label approximations very effective and precise; label matches are also favored by, most notably, the relatively low number of different structural labels w.r.t. other scenarios and kinds of graph data (e.g., knowledge graphs). Moreover, the semantic path approach makes it easy to answer queries such as Q5 and Q6 without producing a large number of non-relevant results.

	SocialGQ		Exact		Web	Non-semantic		NAGA		TALE	
Query	P	recall	P	recall	#q	P	recall	P	recall	P	recall
Q1	1	1	1	1	5	0.0001	1	1	1	1	1
Q2	1	1	1	1	57	0.0000	1	1	1	1	1
Q3	1	1	n/a	n/a	n/a	0.0000	1	n/a	n/a	n/a	n/a
Q4	1	1	n/a	n/a	1	0.0000	1	1	1	0.07	1
Q5	1	1	n/a	n/a	n/a	0.0000	1	0	1	1	1
Q6	1	1	n/a	n/a	n/a	0.0000	1	0	1	0	1
Q7	1	1	n/a	n/a	n/a	0.0000	1	n/a	n/a	1	1
Q8	1	1	n/a	n/a	n/a	0.0000	1	n/a	n/a	n/a	n/a

Table 2. Effectiveness results and comparison

Other approaches, instead, typically struggle in supporting all the required features. For instance, an exact approach ("Exact" column) is only able to solve the first two queries. On the other hand, standard website search pages require a typically very high number of submitted queries ("Web" column) to solve a complex request. In our case, queries Q1, Q2 and Q4 can be partially answered through the Bibsonomy built-in search but: (a) this requires sending up to 57 simpler requests (Q2) and combining the results; (b) ranking and user-aware filtering are not supported anyway. We also see that the large number of nodes with same structural labels present in a social-graph makes it particularly infeasible to use syntactic-only approaches. A naïve approach ("Non-semantic" column) that computes node matches and connects them in all possible ways (e.g. a title to all available BibTeX nodes, disregarding semantic path information), achieves a near-null precision. Finally, we consider two flexible and well-known graph matching approaches presented in the literature. NAGA [8] is able to correctly deal with subclass approximations (e.g., Q4), however: it

manages '#' edges in a syntactic way (very low precision n Q5 and Q6); it does not manage label approximation (Q7 and Q8 are not supported). TALE [14], instead, manages all structural approximations by defining a syntactic length threshold in the matching paths which proves not always effective: while Q5 is ok, in Q4 it retrieves results involving generalizations of "apple", e.g., non-relevant "technology" documents. Please also note that neither NAGA or TALE natively support full-text and user-aware conditions.

As a final note regarding efficiency, for all the above queries the initial SocialGQ prototype was able to retrieve the top-5 results in under 0.04 secs (0.12 secs in the large collection) on a standard single-node configuration.

6. Concluding remarks

Successfully querying social-graphs requires to jointly exploit the complex nature of such kind of data: graph structure, semantic meaning, unstructured textual resources, user-aware dimension. As we have seen, this typically goes beyond the capabilities offered by the social sites' search functions. In the literature, several works provide interesting, even if separate, results in each of the involved fields. In the area of (approximate) graph matching, many proposals [4, 8, 14] recognize the need to overcome the keyword-based paradigm and support early forms of approximation that, however, do not take into account semantics. [11] proposes a framework to support the semantic approximation of a complex query on a graph. None of these works, however, considers the specifics of social data and/or the contained unstructured contents. On the other hand, in the field of social data, a recent survey [2] analyses several search tool proposals, including [6, 7], all however limited to keyword-based search. Furthermore, it underlines the current sharp distinction between social search approaches (working only on the graph) and social web search ones (working only on unstructured content), noting the lack of approaches exploiting both components. Finally, a number of works [12, 15] highlight the effectiveness of identifying user-aware techniques that allow to customize the search results based on the user profile, but always in the context of unstructured data.

In this paper we laid the foundations of SocialGQ, a new proposal aimed to overcome the above mentioned limitations. Taking into account our past experiences in different scenarios (e.g., generic structured graph [11] and full-text enterprise search [12]), we sketched a framework combining for the first time all the features that are deemed essential for effective and efficient social-graph querying. The underlying techniques leverage on the strengths of approximate graph matching, semantic approximation, textual and user-aware retrieval to retrieve the most relevant answers first. Preliminary effectiveness results on a real data scenario are encouraging. Moreover, strong efficiency foundations have also been laid, being the proposed architecture based on extensions to widespread and reliable big data graph management technologies (i.e., Neo4j).

Making full use of the huge potential given by the ever-increasing amount of social-graph data is indeed a very ambitious goal for the research community. Powerful social-graph search techniques can have a large impact and become the basis of a wide range of services integrated into leisure and business social networks: complex and personalized search tools to find products and information; intelligent help desk services to answer customer questions; tools to acquire a deep real-time knowledge on what is happening within the organization [5]. This work represents only one of the first steps toward this vision. In the future, we plan to: (a) work more deeply on efficiency evaluation; (b) perform detailed tests on additional and larger social-graph scenarios; (c) consider new techniques for automatic semantic path rules identification and refining.

This work is partially supported by UniMoRe within the FAR 2016 Department Project "SocialGQ".

References

[1] Neo4j Graph Platform and Cypher language. http://neo4j.com.

[2] M. R. Bouadjenek, H. Hacid, and M. Bouzeghoub. Social Networks and Information Retrieval, How Are They Converging? A Survey, a Taxonomy and an Analysis of Social Information Retrieval Approaches and Platforms. *Inf. Syst.*, 56(C):1–18, 2016.

[3] R. Fagin, A. Lotem, and M. Naor. Optimal Aggregation Algorithms for Middleware. In *PODS*, pages 102–113, 2001.

[4] W. Fan, J. Li, S. Ma, N. Tang, and Y. Wu. Adding Regular Expressions to Graph Reachability and Pattern Queries. In *Proc. of ICDE*, pages 39–50, 2011.

[5] J. Hagel and S. K. Ellis. Four Ways Social Data Can Generate Business Value. http://sloanreview.mit.edu/article/four-ways-social-data-can-generate- business-value/.

[6] X. He, M. Gao, M.-Y. Kan, Y. Liu, and K. Sugiyama. Predicting the popularity of web 2.0 items based on user comments. In *Proc. of SIGIR*, SIGIR '14, pages 233–242, 2014.

[7] D. Horowitz and S. D. Kamvar. The anatomy of a large-scale social search engine. In *Proc. of WWW*, pages 431–440, 2010.

[8] G. Kasneci, F. Suchanek, G. Ifrim, M. Ramanath, and G. Weikum. NAGA: Searching and Ranking Knowledge. In *Proc. of ICDE*, pages 953–962, 2007.

[9] KDE Group, University of Kassel. Benchmark Folksonomy Data from BibSonomy, 2017-07-01 dump. https://www.kde.cs.uni-kassel.de/bibsonomy/dumps/.

[10] F. Mandreoli and R. Martoglia. Knowledge-based sense disambiguation (almost) for all structures. *Information Systems (Information)*, 36(2):406–430, 2011.

[11] F. Mandreoli, R. Martoglia, and W. Penzo. Approximating expressive queries on graph-modeled data: The GeX approach. *Journal of Systems and Software*, 109:106–123, 2015.

[12] R. Martoglia. AMBIT: semantic engine foundations for knowledge management in context-dependent applications. In *Proc. of SEKE*, pages 146–151, 2015.

[13] G. Salton and C. Buckley. Term-Weighting Approaches in Automatic Text Retrieval. *Inf. Process. Manage.*, 24(5):513–523, 1988.

[14] Y. Tian and J. Patel. TALE: A Tool for Approximate Large Graph Matching. In *Proc. of ICDE*, pages 962–973, 2008.

[15] T. Vu, A. Willis, U. Kruschwitz, and D. Song. Personalised query suggestion for intranet search with temporal user profiling. In *Proceedings of CHIIR '17*, pages 265–268. ACM, 2017.