Enabling Social Search in Time through Graphs

Kostas Stefanidis
FORTH, Heraklion, Greece
kstef@ics.forth.gr

Georgia Koloniari
University of Macedonia, Thessaloniki, Greece
gkoloniari@uom.gr

ABSTRACT

Recently, social networks have attracted considerable attention. The huge volume of information contained in them, as well as their dynamic nature, make the problem of searching social data challenging. In this work, we envision the design of a complete framework for social search by exploiting both the underlying social graph and the temporal information available in social networks. To encompass the numerous search needs, our framework includes a time-aware graph and query model. To deploy the proposed query model over any social network, we define a logical algebra that provides a set of operators required for query evaluation, and for supporting a ranking functionality. We conclude by presenting SQTime, a prototype that implements our framework.

1. INTRODUCTION

Due to the increasing popularity of social networks and the vast amount of information in them, there have been many efforts in enhancing web search based on social data. Also, given that social networks contain data about the network, i.e., data for which users visit the network, data about the users and their social connections and data about the social activities users perform, it is evident that an effective and efficient way to manage such data is essential for satisfying the user search needs. This has lead to the emergence of social search that utilizes the underlying graph structure and the content of a social network to provide more personalized and expressive search features for the users.

Besides the graph structure dictated by the relationships among the entities of the network, another important dimension of social networks is their dynamic nature. New content is added through user activities and updates occur both in the structure of the graph and the content shared, representing respective changes in the users' interests. This temporal aspect of the information should influence social search either explicitly by enabling users to query for particular time points or periods, or implicitly by providing the most recent results and higher ranking of fresher content.

Here, we envision the design of a complete social search framework that exploits the underlying social graph and its temporal aspects. To satisfy the varying search needs, the framework includes a time-aware query model and a corresponding logical algebra. To deal with the temporal aspect of the social network, instead of relying on managing different graph snapshots, as [3, 5], we adopt an annotated graph model that incorporates time by associating each element in the graph with a label with its temporal information. Nodes, representing users and objects, and the edges between them, representing social relationships, have a label that indicates their *valid time*. Dissimilarly to [3] that focuses on graph evolution, to encompass various querying needs, we define a query model that exploits the underlying graph representation. The model defines queries for the entities of the network, i.e., users and objects. *User-centric* queries offer a personalized search feature by exploiting the social relationships of a user, whereas, *system-centric* queries provide a global search feature with many applications in online-shopping and target-advertising, so as to select the best group for a new product or the best products to promote to a given user. Unlike [2] that does not concern about any temporal information, our model enables time-awareness by allowing *time-dependent queries* that exploit time explicitly using it as a hard constraint to filter out irrelevant results.

To deploy our querying model over any social network, we define a logical algebra that provides the set of basic operators required to evaluate the queries specified by our model. Besides the basic operators, the algebra includes a set of operators for supporting a *ranking* functionality. Specifically, the ranking mechanism enables the implicit use of time to enhance the results of a query by providing a time-dependent ranking, so that more recent or fresher results are returned first. We conclude with a description of SQTime, a prototype that implements our framework offering a graphical interface for running time-dependent queries expressed in natural language, and visualizing the returned results.

2. DATA & QUERY MODEL

2.1 Data Model

The entities of a social network represent users and objects. An entity n_i is described by a set of predicates a_{i_j} of the form ($a_{i_j}.attribute = a_{i_j}.value$). For example, an attribute for a user can be "$name = Alice$" and an attribute for an object, e.g., an event, can be "$location = Shanghai$".

We model a social network as an undirected graph, $G = (V, E)$. The set of nodes V corresponds to the entities that belong to the social network, i.e., $V = U \cup O$, where U is the set of users and O the set of objects. The set of edges E captures the relationships between the entities that belong to V; user-to-user edges capture the friendship between the corresponding users, while user-to-object edges declare that a user uses or participates in some way in an object o_j.

We extend the typical graph model with temporal information towards making social search time-dependent. We exploit the valid time of data items and consider an element, node or edge, of a graph G as *valid* for the period for which the corresponding element of the network it represents is also valid. To do this, each element e_i in the social graph is annotated with a label with the time intervals for which the element is valid. To cope with the dynamic nature of the social network that causes elements to become valid (a user joins the network), invalid (it leaves) and then valid again (rejoins the network), for each element $e_i \in G$, its label is defined as a set of disjoint intervals $l(e_i) = \{(t^j_{start}, t^j_{end}) | j \geq 1\}$, which implies that element e_i is valid for the time intervals $\{[t^1_{start}, t^1_{end}),\ [t^2_{start}, t^2_{end}), \ldots\}$.

2.2 Query Model

The goal of our framework is to support queries for the social graph that exploit both the graph structure and its time dimension. Let us consider an example of a popular query, $Q1$: "*find all of Ross' friends that attend sports events in July 2014*". This query expresses constraints both on time and the structure of the graph. We use $Q1$ as our illustrative example to demonstrate the different possibilities one has for querying the social graph, so as to deduce a complete query model that covers our social search needs[1].

Let us begin with the result of $Q1$. The query retrieves a subset of the friends of the user *Ross*, i.e., the result set, V', is a set of user nodes, $V' \subseteq U \subseteq V$. Similarly, one could query for object instead of user nodes, i.e., $Q2$: "*find all events in July 2014 that Ross' friends attend*", where it holds: $V' \subseteq O \subseteq V$. Thus, we can discern, according to the type of nodes that form our result set, between *queries for friends*, such as $Q1$, and *queries for objects*, such as $Q2$. The first type gives emphasis on the company of the user, while the second focuses on the objects to be consumed.

We ignore the time constraint for now, as we treat time as a separate dimension. As our goal is to exploit the graph structure, our queries include constraints on the connectivity between the different entities that are referenced in the query context. In particular, $Q1$ requires the result nodes to be directly connected to a set of intermediate object nodes specified through a set of predicates, i.e., $Q1$ requires the friends of Ross that are connected with objects of type sports events. Let P be a set of predicates concerning the attributes in the nodes descriptions of the form ($attribute\ \theta\ value$). For numerical attributes, $\theta \in \{=, <, >, \leq, \geq, \neq\}$, and, for non-numerical attributes, $\theta \in \{=, \neq, prefix\}$. We say that $P(v_i) = true$, if and only if, node v_i satisfies all predicates in P. We refer to nodes, for which P is $true$, as the *qualifying* nodes in the query, denoted as QV. The retrieved nodes (the ones forming the result set) need to be connected to all the qualifying nodes, i.e., we want the friends of Ross that attend all sports events in July 2014. Since the *all* operator is a very strict requirement, it can be relaxed to require that the retrieved nodes are at least connected to one qualifying node, i.e., the friends of Ross that attend *any* sports event in July 2014.

Similarly to the result nodes, the predicates can also refer to either user or object nodes. To illustrate this, for $Q2$, the result nodes, i.e., events, need to be connected to all Ross' friends. In this example, P is empty as we have no predicates specified for these user nodes. Note that as our graph model does not define connections between objects, for a query for objects the predicates are only applied on user nodes. However, we may also define queries for users on which the predicates are also applied on other users. For instance, $Q3$: "*find all of Ross' friends that are friends with users with occupation athlete*". In $Q3$, the result user nodes need to be connected (be friends) with other users for which the predicate *occupation* = *athlete* is true.

Furthermore, $Q1$ includes a reference node, i.e., the user node that corresponds to user *Ross*. Our query is centered around this reference node, u_i, and we call such queries *user-centric queries*. $Q1$ requires the user nodes that form the result set to be friends with *Ross*, i.e., it requires for them to be directly connected to the reference node.

We can consider extensions, such as $Q4$: "*find all the friends of the friends of Ross that attend sports events in July 2014*". In this example, the result nodes are at distance $d = 2$ from the reference node, which means that there is a path of 2 edges connecting them. Although $d > 2$ is also plausible, in practice it is rarely used.

$Q2$ is also a user-centric query, but in this case, the reference node is not directly connected to the result nodes. Instead, it is required that the qualifying user nodes of the query are directly connected to *Ross*.

On the other hand, one may also want to pose queries where no reference node is specified at all, i.e., *system-centric* queries. For instance, consider query $Q5$: "*find all users that attend sports events in July 2014*". Such queries, capture the system perspective and their goal is to identify either sets of users that share some common interests and, for instance, may be interested in a particular product or event the system wants to promote, or similarly, sets of objects that may be of interest to some particular users so that they can choose to promote these objects to them.

System-centric queries can be extended to allow for connectivity constraints to be specified on the returned results. For instance, $Q6$: "*find all groups of users that are friends and attend sports events in July*" requires not a set of user nodes as a result, but groups of such users that are directly connected within each group.

The last part of our query is the time constraint. Thus, for $Q1$, T specifies that the sports events that Ross' friends attend are valid in July 2014. In $Q1$, T is applied on the qualifying nodes QV and therefore can be treated as another predicate that however does not concern the object descriptions, but rather their labels. Another possibility is to specify time constraints on the result nodes, i.e., $Q7$: "*find all the friends of Ross that are valid in July 2014 and have attended sports events*". If a time constraint is specified for any part of the query, then we are able to capture *time-dependent* queries.

T can be also applied on the labels of the edges of the social graph, introducing a different type of queries. For instance, $Q8$: "*find all the friends of Ross that express that they will attend CIKM 2014 in March 2014*", requires a set of users that have established their connections with an event during a specific time period.

[1]The proposed model extends and generalizes a preliminary query model that was presented in [4].

Given that T is defined as a time period $[b, c)$, we discuss next how we handle *validity*. Specifically, to determine whether a node v_i is valid for T, one needs to compare T against $l(v_i)$. If one of the time intervals included in $l(v_i)$ is included in the interval specified by T, i.e., $\exists(t^j_{start}, t^j_{end}) \in l(v_i)$, such that, $t^j_{start} \geq b$ and $t^j_{end} < c$, then v_i is valid for T. A reverse interpretation is also possible. That is, one could require the valid time of the node to include T, i.e., $\exists(t^j_{start}, t^j_{end}) \in l(v_i)$, such that, $t^j_{start} < b$ and $t^j_{end} > c$. From a different perspective, we consider *before* and *after* semantics, requiring $t^j_{end} < b$ or $t^j_{start} > c$, respectively. In a more relaxed interpretation, it suffices for the valid time of a node to simply intersect with T, for the node to be considered valid. Finally, for $b = c$, time point queries are supported. Similarly, we can define the validity of the edges appearing in the social graph.

To sum up, one can discern between four structural parts that compose a query:

(i) The set of result nodes V', which are the nodes to be retrieved and form the query result.

(ii) The set of qualifying nodes QV, which are the intermediate nodes through which the different entities in the query are connected, and are specified by a set of predicates P.

(iii) The reference node u_i, which is the node around which the query is centered, and a distance d from it.

(iv) The time constraint T that checks for the validity of nodes and edges involved in the query.

More formally, we can define a generic query Q that encompasses the queries we have described.

Definition 1. *Given a graph $G = (V, E)$, $V = U \cup O$, predicates P, a user node u_i, a distance d and time constraints T, we define a query Q as a query that retrieves a set of nodes $V' \subseteq V$, such that, $v_j \in V'$, if and only if, $\forall v_l \in V$ for which $P(v_l) = true$ and (v_l is valid according to T and $\exists(v_j, v_l) \in E$), or ($\exists(v_j, v_l) \in E$ that is valid according to T), and*

- *if $v_l \in U$, $\exists$ path between v_i and v_l with length $= d$, or*
- *if $v_l \in O$, $\exists$ path between v_i and v_j with length $= d$.*

3. A LOGICAL ALGEBRA

Developing a flexible and expressive mechanism to manipulate data in social graphs is an important challenge that we need to face. Our focus here, is on integrating in a principled way the search process into the context of social graphs. Towards discovering information that covers the needs of users in a flexible manner, we propose a logical algebraic framework. By using this logical algebra, we can express sophisticated tasks for retrieving data relevant to the user queries both at a semantic (e.g., by querying for specific predicates) and social (e.g., by querying for friendships) level. We start this section with the core operators of the algebra that can be used for evaluating the system- and user-centric queries introduced above, and then define temporal-dependent and ranking operators.

The *select node* operator takes as input a set of nodes V, a set of predicates P and a parameter Z that defines the retrieving focus of the operator, i.e., users U or objects O. The operator outputs the nodes that satisfy the predicates in P. Formally:

Definition 2. (Select Node Operator). $\sigma_{P,Z}(V) = \{v | v \in Z \wedge P(v) = true\}$.

This operator is appropriate for implementing system-centric queries.

For user-centric queries, we have additional parameters, i.e., the reference node and a distance from it. Thus, we define the *select node from graph* operator that takes as input a social graph G, a node v_i corresponding to a user $u_i \in U$, a distance d, a set of predicates P and a parameter Z that defines the retrieving focus of the operator, i.e., users U or objects O. The operator outputs the nodes v_j from G that satisfy the predicates in P for which there exists at least one path with length at most d between v_i and v_j. Formally:

Definition 3. (Select Node from Graph Operator). $\sigma_{P,Z,d}(v_i, G) = \{v_j | v_j \in Z \wedge P(v_j) = true \wedge \exists$ *path between v_i and v_j with length* $\leq d\}$.

In the main framework, we define also the *difference operator* for excluding from a set, nodes that are not directly connected with the nodes of a different set.

Definition 4. (Difference Operator). $dif(V_i, V_j, G) = \{v_x | v_x \in V_i \wedge \exists v_y \in V_j$, *such that*, $\exists$ (v_x, v_y) *in* $G\}$.

Given $G = (V, E)$, the query $Q9$: "*find all Ross' friends that have attended sports events*" can be accomplished as follows. Direct friends of Ross, say V_1, are captured by $\sigma_{\{\},friends,1}(Ross, G)$. We retrieve the objects, say V_2, with *topic = sports events*, by $\sigma_{topic=sports\ events,objects}(V)$, and $dif(V_1, V_2, G)$ locates the subset of Ross' friends that have attended sports events.

For handling time-dependent queries, we extend our operators in order to take into account time constraints T for users and/or objects.

Definition 5. (Temporal Select Node Operator). $\sigma_{P,Z,T}(V) = \{v | v \in Z \wedge P(v) = true \wedge v$ *is valid for* $T\}$.

Definition 6. (Temporal Select Node from Graph Operator). $\sigma_{P,Z,T,d}(v_i, G) = \{v_j | v_j \in Z \wedge P(v_j) = true \wedge \exists$ *path between v_i and v_j with length* $\leq d \wedge v_j$ *is valid for* $T\}$.

Then, for evaluating $Q1$ that augments $Q8$ with the temporal constraint [1/12/2013, 31/12/2012) for events, we locate set $V2$ as:

$\sigma_{topic=sports\ events,objects,[1/12/2013,31/12/2012)}(V)$.

Operators for locating valid edges are defined in a similar manner.

Besides the basic operators implementing our query model, additional operators can be specified to support extended functionalities. To provide more meaningful results than a simple set of returned nodes and enable the implicit use of time, we define a ranking functionality. Ranking is time-dependent, in the sense that it ranks more recent nodes higher. To determine how recent a node is, we rely on the time of the user activities rather than the actual valid time of the nodes themselves, as we expect that more recent activities tend to better reflect the current trends in the network. This information is captured in the labels of the edges of the network that connect the result and qualifying nodes.

Thus, given a node v_j, we define the $freshness$ of a node v_i ($fresh(v_i)$) as the maximum t_{start} value in the labels of the edges that connect v_i and v_j. To support our ranking functionality, we introduce the *temporal social winner* and *rank* operators. Then, a node v_i belongs to the winner if there is no node v_l with age greater than the age of v_i.

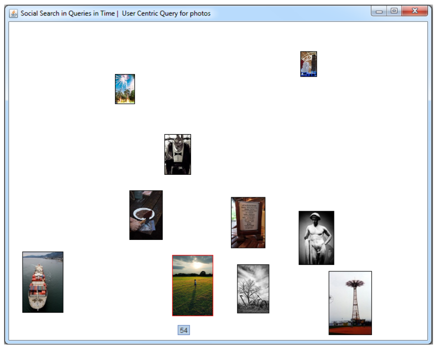
Social Search in Queries in Time | User Centric Query for photos
54

Figure 1: SQTime results presentation.

Definition 7. (Temporal Social Winner Operator). $winner(V) = \{v_i | v_i \in V \wedge \nexists v_l \in V, such\ that, fresh(v_l) > fresh(v_i)\}$.

Ranking all nodes in V can be achieved by repetitive applications of this operator.

Definition 8. (Temporal Social Rank Operator).

$$rank^i(V) = \begin{cases} rank^1(V) = winner(V) \\ rank^{i+1}(V) = winner(V - \cup_{k=1}^{i} rank^k(V)) \end{cases}$$

In general, there are different ways for handling operators. Firstly, operators can be implemented on-top of a DBMS either as stand-alone programs or as user-defined functions. Alternatively, operators may be translated into other, existing relational algebra operators during a pre-processing step. Finally, operators can be implemented inside the database engine using specific physical operators and algorithms. It is our purpose, in this line of work, to study which direction better fits our complex scenario.

4. USER-SYSTEM INTERACTION

In this section, we present SQTime, a prototype that allows users to explore the different types of social search queries in our model, and the implications of enhancing queries with time both implicitly and explicitly. SQTime is built on top of a social graph with data from [1], that includes anonymized information about the evolution of the Flickr social network and in particular, user-to-user links, photos and favorite markings of photos by users. Photos are randomly assigned to given photo ids to preserve the relationships among them and the users. This first version of SQTime offers a graphical interface for running time-dependent queries and visualizing the returned results sets.
Querying Interface. The querying interface provides lists of options for each of the building blocks of a query, which users can combine in four easy steps to compose their own queries. All options are expressed in natural language not requiring from the user to have an understanding of the underlying graph structure. Firstly, users select between the user-centric or system-centric query perspective. Then, they determine whether the query concerns users or objects, i.e., Flickr photos. Options such as "*get all photos liked by my friends*", for user-centric queries, and "*get all photos liked by the group of users specified in step 1*", for system-centric queries, are available. SQTime includes as well queries at distance two. Thus, one may also select to "*get all photos the friends of my friends liked*". The next two steps guide users to augment their queries with time. Step 3 supports the explicit use of time allowing users to add a time constraint in their queries. One can either select a predefined constraint that concerns the most recent past, as such queries are more popular, or specify her own time period for the constraint. For example, for the time constraint "*six months ago*", we may construct queries like "*get all photos that were liked by my friends 6 months ago*" and "*get all photos that were valid six months ago and were liked by my friends*". The last parameter configures ranking. Two options are provided, "*no ranking*" and "*ranking based on freshness*". Ranking enables users to see the implicit use of time, as the results are returned ranked based on their freshness.
Results Presentation. The result presentation interface visualizes query results in an intuitive way that clearly illustrates their temporal relationships. For a query for objects, the qualifying photos are displayed to the user. For clarity, we limit the returned results to 10. If no ranking is used, 10 random photos from the result set are returned. When ranking based on freshness is selected, the 10 fresher results are displayed. As shown in Fig. 1, we display results as photos of different size, where photos of bigger size represent the higher ranked objects, i.e., the fresher ones. For queries for users, a graph is used to display the qualifying users and their relationships. For the ranking based on freshness option, shades of green are used to represented the ids of each user, with bolder colored ids reflecting the fresher results.

5. SUMMARY

In this paper, we have shown the growing importance of introducing new mechanisms for social search queries through graphs. The user experience in querying a social network can be significantly improved by exploiting, in addition to the underlying graph structure, the temporal characteristics of social data, towards offering query results that better meet the users information needs, as well as more expressive user-system interactions. One of our first goals for future work, is to experimentally evaluate the effectiveness of our approach. Moving forward, we envision enabling open access to a social graph that as a whole brings together different social networks. For this scenario, a time-aware search functionality is essential to answer the query needs of users who are looking for information obtained by integrating numerous and heterogeneous sources. To support such cross-network queries, we need to extend both our algebraic framework and the way that users interact with the system.

Acknowledgments

This work was partially supported by the European project DIACHRON (IP, FP7-ICT-2011.4.3, #601043).

6. REFERENCES

[1] M. Cha, A. Mislove, and K. P. Gummadi. A Measurement-driven Analysis of Information Propagation in the Flickr Social Network. In *WWW*, 2009.
[2] M. Curtiss, I. Becker, T. Bosman, S. Doroshenko, L. Grijincu, T. Jackson, S. Kunnatur, S. Lassen, P. Pronin, S. Sankar, G. Shen, G. Woss, C. Yang, and N. Zhang. Unicorn: A system for searching the social graph. *PVLDB*, 6(11):1150–1161, 2013.
[3] U. Khurana and A. Deshpande. Efficient snapshot retrieval over historical graph data. In *ICDE*, 2013.
[4] G. Koloniari and K. Stefanidis. Social search queries in time. In *PersDB*, 2013.
[5] A. G. Labouseur, P. W. Olsen, and J.-H. Hwang. Scalable and robust management of dynamic graph data. In *BD3*, 2013.