Birds of a tag flock together*

Serge Abiteboul
INRIA Saclay – Île-de-France
serge.abiteboul@inria.fr

Sihem Amer-Yahia
Yahoo! Labs
sihem@yahoo-inc.com

Alban Galland
INRIA Saclay – Île-de-France
alban.galland@inria.fr

Amélie Marian
Rutgers University
amelie@cs.rutgers.edu

Pierre Senellart
Institut Télécom
Télécom ParisTech
pierre@senellart.com

Motivation and context. Recently, extracting knowledge from user-generated social data has attracted a lot of attention. Several works have focused on modeling user-generated tags such as the study of tag clouds [3], cross-floksonomy analysis [7], or the use of tag-driven communities for content search [4] or recommendation [6]. The huge and increasing amount of raw tags and annotations clearly contains valuable social information and is thus an essential asset for helping community members find information they are interested in. For example, the exploration of communities in del.icio.us, a well known social tagging website, can be enhanced through the use of tags, see e.g., [1]. Our goal is twofold: extract knowledge from this rich social information by *clustering social data based on affinity (i.e., proximity in the social network)*, and provide better *query support and navigation* on the semantically enriched data.

We consider a general data model where data of interest is captured by a *social tag graph*. This graph consists of user and item nodes connected by tagged edges. The terms item, semantic tag and user, should be understood here very broadly. An item can be any entity that community members may want to share information on. A semantic tag may be a tag in the del.icio.us sense, but it may also be a word or a concept extracted from a review, a comment, an opinion, etc. A user could be an actual person in the social network, a group of persons, an institution, even possibly a program that extracts information from the social network. A very concrete application we had in mind when working on this is del.icio.us. Items are URLs. Tags are simply words. Users are del.icio.us users. However, our model can be used in many other contexts, e.g., photos in Flickr (which is rather similar) but also books in Amazon, movies in Yahoo! Movies, videos in YouTube or products in Ebay.

An essential aspect of extracting knowledge from a social tag graph is the identification of groups of users, items and tags, referred to in this paper as **objects**. The clustering of these objects is based on a distance called *affinity* that defines object proximity. It is used for building expressive queries in a QBE (query-by-example) style on the social tag graph. Of course, there are important differences between various applications, and due to that, we will rely heavily on different clustering distance measures. However, our thesis is that knowledge extraction, in the applications we are focusing on, shares such a large number of features that it is worth developing a general model capturing them and generic knowledge extraction tools to support it.

Tag-based navigation has been studied in the past [5]. For example, in [2], the authors evaluate early del.icio.us data for efficient navigability, and reveal that efficiency is decreasing over time. In [8], the authors examine usage patterns to improve navigability on CiteULike and BibSonomy. In our work, navigation is *dynamic* and relies on the incremental graphical construction of queries. As we will see, we facilitate the common tasks one would like to perform in such an environment: consider only portions of the tags (e.g., those by your friends, or those from last week), filter or cluster objects according to certain criteria, get descriptions of objects or groups of objects, zoom on some aspects of interest.

In this paper, we briefly introduce the data model and describe knowledge extraction and navigation on the enriched social graph. In the future, we plan to run experiments to compute affinities on different data sets and address scalability and pre-computation issues to speed up query processing.

Knowledge Extraction. We consider three sets of objects: users $\mathcal{U}$, items $\mathcal{I}$ (e.g., URLs, movies), and tags $\mathcal{T}$. Objects are denoted by small letters prefixed by @ for users, $ for items, and # for tags, e.g., @marie, $Delicatessen, #French-Movie. When we use variables, they are always typed, e.g., @x for a user variable. The core of our model is a relation G. An entry $G(@u, \#t, \$i, w)$ with $w \in [0, 1]$ indicates that user $@u$ tagged item $\$i$ with tag $\#t$ and that the weight of this tagging is w. This social tag graph may be complemented with other database relations between tags (e.g., an ontology), between items (e.g., URLs from the same Web site), between users (e.g., contacts in Facebook, proximity based on GPS geo-localization). These relations will be used as standard database relations in queries, e.g., to restrict the social tag graph of interest. They may also be considered in defining object affinities.

From this social tag graph (and possibly other knowledge

*This work has been partially funded by the European Research Council under the European Community's Seventh Framework Programme (FP7/2007-2013) / ERC grant Webdam, agreement 226513. http://webdam.inria.fr/

relations), we define an *affinity distance* between objects of the same kind or of different kinds. There are many ways of defining distances between objects. Indeed, we believe that such a distance depends on the nature of the tagging system that is considered. The system should thus offer a library of such distance functions for the application developers to choose from. As we will see, users of the system will not have to modify or customize this distance. Their preferences will be taken into consideration at the query language level. In what follows, we assume these distances are defined.

In a next step, we use the affinity distance and standard clustering tools to cluster objects (i.e., users, items and tags). We do not insist these clusterings produce partitions, and the precise clustering algorithm used is left open to application developers. The result of this step is then groups of users with related tagging patterns, groups of tags that typically co-occur and collections of semantically related items.

The next step of knowledge extraction consists in lifting the affinity distance between objects to sets of objects. An "aggregation function" is used to compute the affinity between an object and a group of objects of the same kind or of some other kind. Finally, another aggregation function is used to capture affinity between a set of objects of some kind with a set of objects of a possibly different kind.

The last step of knowledge extraction consists in providing "semantic names" to objects or groups of objects resulting from the clustering. For example, one can name a group of users using tags or item collections they have most affinities with. The application programmer provides a default naming function using the query language introduced below.

The query language relies on an affinity predicate *Near* that specifies a ranked list of pairs sorted by affinity measure based on a data set. For instance, consider $Near(G, @X, \#t)$ where the variable $@X$ denotes a set of users and $\#t$ a tag. This returns pairs of groups of users (obtained in the clustering stage) and tags, ordered by the distance between the group and the tag. Recall that the variables we use are always typed and that we do not perform groupings and clusterings among objects of heterogeneous types (e.g., between @marie and $Delicatessen).

To express queries, we use a rule-based syntax that we illustrate with an example. Suppose that we want the groups of users, who have affinities with @marie, are interested in databases but not in XML. We can use the following query

$$query_1((\text{top-10})\ @X) \leftarrow Near(G, @X, @\text{marie}) \\ \wedge Near(G, @X, \#\text{database}) \wedge \neg Near(G, @X, \#\text{xml}).$$

The $\neg Near$ predicate is defined as the inverse of *Near*. Since affinities are measured from 0 to 1 (with 0 being the closest), we define $\neg Near(x, y, z) = 1 - Near(x, y, z)$ for each x, y, z. Each predicate in the query defines a ranked list of weighted results. These are joined together to produce the results. Of course, one typically does not want to get *all* but only some, the best results (e.g., the annotation at the head of the rule specifies top-10). One can also use a threshold, e.g., replace in the rule $Near(G, @X, @\text{marie})$ by $Near(G, @X, @\text{marie}) < 0.5$.

Now to present these groups to a human, one would like to attach some semantics to them using some terms, i.e., name them. Users can use this default naming or control more directly how they would like to assign names. For instance, in the running example, one can use the following query:

$$query_2(@X, (\text{top-5})\ \#t) \leftarrow Near(G, @X, \#t) \wedge query_1(@X).$$

For each of the top 10 groups in the previous results, we provide the 5 tags that best qualify them.

In the previous examples, we only use the social tag graph G. In queries, we can also use relations beyond G. For instance, for each of her friends, Marie can evaluate $query_1$:

$$query_3(@u, (\text{top-10})\ @X) \leftarrow Friends(@\text{marie}, @u) \\ \wedge Near(G, @X, @u) \wedge Near(G, @X, \#\text{database}) \\ \wedge \neg Near(G, @X, \#\text{xml})$$

where *Friends* is a classical relation.

Queries can also be specified graphically in a QBE style. The part of the query concerning the database relations is exactly like in QBE. Three areas are used for items, tags and users. Examples are used in the different areas to denote object variables as in QBE. Spheres are used to denote variables corresponding to groups of objects. Arcs between object examples and group examples denote affinities.

Navigation. For navigation, the user also sees three areas, one for each object type. Each area is divided into two parts, one for elementary objects and one for groups. At each stage of the process, each of these six sub-areas holds a ranked list. User actions result in modifying these lists hence, the dynamicity of navigation.

The user interacts with the system using two main actions: filtering and zooming. For filtering, the user can specify some positive or negative affinity with an object or a group of objects. The object may come from one of the six areas. It may also come from another window by cut and paste. The query is entered in a "query zone". Conditions may be added or removed from that zone. For instance, a user may want to remove some filtering that turned out to be too selective. For zooming, groups of objects typically go by a hierarchy. The ranking algorithm takes this into account and favors presenting a group A high in the result instead of presenting many subgroups $A_1 \dots A_n$, which would prevent other possibly interesting groups to appear high in the result. The user is able to zoom in A to see subgroups of interest.

References

[1] S. Amer-Yahia, J. Huang, and C. Yu. Building community-centric information exploration applications on social content sites. In *Proc. SIGMOD*, 2009.

[2] E. H. Chi and T. Mytkowicz. Understanding navigability of social tagging systems. In *Proc. CHI*, 2007.

[3] M. Harvey, M. Baillie, I. Ruthven, and D. Elsweiler. Folksonomic tag clouds as an aid to content indexing. In *Proc. SSM*, 2009.

[4] A. Joshi and J. Cho. Improving image search based on user created communities. In *Proc. SSM*, 2009.

[5] R. Li, S. Bao, Z. S. B. Fei, and Y. Yu. Towards effective browsing of large scale social annotations. In *Proc. WWW*, 2007.

[6] C. Marlow, M. Naaman, D. Boyd, and M. Davis. HT06, tagging paper, taxonomy, Flickr, academic article, to read. In *Proc. Hypertext*, 2006.

[7] S. Oldenburg, M. Garbe, and C. H. Cap. Similarity cross-analysis of tag / co-tag spaces in social classification systems. In *Proc. SSM*, 2008.

[8] E. Santos-Neto, M. Ripeanu, and A. Iamnitchi. Tracking user attention in collaborative tagging communities. In *Proc. Workshop on Contextualized Attention Metadata*, 2007.