# Interactive Recommendations in Social Endorsement Networks

Theodoros Lappas
UC Riverside, CA, USA
Computer Science Dept.
tlappas@cs.ucr.edu

Dimitrios Gunopulos
University of Athens
Dept. of Informatics and Telecommunications
dg@di.uoa.gr

## ABSTRACT

An increasing number of social networking platforms are giving users the option to endorse *entities* that they find appealing, such as videos, photos, or even other users. We define this model as a *Social Endorsement Network*, visualized as a bipartite graph with edges (endorsements) from users to endorsed entities. In this work, we formalize the problem of *interactive recommendations in social endorsement networks*: given a query of tags and a social endorsement network, the problem is to recommend entities that match the query and also share a significant number of common endorsers. We propose an efficient search engine for the solution of the problem, able to produce high-quality and explainable recommendations. The entire framework is designed in a principled and efficient manner, making it ideal for large-scale systems. In a thorough experimental evaluation on real datasets, we illustrate the efficacy of our methods and provide some valuable insight on social endorsement networks.

## Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Search process*; E.1 [**Data**]: Data Structures—*Graphs and networks*; H.2.8 [**Database Management**]: Database applications—*Data Mining*

## General Terms

Algorithms, Experimentation

## Keywords

Social Networks, Recommendation Systems, Social Endorsement, Twitter

## 1. INTRODUCTION

The desire of users to exchange information and share their personal opinions has been one of the main causes of the astounding popularity of social networks. Users use social networks to comment on a variety of different *entities*, such as photos, movies, products, or even other users. In many popular platforms this method of expression has been formalized, allowing users to express their approval of an entity by *endorsing* it. On `Facebook.com`, users have the option to "Like" different types of entities, including photos, videos, celebrities and commercial products. On `CiteULike.org` users can show their approval of a published paper by including it in their "Library". On `Flickr.com` users can add a picture to their "Favorites" folder. Finally, on `Twitter.com` users can express their interest and approval by becoming "followers" of other users.

By visualizing an endorsement as an edge from a user to an entity, we can view a *Social Endorsement Network* as a bipartite graph $G = (U, V, E)$, where $U$ is the set of users, $V$ is the set of entities, and $E$ is the set of endorsement edges. A problem that naturally arises in such a network is recommending to the user other entities that he is likely to be interested in. As we show in our work, the information encoded in the graph of the social endorsement network can serve as an exceptional foundation for a solution to this problem. The intuition is simple: an endorsement serves as a verification that the user approves the endorsed entity. Examining the set of entities that are endorsed by a single user can provide some information on his preferences, but it does not answer the most important question: *Why did the user choose to endorse this particular entity?*

To answer this question, we call upon the wisdom of crowds: first, we find groups of entities that are endorsed by the same large groups of users. For each group, we then examine the common characteristics of the included entities and identify the aspects that truly appealed to the same large set of users. The product of this first phase is a collection of groups. For each group, we have a set of *tags* that encode its most attractive and characteristic aspects. Given this information, the next step toward a great recommendation framework comes naturally: we make our system *interactive*, allowing the user to specify his own personal interests in the form of a textual query. The submitted query is then streamed through the mined groups, in order to identify those that best match the user's interests. The main problem addressed in this paper is the following:

PROBLEM 1. Given a user-submitted query and a social endorsement network $G$, we want to identify and recommend groups of entities that match the query and also share a significant number of common endorsers.

In order to accurately encode the user's preferences, we formalize queries as sets of tags (keywords). This is an intuitive and flexible mechanism, with which practically every user is


*RecSys2010,* September 26–30, 2010, Barcelona, Spain.

familiar. Below are some examples of relevant queries, as could be formed in popular social networking platforms:

- `Twitter`: Recommend *female singers* from *the USA*.
- `Facebook`: Recommend *Chinese Restaurants in San Fransisco*
- `CiteULike`: Recommend research papers *on Social Networks*

The underlined terms represent the query-tags that encode the user's interests. A possible recommendation of our framework for the 1st query is : "Whitney Houston, Mariah Carey and Celine Dion match the query and also share 50,000 followers". The interaction with the user and the authority offered by the large number of common endorsers make our recommendations *explainable* and intuitive to the user. Explainable recommendations are increasingly popular and have been the focus of numerous research efforts [24].

An overview of our framework is shown in Figure (1): Given a social endorsement network, we first extract groups of entities that share a significant number of common endorsers. Next, we identify the appropriate set of tags for each of the reported groups. Taking the assigned tags into consideration, we then apply a filter that eliminates redundant groups (i.e. groups that can be induced by others), and produces a compact and informative corpus. The final corpus is then organized in an appropriate index structure. By combining the index with an efficient algorithm for query evaluation, we create a search engine able to recommend groups of entities that are relevant to the given query.

**Contribution:** Our work is the first to formalize and solve the problem of *interactive recommendations in social endorsement networks*. We thoroughly discuss the architecture of the proposed framework and demonstrate its efficacy through a thorough experimental evaluation on real datasets. The benefits of our approach are clear:

- It is interactive, allowing the user to repeatedly query the system for different types of entities.
- Its principled and efficient architecture make it ideal for large-scale systems.
- The recommended entities come organized into groups, with each group representing a different cohesive set of similar entities.
- The recommendations are easily *explainable* and, thus, more intuitive to the user.

Another significant contribution of this paper is the release of a brand new dataset (crawled from `Twitter.com`), which is ideal for research on social endorsement networks.

## 1.1 RoadMap

We begin in Section 2 with an overview of the related work. In Section 3 we discuss the identification of popular groups of entities. In Section 4 we discuss the process of tagging the reported groups. In Section 5 we introduce a principled filtering method for the elimination of redundant groups. In Section 6 we describe our interactive recommendation mechanism. In Section 7, we present our experimental evaluation. Finally, we conclude in Section 8 with a brief overview of the paper.

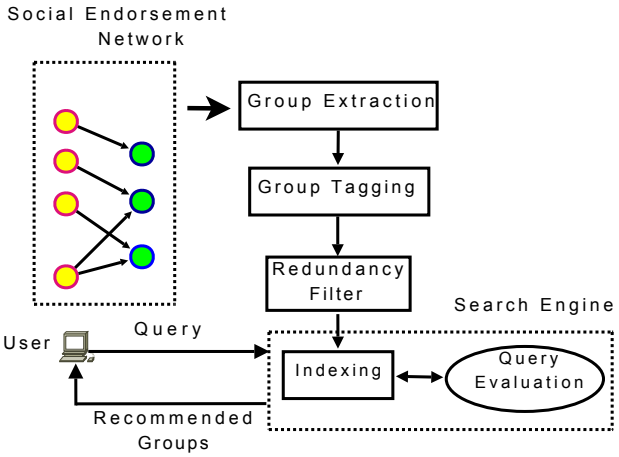



Figure 1: An overview of our complete search framework

# 2. RELATED WORK

To the best of our knowledge, ours is the first work to consider the problem of *interactive recommendations in social endorsement networks*. Nonetheless, our work has ties to numerous fields.

Our recommendation system is defined in the context of a social endorsement network, which we formalize in the paper. Social endorsement has been also considered in the past, albeit in a different context. Kunegis et al. [12] analyze different aspects of the social graph from `Slashdot.org`, where users have the option to tag others as "friends" or "foes", thus providing positive or negative endorsements. In another relevant paper, Leskovec et al. [13] discuss the prediction of positive and negative edges in social networks.

Different types of recommendation systems have been proposed in the broad context of social networks: Guy et al. [7] considered *the familiarity network* among the users to support their recommendation system. In a related paper, Bonhard et al. [1] explore how the familiarity and similarity among users can be utilized to improve recommendations.

In the first phase of our framework we mine frequent (popular) groups of entities. Pattern mining has been explored in the context of recommendation systems [14, 17, 16], albeit in contexts that are completely different to our own. Further, our framework employs a type of *social tagging*. Social tagging is an increasingly popular research topic, following the popularity of social networking platforms. A significant amount of work has been devoted to methods for automatic tag extraction [11, 3, 23, 6] and to using tagging to enhance recommendation systems [26, 4, 21, 15, 22].

Users can interact with our recommendation system via queries. Interactivity in the context of recommendations systems has also been studied in the past: Viappiani et al. [25] propose a conversational recommender that collects information and adapts to the user's preferences. In a similar setup, Bridge at al. [2] try to identify the most suitable items for a user, while keeping query updates to a minimum. Schenkel et al. [20] propose an incremental top-k querying-algorithm that takes into consideration the relationships among users to rank tagged entities.

Finally, our work has ties to collaborative filtering, an extensively studied problem in the context of recommendation systems [8, 19, 18, 27]. Though relevant, our work is the

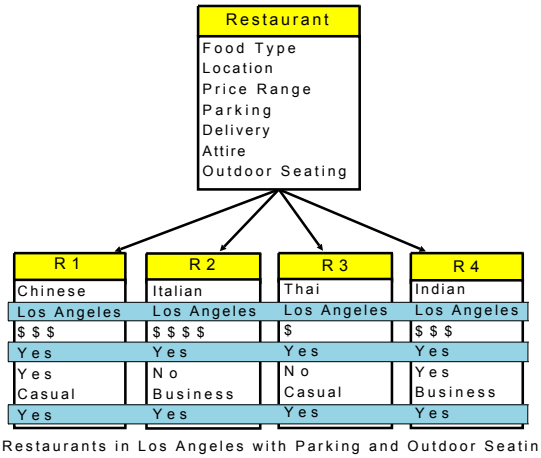



Figure 2: An example of using tags to identify the correlation among the entities in a group. In this case, the four entities are all Restaurants in Los Angeles that offer both Parking and Outdoor seating.

first to focus on social endorsement networks and enables *interactive* and *explainable* recommendations.

## 3. EXTRACTION OF POPULAR GROUPS

In this section, we describe the process of identifying groups of entities with a significant number of common endorsers, given a social endorsement network. This is only the first phase of our framework, albeit an important one, since it produces an initial collection of popular entity-groups. These groups will then be processed, tagged, filtered, and finally organized toward an efficient recommendation engine.

We formalize the problem of extracting popular groups as an instance of the problem of mining frequent itemsets: we are given a set of transactions, where each transaction includes a set of items. We then want to find groups of items that were often grouped together. In our context, a transaction is the set of entities that are endorsed by a user. Formally, we define the problem as follows:

PROBLEM 2. *Given a social endorsement network $G = (U, V, E)$ and a group of endorsed entities $g \in 2^V$, let $N(g)$ return the set of common endorsers of $g$ in $G$. Then, find the set of entity-groups $\mathcal{G}$, so that*

$$\mathcal{G} = \{g \mid g \in 2^V,\ |N(g)| \geq T\} \quad (1)$$

As formulated above, the problem asks for all groups that have at least $T$ endorsers in common. In the Experiments section, we show how tuning the value of $T$ affects the number of reported groups.

By representing the set of entities endorsed by each user as a transaction, Problem 2 can be efficiently solved by any of the popular algorithms for mining frequent itemsets. In our experiments, we use the algorithm proposed in [9], which proved efficient enough to easily handle a database of over six million transactions.

## 4. GROUP TAGGING

After we obtain the popular groups, the next step is to tag them in a way that facilitates search. Given a group, we want to answer the following question: *Why where these entities endorsed by the same large set of users?* To answer this, we need to identify the common characteristics that make these entities appealing to the same crowd. In order to achieve this, we need to obtain and record information on the different attributes of each entity. For example, if the endorsed entity is a restaurant, the list of attributes may include the type of food served or the restaurant's location. Such information can be easily encoded in the form of *tags*. Tagging is an increasingly popular feature, available in many social networking platforms. For example, in Flickr and Facebook, users can tag photos and videos with descriptive terms or phrases of their choice. Such tags can be submitted by users who manually assign descriptive tokens to each entity, or produced by an automated tagging method [23, 3, 11, 6]. **Our framework is compatible with any tagging method** that can assign a set of tags $ts(e)$ to each entity $e$. These TagSets can be then used to compute the TagSet $ts(g)$ of an entire group $g = \{e_1, e_2, ...\}$ as follows:

$$ts(g) = \bigcap_{e \in g} ts(e) \quad (2)$$

Even though this definition worked superbly in our experiments, it can easily be relaxed to include tags that appear in **a large majority of the group's entities, rather than all of them**.

**Getting the Tags**:In the case of automated tag-extraction, a question that arises is the following: *Where can we mine the required TagSets from?* Typically, automated methods are based on a piece of descriptive textual information that is available for each entity. In cases where the entity is itself consisting of text (e.g. a webpage or other document), then obtaining such information is a non-issue. Given the abundance of information that are available on the Web, such text summaries can be easily obtained for virtually any type of entity: Facebook Groups and Fan Pages have a short passage describing the nature and purpose of the group. On Twitter and MySpace, users provide a self-written description in their profiles. Informative pieces of text can also be extracted from sources outside the network: if the endorsed entity is a product, the text from the product's official website can serve as a descriptive summary. If the entity is a movie, the source can be the plot summary from sites like `imdb.com`. If the entity is an influential person, we can use the text from his personal page or the respective entry on sites like `Wikipedia.com`. Another option is to submit the name of the entity (e.g. a restaurant) as a query to a search engine. The aggregated text of the top-k returned webpages can then be mined for frequent and informative tags.

**Structured Content**: In many cases, informative content can be found in a semi-structured format in the Web. The templated entries on the right side of the pages on Wikipedia.com serve as a characteristic example of such a format. Such templates facilitate the direct extraction of informative tags. An intuitive way to compose TagSets from such data is via the construction of *profiles*. Examples are given in Figures (2) (*Restaurant*) and (3) (*Athlete, Singer, Politician*). The profiling process adds an additional level of abstraction and facilitates the grouping of different entities; Since the available entities are evaluated on a fixed set of attributes, it is easier to identify common characteristics

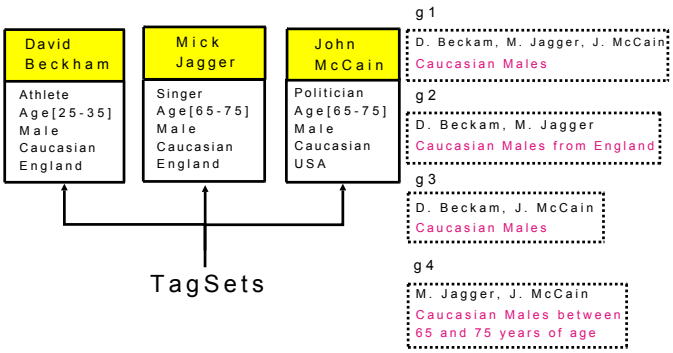



Figure 3: An example of redundancy: $g_3$ is pruned, since it is a subset of $g_1$ and $ts(g_3) \subseteq ts(g_1)$.

and decode the correlation among the members of a group. An illustrative example is given in Figure (2): we are given a profile representing a restaurant, along with a group of four matching entities. In this case, since all the entities in the group belong to the same profile, the attribute-set of the group includes only the seven attributes of the profile: {*Food Type, Location, Price Range, Parking, Delivery, Attire, Outdoor Seating*}. Then, the TagSet of the group will contain the attribute values that remain the same for all restaurants. In this case: $ts(g)$ ={*Los Angeles, With Parking, With Outdoor Seating*}. These three tags compose the TagSet of the group , and reveal why such a large number of users endorsed all four restaurants.

## 5. ELIMINATING REDUNDANCY

In this section we identify a type of redundancy among entity-groups and propose a principle method to eliminate it. Consider the example given in Figure (3): we are given three entities: David Beckham (athlete), John McCain (politician) and Mick Jagger (singer). We assume that the three individuals have a significant number of common endorsers and have been identified as a popular group. On the right, the figure shows all the possible (sub)groups with at least two members, along with their respective TagSets. We observe that $g_3$ is a subset of $g_1$, and also bears no additional tags (i.e. $g_3 \subset g_1$ and $ts(g_3) = ts(g_1) = \{Caucasian, Male\}$). Therefore, $g_3$ is redundant and we can safely prune it without losing any information. On the other hand, even though both $g_2$ and $g_4$ are also subsets of $g_1$, they also have richer TagSets and thus have to be included in the final set. Formally, we define the problem as follows:

PROBLEM 3. *Given a set of groups $\mathcal{G}$, find a filtered set $\mathcal{G}^* \subseteq \mathcal{G}$, so that:*

1. $\forall g \in \mathcal{G},\ \exists g' \in \mathcal{G}^* s.t.\ \{ts(g) \subseteq ts(g')\ and\ g \subseteq g'\}$
2. *$\mathcal{G}^*$ is the smallest set among all those that satisfy the 1st condition.*

The first condition requires that, for every group $g \in \mathcal{G}$, there exists a group $g' \in \mathcal{G}^*$ that contains all the entities of $g$, and is also tagged with all the tags included in $ts(g)$ (among others). The second condition implicitly asks for a set consisting exclusively of non-redundant groups: even if a single redundant group exists in $\mathcal{G}^*$, we can safely prune it and thus get a set of smaller size. In order to address this problem, we propose the `GroupFilter` algorithm, which reports a filtered set, consisting only of non-redundant groups. The pseudocode is given in Algorithm (1).

**Algorithm 1** `GroupFilter`

**Input:** Set of Entity Groups $\mathcal{G}$
**Output:** Filtered set of non-redundant Groups $\mathcal{G}^*$

```
1: filteredIndex ← ∅  // supports superset queries.
2: Sort G in desc order by group size
3: for each group g ∈ G do
4:     isRedundant ← false
5:     S ← lookup_sups(filteredIndex, g)
6:     for (each super-group S ∈ S) do
7:         if (ts(g) ⊆ ts(S)) then
8:             isRedundant ← true
9:             break
10:    if (!isRedundant) then
11:        filteredIndex.insert(g)
12: return filteredIndex.getGroups()
```

**Details of Algorithm (1)**: The input consists of the complete set of groups $\mathcal{G}$, while the output is a filtered set $\mathcal{G}^*$ of all non-redundant groups. The algorithm maintains an index of the non-redundant groups ($filteredIndex$). For every group $g$, we probe the index to retrieve the set of (non-redundant) super-groups (i.e. groups that contain, among others, all the entities included in $g$). Any structure that supports such *superset queries* can be used to build the index. We use the UBTree [10], a simple and efficient structure for indexing sets. We refer the reader to the original paper for more details on the structure.

`GroupFilter` begins by sorting all the groups by size (i.e. number of members), in descending order. This ensures that all super-groups of a redundant group will be evaluated before it is. Then, for each group $g \in \mathcal{G}$, we probe the index to retrieve its set of super-groups $\mathcal{S}$. If there exists a super-group $S \in \mathcal{S}$ that has all the tags of $g$ (i.e. $ts(g) \subseteq ts(S)$), then $g$ is redundant and can be ignored. Note that, since the TagSet of a group is guaranteed to contain all the tags included in any of its super-groups, it is sufficient to check if $|ts(g)| \leq |ts(S)|$. If there exists no superset $S$ that satisfies this inequality, $g$ is non-redundant and can be safely inserted in the index. At this point we know that $g$ is non-redundant, otherwise it would have been pruned earlier (since the groups in $\mathcal{G}$ are sorted). This guarantees that our index only contains non-redundant groups, leading to a structure that is smaller and faster to probe. After all the groups have been evaluated, the algorithm returns the filtered set of non-redundant groups.

## 6. INTERACTIVE RECOMMENDATIONS

In this section, we describe a search engine for the recommendation of entity-groups. Conceptually, we want to respond to queries of the type: *"Find large groups of entities that share a set of tags $\{t_1, t_2, ..., t_m\}$, and also have a significant number of common endorsers'.* By asking for larger groups, we maximize the amount of information returned to the user, who can then further investigate the numerous entities in a group. Maximizing the number of endorsers would not be reasonable in our context, since it would lead

**Algorithm 2** `TopKFinder`

**Input:** Inverted Index *index*, query of tags $q = \{t_1, t_2, ..., t_m\}$, int $k$
**Output:** set of $top-k$ matching groups

1: $TopK \leftarrow \emptyset$ // sorted, holds at most `k` elements
2: $\mathcal{L} \leftarrow \{Li \mid t_i \in q\}$
3: **while** $TopK.size() < k$) **do**
4: **for** (*every List* $L \in \mathcal{L}$) **do**
5: $g \leftarrow getNext(L)$
6: **if** ($g =$ `null`) **then** *return* $TopK$
7: **else if** ($L'[g] \neq \emptyset, \forall L' \in \mathcal{L}$) **then**
8: $TopK.insert(g)$
9: **return** $TopK$

to trivial, single-entity groups. We formalize the problem as one of top-k evaluation, as follows:

PROBLEM 4. *Given a set of entity-groups* $\mathcal{G} = \{g_1, g_2, ..., g_n\}$ *and a query of tags* $q = \{t_1, t_2, ..., t_m\}$*, find the* $k$ *largest groups from* $\mathcal{G}$ *that satisfy the following condition:*

$$ts(g) \cap t_i \neq 0, \ \forall t_i \in q \tag{3}$$

Conceptually, Problem 4 asks for the $k$ largest groups that contain all the tags of the query in their respective TagSets. To address the problem, we use an inverted index structure, mapping each tag to the list of groups that contain it. The groups in each list are primarily sorted in descending order by their size. In addition, a secondary sort is done by the number of endorsers, also in descending order. This ensures that, among groups of equal cardinality, those with the highest number of endorsers will have priority. Given the inverted index, we can retrieve the top-k results using a simple evaluation algorithm, shown in Algorithm (2).

The algorithm, which we refer to as `TopKFinder`, begins by retrieving the set $\mathcal{L}$ of group-lists that correspond to the $m$ tags of the query. Then, for each list $L \in \mathcal{L}$, the $getNext(L)$ function is used to retrieve the next group under sorted access. The function returns `null` if $L$ has been exhausted. For each candidate group $g$ in $L$, the algorithm checks if it is also included in all other lists in $\mathcal{L}$. Each list is checked using a random access probe, supported by an appropriate structure. An example of such a structure is a hash-set, where each group is hashed by a label consisting of its tags (or tag IDs) in lexicographical order. If $g$ is indeed included in all the lists, then it is included in the $top-k$. The algorithm continues, until $k$ groups have been identified or until at least one of the lists has been exhausted (Line 6).

`TopKFinder` is essentially a simplified version of the popular Threshold Algorithm (`TA`) [5]. In the typical use case of `TA`, the score of each object (group) is different in every list. Therefore, the algorithm has to retrieve the respective scores of the object from all the lists, and compute the cumulative value. A threshold mechanism is used as a termination criterion. In our case, this mechanism is redundant, since the score of each group is the same in all the lists (i.e. equal to the group's size).

With `TopKFinder`, we can evaluate any multi-tag query submitted by a user and efficiently solve Problem 4. Even though the inverted index itself is not original, its application to interactive recommendation systems is one of the novelties of our work.

# 7. EXPERIMENTS

In this section we present the thorough experimental evaluation that we conducted to evaluate our approach. We begin with a discussion of our datasets and proceed with a detailed discussion of each experiment.

## 7.1 Datasets

**The `Twitter` dataset:** This is a new corpus, which we composed particularly for the purposes of this paper. **The data is available upon request.** The corpus is built based on data collected from Twitter.com, a popular social networking platform, where one can "follow" other users and get updates on their posts. The dataset is constructed as follows: first, we obtain the list of the 1000 users in Twitter with the most followers (from TwitterHolic.com. We then crawl Twitter to retrieve the set of followers for each of these users. We focus on the top-1000 users since they are typically well-known public figures, making the verification of our results intuitive.

After a detailed inspection of the data, we identified five entity profiles that represent the most dominant types among these highly-followed entities: *Music Artist, TV Personality, Athlete, Business Person* and *Other* (e.g. authors, bloggers, politicians). These include real-life public figures and share the following attributes: the occupation (also the name of the profile), the gender, the age group (e.g. 20-30), the country of origin, the state of origin (or city if non-usa), and the particular type or genre each person belongs to, within their bounds of their profession. This includes the music type(s) for artists, the genre(s) for TV personalities, the different properties of people assigned to the *Other* profile (e.g. author, blogger, columnist), and the specific sport for athletes.

**The TagSets for the followed individuals are populated in an entirely automated manner**. we build a focused crawler which, given a name, retrieves the required information from the Web. For TV Personalities, we use the `imdb.com` website, which hosts all the required information, including the relevant genres for each person (we only kept the top-3 genres per person, as ranked by imdb). For all other profiles we use `Wikipedia.com`, which maintains all the required profile information in a separate entry within the HTML template. Our crawler retrieved the profile information for about 500 individuals. A manual examination of the unidentified entities verified that they were either not real-life people (e.g. cnn.com), spam (e.g. fake accounts), or simply users for which the information was not available on Wikipedia or imdb. The 500 profiled individuals constitute the set $V$ of endorsed entities, in the context of a social endorsement network $G = (U, V, E)$. The set of endorsers $U$ is represented by the entire population of followers, which consisted of $6,436,382$ distinct Twitter users (by username). The minimum number of endorsers per group (as a percentage of the total number of users) was set to 0.007.

**The `DBLP` dataset:** To create the second benchmark for our experiments, we use a snapshot of the data taken from the DBLP Bibliography Server on April 12, 2006[1] .

For each published paper, the snapshot contains the title, the set of authors, and the set of cited papers. Using this information, we construct our social endorsement network $G = (U, V, E)$ as follows: the set of endorsers $U$ consists of all the papers that reference at least one other paper. The

[1]http://kdl.cs.umass.edu/data/dblp/dblp-info.html

set of endorsed entities $V$ consists of the authors that have at least one citation to one of their papers. Thus, the set of endorsement edges $E$ is populated by adding an edge from a paper in $U$ to an author in $V$, if the paper cites the author's work. Finally, the TagSet of each author consists of the distinct (stemmed) terms that appear in his papers' titles. Alternatively, one could use the set of authors to represent both the endorsers and the endorsed entities. However, this would fail to capture cases where an author $A$ cites multiple papers of another author $B$. The collection includes 456764 distinct authors and 728510 research papers (we discarded PhD and masters theses). The minimum number of endorsers per group (as a percentage of the total number of users) was set to 0.005.

## 7.2 Quantitative Analysis

Here, we perform a detailed quantitative analysis of our framework on the `Twitter` and `DBLP` datasets.

**Group Size Distribution:** Figures 4(a) and 4(c) show the distribution of the various group sizes for `DBLP` and `Twitter`, respectively. The x-axis holds the different group cardinalities, while the y-axis shows the number of groups with each particular cardinality (in log-scale). For `DBLP`, the majority of the groups consist of 2-10 authors, while very few have over 20 members. For `Twitter`, the reported groups are generally smaller, with the largest groups consisting of 12 entities. This can be explained by the fact that the number of distinct entities in `Twitter` is considerably smaller (500 individuals, Vs. several thousand authors in `DBLP`), making it less likely to find large groups that share a significant number of followers and also have overlapping TagSets. In addition, the profiles in `DBLP` typically consist of numerous tags (twelve per author, on average), making it easier to identify groups of authors with overlapping TagSets.

**Inverted Index:** Next, we evaluate the inverted-index structure employed by our framework, by examining the size of the group-lists mapped to the indexed tags. For both datasets, a clear majority of the lists in the inverted index are small, leading to a compact structure that is easy to stored and probe.

Figures 4(b) and 4(d) show histograms of the list sizes for `DBLP` and `Twitter`, respectively. Each bar represents a size range (e.g. the first bar on Figure 4(b) represents all lists of size between 1 and 20). The y-axis (in log-scale) marks the percentage of tags that are mapped to a list with a size that falls within the respective range.

For `DBLP`, Figure 4(b) shows that over 50% of the tags where included in the TagSets of less than 20 groups. The 3 most popular tags were "databases", "systems" and "data", which appeared in 22326, 20015 and 19645 groups, respectively. These fall within the 2% of the tags that were mapped to more than 3000 groups.

For `Twitter`, Figure 4(d) shows that around 50% of the lists in the index contained between 1 and 5 groups. For this dataset, the 3 most popular tags where "Male", "Age[30-40]" and "TV Personality", which appeared in 48033, 2441 and 2054 groups, respectively.

## 7.3 Qualitative Analysis

Next, we evaluate the quality of our results on the `Twitter` and `DBLP` datasets. First, we create a set of 10 queries for each dataset. For `DBLP`, the first 9 queries are taken from the session names of the SIGKDD conference from 2006 (the same year when the data was collected). We also added a 10th query ("world wide web") for relevance. For `Twitter`, the queries consist of popular tags from the corpus, in order to enhance the verifiability of the results. For each query, we report the top-1 group of entities returned by our search framework, as well as the number of common endorsers per group. The results for `DBLP` and `Twitter` are shown in Tables 1 and 2, respectively.

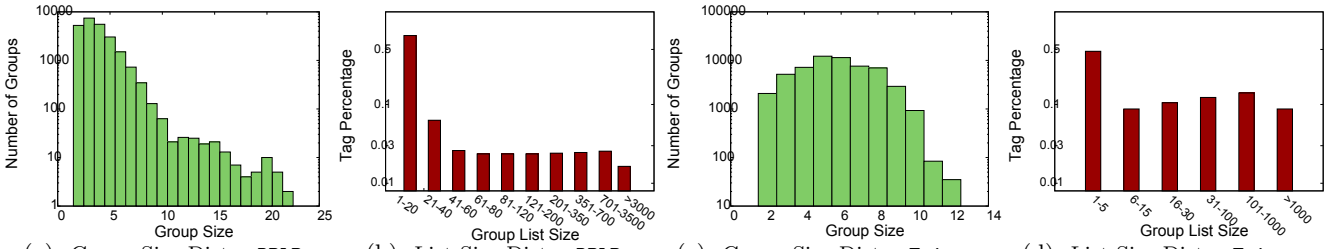
**Discussion of the Results:** For `DBLP`, we are looking for large groups of authors that have the terms of the query in their TagSets, and whose work is often cited in the same papers. As can be seen from the table, the reported groups consist of highly-cited and well-known authors. This was anticipated, since authors with numerous papers are not only more likely to be cited, but also more likely to have larger, more diverse TagSets that overlap with those of other authors. Certain names are included in the groups for many queries, indicating that the respective authors have been active in different areas, while being able to attract a significant number of citations. Another important observation is that co-authorships can be a deciding factor in the formation of groups of co-cited entities. A characteristic example is that of entry #9: Won Kim, Nat Ballou, Jorge F. Garza and Darrell Woelk were all included in the top-1 group for the query "privacy", partly due to their highly-cited paper "A Distributed Object-Oriented Database System Supporting Shared and Private Databases".

For `Twitter`, we are looking for groups of individuals that match the specified query, and share a significant number of followers. As can be seen in Table 2, the reported groups consist of people who are well-known in their respective fields. Less focused queries lead to more diverse groups: the group reported for "Men between the ages of 30 and 40" consists of 2 actors, 1 athlete and 4 people from the business world. Interestingly enough, over 68000 Twitter users chose to follow these individuals.

Particularly interesting observations can be made from examining the groups reported for queries #2, #3 and #4, which are ordered from the more general to the more focused one. For query #2, a group of 4 pop-music artists is reported. Query #3 is more refined, asking for groups of *female* pop-music artists. Britney Spears is the only person reported for both queries, indicating that she shares a significant number of followers with both male and female artists. Note that Britney Spears had the third largest number of followers among all the individuals in `Twitter`. The first 2 positions are held by Ashton Kutcher and Ellen DeGeneres (TV personalities), who are also included in top-1 groups. Britney Spears is also included in the top-1 group for query #4. This query is even more focused, asking for women that are also between the ages of 20 and 30. An interesting observation here is that the reported group has more followers than those reported for the previous two queries, even though it is more refined. This is because the group has only 3 members (while the groups for queries #2 and #3 had 4). As described in Section 6, we prefer large groups, while using the number of followers for secondary ranking.

## 7.4 Redundancy Filtering

Here, we evaluate the redundancy filter described in Section 5. First, we use the mechanism described in Section 3 to obtain the complete set of popular groups for both datasets. We repeat the experiments for different values




(a) Group-Size Distr., DBLP (b) List-Size Distr., DBLP (c) Group-Size Distr., Twitter (d) List-Size Distr., Twitter

Figure 4: Quantitative analysis of our framework, as applied on the Twitter and DBLP datasets. Figures 4(a) and 4(c) show the distribution of the reported groups' sizes for DBLP and Twitter, respectively. Figures and 4(b) and 4(d) show histograms of the different list sizes in the Inverted Index. For example, for DBLP, more than 50% of the tags were mapped to lists of at most 20 groups.

for the minimum number of common endorsers $T$, expressed as a percentage of the total number of users (i.e. followers on Twitter and papers on DBLP). For each value of $T$, we apply the GroupFilter algorithm and compute the number of eliminated groups. The results are shown in Figures 5(a) and 5(b). The x-axis represents the common endorsers threshold $T$, while the $y-axis$ holds the number of reported groups (in logarithmic scale). For each value of $T$, we plot the number of groups before and after the application of the redundancy filter.

The results illustrate that, for DBLP, redundant groups cover a very high percentage of the unfiltered set. Our filter eliminates this redundancy and produces a compact and informative set. This translates to significant computational savings for a framework that needs to maintain and search the corpus of groups. For Twitter, the volume of pruned groups was reduced. This can be explained by the fact that, especially for higher values of $T$, the size of the unfiltered corpus was already quite small, compared to the respective number for DBLP. However, for lower values of $T$, the number of filtered groups was still significant. For example, for $T = 0.007$, the number of groups dropped from 6436171 to 56695, while for $T = 0.008$ it went from to 27410 to 5756.

# 8. CONCLUSION

In this paper, we formalized the problem of *interactive recommendations in social endorsement networks*. We presented an efficient, query-driven framework for the solution of the problem, able to make high-quality and explainable recommendations. In addition, our framework is equipped with a filtering mechanism for the elimination of redundancy, which can be used reduce the size of the corpus and produce a crisp and informative dataset. The entire recommendation system is designed in a principled and efficient manner, making it ideal for large-scale systems. Finally, we illustrated the efficacy of our methods in a thorough experimental evaluation on real datasets.

**Acknowledgements:** This research was partially supported by the MODAP and the SemsorGrid4Env EU projects and by NSF.

# 9. REFERENCES

[1] P. Bonhard and M. A. Sasse. 'knowing me, knowing you' – using profiles and social networking to improve recommender systems. *BT Technology Journal*, 2006.

[2] D. Bridge and F. Ricci. Supporting product selection with query editing recommendations. In *RecSys '07*.

[3] P. A. Chirita, S. Costache, W. Nejdl, and S. Handschuh. P-tag: large scale automatic generation of personalized annotation tags for the web. In *WWW '07*.

[4] M. de Gemmis, P. Lops, G. Semeraro, and P. Basile. Integrating tags in a semantic content-based recommender. In *RecSys '08*.

[5] R. Fagin, A. Lotem, and M. Naor. Optimal aggregation algorithms for middleware. In *PODS '01*.

[6] N. Garg and I. Weber. Personalized, interactive tag recommendation for flickr. In *RecSys '08*.

[7] I. Guy, N. Zwerdling, D. Carmel, I. Ronen, E. Uziel, S. Yogev, and S. Ofek-Koifman. Personalized recommendation of social software items based on social relations. In *RecSys '09*.

[8] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl. Evaluating collaborative filtering recommender systems. *ACM Trans. Inf. Syst.*, 2004.

[9] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. pages 487–499, 1994.

[10] J. Hoffmann and J. Koehler. A new method to index and query sets. In *IJCAIŠ99*.

[11] R. Krestel, P. Fankhauser, and W. Nejdl. Latent dirichlet allocation for tag recommendation. In *RecSys '09*.

[12] J. Kunegis, A. Lommatzsch, and C. Bauckhage. The slashdot zoo: mining a social network with negative edges. In *WWW '09*, NY, USA.

[13] J. Leskovec, D. Huttenlocher, and J. Kleinberg. Predicting positive and negative links in online social networks. In *WWW '10*.

[14] W. Lin, S. A. Alvarez, and C. Ruiz. Efficient adaptive-support association rule mining for recommender systems. *Data Min. Knowl. Discov.*, 2002.

[15] A. K. Milicevic, A. Nanopoulos, and M. Ivanovic. Social tagging in recommender systems: a survey of the state-of-the-art and possible extensions. *Artif. Intell. Rev.*, 2010.

[16] B. Mobasher, H. Dai, T. Luo, and M. Nakagawa. Effective personalization based on association rule discovery from web usage data. In *WIDM '01*.

[17] J. J. Sandvig, B. Mobasher, and R. Burke. Robustness of collaborative recommendation based on association rule mining. In *RecSys '07*.

[18] B. Sarwar, G. Karypis, J. Konstan, and J. Reidl. Item-based collaborative filtering recommendation algorithms. In *WWW '01: Proceedings*.

[19] J. B. Schafer, D. Frankowski, J. Herlocker, and S. Sen. In *The Adaptive Web: Methods and Strategies of Web Personalization*. 2007.

[20] R. Schenkel, T. Crecelius, M. Kacimi, S. Michel, T. Neumann, J. X. Parreira, and G. Weikum. Efficient top-k querying over social-tagging networks. In *SIGIR'08*.

[21] S. Sen, J. Vig, and J. Riedl. Tagommenders: connecting users to items through tags. In *WWW '09*.

[22] A. Shepitsen, J. Gemmell, B. Mobasher, and R. Burke. Personalized recommendation in social tagging systems using hierarchical clustering. In *RecSys '08*.

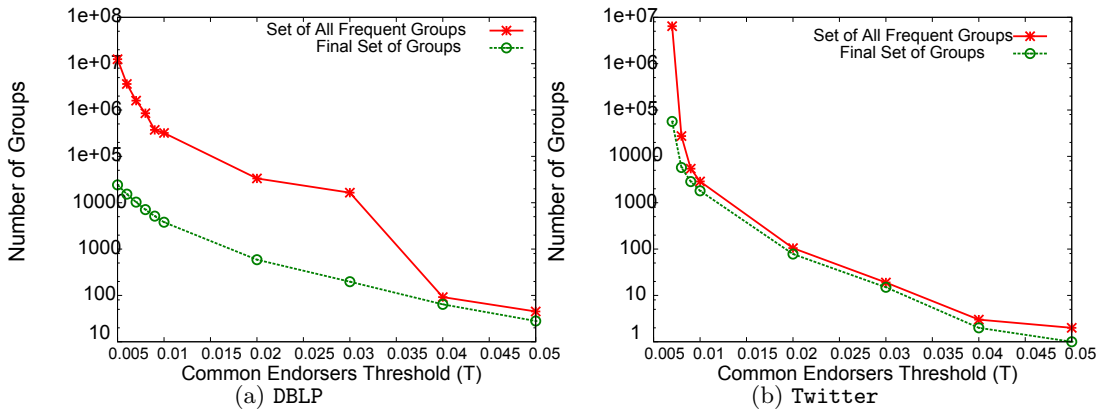
[23] Y. Song, Z. Zhuang, H. Li, Q. Zhao, J. Li, W.-C. Lee, and C. L. Giles. Real-time automatic tag recommendation. In *SIGIR '08*.

[24] N. Tintarev and J. Masthoff. A survey of explanations in recommender systems. In *ICDEW '07*.

[25] P. Viappiani, P. Pu, and B. Faltings. Conversational recommenders with adaptive suggestions. In *RecSys '07*, 2007.

[26] V. Zanardi and L. Capra. Social ranking: uncovering relevant content using tag-based recommender systems. In *RecSys '08*.

[27] Y. Zhen, W.-J. Li, and D.-Y. Yeung. Tagicofi: tag informed collaborative filtering. In *RecSys '09*.




(a) DBLP (b) Twitter

Figure 5: Effects of the Group Redundancy Filter for the Twitter and DBLP datasets.

Table 1: Groups of Authors in DBLP

| | Tag-Query | Top-1 Group of Authors | # Common Citations |
|---|---|---|---|
| 1. | classification | Tomasz Imielinski, Rakesh Agrawal, Sakti P. Ghosh, Balakrishna R. Iyer, Arun N. Swami | 54 |
| 2. | web mining | Serge Abiteboul, Stefano Ceri | 70 |
| 3. | clustering | Jiawei Han, Philip S. Yu, Rakesh Agrawal, Jong Soo Park, Arun N. Swami | 43 |
| 4. | graph mining | Jiawei Han, Philip S. Yu, Rakesh Agrawal, Ming-Syan Chen | 54 |
| 5. | time series | H. V. Jagadish, R. Ramakrishnan | 127 |
| 6. | pattern mining | Jiawei Han, Philip S. Yu, R. Agrawal, R. Srikant, Jong Soo Park, Ming-Syan Chen | 49 |
| 7. | text mining | Rakesh Agrawal, Ramakrishnan Srikant, Heikki Mannila | 71 |
| 8. | structured data | Nievergelt, C. Faloutsos, H. Hinterberger, B. Seeger, J. , K. C. Sevcik, H.-P. Kriegel, A. Guttman | 55 |
| 9. | privacy | Won Kim, Nat Ballou, Jorge F. Garza, Darrell Woelk | 228 |
| 10. | world wide web | Alberto O. Mendelzon, Alon Y. Halevy, Anand Rajaraman, Joann J. Ordille | 40 |

Table 2: Groups of Infividuals in Twitter

| | Tag-Query | Top-1 Group of Individuals | #Followers |
|---|---|---|---|
| 1. | Athlete | Shaquille O'Neal (basketball), Lance Armstrong (cycling), Tony Hawk (skateboarding) | 83309 |
| 2. | Music Artist, Pop | Britney Spears, Diddy, MC Hammer, Sara Bareilles | 79443 |
| 3. | Music Artist, Pop, Female | Britney Spears, Ashlee Simpson, Sara Bareilles, Maledy Moore | 69056 |
| 4. | Music Artist, Pop, Female, Age[20-30] | Britney Spears, Ashlee Simpson, Lily Rose Allen | 99765 |
| 5. | Business person, Age[30-40] | Evan Williams, Biz Stone, Jack Dorsey (Twitter), Michael Arrington (TechCrunch) , Kevin Rose (Digg) | 71017 |
| 6. | TV Personality, Female | Ellen DeGeneres, Martha Stewart, Brooke Burke, Felicia Day, Veronica Belmont | 69910 |
| 7. | TV Personality, Female, Age[50-60] | Ellen DeGeneres, Oprah Winfrey | 286545 |
| 8. | Music Artist, New York | Diddy, 50 Cent, Mariah Carey | 90294 |
| 9. | Comedian, New York | Jimmy Fallon, Danny Masterson | 117300 |
| 10. | Male, Age[30-40] | Ashton Kutcher, Lance Armstrong, Evan Williams, Kevin Rose, Wil Wheaton, Michael Arrington, Biz Stone | 68429 |