Noname manuscript No.
(will be inserted by the editor)

An Expressive Framework and Efficient Algorithms for the Analysis of Collaborative Tagging

Mahashweta Das[‡] · **Saravanan Thirumuruganathan**[‡] · **Sihem Amer-Yahia**[◊] · **Gautam Das**[‡,†] · **Cong Yu**[ℵ]

Received: date / Accepted: date

Abstract The rise of Web 2.0 is signaled by sites such as Flickr, del.icio.us, and YouTube, and social tagging is essential to their success. A typical tagging action involves three components, *user*, *item* (e.g., photos in Flickr), and *tag*s (i.e., words or phrases). Analyzing how tags are assigned by certain users to certain items has important implications in helping users search for desired information. In this paper, we develop *a dual mining framework* to explore tagging behavior. This framework is centered around two opposing measures, *similarity* and *diversity*, applied to one or more tagging components, and therefore enables a wide range of analysis scenarios such as characterizing similar users tagging diverse items with similar tags, or diverse users tagging similar items with diverse tags. By adopting different concrete measures for similarity and diversity in the framework, we show that a wide range of concrete analysis problems can be defined and they are NP-Complete in general. We design four sets of efficient algorithms for solving many of those problems and demonstrate, through comprehensive experiments over real data, that our algorithms significantly out-perform the exact brute-force approach without compromising analysis result quality.

Keywords collaborative tagging · dual mining framework · optimization · algorithm

1 Introduction

Tagging is a core activity on the social web. It reflects a wide range of content interpretations and serves many purposes, ranging from bookmarking websites in del.icio.us, organizing personal videos in YouTube, and characterizing movies in MovieLens. While one can possibly examine tags used by a single user on a single item, it is easy to see that the task becomes quickly intractable for a collection of tagging actions involving multiple users and items. In this paper, we aim to formalize the analysis of the tagging behavior of a set of users for a set of items and develop appropriate algorithms to complete that task.

A typical tagging action involves three components (i.e., dimensions), *user*, *item*, and *tag*. We propose to study a variety of analysis tasks that involve applying two alternative measures, *similarity* and *diversity*, to those components and producing groups of similar or diverse items, tagged by groups of similar or diverse users with similar or diverse tags. For example, one possible analysis outcome could be: *teenagers use diverse tags for action movies* or *males from New York and California use similar tags for movies directed by Quentin Tarantino.* In Section 1.1, 2.3, and 6.1, we will describe some of these problem instances that are enabled in our framework. A general dual mining framework that encompasses many common analysis tasks is defined in Section 2.4, and an extension to the framework is defined in Section 6.

A core challenge in this dual mining framework is the design of similarity and diversity measures. For user or item components, defined by (attribute, value) pairs, several existing comparison techniques have been proposed that can leverage their structured nature or bipartite connections. Section 2.1 illustrates some of those techniques.

Comparing similarity and diversity of tags used by various users on different items, however, presents a new challenge. First, tags are drawn from a much larger vocabulary than user or item attributes and exhibit a long tail characteristic. Second, it is often the case that different tags are used for the same set of items and, accounting for those tags separately would not capture their co-usage. Finally, tags may have linguistic connections such as synonymy. In order to capture tag similarity and diversity, we propose to *summarize* tags first to account for their co-usage and semantic re-

[‡]University of Texas at Arlington; [†]QCRI, Qatar; [◊]CNRS, LIG, France; [ℵ]Google Research

Fig. 1 Tag Signature for All Users

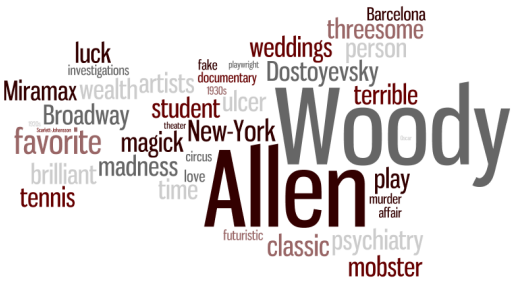

Fig. 2 Tag Signature for CA Users

lationships. Section 2.2 describes some techniques that we borrow from Information Retrieval and Machine Learning that can be used.

The tag component is also the most interesting among the three to be analyzed. Figure 1 shows a rendering of a tag summarization for *Woody Allen* movies in the form of a tag cloud. Similarly, Figure 2 shows a summarization of tags for the same movies from California users only. In both cases, summarization is defined as a simple frequency-based tag cloud where the size of a tag corresponds to how often it has been used on those movies. While *Woody* and *Allen* are not surprisingly common to both, the two clouds are different: all users highlight the *dramatic*, *tragic* and *disturbing* nature of those movies, and California users emphasize tags such as *classic* and *psychiatry*. Moreover, one of the director's popular movies, *Noiva Nervosa* is prominent in the tag cloud of all users, and yet is conspicuously absent in that of California users. Our goal is to define analysis tasks that can help users easily spot those interesting patterns and use that knowledge in subsequent actions.

We emphasize that, in this study, it is *not* our goal to advocate one particular similarity or diversity measure over another. Rather, we focus on formalizing the **Tagging Behavior Dual Mining (TagDM)** framework and the problem definitions, and designing algorithms that will work well for most measures. The analysis problems formally defined in our proposed framework fall into the wider category of constrained optimization problems. We are looking for groups of tagging actions that achieve maximum similarity or diversity on one or more components while satisfying a set of conditions and constraints. We first discuss a set of mining tasks that our TagDM framework can handle, and then move on to formalizing the general TagDM framework.

1.1 Problem Instances

Given that a typical tagging action involves three components (i.e., users, items, and tags), a large number of concrete problem instances can be defined, with their variations coming from two main aspects. The first category of variations depends on which measure (similarity or diversity) is applied to which tagging components (users, items, or tags). For example, a user can be interested in identifying *similar* user groups who use *similar* tags for *diverse* item groups, or in identifying *diverse* user groups who use *similar* tags for *similar* item groups. Since there are three components, each of which can adopt one of two measures, this variation alone can lead to $2^3 = 8$ different problem instances.

Since we are looking for result groups that achieve maximum similarity or diversity on one or more components while satisfying a set of conditions and constraints, the second category of variations depends on which components the user is adding to the optimization goal and which components the user is adding to the constraints. For example, a user can be interested in finding tagging action groups that maximize a *tag diversity* measure while satisfying *user diversity and item similarity* constraints, or groups that maximize a combination of *tag diversity and user diversity* measures while satisfying an *item similarity* constraint. Figures 1 and 2 depict an example of the former instance - *diverse users (all users and California users) maximizing tag diversity for similar items (Woody Allen movies)*. Since each component can be part of the optimization goal, or part of the constraint, or neither, this variation can lead to $3^3 - 1 = 26$ different problem instances. Combining both categories of variations, there is a total of **112** concrete problem instances that our framework captures! We formalize such a TagDM framework in Section 2. Of course, we can extend the TagDM framework to add additional conditions to the optimization goal(s) of these 112 instances based on user needs, the details of which are described in Section 6.

Table 1 illustrates six of the problem instantiations that we have investigated in details through the rest of the paper. In particular, we focus on problems with all three components with constraints on user and item and optimization on the tag component, since those are the most novel and intuitive mining problems. Under this setting, the user and item components are used primarily for enforcing constraints (similarity/diversity) and in providing an intuitive description of the tagging action groups. However, the optimization would be done over the tagging component. Two

tagging action groups would be considered similar if their tagging components have small distance between them.

Table 1 Concrete *TagDM* Problem Instantiations. Column *C* lists the constraint dimensions and column *O* lists the optimization dimensions.

ID	User	Item	Tag	*C*	*O*
1	similarity	similarity	similarity	U,I	T
2	similarity	diversity	similarity	U,I	T
3	diversity	similarity	similarity	U,I	T
4	diversity	similarity	diversity	U,I	T
5	similarity	diversity	diversity	U,I	T
6	similarity	similarity	diversity	U,I	T

1.2 Algorithms

Not surprisingly, as our complexity analysis shows in Section 3, those problems are NP-Complete in general. We propose four sets of efficient algorithms for solving them, the first three of which consider pair-wise aggregation measures for capturing similarity and diversity while the fourth employs a more general mining measure. The first set incorporates Locality Sensitive Hashing (LSH) and can be used for problems maximizing tagging action component similarity. While traditional LSH is frequently used for performing nearest neighbor search in high-dimensional spaces, our algorithm finds the bucket containing the result set of our tagging behavior analysis. The second set of algorithms borrows ideas from techniques employed in Computational Geometry to handle the Facility Dispersion Problem (FDP) and is effective for problems maximizing diversity. Both sets of algorithms possess compelling theoretical characteristics for problem instances optimizing the dual mining goal without any constraints. For both sets, we also propose advanced techniques that return better quality results in comparable running time. The third set of algorithms use Hierarchical Agglomerative Clustering (HAC) techniques and can be applied to problems maximizing either similarity or diversity. It is particularly useful for handling complex mining problems, having additional conditions in optimization goals. All three sets of algorithms consider some form of pair-wise distance measure for computing similarity or diversity. Therefore, we propose a fourth set of more general algorithm that is based on the Hill Climbing (HC) technique and can handle any arbitrary mining function for similarity and diversity.

1.3 Contributions

In this paper, we make the following main contributions:

- We formalize the task of analyzing the tagging behavior of a set of users for a set of items and propose a novel general constrained optimization framework for tagging behavior mining.
- We show that the tagging analysis problems are NP-Complete and propose efficient algorithms for solving the problems.
- We develop locality sensitive hashing based algorithms for solving problems maximizing tagging action component similarity. We design computational geometry based algorithms for problem instances maximizing diversity. We provide theoretical guarantees for both sets of algorithms for handling problems optimizing the dual mining goal without any constraints. We also propose hierarchical agglomerative clustering based and hill climbing based algorithms that apply to both similarity and diversity maximization problems, and are especially useful in handling complex mining tasks.
- We perform detailed experiments on real data to show that our proposed algorithms generate equally good results as exact brute-force in much less execution time.

2 The TagDM Framework

We model the data on a social tagging site as a triple $\langle U, I, T \rangle$, representing the set of users, the set of items and the tag vocabulary, respectively. Each tagging action can be considered as a triple itself, represented as $\langle u, i, \mathtt{T} \rangle$, where $u \in U$, $i \in I$, $\mathtt{T} \subset T$, respectively. A group of tagging actions is denoted as $g = \{\langle u_1, i_1, \mathtt{T}_1 \rangle, \langle u_2, i_2, \mathtt{T}_2 \rangle, \ldots, \}$. We define a user schema, $S_U = \langle a_1, a_2, \ldots \rangle$, to represent each user as a set of attribute values conforming to the user schema: $u = \langle u.a_1, u.a_2, \ldots \rangle$, where each $u.a_x$ is a value for the attribute $a_x \in S_U$. For example, let $S_U = \langle \mathtt{age}, \mathtt{gender}, \mathtt{state}, \mathtt{city} \rangle$, a user can be represented as $\langle \mathtt{18}, \mathtt{student}, \mathtt{new\ york}, \mathtt{nyc} \rangle$. Similarly, we define an item schema, $S_I = \langle a_1, a_2, \ldots \rangle$, to represent each item as a set of attribute values, $i = \langle i.a_1, i.a_2, \ldots \rangle$, where each $i.a_y$ is a value for the attribute $a_y \in S_I$.

Each tagging action therefore can be represented as an expanded tuple that concatenates the user attributes, the item attributes and the tags: $r = \langle r_u.a_1, r_u.a_2, \ldots, r_i.a_1, r_i.a_2, \ldots, \mathtt{T} \rangle$. G denotes the set of all such tagging action tuples. Many social sites have hundreds of millions of such tuples. Most, if not all, mining tasks involve analyzing sets of such tuples collectively. While there are a number of different ways tagging action tuples can be grouped, we adopt the view proposed and experimentally verified in [9], where groups of users (or items) that are *structurally describable* (i.e., sharing common attribute value pairs) are meaningful to end-users. Such groups correspond to conjunctive predicates on user or item attributes. An example of a *user describable tagging action group* is $\{\mathtt{gender=male}, \mathtt{state=new\ york}\}$, and of an *item describable group* is $\{\mathtt{genre=comedy}, \mathtt{director=woody\ allen}\}$. Next we define an essential characteristic of a set of tagging action groups.

Definition 1 Describable User Group. *A group of tagging actions, $g = \{\langle u_1, i_1, \mathtt{T}_1\rangle, \langle u_2, i_2, \mathtt{T}_2\rangle, \ldots, \}$, are considered a* meaningful user group *iff: $\exists A \subset S_U, |A| > 0$, such that, for each $a \in A$, $\exists v, \forall r \in g, r_u.a = v$. A set of attribute value pairs, $D^g_{user} = \{a_1{=}v_1, a_2{=}v_2, \ldots, \}$, where $a_x \in A$ and v_x is the value of a_x shared by all tuples in g, are called the* user group description. *Furthermore, we say that tuples in g* satisfy D^g_{user}.

Definition 2 Describable Item Group. *A group of tagging actions, $g = \{\langle u_1, i_1, \mathtt{T}_1\rangle, \langle u_2, i_2, \mathtt{T}_2\rangle, \ldots, \}$, are considered a* meaningful item group *iff: $\exists A \subset S_I, |A| > 0$, such that, for each $a \in A$, $\exists v, \forall r \in g, r_i.a = v$. We define* item group description *as $D^g_{item} = \{a_1{=}v_1, a_2{=}v_2, \ldots, \}$, where $a_x \in A$ and v_x is the value of a_x shared by all tuples in g. We say that tuples in g* satisfy D^g_{item}.

Definition 3 Group Support. *Given the input set of tagging action tuples G, the* support *of a set of tagging action groups $\mathscr{G} = \{g_1, g_2, \ldots\}$ over G, is defined as $Support^{\mathscr{G}}_G = |\{r \in G \mid \exists g_x \in \mathscr{G}, r \in g_x\}|$.* Intuitively, group support measures the number of input tagging action tuples that belongs to at least one of the groups in $\mathscr{G}$.

Before we formalize the mining problems, we introduce the core concept of Mining Function that computes a similarity or diversity score using arbitrary evaluations over the tagging action groups. Similarity and diversity are usually estimated as functions of distance. In particular, aggregation of pair-wise distances between the different objects (i.e., tagging action groups) offers powerful means for solving many real mining scenarios:

Definition 4 Pair-Wise Aggregation Dual Mining Function. *A Pair-Wise Aggregation (PA) Dual Mining Function, $F_{pa} : \mathscr{G} \times b \times m \rightarrow \mathtt{float}$, takes as inputs: $\mathscr{G}$, a set of tagging action groups; $b \in \{\mathtt{users}, \mathtt{items}, \mathtt{tags}\}$, a tagging behavior dimension; $m \in \{\mathtt{similarity}, \mathtt{diversity}\}$, a dual mining criterion. It has two component function $F_p : g_i \times g_j \times b \times m \rightarrow \mathtt{float}$ and $F_a : \{s_1, s_2, \ldots\} \rightarrow \mathtt{float}$, where (g_i, g_j) is a pair of distinct tagging action groups and each s_i is an intermediate score produced by F_p, such that: $F_{pa}(\mathscr{G}, b, m) = F_a(\{F_p(g_i, g_j, b, m)\}, \forall g_i, g_j \in \mathscr{G}, i \neq j$.*

We now present a few examples of the pair-wise dual mining function. The key to a pair-wise dual mining function is the pair-wise comparison function, $F_p(g_1, g_2, b, m)$, where g_1 and g_2 are distinct tagging action groups, and $b \in \{\mathtt{users}, \mathtt{items}, \mathtt{tags}\}$, is a tagging behavior dimension, and $m \in \{\mathtt{similarity}, \mathtt{diversity}\}$, is a dual mining criterion.

2.1 User & Item Dimensions Dual Mining

Given a user describable tagging action group[1], its user dimension is effectively its user group description, i.e., a set of (attribute, value) pairs that describes the group. Therefore, given two user groups, g_1 and g_2, their similarity or diversity can be captured mainly in two ways: 1) structural distance between the user group descriptions and 2) set distance based on the items they have rated.

Let A be the set of user attributes shared between two user describable tagging action groups g_1 and g_2, an example of the pair-wise comparison function leveraging **structural distance** is the following:

$F_p(g_1, g_2, \mathtt{users}, \mathtt{similarity}) = \sum_{a \in A} sim(v_1, v_2)$

where $a.v_1$ and $a.v_2$ belong to the set of user attribute value pairs and *sim* can be a string similarity function that simply computes the edit distance between two values or a more sophisticated similarity function that takes domain knowledge into consideration. For example, a domain-aware similarity function can determine *New York City* to be more similar to *Boston* than to *Dallas*. $F_p(g_1, g_2, \mathtt{users}, \mathtt{diversity})$ can be similarly defined using the inverse function.

Let $g_1.I$ and $g_2.I$ be the sets of items tagged by tuples in g_1 and g_2, respectively, an example of the pair-wise comparison function leveraging **set distance** is the following:

$F'_p(g_1, g_2, \mathtt{users}, \mathtt{similarity}) = \frac{|\{r | r \in g_1.I \wedge r \in g_2.I\}|}{|\{r | r \in g_1.I \vee r \in g_2.I\}|}$

which simply computes the percentages of items tagged by both groups (akin to Jaccard distance.) If numerical ratings are available for each tagging tuple, a more sophisticated set distance similarity function can further impose an additional constraint that an item is common to both groups if its average ratings in both are close. $F'_p(g_1, g_2, \mathtt{users}, \mathtt{diversity})$ can be similarly defined using the inverse function.

2.2 Tag Dimension Dual Mining

The tag dimension is fundamentally different from the user and item dimensions. First, there is no fixed set of attributes associated with the tag dimension, therefore the structural distance does not apply. Second, tags are chosen freely by users using diverse vocabularies. As a result, a single tagging action group can contain a large number of tags. Both characteristics make comparing two sets of tags difficult.

We propose a two-step approach for handling the tag dimension. First, we propose an initial step to summarize the set of all tags of a tagging action group into a smaller representative set of tags, called *group tag signature*. Second, we apply comparison functions to compute distance between signatures. Once again, we are not advocating any particular way of producing signatures and/or comparing them.

[1] Since the user and item dimensions share the same characteristics in the dual mining framework, we present here only the user dimension for simplicity.

Rather, we simply argue for the need for tag signatures and their comparisons.

Group tag signature generation: Given a group of tagging actions $g = \{\langle u_1, i_1, \mathtt{T}_1\rangle, \langle u_2, i_2, \mathtt{T}_2\rangle, \ldots\}$, we aim to summarize the tags in $T_1 \cup T_2 \cup \ldots$ into a tag signature $T_{rep}(g)$. The general form of $T_{rep}(g)$ is $\{(tc_1, w_1), (tc_2, w_2), \ldots\}$ where tc_i is topic category (can be a tag itself) and w_i is weight, i.e., relevance of g for c_i.

One can define several methods to compute tag signatures. For example, when tags are hand-picked by editors and hence the number of unique tags is small, a simple definition can be $T_{rep}(g) = \{(t, \mathtt{freq}(t)) \mid t \in \mathtt{T}_1 \cup \mathtt{T}_2 \cup \ldots\}$, where $\mathtt{freq}(t)$ computes how many times t is used in g.

Most collaborative tagging sites encourage users to create their own tags, thereby creating a long tail of tags. This raises challenges such as sparsity and the choice of different tags to express similar meanings. Techniques from Information Retrieval and Machine Learning such as tf*idf and Latent Dirichlet Allocation (LDA) can be used for tag summarization. LDA aggregates tags into topics based on their co-occurrence and reason at the level of topics, and handles long tail issues [2]. Also, a Web service such as Open Calais[2] can be used to match a set of tags to a set of predefined categories through sophisticated language analysis and information extraction.

Comparing group tag signatures: When tagging action groups are represented as tag signatures over the same set of topics, we can leverage many existing vector comparison functions to compute the distance between any two group tag signature vectors pair-wisely. An example is simply **cosine similarity** as follows:
$F''_P(g_1, g_2, \mathtt{tags}, \mathtt{similarity}) = \mathtt{cos}(\theta(T_{rep}(g_1), T_{rep}(g_2)))$, where θ is the angle between the two vectors. $F''_P(g_1, g_2, \mathtt{tags}, \mathtt{diversity})$ can be defined similarly

The comparison can also be enhanced by using an ontology such as WordNet to compare entries of similar topics.

2.3 Concrete Problem Instances

We are now ready to formally define two of the concrete dual mining problems listed in Table 1 in the introduction. The first one (Problem 2 in Table 1) aims to find similar user sub-populations who agree most on their tagging behavior for a diverse set of items. The second one (Problem 4 in Table 1) aims to find diverse user sub-populations who disagree most on their tagging behavior for a similar set of items.

Problem 2 *Identify a set of tagging action groups, $G^{opt} = \{g_1, g_2, \ldots\}$, such that:*

- $\forall g_x \in G^{opt}$, *g_x is user- and/or item-describable;*
- $1 \leq |G^{opt}| \leq k$;
- $Support_G^{G^{opt}} \geq p$;
- $F_1(G^{opt}, \mathtt{users}, \mathtt{similarity}) \geq q$;
- $F_2(G^{opt}, \mathtt{items}, \mathtt{diversity}) \geq r$;
- $F_3(G^{opt}, \mathtt{tags}, \mathtt{similarity})$ *is maximized.*

where F_1 and F_2 are structural similarity based dual mining functions as defined in Definition described in Section 2.1, and F_3 is the LDA based tag dual mining function as described in Section 2.2.

For $k = 2$, $p = 100$, $q = 0.5$, and $r = 0.5$, solving the problem on the full set of tagging action tuples in MovieLens [8] can give us the following G^{opt}:

$g_1 = \{\langle \mathtt{gender}, \mathtt{male}\rangle, \langle \mathtt{age}, \mathtt{young}\rangle, \langle \mathtt{actor}, \mathtt{j.aniston}\rangle, (\mathtt{comedy}, \mathtt{drama}, \mathtt{friendship})\}$
$g_2 = \{\langle \mathtt{gender}, \mathtt{male}\rangle, \langle \mathtt{age}, \mathtt{young}\rangle, \langle \mathtt{actor}, \mathtt{j.timberlake}\rangle, (\mathtt{drama}, \mathtt{friendship})\}$

which illustrates the interesting pattern that male young users assign similar tags, *drama* and *friendship*, to movies with *Jennifer Aniston* and *Justin Timberlake*, the former for her involvement in the popular TV show *Friends* and the latter for his movie *The Social Network*.

A closely related problem to Problem 2 is to inverse the similarity and diversity constraints for the user and item components, i.e., finding diverse user sub-populations who agree most on their tagging behavior for a similar set of items (Problem 3 in Table 1 in the introduction). Both problems focus on optimizing the tag similarity and therefore can be solved using similar techniques. Next, we define a problem that aims to identify groups that disagree on their tagging behavior.

Problem 5 *Identify a set of tagging action groups, $G^{opt} = \{g_1, g_2, \ldots\}$, such that:*

- $\forall g_x \in G^{opt}$, *g_x is user- and/or item-describable;*
- $1 \leq |G^{opt}| \leq k$;
- $Support_G^{G^{opt}} \geq p$;
- $F_1(G^{opt}, \mathtt{users}, \mathtt{diversity}) \geq q$;
- $F_2(G^{opt}, \mathtt{items}, \mathtt{similarity}) \geq r$;
- $F_3(G^{opt}, \mathtt{tags}, \mathtt{diversity})$ *is maximized.*

where F_1, F_2, and F_3 are similarly defined as in Problem 2.

For $k = 2$, $p = 100$, $q = 0.5$, and $r = 0.5$, solving the problem on the full set of tagging action tuples in MovieLens can give us the following G^{opt}:

$g_1 = \{\langle \mathtt{gender}, \mathtt{male}\rangle, \langle \mathtt{age}, \mathtt{teen}\rangle, \langle \mathtt{genre}, \mathtt{action}\rangle, (\mathtt{gun}, \mathtt{special\ effects})\}$
$g_2 = \{\langle \mathtt{gender}, \mathtt{female}\rangle, \langle \mathtt{age}, \mathtt{teen}\rangle, \langle \mathtt{genre}, \mathtt{action}\rangle, (\mathtt{violence}, \mathtt{gory})\}$

which illustrates teenaged male users and female users have entirely different perspectives on action movies. This gives a user a new insight that there is something about action movies that is causing the different reactions among two different groups of users.

[2] *https://www.opencalais.com/*

2.4 The TagDM Framework

The formal definitions for Problems 1 and 4 share a number of similarities. The objective is to identify a set of tagging action groups that maximizes similarity/diversity over a specific dimension. In addition, it also has constraints on the number of results, their coverage and whether similarity/diversity measures over individual dimensions exceed certain threshold.

In this paper, we describe a constrained optimization based framework for tagging behavior mining. The framework is very general such that each of the concrete problem instances previously described can naturally be defined. Additionally, it is also easily extensible to explore complex objective functions and constraints depending on user needs.

We formally define the TagDM framework in the following definition.

Definition 5 Tagging Behavior Dual Mining (TagDM) Problem. *Given a triple $\langle G,C,O\rangle$ in the TagDM framework where G is the input set of tagging actions and C, O are the sets of constraints and optimization criteria respectively, the Tagging Behavior Dual Mining problem is to identify a set of tagging action groups, $G^{opt} = \{g_1, g_2, \ldots\}$ for $b \in \{\texttt{users}, \texttt{items}, \texttt{tags}\}$ and $m \in \{\texttt{similarity}, \texttt{diversity}\}$, such that:*

- $\forall g_x \in G^{opt}$, *g_x is user- and/or item-describable;*
- $k_{lo} \leq |G^{opt}| \leq k_{hi}$;
- $Support_G^{G^{opt}} \geq p$;
- $\forall c_i \in C, c_i.F(G^{opt}, b, m) \geq threshold$;
- $\Sigma_{o_j \in O}, o_j.F(G^{opt}, b, m)$ *is maximized.*

Intuitively, TagDM aims to identify a set of user- and/or item-describable sub-groups from input tagging actions, such that the dual mining constraints are satisfied and a dual mining goal is optimized. We now clearly see how this framework generalizes the common problem instances given in Section 2.3. The notation $c_i.F$ refers to a function (associated with constraint c_i) that measures similarity/diversity of the corresponding dimension. Similarly, the notation $o_j.F$ represents the dual mining function that operates over the specific dimension whose value must be optimized.

Notice that, the definition of TagDM problem is not limited to pair-wise aggregation dual mining functions, described in Definition 4. Pair-wise dual mining functions has a number of appealing properties - they are very common, can be computed efficiently, and offers a vast literature of techniques to exploit for developing fast solutions. However, restricting to pair-wise functions severely limits the expressiveness and generality of our TagDM framework.

General dual mining functions takes as input a set of tagging action groups and performs a holistic analysis over the entire set. They are not restricted to identifying and then accumulating the local properties at the tagging action group level. A number of optimization functions cannot be easily be defined in terms of pair-wise aggregation functions. Here are some examples:

- Identify a set of tagging action groups such that the variance of individual tagging actions is minimized. This helps us find user-describable and/or item-describable groups who share similar tagging behavior.
- Identify a set of tagging action groups such that the number of distinct tags they use is minimized. This helps us discover user-describable and/or item-describable groups where there is an universal agreement on the tag usage.
- Identify a set of tagging action groups such that their tag vocabulary (obtaining by performing union of all tagging actions) form a coherent ontology - either they all are synonyms, antonyms or occur within a specific distance of each other in an Ontology service such as Wordnet.

The generalized dual mining function can be formally defined as :

Definition 6 Dual Mining Function. *A Dual Mining Function, $F : \mathscr{G} \times b \times m \rightarrow \texttt{float}$, takes as inputs: $\mathscr{G}$, a set of tagging action groups; $b \in \{\texttt{users}, \texttt{items}, \texttt{tags}\}$, a tagging behavior dimension; $m \in \{\texttt{similarity}, \texttt{diversity}\}$, a dual mining criterion; and produces a float score, s, that quantifies the mining criterion over the particular dimension for the set of tagging action groups.*

3 Complexity Analysis

In this section we provide the proof that the Tagging Behavior Dual Mining problem is NP-Complete. The decision version of the TagDM problem is defined as follows:

Given a triple $\langle G,C,O\rangle$, is there a set of tagging action groups $G^{opt} = \{g_1, g_2, \ldots\}$ such that $\sum_{o_j \in O}(o_j.Wt \times oj.F(G_{opt}, o_j.D, o_j.M) \geq \alpha$ subject to:

- $\forall g_x \in G^{opt}, g_x$ is user- and/or item-describable.
- $k_{lo} \leq |G^{opt}| \leq k_{hi}$
- $Support_G^{G^{opt}} \geq p$
- $\forall c_i \in C, c_i.F(G^{opt}, c_i.D, c_i.M) \geq c_i.Th$

Theorem 1 *The decision version of the TagDM problem is NP-Complete.*

Proof: The membership of decision version of TagDM problem in NP is obvious. To verify NP-Completeness, we reduce Complete Bipartite Subgraph problem (CBS) to our problem and argue that a solution to CBS exists, *if and only if*, a solution our instance of TagDM exists. First, we show that the problem CBS is NP-Complete.

Lemma 1 *Complete bipartite subgraph problem (CBS) is NP-Complete.*

Proof: The decision version of CBS is defined as follows:

Given a bipartite graph $G' = (V_1, V_2, E)$ and two positive integers $n_1 \leq |V_1|, n_2 \leq |V_2|$, are there two disjoint subsets $V_1^{'} \subseteq V_1, V_2^{'} \subseteq V_2$ such that $|V_1^{'}| = n_1, |V_2^{'}| = n_2$ and $u \in V_1^{'}, v \in V_2^{'}$ implies that $\{u,v\} \in E$.

The membership of CBS in NP is obvious. We verify the NP-Completeness of the problem by reducing it to Balanced Complete Bipartite Subgraph (BCBS) problem which is defined as : Given a bipartite graph $G'' = (V_1^{''}, V_2^{''}, E^{'})$ and a positive integer $n^{'}$, find two disjoint subsets $V_1^{'''} \subseteq V_1^{''}, V_2^{'''} \subseteq V_2^{''}$ such that $|V_1^{'''}| = |V_2^{'''}| = n^{'}$ and $u \in V_1^{'''}, v \in V_2^{'''}$ implies that $\{u,v\} \in E^{'}$. This problem was proved to be NP-Complete by reduction from Clique in [26]. We can reduce BCBS to CBS by passing the input graph $G''(V_1^{''}, V_2^{''}, E^{'})$ of BCBS to CBS and setting n_1 and n_2 to $n^{'}$. If a solution exists for the CBS instances, then the disjoint subsets $V_1^{'''}, V_2^{'''}$ form a balanced complete bipartite subgraph in G''. □

We have already established that TagDM problem is in NP. To verify its NP-Completeness, we reduce CBS to the decision version of our problem. Given an instance of the problem CBS with $G^{'} = (V_1, V_2, E)$ and positive integers n_1, n_2, we construct an instance of TagDM problem such that there exists a complete bipartite subgraph induced by disjoint vertex subsets $V_1^{'} \subseteq V_1, V_2^{'} \subseteq V_2$ and $|V_1^{'}| = n_1, |V_2^{'}| = n_2$, *if and only if*, a solution to our instance of TagDM exists.

First, we create an user schema $S_U = \langle a_1, a_2, \ldots, a_{|V_2|} \rangle$ such that for each vertex $v_j \in V_2$, there exists a corresponding user attribute $a_j \in S_U$. Next, we define a set of users $U = \{u_1, u_2, \ldots, u_{|V_1|}\}$. Again, for each vertex $v_i \in V_1$ there exists a corresponding user $u_i \in U$.

For all pairs of vertices $(v_i, v_j), v_i \in V_1, v_j \in V_2$, we set $u_i.a_j$ to 1 if $\{v_i, v_j\} \in E$; else, we set it to a unique value such that $u_{x1}.a_{y1} \neq u_{x2}.a_{y2}$ unless $x_1 = x_2, y_1 = y_2$. Intuitively, we set the j-th attribute of i-th user to 1 if an edge exists between vertex pairs (v_i, v_j); else, we set it to a unique value that is not shared with any attribute of any user. One possible way to assign the unique attribute values is to pick a previously unassigned value from the set $[2, |V_1| \times |V_2| + 1]$. Since the number of possible edges is at most $|V_1| \times |V_2|$, this set suffices to generate unique attribute values.

We construct an instance of the TagDM problem where $I = \{i\}$ and $T = \{t\}$. This results in a set of tagging actions, $G = \{\langle u_1, i, t \rangle, \ldots, \langle u_{|V_1|}, i, t \rangle\}$ where only the user dimension plays a non-trivial role in determining the problem solution. Given a pair of users, the pairwise similarity function F_1 on user dimension measures their structural similarity by counting the number of attribute values that are shared between them. Intuitively, the problem collapses to that of finding a subset of users who share a subset of attributes.

We then define our TagDM problem instance as : For a given a triple $\langle G, C, O \rangle$, identify a set G^{opt} of tagging action groups such that $F_3(G^{opt}, tags, m) \geq 0$ subject to:

- $1 \leq |G^{opt}| \leq n_1$
- $Support_G^{G^{opt}} \geq n_1$
- $F_1(G^{opt}, \texttt{users}, \texttt{similarity}) \geq n_2 \times \binom{n_1}{2}$

If there exists a solution to this TagDM problem instance, then there are n_1 users who have identical values for at least n_2 of their attributes. If two users u_x and u_y have same values for a set of attributes A, then for all attributes $a \in A$, $u_x.a = u_y.a = 1$. In other words, whenever the attributes of two users overlap, the shared attributes can only take a value of 1. Any other symbol that was assigned is unique and cannot overlap by construction. If there exists a subset of attributes $A' \subseteq S_U$ and a subset of users $U' \subseteq U$, then the corresponding vertices in V_1 and V_2 form a complete bipartite subgraph solving the input instance of BCS. Thus TagDM problem is NP-Complete. □

3.1 Exact Algorithm

A brute-force exhaustive approach (henceforth, referred to as **Exact**) to solve the TagDM problem requires us to enumerate all possible combinations of tagging action groups in order to return the optimal set of groups maximizing the mining criterion and satisfying the constraints. The number of possible candidate sets is exponential in the number of groups. Evaluating the constraints on each of the candidate sets and selecting the optimal result can thus be prohibitively expensive. Each tagging action group is associated with a group tag signature vector (the size of which is determined by the cardinality of the global set of tag topics), which may introduce additional challenges in the form of higher dimensionality. Therefore, we develop practical efficient solutions.

We develop two sets of algorithms that are capable of solving all **112** concrete problem instances that TagDM framework captures. The first set comprises of locality sensitive hashing based algorithms for handling TagDM problem instances maximizing similarity of tagging action components. The algorithms are efficient in practice, but cannot handle TagDM problem instances maximizing diversity. The second set is based on techniques employed in computational geometry for the facility dispersion problem and is our solution for diversity mining problem instances.

Next, we discuss ways to extend the general TagDM framework, i.e., Definition 5, and further develop two additional algorithms which are capable of handling complex mining tasks. The first of these algorithms draws inspiration from hierarchical clustering methods and handles problem instances maximizing similarity as well as those maximizing diversity. The technique is particularly useful for handling mining tasks which involve additional conditions and criteria in the optimization goal of the general TagDM framework. The second of these algorithms consider the more general mining function in Definition 6 as opposed to the pairwise measures used by all the remaining algorithms.

4 LSH Based Algorithms

The first of our algorithmic solutions is based on locality sensitive hashing (**LSH**) which is a popular technique to solve nearest neighbor search problems in higher dimensions [24]. LSH is preferred over several seemingly promising techniques (such as constructing efficient indices, which suffers from the curse of dimensionality) because it scales gracefully to higher dimensional data, is efficient and provides theoretical guarantees.

LSH performs a probabilistic dimensionality reduction of high dimensional data by projecting input items in higher dimension to a lower dimension such that items that were in close proximity in the higher dimension retain their proximity in lower dimensional space with high probability. LSH *hashes* an input item such that similar input items fall into the same bucket (i.e., uniquely definable hash signature) with high probability. In other words, all the input items in the same bucket are highly likely to be similar.

A classical application of LSH is to identify nearest neighbors efficiently. Typically, the input items in LSH are high dimensional vectors such as multimedia content. There are two major parameters for LSH : d', the number of hash functions (which also determines the lower dimension to which items are projected to) and l, the number of hash tables. A hash table is associated with d' hash functions randomly chosen from a hash family that is problem specific. Each hash function accepts a vector in high dimension and returns a scalar. By concatenating the result of d' hash functions, we get a d' dimensional vector that also forms a distinct hash signature for the item. All the input items that have identical hash signatures are said to fall in the same bucket. This process is repeated for all the l hash tables. Intuitively, each hash table can be considered as a partition of the input items such that similar items fall into same partition with high probability. Typically, each hash table results in different partition of the input data based on the hash functions associated with it.

Once the input data has been projected to lower dimensions, it can then be used for applications such as identifying nearest neighbors of a given item. For each of the l hash tables, we project the input query item and identify which bucket it falls into. All the items that co-occur in any of the l buckets the input query items fall into are considered as candidate nearest neighbors. However, the set of candidates is usually much smaller than the size of input items and allows nearest neighbor(s) to be computed efficiently.

LSH guarantees a lower bound on the probability that two similar input items fall into the same bucket in the projected space and also the upper bound on the probability that two dissimilar vectors fall into the same bucket. For any pair of points in a high-dimensional space and a given hash function h, P_1 is the probability of two close points receiving the same value after hashing. P_2 is the probability of two far-apart points falling receiving the same value after hashing. We want $P_2 < P_1$ and typically $P_1 > \frac{1}{2}$. Formally, given a pair of nearby points p_1, q_1 (defined as points within distance R_1) and far-apart points p_2, q_2 (those with distance at least $R_2 = cR_1$) we have:

$$P[h(p_1) = h(q_1)] \geq P_1 \; for \; dist(p_1, q_1) \leq R_1$$
$$P[h(p_2) = h(q_2)] \leq P_2 \; for \; dist(p_2, q_2) \geq R_2 \quad (1)$$

Two items fall into the same bucket if they receive identical values for each of the d' independently chosen hash functions. After projecting the input items from higher dimension d to a lower dimension d', we can see that the following probabilistic bounds hold for similar/dissimilar items falling into same bucket :

$$P(similar \; items \; falling \; in \; same \; bucket) \; \geq P_1^{d'}$$
$$P(dissimilar \; items \; falling \; in \; same \; bucket) \leq P_2^{d'} \quad (2)$$

Given the background on LSH, we now adapt it to select a set of tagging action groups that are similar in their tagging behavior based on locality sensitive hashing. Recall that our input is the set G of n tagging action groups (i.e., n d-dimensional tag signature vectors, where d is the cardinality of the global set of tag topics mentioned in Section 2.2) using a pair-wise comparison function $F''_p(g_1, g_2, \texttt{tags}, \texttt{similarity})$ that operates on group tag signature vectors in order to optimize tag similarity. Our expected result is a set of k tagging action groups G^{opt} such that they have the least average pair-wise distance between them

Note that, our LSH based algorithms works for Problems 1, 2 and 3 in Table 1 maximizing tag similarity. We first introduce an algorithm that returns the set of tagging action groups G^{opt}, $1 \leq |G^{opt}| \leq k$ having maximum similarity in tagging behavior (Column O in Table 1) and then discuss additional techniques to include the multiple hard constraints into the solution (Column C in Table 1).

4.1 Maximizing Similarity based on LSH

Our LSH based algorithm **SM-LSH** deals with TagDM problem instances optimizing tag **S**i**M**ilarity. The classical usage of LSH is to find the nearest neighbor(s) for a given query item. However, in TagDM problems, there is no specific query item - instead, we want to identify a set of k tagging action groups that have high pair-wise similarity over their tagging behavior. We reconcile these two seemingly different usages by exploiting the fact that given a bucket, all points in it are highly likely to be similar. We employ LSH to partition the tagging action groups such that similar groups fall into the same bucket. The next step is to choose the best set of tagging groups for a hash table. We identify the best k-subset for each bucket and pick the best set over all buckets. Different sets are comparable using the dual mining scoring function. We repeat the process for each hash table and finally choose the best set among all hash tables.

One of the key requirement for good performance of LSH is the careful selection of the family of hashing functions. In SM-LSH, we use the LSH scheme proposed by Charikar [7] which employs a family of hashing functions based on cosine similarity. It is possible to utilize other distance measures such as Euclidean, Jaccard, Earth-Movers etc. The only change involves how the hash functions are chosen and how the probability values P_1 and P_2 are computed. As discussed in Section 2.2, the cosine similarity between two group tag signature vectors is defined as the cosine of the angle between them and can be defined as:

$$cos(\theta(T_{rep}(g_x), T_{rep}(g_y))) = \frac{|T_{rep}(g_x).T_{rep}(g_y)|}{\sqrt{|T_{rep}(g_x)|.|T_{rep}(g_x)|}}$$

The algorithm computes a succinct hash signature of the input set of n tagging action groups by computing d' independent dot products of each d-dimensional group tag signature vector $T_{rep}(g_x)$, where $g_x \subseteq G$ with a random unit vector $\mathbf{r}$ and retaining the sign of the d' resulting products. This maps a higher d-dimensional vector to a lower d'-dimensional vector ($d' \ll d$). Each entry of $\mathbf{r}$ is drawn from a 1-dimensional Normal distribution N(0,1) with zero mean and unit variance. Alternatively, we can generate a spherically symmetric random vector $\mathbf{r}$ of unit length from the d-dimensional space. The LSH function for cosine similarity for our problem is given by the following Theorem 2 adapted from [7]:

Theorem 2 *Given a collection of n d-dimensional vectors where each vector $T_{rep}(g_x)$ corresponds to a $g_x \subseteq G$, and a random unit vector $\mathbf{r}$ drawn from a 1-dimensional Normal distribution N(0,1), define the hash function h_r as:*

$$h_r(T_{rep}(g_x)) = \begin{cases} 1 \; if \; \mathbf{r}.T_{rep}(g_x) \geq 0 \\ 0 \; if \; \mathbf{r}.T_{rep}(g_x) < 0 \end{cases}$$

Then for two arbitrary vectors $T_{rep}(g_x)$ and $T_{rep}(g_y)$:

$$P[h_r(T_{rep}(g_x)) = h_r(T_{rep}(g_y))] = 1 - \frac{\theta(T_{rep}(g_x), T_{rep}(g_y))}{\pi}$$

where $\theta(T_{rep}(g_x), T_{rep}(g_y))$ is angle between two vectors.

The proof of the above Theorem 2 establishing that the probability of a random hyperplane (defined by $\mathbf{r}$ to hash input vectors) separating two vectors is directly proportional to the angle between the two vectors follows from Goemans et. al's theorem [13]. Any pair-wise dual mining function for comparing tag signatures must satisfy such properties. We represent the d'-dimensional-bit LSH function as:

$$g(T_{rep}(g_x)) = [h_{r1}(T_{rep}(g_x)), \ldots, h_{rd'}(T_{rep}(g_x))]^T$$

For d' LSH functions and from (2), the probability of similar tag signature vectors g_x and g_y falling into the same bucket for all d' hash functions is at least :

$$P(g(T_{rep}(g_x)) = g(T_{rep}(g_y))) \geq \left(1 - \frac{\theta(T_{rep}(g_x), T_{rep}(g_y))}{\pi}\right)^{d'}$$

Now, each input vector is entered into the l hash tables indexed by independently constructed hash functions $g_1(T_{rep}(g_x)), \ldots, g_l(T_{rep}(g_x))$. Using this LSH scheme, we hash the tag vectors to l different d'-dimensional hash signatures(or, buckets). The total number of possible hash signatures in each of the l lower dimensional space is $2^{d'}$. However, the maximum bound on the number of buckets in each of the lower dimensional space is n.

While LSH is generally used to find the nearest neighbors for new items, we take the novel approach of finding the right bucket to output as result of our problem based on checking for the number of tagging action groups in result set and ranking by scoring function. For each of the l hash tables, we first check for satisfiability of $1 \leq |G^{opt}| \leq k$ in each bucket and then rank the buckets based on the scoring function in order to determine the result set of tagging action groups G^{opt} with maximum similarity.

Theorem 3 *Given a collection of n d-dimensional tag signature vectors where each pair of vectors $T_{rep}(g_x)$ and $T_{rep}(g_y)$ corresponds to a $g_x, g_y \subseteq G$, the probability of finding result set G^{opt} of k most similar vectors by SM-LSH is bounded by:*

$$P(G^{opt}) \geq max\left(0, 1 - \sum_{g_x, g_y \in G^{opt}} \left[1 - \left(1 - \frac{\theta(T_{rep}(g_x), T_{rep}(g_y))}{\pi}\right)^{d'}\right]\right)$$

Proof: The probability of finding the set of tagging action groups G^{opt}, $1 \leq |G^{opt}| \leq k$ having maximum similarity in tagging behavior, $P(G^{opt})$:
= 1 - P(one of ${}^kC_2$ vector pair belongs to different buckets)
$\geq$ 1 - $\sum_{g_x, g_y \in G^{opt}}$ P($T_{rep}(g_x)$,$T_{rep}(g_y)$ in different buckets)
$\geq$ 1 - $\sum_{g_x, g_y \in G^{opt}}$ [1 - P($T_{rep}(g_x)$,$T_{rep}(g_y)$ in same buckets)]
$\geq$ 1 - $\sum_{g_x, g_y \in G^{opt}}$ $\left[1 - \left(1 - \frac{\theta(T_{rep}(g_x), T_{rep}(g_y))}{\pi}\right)^{d'}\right]$ □

Algorithm 1 is the pseudo code of our SM-LSH algorithm. This algorithm may return null result if post-processing of all l hash tables yields no bucket satisfying $1 \leq |G^{opt}| \leq k$. In other words, there are no set of tagging action groups in any bucket such that they satisfy the constraints (size of set, coverage, user/item overlap) of our problem instance. This motivates us to tune SM-LSH by *iterative relaxation* that varies the input parameter d' in each iteration. Decreasing the parameter d' increases the expected number of tagging action groups hashing into a bucket, thereby increasing the chances of our algorithm finding the result set. We start with an initial value of d' and reduce the parameter systematically (in a manner similar to performing a binary search) such that we find a value of d' such that it has some buckets that contain more than k tagging action groups satisfying the constraints. However, the expected size of bucket must not be too high as that will make the problem of finding the best k-subset of tagging action groups in a bucket very expensive. (please see Subsection 4.4 for additional discussion).

Algorithm 1 SM-LSH (G, O, k, d', l)**:** G^{opt}

//Main Algorithm
1: $min = 1$
2: $max = d'$
3: $T^U_{rep} \leftarrow \{\}$; $T^I_{rep} \leftarrow \{\}$
4: **if** $C_1.m =$ `similarity` **then**
5: $T^U_{rep} \leftarrow$ Unarize user vector
6: **end if**
7: **if** $C_2.m =$ `similarity` **then**
8: $T^I_{rep} \leftarrow$ Unarize item vector
9: **end if**
10: **for** x = 1 to n **do**
11: $T_{rep}(g_x) \leftarrow T^U_{rep}(g_x) + T^I_{rep}(g_x) + T_{rep}(g_x)$
12: **end for**
13: **repeat**
14: $Buckets \leftarrow$ LSH(G, d', l)
15: $G^{opt} \leftarrow$ MAX(Rank($Buckets$, k))
16: **if** $G^{opt} = null$ **then**
17: $max = d' - 1$
18: **else**
19: $min = d' + 1$
20: **end if**
21: $d' = (min + max)/2$
22: **until** ($min > max$) **or** ($G^{opt} \neq null$)
23: return G^{opt}

//LSH(G, d′, l): Buckets
1: **for** z = 1 to l **do**
2: **for** x = 1 to n **do**
3: **for** y = 1 to d' **do**
4: Randomly choose **r** from d-dimensional Normal distribution N(0, 1)
5: **if** $\mathbf{r}.T_{rep}(g_x) \geq 0$ **then**
6: $h_{ry}(T_{rep}(g_x)) \leftarrow 1$
7: **else**
8: $h_{ry}(T_{rep}(g_x)) \leftarrow 0$
9: **end if**
10: $g_z(T_{rep}(g_x)) = [h_{r1}(T_{rep}(g_x)), .., h_{rd'}(T_{rep}(g_x))]^T$
11: **end for**
12: **end for**
13: **end for**
14: $Buckets \leftarrow g_1(T_{rep}(g_x)) \cup \cdots \cup g_l(T_{rep}(g_x))$
15: **return** *Buckets*

Example 1 *Let us consider one of the Problems 1, 2, or 3 in Table 1 where the objective is to optimize tag similarity. Consider a dataset where the input G of tagging action groups consists of $n = 5$ 3-dimensional tag signature vectors. Let the group tag vectors be $T_{rep}(g_1) = [0.6, 0.2, 0.2]$, $T_{rep}(g_2) = [0.1, 0.7, 0.1]$, $T_{rep}(g_3) = [0.1, 0.1, 0.8]$, $T_{rep}(g_4) = [0.6, 0.4, 0.0]$ and $T_{rep}(g_5) = [0.4, 0.2, 0.4]$. The tagging vectors can be obtained via multiple methods including tf-idf or LDA. The dimensionality $d = 3$ of the vectors correspond to the tag topics under consideration, and can be words like* `love`, `oscar winning`, `gory`, *etc. The tagging action groups are user-describable. Specifically, let the descriptions of g_1 be* {`gender = female`}, *g_2 be* {`state = texas`}, *g_3 be* {`state = california`}, *g_4 be* {`gender = female, state = texas`} *and g_5 be* {`gender = female, state = california`} *respectively. The objective is to identify the result set G^{opt} of $k = 2$ groups having maximum similarity in tagging behavior for the dataset under consideration. The naive way (Exact Algorithm) would perform $\binom{n}{k} = \binom{5}{2} = 10$ comparisons to find the best pair, which is not always a feasible solution. Our SM-LSH algorithm helps us retrieve G^{opt} in the following manner.*

Let the LSH parameters be $d' = 2$ and $l = 1$; let the randomly chosen vectors be $r_1 = [+1, -1, 0]$, $r2 = [-1, -1, +1]$. We reduce the dimensionality of each vector from $d = 3$ to $d' = 2$. For a vector $T_{rep}(g_x)$, the first component of its corresponding lower dimensional representation is $r_1.T_{rep}(g_x)$, while its second component is $r_2.T_{rep}(g_x)$. If a component is non negative, we set it to 1 else to 0. As an example, given vector $T_{rep}(g_1)$, its lower dimensional representation is $[T_{rep}(g_1) \cdot r_1, T_{rep}(g_1) \cdot r_2] = [0.426, -0.52]$. This is then transformed to $[1, 0]$. Repeating the same procedure for the other vectors, we get their lower dimensional representations as: $g_2 = [0, 0]$, $g_3 = [1, 1]$, $g_4 = [1, 0]$ and $g_5 = [1, 0]$. Out of the $2^{d'} = 2^2 = 4$ possible buckets, we have 3 non empty buckets. SM-LSH finds out that the only bucket with at least $k = 2$ elements is $[1, 0]$. This bucket, incidentally, also contains the optimal solution. This is identified by finding the best $k = 2$ tagging action groups in the set $\{g_1, g_4, g_5\}$ by enumerating all possible pairs. The result $G^{opt} = \{g_4, g_5\}$ can be interpreted as: female users in the dataset under consideration have similar tagging behavior.

Note that, we have considered one of the Problems 1, 2, or 3 in Table 1 in our running example. Each of the problems could have multiple hard constraints. Techniques to refine the SM-LSH results for satisfiability of the hard constraints are discussed later in Sections 4.2 and 4.3.

Complexity Analysis: For each round, the pre-processing of locality sensitive hashing time is bounded by $O(nld)$. The number of rounds is logarithmic in d in the worst case. The time complexity for post-processing phase where the buckets are ordered for ranking by scoring function depends on the maximum number of non empty buckets B for any hash table and the size of the largest bucket n_b resulting in $O\left(Bl\binom{n_b}{k}\right)$. Notice that $\binom{n_b}{k}$ gives the complexity of finding the best k-subset in a bucket. The space complexity of the algorithm is $O(nl)$ since there are l hash tables and each table has at most n buckets.

SM-LSH is a fast algorithm with interesting probabilistic guarantees and is advantageous, especially for high-dimensional input vectors. However, the hard constraints along user and item dimensions are not leveraged in the optimization solution so far. Next, we discuss approaches for accommodating the multiple hard constraints into the solution.

4.2 Dealing with Constraints: Filtering

A straightforward method of refining the result set of SM-LSH for satisfiability of all the hard constraints in TagDM

problem instances is post-processing or **Fi**ltering. We refer to this algorithm as **SM-LSH-Fi**. For each of the l hash tables, we first check for satisfiability of the hard constraints in each bucket and then rank the buckets (satisfying hard constraints) according to the scoring function in order to determine the result set of tagging action groups G^{app} (We represent G^{opt} as G^{app} since LSH based technique now perform approximate nearest neighbor search) with maximum similarity. Such post-processing of buckets for satisfiability of hard constraints may also return null results, if post-processing of hash tables yields no bucket satisfying all the hard constraints. Therefore, we propose a smarter method that folds the hard constraints concerning similarity as part of vectors in high-dimensional space, thereby increasing the chances of similar groups hashing into the same bucket.

4.3 Dealing with Constraints: Folding

Problems 2 and 3 in Table 1 has two out of the three tagging action components to be mined for similarity. In order to explore the main idea of LSH, we **Fo**ld the hard constraints maximizing similarity as soft constraints into our SM-LSH algorithm in order to hash similar input tagging action groups (similar with respect to group tag signature vector and user and/or item attributes) into the same bucket with high probability. We refer to this algorithm as **SM-LSH-Fo**. We fold the user or item similarity hard constraints in Problems 2 and 3 respectively into the optimization goal and apply our algorithm, so that tagging action groups with similar user attributes or similar item attributes, and similar group tag signature vectors hash to the same bucket. For each tagging action group $g_x \subseteq G$, we represent the categorical user attributes or item attributes as a boolean vector and concatenate it with $T_{rep}(g_x)$. We map n vectors from a higher ($d + \sum_{i=1}^{|S_U|}\sum_{j=1}^{|a_i|}|a_i = v_j|$) dimensional space for users (replace $|S_U|$ with $|S_I|$ for items) to a lower d' dimensional space. Similar to Algorithm 1, we consider l LSH hash functions and then post-process the buckets for satisfiability of the remaining constraints in order to retrieve the final result set of tagging action groups G^{app} with maximum optimization score. Problem 1 in Table 1 has all three tagging action components set to similarity. In this case, we build one long vector for each tagging action group $g_x \subseteq G$ by concatenating boolean vector corresponding to categorical user attributes, boolean vector corresponding to categorical item attributes and numeric tag topic signature vector $T_{rep}(g_x)$. The dimensionality of the high-dimensional space for Problem 1 is $d + \sum_{i=1}^{S_U}\sum_{j=1}^{|a_i|}|a_i = v_j| + \sum_{i=1}^{S_I}\sum_{j=1}^{|a_i|}|a_i = v_j|$.

4.4 Practical Considerations

While Theorem 3 establishes the theoretical probabilistic bound of finding the optimal result set, there are a number of practical issues to consider.

The first issue is how to set the initial values for parameters d' and l. Ideally, the parameters must result in buckets such that their expected size be k. This will ensure that all the tagging action groups in the bucket becomes a candidate output set. We can use Theorem 1 in [12] to set the initial values to be

$$d' = \log_{1/P_2}\frac{n}{k}$$

$$l = \left(\frac{n}{k}\right)^{\rho}$$

where $\rho = \frac{\ln 1/P_1}{\ln 1/P_2}$. However, there are a number of work including [42, 10, 33] on LSH-parameter tuning that can be used as well.

An important issue to notice is that LSH is a Monte Carlo randomized algorithm. In other words, while the probability of finding the optimal solution is reasonable, it is possible that we did not find in our first attempt. There are two possible ways to boost the success probability. First, we can increase the number of trials of our algorithm but keeping the same values for parameters d' and l. An alternate approach, which we have used in our algorithm, is to reduce the number of hash functions d'. In both cases, each round of our algorithms are *independent*. In other words, the hash functions are chosen from scratch. We then identify the best set of tagging action groups for each round and choose the best over all rounds.

Consider the approach where we change the parameter d'. Intuitively, as the value of d' decreases, the expected number of input items that fall into any bucket increases. In the extreme case, for $d' = 1$, we expect half the points to fall in the bucket $h_r(.) = 0$ and half in $h_r(.) = 1$. This particular design choice is appealing as the expected size of buckets increases, the chances of the optimal set of tagging action groups to belong to the same bucket also increases.

The next design choice involves how to choose the dimensionality d'' for the next iteration. For example, we can decrement the current number of hash functions to project to the next lower dimension. i.e. $d'' = d' - 1$. Alternatively, we can be more aggressive and reduce the dimensionality by some constance factor (such as $d'' = \frac{d'}{2}$). There is a cost-benefit tradeoff here. On one hand, when the dimensionality is cut by half, the expected number of items falling into a bucket dramatically increases. However, there is a corresponding increase in the probability of finding the optimal set of points also. We choose an approach that balances runtime vs success probability. If we have not identified enough candidates for a given value of d', the value for next invocation is chosen as $d'' = \frac{d'}{2}$. However, if the resulting bucket sizes are too large for d'' (thereby finding the best subset of k points in a bucket would be very expensive), we then choose a new dimension half-way between d'' and d'. In the worst case, the value of d' goes all the way to 1 and our algorithm degenerates to **Exact**. We note that there exist a number of

theoretical and empirical work on tuning LSH parameters. The most common techniques to handle failure (in our case, it is the lack of buckets with atleast k tagging action groups in it) are reducing the dimension [42,10,33,25], choosing the best parameters based on their performance over different samples over dataset[3] or based on their distance profiles [42] and finally multiprobing [31]. We chose the approach of reducing the dimension as it is the most intuitive and requires the least amount of additional information.

Both SM-LSH-Fi and SM-LSH-Fo are efficient algorithms for solving TagDM similarity maximization problem instances and readily out-performs the baseline Exact, as shown in Section 7. However, there are other instantiations namely, Problems 4, 5 and 6 in Table 1 which concern tag diversity maximization. Since it is non-obvious how the hash function may be inversed to account for dissimilarity while preserving the properties of LSH, we develop another set of algorithms (less efficient than LSH based, as per complexity analysis) in Section 5 for diversity problems.

5 FDP Based Algorithms

The second of our algorithmic solutions borrows ideas from methods employed in computational geometry, which model data objects as points in high dimensional space and determine a subset of points optimizing some objective function. Such geometric problem examples include clustering a set of points in euclidean space so as to minimize the maximum intercluster distance, computing the k^{th} smallest or largest inter-point distance for a finite set of points in euclidean space, etc. Since we consider tagging action groups as tag signature vectors, and since the cardinality of the global set of topics (that, in turn, determines the size of each vector) is often high, computational geometry based approach is an intuitive choice to pursue.

We focus on a specific geometric problem, namely the facility dispersion problem (**FDP**), which is analogous to our TagDM problem instances, finding the tagging action groups maximizing the mining criterion. The facility dispersion problem deals with the location of facilities on a network in order to maximize distances between facilities, minimize transportation costs, avoid placing hazardous materials near housing, outperform competitors' facilities, etc. We consider the problem variant in Ravi et al.'s paper [38] that maximizes some function of the distances between facilities. The optimality criteria considered in the paper are MAX-MIN (i.e., maximize the minimum distance between a pair of facilities) and MAX-AVG (i.e., maximize the average distance between a pair of facilities). Under either criterion, the problem is known to be NP-hard by reduction from the Set Cover problem, even when the distances satisfy the triangle inequality [11]. The authors present an approximation algorithm for the MAX-AVG dispersion problem, that provides a performance guarantee of 4. The algorithm initializes a pair of nodes (i.e., facilities) which are joined by an edge of maximum weight and adds a node in each subsequent iteration which has the maximum distance to the nodes already selected.

The facility dispersion problem solution provides an approach to determine a set of tagging actions groups that have maximum average pair-wise distance, i.e., that are dissimilar in their tagging behavior. We consider each of the input n tagging action groups as d-dimensional tag signature vector in a unit hypercube and intend to identify k vectors with maximum average pair-wise distance between them. We compare the input set G of n tagging action groups using a pair-wise comparison function $F''_p(g_1, g_2, \texttt{tags}, diversity)$ that operates on tagging action group signature vectors; and return the set of tagging groups $\leq k$ having maximum diversity in tagging behavior.

Our FDP based algorithms work for Problems 4, 5 and 6 in Table 1 maximizing tag diversity. We first introduce an algorithm that returns the groups having maximum diversity in tagging behavior (Column O in Table 1) and then discuss additional techniques to handle the multiple hard constraints in the solution (Column C in Table 1).

5.1 Maximizing Diversity based on FDP

Our FDP based algorithm **DV-FDP** handles TagDM problem instances optimizing tag **D**i**V**ersity. Given an input set G of n tagging action groups, each having a numeric tag signature vector $T_{rep}(g_x)$, where $g_x \subseteq G$, we build the result set G^{app} (we represent the result set as G^{app} since the technique returns approximate solution) by adding a tagging action group in each iteration which has the maximum distance to the groups already included in the result set. Again, we use cosine similarity measure between two tag signature vectors for determining the distance since the distance metric hold triangular inequality property. Thus, our DV-FDP attempts to find one tight set of k groups with maximum average pair-wise distance between them. The approximation bounds for this algorithm follows from [38]:

Theorem 4 *Let I be an instance of the TagDM problem maximizing the mining criterion with $k \geq 2$ and no other hard constraints, where the collection of n d-dimensional vectors are in a unit hypercube satisfying the triangle inequality. Let G^{opt} and G^{app} denote respectively the optimal set of k tagging action groups returned by Exact and DV-FDP algorithms. Then $G^{opt}/G^{app} \leq 4$.*

Algorithm 2 is the pseudo-code of our DV-FDP algorithm. Once the $n \times n$ distance matrix S^G is built using the cosine distance function, the implementation exhaustively scans S for determining the best add operation in each of the

Algorithm 2 DV-FDP (G, O, k)**:** G^{app}

//Main Algorithm
1: $S^G \leftarrow$ Compute $n \times n$ Distance Matrix(G)
2: $\{g_x, I_x, g_y, I_y\} \leftarrow \text{MAX}(S^G)$
3: $A \leftarrow [g_x, g_y]$
4: **while** $A \neq k$ **do**
5: $\quad g_z \leftarrow \Sigma_{\{z' \in [A], z \in [G-A]\}} \text{MAX}(S^{G-A})$
6: $\quad A \leftarrow [A, g_z]$
7: **end while**
8: $G^{app} \leftarrow A$
9: return G^{app}

subsequent iterations. If A represents the result set, the objective is to find an entry from $G-A$ to add to A, such that its total sum of weight to a node in A is maximum.

Example 2 *Let us consider one of the Problems 4, 5, or 6 in Table 1 where the objective is to optimize tag diversity. Let us consider the same dataset and input set G of tagging action groups, as in Example 1. The objective is to identify the result set G^{app} of $k = 3$ groups having maximum diversity in tagging behavior. Using the cosine distance (computed as 1.0 - cosine similarity score for capturing diversity) as our distance measure, DV-FDP functions as follows. The first step is to compute the matrix S^G, which is shown in Table 2. The pair of tagging action groups with highest distance between them are g_3 and g_4. These form our initial approximate group $G^{app} = \{g_3, g_4\}$. Next, we find the tagging action group that has the largest aggregate distance from g_3 and g_4. The tagging action group has an aggregate distance of $0.55 + 0.08 = 0.63$ from the set g_3, g_4. The corresponding values for g_2 and g_5 are 1.06 and 0.48 respectively. This means that g_2 becomes the next member of the output group and the algorithm returns $G^{app} = \{g_2, g_3, g_4\}$. Observe that in this example, the approximate and optimal answers happen to be identical, which may not be the case in general. The result $G^{app} = \{g_2, g_3, g_4\}$ can be interpreted as: users in California and Texas have diverse tagging behavior for the dataset under consideration.*

Table 2 Distance matrix S^G

	g_1	g_2	g_3	g_4	g_5
g_1	0.00	0.54	0.55	0.08	0.10
g_2	0.54	0.00	0.72	0.34	0.49
g_3	0.55	0.72	0.00	0.83	0.22
g_4	0.08	0.34	0.83	0.00	0.26
g_5	0.10	0.49	0.22	0.26	0.00

Note that, we have considered one of the Problems 4, 5, or 6 in Table 1 in our running example. Each of the problems have multiple hard constraints. Techniques to refine the DV-FDP results for satisfiability of the hard constraints are discussed later in Sections 5.2 and 5.3.

Complexity Analysis: The complexity of the implementation of the DV-FDP algorithm is $O(n^2 + nk)$, i.e., $O(n^2)$ due to operations around the $n \times n$ distance matrix S^G. The space complexity of the algorithm is $O(n^2)$. Note that, our LSH based algorithms have better space and time complexity than FDP based algorithms. However, experiments in Section 7 show comparable execution time for LSH and FDP based algorithms in a practical setting.

Like SM-LSH, this algorithm does not leverage the hard constraints along user and item dimensions into the optimization solution, as well. We now illustrate approaches for including the multiple hard constraints into the solution.

5.2 Dealing with Constraints: Filtering

Similar to SM-LSH-Fi, a straightforward method of refining the result set of groups for satisfiability of all the hard constraints in TagDM problem instances is post-processing or **F**iltering. We refer to this algorithm as **DV-FDP-Fi**. Once the result set G^{app} of k groups is identified, we post-process it to retrieve the relevant answer set of tagging action groups, satisfying all the hard constraints. Now, such post-processing of the result set for satisfiability of hard constraints may return null results frequently and hence we propose a smarter algorithm that folds some of the hard constraints into DV-FDP, thereby decreasing the chances of hitting a null result.

5.3 Dealing with Constraints: Folding

In contrast to general DV-FDP algorithm whose objective is to add groups to the result set greedily so that average pair-wise distance is maximized, we want to retrieve the set in each iteration whose members, besides being dissimilar, satisfy many other constraints. In DV-FDP, the greedy add operation in Line 5 of Algorithm 2 maximizes tag diversity. If the algorithm includes a bad tagging action group to the result set in an iteration, the algorithm may return null result or an inferior approximate result, after final filtering of the result set for hard constraint satisfiability. Therefore, we propose our second approach in which hard constraints maximizing diversity are **F**olded into the add operation. We refer to this algorithm as **DV-FDP-Fo**. During each new group addition to the result set, we not only check for the pair with maximum distance, but also check for the satisfiability of the diversity maximization hard constraints on user and item dimension, if any. The algorithm terminates when the number of groups in result set equals k. Once the result set of k groups is identified, we post-process the set for satisfiability of the similarity maximization hard constraint(s) and support constraint, in order to retrieve the answer result of tagging action groups $G^{app'}$.

Complexity Analysis: The time and space complexity of the algorithm continues to be $O(n^2)$ in the worst case.

6 Extensions to TagDM Framework

The TagDM framework in Definition 5 consists of tagging behavior dimensions (i.e., users, items and tags), a set of constraints and optimization goal, and the mining function for measuring similarity and diversity. In this section, we describe natural extensions to TagDM framework that allow it to be more expressive, and provide additional algorithmic solutions. Our extensions are two fold: first, we allow the inclusion of conditions over the dimension(s) in the optimization goal; second, we generalize the mining function from pair-wise aggregation to arbitrary dual mining functions.

6.1 Conditions in optimization goal: HAC Based Algorithm

Recall that TagDM framework handles a set of social tagging behavior analysis tasks that optimizes one or more of the tagging action components (i.e., users, items, tags), and adds to constraints components which are not included in the optimization goal. However, the type of optimization goal allowed in the TagDM framework is highly specific: it operates on one or more dimensions and the target function maximizes a similarity or diversity score over the tagging action groups. In a number of practical cases, the user may be interested in adding few other conditions over dimensions that are not easily expressed in terms of similarity or diversity. For example, a user can be interested in finding tagging action groups that maximize a *tag diversity* measure and satisfy *user and item similarity* constraints, such that the tagging groups have *frequent taggers*. Or, a user can be interested in finding groups that maximize a combination of *tag diversity and user diversity* measures and satisfy an *item similarity* constraint such that the groups have *median user age of 30*. In other words, the analysis task may require us to optimize a dimension along with the condition that the dimension satisfy some property over its distribution.

We can see that it is not easy to extend the previously described algorithms to handle arbitrary conditions. First, the hash functions used in LSH do not accommodate any mechanism for additional conditions. The algorithm SM-LSH-Fo was possible because the constraints F_1, F_2 and F_3 were based on similarity. However, if they had been based on other properties such as frequency, the folding technique quickly proves inadequate. While the FDP based algorithms can fold the constraints, it is tied to maximizing the average pair-wise distance between the facilities (i.e., the groups). Hence, we need a *general* algorithm that can seamlessly handle arbitrary conditions over dimensions in optimization goal for both similarity and diversity mining problems.

We propose an algorithm based on hierarchical agglomerative clustering (**HAC**) that has the following advantages:

- It can handle both similarity and diversity maximization problems, unlike LSH and FDP based which can handle similarity and diversity respectively.
- It is capable of similarity/diversity maximization along with some property of the distribution associated with the target optimization function, while LSH and FDP based cannot. In addition, by changing how the clusters are merged, it can handle arbitrary objective functions.

Hierarchical agglomerative clustering is a popular *bottom-up clustering* technique in which each data observation is treated as a singleton cluster at the outset, and then successively merged pair-wise until all clusters have been merged into a single cluster containing all data observations [34]. This unsupervised technique outputs an informative tree-like structure (known as *dendogram*) efficiently. It takes a symmetric matrix of distances between data observations as input, thereby helping us accommodate both tag similarity and tag diversity maximization problems. The merges in each iteration are determined greedily by searching the distance matrix for a pair separated by smallest distance (in case of similarity maximization) and by largest distance (in case of diversity maximization.)

We particularly consider the average linkage HAC variant which merges pair of clusters with the minimum (or maximum) average distance from any member of one cluster to any member of the other cluster, since this function is equivalent to our Pair-Wise Aggregation Dual Mining Function F_{pa}. For the problem instantiations concerning tag similarity or diversity maximization, we need to compare the input set G of n tagging action groups using a pair-wise comparison function $F''_p(g_1, g_2, \texttt{tags}, m)$, $m \in \{\texttt{similarity}, \texttt{diversity}\}$ that operates on $T_{rep}(g_x)$, where $g_x \subseteq G$. Once again, the result set of tagging action groups with maximum tag similarity or diversity, can be retrieved by determining the k vectors (from n d-dimensional tag signature vectors) with minimum or maximum average pair-wise distance between them respectively. Our HAC based algorithm is often less efficient than LSH and FDP based algorithms as we see in Section 7, but are capable of handling a wide variety of complex mining problems which LSH and FDP based techniques cannot.

We adopt the HAC technique in the following way to handle our problem instances. Given an input set G of n tagging action groups, each having a numeric tag signature vector $T_{rep}(g_x)$, where $g_x \subseteq G$, we employ average linkage HAC to merge clusters in each iteration. Once the dendogram optimizing tag similarity or diversity is generated, we post-process the dendogram in a top-down manner to retrieve the result set of tagging action groups G^{app} (We represent G^{opt} as G^{app} since HAC return approximate solutions) satisfying the hard constraints. Note that, the handling of the additional property, i.e., satisfiability of condition in the optimization goal is conducted during the merging of clusters in each iteration. In other words, we want to construct a clustering such that members of the cluster, besides being similar (or dissimilar), satisfy additional properties and many constraints.

Thus, while traditional HAC algorithms are used for clustering a dataset into different partitions, our algorithm attempts to find one tight small group satisfying the constraints and maximizing the conditional mining criterion.

In our HAC based algorithm optimizing tag similarity or diversity, the target function conditions, denoted by $F_C(G^{app}$, `tags`), and multiple hard constraints are *folded* into the merge operation. During each merge operation, we not only check for the pair with maximum (or, minimum) average pair-wise cosine similarity score, but also check for the satisfiability of the condition along the tagging dimension, as well as hard constraints $1 \leq |G^{app}| \leq k$, $F'_P(\mathscr{G}_x, \mathscr{G}_y, \texttt{users}, \text{m}) \geq q$ and $F'_P(\mathscr{G}_x, \mathscr{G}_y, \texttt{items}, m) \geq \text{r}$, where $\mathscr{G}_x$, $\mathscr{G}_y$ are intermediate clusters, each consisting of a set of tagging action groups. The algorithm terminates when $Support_G^{G^{app}} \geq p$, often without having to build the complete dendogram.

Algorithm 3 is the pseudo-code of the HAC based algorithm with average-link based agglomerative method. The naive implementation of HAC algorithm is $O(n^3)$ since it will exhaustively scan the $n \times n$ distance matrix S for determining the best merge in each of the $(n-1)$ iterations. However, if the function to compute the pair-wise distance $F''_p(g_1, g_2, \texttt{tags}, m)$ is *cosine similarity*, then in conjunction with *heap based priority queues* which have $O(\log n)$ time for inserts and deletes, the algorithm will have a complexity of $O(n^2 \log n)$. In Algorithm 3, the rows of the $n \times n$ distance matrix S are sorted based on distance in the priority queues P. $P[x].MAX()$ or $P[x].MIN()$ returns the cluster in $P[x]$ that currently has the highest similarity or dissimilarity with $\mathscr{G}_x$, where $\mathscr{G}_x$ is the intermediate cluster formed in the x^{th} iteration. $\mathscr{G}_x$ is chosen as the representative of the cluster obtained by merging $\mathscr{G}_x$ and $\mathscr{G}_y$. After each merge operation, the priority queues maintained for each cluster are updated. The cluster similarity or dissimilarity computation takes constant time if vector sums $\sum_{g_x \in \mathscr{G}_x} T_{rep}(g_x)$ and $\sum_{g_y \in \mathscr{G}_y} T_{rep}(g_y)$ are available, where $\mathscr{G}_x$ and $\mathscr{G}_y$ are intermediate clusters being selected for merging. This follows from the following Theorem 5 [32]:

Theorem 5 *The group average of the merged clusters for cosine similarity in average linkage hierarchical agglomerative clustering is given by :*

$$F''_P(\{\mathscr{G}_x, \mathscr{G}_y\}, \texttt{tags}, m)$$
$$= \frac{1}{(N)(N-1)} \sum_{g_x \in \mathscr{G}_x} \sum_{g_y \in \mathscr{G}_y; g_x \neq g_y} T_{rep}(g_x).T_{rep}(g_y)$$
$$= \frac{1}{(N)(N-1)} [(\sum_{g_x \in \mathscr{G}_x} T_{rep}(g_x) + \sum_{g_y \in \mathscr{G}_x} T_{rep}(g_y))^2 - (N)]$$

where $N = n_i + n_j$, $\mathscr{G}_x$ and $\mathscr{G}_y$ are intermediate clusters being selected for merging, $T_{rep}(g_x)$ and $T_{rep}(g_y)$ are length normalized tag signature vectors of corresponding to tagging groups g_x and g_y respectively, . denotes the dot product, n_x and n_y are the number of groups in $\mathscr{G}_x$ and $\mathscr{G}_y$ respectively. Therefore, the distributivity of the dot product with respect to vector addition aids constant time cluster merge condition computation. Note that, when the TagDM problem instance optimizes similarity in tagging behavior, the algorithm merges two most similar clusters having maximum average pair-wise cosine similarity score; when the problem instance optimizes diversity in tagging behavior, it merges two most dissimilar clusters having minimum average pairwise cosine similarity score.

Algorithm 3 HAC based (G, O, C, k, p): G^{app}

//Main Algorithm
1: $S, I, P \leftarrow$Compute $n \times n$ Distance Matrix(G)
2: $A \leftarrow []$; $i \leftarrow 1$; $G^{app} \leftarrow \{\}$
3: **while** $Support_G^{G^{app}} \geq p$ **do**
4: $\mathscr{G}_x \leftarrow argmax_{\{i:I[i]=1\}}$ P[i].m().distance //Using *cosine*
//$m() \in \{\texttt{MAX}(), \texttt{MIN}()\}$ for $m \in \{\texttt{similarity}, \texttt{diversity}\}$
5: $\mathscr{G}_y \leftarrow function($ $P[i].m().index$, $F_C(G^{app}, \texttt{tags})$, $|\{\mathscr{G}_x, \mathscr{G}_y\}| \leq k$, $F'_P(\{\mathscr{G}_x, \mathscr{G}_y\}, \texttt{users}, m) \geq q$, $F'_P(\{\mathscr{G}_x, \mathscr{G}_y\}, \texttt{items}, m) \geq r)$
6: A.append $(\langle \mathscr{G}_x, \mathscr{G}_y \rangle)$
7: $S, I, P \leftarrow$ Update Priority Queue(C, S, P, x, y)
8: $i \leftarrow i+1$
9: **end while**
10: $G^{app} \leftarrow$ Post-process(A, k, p, C)
11: return G^{app}

//Compute $n \times n$ Distance Matrix(G): S, I, P
1: **for** x = 1 to n **do**
2: **for** y = 1 to n **do**
3: $S[x][y].distance \leftarrow T_{rep}(g_x).T_{rep}(g_y)$
4: $S[x][y].index \leftarrow i$
5: **end for**
6: $I[x] \leftarrow 1$
7: $P[x] \leftarrow$ priority queue for $S[x]$ sorted on cosine similarity
8: $P[x].delete(S[x][x])$
9: **end for**
10: return S, I, P

//Update Priority Queue(S, I, P, x, y): S, I, P
1: $I[y] \leftarrow 0$
2: $P[x] \leftarrow []$
3: **for** z **do**
4: **if** $I[z] = 1 \wedge z \neq x$ **then**
5: $P[z].delete(C[z][x])$
6: $P[z].delete(C[z][y])$
7: $S[z][x].distance \leftarrow F(\{\mathscr{G}^x, g_y\}, tags, m)$
8: $P[z].insert(C[z][x])$ //$\mathscr{G}^x$ is intermediate cluster of g_z and g_x
9: $S[x][z].distance \leftarrow F(\{\mathscr{G}^x, g_y\}, tags, m)$
10: $P[x].insert(S[x][z])$
11: **end if**
12: **end for**
13: return S, I, P

Example 3 *Let us re-examine the problem of identifying the most similar set of tagging action groups, as in Example 1. We consider the same dataset and input set G of tagging action groups. Let us re-define the optimization goal as: identify the result set G^{app} of $k = 3$ groups having maximum similarity in tagging behavior such that majority (i.e., more*

than 50%) of the users in the tagging groups are frequent taggers (i.e., have tagged at least 25 items, say). Using cosine similarity as the distance measure, we first build the distance matrix S which is shown in Table 3.

Table 3 Distance matrix S^G

	g_1	g_2	g_3	g_4	g_5
g_1	1.00	0.46	0.45	0.92	0.90
g_2	0.46	1.00	0.28	0.66	0.51
g_3	0.45	0.28	1.00	0.17	0.78
g_4	0.92	0.66	0.17	1.00	0.74
g_5	0.90	0.51	0.78	0.74	1.00

The pair of tagging action groups with highest similarity between them are g_1 and g_4. If the multiple hard constraints in the problem (such as constraints along user and item dimensions, etc.) as well as the additional condition in the optimization goal is satisfied, we merge these two groups to a single cluster $\mathscr{G}_1$. If the constraints and conditions are not taken care of, we proceed with the second most similar pair of tagging action groups, namely g_1 and g_5. Assume, we merge g_1 and g_4 to $\mathscr{G}_1$. Next, we update the similarity between the remaining groups g_2, g_3 and g_5 with $\mathscr{G}_1$ using the average link similarity. In other words, similarity between $\{g_1, g_4\}$ and say g_2 is computed as the average of similarities between pairs (g_1, g_2) and (g_4, g_2). Using the updated matrix, we observe that the cluster $\mathscr{G}_1 = \{g_1, g_4\}$ and tagging action group g_5 have the highest similarity. We check for the satisfiability of the conditions and constraints, and find out that $\{g_1, g_4, g_5\}$ do not satisfy the frequent tagger condition in the optimization goal. That is, less than 50% of users belonging to the set $\{g_1, g_4, g_5\}$ are frequent taggers. Hence, we do not proceed with this merge operation and move to the second best merge option that satisfies the frequent tagger condition and all the hard constraints - we merge the cluster $\mathscr{G}_1 = \{g_1, g_4\}$ and tagging action group g_2 (say). The merged cluster $\mathscr{G}_2$ has $k = 3$ tagging action groups, and also satisfies the support condition (say). Hence, we terminate our algorithm and return the set $G^{app} = \{g_1, g_2, g_4\}$ as the solution. The result G^{app} can be interpreted as: female users in Texas are frequent taggers and have similar tagging behavior for the dataset under consideration.

Complexity Analysis: The complexity of the efficient implementation of the HAC based algorithm is $O(n^2 \log n)$. A priority queue requires $O(\log n)$ time for inserts and deletes, resulting in $O(n \log n)$ time for n priority queues as opposed to $O(n^2)$ distance matrix update time in naive implementation.

6.2 General dual mining functions: HC Based Algorithm

One aspect responsible for bringing in variations in TagDM problem instances is which measure (similarity or diversity) the user is interested in applying to which tagging components (i.e, users, items, or tags). All the algorithmic solutions proposed consider pair-wise aggregation dual mining function in Function 4. However, a user may be interested in more general mining measures which cannot be computed over pairs of tagging action groups, as discussed in Section 2.4. Transitioning from pair-wise to general dual mining functions allows one to characterize holistic properties that occur at the global level in the optimal set of groups instead of simply aggregating local (and pair-wise) properties. Every dual mining functions that the user may want to explore can be placed in the spectrum between local and global properties. While evaluating holistic properties increases the expressive power of the TagDM framework, they also necessitate the development of generic algorithms that can handle arbitrary dual mining functions. Specifically, without additional knowledge of dual mining function properties (such as monotonicity, sub-modularity or even metric property), it is hard to describe deterministic algorithms that have any meaningful approximation guarantees. The Exact algorithm needs to check an exponential number of combinations and is simply not scalable.

We extend ideas from our earlier work [9] and propose a hill-climbing (**HC**) based technique that is capable of handling all the problem instances we have discussed so far with general dual mining defined in Definition 6, instead of pair-wise aggregation mining function with limited scope defined in Definition 4. We build a lattice of all possible tagging action groups (which are structurally describable by user and/or item attributes), where the nodes correspond to user describable and item describable groups and the edges correspond to parent/child relationships. Note that the number of nodes in the lattice of TagDM framework is usually higher than that in the lattice of MRI in [9], since the latter has either user-describable lattice (for item-based query) or item-describable lattice (for user-based query). Also, the scalar numeric rating values have been replaced with numeric vectors in this framework. Our HC based algorithm is advantageous over all the previously discussed algorithms in the following ways:

- It can handle general dual mining measures for similarity and diversity mining which LSH, FDP and HAC based cannot handle. Therefore, this algorithm is capable of solving a wide range of analysis tasks that none of the other three algorithms can.
- It can handle both similarity and diversity maximization problems like HAC based, unlike LSH and FDP based which can handle similarity and diversity respectively.
- It is also capable of similarity/diversity maximization along with some condition in the optimization goal like HAC based, while LSH and FDP based cannot.

A straightforward adoption of the *random restart hill climbing* [43] technique involves the following steps: we

Algorithm 4 HC based (G, O, C, k, p): G^{app}

- Build lattice L_T of all tagging action groups, as in [9].

//Main Algorithm

$G_C \leftarrow$ randomly select k groups/nodes from L_T
if $Support_G^{G_C} \geq p$, $F_C(G_C, \texttt{tags})$, $F'_p(g_1, g_2, \texttt{users}, m) \geq q$ and $F'_p(g_1, g_2, \texttt{items}, m) \geq r$ **then**
  $C \leftarrow$ satisfy-constraints(G_C, L_T)
end if
$C \leftarrow$ optimize-dual-mining-function(C, L_T)
$G^{app} \leftarrow$ best C so far
return G^{app}

// satisfy-constraints(G_C, L_T): C

while true **do**
  val $\leftarrow$ `coverage`(G_C, L_T)
  for each group g_i in G_C, each neighbor g_j of g_i **do**
    $C' \leftarrow G_C - g_i + g_j$
    $val' \leftarrow Support_G^{C'} \geq p$
    if $val' \geq \alpha$, $F_C(C', \texttt{tags})$, $F'_p(g_1, g_2, \texttt{users}, m) \geq q$ and $F'_p(g_1, g_2, \texttt{items}, m) \geq r$ **then**
      $C \leftarrow C'$
      return C'
    end if
  end for
end while

// optimize-dual-mining-function(C, L_T): C

while true **do**
  val $\leftarrow F(C, L_T)$ // F is general dual mining function
  $\mathscr{C} = \emptyset$
  for each group g_i in C, each neighbor g_j of g_i **do**
    $C' \leftarrow C - g_i + g_j$
    if $Support_G^{C'} \geq p$, $F_C(C', \texttt{tags})$, $F'_p(g_1, g_2, \texttt{users}, m) \geq q$ and $F'_p(g_1, g_2, \texttt{items}, m) \geq r$ **then**
      add $(C', F(C, L_T))$ to $\mathscr{C}$
    end if
  end for
  let $(C'_m, val'_m) \in \mathscr{C}$ be the pair with minimum F
  if $val'_m \geq val$ **then**
    return C // we have found the local minima
  end if
  $C \leftarrow C'_m$
end while

first randomly select a set of k tag signature vectors (corresponding to k tagging action groups) as the starting seed; the process then continues by replacing one group in the current set with one of its neighbors[3] not in the set as long as the substitution maximizes or minimizes the general dual mining function; the algorithm stops when no improvements can be made indicating a *local optima* has been reached. The process is repeated with multiple diverse seed sets to increase the probability of finding the *global optima* that satisfies the conditions and constraints. However, this simple application of hill climbing does not suffice because of the inclusion of constraints and/or conditions in the tasks. For any given set of groups randomly chosen as the starting set, the probability of it satisfying all the constraints is fairly small, thereby necessitating a large number of restarts.

Therefore, we consider the *Randomized Hill Exploration Algorithm* [9](**RHE**) which first initializes a randomly selected set of k vectors as the starting set. However, instead of immediately starting to improve the target function, it *explores* the hill to detect sets of k tagging vectors in the neighborhood that satisfy the conditions and constraints. This new set of k group tag signature vectors is then adopted as the starting point for the tag similarity or diversity maximization, with the added condition that an improvement is valid only when the constraints and conditions hold.

The details of the algorithm are shown in Algorithm 4. Intuitively, we begin with the group tagging vector lattice constructed on L_T. The algorithm starts by picking k random groups in the lattice to form the initial seed set G_C. For each group g_i in G_C, we swap g_i with each of its neighbors g_j in the lattice, while the other groups in G_C remain fixed, to generate a new combination. The exploration phase continues till it finds the *best* set of k tagging action groups that satisfies all the constraints and conditions. The resulting set then acts as the initial condition for the second phase of the optimization to maximize or minimize the general dual mining function F measuring tag similarity or diversity. The configuration that satisfies the constraints and conditions, and incurs the optimum value of F is the *best* tagging behavior explanation for the query.

Example 4 *Once again, we re-examine the problem of identifying the most similar set of tagging action groups, as in Example 1 and 3. We consider the same dataset and input set G of tagging action groups. The objective is to identify the result set G^{app} of $k = 3$ groups having maximum similarity in tagging behavior using the HC-based algorithm. Recall that, here we consider a general dual mining function, defined in Definition 6, instead of the pair-wise aggregation cosine similarity measure used in Examples 1, 2, and 3. The tagging action groups are concisely represented as a lattice L_T, as shown in Figure 3. We can see that there are 3 tagging action groups that can be described using a single attribute and 2 that are described using two user attributes.*

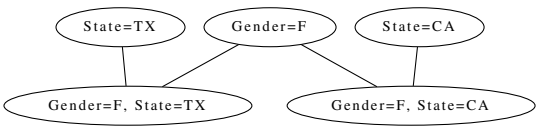

Fig. 3 Lattice representation of tagging action groups for example

When the algorithm starts, it randomly picks 3 groups from L_T. Suppose, it picks the groups containing all female users and the users from Texas and California, i.e., $G_C = \{g_1, g_2, g_3\}$. Our algorithm explores the neighboring groups for each of the candidates in G_C. One of the candidates is g_4,

[3] Two tagging action groups are neighbors if they are directly connected in the lattice.

the set of female users from Texas. If $G'_C = \{g_1, g_3, g_4\}$ improves the aggregate score (measured by general dual mining function) as compared to $G_C = \{g_1, g_2, g_3\}$ while satisfying the multiple constraints and conditions, we update G_C to G'_C. We explore the remaining neighbors (i.e., g_5) and update G_C in a similar fashion. Suppose, $G_C = \{g_1, g_3, g_5\}$ has the highest aggregate score. When there are no more neighbors that can increase the aggregate score, the algorithm terminates since it has reached the local maxima. The result $G^{app} = \{g_1, g_3, g_5\}$ is returned, which can be interpreted as: female users from California have similar tagging behavior.

Complexity Analysis: The worst case complexity of the hill climbing based algorithm is exponential in the size of the search space. The space complexity of the search space, i.e., the number of groups/nodes in the lattice, is a function of the number of attribute values users and items take.

Discussion: Table 4 broadly summarizes our algorithmic contributions for solving the TagDM problem instances in the general TagDM framework and those in the extensions of the TagDM framework.

Table 4 Summary of our Algorithmic Solutions. Column $O.m$ lists the optimization O mining criterion ($m \in \{\texttt{similarity}, \texttt{diversity}\}$), column C_O lists if Algorithm can handle condition(s) in optimization goal, column $O.F$ lists the mining function (F $\in \{F_p(pairwise), F(general)\}$) that Algorithm can handle, and the final column discusses how Algorithm handles hard constraints.

Algorithm	O.m	C_o	O.F	Additional Techniques
LSH based	similarity	no	F_p	fold similarity and filter diversity constraints
FDP based	diversity	no	F_p	fold diversity and filter similarity constraints
HAC based	similarity, diversity	yes	F_p	fold both similarity and diversity constraints during merge
HC based	similarity, diversity	yes	F_p, F	fold both similarity and diversity constraints during exploration

7 Experiments

We conduct a comprehensive set of experiments for quantitative (Section 7.1) and qualitative (Section 7.2) analysis of our proposed algorithms for TagDM problem instances. Our quantitative performance indicators are (a) *efficiency* of the algorithms, and (b) *analysis quality* of the results produced. The efficiency of our algorithms is measured by the overall response time, whereas the result quality is measured by the average pair-wise distance between the k tagging action group vectors returned by our algorithms. In order to demonstrate that the tagging behavior analysis generated by our approaches are interesting to end users, we conduct a user study through Amazon Mechanical Turk. We also present interesting case studies to show how results generated by our algorithms for TagDM problem instances varies.

Data Set: We require dataset(s) that contains information about a set of users tagging a set of items, where attributes associated with users and attributes associated with items are known. We use the MovieLens[4] 1M and 10M ratings dataset for our evaluation purposes. The MovieLens 1M dataset consists of 1 million ratings from 6000 users on 4000 movies while the 10M version has 10 million ratings and 100,000 tagging actions applied to 10,000 movies by 72,000 users. The titles of movies in MovieLens are matched with those in the IMDB dataset[5] to obtain movie attributes.

User Attributes: The 1M dataset has well-defined user attributes but no tagging information, whereas the 10M dataset has tagging information but no user attributes. Therefore, for each user in the 1M dataset with a complete set of attributes, we build her rating vector and compare it to the rating vectors (if available) of all 72,000 users in the 10M dataset. For every user in 10M dataset, we find the user in 1M dataset such that the cosine similarity of their movie rating vector is the highest (i.e., user rating behaviors are most identical). The attributes of user in 10M dataset are obtained from the closest user in 1M dataset. In this way, we build a dataset consisting of 33,322 tagging and rating actions applied to 6,258 movies by 2,320 users. The tag vocabulary size is 64,663. The user attributes are gender, age, occupation and zip-code. The attribute *gender* takes 2 distinct values: male or female. The attribute *age* is chosen from one of the eight age-ranges: under 18, 18-24, ..., 56+. There are 21 different *occupations* listed by MovieLens such as student, artist, doctor, lawyer, etc. Finally, we convert zipcodes to states in the USA (or foreign, if not in USA) by using the USPS zip code lookup[6]. This produces the user attribute *location*, which takes 52 distinct values.

Movie Attributes: Movie attributes are genre, actor and director. There are 19 movie *genres* such as action, animation, comedy, drama, etc. The pool of actor values and director values, corresponding to movies which have been rated by at least one user in the MovieLens dataset, is huge. We pick only those actors and directors who belong to at least one movie that has received greater than 5 tagging actions. In our experiments, the number of distinct *actor* attribute values is 697 while that of distinct *director* is 210.

Mining Functions: The set of tagging action groups is built by performing a cartesian product of user attribute values with item attribute values. An example tagging action group is {`gender=male`, `age=under 18`, `occupation=student`, `location=new york`, `genre=action`, `actor=tom hanks`, `director=steven spielberg`}. The total number of possible tagging action groups is more than 40 billion, while the number of tagging action groups containing

[4] *http://www.grouplens.org/node/73*
[5] *http://www.imdb.com/interfaces*
[6] *http://zip4.usps.com*

at least one tuple is over 300K. For our experiments, we consider 4535 groups that contain at least 5 tagging action tuples. The user and item similarity (or diversity) is measured by determining the structural distance between user and item descriptions of groups respectively. For topic discovery, we apply LDA [4] as discussed in Section 2.2. We generate a set of 25 global topic categories for the entire dataset, i.e., $d = 25$. For each tagging action group, we perform LDA inference on its tag set to determine its topic distribution and then generate its tag signature vector of length 25. Finally, we use cosine similarity (or, some general mining measure) for computing pair-wise similarity between tag signature vectors or some general mining measure.

System Configuration: Our prototype system is implemented in Python. All experiments were conducted on an Ubuntu 11.10 machine with 4 GB RAM, AMD Phenom II N930 Quad-Core Processor.

7.1 Quantitative Evaluation

First, we compare the execution time and result quality of all 6 TagDM problem instantiations in Table 1 for the entire dataset (consisting of 33K tuples and 4K tagging action groups) using Exact, SM-LSH-Fi, SM-LSH-Fo, DV-FDP-Fi and DV-FDP-Fo algorithms. We use the name Exact for the brute-force approach on both tag similarity and diversity maximization instances. We set the number of tagging action groups to be returned at $k = 3$, since the Exact algorithm is not scalable for larger k. Figure 4 and 5 compare the execution time and quality respectively of Exact and LSH based algorithms for Problems 1, 2 and 3 (Tag Similarity). Figure 6 and 7 compare the execution time and quality respectively of Exact and FDP based algorithms for Problems 4, 5 and 6 (Tag Diversity). The execution time is the time taken to retrieve the result set. The quality of the result set is measured by computing the average pair-wise cosine similarity between the tag signature vectors of the $k = 3$ tagging action groups returned. The group support is set at $p = 350$ (i.e., 1%); the user attribute similarity (or, diversity) constraint as well as the item attribute similarity (or, diversity) constraint is set to $q = 50\%$, $r = 50\%$ respectively. For LSH based algorithms, the number of hash tables is $l = 1$ while the initial value of d' is 10.

We observe that the execution time of our LSH based for similarity and FDP based algorithm for tag diversity problem instances are much faster than the Exact equivalent. In Figure 4, the execution times of SM-LSH-Fi and SM-LSH-Fo for Problems 1, 2 and 3 are comparable to each other and is around 2 minutes. In Figure 6, the execution times of DV-FDP-Fi and DV-FDP-Fo for Problems 4, 5 and 6 are slightly more than 3 minutes. Despite significant reduction in time, our algorithms do not compromise much in terms of quality, as evident from Figure 5 and 7.

The number of input tagging action tuples available for tagging behavior analysis is dependent on the *query* under consideration. For the entire dataset, there are 33K such tuples. However, if we want to perform tagging behavior analysis of all movies tagged by {`gender= male`} or {`location= CA`}, the number of available tuples is 26,229 and 6,256 respectively. Or, if want to perform tagging behavior analysis of all users who have tagged movies having {`genre= drama`}, the number of tuples is 17,368. Needless to say, the number of tagging action tuples can have a significant impact on the performance of the algorithms since it affects the number of non-empty tagging action groups on which our algorithms operate. As a result, we build 4 bins having 30K, 20K, 10K and 5K tagging action tuples respectively (assume, each bin is a result of some query on the entire dataset) and compare our algorithm performances for one of the tag similarity maximization problems and one of the tag diversity maximization problems, say Problem 1 and Problem 6 from Table 1 respectively. Both Problems 1 and 6 have user and item dimension constraints set to similarity. Figures 8 and 9 compare the execution time and quality respectively of the Exact algorithm with our smart algorithms: SM-LSH-Fo for Problem 1 and DV-FDP-Fo for Problem 6. The group support is set at $p = 350$ (i.e., 1%); user similarity (or, diversity) constraint and item similarity (or, diversity) constraint are set to $q = 50\%$, $r = 50\%$ respectively, and $k = 3$. For each bin along the X axis, the first two vertical bars stand for Problem 1 (tag similarity) and the last two stand for Problem 6 (tag diversity).

As expected, the difference in execution time between our algorithms and the Exact is small for bins with lesser number of tagging tuples for both tag similarity and diversity. However, our algorithms return results much faster than Exact for bins with larger number of tagging tuples. The quality scores continue to be comparable to the optimal answer, as shown in Figures 9.

Next, we analyze the performance behavior of the HAC based and HC based algorithms introduced in Section 6. Recall that, both these techniques were developed to solve a wide variety of complex mining problems which LSH based and FDP based cannot handle. Before showcasing the power of HAC and HC based algorithms, we first compare all the four sets of algorithms - LSH based, FDP based, HAC based, and HC based, under the same settings as above for the 6 TagDM problem instantiations in Table 1. Then, we investigate the potential of HAC based and HC based algorithms to handle analysis tasks which LSH based and FDP based cannot handle.

Figure 10 and Figure 11 compare the execution time and quality respectively of the four algorithms for the 6 TagDM problem instantiations in Table 1 - the first three dealing with Tag Similarity and the last three dealing with Tag Diversity. We employ LSH based technique SM-LSH-Fo for Problems

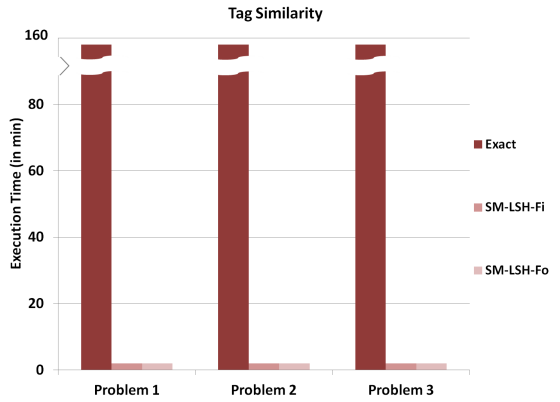

Fig. 4 Execution Time:Problems 1, 2, 3 in Table 1

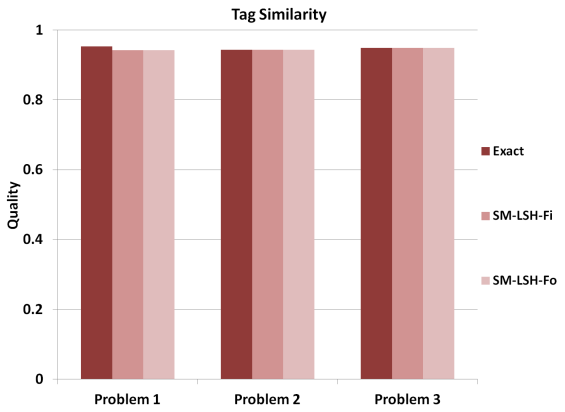

Fig. 5 Quality:Problems 1, 2, 3 in Table 1

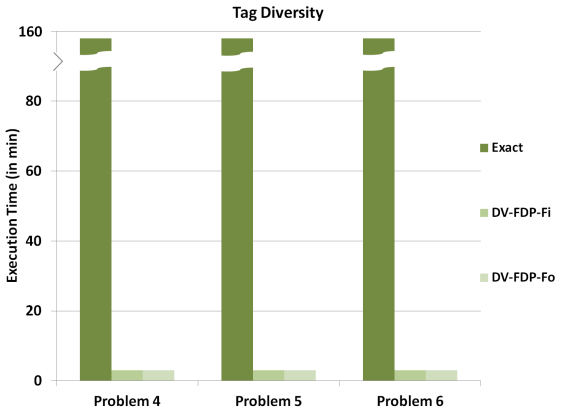

Fig. 6 Execution Time:Problems 4, 5, 6 in Table 1

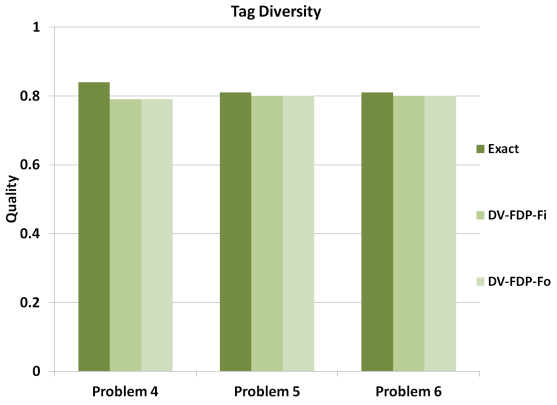

Fig. 7 Quality:Problems 4, 5, 6 in Table 1

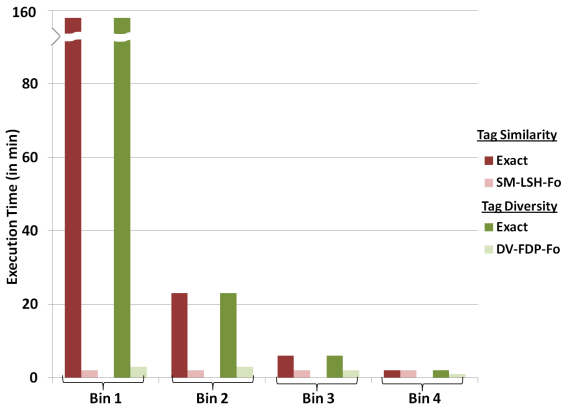

Fig. 8 Execution Time:Varying Tagging Tuples

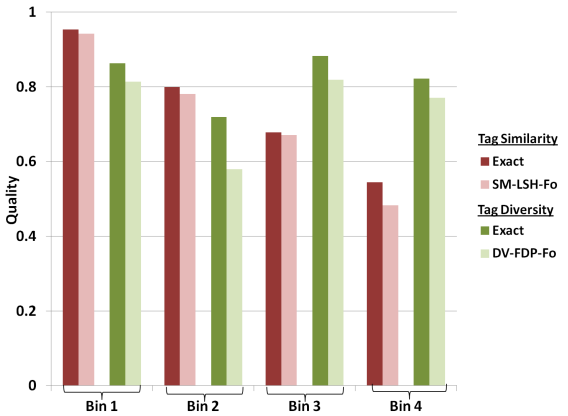

Fig. 9 Quality:Varying Tagging Tuples

1, 2, 3 and FDP based technique SM-FDP-Fo for Problems 4, 5, 6. HAC and HC based techniques are capable of handling both similarity and diversity mining problems. In order to compare the algorithms under the same settings, we consider cosine measure as the dual mining function for HC based algorithm (though it can handle general measures), since the LSH based, FDP based and HAC based methods can only handle pair-wise aggregation mining function. Figure 10 reveals that the time taken by the different algorithms are comparable to each other. For Problems 1, 2, 3, SM-LSH-Fo takes 2 seconds while HAC based and HC based algorithms take around 5 and 6 seconds respectively. For Problems 4, 5, 6, DV-FDP-Fo takes 3 seconds while HAC based and HC based algorithms take around 5 and 6 seconds respectively. From Figure 11, we see that the quality of results returned by the different algorithms for the 6 problems are very close to each other. Though the time taken by HAC and HC algorithms are slightly higher than LSH and

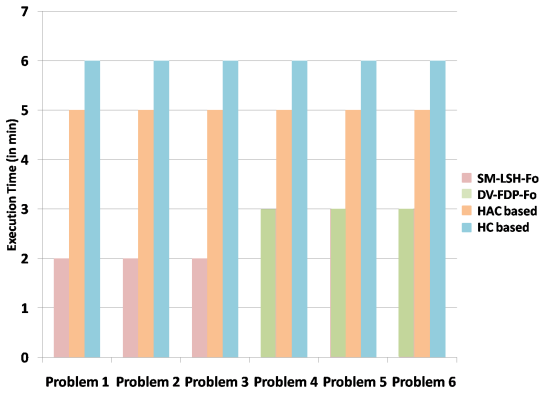

Fig. 10 Execution Time: Different Algorithms

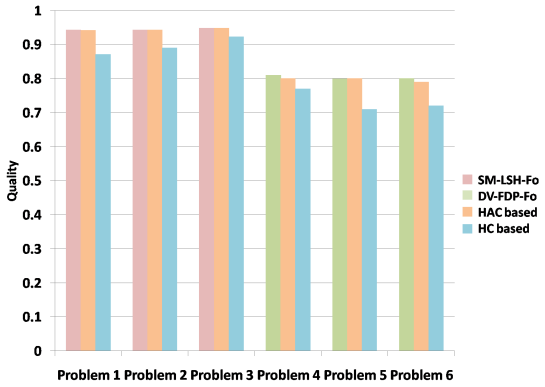

Fig. 11 Quality: Different Algorithms

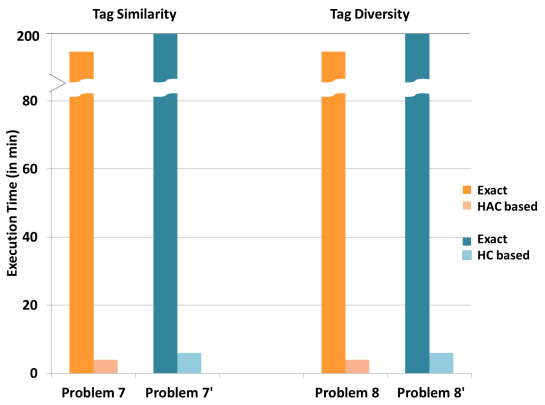

Fig. 12 Execution Time: Exact vs HAC, Exact vs HC

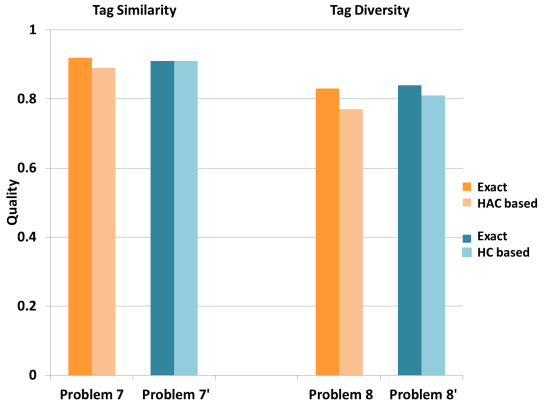

Fig. 13 Quality: Exact vs HAC, Exact vs HC

FDP based techniques for the same set of problems, HAC and HC algorithms returns good quality results and are capable of handling additional complex mining objectives which LSH and FDP based methods cannot handle. Therefore, we recommend the employment of LSH and FDP based algorithms for the simple tasks of TagDM framework, and HAC and HC based algorithms for the more advanced problems belonging to the extended TagDM framework.

In order to evaluate the performance behavior of HAC based algorithm and HC based algorithm for complex mining tasks, we compare the execution time and result quality of two complex TagDM problems - one for similarity, one for diversity - using Exact and HAC based, and Exact and HC based. We use the name Exact for the brute-force approach on both similarity and diversity maximization instances. We set the number of tagging action groups to be returned at k = 3, since the Exact algorithm is not scalable for larger k; the other settings (p,q, etc.) remain same too. The complex mining tasks that we investigate are as follows:

Problem 7: We extend Problem 2 to handle additional condition in the optimization goal. The objective is to find similar user sub-populations who agree most on their tagging behavior for a diverse set of items such that the selected user groups contain at least one tagger from the list of top 50 frequent taggers of the dataset. When the mining measure is general instead of pair-wise, we refer to it as **Problem 7'**.

Problem 8: We extend Problem 5 to handle additional condition in the optimization goal. The objective is to find diverse user sub-populations who disagree most on their tagging behavior for a similar set of items such that the selected user groups contain at least one tagger from the list of top 50 frequent taggers of the dataset. When the mining measure is general instead of pair-wise, we refer to it as **Problem 8'**.

Figures 12 and 13 compare the execution time and quality respectively of Exact and HAC based, as well as Exact and HC based algorithms for Problems 7, 7' (Tag Similarity) and Problems 8, 8' (Tag Diversity). We observe that the execution time of our HAC based and HC based techniques are much faster than the Exact equivalent. Despite significant reduction in execution time, our algorithms do not compromise much in terms of quality, as evident from Figure 13.

7.2 Qualitative Evaluation

We now validate how social tagging behavior analysis can help users spot interesting patterns and draw conclusions about the desirability of an item, by presenting several anecdotal results on real data. We also compare the utility and

popularity of the 6 novel mining problems in Table 1 in an extensive user study conducted on Amazon Mechanical Turk (AMT)[7].

7.2.1 Case Study

First, we present a set of anecdotal results returned by our algorithms for the same query for different TagDM problem instances. Specifically, we focus on Problems 2, 3, and 4 in Table 1 and observe the results returned by our algorithms for the query:
◇ *Analyze tagging behavior of* {`occupation= student`} *users for movies.*

- Problem 2 finds similar user sub-populations who agree most on their tagging behavior for a diverse set of items. We retrieve that male students use similar tags `dystopia`, `sci-fi`, `post-apocalyptic`, etc. for diverse movies "Serenity" and "The Matrix" - the former is a space western movie while the latter is a science fiction action film.
- Problem 3 finds diverse user sub-populations who agree most on their tagging behavior for a similar set of items. We identify that male and female students use similar tags `classic`, `hope`, `friendship`, `based on a book`, etc. for the movie "The Shawshank Redemption".
- Problem 4 finds diverse user sub-populations who disagree most on their tagging behavior for a similar set of items. Our algorithm returns that male and female students use diverse tags for movies directed by "Quentin Tarantino" - male reviewers use tags `crime, cult film, dark comedy`, etc. while female reviewers use `insane`, `visceral`, `ultra-violence`, etc.

Second, we show how TagDM results change due to addition of conditions in optimization goal (Section 6.1) and consideration of general dual mining function (Section 6.2), other settings remaining the same. Let us consider the problem of finding diverse user sub-populations who agree most on their tagging behavior for a similar set of items, and then show how the results change due to inclusion of a condition in the optimization goal. We observe the result returned for the query:
◇ *Analyze tagging behavior of* for {`genre= romance`} movies.

- Young female reviewers and middle-aged female reviewers use similar tags like `classic`, `sweet`, `love`, etc. for romance movies.

If the task is to find out diverse user sub-populations who agree most on their tagging for {`genre = romance`} movies such that majority of the users in the result set have used the tag `love` at least once, the result is:

- Young female reviewers and middle-aged female reviewers use similar tags `Oscar`, `Meg Ryan`, `Nora Ephron`, `Julia Roberts`, etc. for romance movies.

[7] https://www.mturk.com

Thus, we can infer that young and middle-aged female reviewers agree in general in their feedback towards romance movies; when these reviewers use the tag `love`, their agreement is specifically expressed for movies starring Meg Ryan, Julia Roberts, etc.

Let us consider the problem of finding diverse user sub-populations who agree most on their tagging behavior for a similar set of items, where similarity is either measured as pair-wise aggregation (cosine) or is computed using a general mining function. We compare the results returned for the query:
◇ *Analyze tagging behavior for* {`director= steven spielberg`} movies.

- For pair-wise cosine similarity measure: Male and female use similar tags `Oscar`, `true story`, `violence`, etc. for war movies "Saving Private Ryan" and "Schindler's List" directed by Steven Spielberg.
- For general similarity mining measure (say, tagging behavior of result sub-population is closest to tagging behavior of global population): Old male and young male use similar tags `Oscar`, `true story`, `based on a book`, etc. for action movies and fantasy movies directed by Steven Spielberg.

Thus, we can infer that the different mining measures yield different analysis of tagging behavior. Due to the nature of the general mining measure, it returns groups that covers a broader spectrum of movies belonging to the query than that covered by pair-wise cosine similarity aggregation.

Third, we show how results returned by our algorithms for TagDM problem instances varies with increase in k, i.e., the number of user sub-populations being returned. We consider the problem of finding similar user groups who have similar tagging behavior for diverse set of items for the query:
◇ *Analyze tagging behavior of* {`gender= male`} *users for movies.*

- For $k = 2$: Young male students use similar tags `sci-fi`, `dystopia`, `post apocalyptic`, etc. for diverse movies "Serenity" and "The Matrix". The groups returned are:
  g_1 = {⟨`gender`, `male`⟩, ⟨`age`, `young`⟩, ⟨`occupation`, `student`⟩, ⟨`title`, `Serenity`⟩, ⟨`genre`, `spacewestern`⟩, (`sci − fi`, `dystopia`, `apocalyptic`)}
  g_2 = {⟨`gender`, `male`⟩, ⟨`age`, `young`⟩, ⟨`occupation`, `student`⟩, ⟨`title`, `TheMatrix`⟩, ⟨`genre`, `action`⟩, (`sci − fi`, `dystopia`, `apocalyptic`, `martialarts`)}
- For $k = 4$: Young male students use similar tags `sci-fi`, `dystopia`, `philosophical`, `cult`, etc. for diverse movies "Serenity", "The Matrix", "Blade Runner", and movies directed by "Peter Jackson". Movies directed by Peter Jackson include "The Lord of the Rings (film series)".
- For $k = 6$: Young male students use similar tags `sci-fi`, `dystopia`, `based on a book`, `fantasy`, etc. for diverse movies "Serenity", "The Matrix", "Blade Runner", movies

directed by "Peter Jackson", movies directed by "Joss Whedon", and "Eternal Sunshine of the Spotless Mind". Movies directed by Joss Whedon include "Serenity" and "The Avengers".

Thus, we can infer that young male student reviewers agree in general in their feedback towards diverse set of movies - a space western movie, a science fiction action film, a noir detective fiction, a romantic dramedy science fiction film, etc. We can also infer that diversity of items returned as part of the result broadens with increase in the value of k.

TagDM analysis tasks can also throw in surprising results, as we see for the query:
◇ *Analyze tagging behavior of* {`gender= male`, `location= california`} *users for movies.*

– Old male and young male living in California use similar tags for "Lord of the Rings" film trilogy of fantasy genre. However, they differ in their tagging towards "Star Wars" movies having similar genre. This is because, the genre of the latter series borders between fantasy and science fiction. Surprisingly, old male likes it while young male does not.

7.2.2 User Study

We conduct a user study through Amazon Mechanical Turk to elicit user responses towards the different TagDM problem instances we have focused on in the paper, and investigate if the problems are interesting. We generate analysis corresponding to all 6 problem instantiations for the following randomly selected queries:

◇ *Analyze tagging behavior of* {`gender= male`} *users for movies.*
◇ *Analyze tagging behavior of* {`occupation= student`} *users for movies.*
◇ *Analyze user tagging behavior for* {`genre= drama`} *movies.*

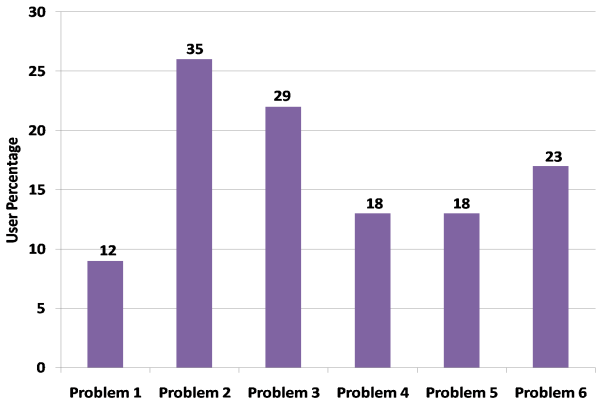

Fig. 14 User Study

We have 50 independent single-user tasks. Each task is conducted in two phases: User Knowledge Phase and User Judgment Phase. During the first phase, we estimate the user's familiarity about movies in the task using a survey, besides her demographics. In the second phase, we ask users to select the most preferred analysis, out of the 6 presented to them, for each query. Since there are 3 queries and 50 single-user tasks, we obtain a total of $3 \times 50 = 150$ responses. The first phase eliminates 5 of the 50 single-users. Therefore, our user study is based on a total of $3 \times 35 = \mathbf{135}$ responses, which are aggregated to provide an overall comparison between all problem instances in Figure 14. The height of the vertical bars represent the percentage of users, preferring a problem instance. We also place the numerical number of user responses against each of the vertical bars. It is evident that users prefer TagDM Problems **2** (*find similar user sub-populations who agree most on their tagging behavior for a diverse set of items*), **3** (*find diverse user sub-populations who agree most on their tagging behavior for a similar set of items*) and **6** (*find similar user sub-populations who disagree most on their tagging behavior for a similar set of items*), having diversity as the measure for exactly one of the tagging component: item, user and tag respectively.

8 Related Work

To the best of our knowledge, our work is the first to develop a general framework that encompasses mining collaborative tagging actions, studies its complexity and develops efficient algorithms. We summarize work related to topic discovery, tag mining and its applications, and the heuristics we use in our algorithms.

Topic Discovery : There are many topic discovery techniques such as Latent Dirichlet Allocation (LDA) [4] [2], tf*idf [39] and OpenCalais. In this work, we use LDA, a generative probabilistic method proven to be robust when looking for hidden topics in Web documents [4] [1]. The classical LDA has to be extended so that they can be applicable for short documents such as microblog tweets and tags [35]. A number of of LDA-based topic models have been proposed for modeling the social annotation using content and tags [36, 30], community interests [47] and user factors [6].

Tag Mining : Tag mining has been used in multiple applications including tag recommendations [29], document navigation [19], item recommendations [28] [18], and tagging motivation [27] and prediction [23]. There has been a steady stream of work on analyzing user's tagging behaviors in collaborative tagging websites such as the nature of tags chosen by users[15, 41], its structure [14], its organizational properties [22]. However, most of these works are tai-

lored to specific datasets and none of them defines a general mining problem, studies its complexity and develops efficient generic algorithms. We posit that despite recent work on improving tag selection for tag clouds [44], tag clouds are an inefficient method to summarize a set of of user subpopulation.

Algorithmic Techniques : Locality Sensitive Hashing (LSH), Facility Dispersion Problem (FDP), Hierarchical Agglomerative Clustering (HAC) and Hill Climbing were first introduced in [24] [12], [20], [5] and [16] respectively. LSH is used in prominent applications including duplicate detection and nearest neighbor queries [24]. In this work, we show how we adapt LSH to rank and choose the best bucket containing tagging analysis result. While being less efficient than LSH, the computational geometry based approach for the facility dispersion problem in [38] serves tag diversity problem instantiations and may be extended to solve similarity problems. While being less efficient than LSH and FDP, HAC serves both tag similarity and diversity. Our idea of using structurally meaningful groups as the basis for tagging behavior interpretation and hill exploration is inspired by our work in [9].

Data Cubes : Our idea of using structurally meaningful cuboids as the basis for tagging interpretation is inspired by studies in the field of data cube mining, first proposed in [17] and [37]. Most of those studies, however, focus on how to efficiently compute aggregate measures for all cuboids and are therefore orthogonal to our work. Among those studies, KDAP [45] and Intelligent Rollups [40] investigate the problem of ranking and summarizing cuboids, which is similar to our goal here. However, none of these adopts formal objective measures based on dual mining over user and item tagging behavior. To the best of our knowledge, our work is the first to leverage structurally meaningful descriptions for collaborative tagging analysis.

Recommendation Explanation : Recommendation systems is a decades old subject that has been gaining popularity recently due to their adoption by online sites such as Amazon and Netflix. In connection to this rising popularity, explaining recommendations has also received significant attention. A systematic study of explanations for recommendation systems is provided by [21]. A discussion of how explanations can be leveraged for recommendation diversification is in [46].

9 Conclusion

In this paper, we developed the first framework to mine social tagging behaviors. We identified a family of mining problems that apply two opposing measures: similarity and diversity, to the main three tagging components: users, items, and tags. We showed that any instance of those is NP-Complete and developed four sets of efficient algorithms based on: local-sensitive hashing (LSH), solutions developed in computational geometry for the facility disperson problem (FDP), hierarchical agglomerative clustering (HAC) and hill climbing (HC). Our extensive experiments on the MovieLens data show the superiority of our algorithms on their baseline counterparts. In the future, we plan to explore the applicability of our framework to other domains such as topic-centric exploration of tweets and news articles, an area that has been receiving a lot of attention lately. In particular, we would like to explore the usefulness of our mining techniques for mining and characterizing events in tweets and news at large.

References

1. S. Amer-Yahia, J. Huang, and C. Yu. Building community-centric information exploration applications on social content sites. In *SIGMOD*, pages 947–952, 2009.
2. S. Amer-Yahia, J. Huang, and C. Yu. Jelly: A language for building community-centric information exploration applications. In *ICDE*, pages 1588–1594, 2009.
3. A. Andoni and P. Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Commun. ACM*, 51(1):117–122, 2008.
4. D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, Mar 2003.
5. C. Böhm, K. Kailing, P. Kröger, and A. Zimek. Computing clusters of correlation connected objects. In *SIGMOD*, pages 455–466, 2004.
6. M. Bundschus, S. Yu, V. Tresp, A. Rettinger, M. Dejori, and H.-P. Kriegel. Hierarchical bayesian models for collaborative tagging systems. In *ICDM*, pages 728–733, 2009.
7. M. Charikar. Similarity estimation techniques from rounding algorithms. In *STOC*, pages 380–388, 2002.
8. Y. Chen, F. M. Harper, J. A. Konstan, and S. X. Li. Social comparisons and contributions to online communities: A field experiment on movielens. In *Computational Social Systems and the Internet*, 2007.
9. M. Das, S. Amer-Yahia, G. Das, and C. Yu. Mri : Meaningful interpretations of collaborative ratings. In *PVLDB*, pages 1063–1074, 2011.
10. W. Dong, Z. Wang, W. Josephson, M. Charikar, and K. Li. Modeling lsh for performance tuning. In *CIKM*, pages 669–678, 2008.
11. E. Erkut, T. Baptie, and B. V. Hohenbalken. The discrete *p*-maxian location problem. *Computers & OR*, 17(1):51–61, 1990.
12. A. Gionis, P. Indyk, and R. Motwani. Similarity search in high dimensions via hashing. In *VLDB*, pages 518–529, 1999.
13. M. X. Goemans and D. P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *J. ACM*, 42(6):1115–1145, 1995.
14. S. A. Golder and B. A. Huberman. The structure of collaborative tagging systems. *CoRR*, abs/cs/0508082, 2005.
15. S. A. Golder and B. A. Huberman. Usage patterns of collaborative tagging systems. *J. Inf. Sci.*, 32(2):198–208, Apr. 2006.
16. S. M. Goldfeld, R. E. Quandt, and H. F. Trotter. *Maximization by Quadratic Hill-Climbing*, volume 34. The Econometric Society, 1966.
17. J. Gray, S. Chaudhuri, A. Bosworth, A. Layman, D. Reichart, M. Venkatrao, F. Pellow, and H. Pirahesh. Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub totals. *Data Min. Knowl. Discov.*, 1(1):29–53, 1997.

18. Y. Guo and J. B. D. Joshi. Topic-based personalized recommendation for collaborative tagging system. In *HT*, pages 61–66, 2010.
19. J. Gwizdka. Of kings, traffic signs and flowers: exploring navigation of tagged documents. In *HT*, pages 167–172, 2010.
20. G. Handler and P. Mirchandani. *Location on networks: theory and algorithms*. MIT Press series in signal processing, optimization, and control. 1979.
21. J. L. Herlocker, J. A. Konstan, and J. Riedl. Explaining collaborative filtering recommendations. In *CSCW*, pages 241–250, 2000.
22. P. Heymann, A. Paepcke, and H. Garcia-Molina. Tagging human knowledge. In *WSDM*, pages 51–60, 2010.
23. P. Heymann, D. Ramage, and H. Garcia-Molina. Social tag prediction. In *SIGIR*, pages 531–538, 2008.
24. P. Indyk and R. Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *STOC*, pages 604–613, 1998.
25. H. Jégou, M. Douze, and C. Schmid. Product quantization for nearest neighbor search. *IEEE Trans. Pattern Anal. Mach. Intell.*, 33(1):117–128, 2011.
26. D. S. Johnson. The np-completeness column: An ongoing guide. *J. Algorithms*, 8(3):438–448, 1987.
27. C. Körner, R. Kern, H.-P. Grahsl, and M. Strohmaier. Of categorizers and describers: an evaluation of quantitative measures for tagging motivation. In *HT*, pages 157–166, 2010.
28. H. Liang, Y. Xu, Y. Li, R. Nayak, and X. Tao. Connecting users and items with weighted tags for personalized item recommendations. In *HT*, pages 51–60, 2010.
29. K. Liu, B. Fang, and W. Zhang. Speak the same language with your friends: augmenting tag recommenders with social relations. In *HT*, pages 45–50, 2010.
30. C. Lu, X. Hu, X. Chen, J.-R. Park, T. He, and Z. Li. The topic-perspective model for social tagging systems. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '10, pages 683–692, New York, NY, USA, 2010. ACM.
31. Q. Lv, W. Josephson, Z. Wang, M. Charikar, and K. Li. Multi-probe lsh: Efficient indexing for high-dimensional similarity search. In *VLDB*, pages 950–961, 2007.
32. C. D. Manning, P. Raghavan, and H. Schtze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008.
33. M. Muja and D. G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *VISAPP (1)*, pages 331–340, 2009.
34. F. Murtagh. A Survey of Recent Advances in Hierarchical Clustering Algorithms. *The Computer Journal*, 26(4):354–359, Nov. 1983.
35. D. Ramage, S. Dumais, and D. Liebling. Characterizing microblogs with topic models. In *Proceedings of the Fourth International AAAI Conference on Weblogs and Social Media*. AAAI, 2010.
36. D. Ramage, P. Heymann, C. D. Manning, and H. Garcia-Molina. Clustering the tagged web. In *WSDM*, pages 54–63, 2009.
37. R. Ramakrishnan and B.-C. Chen. Exploratory mining in cube space. *Data Min. Knowl. Discov.*, 15(1):29–54, 2007.
38. S. S. Ravi, D. J. Rosenkrantz, and G. K. Tayi. Facility dispersion problems: Heuristics and special cases (extended abstract). In *WADS*, pages 355–366, 1991.
39. G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing and Management*, 24(5):513–523, 1988.
40. G. Sathe and S. Sarawagi. Intelligent rollups in multidimensional olap data. In *VLDB*, pages 531–540, 2001.
41. S. Sen, S. K. Lam, A. M. Rashid, D. Cosley, D. Frankowski, J. Osterhouse, F. M. Harper, and J. Riedl. tagging, communities, vocabulary, evolution. In *CSCW*, pages 181–190, 2006.
42. M. Slaney, Y. Lifshits, and J. He. Optimal parameters for locality-sensitive hashing. *Proceedings of the IEEE*, 100(9):2604–2623, 2012.
43. D. E. Vaughan, S. H. Jacobson, and H. Kaul. Analyzing the performance of simultaneous generalized hill climbing algorithms. *Comput. Optim. Appl.*, 37:103–119, 2007.
44. P. Venetis, G. Koutrika, and H. Garcia-Molina. On the selection of tags for tag clouds. In *WSDM*, pages 835–844, 2011.
45. P. Wu, Y. Sismanis, and B. Reinwald. Towards keyword-driven analytical processing. In *SIGMOD Conference*, pages 617–628, 2007.
46. C. Yu, L. Lakshmanan, and S. Amer-Yahia. It takes variety to make a world: diversification in recommender systems. In *Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology*, EDBT '09, pages 368–378, New York, NY, USA, 2009. ACM.
47. D. Zhou, J. Bian, S. Zheng, H. Zha, and C. L. Giles. Exploring social annotations for information retrieval. In *WWW*, pages 715–724, 2008.