On Incentive-based Tagging

Xuan S. Yang, Reynold Cheng, Luyi Mo, Ben Kao, David W. Cheung

Department of Computer Science, University of Hong Kong, Pokfulam Road, Hong Kong
{*xyang2, ckcheng, lymo, kao, dcheung*}*@cs.hku.hk*

***Abstract*— A social tagging system, such as del.icio.us and Flickr, allows users to annotate resources (e.g., web pages and photos) with text descriptions called *tags*. Tags have proven to be invaluable information for searching, mining, and recommending resources. In practice, however, not all resources receive the same attention from users. As a result, while some highly-popular resources are *over-tagged*, most of the resources are *under-tagged*. Incomplete tagging on resources severely affects the effectiveness of all tag-based techniques and applications. We address an interesting question: if users are *paid* to tag specific resources, how can we allocate incentives to resources in a crowd-sourcing environment so as to maximize the tagging quality of resources? We address this question by observing that the *tagging quality* of a resource becomes *stable* after it has been tagged a sufficient number of times. We formalize the concepts of tagging quality (TQ) and tagging stability (TS) in measuring the quality of a resource's tag description. We propose a theoretically optimal algorithm given a fixed "budget" (i.e., the amount of money paid for tagging resources). This solution decides the amount of rewards that should be invested on each resource in order to maximize tagging stability. We further propose a few simple, practical, and efficient incentive allocation strategies. On a dataset from del.icio.us, our best strategy provides resources with a close-to-optimal gain in tagging stability.**

I. Introduction

In recent years, there have been an increasing number of *collaborative tagging systems* on the web, such as *Flickr*, *del.icio.us*, *Bibsonomy*, *Last.fm* and *YouTube*. In these systems, users (or taggers) help describe the various resources (such as photos, webpages, songs, and video) by assigning descriptive words (tags) to them. Given the collective wisdom of a user community, tagging amasses a tremendous amount of meta-data. This tagging information serves as invaluable data from which various applications are derived. For example, tagging data can be used to train a concept space, which captures tag similarity to facilitate semantic retrieval [1], [2]. It has been shown that tag-based resource retrieval is much more effective than traditional full-text search [3]–[5]. Other applications include webpage clustering [6], keyword search [7], and query recommendation [8].

For all these tag-based applications to be effective, we need good-quality tagging of resources. Intuitively, the tags given to a resource should be relevant, succinct, and they should cover the various aspects of the resource. In practice, however, taggers are typically casual users and the tags they assign to resources are hardly perfect. The tags given by a casual tagger to a resource, called a *post*, are often noisy and incomplete [9]. A post is *noisy* if it includes tags that are typos or are irrelevant to the resource; it is *incomplete* if it describes only some of the many aspects of the resource. As an example of the latter issue, consider the URL http://earth.google.com/ (Google Earth homepage), a *del.icio.us* user might tag this resource with the post "*map*, *navigation*", another user might tag it with "*scenery*, *photos*", while a third user might tag it with "*world*, *weather*". It is easy to see that oftentimes a single post does not capture all the aspects of a resource.

We can combat the problems of noisiness and incompleteness by giving each resource a sufficient number of posts — Since typos rarely repeat, their presence in a good number of posts of a resource would be statistically insignificant. Also, given enough posts, the relative occurrence frequencies of a resource's tags reflect the relative significance of the different aspects of the resource. For example, if there are more posts with the tag "navigation" than those with the tag "weather" for the Google Earth's URL, we know that people value the navigation function of Google Earth more than its weather information function. The need to acquire enough posts for tagging resources is echoed by [10]. The observation is that the description of a resource becomes more accurate when the number of tags it receives increases. To illustrate this observation, we have collected all the del.icio.us posts assigned to the Google Earth URL in 2007[1]. Figure 1(a) shows how the relative occurrence frequencies of five selected tags (*google*, *maps*, *earth*, *software*, *travel*) change as the number of posts given to the URL increases from 0 to 500. From the figure, we see that the relative frequencies change significantly when the number of posts received are below 50; these frequencies gradually converge between 50 to 250 posts; and finally, the frequencies become very stable after about 250 posts. Based on Figure 1(a), we call 250 the *stable point*, which is the number of posts beyond which additional posts do not change the tags' relative frequencies in any practical way. Also, we call 50 the *unstable point*, which is the number of posts below which the tags' relative frequencies are not of sufficient accuracy for practical use. (For the moment, we have only given an informal and intuitive description of stable/unstable points. We will give formal definitions of these concepts later in Section III.) Note that different resources could have different stable/unstable points depending on, for example, how multidimensional their contents are. If a resource has received more posts than its stable point, we say that the resource is *over-tagged*; if the posts given to the resource is below the unstable point, we say that the resource is *under-tagged*. To give the reader an idea of

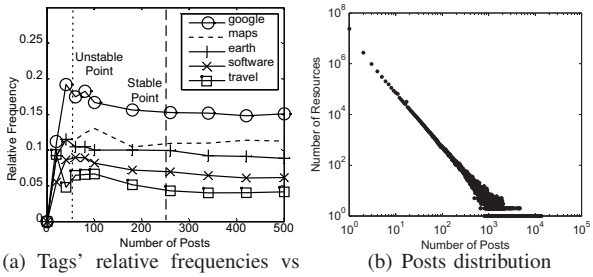
[1]We thank the authors of [11] for providing us with the dataset.

the values of stable/unstable points, we have analyzed a sample of 5,000 URLs and their posts collected by *del.icio.us*[2]. We found that the stable points of most URLs (resources) range from 50 to 200 posts with an average of 112 posts, while a typical unstable point is about 10 posts.

Naturally, we want the tagging of each resource to reach at least the resource's stable point. In that case, tagging data is of very good quality, which translates into high-quality tag-based applications. Unfortunately, in practice, it is hardly the case. We observe that in real systems, only a small portion of resources receive enough posts. This is especially true for collaborative tagging systems in which tagging is largely a volunteering task, which makes it an undermanned endeavor. Moreover, taggers are free to choose the resources that they wish to tag. Consequently, most postings are directed to a small number of highly popular resources. To illustrate, Figure 1(b) shows the distribution of the posts given to the URLs indexed by *del.icio.us* in 2007. We observe that over 10 million URLs were tagged just once, while a few URLs were tagged more than 10,000 times. The number of posts given to the URLs are seen to be very skewed.

We took a closer look of our 5,000 URLs sample and made some interesting statistical observations. First, we found that only 7% of the URLs had passed their stable points (i.e., they are over-tagged) and yet these over-tagged URLs received the lion's share of all posts. Specifically, we found that 48% of all posts were given to those URLs that had already passed their stable points. In other words, roughly half of all posts were made without practically improving the tagging quality of the resources. They are essentially wasted! At the other end of the spectrum, if we take 10 posts as the unstable point of resources (URLs), then about 25% of the URLs are under-tagged. That means about 1/4 of the URLs have poor tagging data. For tag-based applications, such as concept extraction, tag similarity analysis, topic modeling, URL clustering, etc., these under-tagged URLs cannot be reliably used and have to be discarded. We remark that although the 7% over-tagged URLs are popular among users, for many data mining tasks, they are not more significant than the under-tagged ones. Hence, losing 1/4 of the URLs in the training data could affect the quality of many tag-based applications. In fact, these under-tagged URLs can very well be salvaged. We found that if only 1% of the wasted posts (i.e., those that were given to URLs that were already over-tagged) had been channeled to those under-tagged URLs, these URLs would have passed their unstable points.

Incentive-based Tagging. To solve the above problems, we study a novel solution, whose main idea is to award a tagger some *incentive* for annotating under-tagged resources. Specifically, the taggers are informed about which (under-tagged) resources need to be tagged; if the taggers work on these resources, they are given an extra reward (e.g., some amount of money, free membership, or more disk space for sharing photos). Figure 2 illustrates this solution. Given a set of tagged resources, some *incentive allocation strategy* is first



(a) Tags' relative frequencies vs number of posts (b) Posts distribution

Fig. 1. Posts collected by *del.icio.us* in Year 2007.

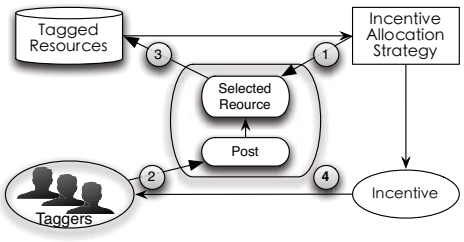


Fig. 2. An Incentive-based Tagging System.

employed to decide how much reward should be assigned to which resources (Step 1). Taggers are then invited to submit their posts to the chosen resources (Step 2), based on which the tagging data of the resources are updated (Step 3). Finally, taggers are rewarded by the system (Step 4). This model can be realized in *crowdsourcing systems* [12], e.g., the Mechanical Turk[3]. In these systems, the owner of the tagged resources creates jobs related to handling under-tagged resources. These jobs are then announced to the Internet workers ("crowds"), who can choose the resource(s) they want to tag. Upon completion of a job, the Internet worker is paid by the system.

Quality and Stability. In this paper, we address an important problem about incentive-based tagging: given that the amount of incentive (e.g., the monetary budget for rewarding taggers) is limited, which resources should we ask taggers to handle? Naturally, we should choose under-tagged resources ahead of those that have already been associated with a lot of tags. But how can we determine whether a resource is under-tagged? To answer this question, we need a metric to systematically measure the *tagging quality* of a resource in terms of the posts given to it. The way we measure tagging quality is to formalize the stability of tags' relative frequencies. We present a formula to compute the *tagging stability score* of a resource, by tracking the moving averages of tags' relative frequencies obtained from the resource's previous posts. If we consider the tags' frequencies shown in Figure 1(a), for instance, when the number of posts is 10, the stability score of the URL given by our scoring function is 0.1, which indicates that tags' frequencies are fairly unstable and more posts should be given to the URL; this score increases to 0.99 when the number of posts is increased to 400, showing that the frequencies are very stable. Hence, the higher the stability score of a resource, the better is the tagging quality of the resource.

Incentive Allocation Strategies. Based on the stability

[2]The details of this dataset is given in Section V-A

[3]http://www.mturk.com

score, we present an *optimal solution*, which gives the best allocation of incentive among the resources concerned. Intuitively, this solution gives more incentive to resources that score low in stability. The solution, which is developed by using dynamic programming techniques, runs in polynomial times. However, this optimal algorithm is an offline algorithm and it assumes that all the posts that are submitted for a resource in the future are known in advance. Hence, this algorithm is of theoretical interest only. We further develop a few practical and efficient incentive allocation strategies (c.f. Figure 2). These strategies analyze previous posts (e.g., the number of posts that have already been given to a resource so far, and their tags' frequencies) as well as the new posts submitted by taggers in order to make a reasonable decision about how incentive should be allocated. In our experiments on the *del.icio.us* dataset, the gain in stability of these strategies is close to that of the optimal solution. We have further performed two case studies on a real application, where the similarity of resources is measured. We found that the application benefited by the posts suggested by our strategies.

The rest of the paper is organized as follows. In Section II, we discuss the related works. Section III gives the problem definition, and presents the optimal incentive allocation algorithm. In Section IV we describe other practical incentive assignment strategies. Section V discusses our experimental results. We conclude in Section VI.

II. Related works

Tag data, also known as "folksonomy" [3], has recently attracted a lot of research interest. In [6], Ramage et al. studied how to use tag data to facilitate webpage clustering. Bi et al. [7] examined the problem of performing effective keyword search over tag data, with the aid of IR techniques. Guo et al. [8] used tag data to facilitate information exploration and query recommendation. Unfortunately, tag data can be noisy and ambiguous. It often contains spams [11], misspelled tags, and synoymous words (i.e., different words with the same meaning) [9], [13]. In [11], Wetzker et al. observed that a lot of resources are *under-tagged*. In this paper, we study how to improve the description quality of under-tagged resources, so that they can provide useful information to other applications.

The important issues of enhancing tag data quality has recently been addressed by a few researchers. In [14], Heymann et al. studied the use of machine learning techniques to enrich the information of resources. To tackle the synonymy problems, Marchetti et al. [9] proposed an algorithm that utilizes external knowledge, such as WordNet and Wikipedia. In [10], [13], the authors observed the phenomenon of *tagging stability* – the more posts submitted to a resource, the more stable are the relative frequencies of the tags that are associated with the resource. We propose a metric to quantify tagging stability. We also study the problem of optimizing the allocation of incentives to a set of resources in order to maximize the average tagging stability of resources. To our best knowledge, these problems have not been studied before.

Resource	Post sequence
r_1	({google, earth}, {google, geographic}, {earth}, ...)
r_2	({pictures}, {pictures}, ...)

TABLE I
POST SEQUENCES

III. Problem Definition

In this section we give a formal definition of the problem studied in this paper. We discuss the data model in Section III-A and define tagging quality of resources in Section III-B. The problem of tagging under a limited amount of incentive is given in Section III-C. We also present a theoretically optimal solution in Section III-D.

A. Data Model

Let $\mathcal{R} = \{r_1, r_2, \ldots, r_n\}$ denote a set of n resources. Let $\mathcal{T} = \{t_1, t_2, \ldots, t_m\}$ be a set of all possible tags. A tagger annotates a resource by performing a tagging operation, which is to assign tags to the resource.

Definition 1: A *post* is a nonempty set of tags assigned to a resource by a tagger in one tagging operation.

Each post is associated with a posting time at which the post is made. Assuming that no posts are made at exactly the same time instant, then the posts of a resource r_i form a sequence with respect to their posting times. We call this sequence the *post sequence* of r_i.

Definition 2: The *post sequence* of a resource r_i is the sequence $(p_i(1), p_i(2), \ldots)$, where $p_i(k)$ $(k \geq 1)$ is the k-th post received by resource r_i.

Example 1: Suppose that $\mathcal{T}$ = {google, earth, geographic, pictures} and there are two resources in $\mathcal{R}$: r_1 = (the homepage of) *Google Earth* and r_2 = *Picasa*. Table I shows example post sequences of the two resources. The first post of r_1 is $p_1(1) = \{\text{google, earth}\}$, while the first post of r_2 is $p_2(1) = \{\text{pictures}\}$.

B. Tagging Stability and Quality

Next we define relative tag frequency distribution (*rfd* for short) of a resource. After that, we define the tagging quality of a resource based on the stability of the resource's *rfd*.

We consider a resource r_i that has received $k \geq 0$ posts and a tag t.

Definition 3: The **frequency** of tag t for resource r_i, denoted by $h_i(t, k)$, is given by:

$$h_i(t, k) = |\{p_i(j) | 1 \leq j \leq k, t \in p_i(j)\}|. \quad (1)$$

Definition 4: The **relative tag frequency** of t for r_i, denoted by $f_i(t, k)$, is given by

$$f_i(t, k) = \begin{cases} \frac{h_i(t,k)}{\sum_{t' \in \mathcal{T}} h_i(t', k)} & k > 0, \\ 0 & k = 0. \end{cases} \quad (2)$$

That is, $h_i(t, k)$ is the number of posts r_i has received that contain tag t among the first k posts, and $f_i(t, k)$ is the frequency normalized by the number of tags (duplicate counted) received by r_i.

Definition 5: The **relative tag frequency distribution (*rfd*)** of a resource r_i after it has received k posts is a vector $\vec{F}_i(k)$

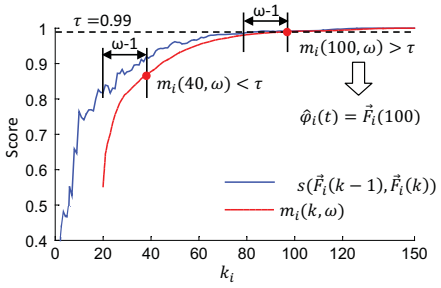


Fig. 3. *MA* score and stable *rfd* ($\omega = 20$)

such that its j-th component is the relative frequency of tag t_j for r_i, i.e., $\vec{F}_i(k)[j] = f_i(t_j, k)$.

One observation in [10] is that the *rfd* of a resource changes little after the resource has received a large number of posts. Assuming that *rfd* converges, we formalize the observation of *rfd* stability by the following:

Definition 6: The **stable *rfd*** of a resource r_i, denoted by φ_i, is given by

$$\varphi_i = \lim_{k\to\infty} \vec{F}_i(k). \tag{3}$$

In practice, we cannot compute φ_i, since we cannot get an infinitely large number of posts for any resource. We next present a practical version of stable *rfd*.

Definition 7: Given a parameter $\omega \geq 2$, the **Moving Average (*MA*) score** of a resource r_i after it has received $k \geq \omega$ posts, denoted by $m_i(k,\omega)$, is given by

$$m_i(k,\omega) = \frac{1}{\omega - 1} \sum_{j=k-\omega+2}^{k} s(\vec{F}_i(j-1), \vec{F}_i(j)), \tag{4}$$

where s is a similarity metric for quantifying the similarity of two consecutive *rfd*s' derived from the same post sequence. For convenience, we call $s(\vec{F}_i(j-1), \vec{F}_i(j))$ the adjacent similarity at the j-th post. In this paper, we use cosine-similarity [15] for s (Appendix A). Notice that if a resource has received less than ω posts, its *MA* score is not defined.

Figure 3 illustrates the idea of *MA* scores. The x-axis is the number of posts (k) given to the resource and the y-axis is the similarity score. In the figure, two lines are displayed, which show the adjacent similarity and the *MA* score as k varies. For example, to compute the *MA* score at the 40-th post, i.e. $m_i(40,\omega)$, we consider the $(40-\omega+1)$-st to the 40-th post and calculate the adjacent similarity among these ω *rfd*s'. The *MA* score is the average of $\omega - 1$ adjacent similarity scores. Intuitively, a high *MA* score indicates that these ω *rfd*s' are highly similar. We now define the concept of *practically-stable rfd* as follows:

Definition 8: Given a parameter $\omega \geq 2$ and a threshold τ, the **practically-stable *rfd*** of a resource r_i, denoted by $\hat{\varphi}_i(\omega,\tau)$, is given by

$$\hat{\varphi}_i(\omega,\tau) = \vec{F}_i(k), \tag{5}$$

where k is the smallest number such that

$$m_i(k,\omega) > \tau \text{ and } k \geq \omega. \tag{6}$$

The threshold τ in Definition 8 is used to determine whether the *MA* score is considered to be high enough. In our definition, the range of cosine-similarity score is $[0,1]$.

	google	*geographic*	*earth*	*pictures*
$\vec{F}_1(3)$	0.4	0.2	0.4	0
$\hat{\varphi}_1$	0.25	0.25	0.5	0
$\vec{F}_2(2))$	0	0	0	1
$\hat{\varphi}_2$	0.33	0	0	0.67

TABLE II

rfd'S AND STABLE *rfd*'S

So τ is set to be close to 1. The fact that the *MA* score is higher than τ indicates that the *rfd* is relatively stable. For instance, in Figure 3, if we set ω = 20 and τ = 0.99, we have $m_i(100,\omega_s) > \tau$. This indicates that the *rfd*'s from the 81-st post to the 100-th post are highly similar. So, r_i's *rfd* is very stable after 100 posts. In fact, 100 is the smallest k in this example such that $m_i(k,\omega) > \tau$. Hence, the practically stable *rfd* of the resource is $\hat{\varphi}_i = \vec{F}_i(100)$. For r_i's practically-stable *rfd* to be defined, the length of r_i's post sequence should be large enough such that there exists a k that satisfies Equation 6.

For notational convenience, we omit ω and τ when their values are understood and simply write $\hat{\varphi}_i$ instead of $\hat{\varphi}_i(\omega,\tau)$. In the following discussion, when we mention stable *rfd*, we refer to the practically-stable *rfd*, i.e. $\hat{\varphi}_i$.

Golder explains that the *rfd* of a resource can be viewed as taggers' description of the resource [10]. The fact that the *rfd* becomes stable reflects that the description is becoming more accurate. Given a resource r_i, our objective is to quantify the quality of the tags the resource has received, even though the resource may not have received enough posts to reach tagging stability. Our quality metric is defined using practically-stable *rfd* as the reference. Intuitively, the more similar the *rfd* of a resource is to the corresponding stable *rfd*, the higher is the tagging quality of the resource.

Definition 9: The **tagging quality** of a resource r_i that has received exactly k posts, denoted by $q_i(k)$, is the similarity between its *rfd* $\vec{F}_i(k)$ and r_i's stable *rfd* $\hat{\varphi}_i$, i.e.,

$$q_i(k) = s(\vec{F}_i(k), \hat{\varphi}_i). \tag{7}$$

Furthermore we define the tagging quality of a set of resources $\mathcal{R}$ as the average tagging quality of the resources in $\mathcal{R}$.

Definition 10: Let $\vec{k} = (k_1, k_2, \ldots, k_n)$ be a vector such that k_i is the number of posts resource r_i has received. The **tagging quality of** $\mathcal{R}$, denoted by $q(\mathcal{R}, \vec{k})$, is given by

$$q(\mathcal{R}, \vec{k}) = \frac{1}{n} \sum_{i=1}^{n} q_i(k_i). \tag{8}$$

Example 2: Consider the resources r_1 = *Google Earth* and r_2 = *Picasa* mentioned in Example 1. Suppose they have received k_1 = 3 and k_2 = 2 posts, respectively. If their *rfd*'s and stable *rfd*'s are those that are listed in Table II, then their tagging qualities are

$$q_1(k_1) = s(\vec{F}_1(3), \hat{\varphi}_1) = 0.953,$$
$$q_2(k_2) = s(\vec{F}_2(2), \hat{\varphi}_2) = 0.897.$$

Therefore, the quality of $\mathcal{R}$ is $q(\mathcal{R}, \vec{k}) = (0.953 + 0.897)/2 = 0.925$.

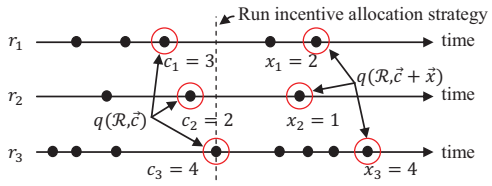


Fig. 4. Time of posts received by resources

C. Tagging Under Limited Budget

Given a limited amount of incentive (or budget), our objective is to assign the incentive to resources so as to maximize the overall quality of the resources. We use the term *post task* to refer to a vacant job of submitting a post for a specific resource. A post task can either be assigned to a tagger by the system, or, in the case of crowdsourcing, chosen by a tagger according to her wish. Upon completing a post task, the tagger is rewarded. For simplicity, we assume that every post task is given one *reward unit*. We remark that our solution can easily be extended to handle post tasks of different reward amounts. Our optimization problem is defined as follows.

Definition 11: Given a set of resources $\mathcal{R}$ such that each resource $r_i \in \mathcal{R}$ has already been tagged with c_i posts. Let $\vec{c}$ = $(c_i, \ldots, c_n)$. Given a budget of B reward units and let x_i be the number of post tasks allocated to resource r_i such that $\sum_i^n x_i = B$. The problem of **incentive-based tagging** is to return an assignment $\vec{x}$ = $(x_1, \ldots, x_n)$ such that the quality of $\mathcal{R}$, after all the B post tasks have been completed, i.e., $q(\mathcal{R}, \vec{c}+\vec{x})$, is maximized.

Figure 4 shows an illustrative example. In the figure, $\mathcal{R}$ consists of three resources, which have received 3, 2, and 4 posts, respectively ($\vec{c}$ = (3,2,4)). Given a budget B = 7 units, one possible post task assignment is $\vec{x}$ = (2,1,4). The tagging quality of $\mathcal{R}$ given this assignment is $q(\mathcal{R}, \vec{c}+\vec{x})$ or $q(\mathcal{R}, (5,3,8))$.

The incentive-based tagging problem can be reformulated as the following optimization problem $\mathcal{P}(B, \mathcal{R})$.

$$\text{given} \quad \mathcal{R}, \vec{c}, \text{ post sequence of each resource} \tag{9}$$

$$\text{maximize} \quad q(\mathcal{R}, \vec{c}+\vec{x}) \tag{10}$$

$$\text{subject to} \quad \sum_{i=1}^{n} x_i = B \tag{11}$$

$$x_i \in \mathbb{Z}^* \tag{12}$$

Equation 10 is the quality of a set of tagged resources after all post tasks are completed. Since maximizing the average quality is equivalent to maximizing the total quality, we replace Equation 10 by Equation 13 to simplify our discussions.

$$\sum_{i=1}^{n} q_i(c_i + x_i) \tag{13}$$

Table III summaries the symbols used in this paper.

Example 3: Continuing with our running example of two resources r_1 = *Google Earth* and r_2 = *Picasa*. Let $\vec{c}$ = (3,2) be the posts that the two resources have already received. From Example 2, we have $q_1(3)$ = 0.953 and $q_2(2)$ = 0.897.

Notation	Description
Data Model	
$\mathcal{R}$	A set of resources
$\mathcal{T}$	A set of tags
n	Number of resources in $\mathcal{R}$
r_i	The i-th resource in $\mathcal{R}$
t	A tag in $\mathcal{T}$
$p_i(k)$	The k-th post received by r_i
Stability and Quality	
$h_i(t,k)$, $f_i(t,k)$	Frequency and relative frequency of t for resource r_i after k posts
$\vec{F}_i(k)$	The *rfd* of resource r_i after k posts
$m_i(k,\omega)$	The *MA* score of resource r_i after k posts
φ_i	The stable *rfd* of resource r_i
$\hat{\varphi}_i(\omega,\tau)$	The practically-stable *rfd* of resource r_i
$q_i(k)$	Tagging quality of resource r_i after k posts
$q(\mathcal{R},\vec{k})$	Tagging quality of resource set $\mathcal{R}$
Incentive-based Tagging Problem	
B	Budget for the incentive-based tagging
c_i	The number of posts already received by resource r_i
x_i	The number of post tasks allocated to resource r_i
ω	The window parameter used under *MU* and *FP-MU*

TABLE III
SYMBOLS USED IN THIS PAPER

(x_1, x_2)	$q_1(c_1+x_1)$	$q_2(c_2+x_2)$	$q(\vec{c}+\vec{x})$
(0, 2)	0.953	0.992	0.973
(1, 1)	0.990	0.990	0.990
(2, 0)	0.943	0.897	0.920

TABLE IV
QUALITY OF RESOURCES

If our budget B = 2, then there are only three possible post task assignments, namely, $\vec{x}$ = (2,0), (1,1), and (0,2). Suppose that the next 2 posts r_1 will get are {geographic, earth} and {google, geographic}, while r_2 will get {google, picture} and {google}, the tagging quality of the resources under the three possible assignments are shown in Table IV. From the table, we see that the assignment $\vec{x}$ = (1,1) results in the highest tagging quality. It is thus the optimal assignment.

D. An Optimal Solution

In this section we present a dynamic programming solution, named *DP*, which gives us a theoretically optimal solution to the incentive-based tagging problem.

This solution assumes that all posts a resource will receive as well as the stable *rfd* of each resource are known. Hence, the value $q_i(c_i + x_i)$ can be computed for each resource r_i and for each value $x_i \leq B$. As a result, the theoretically optimal solution for the optimization problem $\mathcal{P}(B, \mathcal{R})$ can be found by dynamic programming techniques. Specifically, let $Q(B, n)$ be the optimal value to the problem $\mathcal{P}(B, \mathcal{R})$ and let $Q(b, l)$ $(1 \leq b \leq B, 1 \leq l \leq n)$ be the optimal solution to the subproblem $\mathcal{P}(b, \{r_1, r_2, \ldots, r_l\})$. We use the following recurrence equation to computer $Q(B, n)$:

$$Q(b,l) = \begin{cases} q_1(c_1+b) & l = 1, \\ \max_{0 \leq x_l \leq b} Q(b-x_l, l-1) + q_l(c_l+x_l) & l > 1. \end{cases} \tag{14}$$

The idea is that to find the optimal solution for the problem $\mathcal{P}(b, \{r_1, r_2, \ldots, r_l\})$, we enumerate all possible values for x_l, which ranges from 0 post tasks to a maximum of b post tasks. The value x_l^* that maximizes $Q(b-x_l, l-1)+q_l(c_l+x_l)$ is the number of post tasks that should be assigned to resource r_l in

an optimal solution. A correctness proof of the DP algorithm is given in Appendix B. The time and space complexities of DP are $O(n|\mathcal{T}|B^2)$ and $O(nB + n|\mathcal{T}|)$, respectively.

Although *DP* generates an optimal solution, it cannot be applied in practice. The reason is that the stable *rfd*'s $\hat{\varphi}_i$ are generally not know, and hence the quality $q_i(c_i + x_i)$ cannot be computed. Nonetheless, *DP* is of theoretical interest because it serves as an optimal reference against which our practical solutions are compared. In Section IV, we present some practical incentive allocation strategies.

IV. Incentive Allocation Strategies

We now propose five practical incentive allocation strategies. They do not require that the resources' stable *rfd*'s, $\hat{\varphi}_i$, be known, nor do they need any posts that have not yet been received. These strategies follow the framework shown in Algorithm 1. The main idea of this framework is to invest one unit of budget at a time, and to allocate one post task to a chosen resource based on some simple strategies.

Algorithm 1 Strategy Framework

Require: Budget B, Resources $\mathcal{R}$, Initial no. of posts $\vec{c}$
1: **for** $i \leftarrow 1$ to n **do** $x[i] \leftarrow 0$
2: INIT()
3: **while** $B > 0$ **do**
4: $i_0 \leftarrow$ CHOOSE()
5: Present r_{i_0} for a tagger to tag
6: The tagger completes a post task on r_{i_0}
7: UPDATE()
8: $x[i_0] \leftarrow x[i_0] + 1$, $B \leftarrow B - 1$
return $\vec{x}$

In the algorithm, arrays x and c are used to represent $\vec{x}$ and $\vec{c}$, respectively. The value of each x_i is initialized to 0 (Step 1). INIT() is used to initialize some global variables that are customized for each strategy. In each iteration of the while-loop, a resource r_{i_0} is selected by CHOOSE(). Then a new post task for r_{i_0} is presented to a tagger (Step 5). After the post task is completed (Step 6), UPDATE() is called to update the global variables. x_{i_0} and B are also updated accordingly. This process repeats until the budget is exhausted. The various allocation strategies differ in the implementation of CHOOSE(), which is used to choose the next resource to be presented to the tagger.

Let T_{I}, T_{C} and T_{U} be the time complexity of INIT(), CHOOSE(), and UPDATE(), respectively. The total time complexity of Algorithm 1 is $O(n + T_{\mathrm{I}} + B(T_{\mathrm{C}} + T_{\mathrm{U}}))$. We will further elaborate on the time and space complexities of the various allocation strategies at the end of this section. We next describe these strategies.

A. The Free Choice Strategy (FC)

The Free Choice strategy (*FC*) allows taggers to freely decide which resource they want to tag. The function CHOOSE() simply returns the resource a tagger picks. This strategy is a baseline solution which follows the practice of existing collaborative tagging systems.

B. The Round Robin Strategy (RR)

The Round Robin strategy (*RR*) chooses a resource in a round-robin fashion, regardless of how many posts they have received or how stable their *rfd*s are. *RR* does not require much information to be maintained by the system and is very simple to implement. Resources get roughly the same number of post tasks. Algorithm 2 shows the INIT(), CHOOSE() and UPDATE() functions of *RR*. The global variable l stores the id of the the last chosen resource.

Algorithm 2 Round Robin (RR)

1: **procedure** INIT()
2: $l \leftarrow 1$
3: **function** CHOOSE()
4: **return** (l mod n) + 1
5: **procedure** UPDATE()
6: $l \leftarrow l + 1$

C. The Fewest Posts First Strategy (FP)

In general, if a resource has received only a small number of posts, then the quality improvement it gets by giving it an additional post is much more significant than the case if the resource has already been given many posts. To illustrate, Figure 5 shows how the tagging quality (in terms of *MA* scores) of two resources, r_i and r_j, change as the number of posts they receive increases. Suppose r_i and r_j have so far received 10 and 50 posts, respectively, and that we have a budget B of 10 post tasks, the figure shows that allocating these 10 post tasks to r_i results in a much greater quality improvement than if the post tasks are allocated to r_j. The Fewest Posts First strategy (*FP*) follows this observation and chooses the resource that has received the fewest posts to be allocated the next post task.

Algorithm 3 Fewest Posts First (FP)

1: **procedure** INIT()
2: **for** $i \leftarrow 1$ to n **do**
3: Q.push($(c[i], i)$) // Q is a priority queue
4: **function** CHOOSE()
5: $(cx_{i_0}, i_0) \leftarrow$ Q.pop()
6: **return** i_0
7: **procedure** UPDATE()
8: Q.push($(c[i_0] + x[i_0], i_0)$)

Algorithm 3 shows the details of *FP*. A priority queue is used to maintain the set of resources and they are ordered by the total number of posts they have received. *FP* always chooses the resource r_i with the smallest $c_i + x_i$ value to be allocated the next post task. A potential weakness of *FP* is that it does not consider the different properties of the resources in its post task selection. For example, a webpage (URL) may carry complex and rich contents, which require many posts to adequately describe them. In this case, the URL needs many posts to go pass its unstable point. On the other hand, another webpage may focus on one single topic, which only needs a handful of posts to well describe it. In this case, the tagging quality improvement brought about by a post task could be more significant if it is given to the complex webpage rather

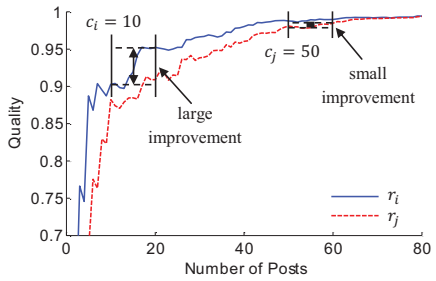


Fig. 5. Quality vs Number of Posts

than to the simple webpage, even if the former has already received more posts than the latter.

D. The Most Unstable First Strategy (MU)

As mentioned in Section III-B, the *MA* score of a resource measures the stability of the resource's *rfd* and thus the resource's tagging quality. The Most Unstable First strategy (*MU*) chooses the resource with the smallest *MA* score because presumably that is the resource which requires quality improving the most. Algorithm 4 gives the details of *MU*. Since the *MA* score is defined over a window of ω posts, resources that have not received at least ω posts are ignored by *MU*. (This issue is addressed by our next strategy.) Similar to *FP*, a priority queue is used under *MU*. Specifically, resources are ordered by their *MA* scores, $m_i(c[i] + x[i], \omega)$.

Algorithm 4 Most Unstable First (MU)

```
1: procedure INIT()
2:     for i ← 1 to n do
3:         if resource r_i has received at least ω posts then
4:             Q.push((m_i(c[i], ω), i))        // Q is a priority queue
5: function CHOOSE()
6:     (ma_{i_0}, i_0) ← Q.pop()
7:     return i_0
8: procedure UPDATE()
9:     Q.push((m_{i_0}(c[i_0] + x[i_0], ω), i_0))
```

To compute $m_i(c_i + x_i, \omega)$, $O(|\mathcal{T}|)$ space and $O(\omega|\mathcal{T}|)$ time are required if the *rfd*s' are stored as vectors. However, in practice, the number of distinct tags associated with a particular resource is usually very small compared with $|\mathcal{T}|$. We can reduce the space and time complexities by using a self-balancing binary search tree (eg. map in *C++* and *Java*) to store the non-zero tag frequency information. This makes *MU* a practical strategy even for large datasets.

There are two issues in applying *MU*. First, we need to set a value for the parameter ω, which is the window size for computing the *MA* score. We will discuss this issue in Section V. The second issue is that resources that have received less than ω posts are ignored by *MU*. This is undesirable because these resources are largely under-tagged and are in need of post tasks to improve their tagging quality.

E. The Hybrid Strategy (FP-MU)

If there are resources that have not received at least ω posts, the *FP-MU* strategy uses *FP* to allocate post tasks to these resources. We call this the warm-up stage of *FP-MU*. Once all resources have received at least ω posts, *FP-MU* switches to the *MU* strategy because the *MA* scores of all resources are now available.

TABLE V
TIME AND SPACE COMPLEXITIES

Strategy	Time	Space
DP	$O(n\|\mathcal{T}\|B^2)$	$O(nB + n\|\mathcal{T}\|)$
FC	$O(n + B)$	$O(n)$
RR	$O(n + B)$	$O(n)$
FP	$O((n + B)\log n)$	$O(n)$
MU	$O((n + B)\log n + (n\omega + B)\|\mathcal{T}\|)$	$O(n\omega + n\|\mathcal{T}\|)$
FP-MU	$O((n + B)\log n + (n\omega + B)\|\mathcal{T}\|)$	$O(n\omega + n\|\mathcal{T}\|)$

Algorithm 5 Hybrid Strategy (FP-MU)

Require: Budget B, Resources $\mathcal{R}$, Initial no. of posts $\vec{c}$
1: $b \leftarrow 0$
2: **for** $i \leftarrow 1$ to n **do** $b \leftarrow b + \max(0, \omega - c[i])$
3: $b \leftarrow \min(B, b)$
4: $\vec{x} \leftarrow \text{FP}(b, \mathcal{R}, \vec{c})$
5: **return** $\vec{x}$ + $\text{MU}(B - b, \mathcal{R}, \vec{c} + \vec{x})$

Algorithm 5 shows the details of the CHOOSE() function of *FP-MU*. Steps 1 and 2 calculate the total amount of reward units that are needed to bring all resources to at least ω posts. Step 3 ensures it does not exceed the given budget. Then the *FP* and the *MU* strategies are employed in that order. Note that a larger ω implies more budget is spent in the warm-up stage, and so *FP-MU* behaves more like *FP*. We will further discuss how ω should be set in Section V.

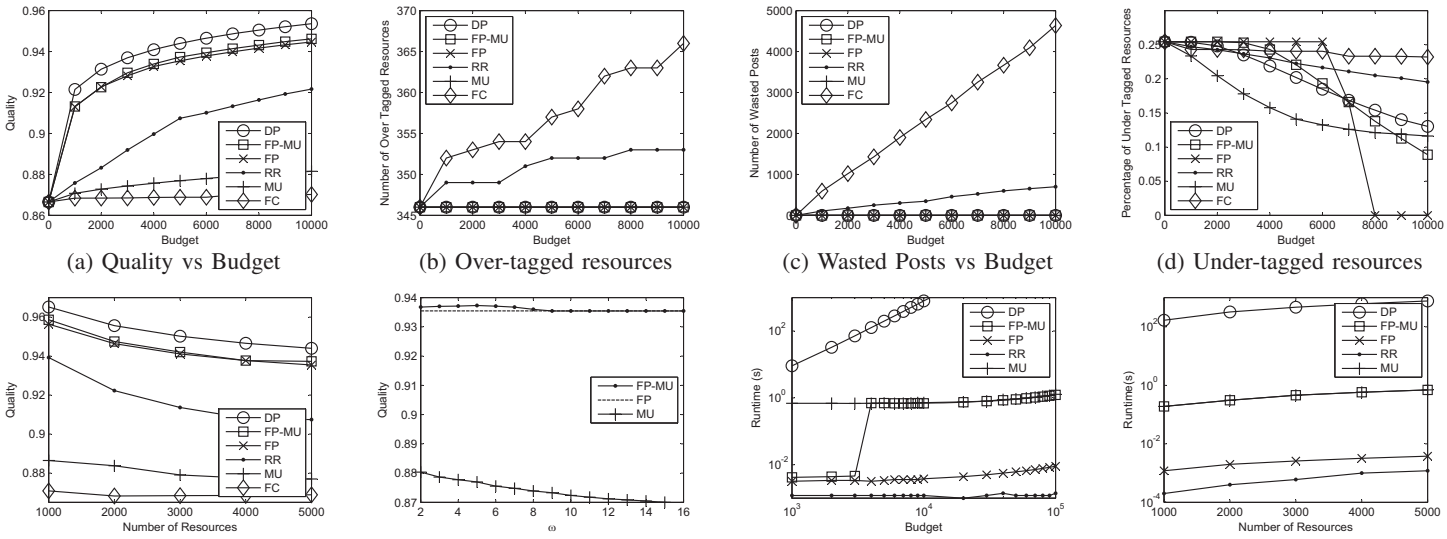
Table V summarizes the time and space complexities of all the proposed strategies. Their derivation can be found in Appendix C. With appropriate data structures (e.g. priority queue, self-balancing binary search tree), these strategies are efficient in both time and space.

V. EXPERIMENTS

In this section we present our experiment results evaluating the various incentive allocation strategies. We first describe the experiment setup in Section V-A. Then we evaluate the strategies in Section V-B. In Section V-C we illustrate the performance of our strategies using two case studies.

A. Experimental Setup

We use the dataset provided by the authors of [11]. This dataset contains all posts collected by *del.icio.us* during the whole year of 2007 for tagging webpages (URLs), which we regard as resources. Ideally, the tagging system (*del.icio.us* in our case) should run long enough so that the post sequence of each resource contains enough posts to cover the resource's stable point. However, since our dataset only contains one-year worth of posts, the post sequences of many resources are way shorter than what the ideal case requires. In order to (1) measure the tagging quality of the various strategies with reference to the theoretical stable *rfd*'s and to (2) compare the strategies against the theoretical optimal strategy *DP*, we focus only on the resources in the dataset that have reached their stable points with the posts they received in Year 2007.



(a) Quality vs Budget (b) Over-tagged resources (c) Wasted Posts vs Budget (d) Under-tagged resources

(e) Quality vs # of Resources (f) Effect of ω (g) Runtime vs Budget (h) Runtime vs # of Resources

Fig. 6. Experiment Results

Specifically, we use relatively large values of $\omega_s = 20$ and $\tau_s = 0.9999$ as the values of ω and τ (See Definition 8, Equation 6) to check if the post sequence of a resource derives a stable *rfd*. We identified 5,000 resources from the *del.icio.us* dataset that satisfy the above test. Our experiments were then conducted on these 5,000 resources (and their post sequences). Note that the parameters ω_s and τ_s were used only to prepare our dataset. They represent very stringent requirements of stability. Strategies *MU* and *FP-MU* require the parameter ω too. As we will see later in our experimental results, *MU* and *FP-MU* can use a much smaller ω and yet they are able to achieve very good performance[4].

There were a total of $562,048$ posts given to the 5,000 selected resources in Year 2007, or 112 posts for each on average. In our model, we assume that resources are given some initial numbers of posts (i.e., $\vec{c}$) before an incentive allocation strategy is run. In our experiment, we take all the posts given in January 2007 as those initial posts. These January posts consist of $148,471$ posts, or 29.7 posts per resource on average. These posts are very unevenly distributed to the resources: while some resources have more than 150 posts, over 1000 of them have 10 posts or less. So, many resources are under-tagged. These posts are accessible by all strategies for making incentive allocation decision.

Any post given on or after Feb 1st is considered a result of a post task assigned by some strategy. For example, consider a resource r_i that has 50 posts between Feb 1st and Dec 31st. If a strategy A decides to allocate three reward units to r_i, then the first three posts of r_i given after Feb 1st are treated as the result of the three post tasks offered by A. The tagging quality of r_i under A is then measured by the post sequence of r_i up to the last one of these three posts. For the theoretically optimal solution *DP*, it can access the complete Year 2007 post sequences of all resources to make its allocation decision. (The post tasks *DP* offers is of course limited by the budget B.) For other strategies, they only have the information about the post sequence up to January 31st, as well as the result of any post tasks they offer. The default budget is $5,000$ reward units, which is 3.4% of the number of posts given in January 2007. For *MU* and *FP-MU*, the default value of ω is 5.

All our experiments were written in C++ and were executed on a machine with 8G memory, i5 CPU, 64-bit linux system.

B. Incentive Allocation Strategies

1. Tagging Quality. Figure 6(a) shows the tagging quality of the resources under different strategies as the budget B varies from 0 to 10,000 units. The tagging quality of resources is 0.865 on January 31st, before any post tasks are done. From the figure, we see that *FC*, which allows taggers free choices of resources to tag, does not improve the tagging quality of the resources much even under a very big budget. For example, when B = 10,000, the tagging quality is only increased by a mere 0.4% compared with the January 31st quality value. The theoretically optimal solution *DP*, on the other hand, improves the quality by 9.1% Interestingly, *FP* and *FP-MU* perform very well. The tagging quality under them are very close to that under *DP*. Between the two, *FP-MU* edges over *FP* when $B >$ 3,000. When $B \leq$ 3000, *FP-MU* operates as *FP* because some of the resources have not received the minimum ω = 5 posts for *FP-MU* to switch to *MU*. We also note that even with a very small budget, such as B = 1,000, *FP* and *FP-MU* both give sharp improvements in the resources' tagging quality. We also observe that *MU* does not improve tagging quality significantly. This is because it ignores those resources that have received less than 5 posts. These resources are heavily under-tagged and are the resources whose tagging quality allows much rooms for improvements. Finally, *RR* shows intermediate performance, since it is not paying particular attention to the under-tagged resources, nor is it focusing on those that have already been over-tagged.

We have extended the budget B to much bigger values until all 5,000 resources' *rfd*'s are practically stable. We found

[4]We have repeated our experiments with other values of (ω_s, τ_s), such as (40, 0.99999), to prepare different datasets. Since the results of these different datasets led to similar conclusions, we omit the discussions on these datasets in this paper.

that *FC* requires more than two million post tasks to achieve stability while *FP* and *FP-MU* require only about $200,000$, which is 90% less than what *FC* needs. This experiment shows that a good incentive allocation strategy can significantly reduce the budget requirement for achieving tagging stability.

2. Over-tagging and Wasted Posts. The reason why *FC* fails to improve tagging quality is that most taggers focus only on popular resources (in our experiment, popular webpages). Since these popular resources are already over-tagged, a significant portion of post tasks offered by *FC* are essentially wasted (i.e., these post tasks are on already over-tagged resources). To illustrate, Figures 6(b) and 6(c) show the number of *over-tagged resources* and the number of *wasted post tasks* under various strategies when the budget increases. We see that the number of over-tagged resources increases under *FC* and *RR*, while there are no increases under the other strategies which focus on helping either under-tagged or low-tagging-quality resources. We also see that *FC* wasted much post tasks, while the other strategies except *RR* did not waste any post tasks on over-tagged resources. In fact, *FC* wasted about 48% or nearly half of its post tasks. Again, this shows that a good allocation strategy can significantly save valuable tagging budget.

3. Under-tagging. Instead of wasting post tasks on over-tagged resources, a good allocation strategy should also assign post tasks to those under-tagged resources to help them pass their unstable points. From our experiment, we see that for most resources, their *rfd*'s are not stable if each is given less than 10 posts. (In particular, the adjacent similarity of those *rfd*'s are typically less than 0.95.) If we consider a resource to be under-tagged if it has received not more than 10 posts, then Figure 6(d) shows the percentage of resources that are under-tagged after a certain budget B of post tasks have been performed under the various strategies. Intuitively, a good strategy should bring this number down as much as possible, so that more resources' tagging information are of reliably good quality, thus improving all tag-based applications.

From the figure, we see that about 25% of the resources are initially under-tagged. *FC* is the worst in reducing this percentage. This is because most taggers ignore the less-popular resources, leaving them severely under-tagged. Since *RR* does not focus on under-tagged resources either, its performance is only marginally better than *FC*. The performance of *FP* is interesting. It doesn't reduce the under-tagged percentage initially, but the percentage drops sharply when B is 6,000 to 7,000. This is because *FP* assigns post tasks to those that have received the fewest posts first. Once *FP* has brought the post counts of all under-tagged resources to 9 posts, any additional post tasks will carry these under-tagged resources over the 10-post threshold. This explains the sharp drop in the under-tagged resource percentage. The performance of *MU* is also very good, particularly when the budget is relatively small (e.g., when $B <$ 7,000). Since *MU* focuses on those resources with the most unstable *rfd's*, it tends to help the under-tagged resources first. For *DP*, the percentage drops gradually and not as sharply as *MU* because it is not optimized for reducing the number of under-tagged resources. The performance of *FP-MU* is seen to be between those of *FP* and *MU*.

4. Effect of Number of Resources. From the 5,000 resources, we randomly pick smaller subsets to evaluate the effect of the dataset size. Figure 6(e) shows the effect of the number of resources on tagging quality. We observe that the tagging quality decreases when the number of resources increases. This is because with a fixed budget, when the number of resources becomes larger, the number of post tasks received by each resource is less. We again observe that *FP* and *FP-MU* come closest to *DP* in terms of tagging quality.

5. Effect of ω. Both *MU* and *FP-MU* use ω to control the moving average window size. Figure 6(f) shows the effect of ω on *MU* and *FP-MU*. We also show the tagging quality of *FP* as a reference. Under *MU*, the tagging quality drops with ω. The reason is that the larger ω is, the more under-tagged resources *MU* ignores. This lowers the tagging quality. For *FP-MU*, a larger ω means a longer warm-up stage, during which *FP-MU* operates as *FP*. From the figure, we see that when $\omega > 8$, the warm-up stage never finishes and so *FP-MU* performs the same as *FP*. When $\omega \leq 8$, *FP-MU* switches to *MU* once its warm-up stage is done. This gives *FP-MU* slightly better performance than the pure *FP* strategy.

6. Efficiency. We test the computational time required by each strategy. Since the running time of *FC* is almost the same as that of *RR* we only show *RR* here. Figure 6(g) shows the effect of budget on the performance of these strategies. We can see that *DP* does not scale well; it takes more than $3,000$ seconds to calculate the optimal number of post tasks, when the budget is $10,000$ reward units. On the other hand, other strategies scale well with the budget. In these experiments, *RR* runs the fastest, while *FP* runs a little bit slower because it has to maintain a priority queue. Since *MU* and *FP-MU* have to calculate the similarity between *rfd*'s and *MA* scores, their running times are much longer than that of *FP*. For *FP-MU*, when the budget is smaller than $3,000$, all post tasks are allocated by *FP*, and so it is fast; when the budget becomes larger, *MU* can be used to allocate post tasks, therefore, its running time is nearly the same as *MU*'s. Figure 6(h) shows the effect of the number of resources on the performance of strategies. We can see that all our strategies scale well and runs significantly faster than *DP*.

Summary. The effectiveness of *FP-MU* is the closest to that of *DP*; however, it is not very efficient. On the other hand, *FP* is almost as good as *FP-MU*. Besides, it is more efficient than *FP-MU* and it is also easy to implement. Moreover, *FP* can be run offline, and does not need the information of the newest post tasks in order to determine the allocation of incentives. Based on these experimental results, we recommend *FP* to be used.

C. Case Study: Similarity Measurement

So far, we have compared the various allocation strategies in terms of the tagging quality of resources. In this section we illustrate how the improvement in tagging quality translates into the improvement of tag-based applications. A fundamental operation of many IR techniques is to measure the similarity

Rank	Jan. 31	FC (B = 10,000)	FP (B = 10,000)	Dec. 31
1	javaranch.com	physicsclassroom.com	physicsclassroom.com	physicsclassroom.com
2	onjava.com	hyperphysics.phy-astr.gsu.edu/hbase/hph.html	practicalphysics.org	practicalphysics.org
3	java.sun.com	practicalphysics.org	hyperphysics.phy-astr.gsu.edu/hbase/hframe.html	hyperphysics.phy-astr.gsu.edu/hbase/hframe.html
4	jguru.com	hyperphysics.phy-astr.gsu.edu/hbase/hframe.html	hyperphysics.phy-astr.gsu.edu/hbase/hph.html	hyperphysics.phy-astr.gsu.edu/hbase/hph.html
5	bluej.org	javaranch.com	physicsforums.com	physicsforums.com
6	kickjava.com	robocode.sourceforge.net	upscale.utoronto.ca/GeneralInterest/Harrison/Flash	upscale.utoronto.ca/GeneralInterest/Harrison/Flash
7	java.sun.com/docs/books/tutorial	www.onjava.com	bethe.cornell.edu	micro.magnet.fsu.edu/primer/java/scienceopticsu/powersof10
8	java.sun.com/docs/codeconv	java.sun.com	grc.nasa.gov/WWW/K-12/Numbers/Math/Mathematical_Thinking/index.htm	bethe.cornell.edu
9	checkstyle.sourceforge.net	jguru.com	micro.magnet.fsu.edu/primer/java/scienceopticsu/powersof10	grc.nasa.gov/WWW/K-12/Numbers/Math/Mathematical_Thinking/index.htm
10	developer.com/java	kickjava.com	sodaplay.com	madsci.org

TABLE VI

TOP-10 RESULTS OF WWW.MYPHYSICSLAB.COM

URL	Description	Jan 31	FC (B = 10,000)	FP (B = 10,000)	Dec 31
dvdvideosoft.com	video editing	10 video sharing	6 video editing 2 video sharing 2 other	8 video editing 2 video sharing	9 video editing 1 video sharing
slashup.com	photo editing	2 photo editing 4 photo sharing 4 other	3 photo editing 1 photo sharing 5 other	4 photo editing 6 photo sharing	4 photo editing 6 photo sharing
bdonline.co.uk	news on architecture	3 architecture 7 news	8 architecture 2 news	9 architecture 1 news	9 architecture 1 news
espn.go.com	sports	10 sports	10 sports	10 sports	10 sports

TABLE VII

SOME MORE TOP-10 RESULT COMPARISONS

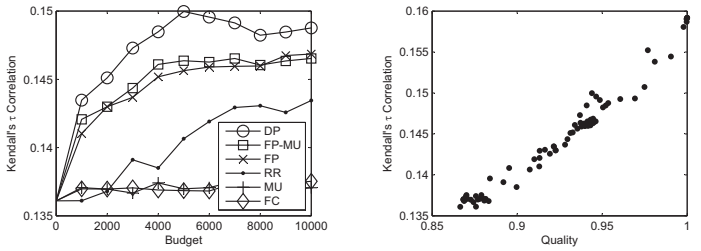
of resources [16]. Such a resource-resource similarity measure is used in many applications such as searching [7], [17], recommendation [8], and clustering [6]. Given the tagging information of resources, one popular method to measure resources' similarity is to compute the cosine similarity of resources' *rfd*'s [6], [16]. We discuss two case studies and show how a good allocation strategy improves the quality of such a similarity measure.

1. Top-10 Similar Resources. In our first case study, we pick a subject webpage, say r_*, indexed by *del.icio.us* and determine r_*'s *rfd*, $\vec{F}_*$. All other webpages *rfd*'s are then compared with $\vec{F}_*$ using cosine similarity. The top-10 most similar webpages are so determined. The result of this top-10 search for the subject webpage www.myphysicslab.com is shown in Table VI. The first column (Rank) shows the ranks of the top-10 similar webpages according to the cosine similarity scores. The 2nd column (Jan 31) shows the top-10 results if the webpages' *rfd*'s are derived from the posts obtained up to Jan 31. In other words, these are the initial *rfd*'s before any allocation strategies are applied. The 5th column (Dec 31) shows the top-10 results if all the posts collected in 2007 are used to derive webpages' *rfd*'s. This is thus the *ideal* top-10 result if all tagging information is used. The 3rd column (FC) and the 4th column (FP) show the result if a budget of 10,000 post tasks are offered under *FC* and *FP*, respectively, to derive the *rfd*'s.

The subject webpage provides simulations of physics experiments. These simulations are written in Java. From the "Jan 31" column, we see that all of the top-10 webpages are java-related. This is because most of the initial posts given to the subject webpage focus on the java implementation of its simulations programs. This is obviously poor top-10 result because the main theme of the subject webpage is about physics, not the Java programming language. From the "Dec 31" column, we see that all the top-10 webpages are about physics, which is a perfect. This shows that by giving webpages enough posts, the tags accurately describe the webpages. The results of *FC* and *FP* given the same budget (10,000) differ significantly. For *FC*, only 4 out of its top-10 webpages are about physics. The other 6 are on Java. On the other hand, the top-10 list of *FP* is almost perfect. It gets 9 out of the top-10 webpages given by the ideal case (Dec 31). Moreover, the rankings of the two lists are almost the same.

We have repeated the same experiment using other subject webpages and in most cases *FP* outperforms *FC* significantly and *FP*'s top-10 lists are very similar to those of the ideal scenario. Table VII shows a few more examples. For example, the webpage dvdvideosoft.com is a website on video editing software. However, all top-10 webpages obtained using the *rfd*'s derived by the initial (Jan 31) posts are video sharing sites. The ideal scenario (Dec 31), on the other hand, gives 9 video editing sites and 1 video sharing site. This combination is matched very closely by *FP*, which gives 8 video editing sites and 2 video sharing sites. This result is much better than that of *FC*, as shown in the table. Finally, we remark that for the webpage espn.go.com, all of the 4 cases give the same (perfect) results. This is because espn is a very popular resource, and for those, *FC* has no problems in describing them with enough posts and tags.

2. Accuracy of Similarity Ranking. Having discussed how our strategy can improve the quality of top-k queries, now let's see how they can help the overall accuracy of resource-resource similarity measurements. As pointed out by [16], different similarity measures have different distributional properties. Besides it is often difficult to quantify the similarity between two resources. On the other hand, to rank



(a) Accuracy vs Budget (b) Accuracy vs Tagging Quality

Fig. 7. Overall Accuracy of Resource-Resource Similarity

the resource pairs based on their similarities is more intuitive. So we follow their framework to evaluate the overall resource-resource similarity derived from their *rfd's*. Specifically, given a set of resources $\mathcal{R}$, it first ranks all pairs of resources by their cosine similarity. It then compares the ranking to a ground truth with *Kendall's* τ correlation coefficient, which is a correlation measurement between two rankings, with value ranging from -1 to 1, where -1 means two rankings are totally opposite, and 1 means they are exactly the same. The ground truth is generated from the Open Directory Project[5], where resources from $\mathcal{R}$ are hierarchically categorized. For a particular resource pair, it measures the similarity based on their distance in the hierarchy; the smaller the distance, the higher is their similarity.

Figure 7(a) shows how our strategies can help to improve the overall accuracy of similarity measurement among the resources. For instance, with $5,000$ more posts assigned by *FP-MU* and *FP*, the overall accuracy of similarity measurement can be improved by 7.6% and 7.1% respectively, which are much more significant than we can get from *FC*.

Besides, an even more interesting discovery is that Figure 7(a) is highly similar to Fig 6(a), which confirms the intuition that if the *tagging quality* of the set of resources $\mathcal{R}$ is high, the overall accuracy of similarity measurement (the *Kendall*'s τ score) should also be high. To verify this, we plot Figure 7(b) in which each point (x_i, y_i) corresponds to a post task assignment. Specifically, x_i is the tagging quality of $\mathcal{R}$ after all the post tasks are completed; and y_i is the accuracy of similarity measurement derived from their posts. From this figure we can observe a strong correlation between the similarity accuracy and the tagging quality. By denoting the *tagging qualities* and overall similarity accuracies with $\vec{x}$ and $\vec{y}$ we calculate their correlation with Equation 15,

$$corr(\vec{x}, \vec{y}) = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{(n-1)s_x s_y} \tag{15}$$

where $\bar{x}$ and s_x ($\bar{y}$ and s_y) are the mean and standard value of $\vec{x}$ ($\vec{y}$) respectively [18]. The correlation between *tagging qualities* and overall similarity accuracies is over 98%, meaning that our tagging quality metric can well indicate the result quality of similarity related IR tasks performed on the tagged resources.

[5]http://www.dmoz.org/

ACKNOWLEDGMENTS

Reynold Cheng, Xuan Yang, and Luyi Mo were supported by the Research Grants Council of Hong Kong (GRF Project 711309E). We would like to thank the anonymous reviewers for their insightful comments.

VI. CONCLUSIONS

In this paper, we study how to improve the quality of tag data, in order to facilitate applications that make use of these data. We propose a metric to measure the quality of tag descriptions, based on the stability of their relative frequencies. We also examine how to improve their quality under a limited amount of incentive. We study an optimal solution, and also present a number of practical and efficient incentive allocation strategies. Our experiments show that the *FP* strategy is highly effective and efficient. In the future, we will study how our solution can handle post tasks with different costs, and also how user preference should be considered in the allocation process. We will also develop a system prototype and a user interface to support incentive-based tagging.

REFERENCES

[1] H. S. Al-Khalifa and H. C. Davis, "Exploring the value of folksonomies for creating semantic metadata," *International Journal on Semantic Web and Information Systems*, 2007.
[2] P. Heymann and H. Garcia-Molina, "Collaborative creation of communal hierarchical taxonomies in social tagging systems," *Technical Report*, 2006.
[3] A. Hotho, R. Jaschke, C. Schmitz, and G. Stumme, "Information retrieval in folksonomies:search and ranking," in *ESWC European Semantic Web Conference*, 2006.
[4] S. Bao, X. Wu, B. Fei, G. Xue, Z. Su, and Y. Yu, "Optimizing web search using social annotations," in *WWW*, 2007.
[5] P. Heymann, G. Koutrika, and H. Garcia-Molina, "Can social bookmarking improve web search?" in *WSDM*, 2008.
[6] D. Ramage, P. Heymann, C. D. Manning, and H. Garcia-Molina, "Clustering the tagged web," in *WSDM, ACM International Conference on Web Search and Data Mining*, 2009.
[7] B. Bi, S. Lee, B. Kao, and R. Cheng, "CubeLSI: An Effective and Efficient Method For Searching Results in Social Tagging Systems," in *ICDE*, 2011.
[8] J. Guo, X. Cheng, G. Xu, and H.-W. Shen, "Structured approach to query recommendation with social annotation data," in *CIKM*, 2010.
[9] A. Marchetti, M. Tesconi, F. Ronzano, M. Rosella, and S. Minutoli, "Semkey: A semantic collaborative tagging system," in *WWW*, 2007.
[10] S. Golder and B. Huberman, "Usage patterns of collaborative tagging systems," *Journal of information science*, vol. 32, no. 2, pp. 198–208, 2006.
[11] R. Wetzker, C. Zimmermann, and C. Bauckhage, "Analyzing social bookmarking systems: A del.icio.us cookbook," in *ECAI Mining Social Data Workshop*, 2008.
[12] P. Heymann, A. Paepcke, and H. Garcia-Molina, "Tagging human knowledge," in *WSDM*, 2010.
[13] H. Halpin, V. Robu, and H. Shepherd, "The complex dynamics of collaborative tagging," in *WWW*, 2007.
[14] P. Heymann, D. Ramage, and H. Garcia-Molina, "Social tag prediction," in *SIGIR*, 2008.
[15] J. Han, M. Kamber, and J. Pei, *Data mining: concepts and techniques*. Morgan Kaufmann, 2011.
[16] B. Markines, C. Cattuto, F. Menczer, D. Benz, A. Hotho, and G. Stumme, "Evaluating similarity measures for emergent semantics of social tagging," in *WWW*, 2009.
[17] K. Bischoff, C. S. Firan, W. Nejdl, and R. Paiu, "Can all tags be used for search," in *CIKM*, 2008.
[18] S. Dowdy, S. Wearden, and D. Chilko, *Statistics for Research*. Wiley, 1983.

APPENDIX

A. Cosine-similarity

Given two *rfd's* $\vec{F}_i(k_i)$ and $\vec{F}_j(k_j)$, cosine-similarity is given by: $s(\vec{F}_i(k_i), \vec{F}_j(k_j))$

$$\begin{cases} \frac{\sum_{t_l \in \mathcal{T}} (\vec{F}_i(k_i)[l] \cdot \vec{F}_j(k_j)[l])}{\sqrt{\sum_{t_l \in \mathcal{T}} \vec{F}_i(k_i)[l]^2} \sqrt{\sum_{t_l \in \mathcal{T}} \vec{F}_j(k_j)[l]^2}} & k_i > 0 \text{ and } k_j > 0, \\ 0 & \text{otherwise.} \end{cases} \tag{16}$$

B. A Dynamic Programming Solution

In this section, we present a dynamic programming solution to the tagging problem. As mention in Section III-D, the value of $q_i(c_i + x_i)(i = 1, 2, \ldots, n)$ can be obtained for resource r_i when it receives x_i more posts.

We note that $\mathcal{P}(B, \mathcal{R})$ exhibits *optimal substructure*, which enables dynamic programming. To show this, let (Q, X) be the optimal solution to $\mathcal{P}(B, \mathcal{R})$, where Q is the optimal value and $X = \{x_i | i = 1, 2, \ldots, n\}$ is the assignment of x_i to r_i in the optimal solution. Consider the subproblem $\mathcal{P}(B - x_n, \mathcal{R} - \{r_n\})$, we claim that $(Q - q_n(c_n + x_n), X - \{x_n\})$ is the optimal solution to this subproblem. Suppose (Q', X') is the optimal solution to $\mathcal{P}(B - x_n, \mathcal{R} - \{r_n\})$, and $Q' > Q - q_n(c_n + x_n)$. Let $(Q'', X'') = (Q' + q_n(c_n + x_n), X' \cup \{x_n\})$. Since $\sum_{i=1}^{n-1} x'_i = B - x_n$, (Q'', X'') is also a solution to $\mathcal{P}(B, \mathcal{R})$. However, $Q'' = Q' + q_n(c_n + x_n) > Q$, which violates the assumption that Q is the optimal value to $\mathcal{P}(B, \mathcal{R})$. Therefore, $(Q - q_n(c_n + x_n), X - \{x_n\})$ must be the optimal solution to the subproblem.

Next, we define the optimal value of $\mathcal{P}(B, \mathcal{R})$ recursively. Let $Q(b, l)$ be the optimal value to subproblem $\mathcal{P}(b, l)$, where $0 \le b \le B$ and $1 \le l \le n$. We obtain the recurrence

$$Q(b, l) = \begin{cases} q_1(c_1 + b) & l = 1, \\ \max_{0 \le x_l \le b} Q(b - x_l, l - 1) + q_l(c_l + x_l) & l > 1. \end{cases} \tag{17}$$

The optimal assignment $X(b, l)$ to $\mathcal{P}(b, l)$ can be defined accordingly. Let $y_{b,l}$ be the point at which the optimal value of Equation 17 is achieved, i.e.,

$$y_{b,l} = \arg\max_{0 \le x_l \le b} Q(b - x_l, l - 1) + q_l(c_l + x_l). \tag{18}$$

Note that $y_{b,1} = b, b = 0, 1, \ldots, B$. We have

$$X(b, l) = \begin{cases} \{x_1 = y_{b,1}\} & l = 1, \\ X(b - y_{b,l}, l - 1) \cup \{x_l = y_{b,l}\} & l > 1. \end{cases} \tag{19}$$

Our bottom-up, dynamic programming algorithm is showed in Algorithm 6. $Q[b, l]$ and $y[b, l]$ are both $B \times n$ arrays storing the optimal value of $\mathcal{P}(b, l)$ and the corresponding point $y_{b,l}$ respectively. So the optimal value of $\mathcal{P}(B, \mathcal{R})$ is $Q[B, n]$, and the optimal assignment is $X[B, n]$.

We first deal with the bound case where $l = 1$(Step 1 to 3). Then for each subproblem $\mathcal{P}(b, l)$, we enumerate all the possible assignments to x_l and figure out the one at which the optimal value is achieved. $Q[b, l]$ and $y[b, l]$ are updated accordingly(Step 4 to 9). The optimal value $Q^* = Q[B, n]$ and the optimal assignment $X^* = X[B, n]$ are calculated in Step 10 to 14.

Algorithm 6 DP

Require: Budget B, Resources $\mathcal{R}$, Initial no. of posts $\vec{c}$
1: **for** $b \leftarrow 0$ **to** B **do**
2: $\quad Q[b, 1] \leftarrow q_1(c_1 + b)$
3: $\quad y[b, 1] \leftarrow b$
4: **for** $l \leftarrow 2$ **to** n **do**
5: $\quad$ **for** $b \leftarrow 0$ **to** B **do**
6: $\quad\quad$ **for** $x \leftarrow 0$ **to** b **do**
7: $\quad\quad\quad$ **if** $Q[b, l] < Q[b - x, l - 1] + q_k(c_k + x)$ **then**
8: $\quad\quad\quad\quad Q[b, l] \leftarrow Q[b - x, l - 1] + q_k(c_k + x)$
9: $\quad\quad\quad\quad y[b, l] \leftarrow x$
10: $Q^* \leftarrow Q[B, n]$
11: $b \leftarrow B$
12: **for** $l \leftarrow n$ **downto** 1 **do**
13: $\quad X^*[l] \leftarrow y[b, l]$
14: $\quad b \leftarrow b - y[b, l]$
return (Q^*, X^*)

As we can see, all subproblems are scanned once during the process and each time all possible values of x_l are checked once to find the optimal one. Besides, calculating the specific value of the quality function requires $O(|\mathcal{T}|)$ time and $O(|\mathcal{T}|)$ space. Therefore, the time and space complexities of this solution are $O(n|\mathcal{T}|B^2)$ and $O(nB + n|\mathcal{T}|)$, respectively.

C. Complexity Analysis

We now describe the time and space complexities of the practical incentive allocation strategies.

1. *FC*. Since *FC* lets users to freely choose a resource to tag, T_{I}, T_{C} and T_{U} are all equal to $O(1)$. The total time complexity is $O(n+B)$ and space complexity is $O(n)$ for keeping $x_i (1 \le i \le n)$.

2. *RR*. As shown in Algorithm 2, INIT(), CHOOSE() and UPDATE() cost $O(1)$ time. Similar to *FC*, the time and space complexities are $O(n + B)$ and $O(n)$, respectively.

3. *FP*. In the priority queue structure, *push* and *pop* can be completed in $O(\log n)$ time. Therefore, T_{I} is equal to $O(n \log n)$ and T_{C}, T_{U} are both equal to $O(\log n)$. The time complexity of *FP* is $O((n + B) \log n)$. Since the size of the priority queue is n, the space complexity of *FP* is $O(n)$.

4. *MU*. In *MU*, we need to compute $m_i(c_i + x_i, \omega)$ before the priority queue is updated (Step 4 and Step 9 in Algorithm 4). A straightforward way is to store r_i's latest ω *rfd*s' in ω vectors with length $|\mathcal{T}|$ and use Equation 16 and Definition 8 to calculate $m_i(c_i + x_i, \omega)$ in $O(\omega|\mathcal{T}|)$ time. When $k_i > \omega$,

$$\begin{aligned} & (\omega - 1)m_i(k_i, \omega) + s(\vec{F}_i(k_i - \omega), \vec{F}_i(k_i - \omega + 1)) \\ = \; & (\omega - 1)m_i(k_i - 1, \omega) + s(\vec{F}_i(k_i - 1), \vec{F}_i(k_i)). \end{aligned}$$

Based on this observation, if we use a queue to store the following ω values: $s(\vec{F}_i(j - 1), \vec{F}_i(j))(j = c_i + x_i - \omega + 1, \ldots, c_i + x_i)$ and $m_i(c_i + x_i - 1, \omega)$ is known, the computation of $m_i(c_i + x_i, \omega)$ can be reduced to $O(|\mathcal{T}|)$ time and the space complexity is reduced to $O(\omega + |\mathcal{T}|)$. Therefore, in *MU*, $T_{\mathrm{I}} = O(n(\omega|\mathcal{T}| + \log n))$ and $T_{\mathrm{C}} + T_{\mathrm{U}} = O(\log n + |\mathcal{T}|)$. The total time complexity of *MU* is $O((n+B) \log n + (n\omega + B)|\mathcal{T}|)$ and the space complexity is $O(n\omega + n|\mathcal{T}|)$.

5. *FP-MU*. *FP-MU* successively employs *FP* and *MU* to allocate the incentive. So the time and space complexities are respectively $O((n + B) \log n + (n\omega + B)|\mathcal{T}|)$ and $O(n\omega + n|\mathcal{T}|)$.