Locally Expandable Allocation of Folksonomy Tags in a Directed Acyclic Graph

Takeharu Eda[1,2], Masatoshi Yoshikawa[2], and Masashi Yamamuro[1]

[1] NTT Cyber Space Laboratories, Japan
{eda.takeharu,yamamuro.masashi}@lab.ntt.co.jp
[2] Kyoto University, Japan
yoshikawa@i.kyoto-u.ac.jp

Abstract. We propose a new classification system based on an analysis of folksonomy data. To find valuable resources from current social bookmark services, users need to specify search terms or tags, or to discover people with similar interests. Our system uses semantic relationships extracted from the co-occurrences of folksonomy data using PLSI and allocates folksonomy tags in a directed acyclic graph. Compared to the hierarchical allocation method of a tree, our method guarantees the number of children nodes and increases the number of available paths to an objective node, enabling users to navigate the resources using tags.

1 Introduction

Web services like blogs and wiki have become very popular. In these services, many entries are updated frequently and many links are generated automatically, degrading the value of web links. There are also a lot of spam blogs, called splogs, which also make link analysis and weighting of web pages difficult.

One method of overcoming these drawbacks, social bookmark services (SBMs) (e.g. del.icio.us[1], Hatena Bookmark[2]) have attracted a lot of attention. They use a bottom-up taxonomy system called *folksonomy*. SBMs offer users bookmarking functions and store bookmark entries on the server. Users can attach tags with varying numbers of keywords to their entries and make comments on entries. Compiling bookmark entries is a good way of ranking fresh user-chosen web pages [15]. Users are loosely connected by bookmarking the same entries.

Folksonomy can be considered one way of classifying resources (URLs). Tags are viewpoints selected by users and used later as clues to find resources. Almost all net surfing users know how to bookmark URLs. In many cases, SBM interfaces are provided as web browsers' plug-ins, which have almost the same functions as default bookmarking functions. Because there is only a small gap between users and SBM systems, users can easily start using SBMs. That is the reason why SBMs are such a popular resource annotating system.

[1] http://del.icio.us
[2] http://b.hatena.ne.jp

J. Bailey et al. (Eds.): WISE 2008, LNCS 5175, pp. 151–162, 2008.

One folksonomy problem is the explosion of the number of tags. Many SBMs offer simple *tag clouds* where tags are ordered in the order of syllabary or usage frequency. However, it is difficult for users to navigate resources using tag clouds since they have no semantic order.

We propose a new hierarchical allocation method of folksonomy tags by extracting the relationships among them. We compute the feature vectors from co-occurrence data of folksonomy and measure and estimate the abstract differences between tags. We also propose a tag allocation algorithm into DAG (Directed Acyclic Graph). Our method enables users to intuitively understand the connections among tags. Our contributions are as follows.

- We propose a new method of allocating folksonomy tags into DAG (Directed Acyclic Graph). We calculate feature vectors using Probabilistic Latent Semantic Indexing (PLSI), measure them using probabilistic vectors, and estimate abstract upper or lower level differences among tags by comparing the entropy value of the tags.
- We implement both our method and the existing hierarchical tag allocation algorithm. We discuss the differences of between the techniques and compare them in experiments.
- In the experiments, we compare not only the hierarchical structure but also the vector models employed in both approaches, and show that the statistical method (PLSI) can grasp hidden semantics from observed bookmarks.

The paper is organized as follows. In Section 2, we survey related work on folksonomy and clarify our research problems. In Section 3 we propose our new tag allocation method. In Section 4 we conducted qualitative experimentation. Finally, we describe future work and conclude the paper in Section 5.

2 Related Work

The number of social web services supporting user online activities and friendships continues to increase. Folksonomy is attracting a lot of attention as a way of accumulating users' viewpoints on resources and is expanding its role from tagging only URLs into pictures, movies, papers, and etc. These services offer users' scores and comments as well as tags.

A lot of work has been done on folksonomy [11], [13]. Brooks and Montanez [1] proposed automatic tagging based on content and discussed hierarchical allocation of tags. Since it is possible to consider hierarchically allocated tags as an ontology, one of their conclusions is that such a technique is useful in many real applications. However, they also noticed that a tree structure is too strict for users as a map.

There are a lot of case studies on folksonomy data [6], [9]. Golder and Huberman [4] analyzed the distributions of tags, users, and resources in del.icio.us and showed that different users act differently. Sen et al. [12] did a case study of MovieLens and found that tags are organized based on user effort. Niwa et al. [10] consider folksonomy a method of making web page recommendations. They

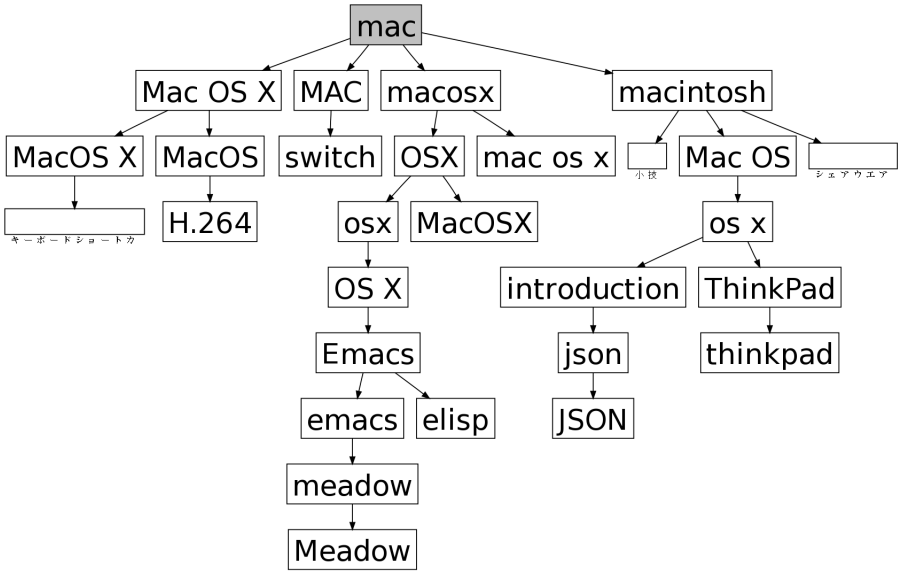

Fig. 1. Heymann's Tree (**EXT** Approach)

insisted that synonym and ambiguity problems can be solved by clustering tags, but they did not evaluate applicability of tags as a classification system. Dubinko et al. [3] visualize frequently used tags in a time series. Since SBMs keep gathering fresh bookmark entries, users can see the history of popular URLs by using such timeline visualizations.

To solve the problem of synonyms and ambiguity in tags, Wu et al. [14] proposed a probabilistic indexing model for folksonomy triples (user, tag, resource). One of the main advantages of their method is the ability to index all attributes at the same time. Another method for tag disambiguation is based on bipartite network analysis [2].

If we consider the problem is one of automatic organization of tags by computers, the obvious choice is to use hierarchical clustering algorithms. However, in such hierarchical clustering, the difficult problem is to determine the labels for inner nodes.

Heymann et al. [5] proposed a method of allocating tags in a tree based on the *folksonomy vector model*. In this model, tag vectors are composed of the number of people who tagged each URL. The dimension of the tag vector is the number of tagged URLs. After extracting tag vectors from folksonomy triples, their method measures the distances among tags with cosine similarities and calculates the centrality of each tag by considering whole tag sets as a network. The centrality they used is betweeness centrality, which can be quickly approximated. They proposed the extensible greedy algorithm where tags are sorted in this order of centrality, and the algorithm starts from the most central tag and builds a tree in top-down manner. However, their method has several problems. One is the

problem of determining the position of a tag to another tag as a child. That is, a node must always have one parent node. Consider real classification systems like Open Directory[3] and Google Directory[4]: we find it natural that several nodes have several parents, making it easier for users to find resources. Another problem is the sparseness of folksonomy vectors, which causes mis-allocation of nodes near the leaf. Fig. 1 show a Heymann's tree which rooted at "mac"[5]. We see that several tags around leaves are far from the topic related to "mac".

3 Proposed Method

We propose a new method of allocating folksonomy tags into DAG. The left side and right side of Fig. 2 illustrate the processing flow of proposed method and Heymann's approach [5], respectively. Both approaches consist of two steps, tag vectorization and hierarchical allocation of tag vectors. In the following, our approach, the PLSI vectors and the DAG allocation algorithm, is referred to as **DAG** and Heymann's approach, the folksonomy vectors and the extensible greedy algorithm, is referred to as **EXT**. At first, we calculate feature vectors using PLSI and we call the vectors PLSI vectors. PLSI exposes hidden relationships among item sets in statistical way; leading to better precision than naive folksonomy vectors.

PLSI vectors of tags are allocated in DAG by determining children of each PLSI vector. We find k nearest neighbors of each vector and set the them as children if a child's abstract level is lower than the parent. We estimate abstract upper or lower level differences among tags by comparing the entropy value of PLSI vectors.

We used three-mode PLSI [14] for 3-tuple co-occurrence data composed of user, tag, and resource. Although PLSI vectors for user, tag, and resource are calculated at the same time, we use only PLSI tag vectors.

3.1 Folksonomy Indexing Using PLSI

PLSI enables us to index co-occurrence data (a subset of the direct product of several item sets) with given dimension feature vectors. This can be understood as projection from each set to a semantic vector space. Another understanding of PLSI is clustering of both documents and terms with *soft* membership functions to clusters.

Although PLSI was first developed for two-mode co-occurrence data (document×term) as a probabilistic variant of LSI [7] , it has been extended to three-modes in several studies. By calculating distances among PLSI vectors, we can index the closeness among items. Since PLSI uses the semantic spaces among item sets, the closeness among PLSI vectors is based on the closeness in the semantic space.

[3] `http://www.dmoz.org`
[4] `http://directory.google.com/`
[5] This was drawn in the setting in Section 4.

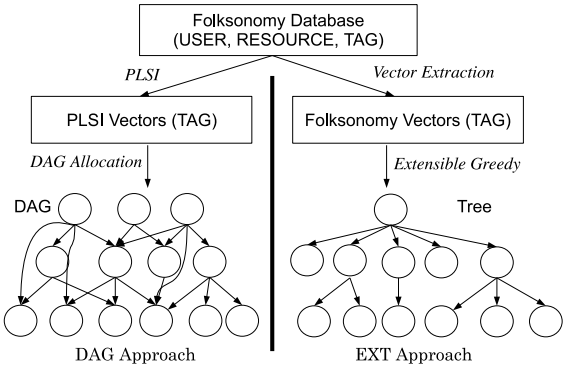

Fig. 2. Overall Processing Flow

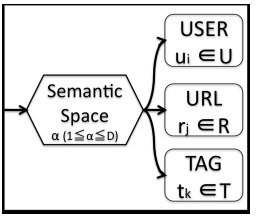

Fig. 3. Probabilistic Model

Fig. 3 shows the model for PLSI used in our method. U, R, and T stand for user set, resource set, and tag set, respectively. The dimension of a semantic space is D. Occurrence probability from $\alpha(\in [1, 2, \ldots, D])$ to $u_i \in U$, $r_j \in R$, $t_k \in T$ is denoted by $p(u_i|\alpha), p(r_j|\alpha), p(t_k|\alpha)$. We assume that occurrence probability of α is $p(\alpha)$. The equation below holds for u_i, r_j, t_k, and the semantic space[6].

$$p(u_i, r_j, t_k) = \sum_{\alpha=1}^{D} p(\alpha)p(u_i|\alpha)p(r_j|\alpha)p(t_k|\alpha)$$

The left-hand side can be observed from the data set, and probabilities in the right-hand side can be estimated using EM algorithms with initial values [14]. EM algorithm consists of E (Expectation) step and M (Maximization) step. Both steps are alternated repeatedly until convergence. In our implementation, tempered EM algorithms are used to avoid local optimums.

Using Bayes' rule, we can calculate conditional probabilities from items to the semantic space. By considering probability from an item to α dimension as a α dimension value in a vector space, we consider all items vectors in D dimensional vector space.

Examples of Calculated PLSI Vectors. Fig. 4 shows an example of PLSI vectors. The horizontal and vertical axes show dimension and probability, respectively. All tags in Fig. 4 related to "mac os x" have a high probability around the 75-th dimension.

Entropy Values of PLSI Vectors. Entropy of tag t_k is calculated as follows.

$$H(t_k) = -\sum_{\alpha=1}^{D} p(\alpha|t_k) \cdot log(p(\alpha|t_k))$$

[6] $p(u_i, r_j, t_k)$ stands for occurrence probability of a triple.

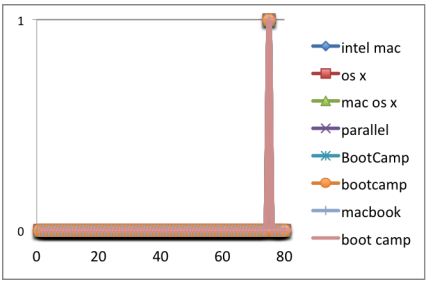

Fig. 4. Example of PLSI Vectors

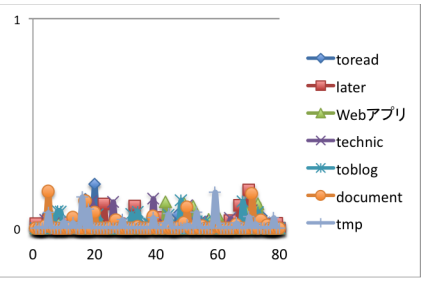

Fig. 5. High Entropy PLSI Vectors

The entropy value equals 0 when the PLSI vector has only one peak and increases as the distribution approaches normal. Entropy is considered as a measure of ambiguity of probabilistic vectors. Fig. 5 shows example PLSI vectors which are of high entropy values, while Fig. 4 shows low entropy vectors.

Distances among PLSI Vectors. In order to measure tag similarity we need distance among PLSI vectors. In this study, we used JS divergence, which is a distance between probabilistic distributions and considered to have better accuracy for information retrieval tasks [8]. JS divergence is calculated as follows[7].

$$D_{js}(t_k, t_l) = \frac{1}{2}[D(t_k||avg(t_k, t_l)) + D(t_l||avg(t_k, t_l))]$$

3.2 DAG Allocation Algorithm

DAG DAG is a directed graph without a directed cycle. Any node leads to an terminal node if the number of nodes in DAG is finite. DAG is intuitively considered to have flow or order of nodes.

Fig. 6 shows a DAG constructed by our method.

We use the entropy value of PLSI vectors to choose this flow and allocate tags in DAG. Table 1 shows the DAG allocation algorithm, which is quite simple. For all nodes, we set children nodes with `set_children`, in which, we first search the nearest neighbors and designate the node as a child if its entropy is less than the parent's entropy. In the data structure constructed by our algorithm, it is impossible to go back to the starting node when navigating directed edges from parent to child. That means that the constructed data structure is a DAG.

One advantage of DAG is that there is no cycle when a user navigates one direction, that is, the user never goes back to the same node while navigating in DAG. The allocation algorithm in Table 1 chooses neighbors whose distances are less than a certain threshold. However, sometimes no children are found. In that case, the condition can be extended to the k nearest neighbor nodes depending on the distribution of JS divergence values. We discuss the distribution of distances in Section 4.

[7] $D(q||r)$ is KL divergence between q and r.

Table 1. DAG Allocation Algorithm

input: PLSI vector list. $\{v_i\}_{i=1}^{i=m}$
output: DAG where PLSI vectors are allocated in.
1 **for**$(i = 1; i < m + 1; i+ = 1)$
2 `set_children`(v_i, k, n)
3 **endfor**

Function: `set_children`
input: Node: v, Threshold: k, Number of Chidren: n
1 $\{w_i\}_{i=1}^{i=n}$: Get all nodes near to v of the distance within k
2 **for**$(i = 1; i < n + 1; i+ = 1)$
3 if$(w_i.entropy < v.entropy)$ then
4 Set w_i as a child of v.
5 **endif**
6 **endfor**

The time complexity of the DAG allocation algorithm is $O(n^2)$, which is in the same class as the extensible greedy algorithm [5]. Though the extensible greedy algorithm is an offline algorithm, which needs to construct the whole hierarchical structure beforehand, our algorithm is an online algorithm that enables us to expand children locally around the required node.

4 Evaluation

We conducted several experiments to test DAG's performance in the following areas.

- Correlation and difference between cosine similarity and JS divergence.
- Precision of children nodes calculated by DAG and the extensible greedy algorithm.

We compared our method with the extensible greedy algorithm using the folksonomy vector model [5]. Both approaches are described in Table 2.

We implemented folksonomy vectors, PLSI vectors, the extensible greedy algorithm, and the DAG allocation algorithm in the Ruby programming language. We first constructed both data structures, a tree and a DAG, and stored them on disk. During the evaluation, we extracted the required sub-structure from disk. We used graphviz[8] to visualize the sub-structures.

As for the data set, we crawled popular entries at Hatena Bookmark in October 2006. We got a list of popular tags from `http://b.hatena.ne.jp/t` and retrieved a set of URLs that were linked from the site. HTML files were scraped into folksonomy triples and stored in a RDBMS. Before PLSI, we filtered out the users, tags, and resources that appeared less than five times, which caused us to filter triples out too, for the purpose of noise reduction.The data set is shown in Table 3. All

[8] `http://graphviz.org`

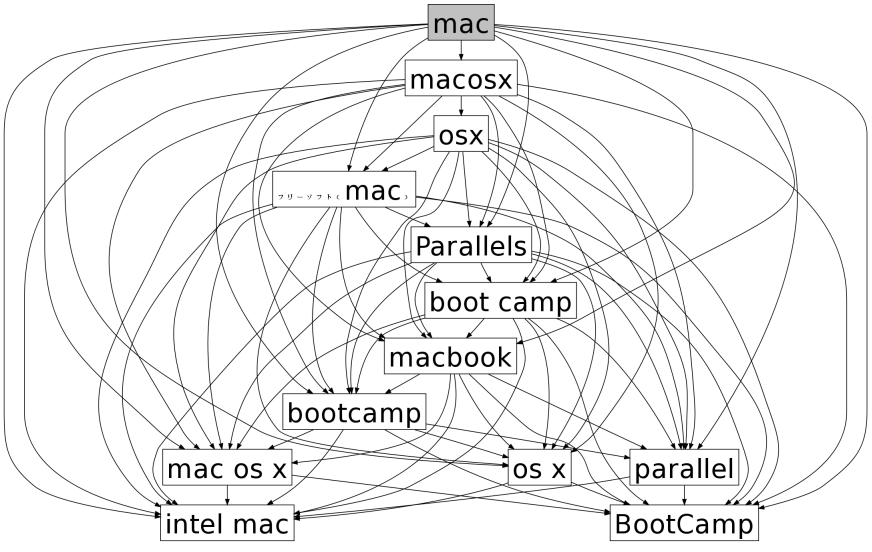

Fig. 6. DAG (**DAG** Approach)

Table 2. DAG and **EXT** Approaches

	DAG	**EXT**
Vector	PLSI Vector	Folksonomy Vector
Distance	JS Divergence	Cosine Similarity
Data Structure	DAG	Tree

the parameters of the algorithms were chosen empirically as shown in Table 4. EM iterations were chosen with the same setting used in [14]. We tried several dimensions for the semantic space and chose the dimension by checking kNN of sampled tags by users. **DAG**'s thresholds k and n were determined in an ad-hoc manner since **DAG** has the ability to expand children locally. In the case of **EXT**, we tried several thresholds and determined the threshold to avoid the very shallow tree[9].

4.1 Differences in Distances

Fig. 7 shows the distribution of distances between 300 randomly chosen tags; the horizontal axis represents cosine similarity and the vertical axis represents JS divergence.

Although, there is no clear association between cosine and JS divergence, we notice that there were several points whose cosines equaled 0 but whose JS divergence varied over a wide range. To verify this point, we calculated the cases

[9] If we increase the threshold of **EXT**, many nodes become children of the root.

Table 3. Data Set

Tags	5496
Users	11713
Resources	4019
Triples	1190504

Table 4. Parameters

DAG	Dimension of Semantic Space	80
	EM Iterations	80
	Threshold: k	0.2
	Num. of Children: n	11
EXT	Threshold: k	0.00001

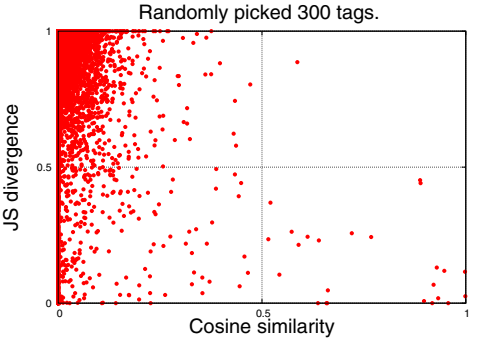

Fig. 7. Distribution of Cosine and JS Divergence

where two tags become maximally distant from each other, that is, the cosine similarity equals 0 and the JS divergence equals 1, respectively. The result is that almost half(45%) of the folksonomy vector pairs in the data set were maximally distant from each other. In contrast, only a few(7%) of PLSI vectors are maximally far from each other. This means that the statistical method(PLSI) enables us to measure the tags which are furthermost in folksonomy vector model.

Revealed Connections

Table 5 shows tag pairs whose cosine similarity equals 0 and whose JS divergence is near 0. Tag pairs are randomly chosen, and only English words are shown[10]. Using **EXT** approach we were unable to detect the relationships shown in Table 5, and detected for our **DAG** approach. These trends are applicable to tags in Japanese.

4.2 Comparison of Local Structures

We randomly sampled tags and compared children nodes rooted at the tag. For all sampled tags, we counted the number of child nodes in both methods and evaluated the precision of child nodes by hand. Since both **DAG** and **EXT** approaches do not aim for the extraction of exact parent-child relationships like *is-a* and *part-of*, the correct answers were judged from the points of relatedness and ancestor-descendant relationship.

[10] Since Hatena bookmark is Japanese bookmark service, there are many tags in Japanese.

Table 5. Revealed Connections. (Example pairs of tags cosine equals 0 and JS divergence is near to 0.)

CRC	partition	recover	nLite
CSS3	xhtml	Wii	RMT
KDDI	Softbank	VB.NET	Trac
attention economy	nagios	KDDI	WILLCOM
mod_proxy_balancer	rrdtool	3DCG	flash
j2ee	BDD	prototype	wysywig
ASP.NET	Trac	WinFS	foldershare
Rails	memcached	trends	NAMAAN
Winny	Privacy	liveup	BRAVIA
vim	memcached	Plagger	LINUX
GoogleEarth	google mars	trac	unison

The average number of child nodes and the average precision are shown in Fig. 8. **DAG**'s average precision is 11% less than that of **EXT**. However, **DAG**'s average number of child nodes is eight. This means that the **DAG** approach has 2.7 times as many child nodes for just an 11% decrease in precision compared to **EXT**. Furthermore, the **DAG** approach's ability of *local expansion* is confirmed from Fig 9.

4.3 Summary

The PLSI vector model enables us to reveal the hidden relationships among tags with JS divergence distance. Several software names which run on Apple's "mac" do not appear in Fig. 1 but appear in Fig. 6. This property is the main advantage of applying the statistical method for indexing co-occurrence data like folksonomy triples.

The **DAG** approach has 2.7 times as many child nodes as the **EXT** approach with 11% less precision when parameters are statically determined. However, the **EXT** approach cannot expand children locally while **DAG** approach enables us to expand children with only a small drop in precision. **DAG** can be used in

(average)	Num. of Children	Precision
DAG	8.00	0.64
EXT	3.00	0.75

Fig. 8. Average Number of Children and Precision

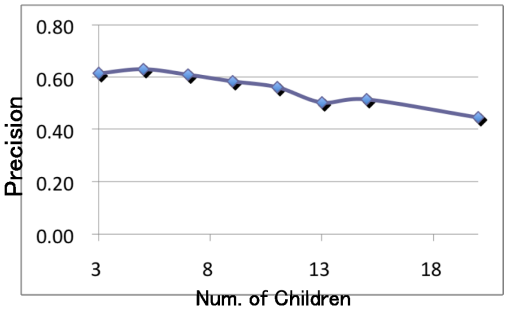

Fig. 9. Precision-Num. of Children Curve

applications such as user navigation tool bars to give users more navigation choices instantly on their demand.

5 Conclusions

We proposed a new method of allocating tags in DAG and evaluated our approach qualitatively comparing to the existing hierarchical tag allocation algorithm. We analyzed folksonomy triples with PLSI and allocate tags in DAG.

Experiments showed that PLSI vectors are scattered in a small dimensional vector space compared to the naive folksonomy vector space, which leads to the revelation of hidden relationships among tags. We found that **DAG** offers 2.7 times as many child nodes with 11% less precision in the statical parameter setting. We also showed that our approach has the ability to expand children locally with small precision degradation. These capabilities can be applied to interactive folksonomy navigation systems.

To exploit the locally expandable tag hierarchy, threshold adjustment is an interesting future work. Increasing the thresholds means that tag islands are grouped into larger islands. By snapshotting DAGs with several thresholds, we might see the hierarchical clusters above DAGs. Since folksonomy systems are continuing to compile new entries, visualization of DAG in a time series is another future work.

References

1. Brooks, C.H., Montanez, N.: Improved annotation of the blogosphere via autotagging and hierarchical clustering. In: Proc. International World Wide Web Conference, pp. 625–632. ACM Press, New York (2006)
2. Yeung, C.A., Gibbins, N., Shadbolt, N.: Tag Meaning Disambiguation through Analysis of Tripartite Structure of Folksonomies. In: Proc. International Conferences on Web Intelligence and Intelligent Agent Technology (2007)
3. Dubinko, M., Kumar, R., Magnani, J., Novak, J., Raghavan, P., Tomkins, A.: Visualizing tags over time. In: Proc. International World Wide Web Conference. ACM Press, New York (2006)
4. Golder, S., Huberman, B.A.: The structure of collaborative tagging systems. Journal of Information Science (2006)
5. Heymann, P., Garcia-Molina, H.: Collaborative Creation of Communal Hierarchical Taxonomies in Social Tagging Systems. Technical report, `http://heymann.stanford.edu/taghierarchy.html`
6. Heymann, P., Koutrika, G., Garcia-Molina, H.: Can Social Bookmarking Improve Web Search? In: Proc. International Conference on Web Search and Data Mining (2008)
7. Hofmann, T.: Probabilistic latent semantic analysis. In: Proc of the Fifteenth Conference on Uncertainty in Artificial Intelligence, Stockholm (1999)
8. Lee, L.: On the effectiveness of the skew divergence for statistical language analysis. In: Proc. International Workshop on Artificial Intelligence and Statistics, pp. 65–77 (2001)

9. Lux, M., Granitzer, M., Kern, R.: Aspects of Broad Folksonomies. In: Proc. International Conference on Database and Expert Systems Applications (2007)
10. Niwa, S., Doi, T., Honiden, S.: Web Page Recommender System based on Folksonomy Mining. In: Proc. International Conference on Information Technology: New Generations (2006)
11. Sabou, M., Gracia, J., Angeletou, S., dAquin1, M., Motta, E.: Evaluating the Semantic Web: A Task-Based Approach. In: Aberer, K., Choi, K.-S., Noy, N., Allemang, D., Lee, K.-I., Nixon, L., Golbeck, J., Mika, P., Maynard, D., Mizoguchi, R., Schreiber, G., Cudré-Mauroux, P. (eds.) ASWC 2007 and ISWC 2007. LNCS, vol. 4825, pp. 423–437. Springer, Heidelberg (2007)
12. Sen, S., Lam, S.K., Rashid, A.M., Cosley, D., Frankowski, D., Osterhouse, J., Harper, M.F., Riedl, J.: Tagging, communities, vocabulary, evolution. In: Proceedings of the 2006 20th anniversary conference on Computer supported cooperative work. CSCW, New York, NY, USA, pp. 181–190 (2006)
13. Vo, J.: Tagging, Folksonomy & Co - Renaissance of Manual Indexing? unpublished (2007), http://arxiv.org/abs/cs/0701072v2
14. Wu, X., Zhang, L., Yu, Y.: Exploring Social Annotations for the Semantic Web. In: Proc. International World Wide Web Conference, pp. 417–426 (2006)
15. Yanbe, Y., Jatowt, A., Nakamura, S., Tanaka, K.: Can Social Bookmarking Enhance Search in the Web? In: Proceedings of the 7th ACM/IEEE Joint Conference on Digital Libraries (2007)