Hierarchical Tag Visualization and Application for Tag Recommendations

Yang Song
Microsoft Research
One Microsoft Way
Redmond, WA 98052, USA
yangsong@microsoft.com

Baojun Qiu
Computer Science & Engineering
Pennsylvania State University
University Park, PA 16802, USA
bqiu@cse.psu.edu

Umer Farooq
Microsoft Corp.
One Microsoft Way
Redmond, WA 98052, USA
umfarooq@microsoft.com

ABSTRACT

Social bookmarking sites typically visualize user-generated tags as tag clouds. While tag clouds effectively show the relative frequency and thus popularity of tags, they fail to convey two aspects to the users: (1) the similarity between tags, and (2) the abstractness of tags. We suggest an alternative to tag clouds known as tag hierarchies. Tag hierarchies are based on a minimum evolution-based greedy algorithm for tag hierarchy construction, which iteratively includes optimal tags into the tree that introduce minimum changes to the existing taxonomy. Our algorithm also uses a global tag ranking method to order tags according to their levels of abstractness as well as popularity such that more abstract tags will appear at higher levels in the taxonomy. Based on the tag hierarchy, we derive a new tag recommendation algorithm, which is a structure-based approach that does not require heavily trained models and thus is highly efficient. User studies and quantitative analysis suggest that (1) the tag hierarchy can potentially reduce the user's tagging time in comparison to tag clouds and other tag tree structures, and (2) the tag recommendation algorithm significantly outperforms existing content-based methods in quality.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval; I.5.3 [**Pattern Recognition**]: Clustering—*algorithms*

General Terms

Algorithms, Visualization, Experimentation

Keywords

hierarchical tag visualization, tag recommendation, minimum evolution criteria

1. INTRODUCTION

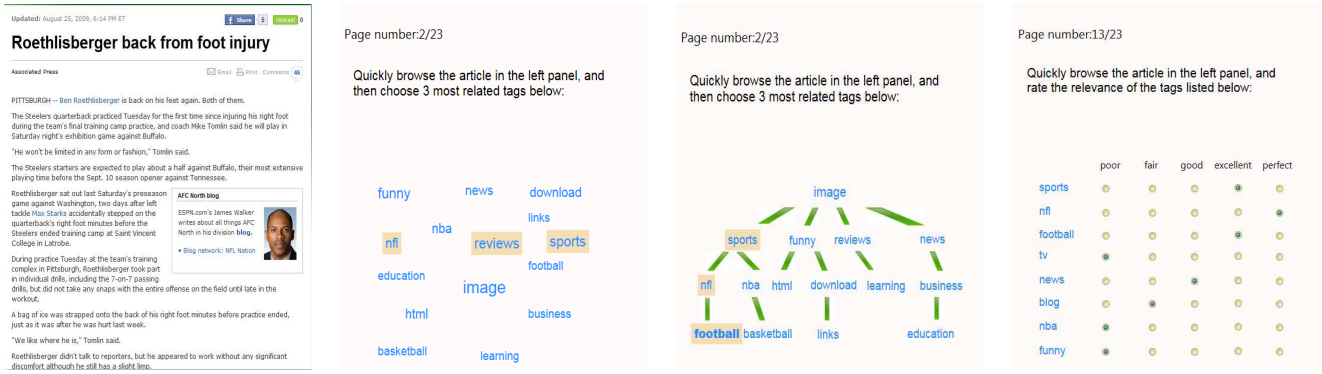
Social bookmarking systems, like Delicious[1] and Flickr[2], are one of the several systems that have been popularized by Web 2.0. Social bookmarking systems allow users to specify keywords or tags for web resources that are of interest to them, helping them to organize and share these resources with others in the community. The most prevalent way of displaying tags is using tag clouds, in which tags are organized in a two-dimensional array with (often) no particular order. Popular tags are typically expressed using larger font sizes or different colors.

While tag clouds offer a natural way of visually exploring the popularity of tags, it has failed to express two important information: (1) the similarity between different tags, and (2) the concept abstractness of different tags. Firstly, tag clouds cannot directly show how closely two tags are in its two-dimensional representation, which is quite important when users try to find relevant tags when performing tagging. Currently, most machine learning-based models for tag recommendations require a similarity measure of tags, e.g., tag co-occurrence or tag correlation [15, 8]. Since the representation of tag clouds can rarely help discovering tag similarities in this case, a lot of approaches have to resort to the content of the resources (e.g., the text of documents, the low-level features of images and so on) for learning models, which is known to be time-consuming [3, 13, 16].

Secondly, although some preliminary studies have tried to improve the tag clouds by clustering-based methods [9, 2], they are unable to observe the underlying concept abstractness of tags. For example, *programming* is a more abstract concept than *java*, which is in turn more abstract than *j2ee*. While more concrete tags give more specific descriptions of resources (e.g., j2ee based applications), more abstract tags provide a higher-level view of the underlying topics in the resources, which usually attract more interests from the general population. On the other hand, more abstract tags can also be (ambiguously) related to multiple topics. e.g., different users may use *tools* to annotate both resources about software or craftsman instruments. Consequently, simply clustering tags and displaying a 2-D array is not sufficient for abstract tags. As a result, it is important to understand the level of abstractness of tags, which can greatly help organizing and managing the user-expressed knowledge of a particular set of resources, as suggested by traditional approach of taxonomy learning. Since the unstructured representation of tag clouds is not capable of reflecting tag abstractness,

[1]http://delicious.com/
[2]http://www.flickr.com/

CIKM'11, October 24–28, 2011, Glasgow, Scotland, UK.



(a) (b) (c) (d)

Figure 1: (a): A web page used in our user study. (b): Typical tag cloud expressed in current social bookmarking systems. (c): Tag tree based on a hierarchical tag representation using the minimum evolution criteria. (d): Judgers rates the tag relevancies. In (b) and (c), the highlighted tags are the most tagged ones.

having an alternative way of organizing and displaying tags is of imminent importance.

In this paper, we propose a novel framework to address both aforementioned issues in tag visualization. We suggest a hierarchical tag representation to visualize tags in a tree-structured taxonomy. In our representation, each node in the tree is a user-assigned tag, whose abstractness is controlled by its level in the tree. The edges in the tree reflect the closeness between tags, e.g., how often are they used to tag the same documents. We propose a principled way to construct the taxonomy by introducing the minimum evolution (ME) criteria from phylogenetics [6]. This structure offers a much richer knowledge representation than traditional tag clouds, thus can be potentially leveraged for many applications. As an extension to the tag hierarchy visualization, we derive an efficient tag recommendation algorithm using the tag taxonomy. The recommendation algorithm considers both the tag abstractness and the similarity between tag taxonomy and the resource content. Our algorithm is novel in the sense that it does not require training models that contain knowledge from previous seen documents.

Figure 1 illustrates one of the examples we tested during a user study. The same document is given to two sets of users with tag cloud and tag hierarchy respectively. Users are asked to find the most relevant tags as quickly as possible. It turned out that once the users have located *sports*, it only takes them 1.79 seconds to select *football* using tag hierarchy. While for tag cloud, users spend 3.56 seconds on average to locate the next tag. Besides being more efficient, tag hierarchy also results in more relevant tags being chosen, according to the user rating in terms of NDCG scores.

Specifically, this paper makes the following contributions:

• We propose an *unsupervised* approach to hierarchical tag visualization, which is a novel way to effectively exhibit both the similarity between tags, the level of abstractness and popularity for individual tags.

• We introduce two novel methods for ranking tags: information theoretic-based and learning-to-rank-based approaches, focusing on the popularity and the abstractness of tags respectively. We then combine the power of these two methods to assign global rankings to tags.

• We use a principled way to construct tree according to the minimum evolution (ME) criteria. Our greedy algorithm along with the minimum description length (MDL) guideline ensures that the tag hierarchy is calibrated to be balanced and well-organized into topics.

• We conducted a preliminary user study comparing tag clouds with tag trees. Results showed that the tagging time using tag trees was lesser than using tag clouds.

• Based on the hierarchical representation, we introduce a novel tag recommendation algorithm. Our algorithm is capable of matching a target document with one or more subtree of tags, then sorting the tags according to their correlations with the document. Comparing to previous approaches that require the content of the training documents for model learning, our framework does not involve any features from document contents, making it efficient and scalable.

It should be noted that although our proposal contains both tag visualization and recommendation, we put much more emphasis on the first part in this paper. Our biggest contribution is the visualization framework, while recommendation algorithm is a plus. Due to space limitations, we are unable to show all the details of our recommendation algorithm. Also, many other tag suggestion methods that we compare to during our study are not shown, so that we pick two most representative ones to present in this paper.

2. RELATED WORK

(ontology learning in folksonomies) Our work resembles that of ontology learning (OL) [4], where the objective is to construct a taxonomy of concepts as well as the relationship between them. Traditionally, OL is often studied in the context of natural language processing by annotating the concepts with labels, calculating the similarity between concepts via lexical and contextual similarities, and leveraging machine learning algorithms such as hierarchical clustering to construct a meaningful knowledge representation of concepts. Note that the labeling of terms in OL usually requires external resources. e.g., many of the existing approaches leveraged WordNet to collect hypernyms or synonyms for all terms in the documents [18]. Apparently, this guided approach is not suitable for tags since tags can be any arbitrary (novel) words which may not be found in WordNet or any other dictionaries. Comparatively, our work is a fully unsupervised approach that constructs the *tag ontology*

by exploring the correlations between tags in an automatic fashion, without any human-labeling or external knowledge.

A similar tree-based tag hierarchy was studied in [5], where the authors used Jaccard coefficient to measure tag similarity and proposed a maximum spanning tree (MST) algorithm for tag hierarchy construction. While similar to our approach, the authors did not consider regularization which could cause the tree to be unbalanced.

(tag visualization) The most prevalent format of tag visualization is called *tag cloud* which represents tags in an ordered or random list. Popular tags are usually highlighted with larger font or unique colors. This representation is simply yet quite intuitive, allowing users to quickly discover popular topics they are interested in. This format is used by most popular web2.0 websites like Delicious and Flickr.

An alternative view of tags that aims at visualizing tag topics over time was proposed in [7]. It was used for Flickr photo-tag visualization by associating photos with the most user-tagged words. The application offers two different views of photo-tag pairs: list view and stream view.

Neither of the aforementioned two approaches considered the relationships between tags. A closer proposal to our approach was found in [9], where the authors proposed a two-dimensional layout for improving the display of tags. In their proposal, both tag popularity and tag correlations are considered so that similar tags can be displayed close to each other. However, the levels of abstractness of different tags are still missing in their representation.

The closest approach to ours so far is the greedy algorithm from Heymann [10]. The authors combined graph centrality theorem with tag similarity measurement to derive a greedy hierarchical method for visualization. While these two approaches seem identical, the underlying principle is quite different, as we shall discuss in Section 3.2.5.

(tag recommendations) Besides discovering more effective tag representations, extensive research attempts have been made recently on automatically recommending relevant tags based on the content of the resources (e.g., web pages, images, videos and so on) or the user interests.

Most of the existing tag recommendations methods are algorithmically equivalent to collaborative filtering techniques, where the goal is to first retrieve a set of similar documents from the corpus given a target document, then use the existing tags from these documents for recommendations [16, 8, 3, 11]. A list of successful tag recommendation methods have been posted on ECML/PKDD Discovery Challenge site[3]. The measurement of similarity between tags is usually the tag co-occurrence, i.e., the number of times tags were used to tag the same document. Other metrics of similarity include tag correlations, random-walk-based similarity and etc.

Overall, existing approaches of tag recommendation mainly aim at improving the relevance of recommended tags while ignoring the underlying structure of the tag taxonomy. This is mainly due to the heterogenous nature of tags where there is no principled way of modeling tag abstractness. In what follows, we propose our framework to address this issue.

3. OUR APPROACH

This section describes our framework of hierarchical tag visualization which consists of mainly two steps: (1) rank the tags across all documents, in which we combine the power of two ranking methods: information-theoretic based ranking and learning-to-rank based ranking, and (2) construct the tag tree iteratively given the ranked list of tags, where minimum evolution (ME) criteria is used to greedily add tags to the tree, with minimum description length (MDL) standard is control the optimal shape of the tree.

3.1 Global Tag Ranking

The first step of our approach is to rank tags according to their level of abstractness as well as popularity. e.g., "*programming*" has more abstract meaning than "*java*" thus should be ranked higher so that "*programming*" can appear at a higher level in the tree. In this paper, we leverage the following features for constructing global tag ranking:

- tag entropy $H(t)$: the entropy of a tag is defined as $H(t) = -\sum_1^N p(t_i) \log p(t_i)$, where the summation is over all N topics that the tag t belongs to, and $\mathrm{p}(t_i)$ the probability that t appears in the i^{th} topic. Tag entropy is a measurement of *specificity* where more general tags like "*fun*" should have higher entropies because they might appear in different topics, while tags like "*javascript*" are often more specific to a topic thus have lower entropies. A similar metric was proposed in [11]. Following [16], the most frequent tag in a document is treated as a topic and we calculate the entropies of all tags on the top 100 topics[4].

- tag raw count $C(t)$: the total number of appearance of tag t in a specific corpus. This generally indicates how popular a tag is during a certain time period, which usually follows a well-known long tail distribution.

- tag distinct count $D(t)$: the total number of documents tagged by t. The tag raw count shows the overall popularity of a tag but does guarantee the influence of the tag in different documents. A tag may be tagged many time on a specific document but not on others. Thus the tag distinct count offers a popularity measurement from a document perspective of view.

3.1.1 Information-Theoretic (IT) Tag Ranking

With the three features defined above, we introduce the *informativeness* of a tag t as follows:

$$I(t) = \frac{H(t) \cdot [D(t) \cdot \log(C(t) + 1)]}{Z}, \qquad (1)$$

where the first term measures the specificity of t and the second term the popularity of t. Z is a normalization factor that ensures any $I(t)$ to be in (0,1). The second term is similar to the measurement of document-frequency and term-frequency in text retrieval applications. We empirically take the logarithm of $C(t)$ here to deemphasize its contribution since in practice, tag raw count is usually of several magnitudes larger than tag distinct count. A direct multiplication will thus over-credit $C(t)$. e.g., tag A is annotated 300 times but only in 2 documents, while tag B is annotated 50 times in 10 different documents. Directly multiply $C(t)$ and $D(t)$ would put tag A ahead of B but apparently B is more popular than A. Using our formula successfully eliminates this issue since tag B scores 56.72 while A only gets 16.46.

[3]http://www.kde.cs.uni-kassel.de/ws/dc09/

[4]Other numbers of topics are also tested which showed similar tag ranking results.

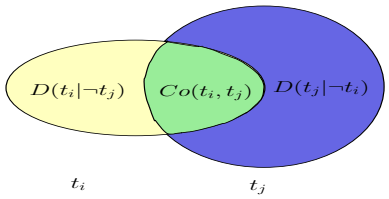


Figure 2: The relationship of two tags. Co is the co-occurrence. $D(t_i|\neg t_j)$ is the relative appearance.

While our information-theoretic approach bears some level of similarity with Heymann's entropy-based method [11], it should be noticed that the tag distinct count are apparently ignored in their approach. Since popular documents could potentially over-credit tags that appear many times in those documents but only a few times in less popular documents, it is important to consider both tag raw count and tag distinct count for a more accurate tag ranking.

3.1.2 Learning-to-rank (LETOR) based Tag Ranking

On the other hand, we construct an alternative tag ranking framework using the idea from learning-to-rank approach [14]. To be concrete, our training data are implicit feedbacks from users who performed tagging. Our objective is to learn an optimal weight vector for linear combination of features for ranking tags across all documents.

Figure 2 depicts a scenario of two tags' relationship. The middle part, $Co(t_i, t_j)$ indicates the number of times two tags t_i and t_j were used to tag same documents, while $D(t_i|\neg t_j)$ means the number of times t_i appeared in a document while t_j did not. Note that for any tag t_i, $D(t_i) = Co(t_i, t_j) + D(t_i|\neg t_j)$.

Intuitively, the relative occurrence of a tag t_i given the absence of another tag t_j is a strong indicator of higher abstractness of t_i than t_j. In our data set, we found that for example, $Co(programming, java) = 200$, $D(java|\neg programming) = 29$ and $D(programming|\neg java) = 239$. These statistics strongly indicates that "*programming*" is more abstract than "*java*", or say "*java*" is a kind of "*programming*".

Consequently, we can construct positive and negative training examples (x, y) automatically by discovering these pairs of tags from a given corpus $\mathcal{T}$. We construct

$$\begin{aligned} &\text{for each } \; i, j \in \mathcal{T}, i < j \\ &(x_k, y_k) = \begin{cases} (\{t_i - t_j\}, +1) & \text{if } \; t_i >_r t_j \\ (\{t_i - t_j\}, -1) & \text{if } \; t_i <_r t_j \\ \emptyset & \text{otherwise} \end{cases} \\ &>_r \; \text{is true when } D(t_i|\neg t_j)/D(t_j|\neg t_i) > \theta, \\ &<_r \; \text{is true when } D(t_i|\neg t_j)/D(t_j|\neg t_i) < 1/\theta. \end{aligned} \tag{2}$$

Here $\{t_i - t_j\}$ is the difference between feature vectors of two tags. In our case, we use the three features defined above, so $x_k \in \mathbb{R}^3$. And since we only consider binary classification, $y_k \in \{-1, +1\}$. Importantly, $>_r$ and $<_r$ define the pair-wise ranking preference. i.e., if the relative appearance of t_i is significantly more than t_j, we treated the pair $\{t_i - t_j\}$ as a positive example. Likewise, a negative example is constructed if t_i relatively appeared much less than t_j. For example, in the above case of "*programming*" (t_i) and "*java*" (t_j), $D(t_i|\neg t_j)/D(t_j|\neg t_i) = 8.24$ which we consider significant. In our experiment, we set $\theta = 2$ empirically.

IT Method	LETOR Method	Combination
web	reference	reference
tools	blog	tools
software	tool	web
web2.0	web	design
design	design	blog
reference	free	free
programming	cool	howto
blog	tutorial	tool
webdesign	news	software

Table 1: Top 10 ranked tags in Delicious using different ranking methods. IT = Information-theoretic, LETOR = Learning-to-rank. The combination uses a ratio of 0.5 as in eq.(5).

In our experiment, we constructed 532 training examples out of 3498 distinct tags in our experiment. For simplicity, we used a regularized logistic regression model for learning the optimal decision in this paper. Other advanced models such as Rank-SVM [14] remained to be explored in future work. Our model specifically maximizes the log-likelihood of the training data,

$$\begin{aligned} L(T) &= -\sum_{i=1}^{N} \log g(y_i z_i) + C \sum_{j=1}^{M} w_j^2, \\ \text{where } g(z) &= \frac{1}{1+e^{-z}}, \quad z_i = \sum_{m=1}^{M} w_m x_{im}. \end{aligned} \tag{3}$$

Here x_{im} is the value of the m^{th} feature of the i^{th} instance constructed using eq.(2), w_m is the weight value of that particular feature and the weight vector $\mathbf{w}$ is the parameter of the model. We used gradient decent to optimize $\mathbf{w}$ in model. After learning, we assign each tag a score

$$Lr(t_i) = w^T * t_i, \tag{4}$$

and normalize Lr to be within $(0, 1)$.

It should be noted that although learning-to-ranking often requires *supervised* learning methods, the way we derive *labels* in our approach does not require any explicit human annotations.

3.1.3 Combine Two Ranking Methods

Since the aforementioned two ranking methods address different perspectives of tag importance, it is natural to combine the power from both of them. A linear combination is proposed here to calculate the final score for each tag:

$$S(t) = rI(t) + (1 - r)Lr(t), r \in [0, 1]. \tag{5}$$

Tags are then ranked according to $S(t)$ in a descending order. Table 1 shows the top 10 ranked tags for each of the methods and the combination uses a ratio of 0.5. This parameter will be tuned in our experiment.

3.2 Constructing Tag Hierarchy

The proposed construction of tag hierarchy is an iterative process that contains two primary operations at each step: (1) select appropriate tags to be included in the tree[5], and (2) choose the optimal position for those tags.

[5]Our method can be applied to other tag ranking methods as well, e.g., FolkRank [12].

3.2.1 Iterative Tag Insertion

We propose a greedy algorithm that iteratively includes tags from the ranked tag list to the hierarchical representation. Following the minimum-evolution (ME) criteria in phylogentics [6], our objective is to minimize the change (cost) of an existing tree when adding a new tree node. As such, we define the cost of a tree R to be the sum of distances between every pair of the nodes:

$$Cost(R) = \sum_{i<j, t_i, t_j \in R} d(t_i, t_j), \tag{6}$$

where the distance between two nodes are the shortest path ($\hat{P}(t_i, t_j)$) that connects them, through their lowest common ancestor ($LCA(t_i, t_j)$),

$$d(t_i, t_j) = \sum_{e(i,j) \in \hat{P}(t_i, t_j)} W(e(i, j)). \tag{7}$$

For example, in Figure 3, $d(t_1, t_2) = 0.7, d(t_3, t_5) = 0.9$. To specify the distance between two tags, we define the edge weight between two directly-connected tag nodes i and j as the normalized correlations of the tags t_i and t_j, i.e.,

$$W(e(i, j)) = \exp(-\frac{||t_i - t_j||^2}{\sigma^2}), \tag{8}$$

where σ is a radius parameter.

In general, our algorithm works as follows. During each iteration, we greedily choose the highest-ranked tag for inclusion to form a new tree. It is worth notice that not all tags are included in the tree. We discard those tags that have zero correlations with all existing tree nodes. The entire algorithm is sketched as in Algorithm 1.

3.2.2 Optimal Position Selection

The number of potential positions that a new tag can be added is equivalent to the number of existing nodes in the tree, which could become quite large when the tree grows. To limit the search space for the optimal insertion position, we make a restriction here that a new node t_{new} can not be added at a higher level than the current leaf nodes, i.e., if the tree has depth $L(R)$, t_{new} can only be inserted at level $L(R)$ or $L(R)+1$. For example, in Figure 3, t_6 can only be added at the same level as of t_4 and t_5 or lower. Our restriction here is legitimate since the ranking of the tags are correlated to their abstractness, so that highly-ranked tags are more likely to be abstract. Under this guidance, we can reduce the search space significantly to $N(L(R) - 1) + N(L(R))$, where $N(i)$ indicates the number of nodes at level i.

Note that after the first step of tag ranking, more abstract tags are ranked at higher positions. Therefore, in our iteration insertion process, abstract tags will always be selected first and inserted at higher level of the tree, which makes our greedy selection process legitimate.

Once the search space is specified, it becomes easier to locate the optimal position for t_{new}. Mathematically, the optimal position for t_{new} thus can be found by minimizing the following criteria:

$$\begin{aligned} R_{new} &= \arg\min_{R'} \Delta Cost(R', R) = Cost(R') - Cost(R), \\ \text{where } R' &= R \cup t_{new}. \end{aligned} \tag{9}$$

The above optimization framework iteratively constructing an optimal tag hierarchy by adding one tag in each step.

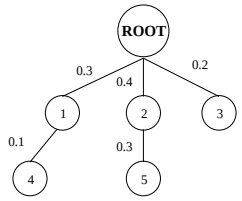


Figure 3: An example of tag representation of 5 tree nodes and a ROOT node.

This greedy method is quite efficient. Since in practice, only the cost difference ($\Delta Cost$) between two trees are required to be computed, the majority part of the tree remains unchanged so that the cost among the rest of the nodes usually does not change at all.

3.2.3 Model Selection via Minimum Description Length

There is an issue comes with the optimal selection criteria mentioned above. Since the nodes are chosen based on the co-occurrence with its parent node, it will then always have shorter distance to the parent node than others. Therefore, connecting the new node with its parent node always returns the optimal result. This could cause, in the worst case, all nodes to have only one child node while each level of the tree only contains one node (Such an example is shown in Figure 8(a), which will be discussed later).

To discourage such effort and make the tree more balance and meaningful, we introduce the idea of regularization, where we penalize the depth of the tree while minimizing the cost of adding new nodes.

This idea is similar to minimum description length (MDL) criteria [1], which has been widely used for model selection. The MDL principle states that the optimal model to encode a set of data essentially minimizes the number of bits that is required to transmit the data. To be specific, the MDL principle minimizes the objective function $L(M|D) + L(M)$, where the first term defines the explanation of the data D using model M, i.e., the likelihood. The second term corresponds to the length of the encoding under model M, i.e., model complexity. As a result, complex models are penalized. In our model, the first term controls the cost of introducing a new node, while the second term aims at reducing the height of the tree. i.e.,

$$\min \left(\Delta Cost(R', R) + \xi \frac{L(R')}{\log |R'|} \right), \tag{10}$$

where $L(R')$ and $|R'|$ indicate the depth and the total number of nodes in the new tree, respectively. Here ξ is a balancing factor between the two terms. The second term of eq(10), while penalizes the depth of the tree in general, encourages the increase of depth when the number of tree nodes at the same level becomes large. Since $L(R')$ remains a constant when the depth does not change, the insertion of new nodes will gradually increase the value of the denominator $\log |R'|$ and therefore decrease the value of the entire second term. Finally, the depth of the tree will increase when the second term exceeds a threshold.

3.2.4 Tree Initialization

We introduce a ROOT node for tree initialization. Because none of the top-ranked node can be treated as a parent node of all others as they all belong to different topics.

Algorithm 1 Tag Hierarchy Construction

Input Ranked tag list $T = \{t_1, ..., t_m\}$, tag correlation matrix $Col(T) \in \mathbb{R}^{m \times m}$
Initialize tag tree $R \leftarrow ROOT$
while $L(R) \leq 2$
  add the top tag t' from T to R according to eq.(10)
  update $W(e(ROOT, child))$ according to eq.(11)
  $T \leftarrow T \backslash t'$
  while sizeof$(T) > 0$
    extract top tag t' from T
    check the correlation of t' with R
    $\forall t_k \in R, Col(t_k, t') = 0$
      continue; (throw away t')
    else
      add t' to R according to eq.(10)
      $T \leftarrow T \backslash t'$
  end while
end while
Output R

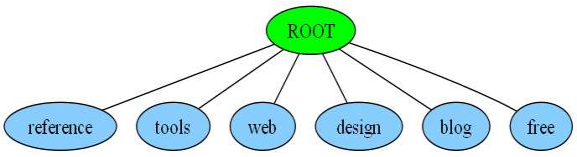


Figure 4: The ROOT node and the first level nodes in our data set. The first level corresponds to the highest level of topics for all tags.

Therefore, ROOT is a special node that controls the number of nodes at the initial level of the tree, which are *real* root nodes that break down the topics of a corpus at the highest level. Since there is no correlation between ROOT and other tags, we dynamically adjust the weights between ROOT and other tags according to the weights between its child nodes. We specify the distance to be equal to the maximum distance between any two child nodes,

$$W(e(ROOT, j)) = \max_{i<j, t_i, t_j \in R} W(e(i, j))$$
$$i, j \in \text{child}(ROOT). \quad (11)$$

Under this criteria, once the first level of nodes are fixed and the tree grows to deeper levels, the distance between ROOT and other child nodes will not change. In practice, it helps constructing the initial topics in the tree hierarchy. Figure 4 shows the first level of nodes in our data set. A total of 6 topics are specified at the first level.

3.2.5 Comparison with Heymann's Hierarchy

Heymann proposed a tag hierarchy construction algorithm in [10] by leveraging graph centrality theorem and tag similarity measurement. Their method greedy's choose the most central tags from tag graph and added to the hierarchy. A tag can only be added as (1) a child node of the most similar tag node or, (2) a root node, depending on the similarity threshold. While the graph centrality theorem bears similarity with our global tag ranking algorithm, there are several fundamental differences:

- A tag is often added as a child node to its most similar node according to Heymann's algorithm. While in our algorithm, it often results in a sibling node. While it is arguable which guideline is more reasonable, we have observed a much clearer hierarchy by using our algorithm in practice. For example, *maps* is at the same level as *map* in our algorithm but treated as child node in their approach, same for *network*, *networking* and many other scenarios.

- Heymann's algorithm does not penalize the height of the tree by controlling the number of nodes at the same level. In practice, we observed that their algorithm is more likely to grow a deep tree than having multiple root nodes, considering for a high similarity threshold. This results in a skewed tree where the hierarchy no longer makes much sense (as an example in Fig. 8 (a)).

4. APPLICATION TO TAG RECOMMENDATIONS

While our tag visualization method offers an alternative view of underlying tag relationships to the traditional tag could representation, its hierarchial structure also provides a potential opportunity of recommending relevant tags for annotation. Traditional approaches of tag recommendations are usually equivalent to the idea of collaborative filtering, in which documents having similar contents with the targeting document are selected and their tags are used for recommendations. Although some researchers have proposed methods for real-time tag recommendations which work well in terms of efficiency [16], the cost of training the model and maintaining large amount of model parameters in memory is still quite expensive and may not be suitable for web-scale applications. In this section, we propose to simplify this recommendation process by leveraging the existing hierarchical tag structures without training an expensive model.

In our proposal, recommending relevant tags to a targeting document is equivalent to (1) retrieve a candidate *tag* list (instead of retrieving a candidate *document* list in traditional way), and (2) rank retrieved tags given a scoring function in terms of document-tag relevance. Our framework breaks down the retrieval of candidate list into three scenarios which will be discussed in the next section. Given a targeting document d with its textual content, as well as a set of prior user tags $\{\tilde{t}_1, ...\tilde{t}_k\}$ (documents with completely no prior tag information will be discussed later). We introduce a scoring function to measure document-tag relevance:

$$Score(d, t_i | \{\tilde{t}_1, ...\tilde{t}_k\}) = \alpha \sum_{j=1}^{k} W(\tilde{t}_j, t_i) + (1 - \alpha) N(t_i, d), \quad (12)$$

where t_i is one of the candidate tags, $W(\tilde{t}_j, t_i)$ is the similarity between user-entered tag $\tilde{t}_j$ and t_i, $N(t_i, d)$ refers to the number of times tag t_i appears in document d. α is a balancing factor between two terms, so $\alpha \in [0, 1]$. This function addresses the similarity between candidate tags and user tags, as well as the relevance between candidate tags and the content of document d. Note that for applications like images and videos which does not contain textual contents, we could leverage the side-information such as user comments, related titles and so on, as proposed in [16]. Or we could simply set the balancing parameter $\alpha = 1$ to ignore the second term of eq.(12).

4.1 Candidates List Extraction

In the first scenario, we assume that document d only

Algorithm 2 Generating Candidate Tag List

Input tag tree R, target document d,
list of user-entered tags $T(d) = \{\tilde{t}_1, ...\tilde{t}_k\}$
Initialize candidate(d) $= \emptyset$
If sizeof(T) $= 0$
  $T(d)$ = top-K-words-in-$d \cap R$
end if
  for each $\tilde{t}_i \in T(d)$ $(i = 1...k)$
   do $t_p = \tilde{t}_i.parent$
    candidate(d).Add(t_p); $t_p = t_p.parent$;
   while($t_p \neq ROOT$)
  for each $t_c \in \tilde{t}_i.childs$
   candidate(d).Add(t_c)
  end for
  end for
Output candidate(d)[1...50]

has one user-entered tag t_d. This basically involves finding the location of t_d in the tree (or location of the most lexically-similar tag), and recommending nearby tags. As an example, for t_d = "*programming*", Figure 5 shows part of the subtree for "*programming*", as well as its path to the ROOT, where the numbers beside the edges indicate the number of tag co-occurrence. The challenge here is whether the child nodes (more specific tags, like "*java*", "*.net*") of t_d or parent nodes (more general tags, like "*development*") should have the priority to be included in the list. In our framework, we set the size of candidate list to be sufficiently large (we empirically include 50 tags in the candidate list) to include as many relevant tags as possible. We treat parent nodes with higher priority and first include them to the candidate list, then perform a breadth-first search (BFS) to include the most relevant child nodes.

For documents with multiple user tags, e.g. *technology* and *webdesign* in Figure 5, we include the joint set of their candidate lists, and output the first 50 with the highest correlations. As a result, their common ancestors, *programming* and *development* will be more likely to be included.

The most common scenario, however, is a document d without any prior knowledge of user-tags. In this case, we leverage the top K most frequent words from d that appear in our tag list, and treat them as *pseudo* tags to extract candidate list. So that for each pseudo tag, we retrieve its candidate tree and combine all candidates together. Algorithm 2 summarizes our method for these three scenarios.

Note that we've been quite generous towards the inclusion of candidate tags for recommendation, which is an arguable factor against efficiency. However, in fact, the correlations of tags are pre-computed and thus can be looked up in constant time. Constructing the word frequency table for a document with F words costs $O(F)$ time. For each candidate tag, the total cost of ranking using eq.(12) is merely $O(K + F + F \log F)$, where K is the number of user-entered tags. Since both K and F are quite small in practice, our framework is fully capable of performing real-time tag recommendation with very little memory and computational cost.

5. EXPERIMENTS

In this section, we present empirical results which address (1) the efficiency of the tag hierarchy on helping users navigate through a large set of tags, and (2) the effectiveness of our tag recommendation algorithm on choosing the most relevant tags for annotations. In our experiments, we mainly focus on user tagging experience on web documents, including News articles, blog articles and so on.

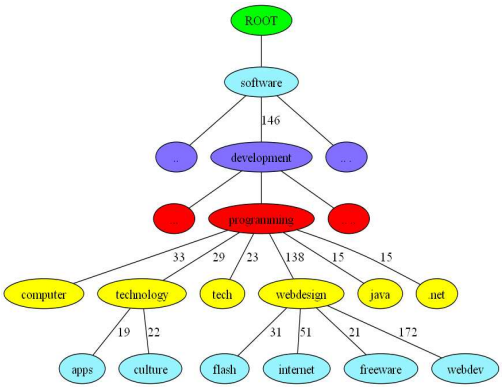


Figure 5: Subtree of *programming* are used for tag recommendation for programming related documents. The weights of the edge indicate the number of co-occurrence of two tags.

5.1 Data Preparation

The most well-known tag repository of web documents is the Delicious web site, which currently has more than 5 million registered users with over 150 million tagged web pages. We crawled Delicious for a period of two months data from May 2008 to July 2008. We first subscribe to the RSS feeds of the top 100 most popular tags in Delicious tag cloud, and continuously crawl these tags to get the associated URLs. For each individual URL, we also retrieve all user tags and download its HTML content. Finally, we ended up with 43,113 unique tags with 36,157 distinct URLs. The data set in evaluation is approximately 3.6 GB.

For user study purpose, we cannot enumerate all combination of the parameters. Therefore, we set both r and α in eq.(5) and eq.(12) to be 0.5, $\xi = 0.7$ in eq.(10). The sensitivity of parameters will be discussed later in this section.

5.2 User Study

As a preliminary user evaluation, we sought to compare the efficiency and effectiveness of tag hierarchies with tag clouds and Heymann's tag tree representation [10]. First, efficiency was being measured by three time-related metrics: (1) time-to-first-selection: The time between the timestamp from showing the page, and the timestamp of the first user tag selection; (2) time-to-task-completion: the time required to select all tags for the task; (3) average-interval-between-selections: the average time interval between adjacent selections of tags. An additional metric considered is (4) deselection-count: the number of times a user deselects a previously chosen tag and selects a more relevant one.

We conducted a between-subjects design with 49 participants where half of the users were presented with one tag visualization method and the remaining users were presented with the other. The 49 users were selected randomly from the information and computer science departments at a large university, and engineers at a large software company. Technical users were sought because of their expected background in tagging. Table 2 shows the basic user information. Note that the size of users in our study is sufficient,

Male	Female	Average Years Using Computer	Average Tagging Experiences
30	19	14.2	2.1

Table 2: Statistics about user study. Tagging experiences are rated in 4-point Likert scale: *none (0), a little (1), some (2)* and *a lot (3).*

which is supported by many previous literature [17] of user study, where much smaller sample sizes were used.

There were two phases to the experiment. In Phase I we compare our tag hierarchy and tag cloud. Users were asked to tag 10 random web documents from the Delicious data corpus. 15 tags were presented with each web document. Users were asked for select 3 tags that best described the web document. An example has been shown in Figure 1(b) at the beginning of the paper. Note that each user can only see either tag cloud or tag hierarchy. The 15 tags are selected based on Algo. 2. The *exactly* same set of tags are shown to users in both visualization methods so that quality of the recommended tags are same for all users. For the users chosen for tag cloud, we randomly serve them with (1) alphabetically ordered or (2) randomly ordered tag clouds. For the users chosen for tag hierarchy, we randomly serve them with (1) our result or (2) Heymann's approach [10].

5.3 Comparison of Tagging Efficiency

Table 3 and 4 list the results from Phase I study for all the 4 metrics. Overall, our method (MDL-Tree) outperforms all other three ones in all categories with statistical significance ($p < 0.01$). Some observations can be made: (1) users were faster in selecting tags using tag hierarchy than tag cloud. Both Heymann's tree and our method show significant less time in the time-to-task-completion metric. (2) assuming the two groups of users have similar reading speed, the time-to-first-selection metric indicates how fast the user can locate the most relevant tags once they finish reading. In this metric, we observe that our MDL-Tree requires the least time. Meanwhile, tag cloud organized alphabetically (Could-Alpha) also exhibits better performance than random tag cloud (Cloud-Rand) as well as Heymann's Tree. (3) Once the user locates the first tag, tag tree shows better performance in average-interval-between-selections than tag cloud thanks to the tree structure. The reason behind this phenomenon is quite obvious: without the hierarchical structure which groups similar tags together, users have to scan the tag cloud every time when they need to find another relevant tag, which also explains why the deselection-count of tag clouds are much higher than tag trees.

Furthermore, we asked users to rate their tagging experience on the following 4-point Likert scale: *none, a little, some* and *a lot*. By removing the sophisticated users in the "a lot" category, the average time for users to tag using the MDL-Tree visualization was 494 seconds versus 643 seconds for the tag cloud visualization (Cloud-Alpha), more than a 30% time advantage. Therefore, our experiment does show the potential for tag hierarchy visualization to be more efficient for tagging than the traditional tag clouds. Controlling for users' tagging experience can be incorporated in a subsequent experiment.

5.4 Evaluate Tag Recommendation Performance

In Phase II of our experiments, we evaluate the performance of our tag recommendation algorithm as in Algorithm 2 and eq.(12). For comparison, we implemented a content-based collaborative-filtering (CF) algorithm which is similar to one of the state-of-the-art P-Tag algorithm [3]. Since our candidate selection algorithm is a *content-free* recommendation method, we believe that the content-based CF method offers a strong baseline comparison. Specifically, the CF algorithm constructs a document-word matrix for all web pages in collection. For a new targeting web page, the algorithm finds the top m most similar web pages in collection by calculating the cosine similarities using word features. CF then recommends the most popular n tags within these m web pages to the targeting page. In our experiment, we fixed both m and n to be 5 for simplicity. Moreover, we also compare with one of the state-of-the-art algorithms in [16] which combines spectral clustering and mixture models for tag recommendation (noted as PMM).

To consider both the relevance of the recommended tags and their relative rankings, we use normalized discounted cumulative gain (NDCG) as the metric, which is a widely-used metric to evaluate the relevance of search engine result pages (SERP) given a set of user queries. We randomly sampled 10 pages from the data set and asked 49 users to measure the relevance of recommended tags. Each algorithm returns top 5 tags for relevance measurement. The tags are randomly ordered so that the judgers are not aware of the underlying algorithm or the relevance scores. Each tag is measured as one of the 5 levels: Perfect (score 5), Excellent (score 4), Good (score 3), Fair (score 2) and Poor (score 1). The NDCG of the tags given a web document with relevance scores $\{s(1), ...s(5)\}$ from a judge can be computed as $NDCG(w) = \frac{\sum_i (2^{s(i)}-1)/\log(i+1)}{Z(w)}$, where $Z(w)$ is a normalization factor corresponding to the ideal discounted cumulative gain, so that the $NDCG$ score will always be between 0 and 1. For each document, we average $NDCG$ from 49 judgements to get the average score.

Figure 7 demonstrates the strength of our method against the strong baseline approach and PMM. Overall, our algorithm beats the baseline in 9 out of 10 questions, and beats PMM in 8 out of 10 questions. We performed paired T-test with null hypothesis that both algorithm performs equally well. The alternative hypothesis is that our algorithm performs better in terms of NDCG (one-sided). The p-value from the statistical significance test is 0.001 comparing with the baseline, which indicates that the result is statistically significant. On average, our algorithm outscores the baseline by 21%. The p-value is 0.02 comparing with PMM–still shows statistically significance.

Furthermore, we break down the evaluation into different positions of recommendation. Here, we consider the quality of recommendation for the 1^{st} tag as well as top-5 tags. Therefore, we measure the NDCG score at position 1 and 5, respectively. Figure 9 summarizes the two cases. Our algorithm has 0.79 NDCG score at position 1, indicating that in 79% of times, the first recommended tag by our algorithm is the most relevant one among all. Our algorithm also outperforms the baseline and PMM in both positions.

5.5 Discussion of Parameter Sensitivity

The regularization parameter ξ in eq.(10) plays an important role in balancing the tree depth and the number of nodes at the same level. As mentioned above, without

#	Time-to-Task-Completion				Time-to-First-Selection			
	Cloud-Alpha	Cloud-Rand	Heymann-Tree	MDL-Tree	Cloud-Alpha	Cloud-Rand	Heymann-Tree	MDL-Tree
1	59.5 ± 38.06	62.5 ± 32.11	61.3 ± 20.82	59.5 ± 21.58	41.81 ±18.79	40.53 ±22.07	51.09 ±19.29	35.55 ±18.49
2	73.5 ± 52.3	77 ± 51.7	37.5 ± 19.37	33 ± 16.56	41.03 ±15.03	35.98 ±23.75	36.6 ±14.43	38.24 ±17.07
3	57.3 ± 45.61	53.2 ± 47.28	39.2 ± 15.88	35.3 ± 16.04	35.7 ±17.77	38.74 ±21.91	46.1 ±21.7	37.47 ±14.59
4	61 ± 24.47	60.1 ± 20.55	49.6 ± 21.05	42.7 ± 17.93	37.54 ±17.53	45.31 ±20.37	51.83 ±19.67	36.01 ±14.16
5	38.2 ± 19.82	39.5 ± 17.38	37.2 ± 14.90	33.3 ± 13.88	31 ±17.64	35.83 ±24.76	40.28 ±19.61	39.47 ±18.51
6	46.6 ± 22.94	44.5 ± 20.92	43.25 ± 28.21	41.8 ± 25.18	38.2 ±18.67	38.83 ±20.9	47.88 ±17.05	36.99 ±14.45
7	88.2 ± 45.83	92.1 ± 49.72	69.9 ± 43.21	46.5 ± 25.8	38.64 ±19.75	40.57 ±19.14	42.76 ±20.53	35.74 ±17.87
8	50.4 ± 33.03	53.1 ± 30.28	50.2 ± 28.98	43.4 ± 28.58	38.02 ±16.1	41.07 ±21.58	44.5 ±20.66	38.26 ±14.94
9	45.8 ± 19.7	47.1 ± 18.29	42.2 ± 15.63	33 ± 13.71	36.15 ±20.33	41.52 ±18.19	47.18 ±14.46	34.09 ±18.67
10	46 ± 19.84	45.8 ± 17.98	37.8 ± 19.02	29.8 ± 11.09	38.52 ±18.75	38.67 ±25.59	41.69 ±14.54	41.44 ±17
Ave	56.65	57.49	46.815	**39.83**	37.661	39.705	44.991	**37.326**

Table 3: Results of Time-to-Task-Completion and Time-to-First-Selection metrics. Our method is named MDL-Tree. Less time indicates better performance.

#	Average-Interval-Between-Selections				Deselection-Count			
	Cloud-Alpha	Cloud-Rand	Heymann-Tree	MDL-Tree	Cloud-Alpha	Cloud-Rand	Heymann-Tree	MDL-Tree
1	1.7 ±1.1	2.7 ±0.8	2 ±1.7	2.2 ±0.6	0.7	0.4	0.3	0.1
2	2.6 ±0.8	1.4 ±1.3	1.9 ±1.1	1.8 ±0.7	0.6	0.4	0.5	0.5
3	2.4 ±0.7	1.7 ±0.7	1.5 ±1.6	1.9 ±0.3	0.4	0.6	0.5	0.5
4	2.2 ±1.4	1.7 ±1.3	1.8 ±0.5	0.8 ±0.5	0.5	0.6	0.3	0
5	2.4 ±0.8	2.2 ±1	1.7 ±0.7	1.4 ±0.4	0.6	0.6	0.2	0.1
6	2.1 ±1.5	2.2 ±1.7	2.4 ±0.9	1.8 ±0.7	0.7	0.6	0.6	0.3
7	2.3 ±1	2.7 ±1.3	1.6 ±0.7	1.7 ±0.6	0.7	0.4	0.5	0.2
8	2.5 ±1.3	1.7 ±1.3	1.9 ±1.2	1.6 ±0.6	0.6	0.6	0.6	0
9	1.8 ±0.8	2.8 ±0.6	1.7 ±0.5	0.9 ±0.4	0.7	0.5	0.2	0.4
10	2.7 ±0.9	1.7 ±0.7	1.8 ±1.4	1.2 ±0.4	0.6	0.7	0.5	0.2
Ave	2.27	2.08	1.83	**1.53**	0.61	0.54	0.42	**0.23**

Table 4: Results of Average-Interval-Between-Selections and Deselection-Count metrics. Our algorithm (MDL-Tree) outperforms others in both criteria ($p < 0.05$).

regularization, the tree could grow infinitely deep without expanding its child nodes at any level. By penalizing the tree depth properly following the MDL criteria, our model is capable of minimizing this issue. Figure 8 shows two bad tag tree construction due to the improper use of regularization. As indicated in the left graph (a), if the regularization is not leveraged (or slightly used), the objective function of eq.(10) will choose to grow the tree first instead of penalizing its height. On the other hand, if we penalize the height too much as in (b), the tree stops growing deeper but keep adding nodes to the current level. We found the range of $[0.4, 0.7]$ best balancing these two factors. As our objective function in eq.(9) is equivalent to the maximum spanning tree (MST) used in [5], without proper regularization, MST is expected suffer the same problem in large taxonomy.

The next important parameter is the tuning parameter α in eq.(12), which controls the document-tag relevance by judging from two factors: the similarity between tags and the tag frequencies in the document content. To measure its sensitivity, we again resort to the $NDCG$ measurement used above. Figure 6 gives the contour plot of $NDCG$@5 as a function of α and ξ, where higher $NDCG$ scores indicate better performance. It can be observed that the highest $NDCG$ occurs when $\alpha = 0.7$ and $\xi = 0.5$. The best $NDCG$@5 is approximately 0.56. Recall that in our user study we set both parameters to be 0.5 which yielded a 0.51 $NDCG$, close to the optimal combination. Overall, when α is in [0.4, 0.8], ξ in [0.4, 0.7], our method performs well.

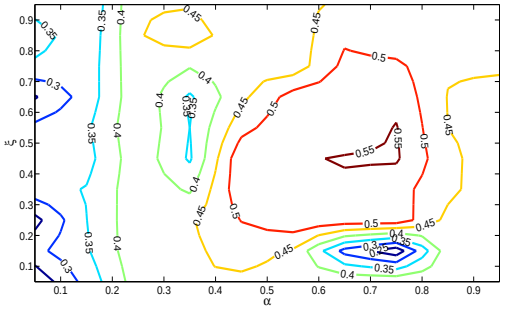


Figure 6: Parameter sensitivity of ξ and α in terms of $NDCG$ scores. The best combination is around $\alpha = 0.7$ and $\xi = 0.5$.

Besides, the combination parameter in eq.(5) that combines two ranking methods decides the global ranking of tags. However, we did not find this parameter to be sensitive in practice. Although the global ranking of a tag determines the order of its inclusion into the hierarchy, what really matters, however, is the relative order between two tags. e.g., as long as "*programming*" is included in the tag hierarchy prior to "*java*", Algorithm 1 can optimize the objective function and keep similar tags close with high probability. Due to space limitation this experiment is omitted here.

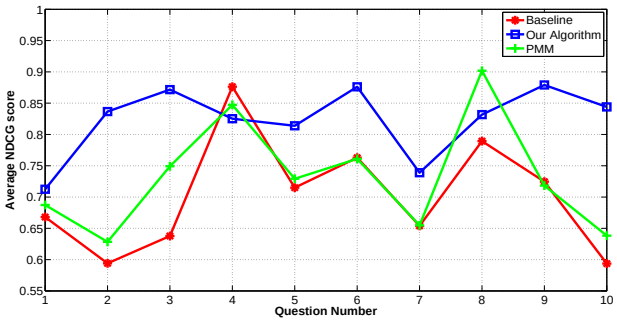
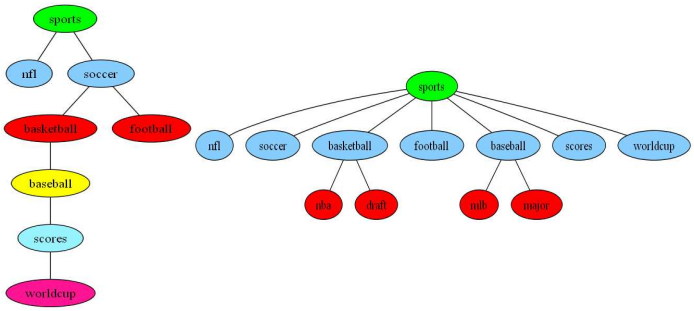


Figure 7: Comparison of algorithm performance in terms of NDCG scores. The improvement of our algorithm is significant to the baseline (p<0.001).



(a) (b)

Figure 8: Two bad examples of tree visualization without proper regularization. (a) $\xi = 0.1$. The tree grows too deep without expanding its children at the same level. (b) $\xi = 0.9$. The model penalizes too much on the depth and the tree grows too slowly.

6. CONCLUSION AND FUTURE WORK

In this paper we proposed a novel visualization of tag hierarchy which addresses two shortcomings of traditional tag clouds: (1) unable to capture the similarities between tags, and (2) unable to organize tags into levels of abstractness. Our proposed method mimicked the principle of minimum evolution criteria and constructed tag hierarchy to overcome previous issues. A greedy algorithm was proposed to iterative select the best tags to be added in the tree. Along with the regularization method, our framework guaranteed a balanced tag hierarchy with well-organized topics.

We then proposed a taxonomy-based tag recommendation method which leveraged the tag hierarchy to find relevant tags. User study and quantitative analysis indicated that our visualization method can reduce the tagging time, especially for users with limited tagging experiences. Also, our tag recommendation algorithm outperformed a content-based recommendation method in NDCG scores.

In the future, we plan to investigate more advanced ranking methods for global tag ranking and compare with existing methods as well [12]. Document-specific tag hierarchy and personalized tag hierarchy are also under consideration.

7. REFERENCES

[1] A. Barron, J. Rissanen, and B. Yu. The minimum description length principle in coding and modeling (invited paper). pages 699–716, 2000.

[2] G. Begelman. Automated tag clustering: Improving search and exploration in the tag space. In *In Proc. of the Collaborative Web Tagging Workshop at WWWŠ06*, 2006.

[3] P. A. Chirita, S. Costache, W. Nejdl, and S. Handschuh. P-tag: large scale automatic generation of personalized annotation tags for the web. In *WWW '07*, pages 845–854.

[4] P. Cimiano, A. Hotho, and S. Staab. Learning concept hierarchies from text corpora using formal concept analysis. *Journal of Artificial Intelligence research*, 24:305–339, 2005.

[5] P. De Meo, G. Quattrone, and D. Ursino. Exploitation of semantic relationships and hierarchical data structures to support a user in his annotation and browsing activities in folksonomies. *Inf. Syst.*, 34(6):511–535, 2009.

[6] R. Desper and O. Gascuel. Fast and accurate phylogeny reconstruction algorithms based on the minimum-evolution principle. *Journal of Computational Biology*, 9(5):687–705.

[7] M. Dubinko, R. Kumar, J. Magnani, J. Novak, P. Raghavan, and A. Tomkins. Visualizing tags over time. In *WWW '06*, pages 193–202, New York, NY, USA, 2006.

[8] S. A. Golder and B. A. Huberman. Usage patterns of collaborative tagging systems. *J. Inf. Sci.*, 32(2):198–208, 2006.

[9] Y. Hassan-montero and V. H. solana A. Improving tag-clouds as visual information retrieval interfaces. In *Merída, InSciT2006 conference*, 2006.

[10] P. Heymann and H. Garcia-Molina. Collaborative creation of communal hierarchical taxonomies in social tagging systems. Technical Report 2006-10, April 2006.

[11] P. Heymann, D. Ramage, and H. Garcia-Molina. Social tag prediction. In *SIGIR '08*, pages 531–538, 2008.

[12] A. Hotho, R. Jäschke, C. Schmitz, and G. Stumme. Folkrank: A ranking algorithm for folksonomies. In *University of Hildesheim, Institute of Computer Science*, pages 111–114, 2006.

[13] D. Liu, X.-S. Hua, L. Yang, M. Wang, and H.-J. Zhang. Tag ranking. In *WWW '09*, pages 351–360, 2009.

[14] T.-Y. Liu. Learning to rank for information retrieval. pages 225–331, 2009.

[15] B. Markines, C. Cattuto, F. Menczer, D. Benz, A. Hotho, and G. Stumme. Evaluating similarity measures for emergent semantics of social tagging. In *WWW '09*, pages 641–650, New York, NY, USA, 2009. ACM.

[16] Y. Song, Z. Zhuang, H. Li, Q. Zhao, J. Li, W.-C. Lee, and C. L. Giles. Real-time automatic tag recommendation. In *SIGIR '08*, pages 515–522, New York, NY, USA, 2008.

[17] M. Toomim, S. M. Drucker, M. Dontcheva, A. Rahimi, B. Thomson, and J. A. Landay. Attaching ui enhancements to websites with end users. In *CHI '09*, pages 1859–1868.

[18] H. Yang and J. Callan. Metric-based ontology learning. In *ONISW '08*, pages 1–8. ACM, 2008.

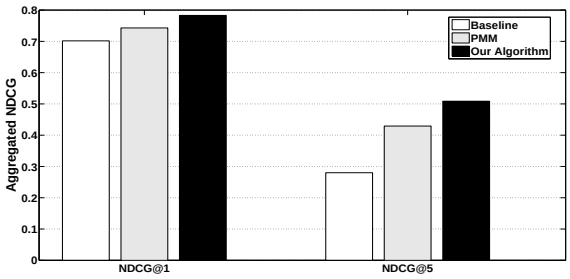


Figure 9: The performance of tag recommendation at position 1 and position 5.