Constructing Folksonomies by Integrating Structured Metadata with Relational Clustering

Anon Plangprasopchok and **Kristina Lerman**
USC Information Sciences Institute
Marina del Rey, CA

Lise Getoor
Department of Computer Science
University of Maryland, College Park

Abstract

Many social Web sites allow users to annotate the content with descriptive metadata, such as tags, and more recently also to organize content hierarchically. These types of structured metadata provide valuable evidence for learning how a community organizes knowledge. For instance, we can aggregate many personal hierarchies into a common taxonomy, also known as a folksonomy, that will aid users in visualizing and browsing social content, and also to help them in organizing their own content. However, learning from social metadata presents several challenges: sparseness, ambiguity, noise, and inconsistency. We describe an approach to folksonomy learning based on relational clustering that addresses these challenges by exploiting structured metadata contained in personal hierarchies. Our approach clusters similar hierarchies using their structure and tag statistics, then incrementally weaves them into a deeper, bushier tree. We study folksonomy learning using social metadata extracted from the photo-sharing site Flickr. We evaluate the learned folksonomy quantitatively by automatically comparing it to a reference taxonomy. Our empirical results suggest that the proposed framework, which addresses the challenges listed above, improves on existing folksonomy learning methods.

Introduction

The social Web has changed the way people create and use information. Sites like Flickr, Del.icio.us, YouTube, and others, allow users to publish and organize content by annotating it with descriptive keywords, or tags. Some web sites also enable users to organize content hierarchically. The photo-sharing site Flickr, for example, allows users to group related photos in sets, and related sets in collections. Although these types of social metadata lack formal structure, they capture the collective knowledge of Social Web users. Once extracted from the traces left by many users, such collective knowledge will add a rich semantic layer to the content of the Social Web that will potentially support many tasks in information discovery, personalization, and information management.

A community's knowledge can be expressed through a common taxonomy, also called a *folksonomy*, that is learned from social metadata created by many users. Compared to existing classification hierarchies, such as Linnaean classification system or WordNet, automatically learned folksonomies are attractive because they are (1) created from collective agreement of many individuals; (2) relatively inexpensive to obtain; (3) can adapt to evolving vocabularies and communitys information needs; and (4) they are directly tied to the annotated content. A learned folksonomy can facilitate users in browsing content produced collectively by the community. It can also help users visualize where their own content fits within the communitys and aid them in organizing it.

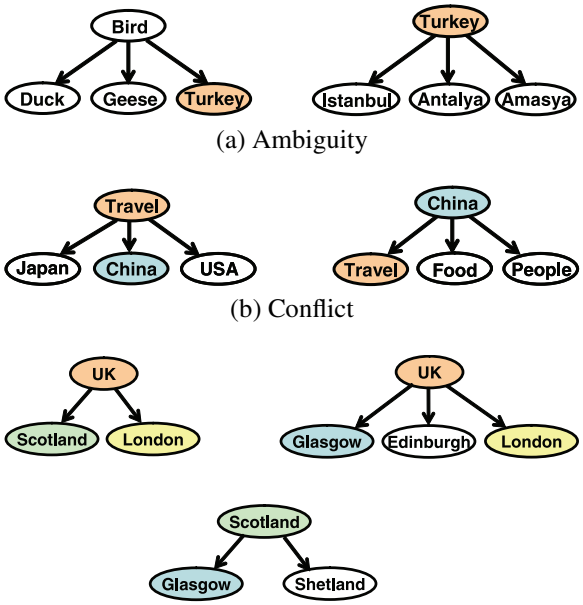
Learning a global folksonomy comes with a number of challenges which arise when integrating structured metadata created by diverse users, with each user freely annotating data according to her own preferences. Consequently, social metadata is *noisy*, *shallow*, *sparse*, *ambiguous*, *conflicting*, *multi-faceted*, and expressed at *inconsistent granularity* levels across many users.

We propose a novel approach to learn folksonomies by exploiting structured social metadata in the form of tags and user-specified shallow hierarchies. Our approach is driven by a similarity measure that utilizes statistics of both kinds of metadata to incrementally weave individual hierarchies into a deeper, more complete global folksonomy. We provide empirical evaluation to demonstrate that the proposed approach improves on the previous method to learn a folksonomy from user-specified relations.

Structured Social Metadata

Structured data in a form of shallow hiearchies is ubiquitous on the Social Web. On Flickr, users can arbitrarily group related photos into sets and then group related sets in collections. Some users create multi-level hierarchies containing collections of collections, etc., but the vast majority of users who use collections create shallow hierarchies, consisting of collections and their constituent sets. These personal hierarchies generally represents subclass and part-of relationships.

We formally define a *sapling* as a shallow tree representing a personal hierarchy which composed of a root node r^i and its children, or leaf, nodes $\langle l^i_1, ..l^i_j \rangle$. The root node corresponds to a user's collection, and inherits its name, while the leaf nodes correspond to the collection's constituent sets and inherit their names. We assume that hierarchical relations



(a) Ambiguity

(b) Conflict

(c) Varying granularity

Figure 1: Schematic diagrams of personal hierarchies created by Flickr users. (a) Ambiguity: the same term may have different meaning ("turkey" can refer to a bird or a country). (b) Conflict: users' different organization schemes can be incompatible (`china` is a parent of `travel` in one hierarchy, but the other way around in another). (c) Granularity: users have different levels of expressiveness and specificity, and even mix different specificity levels within the same hierarchy (`Scotland` (country) and `London` (city) are both children of `UK`). Nodes are colored to aid visualization.

between a root and its children, $r^i \rightarrow l^i_j$, specify broader-narrower relations.

Flickr users can attach tags only to photos. A sapling's leaf node corresponds to a set of photos, and the tag statistics of the leaf are aggregated from that set's constituent photos. Tag statistics are then propagated from leaf nodes to the parent node. We define a tag statistic of node x as $\tau_x := \{(t_1, f_{t_1}), (t_2, f_{t_2}), \cdots (t_k, f_{t_k})\}$, where t_k and f_{t_k} are *tag* and its frequency respectively. Hence, τ_{r^i} is aggregated from all $\tau_{l^i_j}$s. These tag statistics can also be used as a feature for determining if two nodes are similar (of the same concept).

Challenges in Learning from Structured Metadata

Learning folksonomies from social metadata, specifically, from structured metadata, presents a number of challenges. We divide these challenges into five categories, as described below.

Sparseness: Social metadata is usually very sparse. Users provide 4 –7 tags per bookmark on *Delicious* (Plangprasopchok and Lerman 2010) and 3.74 tags per photo on *Flickr* (Rattenbury, Good, and Naaman 2007). Sparseness is also manifested in the hierarchical organization created by an individual. In our Flickr data set, we found only 600 out of 21,792 users — approximately 0.02 percent — who created multi-level (collections of collections) hierarchies. Most users define *shallow* (single-level) hierarchies. Among these shallow hierarchies, moreover, few users express "similar" organization. For instance, of the 433 users who created an `animal` collection, only a few created common child sets, such as `bird`, `cat`, `dog` or `insect`. In order to learn a deeper, more complete folksonomy, we have to aggregate social metadata from many different users.

Noisy vocabulary: There are two main types of noise in social metadata: vocabulary and structure noise, which we explain later in this section. Vocabulary noise has several sources. One common source is variations and errors in spelling. We can mitigate some of the noise by lowercasing and stemming terms in metadata. Noise also arises from users' idiosyncratic naming conventions. While idiosyncratic terms may be meaningful to image owner and her narrow interest group, they are relatively meaningless to other users. We consider as noise tags such as `not sure`, `pleaseaddthistothethemecomppoll`, `mykid4` or an image
owner's name, often added as a tag.

Ambiguity: An individual tag is often ambiguous (Mathes 2004; Golder and Huberman 2006). For example, the tag `jaguar` can be used to refer to either a mammal or a luxury car. Similarly, terms that are used to name photo collections and sets can refer to different concepts. Consider the hierarchy in Figure 1 (a), where a collection `turkey` could be about a bird or a country. Similarly, `victoria` can either be a place in Canada or Australia. When combining metadata to learn common folksonomies, we need to be aware of its meaning. Contextual information may help disambiguate metadata.

Structural noise and conflicts: Like vocabulary noise, structural noise has a number of sources and can lead to inconsistent or conflicting structures. Structural noise can arise as a result of variations in individuals' organization preferences. Suppose that, as shown in Figure 1 (b), user A organizes photos first by activity, creating a collection called travel, and as part of this collection, a set called china, for photos of her travel in China. Meanwhile, user B organizes photos by location first, creating a collection china, with constituent sets travel, people, food, etc. In one hierarchy, therefore, `travel` is more general than `china`, and in the second hierarchy, it is the other way around.

Sometimes conflicts are caused by vocabulary differences among individual users. For example, to some users a `bug` is a "pest," a term broader than `insect`, but to others it is a subclass of `insect`. As a result, some users may express `bug` → `insect`, while the others express an inverse relation. An automatic folksonomy integration should be able to deal with these types of conflicts. Another source of structural noise is variations in degree of knowledge about a topic. Many users, for example, assemble images of spiders in a set called spiders and assign it to an insect collection, while others correctly assign spiders to arachnid.

Varying granularity level: Differences in users' level of expertise and expressiveness may lead to relatively impre-

cise metadata. Experts may use specific breed names to tag dog photos, while non-experts will simply use the tag `dog` to annotate them(Golder and Huberman 2006). In addition, one user may organize photos first by country and then by city, while another organizes them by country, then subregion and then city, as shown in Figure 1 (c). Combining data from these users potentially generate multiple paths from one concept to another.

Learning Folksonomies from Structured Metadata with Relational Clustering

We propose a simple, yet effective approach to combine many personal hierarchies into a global folksonomy that also takes the above challenges into account. Basically, we need to aggregate saplings *both* horizontally and vertically. By horizontal aggregation,we mean merging saplings with similar roots, which expands the breadth of the learned tree by adding leaves to the root. By vertical aggregation, we mean merging one saplings leaf to the root of another, extending the depth of the learned tree. The approach we use exploits contextual information from neighbors in addition to local features to determine which saplings to merge. The approach is similar to relational clustering (Bhattacharya and Getoor 2007) and its basic element is the similarity measure between a pair of nodes.

We define a similarity measure between nodes in different saplings, which combines heterogeneous evidence available in the structured social metadata, and is a combination of *local similarity* and *structural similarity*. The local similarity between two nodes a and b, $localSim(a,b)$, is based on the intrinsic features of a and b, such as their names and tag distributions. The structural similarity, $structSim(a,b)$ is based on features of neighboring nodes. If a is a root of a certain sapling, its neighboring nodes are all of its children. If a is a leaf node, the neighboring nodes are its parent and siblings. The similarity between nodes a and b is:

$$\begin{aligned} nodesim(a,b) &= \alpha \times localSim(a,b) \\ &+ (1-\alpha) \times structSim(a,b), \end{aligned} \quad (1)$$

where $0 \leq \alpha \leq 1$ is a weight for adjusting contributions from $localSim(,)$ and $structSim(,)$.

Local Similarity The local similarity of nodes a and b, has two components: (1) name similarity, and (2) tag distribution similarity. Name similarity can be any string similarity metric, which returns value ranging from 0 to 1. For tag similarity, it can be a simple function that, e.g., checks the number of common tags in the top `K` tags of a and b, and returns 1 if this number is greater than `J`; otherwise, it returns 0. We use a logistic function to smooth the step function. The local similary, which is a weighted combination between name and tag similarities, is defined as follows:

$$\begin{aligned} localSim(a,b) &= \beta \times nameSim(a,b) \\ &+ (1-\beta) \times \frac{1}{1+e^{-2(|\tau_a \cap \tau_b| + \mathtt{K})}}. \end{aligned} \quad (2)$$

Tag similarity helps address the *ambiguity* challenge described earlier. For example, top tags of the node `turkey` that refers to a type of bird include "bird", "beak", "feed", while top tags of `turkey` that refers to a country include different terms about places and attractions within this country.

Structural Similarity The structural similarity between two nodes depends on position of the nodes within the saplings. We define two versions: $structSimRR(,)$ which computes the structural similarity between two root nodes (root-to-root similarity), and $structSimLR(,)$ which evaluates structural similarity between a root of one sapling and the leaf of another (leaf-to-root similarity).

Root-to-Root similarity: Two saplings A and B are likely to describe the same concept if their root nodes r^A and r^B share the same (stemmed) name, and some of their leaf nodes also have the same names. In this case, there is no need to compute $tagSim(,)$ of these leaf nodes. Structural similarity between two root nodes is defined as follows:

$$\begin{aligned} & structSimRR(r^A, r^B) \\ = \; & max \left\{ \begin{array}{l} \frac{1}{Z} \sum_{i,j} \delta(stem(l_i^A), stem(l_j^B)) \\ tagsim(\grave{\mathbb{L}}_{tag}^A, \grave{\mathbb{L}}_{tag}^B) \end{array} \right. \end{aligned} \quad (3)$$

where $\delta(.,.)$ returns 1 if the both arguments are exactly the same; otherwise, it returns 0. $stem(l_i^A)$ is a function that returns stem name of a leaf node l_i^A of sapling A, and $\grave{\mathbb{L}}_{tag}^A$ is an aggregation of tag distributions of all l_i^A, at which $stem(l_i^A) \neq stem(l_j^B)$ for any leaf node l_j^B of the sapling B. Meanwhile, $\grave{\mathbb{L}}_{tag}^B$ is defined in the similar way. In particular, we compute similarity based on: (1) how many of their children have common stem name (they match); (2) the tag distribution similarity of those that do not have the same name. The second one is an optimistic estimate that child nodes of the two saplings refer to the same concept while having different names. We simply use $max(,)$ function to choose the higher score.

The normalization coefficient, Z, can be $min(|l^A|, |l^B|)$, where $|l^A|$ is a number of child nodes of A. We use $min(,)$ instead of union. The reason is that saplings aggregated from many small saplings will contain a large number of child nodes. When merging with a relatively small sapling, the fraction of common nodes may be very low compared to total number of child nodes. Hence, the normalization coefficient with the union ($Z = union(l^A, l^B)$), as defined in Jaccard similarity, results in penalizing small saplings too much. $min(,)$, on the other hand, seems to correctly consider the proportion of children of the smaller sapling that overlap with the larger sapling.

When we decide that root r^A of sapling A and r^B of sapling B are similar, we merge A and B with the $mergeByRoot(A,B)$ operation. This operation creates a new sapling, M, which combines structures and tag statistics of saplings A and B. Particularly, the tag statistics of the root of M is a combination of those from r^A and r^B. The leaves of M, l^M, are from a union of l^A and l^B. If there are leaves from A and B that share the same stemmed name, their tag statistics will be combined and attached to the corresponding leaf in M.

The width of the newly merged sapling will increase as more saplings are merged. Also, since we simply merge leaf

nodes with similar names, and their roots also have similar names, leaf-to-leaf structural similarity $structSimLL(,)$ is not required. This operation addresses the *sparseness* challenge.

Root-to-Leaf similarity Merging the root node of one sampling with the leaf node of another sapling is an important operation for extending the depth of the learned folksonomy. Since we consider a pair of nodes with different roles, their neighboring nodes also have different roles. This would appear to make them structurally incompatible. However, in many cases, some overlap between siblings of one sapling and children of another sapling exists. Formally, suppose that we are considering similarity between leaf l_i^A of sapling A and root r^B of sapling B. There might be some $l_{k\neq i}^A$ of A similar to l_j^B of B. Consider Figure 1 (c). Suppose that we have already merged `uk` saplings. Now, there are two saplings `uk` $\rightarrow$ {`scotland`, `glasgow`, `edinburgh`, `london`} and `scotland` $\rightarrow$ {`glasgow`, `shetland`}, and we would like to merge the two `scotland`s. Since both `uk` and `scotland` saplings have `glasgow` in common, and the user placed `glasgow` under `uk` instead of `scotland`, this *shortcut* contributes to the similarity between `scotland` nodes. The structural similarity between leaf and root nodes that takes this type of shortcut into consideraion is:

$$structSimLR(l_i^A, r^B) = structSimRR(r^A, r^B). \quad (4)$$

Specifically, this is simply the root-to-root structural similarity of r^A and r^B, which measures overlap between siblings of l_i^A and children of r^B. For the case when there is no shortcut, the similarity from this part will be dropped out; hence, the Eq. 1 will only be based on the local similarity.

SAP: Growing a Tree by Merging Saplings

Now that we have describe the similarity scores and the basic merge operations, we next introduce SAP, our algorithm which uses the operations defined above to incrementally grow a deeper, bushier tree by merging saplings created by different users. In order to learn a folksonomy corresponding to some concept, we start by providing a seed term, the name of that concept. The seed term will be the root of the learned tree. We cluster individual saplings whose roots have the same name as the seed by using the similarity measures Eq. 1, Eq. 2 and Eq. 3 to identify similar saplings. Saplings within the same cluster are merged into a bigger sapling using the $mergeByRoot(,)$ operation. Each merged sapling corresponds to a different sense of the seed term.

Next, we select one of the merged saplings as the starting point for growing the folksonomy for that concept. For each leaf of the initial sapling, we use the leaf name to retrieve all other saplings whose roots are similar to the name. We then merge saplings corresponding to different senses of this term as described above. The merged sapling whose root is most similar to the leaf (using similarity measures Eq. 1, Eq. 2 and Eq. 4), is then linked to the leaf. In the case that several saplings match the leaf, we merge all of them together before linking. Clustering saplings into different senses, and then merging relevant saplings to the leaves of the tree proceeds incrementally until some threshold is reached.

Suppose we start with saplings shown in Figure 1(c), and the seed term is `uk`. The process will first cluster `uk` saplings. Suppose, for illustrative purposes, that there is only one sense of `uk`, resulting in a single sapling with root `uk`. Next, the procedure selects one of the unlinked leaves, say `glasgow`, to work on. All saplings with root `glasgow` will be clustered, and the merged `glasgow` sapling that is sufficiently similar to the `glasgow` leaf of the `uk` sapling will then be linked to it at the leaf, and so on.

Handling Shortcuts Attaching a sapling A to the learned tree F can result in structural inconsistencies in F. One type of inconsistency is a *shortcut*, which arises when a leaf of A is similar to a leaf of F. In the illustration above, attaching the `scotland` sapling to the `uk` tree will generate a shortcut, or two possible paths from `uk` to `glasgow` ($r^{uk} \rightarrow l^{uk}_{glassgow}$ and $r^{uk} \rightarrow l^{uk}_{scotland} \rightarrow l^{scotland}_{glasgow}$). Ideally, we would drop the shorter path and keep the longer one which captures more specific knowledge.

There are cases where the decision to drop the shorter path cannot be made immediately. Suppose we have `uk` $\rightarrow$ {`london`, `england`, `scotland`} as the current learned tree is about to attach `london` $\rightarrow$ {`british museum`, `dockland`, `england`} sapling to it. Unfortunately, some users placed `england` under `london`, and attaching this sapling will create a shortcut to `england`. The decision to eliminate the shorter path to `england` cannot be made at this point, since we have no information about whether attaching the `england` sapling will also create a shortcut to `london` from the root (`uk`). We have to postpone this decision until we retrieve all relevant saplings that can be attached to the present leaf (l^{uk}_{london}) and its siblings ($l^{uk}_{england}$ and $l^{uk}_{scotland}$).

Suppose that $l^{uk}_{england}$ does match the root of sapling `england` $\rightarrow$ {`london`, `manchester`, `liverpool`}. Mutual shortcuts to `england` and `london` would undesirably appear once all the saplings are attached to the tree. Hence, the decision to drop $l^{uk}_{england}$ or l^{uk}_{london} must be made. We base the decision on similarity. Intuitively, a sapling that is more similar, or "closer," to r^{uk} should be linked to the tree. Formally, the node to be kept is $l^{uk}_{\hat{x}}$ where $\hat{x} = argmax_x\{nodesim(r^{uk}, r^x)\}$ and $x = \{england, london\}$, while the other will be dropped. This is illustrated in Figure 2.

Handling Loops Attaching a sapling to a leaf of the learned tree F can result in another undesirable structure, a *loop*. Suppose that we are about to attach a sapling A to the leaf l_i^F of F. A loop will appear if there exists a leaf l_j^A of A with the same name as some node in the path from root to l_i^F in F. In order to make the learned tree consistent, we must remove l_j^A before attaching the sapling. For instance, suppose we decide to attach `London` sapling to

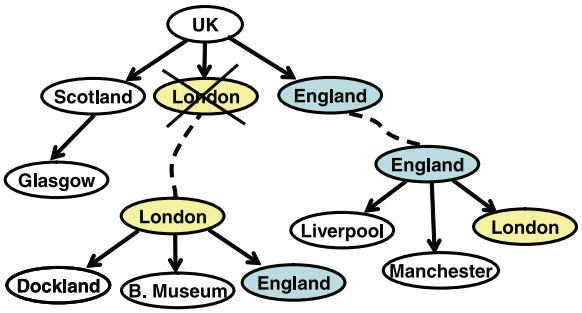


Figure 2: Appearance of mutual shortcuts between London and England when merging London and England saplings. To resolve them, we compare the similarity between UK-London and UK-England sapling pairs. Since England sapling is closer to UK than London sapling, we simply attach England sapling to the tree; while ignoring London leaf under UK.

the `England` sapling in Figure 2 at its `London` node, we have to remove `England` node of `London` sapling first.

In some cases, loops indicate synonymous concepts. In our data set, we found that there are users who specify the relation `animal` → `fauna`, and those who specify the inverse `fauna` → `animal`. Since `animal` and `fauna` have similar meaning, we hypothesize that this conflict appears because of variations in users' expertise and categorization preferences.

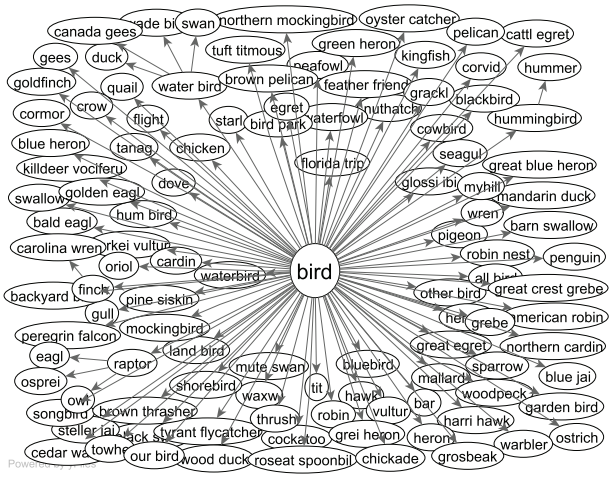
To determine whether a loop is caused by a synonym, we check the similarity between r^A and r^F. If it is high enough, we simply remove l_j^F from F, for which $stem(l_j^F) = stem(l_j^A)$; then, merge r^A and r^F. The similarity measure is based on Eq. 1. More stringent criteria are required since r^A and r^F have different names. Specifically, we modify $tagSim(X,Y)$ to $tagSim^{syn}(X,Y)$, which instead evaluates $\frac{|\tau_X \cap \tau_Y|}{min(|\tau_X|,|\tau_Y|)}$, and modify $structSim(X,Y)$ to $structSim^{syn}(X,Y)$, which only evaluates $\frac{1}{Z}\sum_{i,j}\delta(stem(l_i^X), stem(l_j^Y))$.

Mitigating Other Structural Noise The similarity measure between root-to-leaf defined earlier is only based on contextual information from adjacent saplings. Hence, at a distant leaf node, far from the root of the tree, the measure may consider merging some sapling sense, that is relevant to the leaf, but irrelevant to the tree root. To illustrate, suppose we have the following hierarchy, `flower` → `rose` → `black & white`. There is a chance that the sapling, `black & white` → {`macro`, `portrait`, `landscape`} will be judged relevant to the leaf `white` of the tree, since they share enough common tags such as `macro`, `white`, etc. When deciding to attach this sapling to the tree, we could end up with a tree that mixes concepts from "flower" and "portraiture."

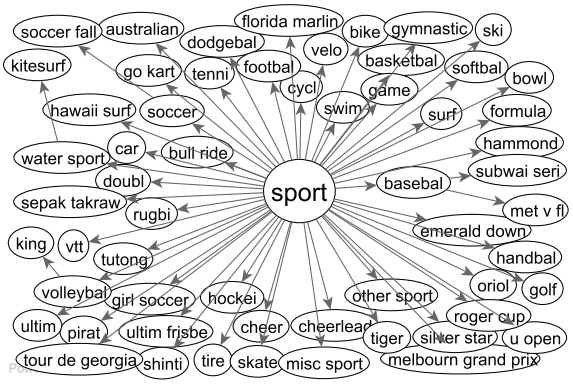
We use a *continuity measure* to check whether the sense of the sapling we are considering attaching is relevant to the ancestors of the leaf. Recall that the root node inherits tags from all of its decendents. We examine the tag overlap and do not attach the sapling if it has less than `L` tags in common with the grand parent node. In addition, we only attach new saplings to leaf nodes which are the result of input from more than one user.

Mitigating Noisy Vocabularies As mentioned earlier, noisy nodes appear from idiosyncratic vocabularies, used by a small number of users. For a certain merged sapling, we can identify these nodes by the number of users who specified them. Specifically, we use 1% of the number of all users who "contribute" to this merged sapling as the threshold. We then remove leaves of the sapling, that are specified by fewer number of users than the threshold.

Empirical Results



(a)



(b)

Figure 3: Folksonomies learned for (a) `bird` and (b) `sport`

We constructed a data set containing collections and their constituent sets (or collections) created by a subset of Flickr users who are members of seventeen public groups devoted to wildlife and nature photography (Plangprasopchok and

Lerman 2009). These users had many other common interests, such as travel and sports, arts and crafts, and people and portraiture. We extracted all the tags associated with images in the set, and retrieved all other images that the user annotated with these tags. We constructed personal hierarchies, or saplings, from this data, with each sapling rooted at one of user's top-level collections. We ignore collections with composite names. This reduces the size of the data set to $20,759$ saplings created by $7,121$ users. A small number of these saplings are multi-level.

The folksonomy learning approach described in this paper, namely SAP, has a number of parameters. We explored a range of parameter values; due to space limitations, we do not include the complete set of results. Here, we report results obtained through a combination that resulted in good performance: we set $\alpha = 0.4$ the weight of the structural similarity for root-to-root, and 0.6 for root-to-leaf similarity, we use 0.5 as the similarity threshold for deciding whether to merge nodes, the number of top tags retained in each node is $\mathtt{K} = 20$, the number of common tags used to compute tag similarity is $\mathtt{J} = 0.10\mathtt{K}$, and the number of tags used in continuity checking was $\mathtt{L} = 0.05\mathtt{K}$. The rationale for selecting the different values for α is that, the merging of roots is based more on overlap between child nodes; hence, we can rely more on the structural similarity. In the case of root-to-leaf, we instead set more weight on local similarity of common tags.

We compare SAP against the folksonomy learning method, SIG, described in (Plangprasopchok and Lerman 2009). Briefly, SIG first breaks a given sapling into (collection-set) relations. With the assumption that the nodes with the same (stemmed) name refer to the same concept, the approach employs hypothesis testing to identify the most informative relations. Informative relations are then linked into a deeper folksonomy. We used a significance test threshold of 0.01.

We quantitatively evaluate the induced folksonomies by automatically comparing them to a reference hierarchy. For the reference hierarchy, we use the hierarchy from the Open Directory Project(ODP).[1] We use methodology described in (Plangprasopchok and Lerman 2009) to automatically evaluate the quality of the learned folksonomies. Although ODP and saplings are generated from different sources, there is substantial vocabulary overlap that makes them comparable. Since the ODP hierarchy is relatively large and composed of many topics, we had to carve out the "relevant" portion for comparison. First, we specified a seed, S, which is the root of the learned folksonomy $\mathbb{F}$ and the reference hierarchy to which it is compared. Next, the folksonomy is expanded two levels along the relations in $\mathbb{F}$. The nodes in the second level are added as leaf candidates, LC. If the spanning stops after one level, we also add this node's name to LC. Given S and LC, we identify leaf candidates, LCD, that also appear in ODP, $\mathbb{D}$. All paths from S to LCD in $\mathbb{D}$ will constitute the reference hierarchy for the seed S.

S is used as seed for learning the folksonomy associated with this concept. In SIG, S and LC are both used to learn the folksonomy. The maximum depth of learned trees is limited to 4. The metrics to compare the learned folksonomies to the reference are *Lexical Recall* (Maedche and Staab 2002) and the modified *Taxonomic Overlap* defined in (Plangprasopchok and Lerman 2009), mTO. Lexical Recall measures the overlap between the learned and reference taxonomies, independent of their structure. mTO measures the quality of structural alignment of the taxonomies. Here, we report the harmonic mean, $fmTO$ instead, because of mTO's asymmetry. Since the proposed approach generates bushy folksonomies whose leaf nodes may not appear in the reference taxonomy, the mTO metric may unfairly penalize the learned folksonomy. Instead, we only consider the paths of the learned folksonomy that are comparable to the reference hierarchy. Specifically, for each leaf l in LCD, we select the path $S \rightarrow l$ in the learned folksonomy and compare it to one in the reference hierarchy. If there are many comparable paths existing in the reference, we select the one that has the highest LR to compare.

	Comparison with ODP					
	#Ovlp Leaves		fmTO		LR	
seeds	SIG	SAP	SIG	SAP	SIG	SAP
anim	68	87	0.602	0.684	0.281	0.307
bird	20	22	0.760	0.773	0.281	0.272
invertebr	3	5	0.762	0.800	0.250	0.412
vertebr	1	0	1.000	n/a	0.600	0.200
insect	5	5	0.924	0.924	0.857	0.857
plant	6	5	0.613	0.707	0.250	0.182
flora	6	18	0.483	0.483	0.130	0.352
fauna	9	10	0.463	0.396	0.113	0.111
flower	1	1	0.379	1.000	0.267	0.250
reptil	2	3	0.625	0.622	0.500	0.667
countri	2	4	0.447	0.592	0.143	0.172
africa	23	27	0.773	0.895	0.508	0.531
asia	80	88	0.734	0.783	0.396	0.453
europ	165	305	0.619	0.661	0.236	0.380
s. africa	3	3	0.431	0.600	0.444	0.444
n. america	67	95	0.545	0.594	0.165	0.201
s. america	12	14	0.706	0.846	0.415	0.400
c. america	1	3	0.631	0.725	0.417	0.500
unit kingdom	31	69	0.787	0.706	0.099	0.181
unit state	35	62	0.620	0.782	0.130	0.199
world	191	392	0.476	0.463	0.085	0.153
craft	1	1	0.603	0.603	0.056	0.050
sport	19	19	0.693	0.690	0.091	0.052
australia	12	29	0.354	0.661	0.123	0.222
canada	11	16	0.620	0.635	0.158	0.173

Table 1: This table presents empirical validation on folksonomies induced by the proposed approach, SAP, comparing to the baseline approach, SIG. The metrics are modified Taxonomic Overlap (fmTO) (averaged using Harmonic Mean), Lexical Recall (LR), where their scales are ranging from 0.0 to 1.0 (the more the better).

In Table 1, we compare the quality of the folksonomy learned for each seed by SAP, and the previous state-of-theart, SIG. Generally, SAP produced bushier trees; and recovers a larger number of concepts, relative to ODP, as indicated by the numbers of overlapping leaves (in 90% of the cases) and better LR scores (in 65.2% of the cases). SAP

[1] http://rdf.dmoz.org/, as of September 2008

produces bushier trees because individual saplings will be judged relevant using structural information, rather than frequencies of relations as in SIG. Although SIG can remove many idiosyncratic relations, it also removes many of informative ones too. SAP produces shallower trees than SIG. Nevertheless, SAP can produce trees with higher quality, relative to the ODP, as indicated by fmTO score (in 76.1% of the cases).

After closely inspecting the learned trees, we found that SAP demonstrates its advantage over the baseline in disambiguating and correctly attaching relevant saplings to appropriate induced trees. For instance, `bird` tree produced by SAP does not includes `Istanbul` or other Turkey locations, as shown in Figure 3 (a). In the `sport` tree, SAP does not include any concept about the `sky` (Note that skies and skiing share common stemmed name). In addition, there are no concepts about irrelevant events like birthdays and parades appearing in the tree, as shown in Figure 3 (b). There are some cases, e.g., `dog` and `cat`, where we could not compute the hand labeling scores because these trees often contained pet names, rather than breeds.

In all, the proposed approach has several advantages over baseline. First, it combines relevant saplings, based on contextual evidence, which can resolve ambiguity of the concept names. Second, only a seed is required to incrementally build a tree, while both seed and leaf nodes are required by the SIG method. Third, it allows similar concepts to appear multiple times within the same hierarchy. For example, SAP allows the `anim` folksonomy to have both `anim` → `pet` → `cat` and `anim` → `mammal` → `cat` paths, while only one of these paths is retained by SIG. Last, SAP can identify synonyms from structure (loops). We learned the following synonyms from Flickr data: {`anim`, `creatur`, `critter`, `all anim`, `wildlife`} and {`insect`, `bug`}.

Related work

Constructing ontological relations from text has long interested researchers, e.g., (Hearst 1992; Cimiano, Hotho, and Staab 2005; Snow, Jurafsky, and Ng 2006; Yang and Callan 2009). Many of these methods exploit linguistic patterns to infer if two keywords are related under a certain relationship. For instance, "such as" can be used to identify hyponym relations. However, these approaches are not applicable to social metadata. Such metadata is usually *ungrammatical* and much more *inconsistent* than natural language text.

Several researchers have investigated various techniques to construct conceptual hierarchies from social metadata. Most of the previous work utilizes tag statistics as evidence. Mika (Mika 2007) uses a graph-based approach to construct a network of related tags, projected from either a user-tag or object-tag association graphs; then induces broader/narrower relations using betweenness centrality and set theory. Other works apply clustering techniques to tags, and use their co-occurrence statistics to produce conceptual hierarchies (Brooks and Montanez 2006). Heymann and Garcia-Molina (Heymann and Garcia-Molina 2006) use centrality in the similarity graph of tags. The tag with the highest centrality is considered more abstract than one with a lower centrality; thus it should be merged to the hierarchy first, to guarantee that more abstract nodes are closer to the root. Schmitz (Schmitz 2006) applied a statistical subsumption model (Sanderson and Croft 1999) to induce hierarchical relations among tags. Since these works are based on tag statistics, they are likely to suffer from the "popularity vs generality" problem, where a tag may be used more frequently not because it is more general, but because it is more popular among users. Plangprasopchok and Lerman (Plangprasopchok and Lerman 2009) proposed approach that induces folksonomies from user-specified relations. Specifically, it filters out conflicting and noisy relations based on parent and child nodes' co-occurrence statistics; then, combines these relations into a larger folksonomy. Although this approach can bypass the "popularity vs generality" problem, like all prior approaches to learning folksonomies from social metadata, it does not address the ambiguity problem.

The sapling merging approach described in this paper is an extension of collective relational clustering approach used for entity resolution (Bhattacharya and Getoor 2007). That work proposed a method to identify and disambiguate entities, such as authors, that utilizes two types of evidences: intrisic and extrinsic features. Intrinsic features are associated with specific instances, such as author names, while extrinsic features derived from structural evidence, e.g., coauthors in a citations database. Intuitively, two names refer to the same author if they are similar and their co-author names refer to the same set of authors. Analogously, we identify and disambiguate concept names from names and tags (intrinsic) and neighboring nodes' features (extrinsic). However, for efficiency reason, we use the naive version of the relational clustering, where we simply directly use the features from neighbors as the extrinsic features, rather than cluster labels.

Handling mutual shortcuts by keeping the sapling which is more similar to the ancestor is similar in spirit to the minimum evolution assumption in (Yang and Callan 2009). Specifically, a certain hierarchy should not have any sudden changes from a parent to its child concepts. Our approach is also similar to several works on ontology alignment (e.g. (Udrea, Getoor, and Miller 2007)). However, unlike those works, which merge a small number of deep, detailed and consistent concepts, we merge large number of noisy and shallow concepts, which are specified by different users.

Conclusion

This paper describes an approach which incrementally combines a large number of shallow hierarchies specified by different users into common, denser and deeper "folksonomies." The approach addresses the challenges of learning folksonomies from social metadata and demonstrates several advantages over the previous work: disambiguating concepts and allowing similar concepts to appear at multiple places within the same folksonomy. Empirical results demonstrate that our approach can induce quite detailed folksonomies, which are also more consistent with taxonomies of the Open Directory Project. Additionally, it is general enough for other domains, such as tags/bundles in *Delicious* and files/folders in personal workspaces.

For the future work, in addition to automatically separating broader/narrower from related-to relations, we would like to develop a systematic way to handle individual saplings whose child nodes are from different facets. This will improve the quality of the learned folksonomies by not mixing concepts from different facets. We are also working on combining more sources of evidence such as geographical information for learning accurate folksonomies. Lastly, we would like to frame the approach in a fully probabilistic way (Plangprasopchok, Lerman, and Getoor 2010), which provides a systematic way to combine heterogeneous evidences, and takes into account uncertainties on similarities between concepts and relations.

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant No.IIS-0812677.

References

Bhattacharya, I., and Getoor, L. 2007. Collective entity resolution in relational data. *ACM Trans. Knowl. Discov. Data* 1(1):5.

Brooks, C. H., and Montanez, N. 2006. Improved annotation of the blogosphere via autotagging and hierarchical clustering. In *Proceedings of the World Wide Web conference*.

Cimiano, P.; Hotho, A.; and Staab, S. 2005. Learning concept hierarchies from text corpora using formal concept analysis. *J. Artif. Intell. Res. (JAIR)* 24:305–339.

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

Hearst, M. A. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Heymann, P., and Garcia-Molina, H. 2006. Collaborative creation of communal hierarchical taxonomies in social tagging systems. Technical Report 2006-10, Stanford University, Stanford, CA, USA.

Maedche, A., and Staab, S. 2002. Measuring similarity between ontologies. In *Proceedings of the Knowledge Engineering and Knowledge Management*.

Mathes, A. 2004. Folksonomies: cooperative classification and communication through shared metadata.

Mika, P. 2007. Ontologies are us: A unified model of social networks and semantics. *J. Web Sem.* 5(1):5–15.

Plangprasopchok, A., and Lerman, K. 2009. Constructing folksonomies from user-specified relations on flickr. In *Proceedings of the World Wide Web conference*.

Plangprasopchok, A., and Lerman, K. 2010. Modeling social annotation: a bayesian approach. *Transactions on Knowledge Discovery from Data (to appear)*.

Plangprasopchok, A.; Lerman, K.; and Getoor, L. 2010. Integrating structured metadata with relational affinity propagation. In *In proceedings of AAAI Workshop on Statistical Relational AI (to appear)*.

Rattenbury, T.; Good, N.; and Naaman, M. 2007. Towards automatic extraction of event and place semantics from flickr tags. In *Proceedings of the conference on Research and development in information retrieval*.

Sanderson, M., and Croft, W. B. 1999. Deriving concept hierarchies from text. In *Proceedings of the conference on Research and development in information retrieval*.

Schmitz, P. 2006. Inducing ontology from flickr tags. In *Proceedings of the WWW workshop on Collaborative Web Tagging Workshop*.

Snow, R.; Jurafsky, D.; and Ng, A. Y. 2006. Semantic taxonomy induction from heterogenous evidence. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Udrea, O.; Getoor, L.; and Miller, R. J. 2007. Leveraging data and structure in ontology integration. In *SIGMOD Conference*.

Yang, H., and Callan, J. 2009. A metric-based framework for automatic taxonomy induction. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.