Growing a Tree in the Forest: Constructing Folksonomies by Integrating Structured Metadata

Anon Plangprasopchok
USC Information Sciences Institute
Marina del Rey, CA 90292, USA
plangpra@isi.edu

Kristina Lerman
USC Information Sciences Institute
Marina del Rey, CA 90292, USA
lerman@isi.edu

Lise Getoor
Department of Computer Science
University of Maryland, College Park
getoor@cs.umd.edu

ABSTRACT

Many social Web sites allow users to annotate the content with descriptive metadata, such as tags, and more recently 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, since it is sparse, shallow, ambiguous, noisy, and inconsistent. We describe an approach to folksonomy learning based on relational clustering, which exploits 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, and demonstrate that the proposed approach addresses the challenges. Moreover, comparing to previous work, the approach produces larger, more accurate folksonomies, and in addition, scales better.

Categories and Subject Descriptors

H.2.8 [**DATABASE MANAGEMENT**]: Database Applications—*Data mining*; I.2.6 [**ARTIFICIAL INTELLIGENCE**]: Learning—*Knowledge Acquisition*

General Terms

Algorithms, Experimentation, Human Factors, Measurement

Keywords

Folksonomies, Taxonomies, Collective Knowledge, Social Information Processing, Data Mining, Social Metadata, Relational Clustering

1. 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 mined 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 such as personalization, data mining, 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 hierarchies, such as Linnaean classification system or WordNet, automatically learned folksonomies are attractive because they (1) represent collective agreement of many individuals; (2) are relatively inexpensive to obtain; (3) can adapt to evolving vocabularies and community's information needs; and (4) they are directly tied to the annotated content. A folksonomy can facilitate browsing of user-generated content, and help users visualize how their own content fits within the community's or aid them in organizing it.

Learning a folksonomy by integrating structured metadata created by many users presents a number of challenges. Since users are free to annotate data according to their own preferences, social metadata is *noisy*, *shallow*, *sparse*, *ambiguous*, *conflicting*, *multi-faceted*, and expressed at *inconsistent granularity levels* across many users. Several recent works have addressed some of the above challenges. For instance, [7, 15] proposed inducing folksonomies from tags by utilizing tag statistics. The basic motivation behind these approaches is that more frequent tags describe more general concepts. However, frequency-based methods cannot distinguish between more general and more popular concepts. In our previous work, SIG [12], we overcame this problem by using user-specified relations, extracted from personal hierarchies. Nevertheless, it ignored other evidence, e.g., structure of hierarchies and tags, which potentially address the challenges listed above.

We propose a novel approach to learn folksonomies from social metadata in the form of tags and user-specified shallow hierarchies. Our approach is driven by a similarity mea-

sure that utilizes statistics of both kinds of metadata to incrementally weave individual hierarchies into a deeper, more complete folksonomy. The approach has several advantages over previous work. Specifically, it: (1) better addresses the challenges of sparse, shallow, ambiguous, noisy and inconsistent data; (2) the approach is more scalable, especially when the learned folksonomies are deep; (3) it produces more consistent and richer folksonomies. We demonstrate the utility of our present approach on real-world data from Flickr, and introduce a simple metric, which evaluates the quality of the learned folksonomies in terms of depth and bushiness.

2. STRUCTURED SOCIAL METADATA

In addition to tagging content, some social Web sites also allow users to organize it hierarchically. *Delicious* users can group related tags into bundles, and *Flickr* users can group related photos into *sets* and then group related sets in *collections*. While the sites themselves do not impose any constraints on the vocabulary or semantics of the hierarchies, in practice users employ them to represent both subclass relationships ('dog' is a kind of 'mammal') and part-of relationship ('my kids' is a part of 'family'). Users appear to express both types of relations (and possibly others) through personal hierarchies, in effect using the hierarchies to specify broader/narrower relations. Even without strict semantics being attached to these relations, we believe that personal hierarchies represent a novel, rich source of evidence for learning folksonomies.

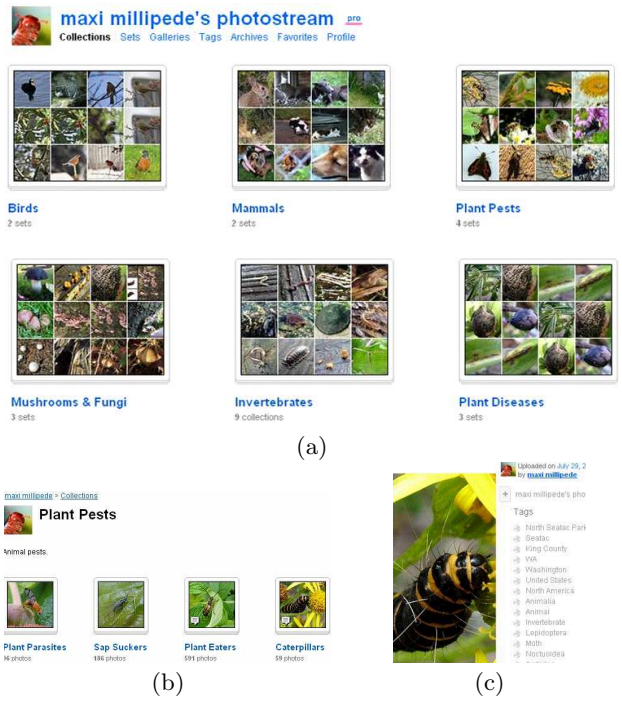

Figure 1: Personal hierarchies specified by a Flickr user. (a) Some of the collections created by the user and (b) sets associated with the Plant Pests collection, and (c) tags associated with an image in the Caterpillars set.

We briefly describe how this feature is implemented on the social photo-sharing site, *Flickr* (`http://www.flickr.com`). Flickr allows users to group their photos in album-like folders, called *sets*. Users can also group sets into "super" albums, called *collections*.[1] Both sets and collections are named by the owner of the image. A photo can be part of multiple sets.

While Flickr does not enforce any specific rules about how to organize photos or how to name them, most users group "similar" or "related" photos into the same set and related sets into the same collection. Some users create multi-level hierarchies containing collections of collections, etc., but the vast majority of users create shallow hierarchies, consisting of collections and their constituent sets. Figure 1(a) shows some of the collections created by an avid naturalist on Flickr. These collections reflect the subjects she likes to photograph: Birds, Mammals, Plants, Mushrooms & Fungi, Plant Pests, Plant Diseases, etc. Figure 1(b) shows sets of the Plant Pests collection: Plant Parasites, Sap Suckers, Plant Eaters, and Caterpillars. Each set contains one or more photos, which are tagged by the user. For example, a photograph in the set Caterpillars (Figure 1(c)), is annotated with multiple tags describing it:(`Animal`, `Lepidoptera`, `Moth`, `larva`, `Caterpillar`), its color (`Black and orange`), condition (`on Senecio`, `eating`), and location (`North Seatac Park`, `King County`, `WA`, `North America`).

3. CHALLENGES IN LEARNING FROM STRUCTURED METADATA

Learning folksonomies from social metadata, specifically, from structured metadata, presents a number of challenges:

3.1 Sparseness

Social metadata is usually very sparse. Users provide 4 –7 tags per bookmark on *Delicious* in our data set and 3.74 tags per photo on *Flickr* [13]. 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. Moreover, among these shallow hierarchies, few users organize content the same way. 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 rich and complete folksonomy, we have to aggregate social metadata from many different users.

3.2 Noisy vocabulary

Vocabulary noise has several sources. One common source is variations and errors in spelling. Noise also arises from users' idiosyncratic naming conventions. While such names as `not sure`, `pleaseaddthistothethemecomppoll`, `mykid` may be meaningful to image owner and her narrow interest group, they are relatively meaningless to other users.

3.3 Ambiguity

An individual tag is often ambiguous [9, 5]. For example, `jaguar` can be used to refer to a mammal or a luxury car. Similarly, terms that are used to name collections and

[1]The collection feature is limited to paid "pro" users. Pro users can also create unlimited number of photo sets, while free membership limits a user to three sets.

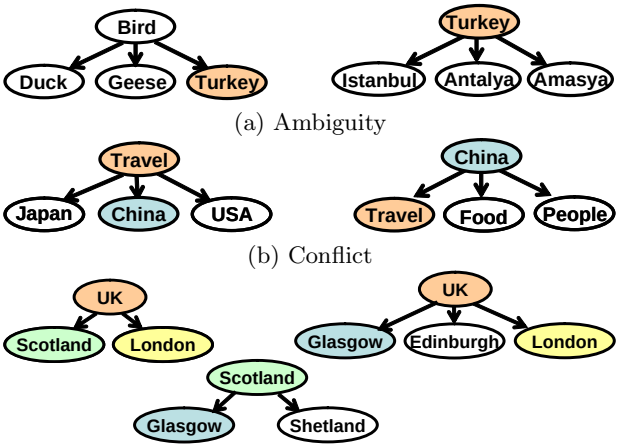

Figure 2: Schematic diagrams of personal hierarchies created by Flickr users. (a) Ambiguity: the same term may have different meaning ("turkey" can refer to a bird or a country). (b) Conflict: users' different organization schemes can be incompatible (`china` is a parent of `travel` in one hierarchy, 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.

sets can refer to different concepts. Consider the hierarchy in Figure 2 (a), where `turkey` collection 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. Structural and contextual information may help disambiguate metadata.

3.4 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 2 (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 `bug` is a "pest," a term broader than `insect`, while to others it is a subclass of `insect`. As a result, some users may express `bug` $\rightarrow$ `insect`, while the others express an inverse relation. Another source of noise is variation in degree of expertise on a topic. Many users assemble images of spiders in a set called spiders and assign it to an insect collection, while others correctly assign spiders to arachnid.

3.5 Varying granularity level

Differences in users' level of expertise and expressiveness may also lead to relatively imprecise metadata. Experts may use specific breed names to tag dog photos, while non-experts will simply use the tag `dog` to annotate them[5]. 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 2 (c). Combining data from these users potentially generates multiple paths from one concept to another.

4. LEARNING FOLKSONOMIES FROM STRUCTURED METADATA

We propose a simple, yet effective approach to combine many personal hierarchies into a global folksonomy that takes above challenges into account. We define a personal hierarchy as a shallow tree, a *sapling*, composed of a root node r^i and its children, or leaf nodes $\langle l_1^i, ..l_j^i \rangle$. 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. Only a small number of users define multi-level hierarchies; for these, we decompose them and represent them as collections of saplings. At the top level, we have a root node, which corresponds to the top-level collection, and its leaf nodes corresponding to the root's sets or collections. We then construct saplings that correspond to the leaf nodes, which are collections, and so on. We assume that hierarchical relations between a root and its children, $r^i \rightarrow l_j^i$, specify broader-narrower relations. Hence, the sapling in Figure 1 (b) is `Plant Pests` $\rightarrow$ {`Plant Parasites`, `Sap Suckers`, `Plant Eaters`, `Caterpillars` }.

In addition to hierarchical structure, each sapling carries information derived from tags. On Flickr, users attach tags only to photos; therefore, the tag statistics of a sapling's leaf (set) are aggregated from that set's constituent photos. Tag statistics are then propagated from the leaves to the parent node. In our example, Plant Parasites aggregates tag statistics from all photos in this set, and its parent Plant Pests contains tag statistics accumulated from all photos in Plant Parasites and its siblings. 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 $\tau_{l_j^i}$s.

Given a collection of saplings, specified by many different users, our goal is to aggregate them into a common, denser and deeper tree. Before describing our approach, we first briefly describe data preprocessing steps that address the sparseness and noise challenges listed above.

4.1 Data Preprocessing

We extract terms representing concepts from collection and set names. We found that users often combine two or more concepts within a single name, e.g., "Dragonflies/Damselflies", "Mushrooms & Fungi", "Moth at Night." Terms can be joined by bridging words that include prepositions "at", "of", "in," and conjunctions "and" and "or," or special characters, such as '&', '<', '>', ':', '/'. We start by tokenizing collection and set names on these words and characters. We do not tokenize on white spaces to avoid breaking up terms like "South Africa." We remove terms composed only of non-alphanumeric characters and frequently-used uninformative words, *e.g.*, "me" and "myself." We then normalize all terms by lowercasing them.

After tokenization, a set or collection name may be split

into multiple terms, which we expand into leaves. Suppose a user created a collection `animal` containing a set `cats and dogs`. After tokenization we get the sapling `animal` → {`cats`, `dogs`}. However, if the root node is determined to have a composite name, we ignore the entire sapling because we do not know which parent concepts correspond to which child concepts.

4.2 Relational Clustering of Structured Metadata

In order to learn a folksonomy, 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 sapling's 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[1] and its basic element is the similarity measure between a pair of nodes.

We define a similarity measure which combines heterogeneous evidence available in the structured social metadata, and is a combination of *local similarity* and *structural similarity*. The local similarity between 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 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:

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

where $0 \leq \alpha \leq 1$ is a weight for adjusting contributions from $localSim(,)$ and $structSim(,)$. We judge whether two nodes are similar if the similarity is greater than the threshold, τ.

4.2.1 Local Similarity

The local similarity of nodes a and b is composed of (1) name similarity and (2) tag distribution similarity. Name similarity can be any string similarity metric, which returns a value ranging from 0 to 1. Tag similarity, $tagSim(,)$, can be any function for measuring the similarity of distributions. Because of the sparseness of the data, and to make the computation fast, we use a simple function which counts the number of common tags, n, in the top `K` tags of a and b; it returns 1 if this number is equal or greater than J, else it returns $\frac{n}{J}$. Local similary is a weighted combination of name and tag similarities:

$$\begin{aligned} localSim(a,b) &= \beta \times nameSim(a,b) \\ &+ (1-\beta) \times tagSim(a,b)). \end{aligned} \quad (2)$$

Tag similarity helps address the *ambiguity* challenge described in Section 3. For example, the top tags of the node `turkey` that refers to a bird include "bird", "beak", "feed", while the top tags of `turkey` that refers to the country include different terms about places within the country.

4.2.2 Structural Similarity

Structural similarity between two nodes depends on position of nodes within their saplings. We define two versions: $structSimRR(,)$ which computes 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 have a similar name and some of their leaf nodes also have similar names. In this case, there is no need to compute $tagSim(,)$ of these leaf nodes. We define the normalized common leaves factor, $\mathbb{CL}$, as $\frac{1}{Z}\sum_{i,j}\delta(name(l_i^A), name(l_j^B))$, where $\delta(.,.)$ returns 1 if the both arguments are exactly the same; otherwise, it returns 0; $name(l_i^A)$ is a function that returns the name of a leaf node l_i^A of sapling A. Z is a normalizing constant, which is described in greater detail later. Structural similarity between two root nodes is then defined as follows:

$$\begin{aligned} structSimRR(r^A, r^B) = \mathbb{CL} &+ (1-\mathbb{CL}) \\ &\times tagSim(\grave{\mathbb{L}}_{tag}^A, \grave{\mathbb{L}}_{tag}^B), \end{aligned} \quad (3)$$

where $\grave{\mathbb{L}}_{tag}^A$ is an aggregation of tag distributions of all l_i^A, at which $name(l_i^A) \neq name(l_j^B)$ for any leaf node l_j^B of the sapling B. From Eq. 3, we compute similarity based on: (1) how many of their children have common name (they match); (2) the tag distribution similarity of those that do not have the same name. The second term is an optimistic estimate that child nodes of the two saplings refer to the same concept while having different names.

The normalization coefficient $Z = min(|l^X|, |l^Y|)$, where $|l^X|$ is a number of child nodes of X. 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 ($Z = union(l^X, l^Y)$), as defined in Jaccard similarity, results in overly penalizing small saplings. $min(,)$, 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 roots r^A and r^B are similar, we merge saplings A and B with the $mergeByRoot(A, B)$ operation. This operation creates a new sapling, M, which combines structures and tag statistics of A and B. In particular, 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 a union of l^A and l^B. If there are leaves from A and B that share a 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 mentioned in Section 3.

Root-to-Leaf similarity. Merging the root node of one sampling with the leaf node of another sapling extends 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 2 (c). Suppose that we have already merged `uk` saplings. Now, there are two saplings `uk` → {`scotland`, `glasgow`, `edinburgh`, `london`} and `scotland` → {`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.

4.3 SAP: Growing a Tree by Merging Saplings

We describe SAP algorithm, which uses 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 2(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,

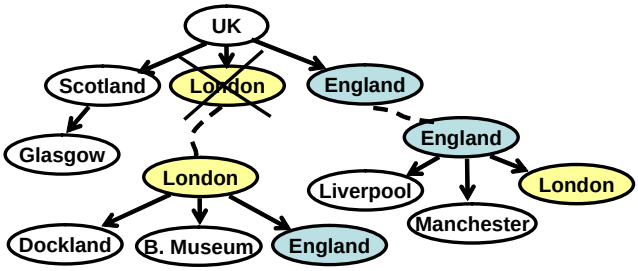

Figure 3: Appearance of mutual shortcuts between London and England when merging London and England saplings. To resolve them, we compare the similarity between UK-London and UK-England sapling pairs. Since England sapling is closer to UK than London sapling, we simply attach England sapling to the tree; while ignoring London leaf under UK.

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` → {`london`, `england`, `scotland`} as the current learned tree, and are about to attach `london` → {`british museum`, `dockland`, `england`} 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` → {`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 3.

Handling Loops. Attaching a sapling to a leaf of the learned tree may 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 the `england` sapling in Figure 3 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 $name(l_j^F) = name(l_j^A)$; 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 $\frac{1}{Z}\sum_{i,j}\delta(name(l_i^X), name(l_j^Y))$.

Mitigating Noisy Vocabularies. As mentioned in Section 3, 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.

Managing Complexity. Computing the similarity measure for all pairs of saplings in the corpus is impractical, even considering local or structural similarity only. We address this scalability issue in two ways. First, we only compare sapling nodes if they share the same (stemmed) name. This reduces the total number of pairs which need to be compared and eliminates the need to compute $nameSim(,)$ in Eq. 2. Second, we apply the blocking approach [11] for efficiently computing similarity and merging sapling roots. The basic idea behind this approach is to first use a cheap similarity measure to "roughly" group similar items. We can then thoroughly compute item similarities and merge them within each "roughly similar" group by using the more computationally expensive similarity measure. We assume that items judged to be dissimilar by the cheap measure will also be dissimilar when evaluated by the more expensive measure. Since the approach applies the expensive measure to a much smaller set of items, it reduces the time complexity of the clustering method.

In our case, we compute an inexpensive similarity measure based on the most frequent tags. Specifically, we map the top tags to some integer code, which can be cheaply sorted by any database. Subsequently, we use the database to sort saplings by their codes, moving roughly similar saplings to neighboring rows. The process begins by scanning sorted saplings in the database table on a sapling by sapling basis. If the presently scanned sapling has not been merged with some other sapling, we add this sapling to the top of the queue. If the present sapling does belong to some merged sapling, we check if this sapling is also similar to some other merged saplings in the queue. We use Eq. 1, Eq. 2 and Eq. 3 to evaluate their similarity. If they are similar enough, we will merge them together into a new merged sapling; then add it to the top of the queue. The scanning is performed repeatedly until the number of merged saplings no longer changes.

4.4 Complexity Analysis

Here we sketch the computational complexity of SAP. Basically, SAP can be decomposed into 2 different parts: (1) root-to-root merging, which expands folksonomies' width; (2) leaf-to-root merging, which extends folksonomies' depth. These two parts are loosely dependent, i.e., one can cluster all saplings into different senses; then "vertically" merge the root of one sapling sense with a leaf of the other. Since we use blocking and only cluster saplings with the same stemmed names, the computational complexity depends on (1) the number of unique stemmed names in the data set; (2) the average number of saplings that share a name. Let N and M be the number of nodes and the number of unique stemmed names in the data set respectively. Hence, for each stem, there are $\frac{N}{M}$ nodes to be compared on average. We use database to first roughly sort saplings, which generally requires $O(\frac{N}{M}log(\frac{N}{M}))$. After saplings are sorted, they are scanned and merged. This is repeatedly, say in i iterations, until the number of clusters no longer changes, which requires $O(i \times \frac{N}{M})$. In all, the complexity of the first part is $O(Nlog(\frac{N}{M}) + iN)$. Empirically, the number of clusters converges in 2-3 iterations on average.

Let b and d be the branching factor and the depth of the tree we want to produce. In addition, suppose that there are s sapling senses for each stemmed name on average. Since we have to traverse each inner node of the tree to attach relevant sapling senses, and for each of these nodes we need to compare the similarity to all sapling senses with similar root names, this requires $O(s \times b^d)$.

Our earlier work, SIG [12], which is described in more detail in Section 5, only considered the best path from a root to a given leaf of the tree, and required enumerating all possible paths between them. In the best case, when there are no shortcuts or loops in the data set, the number of paths from the root to all leaves of a given tree is equal to the number of the leaves, and that only requires $O(b^d + b^{d-1})$ to check whether each edge should be included. In the worst case, when shortcuts appear to all node pairs, we would need $O(\binom{d+1}{2} \times b^d)$ to check all possible edges. Moreover, we also need to enumerate all possible paths for the root to all leaves of the tree, which requires $O(1 + \sum_{e=1:d-1}\binom{d-1}{e})$ per root-to-leaf pair. Hence, we expect our approach to scale better than the previous one as the depth of the output tree increases and when there are many shortcuts.

5. EMPIRICAL VALIDATION

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 [12]. 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. For reasons described in Section 4.1, 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 has a number of parameters as shown in Table 1. In our experiment, we ignored the parameter β since only sapling nodes with the same name are needed to be compared as described the previous section. To explore the range of these

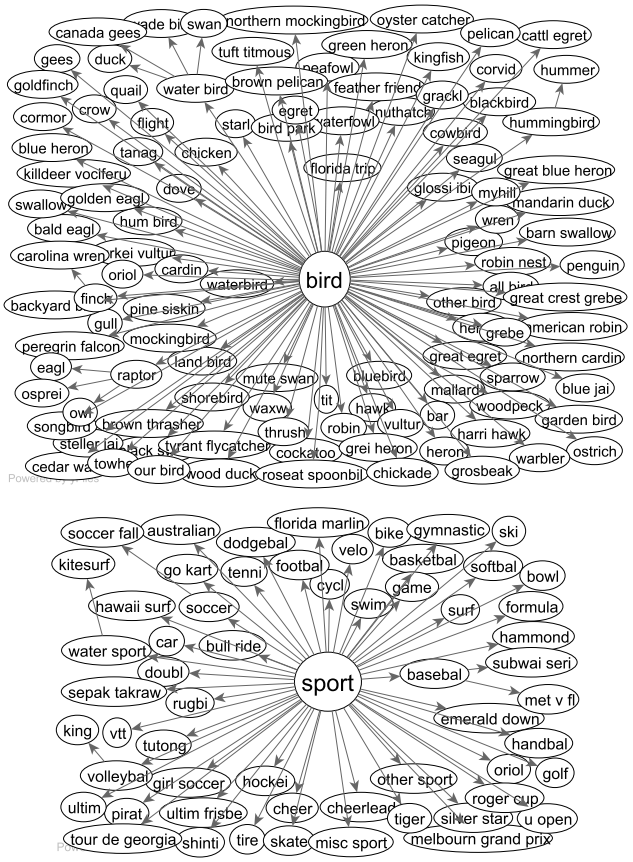

Figure 4: Folksonomies learned for `bird` and `sport`

parameters, we set up a small experiment by first selecting 5 different seed terms[2]; then running the approach with different values. Optimal parameter values would enable the approach to reasonably combine/separate saplings with similar/different senses. We manually inspected the induced folksonomies to check how the saplings were merged/separated.

The parameter `K` allows the approach to consider only top frequency tags, which tend to be more stable and less noisy [5]. Nevertheless, the top tags will not contain enough information if the number is set too low, e.g., $K = 10$. At the fixed values of the common tag threshold, `J` $= 4$, and the structural-local weight combination, $\alpha_{RR} = 0.1$ (in this case, we simply evaluated on merging root-root nodes; hence there is no need for α_{LR}), we found that the approach performs reasonably well when the value of `K` is around 30–60, while the performance starts to degrade for `K` > 60. Smaller values of `J` leads to a weak tag similarity measure, which, in turn, mistakenly causes the approach to merge saplings with different senses. Large `J` will be relatively stringent, and as a result, saplings of the same sense will not be merged. We found that, at `K` $= 40$, the value of `J` between 4 to 6 allows reasonable results.

For α_{RR} and α_{LR}, the weight combination between local and structural similarity for root-root and leaf-root nodes in Eq. 1, the larger the values the more the similarity measure emphasizes on the structural similarity. From our experiments, we found that the structure information is very informative. When α_{RR} is set to a very large value or the maximum, 1.0, the approach clusters "structure-rich" saplings, i.e., saplings containing many children, reasonably well. For leaf-to-root merging or in situations where structural information is uncommon, local similarity becomes more important. We discovered that at $\alpha_{RR} = 0.1$ and $\alpha_{LR} = 0.8$, the approach produces reasonable folksonomies. Due to space limitations, we do not include the complete set of results. Here, we report the parameter values that resulted in good performance: we set `K` $= 40$; `J` $= 4$. In addition, since all similarity measures are normalized to range within 0.0 and 1.0, we set $\tau = 0.5$.

We compare SAP against the folksonomy learning method, SIG, described in [12]. Briefly, SIG first breaks a given sapling into (collection-set) individual parent-child 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 informative relations, i.e., checking if the relation is not generated at random. Informative relations are then linked into a deeper folksonomy. We used a significance test threshold of 0.01.

[2] `ski`, `bird`, `victoria`, `africa` and `insect`

Parameters	Description
`K`	The number of top frequent tags
`J`	The number of common tags for tag similarity
α_{RR}	The weight combination of local and structural similarity for computing root-to-root similarity
α_{LR}	The weight combination of local and structural similarity for computing leaf-to-root similarity
β	The weight combination of name and tag similarity (not required in our experiment)
τ	The similarity threshold

Table 1: Parameters of the folksonomy learning approach.

5.1 Methodology

We quantitatively evaluate the induced folksonomies by (1) automatically comparing them to a reference hierarchy; (2) structural evaluation; (3) manual evaluation.

Evaluation against the reference hierarchy: We use the reference hierarchy from the Open Directory Project (ODP).[3] We selected ODP because, in contrast to WordNet, ODP is generated, reviewed and revised by many registered users. These users seem to use more colloquial terms than appear in WordNet. In addition, like Flickr users, they specify less formal relations, mainly broader/narrower relations. WordNet, on the other hand, specifies a number of formal relations among concepts, including hypernymy and meronymy.

We use methodology described in [12] 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

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

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 $\mathbb{D}$ constitute the reference hierarchy for the seed S.

Next, 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* [8] and the modified *Taxonomic Overlap* defined in [12], 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.

Structural evaluation: Ideally, we prefer an approach that generates bushier and deeper trees. The scope of concepts in such trees are broadly enumerated (tree width); while, each concept is subcategorized in enough detail (tree depth). Although one can use an average depth of a tree and branching factor, it is difficult to justify which trees are better overall since these metrics are independent. A very bushy tree may have only 1 level depth; meanwhile, a very deep tree may have a chain-like structure. In this work, we define a simple, yet intuitive measure, Area Under Tree (AUT), which takes *both* tree bushiness and depth into account. To calculate AUT for a certain tree, we compute the distribution of the number of nodes in each level and then compute the area under the distribution. Intuitively, trees that keep branching out at each level will have larger AUT than those that are short and thin. Suppose that we have a tree with one node at the root, three nodes at 1^{st} level and four at 2^{nd}. With the scale of tree depth set to 1.0, AUT of this tree would be $0.5 \times (1+3) + 0.5 \times (3+4) = 5.5$ (a sum of trapezoids).

Manual evaluation: We use 3 human subjects to evaluate the portions of induced folksonomies which were not comparable to ODP hierarchy. We randomly selected 10% of the paths (all of them if there are fewer than 10 paths in the learned folksonomy) that are not in the reference hierarchy and asked three judges to evaluate them. If a portion of the path is incorrect, either because an incorrect concept appears or the ordering of concepts is wrong, the judges were asked to mark it incorrect, otherwise it is correct. They can also mark the path "unsure" if there is not enough evidence for a decision. A path's label is based on the majority decision. If there is no agreement, or the path is marked uncertain by all judges, we exclude it.

5.2 Results

In Table 2, we compare the quality of the folksonomy learned for each seed by SAP, and the earlier work, SIG. SAP generally 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 76% of the cases). Moreover, SAP can produce trees with higher quality, relative to the ODP, as indicated by $fmTO$ score (in 68% of the cases). From the structural evaluation, SAP produced bushier trees as indicated by AUT in 87% of the cases. In addition, the average depth (not shown in the Table) from roots to all leaves of the trees over all cases generated by SAP is deeper than SIG (2.68 vs. 2.37).

Approach	Incorrect Path
SAP	anim/**other anim**/mara
SAP	world/landscap/**architectur**/scarborough
SAP	world/**scotland**/through viewfind
SAP	europ/**franc**/flight to
SIG	anim/pet/**chester**/chester zoo
SIG	bird/**turkei**/antalya
SIG	bird/**turkei**/ephesu
SIG	fauna/**underwat**/destin
SIG	south africa/**safari**/isla paulino
SIG	south africa/**safari**/la flore
SIG	sport/**golf**/adamst
SIG	sport/**ski**/cloud/other/new year
SIG	world/canada/**victoria**/melbourn

Table 3: The table lists all incorrect paths caused by possibly ambiguous nodes, which are in bold.

Although the manual evaluation suggests that both approaches can induce about the same quality on the paths that are uncomparable to ODP, after closely inspecting the learned trees, we found that SAP demonstrates its advantage over SIG 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 4. In the `sport` tree, SAP does not include any concept about the `sky` (Note that skies and skiing share common name). In addition, there are no concepts about irrelevant events like birthdays and parades appearing in the tree. 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.

We further considered how many of the incorrect paths are caused by node ambiguity. To do so, we first identified ambiguous terms, and checked to see how many of the incorrect paths contain these terms. Although it is not obvious how to automatically identify ambiguous terms, we use the following heuristic to determine the possible ambiguities: for a given leaf of the induced tree, if many different merged senses exist (i.e., > 10), then we consider the leaf ambiguous. During the tree induction process, we keep track of these nodes and the root. Subsequently, we use the ambiguous terms and their root names to check the accuracy of paths in the hand labeled data containing them. As presented in Table 3, there is about a half reduction in error for ambiguous paths using SAP. This supports our claim about superiority of SAP on node disambiguation.

In all, the proposed approach, SAP, has several advantages over the baseline, SIG. First, it exploits both structure information and tag statistics to combine relevant saplings, which can produce more comprehensive folksonomies as well as resolve ambiguity of the concept names. Second, it al-

	Whole folksonomies				Comparison with ODP								Manual	
	#leaves		AUT		#ovlp lvs		fmTO		LR		AUT		Acc (10%)	
seeds	sig	sap	sig	sap	sig	sap	sig	sap	sig	sap	sig	sap	sig	sap
anim	268	**583**	694.0	**1076.0**	68	**92**	0.602	**0.659**	0.281	**0.360**	160.0	**189.5**	**0.89**	0.74
bird	73	**103**	84.5	**113.5**	20	**22**	**0.760**	0.755	0.281	**0.315**	21.5	**28.5**	0.60	**1.00**
invertebr	11	**15**	15.5	**19.5**	**3**	1	0.762	**1.000**	**0.250**	0.125	**4.5**	1.5	1.00	1.00
vertebr	80	**114**	162.5	**236.5**	**1**	0	1.000	n/a	**0.600**	0.200	2.5	n/a	1.00	1.00
insect	29	**44**	35.5	**61.5**	5	5	0.924	0.924	0.857	0.857	6.5	6.5	1.00	1.00
fish	**7**	6	**7.5**	6.5	0	0	n/a	n/a	0.016	0.016	n/a	n/a	1.00	1.00
plant	110	**194**	265.5	**426.0**	6	**7**	0.613	**0.735**	0.250	**0.273**	**13.0**	11.5	0.67	**1.00**
flora	64	**403**	173.0	**1048.5**	6	**18**	**0.483**	0.481	0.130	**0.407**	16.0	**84.0**	1.00	1.00
fauna	141	**609**	420.0	**1146.0**	9	**31**	0.463	**0.490**	0.113	**0.212**	27.0	**71.5**	**0.91**	0.85
flower	112	**169**	210.5	**226.5**	1	1	0.379	**1.000**	**0.267**	0.250	**3.5**	1.5	1.00	n/a
reptil	3	**4**	4.5	4.5	2	**3**	**0.625**	0.622	0.500	**0.667**	2.5	**3.5**	n/a	n/a
amphibian	1	1	1.5	1.5	1	1	1.000	1.000	1.000	1.000	1.5	1.5	n/a	n/a
build	7	**23**	11.5	**37.5**	0	0	n/a	n/a	1.000	1.000	n/a	n/a	1.00	1.00
urban	6	**80**	15.0	**145.5**	0	0	n/a	n/a	0.071	0.071	n/a	n/a	1.00	1.00
countri	378	**1605**	798.5	**4504.0**	2	**4**	0.447	**0.665**	0.143	**0.214**	8.0	**8.5**	1.00	1.00
africa	53	**71**	90.5	**119.5**	23	**27**	0.773	**0.895**	0.508	**0.547**	37.5	**40.5**	1.00	1.00
asia	187	**284**	389.0	**631.5**	80	**85**	0.734	**0.788**	0.396	**0.484**	165.5	**168.5**	1.00	1.00
europ	379	**1073**	916.0	**2706.5**	165	**301**	0.619	**0.670**	0.236	**0.418**	369.0	**874.5**	**1.00**	0.94
south africa	12	**17**	15.5	**18.5**	3	3	0.431	**0.600**	0.444	0.444	3.5	3.5	0.78	**1.00**
north america	166	**731**	435.0	**2203.5**	67	**118**	0.545	**0.576**	0.165	**0.319**	170.5	**361.5**	**1.00**	0.92
south america	32	**50**	54.5	**101.5**	12	**15**	0.706	**0.832**	0.415	**0.463**	20.5	**28.5**	1.00	1.00
central america	**27**	8	**53.5**	12.5	1	**2**	0.631	**0.754**	0.417	**0.500**	2.5	**4.5**	1.00	1.00
unit kingdom	106	**267**	274.5	**658.5**	31	**82**	**0.787**	0.724	0.099	**0.127**	71.5	**179.5**	1.00	1.00
unit state	102	**375**	217.0	**936.5**	35	55	0.620	**0.749**	0.130	**0.256**	74.5	**122.0**	1.00	1.00
world	545	**3177**	1437.0	**9235.0**	191	**475**	**0.476**	0.461	0.085	**0.215**	490.0	**1676.5**	**0.97**	0.96
citi	123	**448**	234.0	**927.5**	0	0	n/a	n/a	**0.111**	0.100	n/a	2.5	1.00	1.00
craft	**5**	1	**10.5**	1.5	1	0	0.603	n/a	**0.056**	0.050	2.5	n/a	1.00	n/a
dog	15	**26**	17.5	**28.5**	0	**1**	n/a	1.000	0.045	**0.080**	n/a	1.5	n/a	n/a
cat	11	**39**	13.5	**41.5**	0	0	n/a	n/a	0.100	0.100	n/a	n/a	n/a	n/a
sport	**207**	74	**407.0**	86.5	19	**27**	**0.693**	0.647	**0.091**	0.084	30.0	**31.5**	0.28	**1.00**
australia	47	**83**	71.0	**147.5**	12	**27**	0.354	**0.665**	0.123	**0.216**	14.5	**36.5**	0.67	**1.00**
canada	55	**763**	128.0	**2502.0**	11	**27**	**0.620**	0.587	0.158	**0.241**	21.5	**75.5**	1.00	1.00

Table 2: This table presents empirical validation on folksonomies induced by the proposed approach, sap, comparing to the baseline approach, sig. The first column group presents properties of the whole induced trees: the number of leaves and Area Under Tree(AUT). The second column group reports the quality of induced trees, relatively to the ODP hierarchy. The metrics in this group 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), as AUT is computed from portions of the trees, which are comparable to ODP. "#ovlp lvs" stands for a number of overlap leaves (to ODP). The last column group reports performance on manually labeled portions of the trees, which do not occur in ODP. In some cases, "n/a" exists since we cannot compute its corresponding value.

lows 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`}.

6. RELATED WORK

Constructing ontological relations from text has long interested researchers, e.g., [6, 16, 19]. Many of these methods exploit linguistic patterns to infer if two keywords are related under a certain relationship. However, these approaches are not applicable to social metadata because it 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 [10] 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 [3]. Heymann and Garcia-Molina [7] 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 [15] applied a statistical subsumption model [14] 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.

Our present work, SAP, is different from our earlier approach, SIG [12] in many aspects. First, SAP exploits more evidence, i.e., structure and tag statistics of personal hierarchies rather than individual relations' co-occurrence statistics as in SIG. Second, SAP is based on the relational clustering approach that incrementally attaches relevant saplings to the learned folksonomies, as SIG exhaustively determines the best path out of all possible paths from the root node to a leaf, which is computationally expensive when the learned folksonomies are deep. Last, SAP demonstrates many advan-

tages as presented in Section 5.

The sapling merging approach described in this paper is an extension of collective relational clustering approach used for entity resolution [1]. That work proposed a method to identify and disambiguate entities, such as authors, that utilizes two types of evidence: intrisic and extrinsic features. Intrinsic features are associated with specific instances, such as author names, while extrinsic features are derived from structural evidence, e.g., co-authors 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 reasons, we use the naive version of the relational clustering, where we 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 [19]. 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. [4, 17]). 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.

7. 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. 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 (e.g., [18, 2]), which provides a systematic way to combine heterogeneous evidence, and takes into account uncertainties on similarities between concepts and relations.

Acknowledgements

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

8. REFERENCES

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

[2] M. Broecheler and L. Getoor. Probabilistic similarity logic. In *Proceedings of International Workshop on Statistical Relational Learning*, 2009.

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

[4] J. Euzenat and P. Shvaiko. *Ontology Matching*. Springer-Verlag, 2007.

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

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

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

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

[9] A. Mathes. Folksonomies: cooperative classification and communication through shared metadata. 2004.

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

[11] A. E. Monge and C. Elkan. An efficient domain-independent algorithm for detecting approximately duplicate database records. In *Proceedings of the SIGMOD workshop on Data Mining and Knowledge Discovery*, 1997.

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

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

[14] M. Sanderson and W. B. Croft. Deriving concept hierarchies from text. In *Proceedings of the conference on Research and development in information retrieval*, pages 206–213, 1999.

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

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

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

[18] J. Wang and P. Domingos. Hybrid markov logic networks. In *Proceedings of Association for the Advancement of Artificial Intelligence*, 2008.

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