Crowdsourcing Taxonomies*

Dimitris Karampinas and Peter Triantafillou

Computer Engineering & Informatics Department, University of Patras
{karampin,peter}@ceid.upatras.gr

Abstract. Taxonomies are great for organizing and searching web content. As such, many popular classes of web applications, utilize them. However, their manual generation and maintenance by experts is a time-costly procedure, resulting in static taxonomies. On the other hand, mining and statistical approaches may produce low quality taxonomies. We thus propose a drastically new approach, based on the proven, increased human involvement and desire to tag/annotate web content. We define the required input from humans in the form of explicit structural, e.g., supertype-subtype relationships between concepts. Hence we harvest, via common annotation practices, the collective wisdom of users with respect to the (categorization of) web content they share and access. We further define the principles upon which crowdsourced taxonomy construction algorithms should be based. The resulting problem is NP-Hard. We thus provide and analyze heuristic algorithms that aggregate human input and resolve conflicts. We evaluate our approach with synthetic and real-world crowdsourcing experiments and on a real-world taxonomy.

Keywords: Collective Intelligence, Crowdsourcing, Taxonomy, Tagging.

1 Introduction

Social media applications and research are increasingly receiving greater attention. A key defining characteristic is the increased human involvement. Even before todays' success of social media applications, many applications became extremely successful due to the clever exploitation of implicit human inputs (e.g., Google's ranking function), or explicit human input (e.g., Linux open source contributions). Social media and the social web have taken this to the next level. Humans contribute content and share, annotate, tag, rank, and evaluate content. Specialized software aggregates such human input for various applications (from content searching engines to recommendation systems, etc). The next wave in this thread comes from *crowdsourcing systems* in which key tasks are performed by humans (either in isolation or in conjunction with automata) [7]. Lately, within the realm of data and information retrieval systems, crowdsourcing is gaining momentum as a means to improve system performance/quality [3,13]. A

* This work was partially funded by the EIKOS research project, within the THALES framework, administered by the Greek Ministry for Education, Life Long Learning, and Religious Affairs.

E. Simperl et al. (Eds.): ESWC 2012, LNCS 7295, pp. 545–559, 2012.

recent contribution suggests to engage humans during the processing of queries for which humans are better suited for the task (e.g., entity disambiguation) [8]. Further, (anthropocentric) data systems are proposed whose functioning (including semantics enrichment and related indices) depends on the users' contributions and their collective intelligence [21].

The central idea is to respect and exploit the fact that for some tasks humans can provide excellent help. The challenge then rests on our ability to harness and properly aggregate individual input to derive the community wisdom and exploit it to solve the problem at hand. One particularly interesting problem is that of constructing taxonomies. Taxonomies provide great help for structuring and categorizing our data sets. As such, currently they are at the heart of many web applications: Products are available that exploit taxonomic knowledge in order to improve results in product search applications [1,2]. In local search, location taxonomies are used to improve results by utilizing a querying user's location and the known location of search result items to improve result quality. Google's search news, localized results, and product categories is a prime example for these. As another example, domain-specific (a.k.a. vertical) search engines utilize taxonomies (topic hierarchies) which are browsable and searchable using complex queries and receiving ranked results.

Our work rests on the realization that social media users (contributing and annotating content) have a good understanding of the fundamental (subtype-supertype) relationships needed to define a taxonomy for their content. For instance, biologists collectively can help define the taxonomy to be used, for example in a vertical (biological) search engine. Users from different locations can collectively define a locality taxonomy used in search engines with localization services. Traders in e-shops can easily collectively come up with product categorizations. These examples show that **humans can** offer great help! Further, the vast success enjoyed by a great number of social web applications, prove that **humans are willing** to provide such annotations. Hence, the human's willingness and ability can help solve a problem that is close to the heart of the semantic web community and which is addressed here: Crowdsourcing taxonomies inherently promise to provide a way to come up with high-quality taxonomies, based on the collective knowledge of its users, which will be dynamic and reflect the user-community's understanding of the data space.

2 Problem Statement, Rationale, and Challenges

Our model does not depend on any "experts". We adopt an automated approach, with the additional feature that users explicitly provide us with relations between the keywords (so-called "tags") they employ to annotate the content they share. Humans have a good understanding of the supertype-subtype relations between various thematic categories, since these naturally exist around them. So, we aim to exploit **extended tagging** and a categorization capability in order to develop high-quality taxonomies. We ask users to contribute with metadata in this format: $tag_a \rightarrow tag_d$. Here, tag_a is a supertype topic and represents a higher

node to a potential concept hierarchy whilst tag_d is a subtype topic. The arrow between them connotes an *Ancestor* $\rightarrow$ *Descendant* ($A \rightarrow D$) relation. In figure

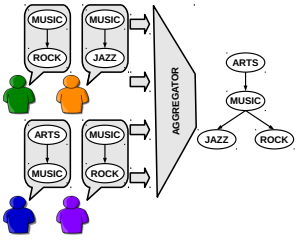


Fig. 1. Crowdsourcing Taxonomy Example

1 we demonstrate the basic idea of our effort. A community of users, forming a crowdsourcing environment, provides the system with tag relations. These can either be different between each other or depict the same relationship (i.e. Music $\rightarrow$ Rock). We refer to these tag relations also with the term **"votes"**, since they incorporate users' personal opinions for parts of the taxonomy tree. Our goal is to aggregate all given tag relations (votes) and produce a taxonomy that is derived using our community's knowledge/wisdom.

The problem is not easy! The following challenges arise:

- **Individuals are prone to errors.** Sometimes they specify incorrect tag relations, e.g., because of constrained knowledge and highly complex datasets. When building large scale taxonomies, the granularity level between nodes in adjacent levels is "fine" (especially at the lower levels of the taxonomy) and thus the frequency of such "false votes" may be high. So, contradicting opinions arise. But, this happens not only directly, but also indirectly, as a result of a combination of various relationships. Our goal is to resolve such conflicts. Since we depend on the crowd's wisdom, a natural discourse is to have the majority opinion prevail.
- **Incomplete (structural) information.** Users' knowledge might not be wide enough to completely cover an ideal, "golden rule" taxonomy (e.g. as constructed by experts). For example, in figure 1, suppose that a vote Arts $\rightarrow$ Jazz (which is valid) were given instead of the Arts $\rightarrow$ Music one. In this scenario we have evidence that both Arts and Music nodes are ancestors of node Jazz but we have no insight as far as their relative relationship is concerned. In this case, the system must be able to implicitly utilize the users' given input and fill in the missing structural information. This filling may be done incorrectly. Our approach will be to make a best guess (according to some metrics) and rely on future incoming votes to correct any such mistakes.
- **Incremental, online taxonomy development.** The problem is set in a dynamic environment. Users provide us with relations in an online way. The objective here is to introduce any newly incoming relations into the current structure in a cumulative manner. For every new vote, we shall be able to modify and alter the current taxonomy state without having to destroy or

build it from scratch. Based on the above example, suppose that we eventually receive an Arts → Music vote. If previously we had made a mistake when filling the lack of knowledge between Arts and Music, an efficient rectification must take place based on the now completed information. To sum up, dynamic, piece-wise, online taxonomy maintenance is is a key characteristic.

The "human-centric" approach we describe exploits users' knowledge (which is very difficult to correctly derive by automatic means) and produces taxonomies that are in accordance to the beliefs of the system's user base. In a sense, instead of constraining user input to characterizing the content (as is typically done in social media environments), we go one step further and allow users to provide with input that will lead to the construction of taxonomized datasets.

2.1 The Model, Solution Invariants, and Assumptions

As mentioned, the shape and structure of the taxonomy emerge from the crowd's *subjective* will, as it evolves. As the number of participants increases, in general, the higher the output quality becomes. When conflicts arise, several conflicting taxonomy states emerge as alternatives. One of them will be associated with the greatest number of votes. In this way, the new accepted state of the taxonomy will emerge. At the end, this process will converge to a structure, entirely defined from the community's aggregated knowledge. At this point, we can claim that this final product *objectively* depicts a complete taxonomy. But how do we evaluate such a taxonomy? We wish we could compare it against a golden rule taxonomy and see how they differ; but there is no standardized, "ideal" structure on which everyone agrees. Even if we compare taxonomies created by experts there will not be a 100% match, since both the rules for creating the taxonomy and the input data are often contradictory and obscure.

Users are asked to provide us with *Ancestor* → *Descendant* relations but any given vote has it's own interpretation, depending of the current state of the taxonomy. Any incoming tag relation will be classified to a category, based on the relative positions of the nodes it touches. For example, in figure 1 the following possible scenarios can arise as far as an incoming vote is concerned:

- **Ancestor → Descendant (A-D)**: i.e. Arts → Jazz. These relations practically increase our confidence for the current state of the taxonomy and generally leave it intact.
- **Descendant → Ancestor (D-A)**: i.e. Rock → Arts. These votes form what we call **backedges** and create cycles on the current structure. They require special handling and may change the current state of the taxonomy. They are not necessarily "false votes" according to a golden rule taxonomy and they are usefull, since they can restore possible invalid relations which were based on previous erroneous input or assumptions.
- **Crosslinks**: Relations like Jazz → Rock, do not belong to any of the former classes. This type of links interconnects nodes that have a common ancestor. If according to the golden rule taxonomy, there is a supertype-subtype

relationship between these nodes, our algorithms for handling this situation will be able to eventually yield the proper tree structure. If, however, there is no such relation between the nodes of a crosslink, then our algorithms will inescapably produce a supertype-subtype realtion between the two. When a relation like this arises, special handling is required: our current idea for this involves users supplying "negative" votes when they see incorrect crosslink relationships established in the current taxonomy. We leave discussion of this to future work.

Before we continue with the problem formulation we specify two **solution invariants** our approach maintains and give some insight of the taxonomy building algorithm that follows.

Tree Properties: This is the primary invariant we maintain. A tree is an undirected graph in which any two vertices are connected by exactly one path. There are no cycles and this is a principle we carry on throughout the taxonomy evolution. Starting with many shallow subtrees, as votes enter the system, we detect relations between more and more tags. The independent trees gradually form a forest and we use a common "Global Root" node to join them.

Maximum Votes Satisfiability: We also wish to preserve a *quantitative* characteristic. Our purpose is to utilize every incoming $Ancestor \rightarrow Descendant$ relation and embed it on the current structure. If this raises conflicts, our solution to this is to derive a taxonomy structure which as a whole satisfies the maximum number of users' votes. At this point we need to mention that according to our model, there is no constraint to the number of votes a user can submit to the system (see "free-for-all tagging" at [15]). Satisfiability is measured not on per individual basis but over all votes, overall.

Finally, we need to state that our solution does not take any measures to face synonyms or polysemy issues. Although according to [12] these are not major problems for social media, we admit that users tend to annotate their content with idiosyncratic tags which in our case can lead to wrong keyword interpretation and create links that users do not intent to recommend. This issue is orthogonal to our work since we focus on structural development and thus we can assume a controlled vocabulary without loss of generality.

3 Formal Formulation and Analysis

Leaving aside the added complexity due to the online nature of our venture, we show that even an offline approach yields an NP-Hard problem.

Note, that at first thought, one could suggest the following solution to our problem: First, create a directed graph $G(V, E)$ where each vertex $v \in V$ represents a given tag and each $e \in E$ represents a relationship between the two nodes. Edges bear weights w reflecting the number of votes from users for this relation. Intuitively, this calls to mind minimum/maximum spanning tree algorithms. Thus, second, run a "variation" of one of any well-known algorithms to retrieve a Maximum Spanning Tree. Because, however, our graph is directed, what we need here is a 'maximum weight arborescence' (which is defined as the

directed counterpart of a maximum spanning tree). However, this simplistic idea has a major flaw. If the graph is not strongly connected (something that regularly happens especially during the early stages of the taxonomy development) then a number of nodes has to be omitted from the final output.

Our problem is formalized as follows:
INPUT: Complete graph $G = (V, E)$, weight $w(e) \in Z_0^+$ for each $e \in E$.
OUTPUT: A spanning tree T for G such that, if $W(\{u, v\})$ denotes the sum of weights of the edges on the path joining u and v in T, then find B where:

$$B = max(\sum_{u,v \in V} W(\{u, v\})) \tag{1}$$

This problem is a straightforward instance of the Optimum Communication Spanning Tree problem in [9] and has been proven to be NP-Hard[1]. The key idea here for the matching of the two problems, is that **shortcuts** (edges weighting 0) are permitted. Obviously, as an algorithm proceeds online maintaining the tree invariant, there will be cases when nodes are connected with their edge having zero weights (as users may have not supplied yet any votes for this edge). B represents the maximum number of votes that is satisfied and along with the tree notion meets the standards set by the invariants in the previous section.

4 Crowdsourced Taxonomy Building Algorithm

As noted, our problem is NP-Hard. For n nodes, an optimal solution would require an exhaustive search of all possible n^{n-2} spanning trees and the selection of the one that maximizes value B in (1). So, we adopt a heuristic approach. We relax the second invariant: we demand the maximum number of satisfied votes, not overall, but only between consecutive algorithmic steps. Each vote is embodied into the taxonomy via an algorithm that introduces a series of transformations for every incoming vote (tag pair).

4.1 The Core Algorithm: CrowdTaxonomy

Algorithm 1 is called on every incoming vote. For every vote $(u \rightarrow v)$ we first need to identify whether the named nodes are new to the system or if they are already part of the tree. In case both nodes already exist (line: 9) we need to specify their relative position and thus we call a Lowest Common Ancestor (LCA) routine that returns the LCA node w. If $w = null$ (line: 11), there is no common ancestor and nodes u and v belong to different trees. If w coincides with u (line: 14), then u is already an ancestor of v, which is something that strengthens the evidence we have for the current state of the taxonomy. When w equals v (line: 16) the $v \rightarrow u$ relation introduces a conflict and implies that a modification may be needed. If we accept this edge, the structure's constraints are violated since a cycle is created.

[1] We consider the requirements equal to the standard basis vector and refer to the Optimization version.

Algorithm 1. VOTE PROCESSING

Require: A vote $tag_x \rightarrow tag_y$

```
 1: Node u ← hash(tag_x)
 2: Node v ← hash(tag_y)
 3: if ((u = null) and (v = null)) then
 4:    CREATE NEW TREE
 5: else if ((u ≠ null) and (v = null)) then
 6:    ATTACH NEW CHILD
 7: else if ((u = null) and (v ≠ null)) then
 8:    MERGE
 9: else
10:    Node w ← LCA(u, v)
11:    if (w = null) then
12:       MERGE
13:    else if (w = u) then
14:       CREATE FORWARD EDGE
15:    else if (w = v) then
16:       BACKEDGE CONFLICT RESOLUTION
17:    else
18:       EXPAND VERTICALLY
19:    end if
20: end if
```

Lastly, in case w is a separate node on the tree (line: 18), the new relation forms a crosslink and is handled appropriately. Hereafter, we describe every tree transformation triggered by each of these cases.

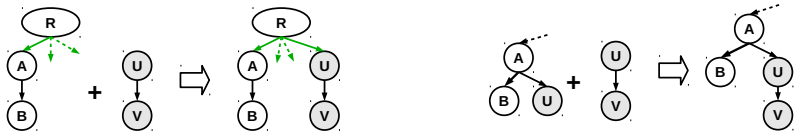
TRSFM CREATE NEW TREE**:** In this simple scenario the taxonomy does not yet include any of the two nodes of the new vote. So the ancestor node u is attached to the global root via a *shortcut* ($R \rightarrow u$) and node v plays the role of its child (see figure 2a.).

Definition 1 ***Shortcut:*** *An "artificial" Parent → Child link that is not an explicit user supplied vote,. Its weight is 0, and it is utilized to preserve structural continuation.*

This addition forms a new tree with only two nodes. In the future, it will be expanded with more nodes or get merged to another expanding tree.

TRSFM ATTACH NEW CHILD**:** This is another straightforward case. Node u pre-exists and the incoming relation can be easily assimilated by adding an extra child to it.

TRSFM MERGE**:** MERGE is used in two similar cases. In line 8 of the core algorithm we ask to attach a new node u to our taxonomy but in a generic scenario its descendant node v does already have a parent node. So does happen



(a) Create New Tree (b) Attach New Child

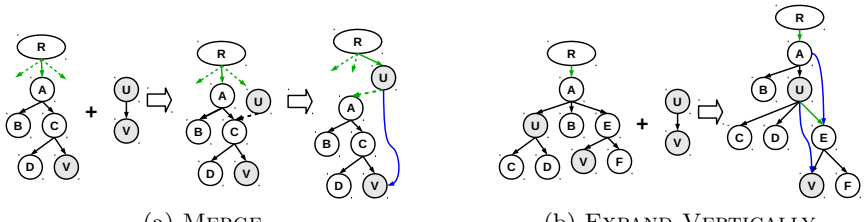
Fig. 2. Creating a new tree or attaching a new child

in line 12 where we need to annex u's participating tree to that one of v's with a link between them. In figure 3a we observe that both node C and u "compete for the paternity" of v. In order to maintain the tree properties, only one of the potential parents can be directly connected with v. We arbitrarily choose node C to be the direct ancestor (parent) of v and set node u to be parent of C which is in accordance with the *Maximum Vote Satisfiability* invariant since an *Ancestor* $\rightarrow$ *Descendant* ($u \rightarrow v$) relation takes place. The (temporary) state formed in the middle of figure 3a suffers from the same "paternity conflict" problem - now between A and u over C. Following the same reasoning, we finally place node u on top of v's tree being now the parent of v's root. Since there is no given relation yet between u and A we form a shortcut between them. We also maintain a *forward edge* from u to v so not to lose the information we have regarding the vote for the $u \rightarrow v$ relationship.

Definition 2 ***Forward Edge:*** *A latent relation between two nodes. The source node is an ancestor in the taxonomy and the target is a descendant. Forward edges do not refer to Parent* $\rightarrow$ *Child links and remain hidden since they violate tree's properties.*

The idea behind this transformation is that since we don't have enough evidence to decide on the partial order of v's ancestors we *temporarily* send u node to the root. Relations that will follow will illustrate the correct order.

If v is a root of a tree, an Attach New Child transformation is called.



(a) Merge (b) Expand Vertically

Fig. 3. The Merge and Expand Vertically transformations

TRSFM Expand Vertically**:** In this case the newly incoming vote is interpreted as a crosslink according to the current state of the taxonomy. As shown in figure 3b the logic we follow resembles at some point the Merge transformation. First, we locate the common ancestor A of u and v and break the link between it and the immediate root E of the subtree that includes v. The independent subtree is now linked with u via a shortcut formed between u and its root E. Two forward edges are spawned to indicate latent $Ancestor \rightarrow Descendant$ relations. For the sake of completeness we report here that in contrast to *common* edges between nodes, shortcuts are not converted to forward edges when the link that touches the two nodes brakes.

TRSFM Create Forward Edge**:** Straightforward scenario. A forward edge is added from u to v unless their node distance is 1 ($Parent \rightarrow Child$). In any other case (distance = 1) no operation takes place.

TRSFM Backedge Conflict Resolution **(BCR):** At line 16 of the Vote Processing algorithm we need to handle a vote whose interpretation is against the relationship of the nodes it touches in the current taxonomy state and any attempt to adjust it, leads to a backedge.

Definition 3 ***Backedge:*** *A latent relation between two nodes. The source node appears as a descendant in the current taxonomy whereas the target as an ancestor. Backedges remain hidden since they violate the tree's properties.*

Besides the cycle that it creates, a backedge might also violate the *Maximum Votes Satisfiability* invariant. The idea to solve this problem is to isolate the strongly connected component that the newly incoming backedge created and resolve any vote conflict locally.

We will present the logic of this transformation with a specific example on figure 4. For the sake of simplicity we assume that the incoming vote appears in a "triple form" and will be processed as such. On the left we observe the initial state of the isolated subgraph. We are asked to embody a $u \rightarrow v$ relation whose weight (number of votes) equals 3. We apply a what-if analysis. We instantiate all possible states, which the subraph component can reach, every time we apply a vertical rotation to its nodes. These are presented on the right part of the figure. The state whose backedges add up to the minimum weight is picked as a final state. In this example we arbitrary choose between state 3 or 4. Node A and its in- or out- going edges does not take part into this computation and is displayed here just to illustrate how it interfaces with the rest of the graph.

Formally, the aforementioned is an instance of the *All-pairs Bottleneck Paths* problem [20] and aims to find out the state in which we displease (or dissatisfy) the least number of votes. It's dual equivalent problem is the *All-pairs Maximum Capacity* and can be respectively interpreted as an effort to preserve our second invariant (*Maximum Votes Satisfiability*).

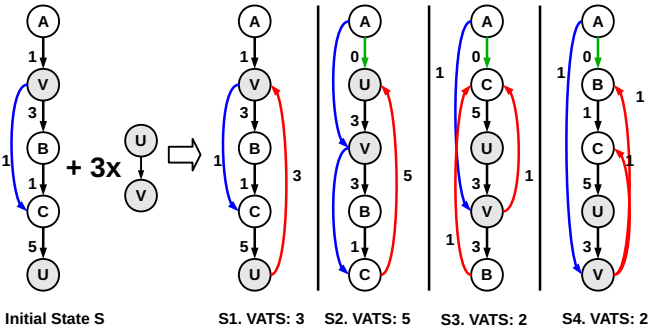


Fig. 4. Backedge Conflict Resolution What-if Analysis. VATS: Votes Against This State.

Elimination of Rising Crosslinks

In fig. 5 we observe the big picture during a BCR. The change caused by the appearance of $(u \rightarrow v)$, practically decomposes the taxonomy into 3 components. The one on top, T, right above v, remains put. The lower ones are being inverted after we identify and break the weakest edge between them. This transformation has some corner cases. Former forward edges, such as $b \rightarrow d$ now appear as crosslinks, violating the first invariant. A solution to this anomaly is offered by Algorithm 2, which simply certifies that all forward edges on path $\{v, ..., b\}$ adhere to the obvious ancestor-descendant relation.

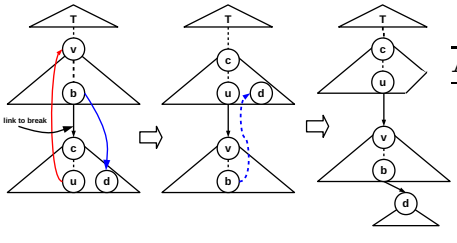


Fig. 5. The big picture of BCR and the algorithm for crosslinks elimination.

Algorithm 2. Crosslink Elimination

```
1: for all Nodes b_i ∈ {v...b} do
2:     for all forward edges (b_i, d) do
3:         if (b_i ≠ LCA (b_i, d)) then
4:             Expand Vertically(b_i, d)
5:         end if
6:     end for
7: end for
```

4.2 Asymptotic Complexity Analysis

The core Algorithm 1 is called once for every new vote. In turn, it calls at most one of the transformations. Therefore, its worst-case asymptotic complexity is equal to the worst of the worst-case complexities of each of the transformations. Having computed all these partial complexities we state that the worst one, is the one corresponding to Backedge Conflict Resolution. Every time a backedge appears it interconnects two nodes and the path connecting them (coined BCR path) has a maximum length of n, where n denotes the number of

total nodes. This is an extreme scenario where all the tree nodes form a chain. As we already stated, at this point, the BCR algorithm forms n possible tree instances, with every one of them representing a unique cyclic rotation of the nodes making up the BCR path. For every one of these n instances, we iterate over the n nodes it consists of and explore their outgoing edges to verify whether they form a backedge or not. The number of these outgoing edges is obviously also at most n-1. Therefore, the overall asymptotic worst-case complexity of the Backedge Conflict Resolution transformation is $O(n^3)$.

Theorem 1. *If s denotes the number of votes, the worst-case asymptotic complexity of CrowdTaxonomy Algorithm is $O(s * n^3)$.*

Proof. Vote Processing is called s times, once for each incoming vote. Every time a single transformation is applied. The worst-case complexity of the latter equals $O(n^3)$. Therefore the worst-case asymptotic complexity of the CrowdTaxonomy algorithm is $O(s * n^3)$.

This analysis depicts a worst-case scenario and is basically presented for the sake of completeness, vis a vis the NP-Hard result presented earlier. As the experiments showed, the algorithm's behaviour in matters of absolute time is approximately linear to the number of votes. This is because (i) BCR paths are much smaller than n and (ii) outgoing links per node are also much smaller than n. In future work we plan to present an analysis of the average complexity and provide with better estimations than n which is a very relaxed upper-bound.

5 Experimentation

We evaluate the CrowdTaxonomy algorithm and demonstrate its quality under: i) Lack of (structural) information and ii) the presence of conflicting votes. Further, we perform a real world crowdsourcing experiment: we test our initial assertion, that users are capable of providing valuable input when creating a taxonomy. We also test the quality of the taxonomy produced by CrowdTaxonomy with real-world input. Our metrics for evaluating the resulting taxonomy are based on the similarity between it and the golden rule one. Consider a taxonomy T_o as the golden rule taxonomy, and T as the one derived from our algorithm. Let R_o be the set of all $Parent \rightarrow Child$ relations on the T_o and R the set of all $Parent \rightarrow Child$ relations on T. We define:

$$Recall = \frac{|R_0 \cap R|}{|R_0|}, Precision = \frac{|R_0 \cap R|}{|R|}, FScore = \frac{2 \cdot Recall \cdot Precision}{Recall + Precision}$$

$|S|$ denotes the cardinality of a set S.
Recall certifies the completeness of the taxonomy, whereas *Precision* measures its correctness. *FScore* is a combination of the former into a harmonic mean.

We use the ACM Computing Classification System (version 1998) as the golden rule taxonomy. It contains 1473 concepts, forming a four-level tree. We

"break" this tree into distinct node (pair) relations, generate additional conflicting pairs and feed (correct and incorrect pairs) to our algorithm.

Our algorithm was written in C and we used GLib. The interface of the crowdsourcing experiment was implemented in HTML/PHP and ran over Apache/MySQL.

5.1 Synthetic Experiments

A key challenge we face is to provide a high-quality taxonomy under incomplete structural information. The votes are given ad-hocly and there is no guarantee that they form a complete taxonomy. To examine this characteristic we form all possible *Ancestor* → *Descendant* relations of the ACM tree and gradually feed the algorithm with a fraction of them. The result is shown in figure 6.

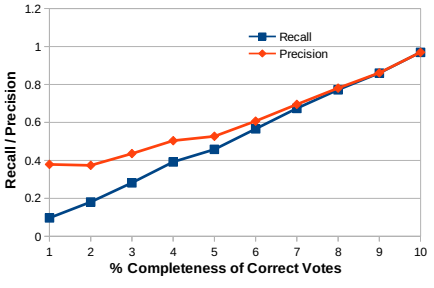


Fig. 6. Recall and Precision using a percentage of Correct votes

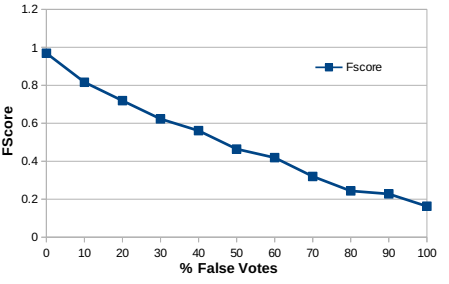


Fig. 7. FScores having a mixture of Correct and False votes

We repeatedly increase by 10% the number of available votes and measure the quality of our taxonomy. The absence of *Descendant* → *Ancestor* relations keeps the number of *permanent* backedges to zero. Any backedges that arise are instantly resolved and one, and only one, tree instance (representing 0 conflicts) is produced. The Recall value is linear to the size of input as expected. The Precision metric equals 0.4 at the begining and gradually increases. The taxonomy edges consist of a set of correct edges and a number of additional shortcuts, with a fraction of them playing a role of 'noise'. As we reach 100% input completeness, the assumptions (shortcuts) are gradually eliminated and both metrics end to 1.0 verifying a sanity-check, since the experiment configuration deals with only correct votes. Next we 'infect' the input with additional false votes. The number of correct *Ancestor* → *Descendant* votes remains 100% but on every iteration we increase the number of false (*Descendant* → *Ancestor*) votes. We form these by reversing the direction of a relation between two nodes. In figure 7 we observe that even with a high number of bad input, the taxonomy quality is high. For instance, even with 30% false votes, the F score remains above 60%. Under normal circumstances, where users can provide a largely complete and correct set of A-D relations, our algorithms produce a high quality taxonomy. Note that in this configuration $Recall = Precision = FScore$.

5.2 Crowdsourcing Experiment

We asked students of our department to voluntarily participate in crowdsourcing a taxonomy. We pregenerated all the possible *Ancestor* $\rightarrow$ *Descendant* relations and derived all the 'false' *Descendant* $\rightarrow$ *Ancestor* counterpart votes. We removed all misleading concepts (e.g., with labels 'General' or 'Miscellaneous') and attached their child nodes to their direct ancestors. As the total number of possible A-D relations is very high we used a fraction of the tree with 245 nodes which led to 620 relations (A-D) plus their counterparts (D-A). The users were presented with pairs of votes, the original correct vote, and the (inverted) false one. They had to choose between the one that depicted a $A \rightarrow D$ relation, linking two concepts of the ACM tree nodes. The user task consisted of 25 pairs of votes that were presented in groups of 5. Users could also select not to answer a vote-question marking it as *unspecified*. We counted 102 distinct http-sessions but the number of collected votes was smaller than the theoretical (2550), since some users dropped out early. In contrast to commercial crowdsourcing hubs, our user base had no financial or other type of gain. We collected input for a 3-day period. Some statistics of the experiment are shown on Table 1. We observe that the false/correct votes ratio is 0.283, which testifies that users are capable of providing high-quality input. Overall, the input accounted for 94.7% of all correct relations. With these statistics, we ran a synthetic experiment with controlled input, which predicted an $FScore$ ($Predicted$) whose value is close to the one in the real-world experiment. Thus, we can conclude that (i) the real-world experiment corroborated our conclusions based on the synthetic one; (ii) users can indeed provide high-quality input en route to a crowdsourced taxonomy, and (iii) despite the voluntary nature of user participation and the heuristic nature of our algorithms, the end result is a taxonomy preserving the large majority of the relationships found in expertly constructed taxonomies.

Table 1. Crowdsourcing Experiment Statistics

Total Votes	2155
Correct Votes	1501
False Votes	435
Unspecified	229
FScore	0.486
FScore (Predicted)	0.519

6 Related Work

[4], [5], [19] and [18] apply association mining rules to induce relations between terms and use them to form taxonomies. For text corpora, Sanderson and Croft automatically derive a hierarchy of concepts and develop a statistical model where term x subsumes term y if $P(x|y) \geq 0.8$ and $P(y|x) < 1$ where $P(a|b)$ defines the probability of a document to include term a assuming that term b is contained. Schmitz extended this and applied additional thresholds in order

to deal with problems caused by uncontrolled vocabulary. [6] and [14] underline the importance of folksonomies and the need to extract hierarchies for searching, categorization and navigation purposes. They present approaches that operate based on agglomerative clustering. A similarity measure is used to compute the proximity between all tags and then a bottom-up procedure takes place. Nodes under a threshold are merged into clusters in a recursive way and eventually compose a taxonomy. Heyman & Garcia-Molina in [11] present another technique with good results. Given a space of resources and tags, they form a vector for every tag and set to the $i-th$ element the number of times it has been assigned to object i. They also use cosine similarity to compute all vectors' proximities and represent them as nodes of a graph with weighted edges that correspond to their similarity distance. To extract a taxonomy they iterate over the nodes in descending centrality order and set every of its neighbours either as children or to the root based on a threshold.

As Plangprasopchok et al. note in [17] and [16] all these approaches make the assumption that frequent words represent general terms. This does not always hold and any threshold tuning approach leads to a trade-off between accurate but shallow taxonomies against large but noisy ones. Also, all above works assume a static tag space, despite its dynamicity [10].

7 Conclusions

We have presented a drastically new approach to create taxonomies, exploiting the wisdom of crowds and their proven desire and ability to provide rich semantic metadata on several social web applications. Our contributions include the definition and analysis of the problem of crowdsourcing taxonomies. We showed how to model the problem and the required human input and the (meaningful in a crowdsourcing environment) invariant of maximum vote satisfiability. Then we proceeded to show that the resulting problem is NP-Hard. Next, we contributed a novel heuristic algorithm to online aggregate human input and derive taxonomies. We conducted both synthetic and real-world crowdsourcing experiments. Our synthetic experiments showed that when the human input is adequately complete and correct, our solution can derive high-quality taxonomies. Conversely, even when the input is incomplete and incorrect to a significant extent, still a good quality taxonomy can be constructed. Our real-world crowdsourcing experiment additionally showed that indeed humans can provide high quality input in terms of completeness and correctness (at least for the taxonomy examined). And as our synthetic results showed, when fed into our algorithms, good quality taxonomies emerge. To the best of our knowledge this is the first work to study this problem and provide a promising solution.

Future work includes optimizations, straddling the complexity-quality trade-offs, and appropriate measures for crowdsourced taxonomy quality evaluation.

References

1. Endeca, http://www.endeca.com/
2. Facetmap, http://facetmap.com/
3. Alonso, O., Lease, M.: Crowdsourcing 101: Putting the wsdm of crowds to work for you: A tutorial. In: International Conference on WSDM (February 2011)
4. Au Yeung, C.-M., Gibbins, N., Shadbolt, N.: User-induced links in collaborative tagging systems. In: Proceeding of the 18th ACM Conference on Information and Knowledge Management, CIKM 2009 (2009)
5. Barla, M., Bieliková, M.: On deriving tagsonomies: Keyword relations coming from crowd. In: Computational Collective Intelligence. Semantic Web, Social Networks and Multiagent Systems (2009)
6. Brooks, C.H., Montanez, N.: Improved annotation of the blogosphere via autotagging and hierarchical clustering. In: Proceedings of the 15th International Conference on World Wide Web, WWW 2006 (2006)
7. Doan, A., Ramakrishnan, R., Halevy, A.Y.: Crowdsourcing systems on the world-wide web. Commun. ACM (2011)
8. Franklin, M.J., Kossmann, D., Kraska, T., Ramesh, S., Xin, R.: Crowddb: answering queries with crowdsourcing. In: ACM SIGMOD Conference (2011)
9. Garey, M., Johnson, D.: Computers and Intractability: A Guide to the Theory of NP-Completeness
10. Halpin, H., Robu, V., Shepherd, H.: The complex dynamics of collaborative tagging. In: 16th WWW Conference (2007)
11. Heymann, P., Garcia-Molina, H.: Collaborative creation of communal hierarchical taxonomies in social tagging systems. Technical report (2006)
12. Heymann, P., Paepcke, A., Garcia-Molina, H.: Tagging human knowledge. In: Third ACM International Conference on Web Search and Data Mining, WSDM 2010 (2010)
13. Ipeirotis, P.: Managing crowdsourced human computation: A tutorial. In: International Conference on WWW (March 2011)
14. Liu, K., Fang, B., Zhang, W.: Ontology emergence from folksonomies. In: 19th ACM CIKM (2010)
15. Marlow, C., Naaman, M., Boyd, D., Davis, M.: Position Paper, Tagging, Taxonomy, Flickr, Article, ToRead. In: Collaborative Web Tagging Workshop at WWW 2006 (2006)
16. Plangprasopchok, A., Lerman, K.: Constructing folksonomies from user-specified relations on flickr. In: 18th WWW Conference (2009)
17. Plangprasopchok, A., Lerman, K., Getoor, L.: Growing a tree in the forest: constructing folksonomies by integrating structured metadata. In: 6th ACM SIGKDD Conference (2010)
18. Sanderson, M., Croft, B.: Deriving concept hierarchies from text. In: 22nd ACM SIGIR Conference (1999)
19. Schmitz, P.: WWW 2006 (2006)
20. Shapira, A., Yuster, R., Zwick, U.: All-pairs bottleneck paths in vertex weighted graphs. In: 18th ACM-SIAM SODA Symposium (2007)
21. Triantafillou, P.: Anthropocentric data systems. In: 37th VLDB Conference (Visions and Challenges) (2011)