Automatically Inferring Tag Hierarchies in Social Tagging Systems

SAMUEL C. GRUBER, ADAM K. MIHALCIN and PRASHANT SRIDHAR
Carnegie Mellon University

We propose new models to induce a hierarchy over a set of tags in a social tagging system, also known as a folksonomy. A heirarchy is a set of hypernym hyponym relationships that forms a spanning forest over the set of tags Such a heirarchy induces a structure on the classes of the ontology built from all the tags. This ontology could then be used to provide better information retreival algorithms to search for tags on the website. It can also be used to suggest tags that a user might have missed while posting his question based on the tags he included. We compare hierarchical clustering based solely on the tag descriptions with a method that first classifies tags into classes and then performs hierarchical clustering on the classes, and finally a method that looks at each tag as a set of posts, and considers two tags to be linked in a parent-child relationship if one tag's set of posts is close to a subset of the other's set of posts.

Categories and Subject Descriptors: I.2.6 [**Artifical Intelligence**]: Learning—*Knowledge Acquisition*

General Terms: Algorithms, Experimentation

Additional Key Words and Phrases: Folksonomy, hierarchy, social tagging

ACM Reference Format:

Samuel C. Gruber, Adam K. Mihalcin, and Prashant Sridhar. 2013. Automatically Inferring Tag Hierarchies in Social Tagging Systems.

1. INTRODUCTION

Communities that are built around user-generated content often develop tagging systems to organize their large corpora of documents. These tagging systems become vast, and they depend upon the content creators to appropriately label their contributions to the community. However, it is often difficult for users, particularly those who are new and inexperienced, to accurately classify their submissions in full detail. As a result, discovery of relevant documents becomes more difficult, and these communities are less effective than they could be.

Furthermore, a tag hierarchy can prove useful in searching a database of community-generated content. One of the main reasons to tagging community contributions is to improve search quality by interpreting the tags as alternate descriptions of community contributions. Information retrieval research has already explored how to use alternate descriptions of a document to improve results: indeed, even the single BM25 retrieval method has seen at least two papers devoted to this very topic, one on the subject of extending BM25 to multiple fields [Robertson et al. 2004], and one on extending BM25 to XML documents with hierarchical fields [Lu et al. 2006].

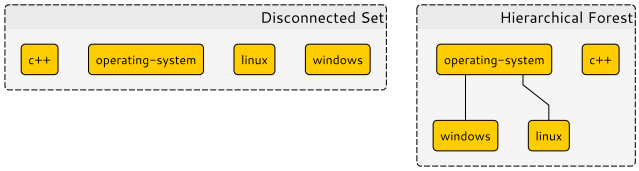

Fig. 1. The conventional model of a folksonomy as a set of disconnected tags, as compared to a more semantically accurate model as a forest.

Given a tag hierarchy, though, this work could be extended even further. The tag hierarchy is similar to an ontology, or relationship between a set of concepts within a domain. Mandala and other authors examine the use of WordNet to improve information retrieval performance, and two problems they solve relate to missing data in WordNet: WordNet is missing many terms, and many relationships between even the terms it contains [Mandala et al. 1998]. Part of the reason for these problems is that the construction of a database such as WordNet is laborious and expensive, and part of the reason is that WordNet is not meant to contain all terms and relationships. WordNet has hitherto not focused on specialized technical terms, and focuses on synonym, hypernym, and hyponym relationships between terms. Social tagging might be able to fill in these gaps, because a large number os social tagging systems exist as part of forums, question-and-answer sites, wikis, and blogs (a particular blog usually has tags only from one author, but the set of all blogs devoted to a particular topic comes from a wide variety of authors, making this set of blogs similar to a single very large forum or wiki).

We propose a system that can analyze a folksonomy, which is a set of community-generated tags, in association with a corpus of documents labeled from that folksonomy, and develop a hierarchy of those tags. Currently, tags are often represented as completely disconnected members of a large set, but they often fall naturally into a set of hierarchical trees. For instance, an existing system might represent the tag set {windows, linux, operating-system, C++}, but the correct hierarchy is {windows : operating-system, linux : operating-system, operating-system, C++} (Figure 1).

We define a hierarchy as a forest of rooted trees, in which each node represents a distinct tag. The hierarchy should be organized not simply by a definitional hypernym-hyponym relationship, but such that when a document is labeled with a particular tag, it should be labeled with all of that tag's ancestors in the hierarchy. In this way our hierarchy captures the conceptual model of tagging that practiced by users of community-generated content collections to

label documents with the most complete descriptions of their content.

These hierarchical relationships have several applications in the context of community-generated content aggregators. By integrating this data into search, more accurate results can be returned to the user, by expanding the input search terms to consider related tags. These hierarchies also have applications in tag suggestions, so that a user entering a specific term, such as "wostringstream" would be suggested other tags which are ancestors in the hierarchy, such as "cstringw", "c++", "mfc" and "unicode", though the suggested tags may not have direct hypernym-hyponym relationships to the original tag.

2. DATASET

Our results were obtained over the StackOverflow dataset, which is the largest and most well-developed of the StackExchange network of question-and-answer sites. In our analysis, we consider the corpus of questions asked on StackOverflow as available in March 2013, the set of tags available at that time, and the set of community-generated descriptions associated with those tags.

The presence of community-generated tag descriptions alongside the tags and posts themselves offers an degree of information about the tags that is not present in most community-generated, tagged corpora. This information allows these tags to be compared directly, rather than through secondary information such as associations with posts or users that is typically the only data available on a folksonomy.

We define our folksonomy as a tuple $\mathbb{F} := (T, P, L)$ where T is the set of all tags present in the dataset, P is the set of all posts, and $L : P \to \mathcal{P}(T)$ is a mapping from a post to its set of labeling tags. For our StackOverflow dataset, $|P| = 4708656$, $|T| = 32403$, and $\sum_{p \in P} |L(p)| = 13671739$. Our analysis is restricted to posts which possess at least one tag, and tags which label at least one post.

3. RELATED WORK

Existing models of tag suggestion do not consider the natural hierarchical relationships present in a folksonomy. As a result, their effectiveness is limited to suggesting other tags which have been observed in conjunction with the tags that have already been input. This model relies on a large portion of users to accurately label their content all the way up the hierarchy of the folksonomy; however, many users will only label near the bottom of the tree.

Cattuto explores a variety of algorithms for analyzing the relationships between tags in a folksonomy [Cattuto et al. 2008]. However, these algorithms only capture relationships that occur directly between each pair of tags. In this way, all of the algorithms presented by Cattuto stop short of leveraging the inherent hierarchy across the folksonomy.

Research has also been done into automatically extracting hypernym-hyponym pairs from text documents for building hierarchical models [Snow et al. 2004]. Snow's research has little applicability to the problem of constructing folksonomy hierarchies, however. Firstly, the user-generated content over which the folksonomy hierarchy is to be used is unlikely to contain the relationships that this algorithm could find in its text, because the tags themselves are not typically referenced in the body of the post. Additionally, Snow relies upon a hypernym-hyponym model expressed "A is a B", and similar such relationships, which is not an appropriate model for analyzing user-generated content folksonomies. In these corpora, the hierarchy must express that when a document is related to tag A, it is also implicitly related to B.

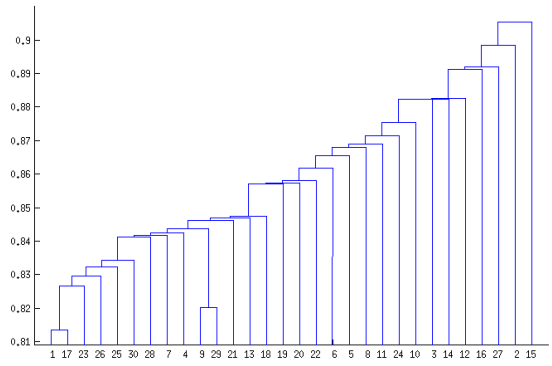

Fig. 2. Dendrogram representing the hierarchy based on tag description comparisons. Observe the rightward imbalance, indicating that the hierarchy contains many singleton nodes at high levels.

4. CLUSTERING BY DESCRIPTION SIMILARITY

The StackOverflow dataset provides not only a corpus of posts and their associated tags, but also detailed community-generated descriptions of these tags. These descriptions suggest the opportunity to directly analyze the tags against each other in order to discover clustering relationships and build a hierarchy. Intuitively, one is likely to assume that the descriptions of related tags would possess similarities that would become apparent as a linkage in the clustering process. In short, we treat each tag as a document, and apply hierarchical clustering to those documents, which has been well-studied in the information retrieval community [Zhao et al. 2005].

We construct a description correlation graph D in order to model the relationship between tags on the basis of their descriptions. Let the vertex set $D_V = T$, let $w(t)$ return the set of all words in the description of the tag, and let the edge weights be defined by Equation 4 below.

$$tf(t,d) = \frac{freq(t,d)}{\max_{w \in d} freq(w,d)} \tag{1}$$

$$idf(t,D) = \log \frac{|D|}{|d \in D : t \in d|} \tag{2}$$

$$W(i,j) = \sum_{t \in w(v_i) \cap w(v_j)} tf(t,v_i) \cdot tf(t,v_j) \cdot idf(t, \bigcap_{p \in P} w(p)) \tag{3}$$

$$E_{ij} = e^{-W(i,j)} \tag{4}$$

Note that the tf term is large for a given word if it is highly frequent in a given document but is rare throughout the corpus signifying it is important. For each word t, $W(i,j)$ will have a corresponding large term only if $tf(t)$ is large for both the descriptions. This implies that the word occurs frequently in both descriptions but rerely in the rest of the corpus. This would imply that the two descriptiptions are similar. Since $W(i,j)$ is a similarity metric, we used $e^{W(i,j)}$ as a distance metric for clustering.

Consider also the hierarchy graph H, where initially $H_V = T$ and $H_E = \emptyset$. Our algorithm to produce a hierarchy across the tags then proceeds as follows: At each iteration, select the edge of maximum weight. Let the two vertices connected by these edge be X and Y. We create a new node Z and weight its edges as given by Equation 5. Then remove the two vertices X and Y from D and merge the corresponding nodes in H.

$$W(v_i, Z) = W(v_i, X) + W(v_i, Y) \tag{5}$$

At the end of this progressive merging operation, we are left with a single vertex in D, and a single tree in H.

An intuitive expectation would be that similar concepts would have possessed similar descriptions. However, we found that clustering on the basis of similarities between the tags' descriptions using tfidf did not produce a model which could discover relationships between even very similar tags.

We also tried assigning each discription with a vector containing a feature for each word in our dictionary and clustering based on the cosine distance between two discriptions. This did not however, turn up significantly better results; the tree still resembeled a linked list.

Since the algorithm clustered from bottom up to build a tree, if it encountered poor edges in the lower levels of the tree, it might give poor top level clusters. To this end we attempted to train a multiclass SVM [Chang and Lin 2011] to partition the tags into six classes and then build a tree in each class. We hand annotated a hundred tags into one of six classes and used this as training data to classify the remaining tags. To do this we used an inverted list of all the posts fo each tag. The inverted list for a tag is simply the set of the post IDs that are marked with that tag. We can view this inverted list as a sparse vector with a one if it is a tag of the i^{th} post and a zero otherwise. Given these inverted lists, we used an RBF kernel to build the SVM. Further, we used 5 fold cross validation to select C and γ for the model. However, when cross validating to evaluate the performance of this algorithm, the best we could do was a success rate of 0.28. Hence the results from this approach were not very good.

5. SOFT SUBSET HIERARCHY

We propose a new method for constructing a hierarchy over a folksonomy which labels an existing document corpus. The intuition of our method is that a tag which frequently occurs in conjunction with another tag is a likely child of that tag in our hierarchical model. Because it measures only the sets of posts which are labeled with each tag, this method also does not rely on quality tag descriptions, as the above clustering process does. Thus it is both more widely applicable to other community-generated, tagged corpora which may not possess internal tag definitions, and also it is more responsive to revisions to those corpora, for the distributions of tags across the dataset is continually evolving, but the descriptions of the tags are relatively stagnant.

We begin by defining the relation $A : T \to \mathcal{P}(P)$ which maps a tag to the set of all posts that it labels. This function is used to construct the tag correlation graph C with the vertex set $C_V := T$ and the edge weights defined by Equation 6 below. Self-loops are ignored when constructing the graph, as their weight would always be equal to 1, and they have no value in constructing a hierarchy.

$$W(v_i, v_j) = \frac{|A(v_i) \cap A(v_j)|}{\min[|A(v_i)|, |A(v_j)|]} \tag{6}$$

Table I. Number of Unpruned Relationships in Soft-subsets by Threshold

0.05	341,228	0.10	185,941
0.15	116,981	0.20	90,948
0.25	70,895	0.30	54,007
0.35	40,740	0.40	36,574
0.45	31,471	0.50	29,364
0.55	20,377	0.60	18,371
0.65	16,099	0.70	13,538
0.75	12,150	0.80	10,376
0.85	8,720	0.90	7,494
0.95	6,322	1.00	5,610

A significant correlation is defined as occuring when the size of the intersection is greater than or equal to some ratio α to the size of the smaller post set. If such a correlation is observed, the pairing is recorded as a parent-child relationship where the tag with the greater number of associated posts is the parent, and the lesser number of posts is the child. Table I shows the quantity of relationships that are captured at various values of α. The range of captured relationships is fairly narrow, in comparison to the total $|T|^2 = 533,863,887$ possible relationships. Therefore we reason that the choice of α has little effect on the resulting hierarchy. We provide an example post with suggestions in Figure 4 which compares a threshold of 0.75 to a threshold of 0.50.

Since each child is smaller than its parent we do not introduce any cycles into the graph over our folksonomy, and therefore it remains topologically sortable and thus a hierarchy. To simplify the output graph, we consider all parents of each tag, and if that parent is reachable by a longer path, meaning that there is some intermediate tag which is a descendent of the parent and an ancestor of the child, we remove the edge in question (Figure 3). In this way we ensure that there is only one path between a tag and each of its ancestor tags, and that this path passes through as many intermediate levels of the hierarchy as possible.

Lastly, to ensure that the hierarchy is represented as a forest of disjoint trees, we apply the Chu-Liu/Edmonds algorithm [Tarjan 1977] to the graph to select the best single parent node for each child tag. This algorithm finds the maximal branching on a weighted directed graph $G = (V, E)$, where a branching is defined as a set B of edges such that:

(1) If $(x_1, y_1) \in B$ and $(x_2, y_2) \in B$ are distinct edges, then $y_1 \neq y_2$. In other words, each node has at most one incident edge.

(2) B does not contain a cycle.

We do not, however, need to implement the full algorithm. Instead, we just scan through all the nodes, and for each node and each scored set of parents, take the parent with maximal score, which can be accomplished in $O(|E|)$ time rather than the $O(|E| \log |M|)$ time required for the Edmonds algorithm. Thankfully, this algorithm produces the optimal forest, as we prove in Theorem 1.

THEOREM 1. *Given a pruned set of potential parents, the problem of finding the maximum total weight set of edges is solved by simply taking the maximal-weight potential parent for each node.*

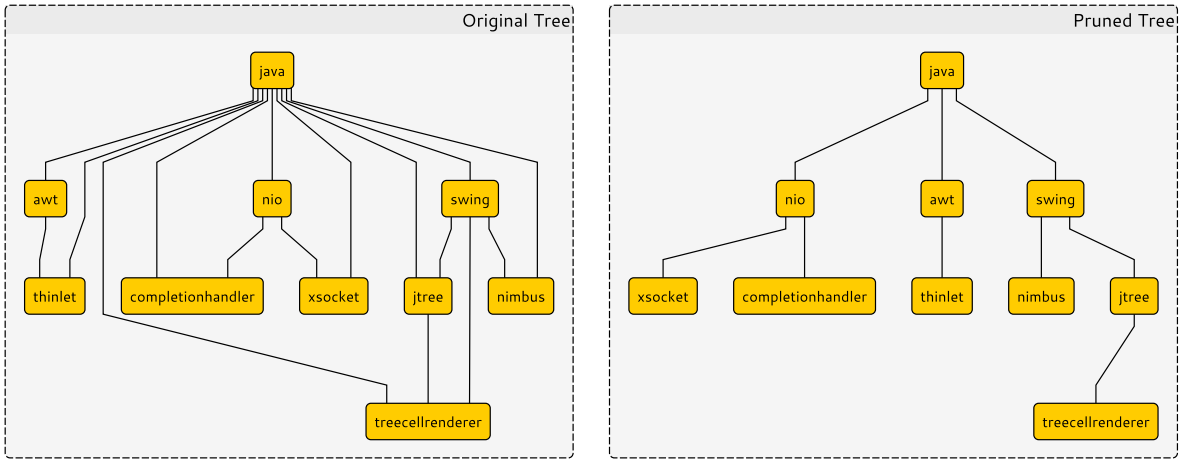

Fig. 3. A partial tree beneath the java parent tag. Initial soft subset analysis produces the tree at left with many redundant edges. By removing these edges, we arrive at a simpler tree, shown at right, which better corresponds to actual semantic relationships, and is additionally less costly to traverse.

PROOF. *Let G be the weighted directed graph (V, E), consisting of all tags (nodes) V and and all potential child-parent edges E, but reversed to point from parent to child rather than child to parent. This graph represents exactly the same relationship, but it represents trees with the more familiar link from parent to child, as Tarjan does.*

Without loss of generality, we assume that G is weakly connected (if it isn't, we split into weakly connected components, and apply this proof to each, just as the Chu-Liu/Edmonds algorithm is applied separately to each connected component). Then, let E_{PARENT} be the set of edges taken by the algorithm that simply chooses the maximum-weight parent for each node, if such a parent exists. Let H be the set of edges taken by the Chu-Liu/Edmonds maximal branching algorithm [Tarjan 1977].

Notice that there are no cycles in the original graph G, since only nodes with larger post sets can be parents of nodes with smaller post sets. Therefore, the only possible strongly connected components are individual nodes. By Tarjan's lemma 1, then, each node v of $G(H)$ —the subgraph with edges G —has at most one edge (u, v). Furthermore, each weakly connected component W of $G(H)$ contains exactly one root component i.e. root node.

Because there is at most one edge $(u, v) \in H$ for every node v, we can specify H by giving at most one u parent for every node v. We claim that H always specifies this u if v has any potential parents, and that the u specified in H is always the largest-weight parent of v.

Let us assume, for sake of contradiction, that there exists some edge (u, v) in the graph G, but H does not include any edge incident to v. But then H is not maximal! The set of edges $H \cup \{(u, v)\}$ will still be a branching, because it will not contain two edges incident to a single node, and because there is no way to create a cycle from the edges in E. Therefore, H must contain (u, v) i.e. H specifies a parent for every node v in the graph with at least one potential parent.

Now let us assume, for the sake of contradiction, that there exists some edges (u, v) and (u', v) in the graph G, where the first edge has greater weight than the second edge does, but that H includes edge (u', v). Certainly, because H is a branching, $H \setminus \{(u', v)\}$ is a branching as well, which does not contain any edge incident to v. Therefore, $H' = (H \setminus \{(u', v)\}) \cup \{(u, v)\}$ is a branching as well, since it includes only one edge incident to v and no cycles are possible from the edges in E. Notice that H' has larger weight than H, contradicting the assumption of optimality of H.

Therefore, we have shown that H must include one parent for each node v if any possible parent u exists (i.e. there exists node u such that $(u, v) \in E$), and furthermore that this u must be chosen as the node whose edge to v has maximal weight. This is exactly how the set E_{PARENT} is constructed, so $H = E_{PARENT}$. □

6. RESULTS

Validations for these models are difficult to produce because they rely on true experts to verify the results. Precision is relatively simple to compute given gold-standard judgments, which can be determined by asking human experts which of the assigned tags are correct. Finding such experts is expensive, and this task would require quite a bit of time. However, it is *prohibitively* expensive to compute recall with human judgments, since that requires experts to consider all the possible tags that could have been applied to each post, and determine which should have been applied.

Our evaluation task is complicated by the fact that we are dealing with unsupervised learning algorithms (except for the SVM learning step we tried, which uses a small amount of hand-labeled classification data for tags). Consequently, we propose a method for evaluation, which is completely automated and does not suffer from the lack of gold-standard data. In particular, we implement one of the potential applications for tag hierarchies, and compute performance on that application.

One of the reasons for computing a tag hierarchy in the first place is that the existing corpus of posts lacks full detail in terms of tagging, particularly in that tags which would occur higher in a hierarchy are frequently omitted from actual content. We cannot know for sure which tags are omitted from content without expert judgments, but we can tell whether a tag suggestion engine we implement would at least save the user time in typing tags. Therefore, we implement a tag suggestion engine, which trains on the first 2.77 million posts, and tests on the last 223 thousand posts. Any post in the test set with only one tag is discarded, and then, for each post in the remaining test set, the suggestion engine is given a randomly

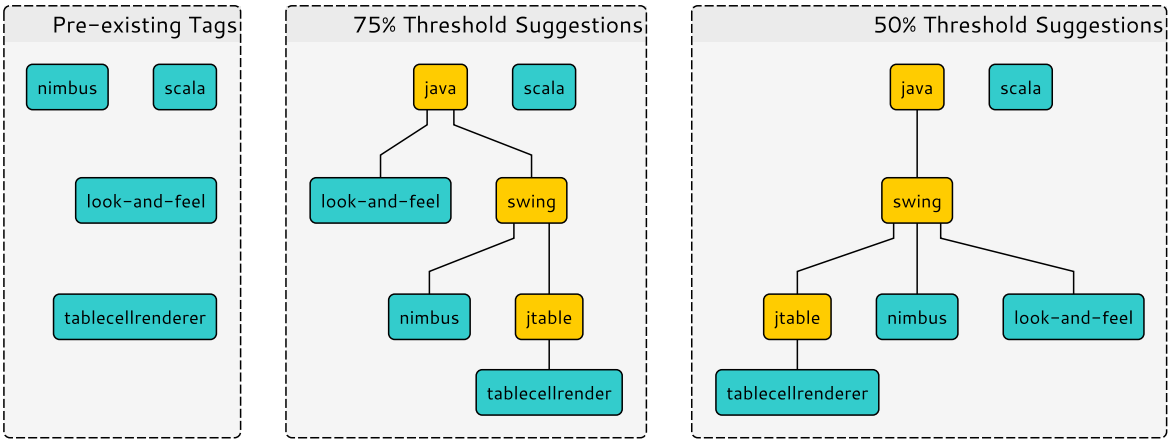

Fig. 4. Tags for post 1326763 "Table Cell renderer using Nimbus and Scala". A comparison of the user-defined tags and the tags returned by our subset hierarchy with a 75% threshold on subset size and 50% threshold. Decreasing the threshold does not alter the suggestions, but it does result in a more accurate suggestion tree.

chosen subset of tags which is half the size of the true set of tags. Its recommendations are intersected with the set of hidden tags, and the percentage of tags found is reported as the precision.

Our suggestion engine simply suggests the parents of all given tags as the set of new tags, which is similar to one of the methods of [Cattuto et al. 2008].

Our suggestion validator returned a precision rate of 5.3%; however, we attribute this largely to the fact that Stack Overflow only allows a maximum of five tags on any post. This means that users will tend to pick tags from very different subtrees in order to cover the entire content of their posts, and therefore our validator is unlikely to test our suggestions against tags which are similar to the inputs. If users were able to tag their posts more completely, we expect our results to improve significantly.

7. CONCLUSIONS

In this paper, we examined the problem of inducing a tag hierarchy over a folksonomy. We examined two algorithms which treat the tag description as a document and then perform standard document clustering. These algorithms do not work well, so we used the soft subset algorithm, which examines the post vector for each tag and finds relationships between tags based on the intersection of the posts for two tags. We found that this soft subset relationship produced superior trees.

8. FUTURE WORK

In the future we would like to validate the tag suggestions using human experts, which allows us to test the precision of our method and to develop an on-line algorithm which incorporates improved tagging as a result of these suggestions into future suggestions. This closely approximates the methodology that would be taken by a user-generated content community itself, where the suggestions would be returned to the creator as tags are being input.

Even this verification system only allows us to test our algorithm's results for precision, not recall. In order to test recall, we would need to possess the complete set of tags appropriate to a particular approach and then verify against this set. A direction of further research could be to estimate this set efficiently for the purposes of testing recall in the algorithm.

In future work we would like to test on a more general dataset, which would allow us to leverage Mechanical Turk as a verification method. Mechanical Turk verifications need not be limited to the suggested tags, but could also be made against the hierarchy itself, to verify that our parent-child relationships are legitimate. A dataset which allowed for a greater quantity of tags associated with each post would also improve our automatic suggestion validation procedure.

REFERENCES

Ciro Cattuto, Dominik Benz, Andreas Hotho, and Gerd Stumme. 2008. Semantic grounding of tag relatedness in social bookmarking systems. In *The Semantic Web-ISWC 2008*. Springer, 615–631.

Chih-Chung Chang and Chih-Jen Lin. 2011. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology* 2 (2011), 27:1–27:27. Issue 3. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

Wei Lu, Stephen Robertson, and Andrew MacFarlane. 2006. Field-weighted XML retrieval based on BM25. In *Advances in XML Information Retrieval and Evaluation*. Springer, 161–171.

Rila Mandala, Tokunaga Takenobu, and Tanaka Hozumi. 1998. The use of WordNet in information retrieval. In *Use of WordNet in Natural Language Processing Systems: Proceedings of the Conference*. 31–37.

Stephen Robertson, Hugo Zaragoza, and Michael Taylor. 2004. Simple BM25 extension to multiple weighted fields. In *Proceedings of the thirteenth ACM international conference on Information and knowledge management*. ACM, 42–49.

Rion Snow, Daniel Jurafsky, and Andrew Y Ng. 2004. Learning syntactic patterns for automatic hypernym discovery. *Advances in Neural Information Processing Systems 17* (2004).

Robert E Tarjan. 1977. Finding optimum branchings. *Networks* 7, 1 (1977), 25–35.

Ying Zhao, George Karypis, and Usama Fayyad. 2005. Hierarchical clustering algorithms for document datasets. *Data Mining and Knowledge Discovery* 10, 2 (2005), 141–168.