Clustering Tags in Enterprise and Web Folksonomies

Edwin Simpson
Digital Media Systems Lab
HP Laboratories Bristol
HPL-2007-190
December 7, 2007*

topic recognition, social media tools, navigation, relevance of blogs, tagging, folksonomy, clustering, similarity, betweenness, graph, cooccurrence

Recently there has been massive growth in the use of tags as a simple, flexible way to categorize resources. Tags are often used collaboratively to help share information using website; such as del.icio.us. However, the number of tags used in such a service is extremely large, so the unstructured nature of tags limits their value when navigating these websites, and prevents users from fully exploiting tags added by others. Clustering similar tags can improve this by adding structure. In this paper we discuss techniques for deriving tag similarity and explain two tag clustering algorithms. We applied the algorithms to two datasets containing tags provided by users with common interests. The first dataset is from a tagging service used by a small group of colleagues and the second is a public, web-based service. The paper examines the effectiveness of both clustering algorithms and their robustness to the different types of data, giving suggestions of possible ways to improve the algorithms.

Internal Accession Date Only
Submitted to Second International Conference on Weblogs and Social Media, March 31st, 2008.
Approved for External Publication

Clustering Tags in Enterprise and Web Folksonomies

Edwin Simpson
edwin.simpson@hp.com
HP Labs, Filton Road, Stoke Gifford, Bristol, UK

Abstract

Recently there has been massive growth in the use of tags as a simple, flexible way to categorize resources. Tags are often used collaboratively to help share information using websites such as del.icio.us [1]. However, the number of tags used in such a service is extremely large, so the unstructured nature of tags limits their value when navigating these websites, and prevents users from fully exploiting tags added by others. Clustering similar tags can improve this by adding structure. In this paper we discuss techniques for deriving tag similarity and explain two tag clustering algorithms. We applied the algorithms to two datasets containing tags provided by users with common interests. The first dataset is from a tagging service used by a small group of colleagues and the second is a public, web-based service. The paper examines the effectiveness of both clustering algorithms and their robustness to the different types of data, giving suggestions of possible ways to improve the algorithms.

Keywords

topic recognition, social media tools, navigation, relevance of blogs, tagging, folksonomy, clustering, similarity, betweenness, graph, cooccurrence

Introduction and Motivation

Tags are simple, ad-hoc labels that are assigned by users to describe or annotate any kind of resource. They are commonly used to organize weblogs, bookmarks, pictures and research papers [2]. As users can assign any tag to an item, tags are extremely easy to add [12] and capture a user's personal view of resources they are interested in. This important advantage of tagging systems (also known as FOLKSONOMIES) also causes problems when tags become numerous, or when we wish to utilize the tags provided by other users. These factors may be limiting the adoption of tagging for other domains such as corporate intranets. As well as vocabulary differences (e.g. use of different synonyms) and use of tags with personal meaning only, the number of tags used by a single user can become overwhelming when presented in a flat list. However, research suggests that there are many underlying, emergent structures in folksonomies [10, 5] that can help group similar tags from multiple users. Tags relating to a common topic are often used together, providing a simple estimate of tag similarity which can be used to find groups of similar tags relating to a single topic.

In this paper we explain different tag similarity measures and apply two clustering algorithms to tag data from two different types of folksonomy. Of these clustering algorithms, the second is a *betweenness centrality* divisive clustering algorithm often used in Social Network Analysis [11]. Here we test its applicability to clustering folksonomies. Both datasets were obtained from bookmarking services used to record, share and tag URLs, where users share a common interest. The aim was to demonstrate clustering tags relating to a common interest rather than to a set of unrelated and very distinct topics. We also consider the differences between clustering data from a public, web environment and data from a group of colleagues.

The contributions of this paper are the application of the two algorithms to the two datasets. We show that both algorithms produce effective clusterings for the first dataset. However, both algorithms produce some overly large clusters for the second dataset due to the prevelance of more densely inter-related tags. We show that pre-filtering of unpopular tags from the second dataset improved clustering performance for the first algorithm but not for the betweenness centrality algorithm.

Tag Co-occurrence Graphs

To find similarities between tags we require a way of relating tags and measuring the strength of their relationships. Tags which are similar may be synonyms, related concepts and common topics. A convenient representation of tag relationships is a graph, with tags represented by nodes and edges drawn between those with strong relationships. Several methods are outlined below for determining the strength of tag relationships.

Lexical Similarity

[5] suggest using an external thesaurus or semantic lexicon such as Wordnet to obtain the relationship data between terms used as tags. This suffers when new concepts and words are used, or when spell-checking on misspelled tags fails. The tag relationships are also less likely to reflect the way that tags are used by a particular group of users, as the meaning and association between tags is

likely to vary between users interested in different topics. For example, the tag "cluster" has two specific meanings in Computer Science, which are both different to general English.

Document-term Similarity

The similarity between tags is calculated from the textual similarity of documents they annotate. A term-frequency vector is created for each tag, and cosine similarity is used to compare pairs of tags [6].

Vector-space Similarity using Tagged Items as Features

A vector can be created containing the frequencies with which a tag was used with each item in the system. Cosine similarity can then be used over these vectors. This measure produces intuitive results because more general tags have a low similarity to more specific, related tags, and more specific tags have a high similarity to other, closely related specific tags. Methods that use the contents of text documents to derive tag relationships also rely on the accessibility and interpretability of the content. This is not always possible as tagged items may be photographs or videos, which are difficult to cluster, or require too much processing power.

Co-occurrence Similarity

The number of co-occurrences between two tags (uses of the tags on the same item). A high co-occurrence value suggests a strong similarity between tags.

In our experiments we clustered tags using co-occurrences extracted from the folksonomy, but there are other sources of data that enable the finding of related concepts from a given tag. [3] demonstrate the use of co-occurrence similarity by applying graph filtering combined with a spectral clustering algorithm to cluster tags. Tag co-occurrence was also used by [9] to filter tags in a user profile to produce a set of interest clusters. In our paper we have examined how to estimate the level of graph filtering automatically, using an iterative divisive clustering algorithm. We have also examined another divisive clustering algorithm, using betweenness centrality to determine cluster boundaries. This method was used by [11] with social networks to cluster actors into communities of practice.

Extremely popular tags are used many times with many other tags, so their co-occurrence values would suggest high similarity to many other tags. To remove this bias, we trial proposals from [3] to the Jaccard index, which normalizes the raw co-occurrence value relative to the popularity of two tags. The Jaccard index is $|AandB|/|AorB|$ where A is the set of documents tagged with tag a, and B the set of documents tagged with b. We will call this normalized measure NORMALIZED CO-OCCURRENCE, or NCO. Other alternative normalizations, such as cosine similarity have also been suggested, but were not compared here.

Further research would be needed for a full comparison of similarity measures for tag clustering. We chose to use Jaccard indices with a co-occurrence matrix as this method does not require the use of document contents and is an intuitive measure of similarity.

Related Work

An alternative way of partitioning information is to cluster the contents of tagged documents. Tags may then be placed in multiple clusters depending on the documents they are associated with. For example, in [7], clustering is performed using the contents of documents, and tags are used only to measure the quality of the clusters produced. However, this does not expose latent tag structure to improve the way that tags are used for navigation. Using this model, tags may fall into multiple clusters, but documents appear strictly in one cluster. Our aim is to allow users to reach documents that are associated with multiple topics or concepts by selecting the tags representing the concepts they are currently interested in.

[5] have proposed methods for extracting ontologies from folksonomies, exploiting various statistical relationships between tags.

Dataset

Two datasets were collected from two different types of sources: the first is an extremely popular social bookmarking service, del.icio.us [1], and the second is an internal bookmarking service, Labbies, used by a group of around twenty researchers at HPLabs. Both services allow users to record the URL of a resource, annotate it with some tags and make this record available to other users. However, the vastly different communities and differing features using each service affect the way tags are applied. A key primary difference is that the creator of each bookmark is recorded by del.icio.us, so the service can be used to store personal bookmarks, whereas Labbies bookmarks are created anonymously and always remain public. If users wish to amend tags in Labbies, they may add tags or modify existing tags for a bookmark, regardless of who created it. For these reasons it is harder to estimate how many users are now actively using the service.

Del.icio.us has approximately 3 million users [13] so contains extremely large amounts of data. We obtained a subset of data by selecting users who share a common interest, and collecting all bookmarks added within a 16 week period. The common interest was defined as users who have added the tag "dspace" during the 16 week period. Using a common interest allows us to test whether clustering techniques are effective on tags that are likely to be strongly related. Rather than taking a sample from across all topics covered by del.icio.us, we take a sample that is likely to contain many inter-related topics, which one may also wish to navigate through with the help of cluster analysis. As the boundaries between these topics are expected to be less clear, this dataset may also be more difficult to partition than a dataset containing many unrelated topics.

Simple cleanup operations were performed on the tags: tags were converted to lower case, white-spaces and hyphens were replaced with underscores. Further cleanup may be desirable, such as removing personal tags (e.g. "todo"), merging synonyms or detecting homonyms using a dictionary. However, it is also of interest to see whether tag co-occurrence graphs and clustering may help detect these types of tags without the need to use dictionaries

Dataset	Labbies	del.icio.us
Number of Users	15	136
Number of Bookmarks	1935	95155
Number of Tags	2092	8012
Bookmark Creation Period	01/12/06 - 21/11/07	01/08/07 - 21/11/07
Number of Co-occurrences between Tags	9526	61453

Table 1: *Dataset Statistics*

or pre-specified tag filters, which require greater maintenance.

The statistics of the selected datasets are given in table 1.

Clustering

The clusters that appear in the co-occurrence graph could be used in an interface to indicate related tags or structure a flat list of tags into clusters, possibly as a level in a hierarchy. Membership of the same cluster could also be used as to find related tags that are not directly connected in a graph. This may be used for recommending interesting items, tags or people to users.

To automatically find the set of clusters that best matches the tag co-occurrence graph, we use a divisive clustering algorithm. The graph is divided into separate sub-graphs by removing edges likely to fall between clusters. The sub-graphs then become clusters. We can select edges to remove simply by selecting those with the lowest normalized co-occurrence, as we expect weaker co-occurrences between tags with weaker relationships that should be placed in different clusters. We will refer to this algortihm as TAG-CO-OCCURRENCE DIVISIVE CLUSTERING. A similar algorithm that recursively divided clusters by removing edges with low (un-normalized) co-occurrence was trialled in [3].

An alternative method of selecting edges to remove is to consider the density of edges across the graph. Inside a cluster, the number of edges connecting nodes is high, whereas between clusters the edges are sparse. If we consider how nodes in one cluster are connected to a neighboring cluster, we observe the shortest path between each node in one cluster and each node in the other. The path must include one of the few inter-cluster edges. Within the cluster, however, different edges will form part of different shortest paths, so an internal edge will not be part of a large number of shortest paths. Therefore, we can select edges to remove that lie on the greatest number of shortest paths between nodes. This value is known as BETWEENNESS CENTRALITY and is described in depth in [4]. The advantage of this method is that it removes edges based on the entire graph structure, rather than local edge strengths only.

The BETWEENNESS CENTRALITY CLUSTERING algorithm can be improved by re-calculating the betweenness centrality of edges after some have been removed. When we calculate betweenness centrality, we only consider the shortest path between nodes, so an edge that lies on a slightly longer path could have a low betweenness centrality, even though it lies between clusters. This problem is resolved by re-calculating betweenness centrality once the neighboring, slightly shorter path has been removed. Now, the path that was previously slightly longer becomes the shortest, and gains a high betweenness centrality score.

After removing some edges from the graph we then determine whether we have obtained the best (most intuitive) clustering for the chosen algorithm. Modularity [11] is the fraction of edges connecting nodes in the same cluster minus the fraction of edges that would connect nodes in the same cluster if the clusters were marked randomly in the graph. Calculating the modularity of the original graph (with all edges replaced) for the current clustering gives a numeric evaluation of the quality of the clustering. If the graph naturally divides into clusters, a high value of modularity will be obtained. We can keep removing edges until none remain, taking the set of clusters with the highest modularity as the final, most intuitive clustering.

Unfortunately, the need for re-evaluation and the poor scalability of the betweenness centrality calculation may become impractical for large datasets. Betweenness centrality requires that for each edge in the graph, we must calculate the shortest path between each node and each other node, which requires $O(en^2)$ time, where e is the number of edges and n is the number of nodes. [4] describes a more efficient method for calculating betweenness centrality that requires $O(ne + n^2 logn)$. In addition to using the more efficient algorithm we use an approximation of betweenness centrality: for each edge, we take a subset of nodes which are local to the edge (i.e. shortest path is less than a threshold) and so limit the expected size of n. However, if the graph becomes more densely connected, as is expected for large datasets, the subset of nodes used will increase in size, so the algorithm's performance will still be worse than linear. The del.icio.us dataset we are using here is far more densely connected than the Labbies dataset as it has approximately four times as many tags, but 6.5 times as many edges. We expect the processing task to be far more significant for betweenness centrality clustering with this dataset.

Graph Visualisation

We have created visualizations of the tag co-occurrence graphs and clusterings from our datasets, by representing tags as nodes and co-occurrence relationships as edges. The graphs are also been filtered to removed weak relationships, where NCO value is below the stated threshold, and to remove tags with a usage frequency below a given threshold, giving a clearer view of the graph. The nodes and edges in the visualizations were positioned using a Spring Layout [8]. Nodes in the graph are iteratively repositioned closer together if they have a strong connecting edge, and are pushed apart if unconnected. The aim is to give a pleasing visualization that reflects node relationships using distance. This allows us to see the cluster structure within the folksonomy, and whether it supports the clusterings derived. The size of the nodes reflects their popularity. In diagrams showing clustering

Dataset	Algorithm	Initial no. Clusters	Final no. Clusters	Largest Cluster (no. tags)	Modularity	NCO Threshold
Labbies	Tag-Co-occurrence Divisive	70	674	191	0.75	0.182
	Betweenness Centrality	70	275	126	0.41	-
Del.icio.us	Tag-Co-occurrence Divisive	346	3314	716	0.79	0.142
	Betweenness Centrality	346	2447	4863	0.35	-

Table 2: *Statistics for Clusterings Produced*

we have colored and labeled the graph to indicate different clusters. As there is still a large amount of data after applying this filtering, we select a sub-graph to visualize by specifying a root node and including all members of its cluster and all tags within a given CLUSTER DISTANCE. CLUSTER DISTANCE is the number of edges between a given tag and the closest member of the root node's cluster.

Clustering Results

Both clustering algorithms were applied across the two datasets. Statistics for the clusterings produced are given in table 2. The initial set of clusters is the number of distinct clusters in the graph before any edges have been removed by the clustering algorithm. The initial clusterings consist of one large cluster surrounded by many small peripheral clusters. The small clusters, are tags that have been seldom used and for which data is sparse. The large initial cluster covers all of the main topics and so is divided into smaller clusters by the clustering algorithm.

Labbies Results

Figure 1 clearly shows clusters around topics such as "policies", "digital_library" and "semantic_web". The tags "reactive_rules", "policies" and "eca" (meaning "event, condition, action") all relate to the same specific topic and are similar in meaning. It would therefore be helpful to users browsing one tag to be aware that alternative tags may also identify interesting items. This clustering could be used to identify these highly-similar tags.

Popular nodes connected to many clusters have also been separated into their own clusters, such as "mit". These appear to be terms that are connected with several tightly-knit clusters, and are more general, higher-level or ambiguous terms. These tags are also harder for humans to place into specific clusters: ambiguous terms need special treatment to determine which meaning is intended; higher-level terms may only be useful at an early stage of browsing for homing in on the lower-level topic clusters. Higher-level and ambiguous tags could be identified from these single-tag clusters when the tag has been used many times, as higher level tags are typically more popular than more specific tags.

Clusters produced using the betweenness-divisive method (figure 2) appear similar at first, but there are significant differences. Table 2 shows that far fewer clusters were produced. Nodes such as "dspace" and "simile" have not been placed separate clusters. Table 3 shows that fewer tags were placed into very small clusters or separated into one-tag clusters, and this algorithm has actually produced a more evenly-sized clusters. Since there are still many tags in very large clusters, we might improve results if we modified the algorithm to focus on splitting these clusters to produce a better clustering. The highest modularity achieved by removing edges using the current algorithm is lower than for tag co-occurrence-divisive clustering. However, the largest cluster produced is smaller with betweenness-divisive clustering (126 tags versus 191), so the clustering suffers less from domination by a single cluster.

It is not clear from the graphs and tables produced which set of clusters would be of most use to users navigating through labbies. To assess this would require further user studies with interfaces or tools that make use of the clusters.

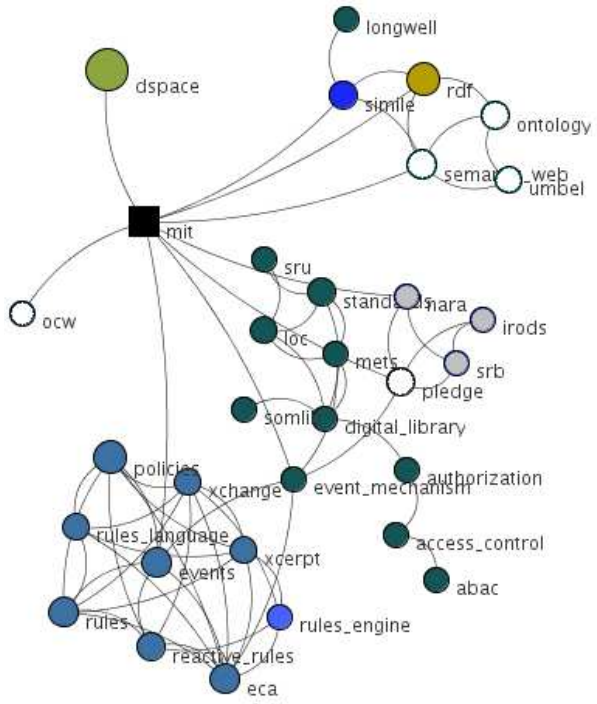

Fig. 1: *Tag Co-occurrence Divisive Clustering on Labbies Dataset. root="mit"; cluster distance < 2; tag use threshold=3; NCO threshold=0.06*

Dataset	Clustering Algorithm	Largest Cluster (no. tags)	No. singleton clusters	No.Tags in Clusters where size<=4	No. Clusters where size<=4	No.Tags in Clusters where size>4 and size <= 40	No. Clusters where size>4 and size <= 40	No.Tags in Clusters where size>40	No. Clusters where size>40
Labbies	Tag co-occurrence Divisive	191	338	1007	596	657	74	426	3
	Betweenness Centrality	126	133	346	211	704	47	1041	16
Del. icio. us	Tag co-occurrence Divisive	716	2074	4593	3079	2185	244	1293	8
	Betweenness Centrality	4863	1678	3414	2386	369	60	4863	1

Table 3: *Cluster Sizes for Labbies and Delicious Datasets*

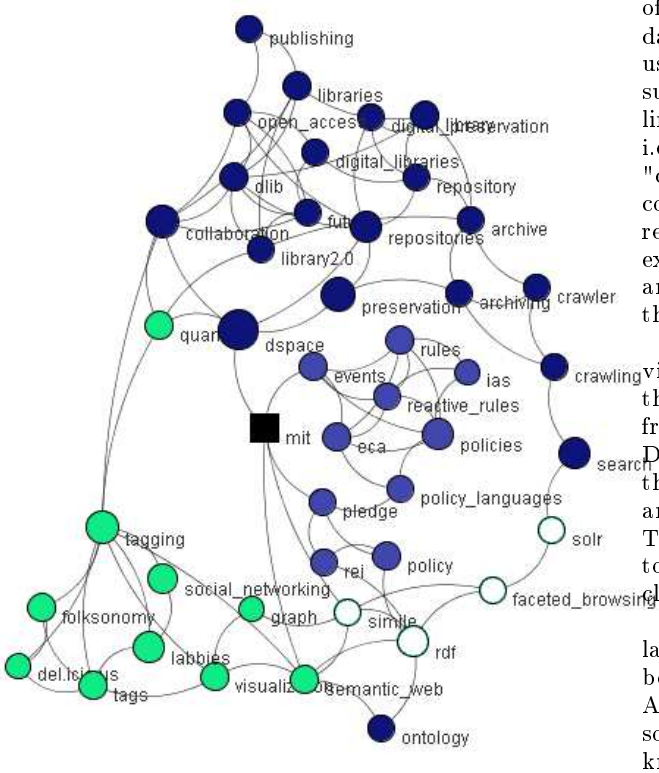

Fig. 2: *Betweenness Centrality Clustering on Labbies dataset; root="dspace"; cluster distance < 3; tag use threshold=12; NCO threshold=0.06*

Del.icio.us Results - Tag Co-occurrence Divisive Clustering

Applying the tag-co-occurrence divisive algorithm to del.icio.us data also produces reasonable clusters, but the handful of very large clusters are larger in proportion with the dataset and so are now of a size that may be difficult for users to use directly. These large clusters have not been sub-divided due to the high density of inter-connecting links. This may be due to the sampling of delicious data, i.e. that we chose only a portion of users related to "dspace", and so have selected an extremely dense graph consisting of tags that are semantically quite close (i.e. representing DSpace concepts). However, one may have expected a number of smaller clusters to emerge as the amount of data and number of different users is larger than for Labbies.

To visualize the clustering more easily, we altered the visualization so it does not include all the nodes from the large cluster, but only shows those nodes separated from the root, “dspace” by no more than six edges (Tag Distance < 7). This is shown in figure 3. Inspection of the clusters shows that tags such as “Spain”, “university” and “relationships” are placed in the same large cluster. These concepts are not semantically close to Dspace or to one another, contrary to the suggestion that the large cluster may contain semantically close tags.

In this graph, you can see the wider connection to related clusters. Note the "fedora" tag and its centrality between clusters relating to completely different topics. Also, note that now the visualization has been filtered, some of the blue tags appear to form a small, tightly knit cluster arond “bookmarks”. This is suggests that there may actually be intuitive clusters within the large, amorphous blue cluster that appear when weaker edges and less common nodes are discarded. The algorithm did not select to remove edges from inside this cluster, but if more were removed, the large blue cluster could have been split into more usable clusters. This suggests that the algorithm should be modified to prefer removing more edges from large clusters.

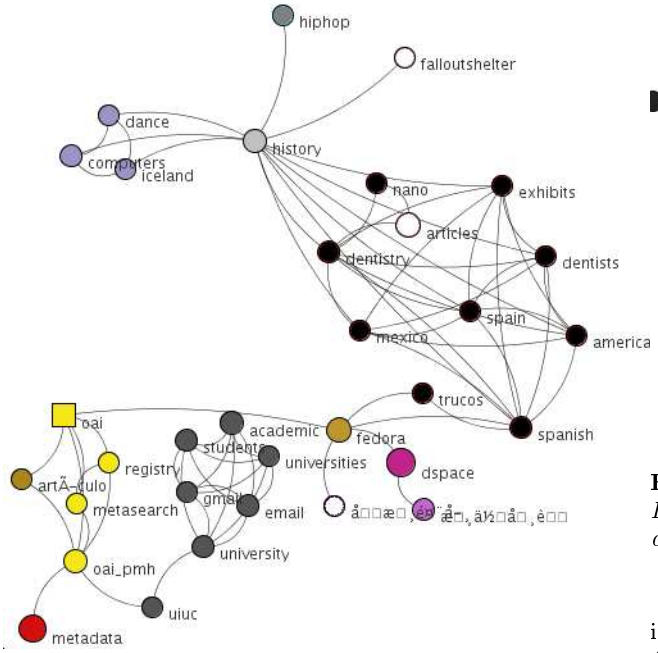

Fig. 3: *Tag-co-occurrence Divisive Clustering, del.icio.us dataset; root="oai"; tag distance < 5; tag use threshold=120; NCO threshold=0.09*

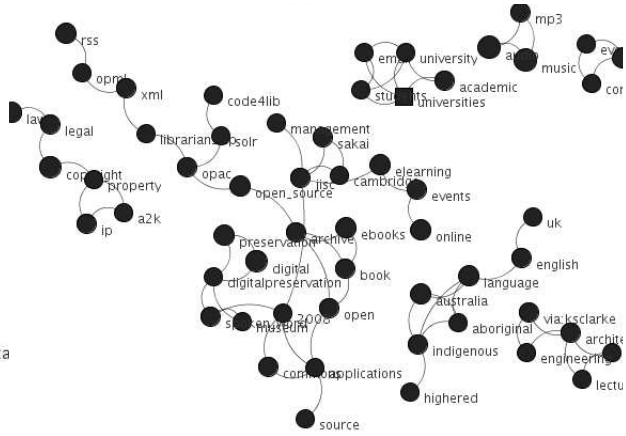

Fig. 4: *Tag-co-occurrence Divisive Clustering on Del.icio.us dataset; single cluster; tag use threshold=400; co-occurrence threshold=0.142*

Table 3 shows there is both a large number of small clusters (less than 5 tags) and that many tags also fall into clusters with a size greater than 40. Many of the smaller clusters may have been produced by over-dividing larger clusters, which should occur less often when using the betweenness centrality algorithm, but they may also relate to peripheral topics for which little co-occurrence data was available.

Some tags in the larger clusters appear to be unrelated to others in the same cluster. This may also be due to inadequate information: there is no data to link them to other topics outside this cluster. If a tag was only used once or twice, it may have always been used in conjunction with a particular tag in the large cluster. The connecting edge would then have a high NCO value, meaning the rare tag might stay attached to the large cluster. Such co-occurrence data could be considered noise data, as it does not represent a meaningful semantic relationship.

To obtain better, smaller clusters for the tags in the large clusters, we could remove noise by excluding the rarely-used tags before performing clustering (TAG PRE-FILTERING). We can test this idea using the visualization techniques. In figure 4 we have filtered the cluster by removing less common tags, and set the minimum NCO value to 0.142, the same as the threshold determined by the clustering algorithm to produce the highest modularity. Here we see that some small clusters appear, showing some useful structure that was not extracted by the clustering algorithm.

Since filtering tags with a low popularity appears to improve the visualization, we applied pre-filtering to the graph before re-running the clustering algorithm. The resulting clustering is shown in figure 5, which shows an improved set of clusters. Statistics are shown in table 4. The resulting modularity is similar to when the graph was not pre-filtered, but we now have a clustering with no extremely large clusters and a larger proportion of more useful medium-sized clusters (4-40 tags). The clustering produced contains more balanced sizes of clusters when compared to the Labbies results, although the number of tags and resulting number of clusters is similar.

We then increased the tag pre-filtering threshold to 250, approaching the level used in figure 4, intending to improve the clustering by further splitting up the largest clusters, as we did in the visualization. However, the number of nodes remaining from the original 8012 is only 290 (see table 4), which means we have discarded most tags and may no longer have useful clusters. Also, the modularity is only 0.231 and there is a high number of clusters relative to tags. This level of pre-filtering may be useful for dividing the large, dense clusters but produces a poor global clustering. It seems preferable to perform the clustering on a moderately filtered dataset to remove noise, but not to completely discard data provided by the less commonly used tags.

The NCO threshold found by the algorithm was different for Labbies, del.icio.us and pre-filtered del.icio.us datasets, so the similarity between tags in the same cluster may also vary between datasets. As a result the clusters produced in some datasets may be more general than in others. One should be aware of this if mixing clusters from different datasets (e.g. web and enterprise).

A disadvantage of tag pre-filtering is that many tags were removed, even with a lower threshold (around 80% tags were removed), and clustering rare tags may be very valuable. If a user is using a rare tags, clustering would help find alternative tags so they can find new relevant items. However, excluding the rare tags in this clustering phase improves the clustering. It may be sensible to re-add the removed rare tags after the clustering is complete, by assigning them to the most similar cluster, so they do not disrupt the divisive clustering phase and are not excluded from the final set of clusters.

Dataset	Algorithm	Tags in Dataset	Initial no. Clusters	Final no. Clusters	Modularity	NCO Threshold
Pre-filtered Del. icio.us	Tag-Co-occurrence Divisive	1594	47	607	0.70	0.264
(tag use > 25)	Betweenness Centrality	1594	47	375	0.18	-
Pre-filtered Del. icio.us	Tag-Co-occurrence Divisive	290	4	120	0.23	0.154
(tag use > 250)	Betweenness Centrality	290	4	4	0	-

Dataset	Clustering Algorithm	Largest Cluster (no. tags)	No. singleton clusters	No.Tags in Clusters where size<=4	No. Clusters where size<=4	No.Tags in Clusters where size>4 and size <= 40	No.Clusters where size>4 and size <= 40	No.Tags in Clusters where size>40	No. Clusters where size>40
Pre-filtered Del.icio.us	Tag Co-occurrence Divisive	87	350	819	532	631	73	143	2
(tag use > 25)	Betweenness Centrality	1115	269	503	365	52	9	1115	1

Table 4: *Statistics for pre-filtered clusterings*

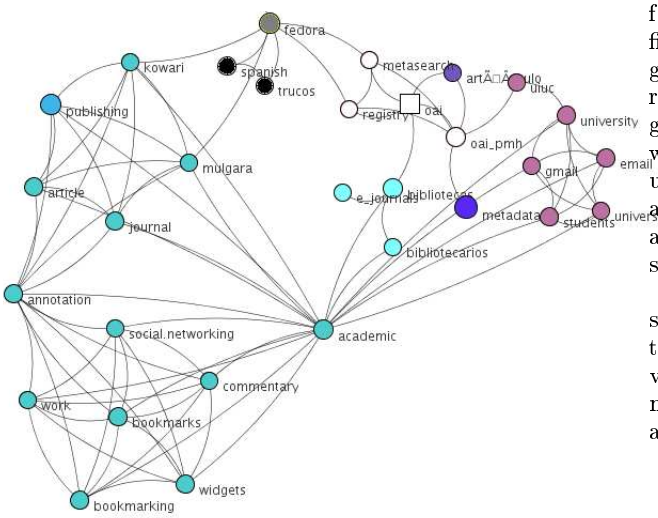

Fig. 5: *Tag Co-occurrence Divisive Clustering on Del.icio.us dataset; pre-filtered to remove tags with usage < 25; root="oai"; tag distance < 4; tag use threshold=25; NCO threshold=0.12*

Del.icio.us Results - Betweenness Centrality Clustering

As with tag co-occurrence clustering, betweenness centrality clustering also produced one large cluster when the dataset was not pre-filtered, although this time the cluster contained over half of the nodes so was not a useful partitioning of the data. We then applied tag pre-filtering, but a large dominant cluster remained. The algorithm was unable to obtain a modularity score comparable with the tag co-occurrence divisive algorithm, suggesting that selecting edges to remove using betweenness was not effective with this dataset. Despite this, many useful clusters were identified correctly, some of which are shown in figure 6. The tags centered around news are part of the large, dominant cluster but appear as a separate cluster when the tag is filtered by tag distance.

Our experiments show it was more difficult to produce small clusters from del.icio.us data, as the common interest between a wide range of different users produced a very dense tag graph. Each tag was related to a greater number of different topics due to the range of vocabulary and user interests.

Future Work

The main problem with all the clustering techniques shown here was the appearance of disproportionately large clusters. This would be problematic with the del.icio.us dataset because the large number of tags meant that the largest clusters contained several hundred tags and could be difficult to use for navigation. We could improve the algorithm by removing edges only from over-sized clusters rather than from the whole graph, or by using the bisecting spectral algorithm used by [3], that recursively removes edges from specific clusters until the modularity peaks.

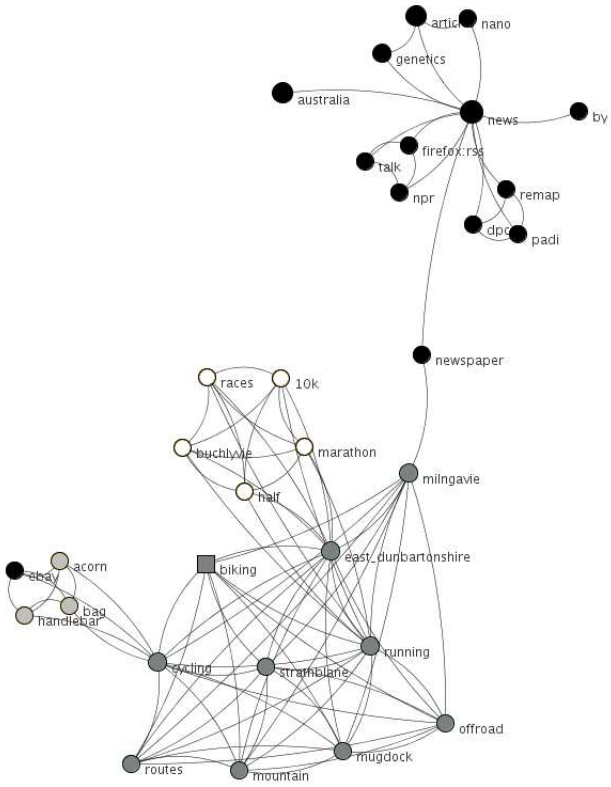

Fig. 6: *Betweenness Centrality Clustering on Del.icio.us dataset, pre-filtered to remove tags with usage < 25; root="biking"; tag distance < 5; tag use threshold=25; NCO threshold=0.075*

The modularity calculation was limited as a measure of the quality of the clustering, as it did not take into account the relative sizes of the clusters and preferred not breaking up the large, dense clusters of tags, such as that which formed around the "dspace" topic in the del.icio.us dataset.

The del.icio.us dataset could be increased in size by taking more data from del.icio.us, which may improve the clustering of tags which were seldom used in this dataset and may have been poorly clustered. We could also improve the available dataset by using co-occurrence of tag terms in document contents. This would particularly help where tags such as "2do" have been associated strongly with a topic-specific tag. "2do" is unlikely to occur in the document contents, so its relationship would be marked as weak despite a high tag co-occurrence. Dealing with ambiguous tags - another likely source of confusion - also requires further work.

An extension of this work would also look at hierarchies of clusterings. The large number of clusters produced for each dataset here could be grouped into super clusters more useful for high-level browsing, and large clusters could be further sub-divided. Rather than settling on the best clustering by choosing that with the highest modularity, a number of clusterings would be chosen and presented in a hierarchical format. An alternative to producing super-clusters would be to extract the tags that represent high-level concepts and are strongly associated with a number of clusters relating to more specific topics.

Conclusion

In this paper we showed that the tag-co-occurrence divisive algorithm effectively clustered tags used by a small group of enterprise users (the Labbies dataset). We also produced a useful clustering of tags from Internet users with a common interest, "dspace", and showed the effects of removing unpopular tags before clustering. We tested the betweenness-divisive algorithm, which was also very effective for Labbies, but performed poorly on the densely inter-related tags in the del.icio.us dataset, and was not improved by pre-filtering unpopular tags. Along with a high cost of calculation, this means it is more suitable for clustering tags in a smaller, less dense folksonomy. We uncovered differences between the web-based and enterprise-based datasets, which make cluster structure more difficult to identify on the web and suggested some ideas to improve this situation in future work.

References

[1] http://del.icio.us.

[2] http://www.citeulike.org/.

[3] Grigory Begelman, Philipp Keller, and Frank Smadja. Automated tag clustering: Improving search and exploration in the tag space. 2006.

[4] U. Brandes. A faster algorithm for betweenness centrality, 2001.

[5] Katharina Siorpaes Celine Van Damme, Martin Hepp. Folksontology: An integrated approach for turning folksonomies into ontologies. *European Semantic Web Conference*, 2007.

[6] Nancy Montanez Christopher H. Brooks. Improved annotation of the blogopshere via autotagging and hierarchical clustering, 2006.

[7] Paolo Avesani Conor Hayes. Using tags and clustering to identify topic-relevant blogs. *ICWSM*, 2007.

[8] P. A. Eades. A heuristic for graph drawing. In *Congressus Numerantium*, volume 42, pages 149–160, 1984.

[9] Steve Cayzer Elke Michlmayr and Paul Shabajee. Adaptive user profiles for information access. *World Wide Web Conference*, 2007.

[10] Paul Heymann and Hector Garcia-Molina. Collaborative creation of communal hierarchical taxonomies in social tagging systems. *InfoLab Technical Report*, 2006.

[11] M. E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical Review E*, 69:026113, 2004.

[12] Rashmi Sinha. A cognitive analysis of tagging, 2005.

[13] Search Engine Round Table. Keynote q and a: Joshua schachter of del.icio.us and garrett camp of stumbleupon, october 2007.