Improving the Exploration of Tag Spaces Using Automated Tag Clustering

Joni Radelaar, Aart-Jan Boor, Damir Vandic,
Jan-Willem van Dam, Frederik Hogenboom, and Flavius Frasincar

Erasmus University Rotterdam
PO Box 1738, NL-3000
Rotterdam, The Netherlands
joni@radelaar.nl, {aartjan.boor,jwvdam}@gmail.com,
{vandic,fhogenboom,frasincar}@ese.eur.nl

Abstract. Due to the increasing popularity of tagging, it is important to overcome challenges resulting from the free nature of tagging, such as the use of synonyms, homonyms, syntactic variations, etc. The Semantic Tag Clustering Search (STCS) framework deals with these challenges by detecting syntactic variations of tags and by clustering semantically related tags. We evaluate our framework using Flickr data from 2009 and compare the STCS framework to two previously introduced tag clustering techniques. We conclude that our framework performs significantly better in terms of cluster precision compared to one method and has a better average precision compared to the other method.

Keywords: Tagging, syntactic clustering, semantic clustering, tag disambiguation.

1 Introduction

On today's World Wide Web, it is becoming increasingly popular to use tags for the purpose of describing resources. Tagging allows users to annotate a resource, such as a video, photo, or Web page, with a keyword or tag of their own choice. Because there are no restrictions on the tags that can be used, tags provide a flexible way of describing resources. However, because of the unstructured nature of tagging, there are some problems associated with retrieving resources using tag-based search engines. These problems are often caused by different tags having the same or closely related meaning. This can be the result of the use synonyms, but it could also be caused by syntactic variations. Examples of syntactic variations are misspellings or the use of the plural or singular form of a specific word. Users may also use different levels of specificity while describing a resource, which is identified as the basic level variation problem by Golder and Huberman [9]. For example, one user might tag a picture of a cat as "animal" (not very specific), while another user would use "persian" (very specific). The usage of homonyms, i.e., words with multiple unrelated meanings, is another problem associated with tagging.

S. Auer, O. Diaz, and G.A. Papadopoulos (Eds.): ICWE 2011, LNCS 6757, pp. 274–288, 2011.

The problems described above can lead to undesirable results when searching for resources using tags. For example if a user is looking for a picture using “cat” as a keyword, he or she would most likely also be interested in pictures which are tagged with “cats” (syntactic variation), “persian” (more specific, semantically related term), “kitty” (synonym), and “kittie” (misspelling of kitty).

One way to deal with these problems is to create clusters of syntactically and semantically related tags. Creating syntactic clusters involves the grouping of tags that are syntactic variations of each other into separate groups or clusters. Search algorithms can then use these clusters to improve the quality of a search query. For example, when a user enters a tag as a search query, the search algorithm could also add tags to the query that are in the same cluster as the tag that was entered. Creating semantically related clusters involves grouping tags that are semantically related, e.g., “sanfrancisco” and “goldengate”. Tags occurring in multiple semantic clusters can be used to identify tags with multiple meanings. If a tag occurs in multiple clusters it most likely also has multiple meanings, e.g., “turkey” can refer to both the country and the animal.

As a solution to the previous problem, we define the Semantic Tag Clustering Search (STCS) framework, which consists of two parts. The first part deals with syntactic variations, whereas the second part is concerned with deriving semantic clusters. We implement and evaluate the use of the Levenshtein similarity measure [13] and a combination of the Levenshtein similarity and the cosine similarity measure, as similarity measures for syntactically related tags. For identifying semantic clusters we implement and evaluate the semantic clustering algorithm proposed by Specia and Motta [21] and a clustering algorithm proposed by Lancichinetti et al. [11]. Additionally, we propose a modification to the Specia and Motta approach to improve the results. We perform a thorough evaluation of the used clustering methods that goes beyond previous evaluations in extent.

The contribution of this paper stems from several aspects. First, although several clustering techniques for clustering tags have already been proposed [3,21,24], the evaluation of these techniques is done using relatively small data sets with a small number of resources. In this paper, we evaluate different syntactic and semantic clustering techniques using a larger data set than previously reported in literature. In this way we aim to analyze the performance of our algorithms more accurately on high volume data, gathered from Flickr [20]. Second, the proposed algorithm for syntactic clustering addresses the issue of identifying syntactic variations among short tags. Third, for the semantic clustering we identify the issues with currently available tag clustering algorithms and propose solutions for them. We have published previous work on STCS in [23]. Compared to this early work, in this paper we provide more details on the used algorithms, use a significantly larger data set for the experiments, and perform a more thorough evaluation.

The rest of this paper is organized as follows. Section 2 discusses related work. Subsequently, in Section 3 and 4 we give an overview of the design and implementation of our STCS framework. Section 5 elaborates on the evaluation of our experimental results and Section 6 concludes the paper.

2 Related Work

This section discusses related work on several key aspects of our methodology. Firstly, Subsection 2.1 presents tag clustering methods. Then, Subsection 2.2 elaborates on similarity measures, and finally, Subsection 2.3 introduces some related work on cluster evaluation.

2.1 Tag Clustering

Echarte et al. [7] discuss the problem of syntactic variations in folksonomies. They propose the utilization of pattern matching techniques to identify syntactic variations of tags. They evaluate the performance of the Levenshtein and Hamming distances using the 10,000 most popular tags and 1,577,198 annotations from CiteULike. Results show that the Levenshtein measure provides the best overall performance. However, both techniques do not perform well with tags shorter than 4 characters.

Specia and Motta [21] propose a method for building semantically related clusters of tags using a non-hierarchical clustering technique based on the co-occurrence of tags. They also explore the relationships between pairs of within-cluster tags. The authors perform a statistical analysis of the tag space in order to identify clusters of possibly related tags. Clustering is based on the cosine similarity among tags given by their co-occurrence vectors. Before creating the clusters, Specia and Motta merge morphologically similar tags using the normalized Levenshtein distance measure. The authors manually evaluated the results based on 49,087 distinct resources and 17,956 distinct tags from Flickr and found that the clustering approach results in meaningful groups of tags corresponding to concepts in ontologies.

Begelman et al. [3] propose to build a directed graph of tags with an edge between two vertices (tags) when there is a (strong) relation. The weight of the edge is based on the co-occurrence of the connected tags. In order to partition the set of tags into groups of semantically-related tags, their recursive algorithm uses spectral bisection to split the graph into two clusters. It then evaluates the split using the modularity function, which was introduced by Newman and Girvan [17]. The modularity function provides a measure of the quality of a particular division of a network. Begelman et al. applied their clustering algorithm to a data set containing 200,000 resources and 30,000 tags.

Yueng et al. [24] also use a graph-based clustering algorithm, where the modularity function is used to evaluate the quality of a division. However, unlike Begelman et al., the authors consider different network representations of tags and documents, e.g., networks based on users, co-occurrence of tags, and context of tags using cosine similarity, and discuss the effects of these various representations on the resulting clusters of semantically related tags. For their clustering experiments, a small data set is gathered from Delicious, containing 20 manually selected tags representing two or more concepts, complemented by randomly selecting 30 tags from the 100 most popular tags, with each tag having about 500 images. The authors find that networks based on tag context similarity capture the most concepts. With these networks the cosine similarity is used to perform a comparison of the context in which two tags are used, as reflected by the tag co-occurrence vectors of the tags.

Lancichinetti et al. [11] present a method that uncovers the hierarchical and overlapping community structure of complex networks. Their algorithm uses a newly defined fitness function that determines the quality of a cover. This function is used to discover the natural community of each node in a graph by optimizing the fitness function using local iterative searching. The authors evaluate their algorithm on artificial networks [2] that are known to have a built-in community structure. Their results show that their algorithm is successful in identifying these communities.

2.2 Similarity Measures

There are many similarity measures available for use in clustering algorithms. Cattuto et al. [4] analyze a variety of these using a Delicious data set containing the 10,000 most popular tags, by comparing the relations established through the use of different similarity measures to WordNet [8] synsets. Cosine similarity turns out to be the best similarity measure for detecting synonyms, while FolkRank and co-occurrence appear to be more useful for detecting various other semantic relations.

Markines et al. [15] evaluate the matching similarity, overlap similarity, Jaccard similarity, Dice coefficient, cosine similarity, and mutual information measures using a more systematic approach. They investigate the performance of these measures by generating several two-dimensional views on the tripartite folksonomy of BibSonomy [10] using aggregated data from 128,500 resources, 1,921 users, and 58,753 tags. Unlike BibSonomy, Flickr only allows one user to annotate a resource. Therefore, for a Flickr data set, only projection aggregation is useful. The result of projection aggregation can be considered as a matrix with binary elements $w_{rt} \in \{0, 1\}$, where rows correspond to resources and columns corresponds to tags. Given a resource and a tag, a 0 in this matrix element means that no user associated that resource with that tag, whereas a 1 means that at least one user has performed the indicated association. All similarity measures can then be derived directly from this information. The usefulness of the various measures as tag-tag similarity measures is evaluated using Kendall's τ correlations between the similarity vectors generated by the various measures and the reference similarity vector provided by a WordNet grounding measure. Mutual information proved to be the best similarity measure when using projection aggregation. When compared to each other the remaining similarity measures have the same performance. Unfortunately, mutual information is a computationally intensive measure, which makes its use unfeasible for large data sets.

2.3 Cluster Evaluation

Several measures exist to analyze clusters. Larsen and Aone [12] describe the precision measure. Average precision is defined as

$$\text{AvgPrec}(\Omega, C) = \frac{1}{|\Omega|} \sum_{w_k \in \Omega} \frac{\max\limits_{c_j \in C} |\omega_k \cap c_j|}{|w_k|} \,, \tag{1}$$

where $\Omega = \{\omega_1, \omega_2, ..., \omega_k\}$ is the set of tag clusters and $C = \{c_1, c_2, ..., c_j\}$ is the set of tag classes. We interpret ω_k and c_j as a set of tags, where ω_k is denoting a tag cluster and c_j a tag class.

Manning et al. [14] describe the purity measure to evaluate clusters. They define purity as

$$\text{Purity}(\Omega, C) = \frac{1}{N} \sum_{w_k \in \Omega} \max_{c_j \in C} |\omega_k \cap c_j| \,, \tag{2}$$

where Ω and C are the same as in the previous equation and N is the total number of tags.

Delling et al. [6] propose the density measure, which is a trade-off between intra-cluster density and inter-cluster sparsity to evaluate a specific clustering. Let us assume that G is an undirected and unweighted graph with n nodes and m edges. A partitioning of the nodes into several clusters c is called a clustering C of a graph. The edges between nodes of the same cluster are called *intra-cluster edges* and the edges between nodes of different clusters are called *inter-cluster edges*. The density of clustering C is then defined as:

$$\text{Density}(C) = \frac{1}{2}\,\text{Intra-cluster-density}(C) + \frac{1}{2}\,\text{Inter-cluster-sparsity}(C) \,, \tag{3}$$

where

$$\text{Intra-cluster-density}(C) = \frac{1}{|C|} \sum_{c \in C} \frac{\#\text{ intra-cluster edges c}}{\binom{|c|}{2}} \,, \tag{4}$$

and

$$\text{Inter-cluster-sparsity}(C) = 1 - \frac{\#\text{ inter-cluster edges}}{\binom{n}{2} - \sum_{c \in C} \binom{|c|}{2}} \,. \tag{5}$$

3 Framework Design

This section introduces the Semantic Tag Clustering Search framework (STCS) framework, which addresses the syntactic and semantic issues in tagging systems. The framework consists of two layers. In the first layer, syntactic variations (e.g., misspellings, morphological variations, etc.) of tags are eliminated by clustering the tags that are syntactic variations of each other and merging them into a single tag. In the second layer, the framework deals with the problem of identifying semantically related tags. This section continues with a problem definition in Subsection 3.1, a discussion of the similarity measures used in Subsection 3.2, and a more detailed elaboration of the framework in Subsection 3.3.

3.1 Problem Definition

We now give a formal problem definition, for which we follow the formulation given in [16]. The data set which is used as input for the framework is defined as a tuple $D = \{U, T, P, r\}$, where U, T, and P are the finite sets of users, tags, and pictures, respectively. The ternary relationship $r \subseteq U \times T \times P$ defines the initial annotations of the users.

Removing Syntactic Variations. In order to effectively find semantically related tags, we first remove syntactic variations of tags from the data set. Syntactic variations usually are misspellings of words but may also include translations of tags in other languages, or morphological variations. To remove these syntactic variations we create a set $T' \subset \mathcal{P}(T)$ in which each element of the set T' is a cluster containing all tags that are syntactic variations of each other. Each tag can only appear in one cluster. To determine the tag to be used as cluster label, we define m', which is the bijective function that indicates a label for each $x \in T'$, $m' : T' \rightarrow L$. For each $l \in L$ and some $x \in T'$, $l \in x$ holds, such that $m'(x) = l$, thus, l is one of the tags that labels the cluster x.

Finding Semantically Related Tags. In our framework, we aim to find semantically related tags by creating a set T'' containing semantic clusters of elements $l \in L$. This denotes that we disregard the syntactic variations in the semantic clusterings by only clustering tags that are labels of syntactic clusters. An example of a semantic cluster is {“nyc”, “newyork”, “manhattan”}. A tag should be able to be part of multiple clusters, each with a different meaning.

3.2 Similarity Measures

Based on the results of the discussed related work, we apply two similarity measures within the STCS framework. In order to determine tag similarity, we employ the Levenshtein distance measure and the cosine similarity. Related work showed that the Levenshtein measure performed better in detecting syntactic variations than the Hamming distance measure [7]. However for short tags the Levenshtein measure does not perform well. In order to cope with this problem we have combined the Levenshtein distance with the cosine similarity. The Levenshtein distance is a measure for the amount of typographic difference between two strings, also called edit distance. It is defined as the minimum number of operations needed to transform one string into the other. An operation can be an insertion, deletion, or substitution of a single character. We call this distance the *absolute* Levenshtein distance. We denote it by alv_{ij}, which is the absolute Levenshtein distance between tag i and j. Our framework needs to deal with different tag lengths so we used the *normalized* Levenshtein similarity, which is a measure that is relative to the tag length. The normalized Levenshtein similarity between tag i and j, denoted by lv_{ij}, is defined as follows:

$$lv_{ij} = 1 - \frac{alv_{ij}}{\max(length(t_i), length(t_j))} \ . \tag{6}$$

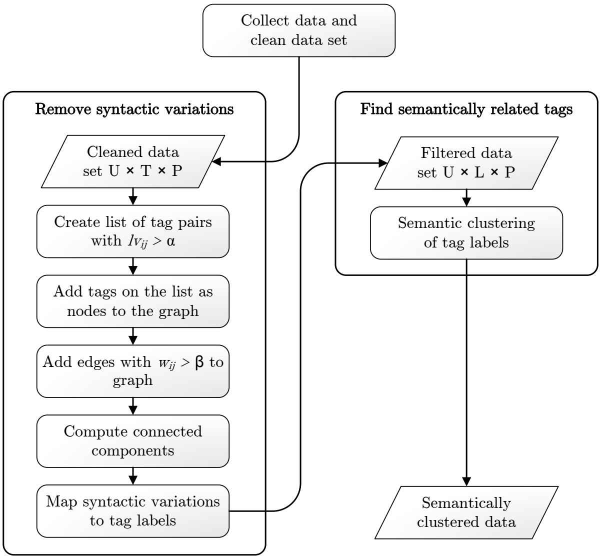
Collect data and clean data set
Remove syntactic variations
Cleaned data set U × T × P
Create list of tag pairs with $lv_{ij} > \alpha$
Add tags on the list as nodes to the graph
Add edges with $w_{ij} > \beta$ to graph
Compute connected components
Map syntactic variations to tag labels
Find semantically related tags
Filtered data set U × L × P
Semantic clustering of tag labels
Semantically clustered data

Fig. 1. Overview of the STCS framework

In order to measure the semantic relatedness between tags and the syntactic similarity of short tags, we use the cosine similarity based on co-occurrence vectors. In essence, this measure describes the similarity of the context in which the tags appear. The context here is how often tags are used together with other tags (i.e., the co-occurrence).

3.3 STCS Framework

This section describes the two layers of the STCS framework in detail. An overview of the framework is presented in Fig. 1.

Removing Syntactic Variations In order to remove syntactic variations, we employ an adapted Levenshtein distance measure. The algorithm requires an initial list of tag pairs with a normalized Levenshtein distance above a certain threshold α as input. The α threshold represents the minimum normalized Levenshtein distance for which we consider two tags to be syntactic variations. The initial list is then used to create sets T and E as input for constructing an undirected graph. The set T contains each unique tag on the list. The set E is a set of weighted edges between the nodes in T, where the weight represent the similarities between tags. The weight w_{ij} of an edge in the tag graph is calculated as

$$w_{ij} = z_{ij} \times lv_{ij} + (1 - z_{ij}) \times \cos\left(\text{vector}\left(i\right), \text{vector}\left(j\right)\right) \ , \tag{7}$$

where lv_{ij} is the normalized Levenshtein similarity between tag i and j and

$$z_{ij} = \tfrac{\max(\text{length}(t_i),\text{length}(t_j))}{\max(\text{length}(t_k))} \in \left(0,1\right], \text{with } t_i, t_j, t_k \in T \ . \tag{8}$$

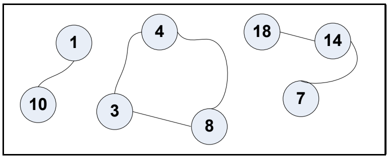

Fig. 2. An example of graph containing three clusters

Using the normalized Levenshtein distances for short tags may result in false positives, i.e., two tags being incorrectly identified as syntactic variations of each other. Therefore, the weight of the cosine similarity between two tags increases as the tags become shorter.

In order to build the tag graph, all the elements from set T are added as nodes to the graph. Subsequently, all the edges in set E for which the weight is above a certain threshold β are added to the graph, connecting the nodes from set T. The β threshold indicates the level of the combined weight measure (w_{ij}) for which we consider two tags to be syntactic variations of each other. Subsequently, the syntactic clusters can be determined by retrieving the connected components in the graph as sets of vertices. A connected component is defined as a maximal subgraph in which all pairs of vertices in the subgraph are reachable from one another. Each subgraph then contains all the nodes (tags) that form a cluster of syntactically related tags. An example of the resulting subgraphs is presented in Fig. 2 with each node containing the id of a unique tag. The resulting graph contains clusters of tags that are syntactic variations of each other. An example of a syntactic cluster is {venetie, venezia, venzia, veneza, venesia, venizia}, which is assigned the label 'venice'. We process this data by creating a new data set in which the tags in the clusters are aggregated and presented as a single tag, which we call the label for the cluster. The label of a cluster is the tag which is used most frequently in the data set. The resulting data set is used as input for the semantic clustering layer.

Clustering Semantically Related Tags. After removing the syntactic variations and misspellings from the data set, the new data set can be used to create semantic clusters. For this we use a partitional clustering algorithm, i.e., a clustering-by-committee-based algorithm [19] used by Specia and Motta [21], both with and without some modifications. The choice for this algorithm is motivated by the fact that it uses all tags instead of the cluster centroid to calculate the similarity between two clusters. This allows us to better capture the semantics associated with the tag space. Also, unlike many other clustering algorithms, it allows for multiple classification of tags, which enables us to deal with tag polysemy.

The algorithm starts with creating the initial clusters, where each tag is a separate cluster. Then, all the tags for which the average cosine similarity with respect to all the tags in the clusters is above a certain threshold (χ) are added to the cluster. This could result in many identical or nearly identical clusters. To avoid a high number of these very similar clusters, Specia and Motta use two smoothing heuristics. For every two clusters, the authors check whether one cluster contains the other, i.e., whether all the elements in the smaller cluster are also present in the larger cluster. If this is the case, the smaller cluster is removed. For each pair of clusters they also evaluate whether the clusters differ within a small margin by checking whether the number of different tags in the smaller cluster with respect to the larger cluster represents less than a percentage of the number of tags in the smaller cluster. If this is the case, the distinct tags in the smaller cluster are added to the larger cluster and the smaller cluster is removed.

A problem with the second heuristic is that the percentage used for merging two similar clusters is constant. This implies that the maximum allowed number of different elements increases with the size of the smaller cluster. Choosing a suitable threshold value is problematic, as we do not want the larger clusters to merge too easily and the smaller clusters too difficultly. The maximum number of different elements for two clusters to be merged, is given by $f(|c|) = \lfloor \epsilon \cdot |c| \rfloor$, where ϵ is the threshold and $|c|$ is the number of elements in the smaller cluster. So for $\epsilon = 0.20$ and $|c| = 30$, the maximum number of different elements is given by $f(30) = 6$. This means that cluster c will be merged into a larger cluster C, if $|D| \leq 6$, where $D = c - C$. Furthermore, as $f(4) = 0$, a cluster with a size below 4 is never merged.

Because of these limitations, we define two new heuristics that replace the original second heuristic. The first new heuristic considers the semantic relatedness of the difference between two clusters. We merge two clusters C and c, where $|C| \geq |c|$, when the average cosine of all elements in D with elements in the larger cluster is above a certain threshold δ. This average cosine is defined as

$$\text{AvgCos} = \sum_{d \in D} \frac{\text{Avg}_d}{|D|}, \text{ with } \text{Avg}_d = \sum_{x \in C} \frac{\cos(x, d)}{|C|} \, . \tag{9}$$

The second new heuristic considers the size of the difference between two clusters in combination with a dynamic threshold. We merge two clusters in case the normalized difference between the clusters is smaller than a dynamic threshold ϵ. The normalized difference η is defined as

$$\eta = \frac{|D|}{|c|} \, . \tag{10}$$

Threshold ϵ is defined as

$$\epsilon = \frac{\phi}{\sqrt{|c|}} \, , \tag{11}$$

and thus

$$f(|c|) = \lfloor \epsilon \cdot |c| \rfloor = \lfloor \phi \cdot \sqrt{|c|} \rfloor \, . \tag{12}$$

The distribution of the maximum allowed difference for which two clusters are merged can then be adjusted by changing ϕ. An example of a semantic cluster is {london, tatemodernart, towerbridge, milleniumwheel, buckinghampalace, thames}.

As a comparison we also use the algorithm proposed by Lancichinetti et al. [11]. We choose this algorithm because as Specia and Motta's method [21] it allows a tag to be part of multiple clusters. The algorithm uses a graph as an input and attempts to determine the natural community for each node in the graph. In this graph, tags are represented by nodes and weighted edges connect the nodes. The weight of an edge is the cosine similarity of the co-occurrence vectors of the two tags the edge connects. A community is a subgraph G identified by the maximization of the fitness of its nodes. If we consider each tag as a node in the graph, the community of a node forms a cluster of semantically related tags. The fitness of a subgraph G is defined as:

$$f_G = \frac{k_{in}^G}{(k_{in}^G + k_{out}^G)^\theta} \tag{13}$$

where k_{in}^G is the strength of the internal links, which in our case is given by two times the sum of the weights of all edges in G and k_{out}^G is the strength of the external links, which in our case is the sum of the weights of all edges linking nodes in G with nodes not belonging G. The parameter θ is used to adjust the size of the resulting communities. Large values of θ yield very small clusters, while small values result in large clusters. The fitness of node A with respect to graph G is defined as $f_G^A = f_{G+\{A\}} - f_{G-\{A\}}$, where $G + \{A\}$ / $G - \{A\}$ are the graphs obtained from G by adding/deleting node A.

The natural community of a node A is detected as follows. We start with a covered subgraph G including only node A. Each iteration then consists of the following steps:

1. Visit all neighboring nodes of G not included in G;
2. Add the neighbor with the largest fitness to G, yielding G';
3. Recalculate the fitness of each node in G';
4. Delete a node that has a negative fitness, yielding G'';
5. If a node is deleted in 4, repeat from 3, else repeat from 1 with G being G''.

This procedure stops when all neighboring nodes considered in step 1 have a negative fitness. However, it is too computationally intensive to perform this procedure for every node. Therefore, the authors describe the following heuristic. First pick a node A at random and detect the community of node A. Next pick a node B not yet assigned to any group and detect the community of this node. This process is repeated until each node is assigned to at least one group.

4 Framework Implementation

This section discusses the implementation of the STCS framework. The implementation of the framework is done in Java in combination with a MySQL database. For data collection and processing we used PHP scripts. This section continues with discussing data collection and processing in Subsection 4.1, and the implementation details for the cosine computation and clustering in Subsection 4.2.

4.1 Data Processing

For our experiments, we gather data from Flickr related to all the pictures uploaded in 2009, together with their associated tags and users. To speed up the data collection process, we distribute this task over four separate machines. The initial data set contains 38,788,518 pictures, 196,344 users, and 1,017,168 tags. After data cleaning, there are 147,064,188 associations, 31,951,884 co-occurrence pairs, and 97,569 tags left.

The data cleaning process consists of several steps that aim to cope with noise encountered in the data due to the lack of restrictions imposed on users assigning tags to pictures. First of all, we remove tags with a tag length larger than 32 characters, to avoid tags that are entire sentences. The number 32 is based on an extensive manual tag analysis. Furthermore, we remove non-Latin characters (e.g., Arabic, Cyrillic, etc.) as well as numeric characters. Subsequently, we remove images from the same user that share identical tag strings. This filter is motivated by the fact that in our data set, we sometimes encounter hundreds of pictures uploaded by the same user with identical tags. These are sets of holiday pictures tagged with identical tags, often unrelated to the picture. To prevent these sets from influencing the co-occurrence measure, we only keep one image of each of these sets and remove the others. Finally, we remove tags which occur in less than 133 different pictures, as they are statistical outliers in our analysis, i.e., $AverageTagOccurence - 1.5 \times IQR \approx 133$.

4.2 Implementation Details

In order to be able to calculate the cosine similarity, one needs the co-occurrence vector for each tag. To obtain this, we construct a matrix with both a row and a column for each tag, with the cells containing the co-occurrence for that particular combination of tags. We aimed to employ the Colt library [5] to store this matrix in memory because of its small memory footprint. However, the size of our matrix is too large to be handled by Colt, and thus we implement our own high performance matrix library which uses a Colt vector to store each column of the co-occurrence matrix. Using the resulting matrix, we calculate the cosine similarity for each unique combination of two tags. Because the co-occurrence matrix is very large and the number of cosine computations increases very fast with the matrix size, we use a distributed system. For this purpose, we utilize Amazon EC2 [1], a service which provides cloud computing resources. We implement the algorithms in a distributed fashion and run them in parallel on multiple high memory instances, each having 17,1 GB of RAM to fit the entire matrix in memory. In our experiments, the total amount of instances running in parallel fluctuate between 3 and 52 instances. In total, these experiments took up 8 computing hours on 2,914,700 cosine similarity calculations for the syntactic clustering, which completed in 2.5 hours of actual time. For the semantic clustering, we use 64 computing hours to perform 50,000,000 cosine similarity calculations within 11 hours of actual time. Each instance loads the full matrix in memory and connects to a central job server which coordinates each instance to perform a distinct portion of the calculations.

In our framework, we implement the syntactic clustering algorithm by means of the Java Universal Network/Graph Framework (JUNG) [18] graph library. This library provides a good method for retrieving the set of connected components in a graph, which are clusters of tags in our case. The semantic clustering algorithms is also written in Java. In order to reduce the time required for the semantic clustering, we only consider the 10,000 most popular tags.

5 Evaluation

This section presents results of experiments conducted on our cleaned Flickr data set. Subsection 5.1 discusses the results related to the first layer of our STCS framework, i.e., removing syntactic variations. Subsection 5.2 elaborates on experimental results related to finding semantically related tags.

5.1 Removing Syntactic Variations

We chose $\alpha = 0.5$ as a threshold for the normalized Levenshtein similarity to identify potential syntactic variations. We chose this constant using a sample of 100 tag pairs that were known to be syntactic variations. We found that the normalized Levenshtein similarity between these tag pairs was never smaller than 0.5. The goal of this threshold was to reduce the number of potential syntactic variations for which the calculation of the cosine similarity was required. A value of 0.5 effectively reduced this number without losing syntactic variations.

For the β threshold we chose a value of 0.7. We have chosen this value because it resulted in the best performance on random samples of 100 clusters. For this threshold we tried all values between 0 and 1 with a step of 0.05. For the evaluation of each value we drew a separate random sample of 100 clusters (our training set). After filtering the cleaned data set discussed in Section 4.1 on syntactic variations, there are 147,064,188 associations, 28,603,077 co-occurrence pairs, and 91,916 tags left.

We evaluate the performance of the clustering technique using the combined measure and the Levenshtein distance using precision and purity. We do not use the density, because in our case it proved to be too computationally intensive. For each clustering technique, we draw a random sample of 100 syntactic clusters and evaluate these manually. For this evaluation we used majority voting with a group of three people. Each person in the group chooses the correct tags in each cluster and majority voting is then used to determine the final number of correct tags that is used in the precision and purity calculations. The average precision of the algorithm used in our framework on the random sample set is 0.89. The average precision of the clustering algorithm using the normalized Levenshtein distance is 0.70. By performing a one-tailed unpaired two sample t-test with a significance level of 0.01, we conclude that the combined measure does perform significantly better than the Levenshtein distance alone in terms of precision per cluster. The purity of the STCS framework is 0.88, while the purity of the technique using only the Levenshtein distance to identify syntactic variations is 0.67. Because we only use one data set and each data set has a single average precision and purity, it was not possible to perform t-tests for the average precision and purity.

5.2 Finding Semantically Related Tags

We choose $\chi = 0.8$, $\delta = 0.7$, and $\phi = 0.9$ as thresholds for our framework. For the clustering algorithm that uses a constant percentage to identify similar clusters, we set $\epsilon = 0.3$. For the θ threshold used in the method proposed by Lancichinetti et al. [11], we choose $\theta = 1.5$. We used these values because they proved to give the best performance on random samples of 50 clusters (our training set). It is a non-trivial task to evaluate the results of the semantic clustering quantitatively due to the lack of external grounding. Semantic lexicons such as WordNet [8] only contain a small portion of the tags in our data set. We considered all values ranging from 0.1 to 0.9 with an increment of 0.1 for the χ, δ, ϕ and ϵ thresholds and values ranging from 0.5 to 3 with an increment of 0.25 for the θ threshold. We chose this range for θ because smaller values resulted in extremely large clusters and larger values resulted in extremely small clusters. For each threshold we evaluated a separate random sample of 50 clusters for each value. All semantic clustering algorithms utilize the syntactic clustering algorithm using the combined measure to filter out syntactic variations.

For the chosen thresholds we used majority voting and a different random sample of 100 clusters (our test set) for each method to compute the average precision and purity. We perform these computations for the clustering method using the original heuristic for merging two similar clusters, the method using the two new heuristics, and the method introduced by Lancichinetti et al. [11]. For the evaluation we again use majority voting with a group of three people. The average precision when using the two new heuristics is 0.86. The average precision when using the constant percentage as a threshold for merging two clusters is 0.80, and the average precision when using the method introduced by Lancichinetti et al. is 0.81. By performing a one-tailed unpaired two sample t-test with a significance level of 0.05, we conclude that the two new heuristics significantly improve the precision per cluster when compared to the heuristic that uses only the constant percentage. However, with a significance level of 0.05, the method using the new heuristic does not perform significantly better than Lancichinetti's method (w.r.t. precision). The purity of the technique using the two new heuristics is 0.89, while the purity of the technique using the original heuristic for merging two similar clusters is 0.87. The purity of Lancichinetti's method is 0.77. We observe that the purity of the technique with the two new heuristics is the highest, but we cannot test for significance as in our measurements we have purity as a single number (based on only one data set).

6 Conclusion

In this paper, we proposed the STCS framework, which performs syntactic and semantic tag clustering. For the syntactic clustering, we make use of a combined measure of the Levenshtein distance and the cosine similarity. We compared the results of clustering on the combined measure to clustering on the Levenshtein distance. Our conclusion is that the combined measure performed significantly better in terms on precision. The clustering method as proposed in our framework was able to effectively filter out syntactic variations from the data set.

For semantic clustering, the framework uses an adaptation of the approach proposed by Specia and Motta [21]. We are capable of identifying numerous and useful clusters. Optimizing the parameters is difficult, as it is a non-trivial task to evaluate the results of the semantic clustering quantitatively due to the lack of external grounding, since existing semantic lexicons only contain a small portion of the tags in our data set. Nevertheless, our experiments show that the proposed method significantly outperforms the original method by Specia and Motta and outperforms on average the method of Lancichinetti in terms of precision. Finally, we have shown that our results are valid on a significantly larger data set than was used before in the existing body of literature.

As future work, it could be interesting to investigate how the cluster information can be used to enhance search results and especially how users experience and value this improvement. This would provide crucial insight into which clustering method in the end provides the best results in terms of user experience. We would also like to experiment with the use of the Wikipedia redirects as a tool to help identify syntactic variations of tags. Additionally, the services provided by the TAGora repository [22] might prove useful for identifying syntactic variations.

References

1. Amazon Web Services LLC: Amazon Elastic Compute Cloud, Amazon EC2 (2010), http://aws.amazon.com/ec2
2. Arenas, A., Diaz-Guilera, A., Perez-Vicente, C.J.: Synchronization Reveals Topological Scales in Complex Networks. Phys. Rev. Lett. 96(11), 1–4 (2006)
3. Begelman, G., Keller, P., Smadja, F.: Automated Tag Clustering: Improving Search and Exploration in the Tag Space. In: Carr, L.A., Roure, D.C.D., Iyengar, A., Goble, C.A., Dahlin, M. (eds.) 15th World Wide Web Conference (WWW 2006), pp. 22–26. ACM Press, New York (2006)
4. Cattuto, C., Benz, D., Hotho, A., Stumme, G.: Semantic Grounding of Tag Relatedness in Social Bookmarking Systems. In: Sheth, A.P., Staab, S., Dean, M., Paolucci, M., Maynard, D., Finin, T., Thirunarayan, K. (eds.) ISWC 2008. LNCS, vol. 5318, pp. 615–631. Springer, Heidelberg (2008)
5. CERN - European Organization for Nuclear Research: Colt Libraries for High Performance Scientific and Technical Computing in Java (2010), http://acs.lbl.gov/~hoschek/colt/
6. Delling, D., Gaertler, M., Görke, R., Nikoloski, Z., Wagner, D.: How to Evaluate Clustering Techniques. Tech. rep., Faculty of Informatics, Universitat Karlsruhe (2006), http://digbib.ubka.uni-karlsruhe.de/volltexte/1000007104
7. Echarte, F., Astrain, J.J., Córdoba, A., Villadangos, J.: Pattern Matching Techniques to Identify Syntactic Variations of Tags in Folksonomies. In: Lytras, M.D., Carroll, J.M., Damiani, E., Tennyson, R.D. (eds.) WSKS 2008. LNCS (LNAI), vol. 5288, pp. 557–564. Springer, Heidelberg (2008)

8. Fellbaum, C.: WordNet: An Electronic Lexical Database. MIT Press, Cambridge (1998)
9. Golder, S., Huberman, B.: The Structure of Collaborative Tagging Systems. Tech. rep., Information Dynamics Lab, HP Labs (2005), `http://www.citebase.org/abstract?id=oai:arXiv.org:cs/0508082`
10. Jäschke, R., Hotho, A., Schmitz, C., Stumme, G.: Analysis of the Publication Sharing Behaviour in BibSonomy. In: Priss, U., Polovina, S., Hill, R. (eds.) ICCS 2007. LNCS (LNAI), vol. 4604, pp. 283–295. Springer, Heidelberg (2007)
11. Lancichinetti, A., Fortunato, S., Kertesz, J.: Detecting the Overlapping and Hierarchical Community Structure in Complex Networks. New Journal of Physics 11(3), 1–19 (2009)
12. Larsen, B., Aone, C.: Fast and Effective Text Mining using Linear-Time Document Clustering. In: 5th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD 1999), pp. 16–22. ACM, New York (1999)
13. Levenshtein, V.I.: Binary Codes Capable of Correction Deletions, Insertions, and Reversals. Soviet Physics Doklady 10(8), 707–710 (1966)
14. Manning, C., Raghavan, P., Schütze, H.: Introduction to Information Retrieval. Cambridge University Press, Cambridge (2008)
15. Markines, B., Cattuto, C., Menczer, F., Benz, D., Hotho, A., Stumme, G.: Evaluating Similarity Measures for Emergent Semantics of Social Tagging. In: 18th World Wide Web Conference (WWW 2009), pp. 641–650. ACM, New York (2009)
16. Mika, P.: Ontologies Are Us: A unified model of social networks and semantics. In: Gil, Y., Motta, E., Benjamins, V.R., Musen, M.A. (eds.) ISWC 2005. LNCS, vol. 3729, pp. 522–536. Springer, Heidelberg (2005)
17. Newman, M.E.J., Girvan, M.: Finding and Evaluating Community Structure in Networks. Physical Review E 69(2), 1–15 (2004)
18. O'Madadhain, J., Fisher, D., Nelson, T., White, S., Boey, Y.B.: Java Universal Network Graph (JUNG) Framework (2010), `http://jung.sourceforge.net`
19. Pantel, P.: Clustering by Committee. Ph.D. thesis, University of Alberta (2003)
20. Schachter, J.: Delicious - Social Bookmarking (2010), `http://www.delicious.com/`
21. Specia, L., Motta, E.: Integrating Folksonomies with the Semantic Web. In: Franconi, E., Kifer, M., May, W. (eds.) ESWC 2007. LNCS, vol. 4519, pp. 624–639. Springer, Heidelberg (2007)
22. TAGora: TAGora Sense Repository (2010), `http://tagora.ecs.soton.ac.uk/tsr/index.html`
23. van Dam, J.W., Vandic, D., Hogenboom, F., Frasincar, F.: Searching and Browsing Tag Spaces Using the Semantic Tag Clustering Search Framework. In: Fourth IEEE International Conference on Semantic Computing (ICSC 2010), pp. 436–439. IEEE Computer Society, Los Alamitos (2010)
24. Yeung, C., Gibbins, N., Shadbolt, N.: Contextualising Tags in Collaborative Tagging Systems. In: 20th ACM Conference on Hypertext and Hypermedia (HT 2009), pp. 251–260. ACM, New York (2009)