A Semantic-Based Approach for Searching and Browsing Tag Spaces

Damir Vandic, Jan-Willem van Dam, Flavius Frasincar

Erasmus University Rotterdam
PO Box 1738, NL-3000 DR
Rotterdam, the Netherlands

Abstract

In this paper we propose the Semantic Tag Clustering Search (STCS) framework for enhancing the user experience in interacting with tagging systems. This framework consists of three parts. The first part deals with syntactic variations by finding clusters of tags that are syntactic variations of each other and assigning labels to them. The second part of the framework addresses the problem of the lack of semantics in tagging systems by recognizing contexts and constructing semantic clusters for tags. The last, and final part of the STCS framework, utilizes the clusters obtained from the first two parts to improve the search and exploration of tag spaces. For removing syntactic variations, we use the normalized Levenshtein distance and the cosine similarity measure based on tag co-occurrences. For creating semantic clusters, we employ two non-hierarchical and two hierarchical clustering techniques. To evaluate the value of the semantic clusters, we develop a Web application called XploreFlickr.com for searching and browsing through Flickr resources.

Keywords: Tag spaces, semantic clustering, syntactic variations, Flickr

1. Introduction

Nowadays, there are a lot of Web services where users can employ tags to label content on the Web. The reason for the popularity of collective tagging lies in its effectiveness. Recent research shows that collective tagging introduces several benefits for organizational knowledge creation and shared document repositories [32]. Furthermore, it has been shown that users mainly participate in collective tagging with the purpose to share information with the community [3, 21]. These are very encouraging signs for the growth of tagging systems and their application in various domains.

Flickr and Delicious are two well-known applications that make use of tags. In this paper we focus on the Flickr service. Users which are registered on the Flickr Web site can upload photographs and assign tags to them. As with most tagging systems, the user has no restrictions on the tags that can be used. Though tags are a flexible way of categorizing data, there are some limitations with the use of tags for the purpose of search and exploration of tagging systems. Because users are free to choose any tag, they can, for example, make typographical errors or use syntactic variations. This results in having different tags with the same meaning. An example of a typographical mistake is 'selfportait', which should be written as 'self portrait'. Searching for 'self portrait' gives 1 265 127 more results than searching for 'selfportait'. Plurals and singulars, like 'self portraits', and 'self portrait', are examples which are considered to be syntactic variations. Typographical errors and syntactic variations of tags are important aspects to consider when designing a search engine for tagging systems. Google, for example, automatically detects spelling mistakes in the query entered by the user. Often you get 'Did you mean...' from Google where the engine tries to suggest the correct query.

Furthermore, at the current moment users can use synonyms for the same concept. These synonyms yield different results when users search, explore, or retrieve information from a tagging system. For instance, one user could have used the tag 'city' to annotate a picture, whereas another user could have used 'town'. When users search for 'city', the pictures who are only tagged with 'town' will not be retrieved by the search engine. This is why it is important to identify semantic variations when considering the performance of search engines.

Users also describe the content of pictures in different ways. For a picture which shows the interior of a house, most users would use the tag 'interior', while others would use a tag like 'inside' or 'furniture'. It is obvious that these tags are semantically related. When someone searches for 'furniture', they are probably also interested in pictures that are annotated with 'interior'. Another example of semantically related tags is 'Web 2.0', 'Ajax', and 'XML'. Additionally, users can use tags that contain homonymous words, like 'apple'. When a user searches for 'apple', the search engine returns pictures related to the brand 'Apple' as well as pictures of apples. The search engine cannot distinguish between the multiple meanings the word 'apple' has.

Email addresses: vandic@ese.eur.nl (Damir Vandic), jwvdam@gmail.com (Jan-Willem van Dam), frasincar@ese.eur.nl (Flavius Frasincar)

Another example of a homonym is 'rock', it can have the meaning of the music style 'rock', but it can also be a large stone. We address this problem by performing context detection. This enables users to choose between the different meanings of tags when they are searching or browsing a tag space.

In order to improve the performance of tagging search engines, we need to deal with the previously described symptoms (typographical mistakes, syntactic variations, synonyms, homonyms, and related tags). The currently used search engines do not cope with these symptoms. In general, there are no structures, hierarchies, classifications, or clusters available in tagging systems. A reason for this situation might be the enormous amount of data. Better search engines for such tagging systems, with higher precision and recall, could be valuable for many users, organizations, and companies. For example, marketing companies often need pictures in their daily activities and these companies would certainly benefit from more structured tagging systems. In this paper we improve the search and exploration of tag spaces by coping with syntactic variations, typographical mistakes, synonyms, homonyms, and related tags.

The research goal of this paper is to gain insight into the possibilities of improving search and exploration in tag spaces, especially for marketing companies. Related work reveals that clustering techniques can be used to improve the search and exploration of tag spaces [5]. Thus, the main research question addressed in this paper is

> *How can one utilize clustering techniques to improve the search and exploration of tag spaces?*

In order to answer the research question we designed and implemented an appropriate framework. Our solution is called the Semantic Tag Clustering Search (STCS) framework. The framework consists of three parts, a part where syntactic variations are identified, a part where semantic clusters are derived, and a part where one can search in tag spaces by using search methods which utilize these clusters. A preliminary (short) presentation of the framework is available in [30].

In our framework we consider two types of semantic clustering algorithms, namely the non-hierarchical and the hierarchical semantic clustering algorithms. The non-hierarchical clusters contain related tags, but there is no hierarchy inside (or between) these clusters. Hierarchical clusters on the contrary, do have these hierarchies, either among tags or clusters of tags. For the non-hierarchical clusters we select the algorithm proposed in [28]. We choose to implement this algorithm because it allows tags to appear in multiple clusters. This enables us to easily detect different contexts by analyzing the different clusters of a tag. The algorithm proposed in [27] is selected for the hierarchical clusters. We chose for this approach as it is proven to have a higher precision than the approaches presented in [25] and [7]. An alternative could have been the algorithm in [12], which is meant specifically for collaborative tagging systems, like Delicious. The difference between a tagging system like Flickr and Delicious, is that resources (Web sites) on Delicious can be tagged by multiple users. This is not the case with the resources (images) on Flickr, which is the focus of this paper.

The innovation of this paper stems from several aspects. The syntactic variation clustering algorithm deals with the issues that are left open in other papers, [27] e.g., and [28], such as the bad performance of syntactic variation detection on short tags. We also propose two adapted algorithms for clustering tags that address the issues left open in [27] and [28]. So, in total we have two types of clustering techniques (hierarchical and non-hierarchical) and two instances of each type, the original and the adapted ones. Thus, we implemented four semantic clustering algorithms in total. Further, we propose two cluster-based search methods, one for the non-hierarchical and one for the hierarchical clusters. We also implemented a search engine that is called the 'Dummy' search engine. This search engine operates without the knowledge of the syntactic or semantic clusters, it is used for benchmark purposes. In our evaluation, we compare the cluster-based search engines and the Dummy search engine with each other. Such an approach gives insight into the possibilities of improving search and exploration in tag spaces.

Based on the proposed clustering algorithms, we have built a Web application called XploreFlickr.com, which is accessible online [29]. The Web application makes it possible to compare the results obtained from different clustering and search techniques directly on a subset of the Flickr database. With these results we investigate if clustering techniques improve the search and exploration of tag spaces.

2. Related Work

For clustering related tags several measures based on co-occurrence data are used in literature. In [28] the cosine similarity is used. In this paper the authors also experiment with different metrics to calculate the similarity between pairs of vectors of co-occurrence data, including Euclidean and Manhattan distance, but achieved the best results with the cosine similarity measure. Metrics computing absolute distance, like the Euclidean and Manhattan distances, showed to be inappropriate, since they are more sensitive to significant variations in a few elements than little variations in a large number of elements. In the case of Flickr, one has to deal with a data sets with little variations in a large number of elements. We therefore choose to use the cosine similarity.

2.1. Syntactic Variations

Syntactic variations between tags is a well-known symptom in tagging systems. Several authors have tried to deal with these variations. In [9] the authors analyze the performance of the Levenshtein distance [16] and the Hamming distance [10]. The authors showed that the Levenshtein and Hamming distances provide similar results for some syntactic variation types, for example typographic errors and simple plurals/singulars. With identifying variations based on the insertion/deletion of characters, Levenshtein gets significantly better results than Hamming. However, both techniques do not perform as well as desired when identifying variations based in the transposition of adjacent characters (library/lirbary) or some kind of singulars/plurals (library/libraries). Moreover, both methods improve their

results when candidate tags with less than four characters are ignored. This indicates that the two methods cannot effectively deal with short tags. The STCS framework addresses this issue by introducing a semantic component in the syntactic clustering algorithm. In [28] the authors also use the Levenshtein similarity metric to group morphologically similar tags. They use a high threshold to determine 'similar' words ('cat' and 'cats') as well as misspellings (such as 'theory' and 'teory'). Within each group of similar tags, one tag is selected to be the representative of the group, and the occurrences of tags in that group are replaced by their representative.

2.2. Semantic Symptoms

In previous approaches, the semantic symptoms are dealt with by either using a clustering technique which results in non-hierarchical clusters of tags, or a hierarchical graph of either tags or clusters of tags. In [28] the authors present a complete framework where they address the syntactic variations in a tagging system, create clusters of semantically related tags, and within each cluster, identify the relationship between each tag pair. The semantic clustering algorithm of [28] distinguishes itself from other approaches, because tags can occur in multiple clusters.

For the clusters of related tags, in [28] the authors use the cosine similarity measure on the co-occurrence data. Given the highly similar pairs of tags, their algorithm considers each pair, for example, 'audio' and 'mp3', as seeds constituting an initial cluster, and then tries to enlarge this cluster by looking for tags that are similar to both initial tags. This procedure is recursively repeated for all tags, i.e., each new 'candidate' tag for a cluster must be similar to the whole (possibly enlarged) set of tags in that cluster. The algorithm generates a set of clusters, including a number of identical clusters, resulting from distinct seeds that are in fact similar to each other. It also generates highly similar clusters, differing in only a few tags, which are in many cases a consequence of the threshold to filter out unrelated pairs of tags. Two smoothing heuristics are used to avoid having a high number of these similar clusters. In order to determine the relationship between tags in a cluster, the authors use Swoogle [8] and WordNet [19] to find ontologies where both tags occur.

In [5] the authors create semantic clusters of tags by using co-occurrence data. For every tag in the data set, they find the tags which co-occur the most with it. Next, the authors use a cut-off value which is determined by the first and second derivative of the co-occurrence count, where the co-occurrence count is on the y-axis and the tag ids are ordered descending on the count for the x-axis. The tags above this cut-off value are placed in a graph with the co-occurrence counts as the weights of the edges. To split the clusters further, the authors use the spectral bisection algorithm [24]. Then they use the modularity function [20] to determine whether or not to reject or accept the partitioning. The algorithm then proceeds recursively on each accepted partition. The authors conclude that clustering techniques can and should be used in combination with tagging. They also argue that these techniques can improve the search and exploration in tag spaces in general.

In [7], [25], and [27] a subsumption-based model is used to derive a hierarchy of semantically related tags. The title of [27] suggests that their algorithms result in an ontology induced from Flickr tags, but that is not the case. Their final result is a hierarchical representation of tags. These hierarchies give no information about domains, ranges, or the nature of the concept relationships, thus, you do not have a real 'ontology'. The results of [7] and [25] are also concept hierarchies.

In [27] an adjusted subsumption model of [25] is used, having as input co-occurrence statistics. The authors use this subsumption model for building hierarchical trees. The subsumption model from [27] differs from [25] as [27] uses additional conditions for statistical thresholds, like tag count restrictions. In [27] the models of [7] and [25] are also implemented and tested on a snapshot of the Flickr database as of July 2005. This database consists of about 25 million images, which yield 65 million annotations. The resulting trees are evaluated manually for all three models. The algorithm proposed in [27] outperformed the other two algorithms on the number of relevant relations found, and on their correctness.

Another example of a hierarchical taxonomy is proposed in [12]. Their algorithm builds a hierarchy of tags from annotation data. It is an extensible greedy algorithm that makes use of graph centrality. The authors determined several features which impact the effectiveness of their algorithm. A prerequisite, as they propose, is that the data contains natural hierarchical relations. They argue that this seems to be a general feature of tagging data. An empirical study in [14] showed that a large proportion of tags in Delicious participate in hierarchical relationships.

In [26] the authors discuss how association rule mining is used to analyze and structure folksonomies. The association rules used in [26] can be seen as subsumption relations, so that the rule mining can be used to learn a taxonomic structure. If many resources tagged with 'tag X' are also tagged with 'tag Y', this indicates, for example, that 'tag Y' can be considered a super topic of 'tag X'. The authors conclude that their method can be applied for different purposes, such as recommending tags, users, or resources, populating the super tag relation of a folksonomy, and community detection.

2.3. Searching Tag Spaces

There is little literature where the focus is primarily on the improvement of search and exploration in tag spaces by using clustering algorithms. The previously discussed papers about semantic clustering state that the main goal is the improvement of search and exploration in tag spaces, but in fact, none of them investigates this aspect. The scope of the papers does not go beyond discussing the derived syntactic and/or semantic clusters. In [1] and [2] seven different ranking algorithms for querying in tag spaces are discussed and evaluated, including FolkRank [13] and SocialPageRank [4]. These algorithms can be applicable for users, tags, and resources (or a combination of these). The authors mainly discuss adjusted algorithms for GroupMe.org.

In this paper we focus on the improvement of search and exploration by using clustering algorithms. Therefore, we only consider FolkRank [13], as SocialPageRank [4] is not suited for topic-related ranking. We have decided not to include FolkRank in the evaluation of the STCS framework. FolkRank is developed

with the idea that a resource can be annotated by multiple users. In Flickr this is not possible, because only one user can upload a specific picture and annotate that picture. In other systems, like Delicious, it is possible to have multiple users linking to a specific resource (a Web page). The FolkRank algorithm can be used on the Flickr service with the necessary adjustments. More specifically, the weight for an annotation a where tag t appears on picture p will always be one. In the FolkRank algorithm this weight is defined as total number of users which has annotated picture p with tag t. The FolkRank algorithm is based on the well-known PageRank algorithm [22]. If we were to use FolkRank on our Flickr data set, we would loose the important measure of 'user weight' (as all weights are 1 for Flickr), which is why we have not included FolkRank in our comparisons.

3. Framework Design

To answer the research question we propose the Semantic Tag Clustering Search (STCS) framework. This framework consists of three parts. The first part deals with syntactic issues by clustering tags that are syntactic variations of each other and assigning labels to them. The second part of the framework addresses the problem of the lack of semantics by identifying semantically related tags in tag bases. The last part of the STCS framework utilizes the clusters obtained from the first two parts to improve search and exploration in tag spaces.

In this section we first give the formal definition of the problem that is investigated in this paper. After that, we discuss the STCS framework in detail. We end the section with a short conclusion on the proposed design and methodology.

3.1. Problem Definition

We now give a formal problem definition, for which we follow the formulation given in [18]. The input data set is defined as a tuple $D = \{U, T, P, r\}$, where U, T, and P are the finite sets of users, tag IDs, and pictures, respectively, and r is the ternary relationship $r \subseteq U \times T \times P$, defining the initial annotations of the users. We can split the problem definition into three parts: removing syntactic variations, finding semantically related tags, and the improvement of searching in tag spaces with this newly derived knowledge.

3.1.1. Removing Syntactic Variations

The first goal is to remove syntactic variations from tags, a consequence of typographic mistakes or morphological variations. In order to detect these, we create a set $T' \subset \mathcal{P}(T)$, where $\mathcal{P}(T)$ represents the power set of T. Each element of T' represents a cluster of tags where each tag occurs in only one element (cluster), i.e., if $X, Y \in T'$, $X \neq Y$, and $a \in X$ and $b \in Y$, then this implies $a \neq b$. Then we denote by m' the bijective function that indicates a label for each $X \in T'$, $m' : T' \rightarrow L$. Furthermore, for each $l \in L$ there is a $X \in T'$ such that $m'(x) = l$ and $l \in X$, i.e., l is one of the tags in cluster X.

To clarify this mathematical definition we give an example. Consider the set of tags (tag IDs) $T = \{1, 2, 3, 4, 5\}$. A possible set T' could then be $T' = \{\{1, 3\}, \{2, 4\}, \{5\}\}$. The mappings could then be $\{1, 3\} \rightarrow \{1\}$, $\{2, 4\} \rightarrow \{4\}$, and $\{5\} \rightarrow \{5\}$. The set L is then $\{1, 4, 5\}$, as these tags are the labels.

3.1.2. Finding Semantically Related Tags

The second goal is to create semantic clusters of tags. We group semantically similar tags together, based on their meaning. This means that we create a set T'', with $T'' \subset \mathcal{P}(L)$, that is a cluster for elements from $l \in L$. This denotes that we cluster only the tags that are labels of a syntactic cluster. An example of a semantic cluster is {'new york', 'manhattan', 'hudson bridge', 'central park'}. A tag should be able to be present in multiple clusters. In this way, we can identify the different contexts of tags, i.e., tags are related with multiple clusters and therefore have multiple meanings.

3.1.3. Improving Search and Exploration in Tag Spaces

We define the 'improvement of search and exploration in tag spaces' by the three aspects:

- The clusters provide information which can be used to more precisely specify the query;
- The search engine recognizes syntactic variations and contexts of tags;
- The precision and recall of the search engine results increases.

3.2. Similarity Measures

In this section we discuss the different similarity measures used in the STCS framework as well as the notation that is used in this paper.

3.2.1. Levenshtein Distance Measure

In the STCS framework we use the *normalized Levenshtein distance*. We denote this similarity by lv_{ij}, which is the normalized Levenshtein distance between tag i and j. The normalized Levenshtein distance is defined as

$$lv_{ij} = \frac{alv_{ij}}{\max\left(\text{length}(t_i), \text{length}(t_j)\right)} \tag{1}$$

where alv_{ij} is the absolute Levenshtein distance [16].

The normalized Levenshtein distance addresses the string lengths. For example, if you have two strings of length 24, then an absolute Levenshtein distance of 3 is not large. However, with two strings of length 6 this distance is quite large (it is 50% of the tag length). According to the absolute Levenshtein distance these two distances are the same, but the normalized Levenshtein distances are in this case 0.125 and 0.5. This indicates that, according to the normalized Levenshtein distance, the first pair is more similar than the second pair.

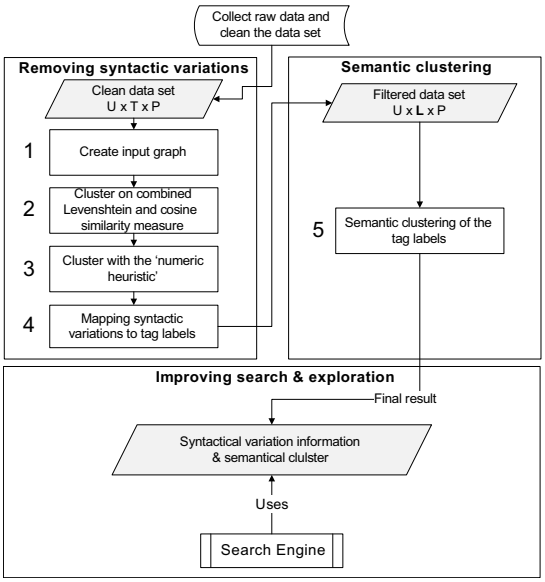

Figure 1: Overview of the STCS framework

3.2.2. Co-occurrence Data and the Cosine Similarity

To measure the semantic relatedness between tags, we use the cosine similarity based on co-occurrence vectors. The reason for this is that the authors of [6], who give a systematic characterization and validation of different tag similarity measures, conclude that co-occurrence vectors combined with the cosine similarity are useful for detecting concept hierarchies. We denote the cosine similarity by $\cos(a, b)$, where a and b are the co-occurrence vectors.

The range of the function $\cos(a, b)$ where $a, b \in \mathbb{R}^m$, with m representing the number of tags, is $[-1, 1]$. In the case that vectors $a, b \in {\mathbb{N}_0}^m$, the range is $[0, 1]$. We can interpret $\cos(a, b) = 0$ as 'not semantically related', and $\cos(a, b) = 1$ as 'fully semantically related'.

3.3. STCS Framework

As discussed in Section 3.1, the STCS framework is composed of three parts. In Section 3.3.1 we describe the process for removing syntactic variations from tags. In Section 3.3.2 we focus on finding semantically related tags, where we compare hierarchical versus non-hierarchical clustering methods. For the non-hierarchical clustering types we implement the method proposed by [28] and an adaptation of that algorithm. For the hierarchical clustering types we use the method proposed by [27] and also an adaptation of that method. In total we compare 4 methods with each other. The improvement of search and exploration is addressed in Section 3.3.3. Figure 1 gives an overview of the STCS framework.

3.3.1. Removing Syntactic Variations from Tags

The algorithm for the syntactic variation clustering uses an undirected graph $G = (T, E)$ as input. The set T contains elements which represent a tag id, and E is the set of weighted edges (triples (t_i, t_j, w_{ij})) representing the similarities between tags. To calculate the weight w_{ij} one needs the normalized Levenshtein distance lv_{ij} and the cosine similarity between tag i and j. The weight w_{ij} of an edge in the graph is then calculated as shown in Equation 2.

$$w_{ij} = z_{ij} \times \left(1 - lv_{ij}\right) + (1 - z_{ij}) \times \cos\left(\text{vector}\,(i), \text{vector}\,(j)\right) \tag{2}$$

where

$$z_{ij} = \frac{\max\left(\text{length}(t_i), \text{length}(t_j)\right)}{\text{length}(t_k)} \in (0, 1] \tag{3}$$

and

$$t_k \in T,\ \text{length}\,(t_k) \geq \text{length}\,(t)\ \forall t \in T,$$

with $t_i, t_j \in T$.

Normalized Levenshtein values are not representative for short tags, that is why the cosine value gets more weight as the maximum tag length gets shorter. This yields better results for shorter tags. Let us clarify this with an example. Given two tags 'walk' and 'wall', the normalized Levenshtein similarity is $1 - 1/4 = 3/4$, a high value for words which are not syntactic variations. Thus, if we use only the normalized Levenshtein distance in this case, these words would be marked wrongly as syntactic variations of each other. To address this problem we use the cosine similarity based on co-occurrence vectors and give it more weight for shorter tags. The cosine similarity indicates the level of semantic relatedness between two tags, as a corrective measure for syntactic similarity of short tags. The cosine similarity for 'walk' and 'wall' is so low that the framework correctly finds that these words are not syntactic variations of each other.

To build the input graph we first construct a set of tag nodes and edges (t_i, t_j, and w_{ij}). When creating the set, only the pairs where $t_i < t_j$ should be considered, i.e., the tag ID i is smaller than the tag ID j, as w_{ij} equals w_{ji}. With this set of tag nodes and edges, the input graph is built. Then, a root node is created and connected to each disconnected component of the graph, i.e., a cluster in the graph. The root is connected to a randomly chosen tag from each cluster. An example of an input graph for the syntactic clustering algorithm is shown in Figure 2. For example, $\{1, 5, 6, 7\}$ is a cluster which is connected to the root by the randomly chosen tag '1'. The root node functions as a pointer to clusters, which are used in the algorithm.

An overview of the algorithm for the syntactic clustering, which uses the previously built input graph, is described in Algorithm 1. This is step 2 of Figure 1.

From lines 1 and 2 we can see that the algorithm traverses each cluster of the initial input graph. For each cluster that is

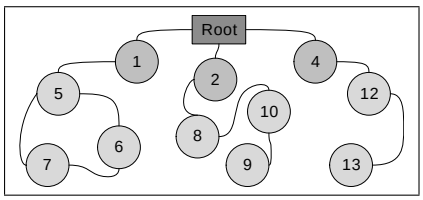

Figure 2: An example of an input graph for the syntactic variation clustering algorithm

Algorithm 1 Creating clusters that remove syntactic variation

1: **for all** $t \in T$ s.t. *root* and t are connected **do**
2: **for all** edges $e = \left(t_i, t_j, w_{ij}\right) \in E$ that are part of the cluster of t **do**
3: **if** $w_{ij} < \beta$ **then**
4: $E = E - \left\{\left(t_i, t_j, w_{ij}\right)\right\}$
5: **if** no edge from *root* to t_i **then**
6: create link from *root* to t_i
7: **end if**
8: **if** no edge from *root* to t_j **then**
9: create link from *root* to t_j
10: **end if**
11: **end if**
12: **end for**
13: cleanClusters()
14: **end for**

encountered, the algorithm checks every edge in the cluster. If the weight of an edge is below a certain threshold β, the edge is removed from the graph. When we remove an edge, we perform a check on the tags that were connected by the removed edge in lines 3 and 4. For both tags we analyze if they are still connected to the root node. If that is not the case, an edge is added from the root to that tag, which indicates that this tag now belongs to a new singleton cluster. This process is described in lines 5 to 9.

The algorithm also ensures that a tag only appears once in a cluster, this is accomplished by the cleanClusters() function in line 13. This function cleans the graph such that the root node is not reachable from two tags t_i and t_j and that there exists a path P from t_i to t_j with $root \notin P$. This is done by performing a depth traversal from the root. When the function visits a tag node, it is marked as 'visited'. During the traversal, if we encounter a tag node that has already been visited, we cut the connection between the root and the currently traversed cluster. If we do not perform this cut, the root is pointing to two identical clusters. We know this because every node should be visited once if we visit all nodes from the root using a depth traversal. The result after the call to cleanClusters() is a set of distinct clusters where each tag appears only once in a cluster. Let us clarify this process with an example. Consider the input graph that is given in Figure 2. Assume that only the edge between tag 1 and 5 and the edge between tag 5 and 6 are candidates for cutting (according to the condition in line 3). After these edges are cut, but before cleanClusters() is called in line 13, the root points both to tag 5 and tag 6 (as a result of the algorithm). The root now points to five tags, which indicates that we have five clusters, i.e., $\{1\}$, $\{5, 6, 7\}$, $\{6, 7, 5\}$, $\{2, 8, 9, 10\}$, and $\{4, 12, 13\}$. The function cleanClusters() starts visiting tags 5, 7, 6. After that, it visits tag 1. Next, it visits tag 6, which has been previously visited. The function cuts the edge between the root and tag 6, which merges the two duplicate clusters. At the end we obtain four clusters, i.e., $\{1\}$, $\{5, 6, 7\}$, $\{2, 8, 9, 10\}$, and $\{4, 12, 13\}$.

As a particularity, we find that the tagging data from Flickr has some properties. There are lots of tags which contain numbers. For instance, there are a lot of camera and lens types present. Because different cameras and lenses are semantically related and the names often differ only by one or two characters, there are many different cameras and lenses in one cluster. Also tags containing years are commonly used, for instance: 'spring 2008' versus 'spring 2009'. Product numbers also contain numeric data, e.g., 'BMW X5' versus 'BMW X6'.

To solve these issues, we propose a heuristic to deal with these numeric properties. Step 3 of Figure 1 is this numeric heuristic, which is similar to the previously described Algorithm 1, but with a different condition for cutting an edge (line 3). For this heuristic in the case of Flickr, we cut an edge if the extracted numbers from two tags are not equal to each other while the letters are the same. For example, an edge connecting 'Canon EF 24-105mm f/4 L IS USM' and 'Canon EF70-200mm f/4L IS USM' is cut as '241054' does not equal '702004'. If only one of the two tags contains numbers, or the extracted numbers are equal, no edges are cut.

In step 4 of Figure 1 we create a new data set which contains the tags that are a label of a syntactic cluster. The label of a cluster is the most frequently occurring tag in the data set. The newly created data set is used as input for the next steps.

3.3.2. Semantic Clustering

As shown in Figure 1, the semantic clustering process starts after the first step is completed, i.e., the syntactic variations have been removed from the data set. We now discuss the non-hierarchical and hierarchical semantic clustering methods.

Non-hierarchical Clustering

The non-hierarchical algorithm that we adapt, is described in Algorithm 2. It has originally been proposed by the authors of [28]. We indicate this clustering method by NHC (Non-Hierarchical Clustering). The algorithm is different from a classical clustering algorithm, as instead of using the centroid, all tags are used to calculate the distance between two clusters. This has the advantage that all the elements within a cluster must be similar amongst each other, instead of being similar just to the centroid. We improve the algorithm by replacing a heuristic for merging similar clusters by two new heuristics.

In lines 1 to 10 of Algorithm 2 the initial clusters are created. This is done by starting with each tag as a cluster, and adding the rest of the tags to that cluster if they are sufficiently similar to that cluster. A tag is sufficiently similar if the average cosine of that tag with respect to all elements in the cluster are larger than χ. This is shown in line 5.

Because many tags are similar to each other, the set of initial clusters can contain many duplicate or nearly duplicate clusters. Therefore we need to merge some of the clusters, which is done in lines 11 to 20 of Algorithm 2. In [28], two heuristics are proposed for this purpose. The first heuristic merges two clusters if one cluster contains the other cluster. This means that if the larger cluster contains all the tags of the smaller cluster, we remove the smaller cluster. The second heuristic checks if clusters differ within a small margin, i.e., the number of different tags in the smaller cluster compared to the larger cluster represents less than a percentage of the number of tags in the smaller cluster. If this is the case, then the distinct words from the smaller cluster are added to the larger cluster and the

Algorithm 2 Semantic clustering

Require: avgcosine (a, b), as defined by Equation 4 gives the average cosine between elements $(a - b)$ and b
Require: normdiff (x, y), as defined by Equation 6 gives the normalized difference between clusters x and y
1: $C = \{\emptyset\}$
2: **for all** $t \in T$ **do**
3: $c = \{t\}$
4: **for all** $t' \in T$ similar to t **do**
5: **if** average cosine of t' with all tags in c is above χ **then**
6: $c = c \cup \{t'\}$
7: **end if**
8: **end for**
9: $C = C \cup \{c\}$
10: **end for**
11: $C' = \{\emptyset\}$
12: **for all** $y \in C$ in descending order of cluster size **do**
13: **for all** $y' \in C$ in descending order of cluster size $\wedge\ y' \neq y$ **do**
14: **if** $y' \subseteq y \vee$ avgcosine $(y', y) > \delta \vee$ normdiff $(y', y) < \varepsilon$ **then**
15: $C = C - \{y'\}$
16: $y = y \cup y'$
17: **end if**
18: **end for**
19: $C' = C' \cup \{y\}$
20: **end for**

smaller cluster is removed. The latter heuristic has limitations, because it uses a constant percentage, i.e., a constant threshold for merging clusters, no matter the size of the smaller cluster.

First, let us clarify these limitations in more detail. If one would use a constant threshold, it is hard to choose such a threshold value where the larger clusters do not merge too easily and the smaller clusters too difficultly. Consider two clusters K and L where $|K| \geq |L|$, with $|\cdot|$ indicating the size of a set. The maximum number of different elements for the two sets to be merged is growing constantly with the size of cluster L. This is clear when we analyze the function which calculates the maximum number of different elements for the sets to be merged. The function is: $f(|L|) = \lfloor \varepsilon \cdot |L| \rfloor$ where ε is the already mentioned threshold. For example, for $\varepsilon = 0.20$ we have $f(|L|) = \lfloor 0.20 \cdot |L| \rfloor$. If we evaluate $f(30)$ we get 6. This means if we have a cluster L with $|L| = 30$ and a cluster K with $|K| \geq 30$, L would be merged into K if $|D| \leq 6$, where $D = L - K$. This also means that any clusters with size below 4 are not merged, because $f(4) = 0$. To address this, we propose a dynamic threshold, which we will discuss below. Based on our experiments, we found that a dynamic threshold, instead of a constant one, improves the clustering technique.

We now discuss the modified version of the NHC algorithm, which we name as STCS NHC (Semantic Tag Clustering Search Non-Hierarchical Clustering). The STCS NHC employs the first heuristic that is used in the NHC algorithm, which is a trivial one, but does not use the second heuristic. The reason for this is that the second heuristic provides serious limitations due to the constant threshold (as previously discussed). We replace this second heuristic with two new heuristics, which we call the second and third heuristic from now on. The second heuristic considers the semantic relatedness of the difference between two clusters. The third heuristic considers the size of the difference between two clusters in combination with a dynamic threshold. We show that these three heuristics, the first proposed by [28], and the second and third proposed in this paper, improve the clustering technique from [28].

The three heuristics are used in line 14 of the algorithm introduced earlier. The part $y' \subseteq y$ represents the first heuristic, avgcosine $(y', y) > \delta$ represents the second heuristic, and normdiff $(y', y) < \varepsilon$ represents the third heuristic. Basically, the second heuristic, which we propose, merges two clusters K and L, where $|K| \geq |L|$, when the average cosine avg of all $d \in L - K$ and elements of the larger cluster is above a certain threshold δ. The average cosine of these elements is defined as

$$\text{avgcosine}\,(K, L) = \sum_{d \in L-K} \frac{Avg_d}{|L-K|}\ , \tag{4}$$

where

$$Avg_d = \sum_{x \in K} \frac{\cos(\text{vector}(x), \text{vector}(d))}{|K|}\ , \tag{5}$$

with $|K| \geq |L|$.

The third heuristic merges the clusters when the normalized difference between the clusters is smaller than a dynamic threshold ε. The normalized difference η is defined as

$$\eta = \frac{|D|}{|L|}\ , \tag{6}$$

where $D = L - K$. We propose to define threshold ε as

$$\varepsilon = \frac{\phi}{\sqrt{|L|}}\ , \tag{7}$$

and thus $f(|L|)$ can be described as

$$f(|L|) = \lfloor \varepsilon \cdot |L| \rfloor = \left\lfloor \phi \cdot \sqrt{|L|} \right\rfloor\ . \tag{8}$$

One is able to tune the distribution of the maximum allowed difference between clusters by means of the parameter ϕ. Thus, we create a function that improves the clustering process, as it takes better in account the size of the smaller cluster.

Hierarchical Clustering

For hierarchical clustering we adapt, as already said, the algorithm of [27]. We denote this algorithm by HC (Hierarchical Clustering). We first discuss the original proposed method and then the modification we propose.

In [27] the authors define a subsumption model. In this model, tag x potentially subsumes tag y (x is a parent of y) if

$$\begin{aligned} P(x|y) \geq t \text{ and } P(y|x) < t, \\ D_x \geq D_{\min}, D_y \geq D_{\min} \\ U_x \geq U_{\min}, U_y \geq U_{\min} \end{aligned} \tag{9}$$

where t is the co-occurrence threshold, D_x is the number of documents in which tag x occurs, and U_x is the number of users

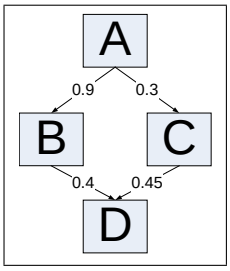

Figure 3: 'Step-by-Step' versus 'Longest Path', hierarchical clustering

that use x in at least one image annotation.

The first step is to calculate the co-occurrence statistics. Once the co-occurrence statistics are calculated, candidate term pairs are selected using the specified constraints. A graph of possible parent-child relationships is then built. To clean up the graph, the co-occurrence of nodes with ancestors that are not parents are removed. So for example, for a given term x, and two potential parent terms p_i and p_j, if p_i is also a potential parent term of p_j, then p_i is removed from the list of potential parent terms for term x. At the same time, the co-occurrence of terms x, p_i and p_j in the given relationships indicates both that the (p_j, x) relationship is more likely than the simple co-occurrence might indicate, and similarly that the (p_i, p_j) relationship should be reinforced in such a case, the author increments the co-occurrence statistic by 1. After the paths are cleaned and reinforced, each leaf in the tree is considered and the 'best' path is chosen up to a root, given the (reinforced) co-occurrence weights. In the end, these paths are coalesced into trees. The best path is chosen by starting at a leaf and then choosing the best parent, i.e., the one with the highest co-occurrence.

We propose a modified version of this algorithm, which we denote by STCS HC (Semantic Tag Clustering Search Hierarchical Clustering). Instead of choosing the 'best' path up to a root using a step-by-step method, we use the longest path from a leaf up to a root. Figure 3 shows an example of a graph where we need to find the 'best' path from leaf 'D'. With the step-by-step method proposed by [27], the 'best' path would be 'D, C, A', because $P(C|D) > P(B|D)$. With the longest path, the path would be 'D, B, A', because $P(B|D)+P(A|B) > P(C|D)+P(A|C)$. Clearly the path 'D,B,A' is preferred as the total co-occurrence values are larger, i.e., the total relationship weight is stronger.

3.3.3. Improving Search and Exploration in Tag Spaces

After the syntactic and semantic clusters are created, the search engine can utilize these clusters to obtain the information with which search can be improved. We start by retrieving all the images which contain at least one of the query tags. The results are sorted on a defined similarity measure between a query and a picture. For this purpose, the cluster of a query tag is also used in order to find the context. We begin by defining the query q as a m dimensional vector of tags q_i, and a picture p as a n dimensional vector of tags p_j: $q = [q_1 \cdots q_m]$ and $p = [p_1 \cdots p_n]$. Equation 10 shows function $g(q_i, p)$, which is used to compute the similarity between a query tag and a picture.

$$g(q_i, p) = \frac{1}{n \times |C_i|} \sum_{c_j \in C_i} \sum_{k=1}^{n} \cos\left(c_j, p_k\right) \tag{10}$$

The function computes an average cosine similarity between all cluster tags (including the query tag) and all picture tags. The term C_i represents the cluster of tags to which q_i belongs. If there is more than one cluster for a particular tag, then this represents the cluster that is chosen by the user.

For a given query, we first collect all the images. Then, for each query tag we compute the similarity $g(q_i, p)$, of which we compute the average across all query tags to obtain a final similarity between the query q and a picture p. The retrieved pictures are then sorted descending on the final similarity and presented to the user.

An important feature of the search engine is the automatic replacement of syntactic variations by their corresponding labels. Steps 1 through 4 of Figure 1 generate labels which are mapped to tags. These tags are then seen as variations of their tag label. When a tag has no variations, the tag label is represented by the tag itself. The search engine can utilize this information by searching for each keyword not only on the verbatim keyword, but also on all syntactic variations of the keyword. For example, when the user searches for 'self portraits', the system searches for 'self portraits', 'self-portraits', 'selfportraits' and even 'self portaits'. This greatly increases the recall, the number of returned correct results. This method is independent of the semantic clustering type (hierarchical or non-hierarchical).

Another feature of the search engine is that it is able to detect contexts. If a tag can have multiple meanings, the search engine asks the user to choose a cluster to indicate the meant sense for the tag. In this way, clusters are used as approximations of the many contexts a tag can participate in. For the non-hierarchical clustering techniques we utilize the semantic clusters of a tag. If a tag occurs in more than one cluster, it is considered to have multiple contexts. The user then gets a message with the different clusters that the current tag is in. For the hierarchical clustering technique we assume the presence of multiple contexts if a tag appears in more than one tree. For example, if the tag 'Apple' is the child of the tag 'Fruit' in one tree, and a parent of 'iPod' in another tree, then we infer that 'Apple' has two contexts.

Searching is also improved by providing the user useful information about the query. Each clustering type provides different information, for instance, a hierarchical clustering technique can provide information concerning the hierarchy, and a non-hierarchical clustering technique can not. With this information the user understands the semantic structure of the query and the data, and can manually relax (or narrow) this query depending on what the goal is.

We choose not to relax the queries automatically, because this is not the pursued goal in this paper. The aim is to find which clustering technique helps the most in improving searching by choosing the correct contexts of tags. If we were to choose an automated query relaxation technique, this could have a biased effect on the clustering techniques. That is the reason why we

decide to let the user relax the queries.

There are different scenarios where the above technique can be applied. Sometimes the user does not find enough results. It could also be that there are too many results, or that the results are not specific enough (when searching on something particular). In any case, the user is able to view semantic information about the query. For the non-hierarchical clusters this means that the user can inspect the whole cluster for each tag. The tags within the cluster are ordered on cosine similarity with the query tag so that the most similar tags are on top of the list. The hierarchical clusters give the opportunity to provide the user with more information about each tag, as we can show the position of a tag within a tree. The user can then decide to manually relax the query (by going one or more levels up the tree) or to use a more constrained query (by going one or more levels down the tree).

4. Framework Implementation

We provide an implementation of the STCS framework, called XploreFlickr.com [29]. XploreFlickr.com is a Web application implemented in Java. In this section we explain briefly XploreFlickr.com and its features. We cover data processing, together with the features for syntactic clustering, semantic clustering, and improving search and exploration.

4.1. Data Processing

For the experiments based on the implementation of XploreFlickr.com, we collected a data set from the Flickr database. To lower the retrieval time, the data has been collected in parallel from two non-overlapping intervals [2008-1-14, 2008-8-1] and [2008-8-12, 2009-2-28]. The data set contains 1 683 111 associations, 57 009 users, 166 544 pictures, and 317 657 tags. The data set is obtained by using the Flickr API [33]. All pictures which belonged to the category 'Interesting Photos' and were uploaded in the previously mentioned time intervals were collected.

After collecting the data, we first perform some cleaning steps. As already mentioned in Section 1, users have no limitations when they add tags to pictures. Because of this, the pictures in the initial data set have many unusable tags. To address this problem we apply a sequence of filters. These filters, for example, remove tags with unrecognizable signs, tags which are complete sentences, etc. After applying these filters, we have a final data set that we use as input for the STCS framework. The final data set contains 1 231 818 associations, 50 986 distinct users, 147 132 pictures, and 27 401 tags. The syntactic clustering makes use of the full cleaned data set. The semantic clustering part and the 'improving search and exploration' part use the data set of the top 5000 most frequent tags (for performance reasons).

4.2. Clustering algorithms

For the syntactic variations part, we implemented the corresponding framework step. The application creates mappings between tag labels and possible syntactic variations of tags. For the semantic clustering part, we implemented two separate components, one for the non-hierarchical clustering and one for the hierarchical clustering. Each component is configured two times, one time for the original clustering algorithm and one time for the adapted clustering algorithm.

The algorithms for semantic clustering rely on the cosine similarities between the tag co-occurrence vectors. For these algorithms, the computational complexity is therefore $O(n^2)$, where n is the number of tags. This is due to the fact that the clustering algorithms need the cosine similarity for each pair of tags, i.e., $(n^2 - n)/2$ cosine similarities.

4.3. XploreFlickr.com search engine

With XploreFlickr.com, the user can choose between 5 different search methods. There is the Dummy search engine, and the search engines which utilize semantic clusters, provided by one of the four semantic clustering methods presented in this paper (NHC, STCS NHC, HC, and STCS HC). The Dummy method is a search method which is used to benchmark the STCS framework. We implemented the Dummy search to simulate 'standard' simple search engines. Such search engines retrieve the pictures that contain at least one tag from the query and do not apply any intelligent sorting of the results. The other four search methods are cluster-based and utilize the search algorithm presented in Section 3.3.3 to retrieve pictures.

Unlike the Dummy search engine, a cluster-based search engine automatically detects syntactic variations in the query and informs the user that the variations are appended to the query. For example, when the query is 'self portait' (a typographical mistake), pictures that are tagged with 'self portait' or 'self portraits' are returned. By considering the clusters a tag occurs in, the Web application also shows to the user the possible contexts for the query. If more than one context is found, the user is asked to select one of the contexts of the tag. After the selection the search results consist of images related to that specific meaning. For instance, when someone searches on 'Apple' the user should be asked to select the context of 'Apple' by proposing two different clusters, one cluster on 'Apple' and 'fruit' and one cluster on 'Apple' and 'brand'. The images that contain at least one of the selected cluster tags are returned and sorted based on the average cosine similarity between the image tags and the selected cluster tags.

The non-hierarchical search engines, which use the NHC and STCS NHC clusters, present the contexts as clusters with a flat textual representation. For instance, for the tag 'New York', a cluster could be {'New York', 'nyc', 'new york city', 'Queens', 'Central Park', 'Times Square'}. For the hierarchical methods, which use the clusters from the HC and STCS HC, the clusters are actually trees of tags which are shown on screen in a hierarchical style. The context choice, which is activated when multiple clusters/trees are found, remains the same for both the non-hierarchical and hierarchical search methods. The user selects one of the presented clusters/trees as the context for a certain tag.

5. Evaluation

In this section we present and explain the results of removing syntactic variations, creating semantic clusters, and the proposed improvements for search and exploration in tag spaces. For all algorithms, the evaluation on the training and test data sets is done by an independent (from the authors) panel of three users. The optimal parameter values for the algorithms were chosen by applying a hill climbing procedure.

5.1. Syntactic Variations

We start by analyzing the results for syntactic variation detection. Using a training set of 50 tags, we have found that the optimal value for the threshold β (for cutting edges) is 0.62. The algorithm for removing syntactic variations finds 1687 syntactic variations of the total 25 714 tags. In order to analyze the performance of the system, we create a test set by randomly choosing 200 tag combinations that are classified as syntactic variations by the STCS framework. The distributions of the tag length for the test data set and the original data set are approximately the same. The results indicate that the framework produces 10 mistakes. Thus, for this test data set, the precision is 0.95. A few examples taken from these 10 errors are 'clouds' and 'colours', 'blueberry' and 'blackberries', and 'Western Australia' and 'BestOfAustralia'. A few correct examples are: 'flat-coated retriever' and 'flatcoatedretriever', 'turquoise' and 'turqoise', and 'autumn' and 'automne'.

Stemming algorithms can also be used to remove syntactic variations between tags, but we have found that basic stemming algorithms do not provide sufficient precision. In order to compare the performance, we have chosen two well-known stemming algorithms, Porter stemming [23] and Lovins stemming [17]. We have found that stemming algorithms only find syntactic variations in our context which are well-formed singular/plural variations. Other variations, for example 'new york' and 'new-york', or 'fought' and 'fight', are not found by the stemming algorithms. This is the reason why the stemming algorithms achieve low precision on our data set. For Porter's stemming algorithm, we report a precision of 0.19 and for Lovins stemming algorithm a precision of 0.23. Clearly our algorithm, which achieved a precision of 0.95, is outperforming the two stemming algorithms.

We also found that the numeric heuristic of the algorithm is important. On Flickr, a lens tag usually contains the brand, product line, product attributes, aperture, and focal length. For example, the tag 'Canon EF 70-200mm f/4 L IS USM' represents a 'Canon' lens, of product line 'EF', with product attributes 'L IS USM', with aperture 'f/4', and focal length '70-200mm'. Without the numeric heuristic, the lens tag 'Canon EF 24-105mm f/4L IS USM' and 'Canon EF 70-200mm f/4 L IS USM' would be considered as syntactic variations, although they represent different lenses. The first lens has a focal length of '70-200mm', while the second lens has a focal length of '24-105mm'.

5.2. Semantic Clustering

We now first discuss the results of the non-hierarchical clustering approach and then the results for the hierarchical clustering approach.

5.2.1. Non-hierarchical Clustering

The STCS NHC algorithm requires three parameters to be set, χ, δ, and ϕ. In order to employ the hill climbing procedure, we have used a training data set size of 100 clusters. The optimal value for the threshold χ is 0.8, this threshold determines whether or not a tag is added to a cluster during the initial cluster creation. For the threshold δ we found a value of 0.7 to give the best results. As parameters for the function that defines the dynamic threshold ε we use $\phi = 0.8$.

For the test data set, we randomly chose 100 clusters. We did not take a larger sample because the process of manual evaluating clusters is a time consuming task. For each cluster the number of misplaced tags is counted, i.e., tags that should have been placed in another cluster. The total number of tags in this randomly selected test data set is 458 and we encounter 44 misplaced tags. Thus, for this data set, the error rate is $44/458 \times 100 \approx 9.61\%$.

One should note that most of the misplaced tags are part of a large cluster (size of 20 or larger). For example, a cluster of size 49 contained 21 misplaced tags. This cluster contains tags about subjects like 'outdoor' and 'nature'. There are also, for example, tags which represent colors. Although colors are somehow related to 'nature', we mark these tags as misplaced as well.

In general, the algorithm finds many relevant clusters. Examples are {'rainy', 'Rain', 'wet', 'raining'}, {'turquoise', 'aqua', 'clear', 'cyan'}, {'iPod', 'iphone', 'mac'}, and {'South', 'north', 'west'}. Furthermore, a lot of clusters are found that actually contain tags from different languages. Examples of these clusters are {'Praha', 'Czech republic', 'praga', 'Czech'}, {'paris', 'frankreich', 'francia'}, {'Eau', 'Wasser'}, and {'springtime', 'primavera'}.

To benchmark the STCS NHC algorithm with the original algorithm proposed by [28], the NHC algorithm, we apply the same evaluation procedure as before. We create a test data set with 100 randomly chosen clusters, which contains 467 tags, to estimate the error rate. The difference between the NHC and STCS NHC algorithm, is that the NHC algorithm essentially uses the heuristics 1 and 3 described in Section 3.3, with a constant threshold ε for heuristic 3. Using the same training set as for the STCS NHC algorithm, we find that the optimal value for this constant threshold is $\varepsilon = 0.2$.

We encounter 61 misplaced tags, thus with this data set, the error rate for [28] is $61/467 \times 100 \approx 13.06\%$. When we compare this error rate to the error rate of our algorithm (9.61%), we conclude, based on the error rate on the test data set, that our algorithm outperforms the algorithm proposed in [28]. Furthermore we see that our algorithm produces 739 clusters, and the algorithm in [28] produces 421 clusters. Thus, our algorithm discovers more clusters and thus relationships between tags. A performance summary is given in Table 1. As the cluster count is not a sufficient measure for cluster quality in an economic sense, we provide an extended user-based search evaluation (see Section 5.3).

To further investigate the performance of our system, we compare it with two other clustering algorithms: a fitness-based

Table 1: Non-hierarchical semantic clustering, performance summary on the test data set

	Error rate	Number of clusters	Avg. cluster size	Min / Max. cluster size
NHC	13.1%	421	4.6	2 / 63
STCS NHC	9.6%	739	4.4	2 / 67

clustering algorithm and the K-means algorithm. The used fitness-based algorithm is presented in [15]. This algorithm uncovers the hierarchical and overlapping community structure of complex networks using a fitness-based clustering algorithm. It discovers the natural community of each node in a graph by optimizing the fitness function using local iterative searching. We implemented this algorithm by using the cosine similarity of the co-occurrence vectors between the tags as edge weight. The test set is the set of all clusters which contain the tags used in the previously described test data set (467 tags). We report an error of 14.22% on this data set.

Finally, we compare our algorithm with the K-means algorithm [11]. We have applied the K-means algorithm on the co-occurrence data using two distance measures, the Euclidean distance, and the cosine distance. We report an error rate of 26.12% for the Euclidean distance and 16.24% for the cosine distance. The cosine distance provides better results for this type of data (co-occurrence data).

We can conclude that the STCS NHC algorithm, with respect to the error rate, also performs better than the K-means algorithm and the fitness-based algorithm presented in [15].

5.2.2. Hierarchical Clustering

For the HC and STCS HC algorithms, we have applied the same procedure as before. The thresholds (t, D_{min}, and U_{min}) are determined using a training data set consisting of 100 clusters. These 100 clusters are randomly chosen and evaluated by the three users. The parameters are set to $t = 0.40$, $D_{min} = 30$, and $U_{min} = 30$. This gives the best trade-off between the number of correct edges in the hierarchy and the error rate. Higher values than 0.40 for the parameter t results in the absence of valid subsumption pairs.

To evaluate both hierarchical clustering algorithms, we selected 91 trees from the total data set (trees represent the clusters for the hierarchical algorithms). For each cluster, the proposed subsumption pairs, i.e., the edges, are evaluated according to the measures presented in [27]. Each proposed subsumption pair is marked as correct, related, synonymous (including language variants), inverted, or noise (erroneous).

The edges that are marked as correct are truly a 'type of' relationship. For example, 'Color' → 'red' is marked as correct, because 'red' is a type of color. For generic terms like 'lake' and 'park', we considered instances of lakes or parks also to be adequate children. An example of an edge that is of type 'related' is 'restaurant' → 'food'. The 'synonymous' relationship type is used when the parent and the child are synonyms; this can be also in different languages. An example of this type of edge is 'eyes' → 'ojos' and 'eyes' → 'eye'. Inverted types of edges are

Table 2: Relationship classification for hierarchical clustering

	Correct	Related	Synonymous	Inverted	Error Noise
HC	37.10%	19.35%	34.68%	3.23%	5.65%
STCS HC	39.70%	23.60%	30.34%	3.75%	2.62%

edges where the child subsumes the actual parent. An example of this is 'red' → 'color', this is clearly wrong as 'red' is a color, and 'color' is not 'a red'. Error or noise edges often contain either Flickr specific or named entities as children or parents. An example of a Flickr specific term is 'HBW', which stands for 'Happy Bokeh Wednesday'. It refers to an online Flickr photo contest (created by users) which is held every Wednesday.

Table 2 shows the evaluation results for the original method proposed by [27] (HC algorithm), and the adapted version of it (STCS HC algorithm). We can see that the STCS HC algorithm, which uses the longest path selection method, has a better result with respect to the total correct and synonymous relationships. In other words, more correct and less synonymous relationships are found. The error rate of the STCS HC is also approximately twice as low.

5.3. Searching Tag Spaces

In this section we evaluate the improvement of search and exploration in tag spaces. We first start by comparing different search engines on XploreFlickr.com. The comparison is based on the 'precision' of the first 24 results when a user queries the system. Other statistical measures like accuracy, sensitivity, or specificity are difficult to derive, as the size of the data set is too large. To evaluate the usefulness of the different semantic clusters, we performed a user-based questionnaire. Three persons were presented for each search engine 70 queries, where for each query 24 pictures were displayed. The queries were tags which were located in multiple clusters and therefore, by the definition of the STCS framework, have multiple contexts. The context is predetermined and the cluster corresponding to this context is utilized by the search engines for the retrieval of the pictures. The user is instructed to mark the pictures which he or she finds not to be a match with the query. We only considered the first 24 results for every query, because this is the number of results which is returned on the first results page of XploreFlickr.com. Table 3 shows the precision and inter-annotator agreements for each search engine. We observe that the clusters obtained from the STCS HC algorithm give the best results. By using the Wilcoxon signed-rank test [31], we found three statistically significant relationships, which are illustrated in Table 4. Our first observation is that the cluster-based search engines clearly outperform the Dummy search engine. We further observe that both the STCS HC and STCS NHC search engines outperform the HC search engine.

Besides evaluating if the semantic clusters improve the search precision, we also evaluated their ability to recognize different

Table 3: Search engine evaluation

	Precision	Inter-annotator agreement
Dummy	88.91%	90.30%
NHC	95.04%	93.69%
STCS NHC	95.62%	94.64%
HC	94.32%	94.01%
STCS HC	96.19%	95.03%

Table 4: Statistically significant findings at $\alpha = 5\%$

Relation
ALL > Dummy
STCS NHC > HC
STCS HC > HC

contexts of tags. The method of [28] (NHC) finds 214 tags which occur in at least two different clusters, this means that these tags have at least two different contexts. The STCS NHC algorithm finds 368 tags with at least two contexts for each tag. After an analysis of all proposed contexts, we find that the STCS NHC algorithm finds more and correct contexts than the NHC algorithm. An example of a tag in the data set which has multiple contexts is the tag 'daughter'. This tag is found in the semantic cluster {'baby', 'daughter'}, but also in the cluster with the tags {'mother', 'daughter', 'father'}. When one uses XploreFlickr.com and chooses the first cluster only baby girl pictures are found, but for the second cluster all kind of family pictures are presented. The same analysis is performed on the suggested contexts for the hierarchical clusters. We find that the HC algorithm detects 46 tags with at least two clusters, and the STCS HC algorithm detects 54 tags with at least two clusters.

6. Conclusions and Future Work

In this paper we have presented the Semantic Tag Clustering Search (STCS) framework for searching and browsing through tag spaces. Our framework makes use of clusters to deal with syntactic and semantic variations of tags.

For the syntactic clustering process we propose a measure that uses the normalized Levenshtein value in combination with the cosine value based on co-occurrence vectors. In this way we are able to detect syntactic variations of short tags. The results also show that tags which contain numbers can cause unwanted results. The 'numeric heuristic' in the STCS framework effectively deals with this issue and is therefore highly recommended. We have shown that the STCS syntactic variation algorithm achieves a precision of 0.95 on our test data set.

We compared two non-hierarchical (NHC, STCS NHC) and two hierarchical (HC, STCS HC) semantic clustering techniques. Besides the regular semantic related tags, we observe that the STCS NHC algorithm is able to find clusters of tags which are synonyms represented in different languages. There are some problems with the non-hierarchical clustering methods, as certain clusters are quite large. The result of this is that there are clusters which contain tags that are individually semantically related, but the semantic relatedness as a whole, is low. We deal with this issue by introducing a heuristic for merging clusters with a dynamic threshold, which is implemented by the STCS NHC algorithm. Consequently, on the test data set, the STCS NHC algorithm outperforms the NHC algorithm. The results obtained from the experiments show that the STCS NHC algorithm performs better in terms of precision, and produces finer-grained clusters. For semantic clustering we have employed a different method for finding a path in order to improve clustering. Unlike the HC algorithm, the STCS HC algorithm considers all paths when determining the path from a tag node to the root. Our results show that the STCS HC algorithm provides better performance than the HC algorithm by finding more correct 'type of' relationships and less synonymous relationships.

We have found that clusters can be used to provide valuable information for tag search engines. In the STCS framework, different tag contexts are detected by considering tags which appear in more than one cluster. When considering the precision for the first 24 pictures, we conclude that all search methods significantly provide higher precision than the Dummy search method. We find that the search methods that utilize the clusters from the STCS HC and STCS NHC algorithms obtain the highest precision. The precision of these search methods is significantly higher than the precision of the search method that uses the clusters of the original HC algorithm.

6.1. Future Work

There are several aspects of the STCS framework which could be improved. First, we would like to further improve the syntactic variation detection process. One could investigate how to detect syntactic variations which contain abbreviations of words. Also, the use of an adapted Levenshtein algorithm, where the different edit operations have different costs, should be investigated. Second, the non-hierarchical semantic clustering leaves also room for improvement. In this paper we propose two new heuristics for merging similar clusters. The condition to merge is a disjunction of the two new heuristics and an earlier proposed heuristic. One could also consider other combinations, like a conjunction of the two new heuristics, to investigate if this improves the clustering process. Finally, for the improvement of search and exploration, one could analyze user statistics by for the different cluster-driven search methods. With this information one gets insight in the user interaction with a tag space search engine for a considerable amount of time.

[1] Abel, F., Henze, N., Krause, D., 2008. Analyzing Ranking Algorithms in Folksonomy Systems. Tech. rep., L3S Research Center, `http://groupme.org/papers/techreport-ranking-in-folksonomies.pdf`.

[2] Abel, F., Henze, N., Krause, D., 2008. Ranking in folksonomy systems: can context help? In: 16th Conference on Information and Knowledge Management (CIKM 2008). ACM, pp. 1429–1430.

[3] Arakji, R., Benbunan-Fich, R., Koufaris, M., 2009. Exploring contributions of public resources in social bookmarking systems. Decision Support Systems 47 (3), 245 – 253.

[4] Bao, S., Xue, G., Wu, X., Yu, Y., Fei, B., Su, Z., 2007. Optimizing Web search using social annotations. In: 16th International conference on World Wide Web (WWW 2007). ACM, pp. 501–510.

[5] Begelman, G., Keller, P., Smadja, F., 2006. Automated tag clustering: Improving search and exploration in the tag space. In: 15th World Wide Web Conference (WWW 2006). ACM, pp. 22–26.
[6] Cattuto, C., Benz, D., Hotho, A., Stumme, G., 2008. Semantic Grounding of Tag Relatedness in Social Bookmarking Systems. In: 7th International Semantic Web Conference (ISWC 2008). Vol. 5318 of Lecture Notes in Computer Science. Springer, pp. 615–631.
[7] Clough, P., Joho, H., Sanderson, M., 2005. Automatically organising images using concept hierarchies. In: Proceedings of Multimedia Information Retrieval 2005. ACM.
[8] Ding, L., Finin, T., Joshi, A., Pan, R., Cost, R. S., Peng, Y., Reddivari, P., Doshi, V., Sachs, J., 2004. Swoogle: a search and metadata engine for the semantic web. In: 13th International Conference on Information and Knowledge Management (CIKM 2004). ACM, pp. 652–659.
[9] Echarte, F., Astrain, J. J., Córdoba, A., Villadangos, J., 2008. Pattern Matching Techniques to Identify Syntactic Variations of Tags in Folksonomies. In: 3rd World Summit on The Knowledge Society (WSKS 2008). Communications in Computer and Information Science. Springer, pp. 557–564.
[10] Hamming, R. W., 1950. Error detecting and error correcting codes. Bell System Technical Journal 26 (2), 147–160.
[11] Hartigan, J. A., 1975. Clustering Algorithms. Wiley.
[12] Heymann, P., Garcia-Molina, H., April 2006. Collaborative Creation of Communal Hierarchical Taxonomies in Social Tagging Systems. Tech. Rep. 2006-10, Stanford InfoLab, http://ilpubs.stanford.edu:8090/775/.
[13] Hotho, A., Jäschke, R., Schmitz, C., Stumme, G., 2006. Folkrank: A ranking algorithm for folksonomies. In: Workshop Fachgruppe Information Retrieval 2006 (FGIR 2006).
[14] Kome, S. H., November 2005. Hierarchical subject relationships in folksonomies. Master's thesis, University of North Carolina, http://etd.ils.unc.edu/dspace/handle/1901/238.
[15] Lancichinetti, A., Fortunato, S., Kertesz, J., 2009. Detecting the Overlapping and Hierarchical Community Structure in Complex Networks. New Journal of Physics 11 (3), 1–19.
[16] Levenshtein, V. I., 1966. Binary codes capable of correcting deletions, insertions, and reversals. Soviet Physics Doklady 10 (8), 707–710.
[17] Lovins, J. B., 1968. Development of a stemming algorithm. Mechanical Translation and Computational Linguistics 11 (1-2), 22–31.
[18] Mika, P., 2005. Ontologies Are Us: A unified model of social networks and semantics. In: 4th International Semantic Web Conference (ISWC 2005). Vol. 3729 of Lecture Notes in Computer Science. pp. 522–536.
[19] Miller, G., 2012. Wordnet - a lexical database for the english language. http://wordnet.princeton.edu.
[20] Newman, M. E. J., Girvan, M., 2004. Finding and evaluating community structure in networks. Physical Review E 69 (2), 026113.
[21] Nov, O., Ye, C., Kumar, N., 2012. A social capital perspective on meta-knowledge contribution and social computing. Decision Support Systems 53 (1), 118 – 126.
[22] Page, L., Brin, S., Motwani, R., Winograd, T., 1998. The pagerank citation ranking: Bringing order to the web. In: 7th International World Wide Web Conference (WWW 1998). ACM, pp. 161–172.
[23] Porter, M. F., 1980. An algorithm for suffix stripping. Program 14 (3), 130–137.
[24] Pothen, A., Simon, H. D., Liou, K.-P., 1990. Partitioning sparse matrices with eigenvectors of graphs. Matrix Analysis and Applications 11 (3), 430–452.
[25] Sanderson, M., Croft, B., 1999. Deriving concept hierarchies from text. In: Conference on Research and Development in Information Retrieval (SIGIR 1999). ACM, pp. 206–213.
[26] Schmitz, C., Hotho, A., Jäschke, R., Stumme, G., 2006. Mining association rules in folksonomies. In: Data Science and Classification. Studies in Classification, Data Analysis, and Knowledge Organization. Springer, pp. 261–270.
[27] Schmitz, P., 2006. Inducing Ontology from Flickr Tags. In: 15th World Wide Web Conference (WWW 2006). ACM, pp. 206–209.
[28] Specia, L., Motta, E., 2007. Integrating Folksonomies with the Semantic Web. In: 4th European Semantic Web Conference (ESWC 2007). Vol. 4519 of Lecture Notes in Computer Science. Springer, pp. 503–517.
[29] Vandic, D., van Dam, J., 2010. Xploreflickr.com. http://www.xploreflickr.com.
[30] Vandic, D., van Dam, J. W., Hogenboom, F., Frasincar, F., 2011. A Semantic Clustering-Based Approach for Searching and Browsing Tag Spaces. In: 26th Symposium on Applied Computing (SAC 2011). ACM, pp. 1698–1704.
[31] Wilcoxon, F., 1945. Individual Comparisons by Ranking Methods. Biometrics Bulletin 1 (6), 80–83.
[32] Wu, H., Gordon, M. D., Fan, W., 2010. Collective taxonomizing: A collaborative approach to organizing document repositories. Decision Support Systems 50 (1), 292 – 303.
[33] Yahoo, 2012. Flickr - photo sharing. http://www.flickr.com/services/api.