SOFTWARE – PRACTICE AND EXPERIENCE
Softw. Pract. Exper. (2012)
Published online in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/spe.2150

Exploiting tag similarities to discover synonyms and homonyms in folksonomies

Davide Eynard[1,*,†], Luca Mazzola[1,‡] and Antonina Dattolo[2,§]

[1]*Institute for Communication Technologies, Università della Svizzera Italiana, Lugano, Switzerland*
[2]*SASWEB Research Lab, Department of Mathematics and Computer Science, University of Udine, Udine, Italy*

SUMMARY

Tag-based systems are widely available, thanks to their intrinsic advantages, such as self-organization, currency, and ease of use. Although they represent a precious source of semantic metadata, their utility is still limited. The inherent lexical ambiguities of tags strongly affect the extraction of structured knowledge and the quality of tag-based recommendation systems. In this paper, we propose a methodology for the analysis of tag-based systems, addressing tag synonymy and homonymy at the same time in a holistic approach: in more detail, we exploit a tripartite graph to reduce the problem of synonyms and homonyms; we apply a customized version of Tag Context Similarity to detect them, overcoming the limitations of current similarity metrics; finally, we propose the application of an overlapping clustering algorithm to detect contexts and homonymies, then evaluate its performances, and introduce a methodology for the interpretation of its results. Copyright © 2012 John Wiley & Sons, Ltd.

Received 5 October 2011; Revised 22 April 2012; Accepted 23 July 2012

KEY WORDS: folksonomies; tag clustering; tag disambiguation; tag similarity

1. INTRODUCTION

A large and increasing number of social Web applications allows users to annotate their shared resources with *tags*. Tags are typically short text strings freely chosen by users [1]; they are democratic and bottom-up, flat (as opposed to hierarchical), inclusive (meaning that if some resource does not fit existing tags, it is sufficient to create a new one), current, and extremely easy to use [2–4].

The set of all the tags associated to a set of resources by a given user defines a personomy, whereas the union of all user personomies within a system defines its *folksonomy* [5]. Thus, a folksonomy is the result of a collaborative activity whose output is a user-generated vocabulary, a rich collection of metadata that has a huge potential for the semantic interpretation of data.

For *recommender systems* (RSs), folksonomies represent a valuable source of knowledge used for generating recommendations. RSs represent a relatively recent research area (it emerged as an independent area in the mid-1990s) that owns an active research community [6, 7]: they play an important role in highly rated Internet sites (e.g., YouTube); several conferences, workshops, and

*Correspondence to: Davide Eynard, Institute for Communication Technologies, Università della Svizzera Italiana, Via Buffi 13, 6904 Lugano, Switzerland.
†E-mail: davide.eynard@usi.ch
‡E-mail: mazzola.luca@gmail.com
§E-mail: antonina.dattolo@uniud.it

journal special issues (e.g., ACM RecSys[¶] or UMAP[‖] conference, or SASWeb workshops series,[**] ACM Transactions on Intelligent Systems and Technology, and so on) are devoted to them.

As a consequence, a correct and effective extraction of semantic knowledge from collections of tags, tagged resources, and tagging users can represent a significative step for research, if different problems that are due to the inherent *ambiguity in the semantics of terms* are appropriately faced.

Tags are liable for spelling (or simply typographical) errors or syntactic variations, but also many other ambiguities [8, 9] are due to the presence of synonyms (e.g. `game` and `juego`, or `web2.0` and `web_2`), homonyms (some of which may be *polysemous*, i.e., `check` as in 'to check' and as in chess, whereas others might be not, such as in `sf` for 'San Francisco' or 'Science Fiction'), and basic level variations (e.g., `dog` and `poodle`). Moreover, even when the meaning of a single word is well defined, the purpose of a tag still might vary from one tagging action to another: as an example, the tag `blog` could be applied to a blog service such as blogger.com, to a blog page, to a blog software, or to a piece of news that we want to blog later. These different types of ambiguity often coexist within the same tag set (i.e., `sf` has synonyms as `sanfrancisco` and `scifi`, and it can be considered as a homonym for these two different concepts).

The motivations to better define the semantics of these tags are different: the control of spelling errors or syntactic variations allows the system to cluster them in a single set and apply distance metrics – which would be otherwise distorted – on this new set, synonym detection increases recall in search and can be used to refine results in recommendation systems; finding homonyms also means finding different contexts of use of the same tag, thus increasing the precision of returned results; basic level variations identify a hierarchy within a tag set that can be exploited to improve search.

Because these issues are not independent from each other, a global solution that takes all of them into account is the most appropriate way to achieve good and reliable results. For this reason, with the aim of supporting RSs through tags [8] and aware of how deep the impact of tag ambiguities in folksonomies is within these systems [10], in this work we decided to suggest a 'holistic approach' to tag analysis: it does not focus on just one type of ambiguity but tackles different ones (namely synonymy and homonymy) at the same time.

The *main objective* of this work is to provide a deeper understanding about synonymy and homonymy in tag-based systems. Our approach follows these steps:

1. Building on the ideas presented in [11, 12], we represent a folksonomy using an edge-colored multigraph of users, tags, and resources; then we simplify it preserving some meaningful properties. The result is a weighted graph, which is our basis for applying analysis methodologies and metrics aimed at extracting semantic information from a folksonomy.
2. We propose a methodology for the analysis of these systems.
3. We evaluate existing techniques for the detection of synonyms, in particular comparing general approaches such as Tag Context Similarity (TCS) [13] with ad hoc ones.
4. We evaluate the application of clustering techniques for the detection of homonyms, identifying the relationships between groups of homonyms and synonyms.

From the technical point of view, our approach relies on a modular and extensible framework of analysis, allowing us to plug in new metrics or change some parameters to test if different setups produce better results. We developed a prototype with scripting languages (mainly Perl and Python) and easily available libraries for local and remote database access and math functions. Then we applied our experimentation on two datasets extracted from delicious. As a contribution to the state of the art, our experimental results show that our normalized variant of TCS can be actually used for synonym detection, but its behavior is strongly related to the particular tag in exam and some of its characteristics like popularity, language, or the fact of being composed of more than one word. Moreover, we propose the application of an overlapping clustering algorithm for the detection of

[¶]6th ACM Conference on Recommender Systems, Dublin, Ireland September 9–13, 2012. http://recsys.acm.org/2012/
[‖]20th conference on User Modeling, Adaptation, and Personalization (UMAP 2012) Montreal, July 16–20, 2012. http://umap2012.polymtl.ca/
[**]3th International Workshop on Semantic Adaptive Social Web – SASWEB 2012, Montreal July 16, 2012. http://sasweb.uniud.it/events/2012sasweb/

contexts and homonymies that is based on tag semantic relatedness. We evaluate its performances and propose a methodology for the interpretation of its results.

The paper is organized as follows: in Section 2, we describe related projects, open issues, and the position of this paper among other works at the state of the art. In Section 3, we introduce some basic definitions and show our process to create a simple graph starting from an edge-colored multigraph. In Section 4, our methodology is described, whereas Section 5 presents the details about our prototype. Section 6 reports on some experimental results, starting from the dataset we gathered up to some interesting conclusions based on the output produced by our analysis tool. Finally, Section 7 presents some conclusions and future outlooks.

2. OPEN ISSUES AND RELATED WORK

Folksonomies have a huge potential as tools for the semantic interpretation of data, but they are strongly limited by the lack, in current tagging systems, of adequate methodologies and tools for dealing with semantic ambiguities of tags such as synonymy and homonymy.

To overcome these limitations and produce a semantic contextualization of social tags [14], we need to address three main issues: (i) how to represent a folksonomy; (ii) how to compute the semantic distance among tags; (iii) how to cluster tags to identify their different contexts of usage. These issues will be discussed in the following subsections.

2.1. Methodological issues

A folksonomy is defined on a ternary relation that maps the tagging activities of all users: this relation keeps information about which tag has been applied by which user on which resource. This is the starting point to model knowledge, relationships, and similarities in a folksonomy. However, mining similarities is not trivial because the ternary relation merges relations among objects of the same type as well as among objects of different types. The efforts to overcome these limitations can be roughly categorized in two main groups: the former aims at the development of theoretical models, such as three-order tensors or tripartite graphs [15, 16]; the latter aims at ad hoc solutions, trying to solve a single issue (or a family of issues) in the most effective and convenient way (see, for instance, the folksonomy ontology enrichment system (FLOR) [17]). The former approach has the advantage of allowing researchers to better understand how folksonomies work and provides a common ground to describe their inner details. The latter, instead, offers a pragmatic and practical way to overcome specific problems.

In this second category, some authors focus on the disambiguation of tags with mutual contextualization [18], clusterization [19], and semantic grounding [13]. In particular, this last work introduces the concept of TCS, computed as a cosine similarity in a vector space $\mathbb{R}^T$, where each vector $v_t \in \mathbb{R}^T$ is defined by $v_{tt'} := w(t,t')$ for $t \neq t' \in T$, where $w(t,t')$ is the number of resources that have been tagged with both t and t', and $v_{tt} := 0$. This measure is central for our work, as starting from it we developed our own version of TCS that stands at the basis of our analyses.

Other algorithms are aimed at automatically restructuring folksonomies [20–22]. A more extensive work [1] presents a list of common issues in tag-based systems related to their intrinsic ambiguity. Other more general approaches exist: in [23], the authors present a methodology for extracting new semantic tags for a resource, using a domain ontology; in [24], the authors use multiple dimensions to enrich tag information for final user support.

Even if practical approaches may appear more efficient in attacking specific problems, some issues (such as homonymy, synonymy, term variations, and spelling errors) require an integrated approach. For this reason, our work builds both on general theories and ad hoc solutions, proposing their integration in a unifying framework that is grounded on a model that builds upon the edge-colored multigraph of users, tags, and resources and adapts it for specific purposes.

2.2. Ambiguity issues

The semantic contextualization of social tags is a current open challenge. Synonyms (at various levels), homonyms, spelling errors, or syntactic variations should always be considered in their co-existence. On the one hand, the absence of synonym control can alter the evaluation of similarity values, and tags (e.g., `web2-0` and `social-web`) could erroneously assume a low neighborhood value, compromising a correct semantic interpretation or recommendation. On the other hand, a homonym (e.g., `sf` can be used to refer both to science fiction and San Francisco) is a word that can be given different interpretations; without knowing its synonyms (i.e., words that have the same meaning, but that only exist in one of its contexts of use), matching it with a well-defined semantic context becomes a complex operation.

Two words in tag-based systems are considered *synonyms* if – similarly to natural text – they can be replaced by each other without affecting the meaning of a 'sentence' (in our case, a tagging action). What is different from natural language (where words occur in the context of a sentence and they are constrained to follow given rules) is that different terms can be variations of the same one (as in `blog`, `blogs`, and `blogging`), translations into other languages, acronyms, and sets of terms joined by nonalphabetic characters. Moreover, the same tag (e.g., `Web`), with just a single meaning, can be used to annotate items about distinct topics (e.g. `Web development`, `Web browsers`, and `Web 2.0`) [14].

The consequence of this heterogeneity is that there is no 'one size fits all' solution to the problem of detecting whether two words are synonyms. *Spelling errors* and *syntactic variations* between tags have been studied in literature: for spelling errors, encouraging results have been obtained using the Levenshtein and Hamming distances, whereas in situations of insertion or deletion of characters, the Levenshtein distance performs better than the Hamming distance, even though problems remain for short tags. In [25], authors suggest a solution to this problem, combining Levenshtein distance and cosine similarity and assigning them a different weight according to tag length. Other works [26, 27] use this distance for large tags. To treat synonymies, we propose a combined approach, which is based on the different methods described in Section 4.2.

Checking for homonyms basically means verifying if the same tag has been used in different contexts. Typically, usage contexts for a tag are computed by clustering tags related to it in groups, according to a chosen measure of similarity (a comprehensive survey on neighborhood-based metrics is provided in [28]). Then the tags most frequently occurring in these groups are used as a way to name and disambiguate the different contexts of use (see [19]). An interesting graph clustering technique is the one proposed in [29], which automatically finds an optimal number of clusters, as opposed to standard hierarchical or partitional clustering strategies, which require the definition of a criterion to stop the clustering processes. The cited technique is applied also in [14], where the authors obtain a set of contexts for each tag; they also use sophisticated tag similarity measures like the ones presented in [30]. Finally, the work described in [31] is based on word-of-mouth simulation: here, authors introduce the idea of contextual collaborative filtering, where tags detect a context in neighbors' collections.

The clustering approach is not the only one that has been applied in this field: there is a class of systems [10, 27, 32] that use existing formal knowledge bases to infer the meaning of a tag; these systems are based on the association of tags to terms belonging to lexical databases (e.g., Wordnet). The use of linguistics and lexical resources helps in solving ambiguities but it can generate problems related to the use of natural language [8].

In our work, we perform context detection (for homonyms but also for tags that are characterized in general by different contexts of usage) by means of clustering. The algorithm we apply is the one introduced in [33] and is described in detail in Section 4.3.

3. REPRESENTING SEMANTIC DISTANCES BETWEEN TAGS

In this section, we introduce our definition/representation of folksonomy. It simplifies the traditional, largely used notion of tripartite graph, proposed in [15, 16], in favor of a more compact representation. Starting from this definition, our aim is to apply a set of transformations and obtain a new simplified graph, which we can use to compute a measure of tag relatedness.

Because a folksonomy is the union of one or more personomies, we start by introducing our definition of personomy.

Definition 1
A *personomy* P_u of a user u is an undirect edge-colored graph of color C_u. It is a five-tuple $P_u = \{T_u, R_u, E_u, C_u, c\}$ such that

- T_u is the finite set of tags used by u;
- R_u is the finite set of resources tagged by u;
- T_u and R_u are disjoint;
- E_u is the set of edges $(x, y) : x \in T_u \wedge y \in R_u$;
- C_u contains the color of each edges in P_u, mapped by the assignment $c : E_u \rightarrow C_u$

Definition 2
Given a set of users U and the family of personomies P_u $(u \in U)$, a *folksonomy* is

$$F = \bigcup_{u \in U} P_u$$

Remember now the formal definition of an edge-colored multigraph (informally, a graph where each couple of vertices may be connected by more than one edge).

Definition 3
An *edge-colored multigraph* is a triple $ECMG = (MG, C, c)$ where $MG = (V, E, f)$ is a multigraph composed of a set of vertices V, a set of edges E, and a function $f : E \rightarrow \{\{u, v\} | u, v \in V, u \neq v\}$. C is a set of colors, and $c : E \rightarrow C$ is an assignment of colors to edges of the multigraph.

We observe that a folksonomy may be just defined in terms of an edge-colored multigraph in the following way.

Definition 4
A *folksonomy* is an edge-colored multigraph described by

$$F = ((T \cup R, E, f), C, c)$$

where

- T, R, E, and C are respectively the union of T_u, R_u, E_u, and C_u, $\forall u \in U$ (see *Definition 1*);
- $c : E \rightarrow$ C is the assignment of colors to the edges.

Figure 1(a) shows a folksonomy example. The set C is composed of three users, identified by three colors (also identified by different graphical notations: normal, double, and dotted lines). The set R is composed of three resources (r_1, r_2, r_3), identified by three solid, round disks, whereas the set T is composed of 17 tags identified by empty circles, and labeled as t_i, with $i = 1, \ldots, 17$. Finally, E contains unordered couple of vertices in R and in T.

This view of a folksonomy as a collection of personomies includes all the information normally available in a social tagging system. However, depending on the analyses that we perform in our approach, a simplified, even though less informative, graph is sufficient.

Figure 2 depicts different possible simplifications of the tripartite graph. The initial space $I = (R, U, T)$ can be reduced with an injective projection through distributional aggregation in three bidimensional spaces (U, T), (R, U), and (R, T). The three spaces can be associated respectively to users' vocabularies (U, T), users' resources (R, U), and resource metadata (R, T).

The following reduction consists in transforming the space to highlight relationships between homologous objects ($T \times T$, $U \times U$, and $R \times R$). Note that the same space (e.g., $T \times T$) might have different interpretations according to the path that has been followed to build it. For instance, tags could be related because they have been applied to the same resource or because they appear in the same user vocabulary. The different outcomes of these transformations, together with their interpretations, follow:

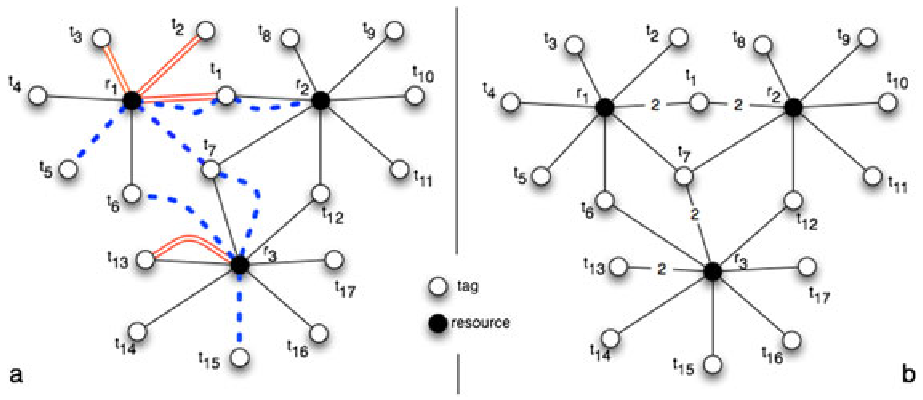

Figure 1. (a) A folksonomy as a collection of three personomies, identified by normal, double, and dotted lines. (b) A bigraph as a weighted folksonomy: weights on edges represent the number of users using a specific tag for a resource.

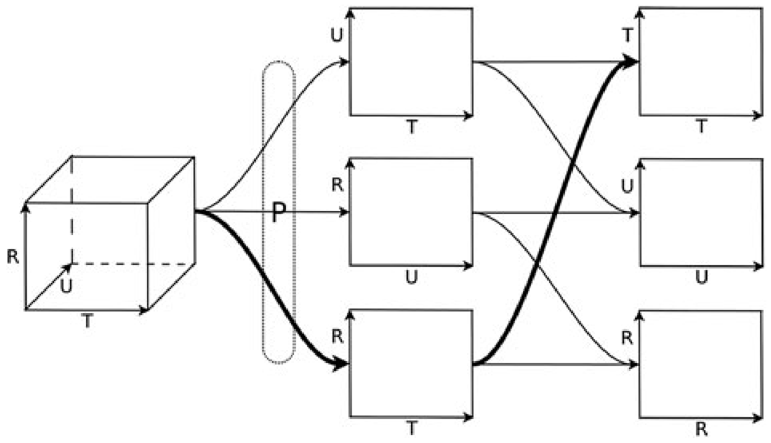

Figure 2. Map of the possible simplifications that can be applied to the tripartite graph (R = resources, T = tags, U = users). The process we followed in this work is shown in bold.

- $(R, U, T) \rightarrow (U, T) \rightarrow (T, T)$: Tag relatedness based on user vocabularies
- $(R, U, T) \rightarrow (U, T) \rightarrow (U, U)$: User relatedness based on tag usage
- $(R, U, T) \rightarrow (R, U) \rightarrow (U, U)$: User relatedness based on their annotated resources
- $(R, U, T) \rightarrow (R, U) \rightarrow (R, R)$: Resource relatedness based on user collections
- $(R, U, T) \rightarrow (R, T) \rightarrow (T, T)$: Tag relatedness based on annotated resources (*)
- $(R, U, T) \rightarrow (R, T) \rightarrow (R, R)$: Resource relatedness based on their metadata

The aim of this work is to obtain a measure of tag relatedness based on the number of resources where tags co-occur. For this reason, we chose one particular simplification process, that is, the one that follows the bold arrows path in Figure 2 and whose outcome is the $(T \times T)$ space marked with (*) in the aforementioned list. During this process, information about users is aggregated, and it is not possible to easily reconstruct it. This, however, does not affect our analysis, as we only focus on common resources. The calculation of the other type of tag relatedness, based on shared user vocabularies, could also provide interesting results and is going to be addressed in our future work.

The first step in our process is to apply an injective transformation from the graph depicted in Figure 1(a) to (b). This transformation is performed by grouping edges that connect the same vertices and consequently losing information about who applied a tag to a resource. The final weights to be assigned to the aggregated edges are calculated according to the following general formula:

$$w(r_i, t_j) = \sum_{u \in U} w_u(r_i, t_j) \qquad (r_i, t_j) \in E$$

This allows us to assign different weights according to which users have performed the tagging (i.e., the weight could be proportional to the user rank within a system, set higher for a community that is object of study, or zero for users identified as spammers). Our current choice is to assign the uniform weight 1 to all the users, reducing the calculation to the number of times a tag has been used on a resource (and, as a user can only use the same tag once for each resource, this also matches the number of users applying a specific tag to a resource). This new graph is useful to have a compact view of the relationships between resources and tags. As it can be noted by looking at Figure 1(b), it is also easy to identify which node $t \in T$ is more central and acts as a bridge between the different resources $r \in R$ (in our example, the tags t_1, t_6, t_7, and t_{12} are the most central).

To further simplify the graph, we can now collapse the resources, creating a link among each pair of tags that are connected to the same resource. The resulting graph is shown in Figure 3(a).

This kind of representation shows edges between tags if they have been used on the same resource, but it does not provide any information about the strength of these bonds. In this case, the weights that have to be assigned to the new edges are not a function of the previous weights anymore (or, at least, not a function of them alone). Different approaches have been considered trying to provide meaningful values for these weights. For instance,

- The number of triples (t_i, r, t_j) where (t_i, r), $(r, t_j) \in E$ represents the number of co-occurrences across all the resources (i.e., the number of resources on which the tags co-occur): this is also what is commonly defined as *co-occurrence*. The main advantage of this metric is that it is simple to calculate and the results are meaningful. Its main limit, instead, is that it is biased towards very popular tags, as they frequently co-occur with many others.
- A *normalized co-occurrence* (NCO) takes into account not only the number of resources on which two tags co-occur but also how common these tags are within the system. An example is the *Jaccard index*, defined as follows:

  $$NCO = \frac{|A \cap B|}{|A \cup B|}$$

  where A is the set of documents tagged with t_a and B is the set of documents tagged with t_b. This metric is more complex, but it also returns much better results: tags that are less popular but still shared within subcommunities tend to be ranked higher, exposing the vocabularies typical of their domains of interest.

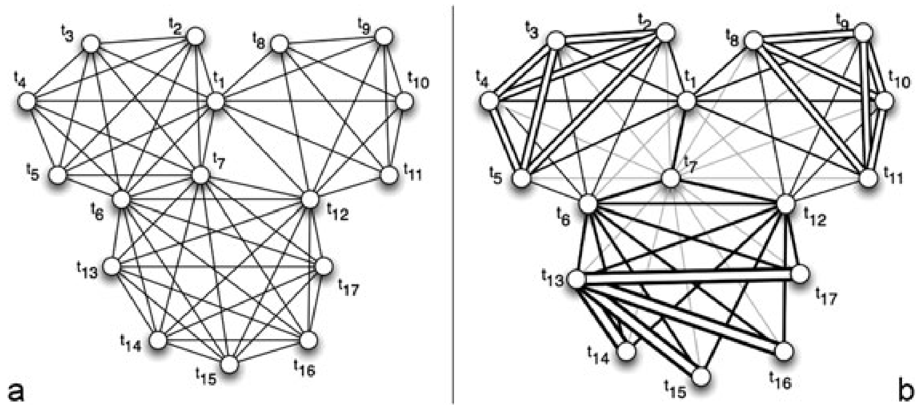

Figure 3. (a) A graph of tagging relationships. (b) An alternative view of the tagging relationship, with Tag Context Similarity measure applied on the edges.

- More advanced metrics [13] take into account other parameters: for instance, distributional measures are based on vector space representations of tags, requiring information related to other tags, resources, or users. The limits of this approach are the same of many vector-based representations (i.e., high dimensionality, small distances between vectors, higher computational complexity). The results, however, are interesting as they tend to represent more 'semantically similar' terms that are related not because they appear *together* with a given tag but rather in the same *contexts*.

In this work, to express the strength of relationships between tags, we calculate the weight of each edge between tags with a variant of TCS [13] in the following way: every tag is described as a vector in a multidimensional space $\mathbb{R}^T$, where T is a chosen vocabulary of tags used as descriptors. For each tag t, the matching vector $v_t \in \mathbb{R}^T$ is defined by $v_{tt'} := d(t, t')$ for $t \neq t' \in T$, where d is the normalized co-occurrence of t and t' calculated using the Jaccard index, and $v_{tt} := 0$. Then TCS between two tags is calculated as the cosine distance between their two respective vectors, that is,

$$TCS(t_1, t_2) := cos\measuredangle(v_1, v_2) = \frac{v1 \cdot v2}{\|v_1\| \|v_2\|}$$

The result, applied on the graph depicted in Figure 1(b), is shown in Figure 3(b), where the different weights are visualized using different line styles. In particular, gray line represents weights included in the interval $(0.52, 0.55)$, thin normal line in $(0.58, 0.62)$, thick normal line in $(0.65, 0.68)$, thin double line the value 0.72, and finally thick double line the value 0.78. Some other links between the tags with associated minor weights (with values included in the interval $(0.08, 0.17)$ are present and considered in our model, but for readability, they are not shown in Figure 3(b). This type of graph, containing the semantic distance between tags, is the starting point of our clustering methodology.

Some considerations on this first phase are appropriate: the process of *aggregation* reduces the dimensionality of the $(user, resource, tag)$ space to the bidimensional space $(resource, tag)$. In literature, four typologies of aggregation are identified, described in detail in [30]: projection, distributional, macro-aggregation, and collaborative aggregations. In our approach, we applied a distributional aggregation. This choice simplifies our graph representation (see Figure 1) and preserves necessary (from our model's perspective) semantic information. Then the computation of similarity measures transforms the $(resource, tag)$ space in a new (tag, tag) space. An analytical study of the most frequently applied similarity measures has been discussed in [28, 30], which describe matching, overlapping, Jaccard, dice, cosine, and mutual information similarities.

In our approach, we used TCS to calculate the relatedness between two tags. The modification we applied to the original algorithm is that instead of describing tag vectors in terms of simple co-occurrences, we normalized them using the Jaccard index. This allows us to give less importance to inflated terms and boost those that are at the same time frequent but not misused.

4. A HOLISTIC APPROACH FOR EXTRACTING SYNONYMIES AND HOMONYMIES

Our approach starts from the observation that although the study of tag synonyms and homonyms is interesting for different application fields, only few approaches address both of the problems at the same time.

Our *trait d'union* between the study of synonyms and homonyms is the concept of *related tag*. As shown in [13], measures of tag relatedness can be defined in different ways. Some of them rely on statistical information about tag co-occurrence, whereas other approaches exploit the *distributional hypothesis* [34], which states that words found in similar contexts tend to be semantically similar. Co-occurrence measures address the so-called syntagmatic relation, where words are considered related if they occur in the same part of text. Contextual measures, instead, address the paradigmatic relation (originally called associative relation by Saussure), where words are considered related if they can replace one another without affecting the structure of the sentence.

In our work, we use a normalized version of TCS (see previous section) to find collections of tags related to a given tag t. We then explore these collections for synonyms and homonyms, providing insights about their relationships and the quality of the applied algorithms. To accomplish our goal, we follow a process (see Figure 4) whose input is a folksonomy as defined in Section 3 and which is composed of the following steps: (i) problem reduction, aimed at simplifying our analyses and calculations, (ii) synonym analysis, aimed at detecting and analyzing the synonyms of a given tag, and (iii) homonym analysis, aimed at detecting tag homonyms in terms of their (possibly partially overlapping) contexts of usage. These steps are detailed in the following subsections.

4.1. Problem reduction

Figure 5 shows the monthly growth of users, tags, and resources according to one of the datasets we performed our calculations on. Considering the growth of the system and the positive correlation between the number of tags and users [35], it is easy to understand that typical comparison algorithms do not scale well on the original graph. For this reason, we decided to operate simplifications on different levels.

The first level is the one related to the tripartite graph itself. As discussed in Section 3, folksonomies can be represented at different levels of expressiveness. For our studies, we have decided to rely on a simplified version of the graph (like the one shown in Figure 3(b)), where nodes represent the tags belonging to our dataset and edges represent their degree of relatedness according to the TCS measure. This measure represents tags as vectors of co-occurring tags; thus, dimensionality is another size factor we needed to trim to obtain satisfying results in a reasonable time. For the analyses we needed to perform, we found that the best trade-off between time and quality was to describe every tag as a co-occurrence vector of the 10,000 most frequent tags.

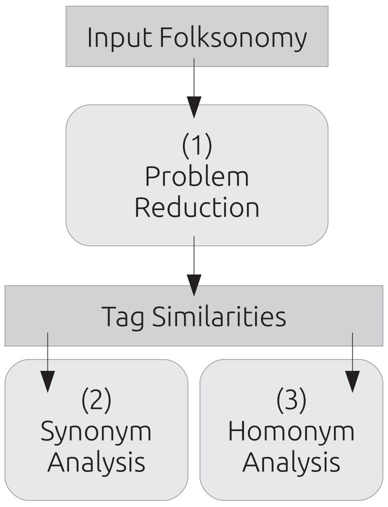

Figure 4. An outline of our analysis process.

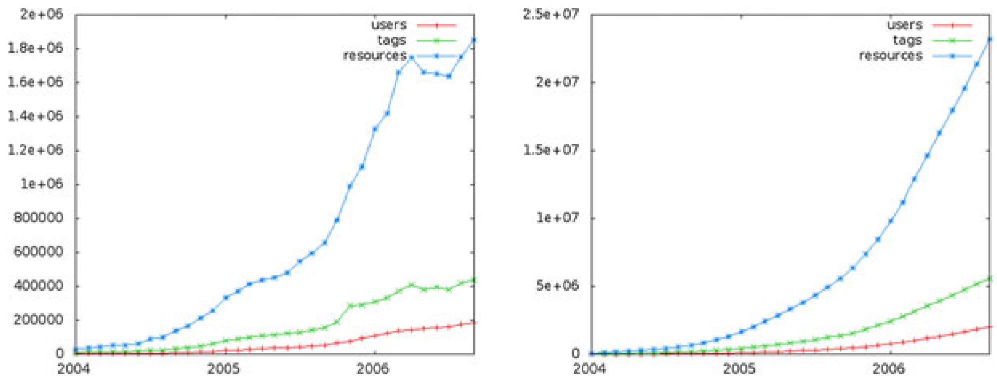

Figure 5. The growth (left: monthly, right: cumulative) of DS2 in terms of users, tags, and resources.

Another simplification we apply follows the obvious assumption that ambiguous tags (i.e., synonyms and homonyms) should at least be related, either by co-occurrence or by presence in the same contexts. For this reason, we are able to reduce the computational expense of our algorithms (in particular, the one that aims to find different contexts for homonym tags) by restricting our analyses to the top n tags that are related to a given one (with values for n that typically range from 50 to 100).

As a result, the output of the problem reduction step is a simplified version of the folksonomy, taking into account only tags and their similarities (calculated over tag vectors of size 10,000), and for each tag t, a view over the top n tags related to t. This information is then passed to the following steps of the process, namely synonym and homonym analyses.

4.2. Synonym Analysis

Cattuto *et al.* [13] suggest TCS as one of the first measures (together with Resource Context Similarity) to choose when trying to discover synonyms. This approach is independent from the specific types of variations a tag might be subject to and competes with ad hoc heuristics like the following ones:

- *Edit distance* such as Levenshtein distance, which calculates differences between two strings in terms of insertion, deletion, and substitution of characters. This would return high similarities for variations of the same word obtained by inserting extra characters to simulate spacing (as in `web20`, `web2.0`, and `web_2_0`) but might return wrong results especially when short strings are involved (to solve this problem, a Levenshtein distance normalized with respect to string length could be used, maybe in a mixed variant like the one proposed in [25]).
- Synonym search within *WordNet*. Results are good in terms of precision; however, the high-quality, top-down thesaurus provided by WordNet only covers a small part of the bottom-up vocabulary built with a folksonomy, which often contains typos, slang, acronyms, and so on. Although this is not a limitation for different applications [13, 36], it is when trying to perform synonym detection over sets that comprise more than few top tags.
- *Online translation* tools. Online services can be used to obtain the translation of a tag in different languages. As fixed vocabularies suffer the same problem of WordNet, collaboratively grown corpora can be chosen instead to obtain translations for recent terms. As an example, Wikipedia articles usually contain links to their translated versions. For every link, both the translated word and the language code for that translation are published in a semistructured way and can be automatically extracted to provide translations for a given term [37].
- *Stemming*. Natural Language Processing can be used to detect if two tags share the same stem, defined as the part of a word that is common to all its inflected variants. If so, then there is a higher probability that these tags are synonyms. Moreover, working on stems allow to merge logical variants of the same word (i.e., nouns, adjectives, and verbs) that are often used as interchangeable tags.

For this reason, we analyze synonymies applying a combined approach: in particular, we evaluate a ‘universal’ method like TCS (which is agnostic with respect to the types of syntactic variations a tag could be subject to) by calculating how many synonyms it can find that could have also been found using, for instance, Levenshtein distance, synonym check on Wordnet, and term translation with Wikipedia. Moreover, we evaluate TCS behavior along five different families of tags showing that the quality of this metric strongly depends on the type of tags that are taken into consideration.

Tag Context Similarity looks by far as the simplest way to find synonyms for a given tag. It is general and does not require external knowledge to work. However, how better is the use of this heuristic if compared with more specific ones? Using this framework and two delicious datasets, we evaluate TCS against other approaches in Section 6.

4.3. Homonym Analysis

Homonym detection derives from the definition of semantic contexts, which are typically regarded as clusters of tags within a given dataset. Clustering algorithms vary in performance and in the type of discovered clusters, but to be suitable for our analyses, an algorithm should at least satisfy the following constraints:

- As the input data are a graph with tags as nodes and edges weighted according to their relatedness, the algorithm should be able to work on the concepts of similarity/dissimilarity rather than the ones of closeness/distance.
- As the same tags could be used in different contexts, the algorithm should be able to find overlapping clusters instead of exclusive ones.

The algorithm we have chosen for our first prototype has been described in [33]. It is an algorithm typically used for the analysis of social networks and the discovery of community structure within them. The basic idea is that a *natural community* of a node a belonging to the network is a subgraph G identified by the maximization of a property of fitness of its nodes. The fitness of G is calculated as follows:

$$f_G = \frac{s_{\text{in}}^G}{\left(s_{\text{in}}^G + s_{\text{out}}^G\right)^\alpha}$$

where s_{in}^G and s_{out}^G are, respectively, the *strength* of the internal links (i.e., double the sum of the weights of all the edges that link the nodes in G with each other) and the strength of the external ones (i.e., the sum of the weights of all the edges linking nodes in the group with nodes not belonging to it). Finding a (local) maximum for f_G means finding a subgraph such that the inclusion of a new node or the elimination of one node from the subgraph would lower f_G. This is obtained through an iterative process, which starts from a subgraph G containing only node a and proceeds as follows:

1. For every node b, neighbor of G and not included in it, calculate a fitness function

$$f_G^b = f_{G+\{b\}} - f_{G-\{b\}}$$

2. The neighboring node with the highest fitness is added to G.
3. The fitness of each node in G is recalculated.
4. If a node has negative fitness, it is removed from G.
5. If 4 occurs, repeat from 3, otherwise repeat from 1.

The process stops when, at step 1, all the examined nodes have negative fitness. After a natural community has been found, the process can be started again with a different seed node c – not yet assigned to any group – to provide a new, possibly overlapping, cluster. This can be repeated until there are no more free nodes, thus providing a *cover* for the graph in exam.

Clustering quality strongly depends on the value chosen for the parameter α, which can be used to tweak the groups' sizes. Large values of α yield many small groups, whereas small ones bring fewer and larger clusters (until they reach the global maximum of one cluster containing all the nodes).

5. PROTOTYPE IMPLEMENTATION

To perform our experiments, we have designed and developed a prototype that implements the process we have described in Section 4. The prototype has been developed with scripting languages (Perl and Python), and it relies on easily available libraries for local and remote database access (i.e., DBI and DBD::mysql for Perl) and math functions (i.e., NumPy and SciPy for Python; see http://www.scipy.org/). The DBMS we used was MySQL 5, running on an Ubuntu Linux system (version 11.04). Our prototype source code has been made available online at http://davide.eynard.it/src/SPE11/.

The architecture of our framework is shown in Figure 6. Inspired by the approach described in [27], we decided to make the system modular to perform different kinds of analyses on the tag corpus: new algorithms should be easily pluggable into the system, but at the same time, it should be possible to group common operations together in the same modules. A common data repository guarantees interoperability between the different components. The four main components

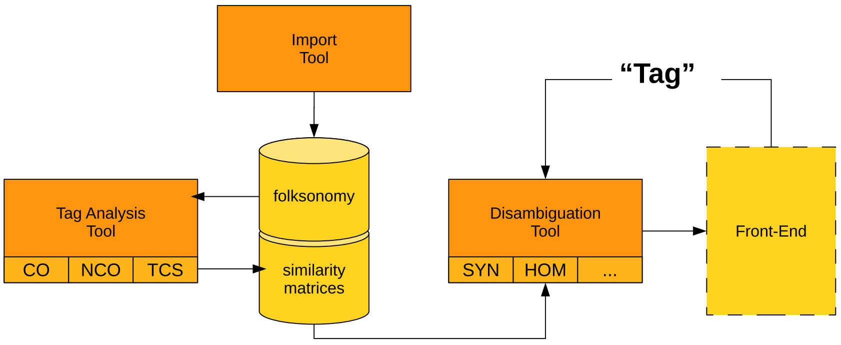

Figure 6. The architecture of our system.

are the Import Tool, the Tag Analysis Tool, the Disambiguation Tool, and the Front-End. For each component, different modules (currently developed as independent scripts sharing the same data) can be made available.

To analyze a folksonomy, our system requires it to be saved inside a database. As datasets are often provided in different formats such as CSV or TSV files, the *Import Tool* can be used to import one of these datasets in a database, following a very simple schema organized in four tables: one table each for users, tags, and resources, and one for tagging actions (i.e., user-tag-resource triples, together with the date when the tagging occurred).

The *Tag Analysis Tool* accesses a folksonomy (that now we can assume stored inside a database), calculates relatedness between tags according to chosen metrics, and finally outputs its results in different matrices. These matrices can be saved as tables in a DB or as CSV files ready to be imported by other tools. We have made the Tag Analysis Tool independent from a database schema: queries are provided as a parameter, and they can be changed to support other schemas. The metrics that we have currently implemented are co-occurrence, normalized co-occurrence according to the Jaccard index, and our implementation of TCS. As co-occurrence measures rely on information found in the table describing taggings (which usually grows up to tens of millions of rows), calculating them typically takes a long time. To make the problem tractable in a reasonable amount of time, we developed an algorithm that avoids the bottleneck due to disk access by allocating big data structures (similarity matrices) directly in memory. The algorithm works as follows:

- The database is queried for all the different (resourceId,tagId) pairs grouped and ordered by resource. As a consequence, all the taggings on the same resource are grouped in a single 'chunk' of results.
- Query results are scanned linearly, saving (unique) tags related to a given resource in a buffer. Whenever a new resource is found, all the tags that have been saved are processed as follows: for every pair of tag IDs (t_1, t_2) such that $t_1 < t_2$ (so to avoid considering the same pair twice), the matching cooccurrence value is incremented by 1.
- At the end of the scan, the cooccurrence matrix is saved onto the database.

The algorithm performed well during our experiments, calculating co-occurrence and normalized co-occurrence from a table of 127M rows in about 5 h on a laptop with a 2.80 GHz Intel Core2 Duo P9700 and 8 GB RAM (which, by the way, was more than double the actual RAM needed by the analysis). The output was a table containing similarity measures for the top 10,000 tags by frequency (see Section 6 for an explanation of this choice and more details), for a total of 100M rows.

The *Disambiguation Tool* is the core component of our system: it takes as inputs a tag t, the number of top-related tags n to take into account, and the similarity matrices generated by the Tag Analysis tool, and provides new information about t and the set of its related tags. This information is generated by the different disambiguation modules that are available in the framework. For the current experiments, we chose to analyze synonyms, identified with the TCS measure, and homonyms, identified as clusters within the set of top-related tags of a given tag.

The homonyms module implements the algorithm presented in [33] and described in the previous section. Given a tag name and the parameter α as inputs, it returns a set of (possibly overlapping) clusters that match different contexts of usage of the provided tag. To limit the analysis to only related tags and thus reduce the cost of the clustering algorithm (whose complexity, in the worst case, is $n^2 log n$), only the subgraph whose nodes match the top n tags related to t is taken into account.

As the output of disambiguation plugins is heterogeneous, the system also needs a component that is able to manage it, showing data to users in a meaningful way or exporting it in a format that is suitable for other applications. This is what the *Front-End* component is in charge of, that is, basically managing all the system inputs and outputs. Our front-end is currently a set of text-only applications that run on a UNIX shell, allowing users to specify different parameters (i.e., tag to analyze, size of the related tag set, number of top tags to take into account) and returning results in different formats.

6. EXPERIMENTAL RESULTS

To perform our experiments, we have first set up two datasets to analyze. Both of them contain data from delicious (www.delicious.com). The former has been created by collecting data from more than 30,000 users with a Web scraper in March 2006,[††] whereas the latter, kindly shared by the DAI-Labor at Technische Universität Berlin [35] as a collection of TSV files, is much larger and covers data from September 2003 to September 2006. Despite being more expensive in terms of calculations, this larger dataset has proved very useful both to test the efficiency of our algorithms and to verify some assumptions we had made on the smaller one.

In a second phase, we have used our prototype to import data and run similarity computations. We have then performed synonym and homonym analyses and collected experimental results to find answers to the following questions:

- Preliminary data analysis: how much do dataset size and currency matter for the following analyses?
- Synonym analysis: how good is TCS as a way to find tag synonyms? Does its quality depend on the chosen tag or not?
- Homonym analysis: how does the chosen clustering algorithm perform? How are tag homonyms and synonyms related?

The details of our experiments and our answers to these questions are presented in the following paragraphs.

6.1. *Preliminar data analysis*

Before the actual analysis, we evaluated the coverage of our delicious datasets in terms of number of tags, resources, and tagging events.

The first dataset, called DS1, has been created by downloading contents from more than 30,000 user accounts and resulted in more than 480,000 tags, 3.7 million resources, and 21 million tag assignments, that is, $< user, tag, resource >$ triples (see the left *Total* column in Table I). We decided to exclude from this dataset the tag `system:unfiled` that is generally used for bookmarks that have been added to the system but have not been tagged.[‡‡]

The second dataset (DS2) is a subset of DAI-Labor's delicious dump. It consists of data about more than 350,000 users, 2.2 million tags, 15.7 million resources, and more than 126 million assignments (see the right *Total* column in Table I). The subset covers 3 years of activity on delicious, that is, from September 2003 to September 2006, whereas the original dump extends to December 2007. We only imported this relatively limited amount of data (less than one-third of available information) because of hardware and time limitations: the full dataset was too big to

[††]The scraper source code is available at http://davide.eynard.it/?p=28. Scraped data are available upon request to the authors.

[‡‡]`system:unfiled` is the most frequent tag in delicious: almost 36 million occurrences, that is about the triple of `design`, which is the second most frequent tag.

Table I. Coverage related to the top 10,000 frequent tags on the elements in the two databases.

	DS1			DS2		
	Total	Covered	Percentage	Total	Covered	Percentage
Users	30,948	1,070	03.46%	357,553	34,885	09.76%
Tags	482,465	10,000	02.07%	2,230,059	10,000	00.45%
Resources	3,744,679	1,983,455	52.97%	15,726,938	7,239,436	46.03%
Assignments	21,698,526	18,616,109	85.79%	126,973,740	106,603,680	83.96%

import in a reasonable time with the hardware we had available. Even in this limited version, the size of the tagging table alone amounts at 10.8 GB, against about 1.7 GB for the same table in DS1, and the size of the complete database (before any data are added by calculations) is 13.6 GB. As the dataset did not include tags like `system:unfiled`, we did not perform additional data filtering steps despite dataset creators warned about the possible presence of spam.

To calculate TCS, we represented tags as vectors of co-occurring tags. The size of the vector is one more parameter to the system: the bigger the size, the higher the expressivity of the model, thus the quality of the results. Of course, as the distance matrices we calculate grow quadratically, increasing the vector size 10-fold means a 100-times higher cost in terms of computation and storage space. Our choice fell on the top 10,000 most frequent tags set (Top10k), which provided experimentally good results but still could be analyzed in a reasonable time.

To verify the validity of this approximation for our objectives, we calculated the percentage of coverage offered by using only Top10k instead of the full set of tags. Results (see Table I) for both the datasets seem to justify our approach: despite the low percentage of tags, which is fixed by construction, close to 85% of all tagging actions – the ones we rely on for the calculation of tag co-occurrencies and similarities – involve one of the Top10k tags calculated for that given dataset. Moreover, despite a very low percentage of users who exclusively adopt these tags (this can be understood as almost everyone also adopts some personal, less popular tags), the ratio of resources tagged only with Top10k tags is curiously high, reaching about half of the available ones.

One of the main questions we wanted to answer was to what extent the size or the currency of a dataset could affect the results of our calculations. To do this, we compared the two datasets in terms of their vocabulary (i.e., in our context, their Top10k tag set) and of the results of the synonym analysis tool.

The two vocabularies of DS1 and DS2 overlap for about 80% of their tags. This was somehow expected, as popular tags are so much more widespread than others that it is very difficult to replace one of them in the top list. In fact, limiting the comparison to only the top 100 tags yields to an overlap of about 90%, that decreases the more we add less and less popular tags. How does this impact synonym analysis results? After manually checking 50 randomly chosen tags from Top10k (see Section 6.2), we found that related tags results present on average a 67% overlap, which raises to 84% if we only consider the tags related to the 10 most popular ones and decreases when we extend the analysis to the less frequent ones. Moreover, our analysis (see Figure 8) shows that on average, the synonym analysis tool has automatically found more synonyms in DS2 than in DS1.

These statistics also suggested us a generalization that might be interesting for those who study the evolution in time of this kind of systems. We decided to run the same kind of comparisons (Top10k overlap, related tags overlap) on different snapshots of the same dataset taken in subsequent moments. We took into consideration DS2 and performed our calculations at different moments in time, that is, at end of years 2003 to 2006. We then compared, between each pair of consecutive years, the overlaps of Top10k tags and, for the ones that appeared in both sets, the percentage of overlap between all of their related tags. Figure 7 shows how the vocabulary in DS2 converges to a more and more stable set of popular tags: every year the curve is higher (i.e., overlaps with the previous year are bigger) and less steep (i.e., even the less popular tags are becoming more and more common between two consecutive years). The second analysis we performed showed us that the relatedness measure we apply converges more quickly than the vocabularies themselves:

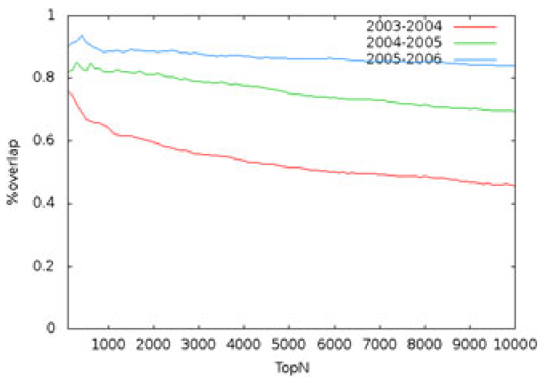

Figure 7. Overlap for tag vocabularies: comparison by year.

the average overlap between related tag sets is 11% in the period 2003–2004, 35% in 2004–2005, and 58% in 2005–2006. This suggests that the more time passes by, the more tags tend to be used in a consistent way and that despite changes in the vocabulary used to describe tags in the vector space model, TCS measure tends to return the same tags as top results.

6.2. *Synonyms analysis*

Experiments concerning synonyms analysis were aimed at understanding the performances of TCS measure as an automatic approach for suggesting synonyms. We tested the system against 50 tags, divided in five different groups:

- Group 1: 10 tags chosen randomly from the top 100 tags.
- Group 2: 10 tags chosen randomly from the top 1000 tags (excluding the top 100).
- Group 3: 10 tags chosen randomly from the top 10,000 tags (excluding the top 1000).
- Group 4: 10 random tags that are related to tags in Group 1 (i.e., a set was built as the union of the top 50 related tags for each tag in Group 1, then 10 tags were randomly chosen from that set).
- Group 5: the same as mentioned but exploiting relatedness with tags belonging to sets 2 and 3.

The list of chosen tags is shown in Table II. The reason why we chose these groups is that on the one hand, we wanted to verify whether there was some relationship between tag popularity and the number of synonyms returned by the algorithm and that on the other hand, we wanted to compare the results found with popular tags with some of their less popular equivalents.

For each tag, we calculated the top n (with $n = 50$) related tags according to TCS and performed a manual synonym analysis on the results.

Three experts independently tagged the automatically generated tags as synonyms of the given tag, adopting the following guidelines. A tag could be considered a synonym if

- It has the same stem, like in the case of singular/plural or name/verb/adjective (i.e., `blog`, `blogs`, and `blogging`).

Table II. The five groups of tags involved in manual synonym analysis.

Group	Tags
1	webdev, games, javascript, politics, css, cool, programming, internet, java, music
2	yahoo, cd, help, conversion, algorithm, ontology, California, cheap, unix, text
3	!important, tolisten, collectiveintelligence, methodology, WWI, posicionamiento, chomsky, Scotland, M$, mono
4	webdesign, juegos, play, api, activism, accessibility, ui, toread, wiki, streaming
5	searchengine, ripping, SemWeb, Reasoning, *nix, totag, wicca, socialnetworks, pagerank, sf

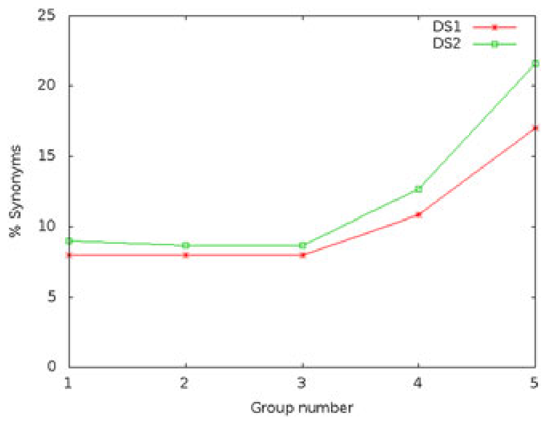

Figure 8. Percentage of synonyms in the top 50 related tags: averages for tag groups 1 to 5.

- It is a translation in another language (i.e., `game`, `juego`, and `spiel`).
- It is a part of a composed word that is a synonym of the given tag (i.e., `semantic`, and `web` can be tagged as synonyms of `semweb` if they appear in its top-related tags set).
- It is a spelling variation of the original tag (i.e., `semantic-web` and `semantic_web`).
- It is recognized as a well-known synonym (i.e., `programming` and `coding`). In this case, Wordnet could be used as an aid to find synonyms of a given tag.

After collecting manual annotations, we calculated the percentage of synonyms each tag had on average. Then for each group of tags (1–5), we calculated a global average. A first result is that as specified in [13], TCS tends to provide (some) synonyms as top-related tags: only two out of 50 tags in both datasets had no synonyms in their top 50 related tags. Then as already noted, we experimentally found that the bigger dataset was also the richer in synonyms (which is somehow intuitive, as we cannot lose tags in time, but continuously add new ones). Moreover, in both datasets, we saw that the higher the group number, the higher was also the percentage of synonyms (see Figure 8). This required a deeper analysis that we performed finding the following results.

First of all, less popular synonyms of popular tags usually present a higher number of synonyms in their top-related tags. For instance, `games` has 12% of synonyms, whereas its translations `juegos` has 38%; `unix` has 4.67%, whereas its spelling variant `*nix` has 10%, and so on. As another example (not belonging to the set of tags we analyzed but studied in a previous work), looking at the top 50 tags related to the very popular tag `toread`, we see that it can also be spelled as `read`, `read_later`, and `to_read`. Repeating the analysis with a less popular term with the same meaning (`@readit`), the list of synonyms grows covering nine out of the top 10 tags, and 17 out of the top 50. We attribute this different behavior to the fact that, as these tags are used transversally, being less popular also means covering fewer contexts, increasing the similarity with other less popular synonyms.

Another discovery is that some families of tags have more or less tendency to spawn synonyms. For instance, composed words (i.e., `searchengine`, `socialnetworks`, and so on) have a much higher average percentage of synonyms than the global average (19.87% against 11.81%). This is mainly because different characters (or none at all) can be used as word separators. The higher number of synonyms is also typical of translations of English tags in other languages. The main reason here is that English is the most common language in delicious; thus, this family of tags falls into the first one we described (i.e., less popular synonyms of popular tags). Conversely, the tags that had zero synonyms are all named entities (`yahoo` and `chomsky`), one of the categories that present fewer synonyms on average (5.73%).

6.3. *Homonyms analysis*

The main goal of experimentation with the homonyms analysis tool was twofold: on the one hand, we wanted to evaluate how the clustering algorithm performed in identifying different contexts of usage of the same tag; on the other hand, we aimed at verifying its behavior when real homonyms (and not just contexts) were found. This is because although the algorithm always tries to find a grouping for a set of tags related to a given one, many but not all of them are characterized by different usage contexts, and only few actually have homonyms.

To test our clustering algorithm, we ran it on a set of 12 tags containing (i) seven tags that were known in literature to be ambiguous (`apple`, `cambridge`, `sf`, `stream`, `tube`, `turkey`, `fps`), (ii) three tags that were known to have different contexts of usage (`retro`, `hack`, `horror`), and (iii) two popular tags (belonging to the set of the top 100 most frequent tags) that we considered not to have different contexts of usage (`flash`, `wiki`).

The algorithm has been tested by tweaking different parameters: (i) values of α, ranging from 1.5 to 2.5 (the interval is different from the one described in [33], but we experimentally found that the best clusterings with our data all fell within this range); (ii) values of n, describing the size of the graph of top-related tags we wanted to cluster (we tested $n = 50$ and $n = 100$); and (iii) full coverage of the clustering algorithm or not, performed through the deletion of nodes not yet assigned to a cluster whose fitness was negative.

In particular, parameters (ii) and (iii) allowed us to better understand the behavior of the clustering algorithm. Increasing the number of related tags basically means increasing the size of the graph we want to cluster, however with nodes that are in general less strictly connected with each other and definitely less relevant with respect to the examined tag. Conversely, algorithm performance decreases quadratically in the worst case. As a result, in most of the cases, $n = 50$ was the best choice. Similarly, performing a full coverage is an operation that tends to provide bigger clusters that overlap more with each other, leading to a worst-case behavior (for low values of α) of many overlapping clusters with all their elements in common but one. Partial coverage, instead, allows for the deletion of noise nodes and provides fewer clusters with smaller overlaps. The price is a loss of information, which however did not seem to negatively influence the interpretations of our clusters.

As a consequence, we analyzed only a subset of the results, that is, the one generated with $n = 50$, partial coverage, and α ranging from 1.5 to 2.5, at steps of size 0.01. For every single tag, we generated 100 different clusterings that we used to evaluate the algorithm behavior. To perform this evaluation, we relied on the common methodology of finding the knee in an error curve [38]. This technique is very common in the evaluation of clustering algorithms that require a parameter (i.e., the number of clusters) to be provided in advance. It works by measuring the compactness of the generated clusters in terms of the distances between their points, or between points and centroids. This value is plotted against the input parameter, and the generated curve is then analyzed to find peaks or knees. We performed this analysis manually, by plotting cluster compactness in terms of tag dissimilarities and finding the values of α that matched knees in the plot. Sometimes, more than one peak or knee were found: in those cases, the one that maximized the amount of information with respect to the number of clusters was chosen.

Figure 9 shows the behavior of the clustering algorithm for the tags `tube` and `fps`. The red line shows how the number of clusters varies with α, whereas the green line with points shows the sum

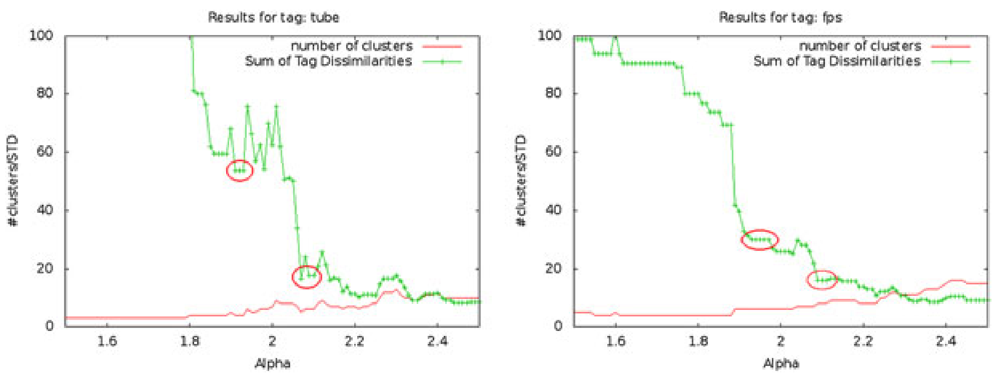

Figure 9. Knee finding in the results of the clustering algorithm for the tags `tube` and `fps`.

of tag dissimilarities. Two evident knees (circled in red) can be found in the plot for `tube`: the one on the left is around $\alpha = 1.91$, whereas the one on the right is around $\alpha = 2.07$. We chose the left-most knee as the one maximizing the amount of information with respect to the number of clusters (four instead of five). The results of this clustering are shown in Table III and are quite satisfying: the first cluster is all about amplifiers (tube or valve amplifiers are a particular family of electronic amplifiers); the second is related to public transportation (i.e., tube as the subway in London); the third is about youtube and streaming videos); and the fourth contains tags that describe qualities (e.g., bored, brilliant, coolstuff, and so on). A similar approach has been adopted for the tag `fps`: results show four clusters about First Person Shooter videogames (two of which are exactly split according to the software houses that produced these games) and two about the Fake Person Slash fan fiction genre.

Extending our analysis to the other tags, we were able to witness different clustering behaviors for the tag classes we have previously defined. Homonym tags typically match well separated clusters devoted to the different *meanings* of a tag. In our test sample, we found they were different from

Table III. The clusterings found by our algorithm (for each tag, the best performing alpha has been specified).

Tag	Alpha	Clusters
apple	2,06	(applescript, cocoa, tiger, macbook, macos), (software, imported, tools, howto, technology, tips, work, tech, cool), (system, unix, linux, windows, opensource, software)
cambridge	2,08	(boston, massachusetts, seattle, chicago, london), (college, mit, stanford, academics, academia, phd, academic)
sf	1,99	(science_fiction, science-fiction, startrek, sciencefiction, starwars, sci-fi), (books, book, writing, reading, read, interesting, people, culture, misc, future), (san_francisco, bayarea, sanfrancisco), (literary, lit, novels, authors, writer, writers, author, novel), (weird, random, awesome, misc, interesting, entertainment, archive, people, culture, read, future, events)
stream	1,99	(pod, towatch, webtv, iptv, vid, unmediated), (mp3s, muzyka, muziek, música, musique, musica, musik, sounds, music), (multimedia, audio, mp3, sound, radio, podcasts), (stations, shoutcast, netradio, internetradio, radios, webradio), (jukebox, mediaplayer, winamp, player, av)
tube	1,91	(amps, amplifiers, amplifier, tubes), (metro, transit, train, subway, bus, transportation, transport), (vids, videoclip, googlevideo, vid, webtv, iptv, vídeos, vídeo, vidéo, broadcasting, broadcast), (bored, funstuff, neato, neat, brilliant, coolstuff, entretenimiento, pasaradio, b, see, can)
turkey	2,02	(memphis, bulgaria, tenerife, cyprus, dubai, bankruptcy, victoria, abroad, las, arkansas, tennessee, owner, dreams, dream, denver), (a, how, on, in, your), (homes, property, florida), (beef, bacon)
fps	1,91	(sam/dean, dean/sam, wincest, dean, spn, supernatural), (halflife2, half-life, valve, hl2), (quake, unreal, doom), (giochi, onlinegames, spiel, juego, jeu, spel, spiele, jogos), (video_games, videogame, videogames, videojuegos, gamedev, gamedesign), (fic, fanfiction, recs, au, slash, angst, crackfic, pwp, threesome, kink)
retro	2,00	(cool, fun, art, images, animation, entertainment), (oldschool, atari, retrogaming, abandonware, 8bit), (wtf, strange, odd, amusing), (arte, posters, images, art, arts, animation, creative)
hack	2,11	(diy, make, projects), (utilities, utility, apps, applications, application), (read_later, tip, tricks, how-to, todescribe), (technology, tools, howto, tips, tutorials, toread, work)
horror	2,06	(sciencefiction, science_fiction, science-fiction, scifi), (creepy, gross, disturbing, scary, sad, evil, death), (movie, cinema, films), (pulp, monsters, zombie), (novels, authors, writers, author, novel)
flash	1,98	(animation, art, portfolio, graphic, photoshop, 3d, graphics, images, image), (tech, tools, technology, reference, web, internet, work, toread, imported, tutorials, webdev, webdesign, web2.0, resources, design)
wiki	2,07	(opensource, tools, tech, reference, imported, technology, internet, safari_export), (article, toread, reference, imported, research, technology, internet, tools, tech, work), (knowledge, wikis, communication, collaboration, content, social)

the ones we originally chose: whereas `sf`, `stream`, `tube`, `cambridge`, and `fps` performed well, `apple` and `turkey` did not return the expected results. We attribute this behavior to the fact that tags related to other meanings of the homonyms were not included in the chosen set of top-related tags. Moreover, `apple` was penalized by the presence of too many inflated tags (so popular that their actual informative content is very low), and `turkey` was victim of spam advertising. Multicontext tags also present a clear clustering, related to different *contexts of usage* rather than meanings, with some more overlap than in the previous group. Tags that are representative of this group are `retro` (clustered in two groups related to art, one about retrogaming and one containing subjective tags) and `horror` (clustered in groups covering movies, novels, science fiction, monsters, and subjective tags). The last two tags we supposed without contexts (i.e., `flash` and `wiki`) actually presented some more information than we expected: the former has a cluster devoted to graphics and related art, whereas the latter shows a group mostly related with social aspects of wikis. Both, however, also present mostly noise information in other groups, mainly because they are strongly related with many popular inflated tags.

As far as system performances are concerned, as the execution of the clustering algorithm is limited only to the subgraph identified by the top n related tags, the homonyms analysis tool can return clustering results in real time for a given value of the alpha parameter. As lower values of alpha bring bigger clusters, and as the algorithm's complexity is more than quadratic in the number of nodes per cluster, the higher the value of alpha, the faster is the execution. A test like the ones we performed, generating 100 samples for values of alpha that range between 1.5 and 2.5, takes on average less than 5 min on a laptop with a 2.80 GHz Intel Core2 Duo P9700 CPU.

7. CONCLUSION AND FUTURE WORK

Trying to overcome some intrinsic limits of folksonomies, we have proposed a methodology that relies on the concept of related tag to discover synonyms and homonyms within a tag-based system. Starting from a theoretical model that simplifies the tripartite graph by focusing on relations between tags, our analysis process follows the three stages of problem reduction, synonym and homonym analyses. The process has been developed as a prototype software, able to import data from different systems, efficiently calculate tag similarities, and run synonym and homonym detection algorithms.

Our experiments have been performed with the aim of getting a deeper understanding of synonymy and homonymy in tag-based systems. From this point of view, they have been a success, as they provided us interesting results whose interpretation led us to the conclusions that follow. Studying the evolution of the dataset in time, we observed that the vocabulary slowly converges: every year, the differences between the top 10,000 most popular tags become smaller and smaller. Another interesting result is that in the last comparison between two consecutive years, the tag similarity algorithm returned a high percentage (more than 60% on average) of top-related tags in common: this means that despite vocabulary evolution (i.e., people call things with different names or introduce new categories of resources), similarities tend to converge. Given that similarities depend, in our case, on co-occurrences between tags, this means that people call things differently, but they still do that in a consistent way.

Our tests on synonym detection showed that the metric we took into consideration, that is, TCS, performs well with some categories of tags such as composed words, translation of words in foreign languages, and less popular tags. Results are worse, instead, for other groups such as named entities. Experiments on homonym analysis showed the results of the application of an overlapping clustering algorithm to detect tag usage context and homonyms. The algorithm seems to perform well for values of the parameter α that, despite remaining in the same small range, depend on the tag taken into consideration. The methodology we introduced, however, provided satisfying results in a way that can be easily encoded into an algorithm (operation that we foresee as a possible extension of our work).

Finally, after the set of experiments we performed, results confirmed us that synonyms and homonyms are strongly related. On the one hand, limiting the discovery of different contexts of use to a list of semantically similar tags allowed us to perform clustering algorithms more efficiently without losing much useful information. On the other hand, homonym tag disambiguation allows us to clearly separate tag usage contexts and – as made evident by clustering results – makes it easier to find synonyms by automatically clustering them within the same context.

Future works include the import of the full version of dataset DS2 and the extension of our calculations to the new dataset. This would allow us to gain more insights about vocabulary evolution and verify whether there is also convergence on related tags. Working on the parallelization of our algorithms and relying on cloud computing services will help us to easily overcome the problems that are bound to the dataset size.

We also plan to extend our prototype with other similarity metrics (e.g., tag relatedness based on user vocabularies) and use them as a tool to make the manual work of annotating synonyms at least semi-automatic. This would help the evaluation of future synonym detection tools by lending a hand in finding syntactical variations of a tag. Finally, we want to extend the evaluation of our homonym detection module by comparing its results with other existing approaches.

REFERENCES

1. Dattolo A, Ferrara F, Tasso C. On social semantic relations for recommending tags and resources using folksonomies. In *Human-computer Systems Interaction: Backgrounds and Applications 2*, Vol. 98, Advances in Intelligent and Soft Computing. Springer: Berlin / Heidelberg, 2011; 315–332.
2. Vander Wal T. Folksonomy Definition and Wikipedia, November 2005. Available from: http://www.vanderwal.net/random/entrysel.php?blog=1750 [last accessed 05 August 2012].
3. Shirky C. Ontology is overrated: categories, links, and tags, 2005. Available from: http://www.shirky.com/writings/ontology_overrated.html [last accessed 05 August 2012].
4. Kroski E. The hive mind: folksonomies and user-based tagging, December 2005. Available from: http://infotangle.blogsome.com/2005/12/07/the-hive-mind-folksonomies-and-user-based-tagging/ [last accessed 05 August 2012].
5. Vander Wal T. Folksonomy, 2007. Available from: http://vanderwal.net/folksonomy.html [last accessed 05 August 2012].
6. Jannach D, Zanker M, Felfering A, Friedrich G. *Recommender Systems an Introduction*. Cambridge University Press: Leiden, 2011.
7. Ricci F, Rokach L, Shapira B, Kantor PB. *Recommender Systems Handbook*. Springer: New York; London, 2011.
8. Dattolo A, Tomasi F, Vitali F. Towards disambiguating social tagging systems. In *Handbook of Research on Web 2.0, 3.0 and x.0: Technologies, Business, and Social Applications*, Murugesan S (ed.). IGI Global: Hershey, PA, 2010; 349–369.
9. Golder S, Huberman BA. The structure of collaborative tagging systems. *Journal of Information Science* 2006; **32**(2):198–208.
10. Gemmell J, Ramezani M, Schimoler T, Christiansen L, Mobasher B. The impact of ambiguity and redundancy on tag recommendation in folksonomies. *Proceedings of the Third ACM Conference on Recommender Systems (RECSYS '09)*, New York, NY, USA, 2009; 45–52.
11. Dattolo A, Ferrara F, Tasso C. Supporting personalized user concept spaces and recommendations for a publication sharing system. In *17th International Conference, UMAP 2009, formerly UM and AH*, Vol. 5535, Lecture Notes in Computer Science. Springer, 2009; 325–330.
12. Schmitz C, Hotho A, Jäschke R, Stumme G. Mining association rules in folksonomies. In *Proceedings of the IFCS 2006*, Studies in Classification, Data Analysis, and Knowledge Organization. Springer: Berlin/Heidelberg, 2006; 261–270.
13. Cattuto C, Benz D, Hotho A, Stumme G. Semantic grounding of tag relatedness in social bookmarking systems. In *ISWC 2008*, Vol. 5318, LNCS. Springer, 2008; 615–631.
14. Ignacio Fernández-Tobías AB, Cantador I. cTag: semantic contextualisation of social tags. In *Proceedings of the International Workshop on Adaptation in Social and Semantic Web (SASWeb 2011)*, Vol. 730, Cena F, Dattolo A, De Luca EW, Lops P, Plumbaum T, Vassileva J (eds). CEUR: RWTH, Aachen, 2011; 40–49.
15. Lambiotte R, Ausloos M. Collaborative tagging as a tripartite network. In *Computational Science ICCS 2006*, Vol. 3993, LNCS. Springer: Berlin / Heidelberg, 2006; 1114–1117.
16. Mika P. Ontologies are us: a unified model of social networks and semantics. In *ISWC 2005*, LNCS. Springer: Amsterdam, The Netherlands, 2005; 522–536.
17. Angeletou S. Semantic enrichment of folksonomy tagspaces. In *ISWC 2008*, Vol. 5318, LNCS. Springer: Berlin, Heidelberg, 2005; 889–894.
18. man Au Yeung C, Gibbins N, Shadbolt N. Understanding the semantics of ambiguous tags in folksonomies. *Proceedings of the International Workshop on Emergent Semantics and Ontology Evolution (ESOE2007) at ISWC/ASWC2007*, Busan, South Korea, November, 2007; 108–121.
19. man Au Yeung C, Gibbins N, Shadbolt N. Contextualising tags in collaborative tagging systems. *Proceedings of the Twentieth ACM Conference on Hypertext and Hypermedia (HT '09)*, New York, NY, USA, 2009; 251–260.
20. Begelman G, Keller P, Smadja F. Automated tag clustering: improving search and exploration in the tag space. *Proceedings of the Collaborative Web Tagging Workshop at WWW2006*, Edinburgh, Scotland, 2006; 15–33.

21. Brooks CH, Montanez N. Improved annotation of the blogosphere via autotagging and hierarchical clustering. *Proceedings of the 15th International Conference on World Wide Web (WWW2006)*, Edinburgh, Scotland, 2006; 625–632.
22. Grahl M, Hotho A, Stumme G. Conceptual clustering of social bookmarking sites. *Proceedings of the 7th International Conference on Knowledge Management (I-KNOW '07)*, Know-Center, Graz, Austria, 2007; 356–364.
23. Pudota N, Dattolo A, Baruzzo A, Ferrara F, Tasso C. Automatic keyphrase extraction and ontology mining for content-based tag recommendation. *Int. J. Intell. Syst.* 2010; **25**(12):1158–1186.
24. Dix A, Levialdi S, Malizia A. Semantic halo for collaboration tagging systems. *Proceedings of the Workshop on the Social Navigation and Community Based Adaptation Technologies*, Dublin, Ireland, 2006; 514–521.
25. Vandic D, van Dam J-W, Hogenboom F, Frasincar F. A semantic clustering-based approach for searching and browsing tag spaces. *Proceedings of the 2011 ACM Symposium on Applied Computing*, SAC '11, New York, NY, USA, 2011; 1693–1699.
26. Dattolo A, Ferrara F, Tasso C. Neighbor selection and recommendations in social bookmarking tools. *Proceedings of the 2009 Ninth International Conference on Intelligent Systems Design and Applications*, ISDA '09, Washington, DC, USA; 267–272.
27. Specia L, Motta E. Integrating folksonomies with the semantic web. In *Proceedings of the European Semantic Web Conference (ESWC2007)*, Vol. 4519, LNCS. Springer-Verlag: Berlin Heidelberg, Germany, 2007; 624–639.
28. Desrosiers C, Karypis G. A comprehensive survey of neighborhood-based recommendation methods. In *Recommender Systems Handbook*, Ricci F, Rokach L, Shapira B, Kantor PB (eds). Springer US, 2011; 107–144.
29. Newman MEJ, Girvan M. Finding and evaluating community structure in networks. *Physical Review E* 2004; **69**(2):026113.1–15.
30. Markines B, Cattuto C, Menczer F, Benz D, Hotho A, Stumme G. Evaluating similarity measures for emergent semantics of social tagging. *Proceedings of the 18th International Conference on World Wide Web (WWW'09)*, Madrid, Spain, April 2009; 641–650.
31. Nakamoto R, Nakajima S, Miyazaki J, Uemura S. Tag-based contextual collaborative filtering. *Journal of Intelligent Information Systems* 2008; **34**(2):214–219.
32. Angeletou S, Motta E, Sabou M. Improving folksonomies using formal knowledge: a case study on search. *Proceedings of the 4th Asian Semantic Web Conference*, Springer-Verlag, 2009; 276–290.
33. Lancichinetti A, Fortunato S, Kertesz J. Detecting the overlapping and hierarchical community structure of complex networks. *New Journal of Physics* 2009; **11**(3):033015 (18pp). DOI: 10.1088/1367-2630/11/3/033015.
34. Harris Z. *Mathematical Structures of Language*. John Wiley and Son: New York, 1968.
35. Wetzker R, Zimmermann C, Bauckhage C. Analyzing social bookmarking systems: a del.icio.us cookbook. *Proceedings of the Mining Social Data (MSODA) Workshop at ECAI 2008*, Patras, Greece, 2008; 26–30.
36. Laniado D, Eynard D, Colombetti M. Using wordnet to turn a folksonomy into a hierarchy of concepts. *Semantic Web Application and Perspectives – Fourth Italian Semantic Web Workshop*, Bari, Italy, 2007; 192–201.
37. Dattolo A, Eynard D, Mazzola L. An integrated approach to discover tag semantics. *Proceedings of the 2011 ACM Symposium on Applied Computing*, Taichung, Taiwan, 2011; 814–820.
38. Tan P-N, Steinbach M, Kumar V. *Introduction to Data Mining, (first edition)*. Addison-Wesley Longman Publishing Co., Inc.: Boston, MA, USA, 2005.