From Web 2.0 to Semantic Web: A Semi-Automated Approach

Andreas Heß, Christian Maaß and Francis Dierick

Lycos Europe GmbH, Gütersloh, Germany
{andreas.hess, christian.maass, francis.dierick}@lycos-europe.com

Abstract. Web 2.0 and Semantic Web are regarded as two complementary paradigms that will probably converge in the future. However, whereas the Semantic Web is an established field of research, there has been little analysis devoted to Web 2.0 applications. For this reason it remains unclear how the advantages of both paradigms could be merged. In this paper we make three contributions in this direction. First, we discuss why merging Web 2.0 and the Semantic Web is beneficial and propose five approaches. Second, we show that (semi-) automated tagging of content improves the quality of annotations. Third, we present an automatic approach for improving the tag quality by using duplicate detection techniques. We verify our approach on a large-scale data set from the social search service Lycos IQ.

1 Introduction

The Semantic Web promises an easy access to information sources using a machine-understandable (not only machine-readable) representation of knowledge. This requires web resources be annotated with machine-understandable meta-data. Presently, the primary approach to achieve this is to first define an ontology and then use the ontology to add semantic markup for web resources. However, at present only a fraction of web users can take part in the process of building ontologies. Ontology tools and ontology languages impose high entrance barriers for potential users [11]. This is likely to contribute to the fact that the most popular approach of creating ontologies is engineering-oriented, i.e. a small number of individuals carefully construct the representation of the domain of discourse and release the results at some point in time to a wider community of users. In other words, the ontology evolution is not under full control of the users. For example, missing entries cannot be added by any user who finds the need for a new concept, but have to be added by the small group of creators. In natural language, in comparison, the evolution of the vocabulary is under the control of the user community. Anybody can invent and define a new word or concept in the course of communication.

Against this background there was a large debate in academic literature on how the process of meta-data generation can be automated to address the problem of cost-intensive ontology construction and generation of meta-data, e.g. [3, 11]. This is an important research question as several researchers agree

that without the proliferation of formal semantic annotations the Semantic Web is certainly doomed to failure [3].

One way to lower the threshold for a user to enter the Semantic Web is to move away from strict 'heavy' ontologies towards light-weight ontologies, folksonomies or tagging. With the proliferation and growth of so-called Web 2.0 sites, tag-based folksonomies promise to be a useful tool for search and navigation. Unlike an ontology, which is usually defined as a "specification of a conceptualisation" [9] and due to its formal nature created by trained experts, a folksonomy is "a type of distributed classification system" and "usually created by a group of individuals, typically the resource users" [10].

However, as the user and content base of a site grows, folksonomies tend to become more diffuse and imprecise. A certain degree of automation would help to make the maintenance of folksonomies as well as the annotation of content easier. In spite of the obvious need for this, up to now only a few automated approaches have been presented.

Even though tagging is not comparable to annotations using a full ontology, it can be regarded as a first step towards the Semantic Web. Even more important, tagging is already used every day by millions of web users posting blog entries. It is accepted that automated tagging algorithms, unlike manually tagged data, can have significant levels of mis-classification [5]. In a semi-automatic setting predicted tags do not have to be perfectly accurate in order to be useful. It is still easier for a user to browse through a list of only a few possible tags than to enter their own free-text tags. Furthermore, entering new tags is error-prone, since synonyms or spelling mistakes are not always detected. If the tags are drawn from a full ontology or at least a controlled vocabulary, looking only at a few suggestions is easier than looking at a complete ontology with possibly thousands of concepts.

In the remainder of this paper we will make three contributions to address the question, how the top-down approach of ontology engineering could be merged with a bottom-up approach that is typical for so-called Web 2.0 applications. First, we provide an overview of the advantages and disadvantages of ontologies and folksonomies and why a merging of these concepts is beneficial. Second, we will focus on the semi-automated classification or tagging of content. We believe that when using algorithmic assistance for annotating text, the quality of annotations will increase. We present a machine-learning-based classification algorithm that is tailored for use with short texts and folksonomies. We show that part-of-speech-tagging can be used in text classification and retrieval to dramatically reduce the dimensionality of the corpus without affecting performance. The algorithm is fast enough for interactive use. Third, we present an automatic approach for tag merging and correction. This is an important issue as folksonomies usually do not consist of a limited number of well-written tags. Rather, almost every Web 2.0 application faces the problem that tags are misspelled or are redundant. To address these problems, we present a method for detecting different (mis-)spellings of a tag that is based on a spell-checker in con-

junction with string edit distance metrics. We use a rule learner for fine-tuning the parameters of the algorithm.

1.1 Folksonomies: Usage Scenario

Despite their complementary nature, currently folksonomies (or tag clouds) and ontologies are used in quite distinct usage scenarios. Folksonomies are mainly used for tagging in Web 2.0 applications, the main use cases being search, navigation and recommendation. For social bookmarking sites such as `del.icio.us` tagging is an essential part of these processes: links are annotated and thereby sorted into categories, a user can search by category, etc. For applications like photo sharing, tagging is a prerequisite for effective searching, since a picture cannot easily be searched by its actual content. By arranging related tags (e.g. by co-occurrence) folksonomies also allow for browsing through different categories. While folksonomies have the clear advantage of being cheap and reflecting the language of the user, there are certain problems related to their use: if tags are not drawn from a controlled vocabulary but are just plain text keywords, several issues that affect the usefulness of tagging will arise. Especially inexperienced users tend to assign tags that are not meaningful to other users or to assign no tags at all. For example, as a result of an analysis of the leading social bookmarking system `del.icio.us`, Lee points out that about 20% of its users do not annotate or tag any of their bookmarks [15]. Moreover, different spellings and subjective combinations of tags lead to more or less diffuse folksonomies. Therefore, errors occur frequently while searching for related issues and subjects. Some users are well aware that this is a problem and that there should be some tagging guidelines. This problem and whether tools should be used is discussed both in the blogosphere[1] and by scientists [21]. We argue that by moving from folksonomies towards ontologies, the usefulness for Web 2.0 scenarios will improve as well. Therefore, we investigate the use of semi-automatic techniques for assisting the user in annotating content and cleaning existing tag clouds and thus moving to a more structured representation.

1.2 Proposed Approaches

Making the transition from Web 2.0 to Semantic Web smooth and user-friendly is a difficult task. We propose a combination of five approaches to address the different aspects of the problem:

Semi-Automated Tagging As a first step, we believe it is very important to reduce the uncontrolled growth of tag clouds due to usage of synonyms, misspelled words and inconsistent tagging. We propose using semi-automated tag suggestions based on text classification to guide users towards consistent tagging.

[1] e.g. `http://paolo.evectors.it/2005/05/24.html#a2532` or `http://ross.typepad.com/blog/2005/05/tags_and_simple.html`

We believe that users will more likely choose from a list of suggested tags than entering new tags. As a consequence the quality of annotations will increase. In section 2 we present our algorithm for semi-automated tagging in detail.

Tag Merging We believe that merging of synonyms and misspelled tags will increase the quality of annotations in the same way as automated tagging. The result of merging similar tags denoting the same concept will be a more consistent tagging. We discuss our algorithm for tag merging in section 3.

Identification of Related Tags Based on co-tagging (i.e. tags used together to annotate the same content) it is possible to identify a network of relationships between tags. Approaches that involve further analysis of such a network of tags have been discussed, e.g. [13]. However, this topic is out of the scope of our own research area.

Tag Rating For the combination of tags drawn from a folksonomy and concepts in an ontology we follow a layer concept: User-entered tags are located in an outer layer, whereas concepts in an ontology that is maintained by experts are located in an inner core. When tags in the outer layer are identified as being consistent and precise and there is no equivalent concept in the ontology already, these tags should be included as concepts. We propose that the user community can rate annotations. When a tag gets a high number of good ratings it should be recommended to the experts for inclusion in the core ontology. We will investigate using such a rating mechanism in future work.

Information Extraction We propose to make extensive use of information extraction techniques to fill the core ontology with facts. DBPedia [1] is an approach for extracting facts from Wikipedia articles. Furthermore, a large amount of previous work on information extraction from websites (e.g. [14]) and free form text (e.g. [7]) exists. Some approaches are targeted directly towards use in the Semantic Web area, e.g. [4]. In [16] an approach for finding relationships via a web search is presented. We are currently researching in the same area. In our approach, we are combining results from information extraction sources with a web search in order to identify the type of relation between two persons and to either confirm or disprove whether such a connection exists. We will present this approach in greater detail in a future publication.

1.3 Case Study: Lycos iQ

Lycos iQ is a question-and-answer community web site. Q&A communities try to deliver answers where algorithmic search engines fail to generate high quality results by activating users from the Web 2.0 community. For example, current search technologies have problems to answer search enquiries just like “Who is the Swedish singer that sounds like Heather Nova”. These kinds of questions

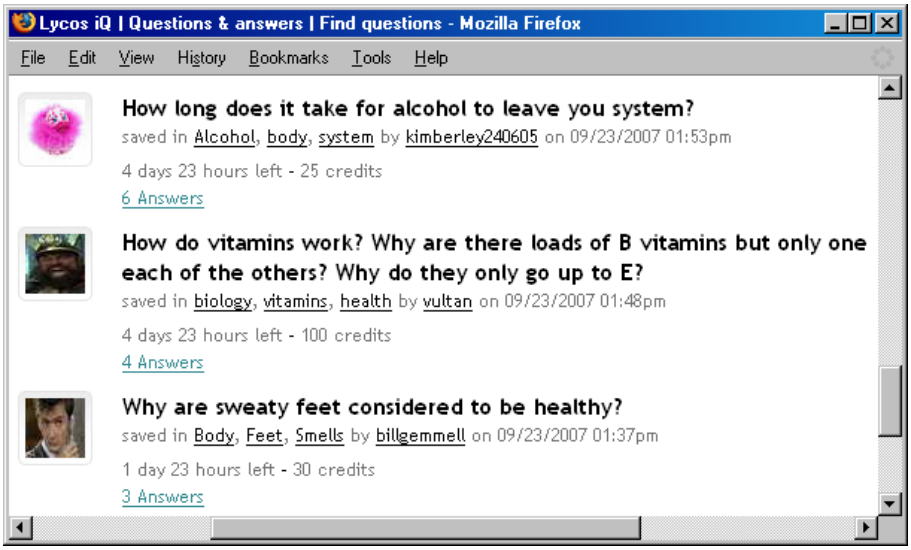

Fig. 1. Screenshot of the Lycos iQ website, illustrating our dataset

could only be answered by humans or by a group of users. Q&A services provide the necessary infrastructure to discuss such questions and enable a broad community to share their knowledge. Figure 1 shows a screenshot from the Lycos iQ website[2]. Q&A communities such as Lycos iQ or Yahoo Answers must not be confused with Q&A systems as known in Information Retrieval.

Tags are a certain form of meta-data that serves as a description for a particular content. For example, a question like the above could be published with the tags 'music', 'Sweden', 'singer' or 'songwriter'. Through these tags the posting is associated with the topics 'music' and 'songwriter', although the terms are not explicitly mentioned in the text.

From the technical point of view the 'tagging' of questions is the key to success for these kinds of services: Based on tags, expert users that can answer questions from a specific topic can be identified and brought together with users seeking information. The quality of tags is a crucial element in the functionality of such a service. However, as social bookmark systems have attracted a great attention [15, 8], there has been little analysis devoted to Q&A- and other communities.

2 Semi-Automated Tagging

2.1 Related Work

Although text classification is an old and well-researched topic in information retrieval and machine learning (e.g. [17]) it has not been widely used for au-

[2] `http://iq.lycos.co.uk/`

tomatic tagging in Web 2.0-applications yet. An exception is AutoTag [19], a system that uses a k-nearest-neighbour classifier for automated tagging of blog posts. This work is closely related to ours. We will highlight the differences in the next section. A more complex case-based system for semi-automated tagging is TagAssist [22]. As far as we are aware, these systems have not been deployed in an environment outside of the research community. No well-known Web 2.0 sites actually use semi-automated tagging based on learning or natural language techniques.

2.2 Problem Formulation

Formally, semi-automated tagging is a multi-value text classification problem.[3] Most machine learning algorithms can only handle single-value classification. Therefore it is common practise that single-value classification algorithms are adapted by means of some combination method; see [23] for a recent survey. However, these strategies are out of the scope of this paper: given that in our scenario many annotations are plausible and the user is involved in the classification process, it is not necessary that the algorithm predicts the exact set of true annotations and presenting a ranked list is acceptable. Considering the high number of classes and the need for an incremental learning algorithm, using vector space classification algorithms such as kNN or Rocchio is a logical choice. AutoTag [19] uses a search engine to locate similar blog posts. The search query is derived from the text that is to be classified using statistical query rewriting techniques. In the next step, tags from the search results are aggregated and re-ranked using information about the user. Yet, this method of predicting tags for posts has a disadvantage. Re-writing the query at classification time is computationally costly. In a semi-automatic setting, where users can annotate their content online immediately after they type, response time is critical. Therefore, to avoid the need for query rewriting, we decided to perform a feature selection at training time. First we tokenised the short headline of the questions. This headline usually consists of just one sentence. We applied part-of-speech tagging and kept only nouns and proper nouns. Although the headlines in Lycos iQ are much shorter than blog posts (usually just one sentence) and we did not use re-ranking, we achieved similar performance results in preliminary experiments conducted with the current live version of Lycos iQ.

2.3 kNN classification vs. Rocchio

For comparison, we implemented two classifiers, both based on an index created using the dimensionality reduction method described above (POS tagging). First, we implemented a kNN classifier with $k = 10$ that queries the index for the ten nearest questions (postings) in the database and aggregates the tags from

[3] For the remainder of this paper, we will use the terms 'class' as in text classification and 'tag' as synonyms.

the results. Preliminary tests showed that IDF weighting does not improve classification results in this setting, so we decided not to use it in our experiments.

Second, we implemented a Rocchio-style classifier. Rocchio classification is based on Rocchio relevance feedback [20]. The centroids for each class were simply computed as a big bag of words containing all tokens from all posts labelled with this specific class (tag). Note that if a posting is tagged with more than one tag the tokens in the post are indexed for all its classes. As opposed to the kNN classifier, we found out that IDF weighting slightly improves performance, so we used it in our experiments. Although the difference in accuracy between using IDF weighting and not using it were rather low, we decided to use the best setting for each of the two approaches to achieve a fair comparison.

Given the nature of the two algorithms, we expect that Rocchio classification will be faster at classification time, a factor that is very important in an interactive setting, when users are not willing to accept long response times.

When comparing the classification performance of Rocchio vs. kNN, Rocchio has some known disadvantages: It becomes inaccurate, when classes are not spheres of similar size in vector space, and it does not handle non-spherical classes (e.g. multi-modal classes that consist of more than one cluster) very well. However, we argue that these properties of Rocchio classifiers will not affect performance in our setup, because of three main reasons: First, while the dataset is indeed skewed and classes have very different size, we expect that most of the classes will be mono-thematic. Second, while large classes will be preferred by the classifier, this is not a disadvantage in our scenario. When considering semi-automated tagging of postings, it is actually an advantage when we direct users towards choosing more popular tags, as explained in section 1. Third, since our scenario is a multi-value classification problem where not only a single prediction is correct, overlap between classes does not necessarily influence accuracy negatively.

2.4 Evaluation

To evaluate our approach, we used a corpus of 116417 question from the knowledge community web site Lycos iQ. Figure 1, showing a screenshot from the Lycos iQ website, gives a good impression of what our dataset looks like. Note the different levels of tags the users assigned and the style of the texts. We use only the headlines as shown on this screenshot for classification. After tokenisation and POS tagging, there were 89275 distinct tokens. After deleting frequent tokens according to Zipf's law, this number was further reduced to 27309, leading to an average of only 2.63 tokens per question. Questions were tagged with 49836 distinct tags that follow the usual distribution with some frequently used tags followed by a long tail. It should be noted that the number of classes exceeds the number of tokens. We expect that tags drawn from the long tail will be suggested very rarely if at all. However, this does not pose a problem, since guiding the user towards common tags accepted by many users is one of the goals of our approach. Suggesting tags that were initially only used very few times or even only once does not make sense. It is not useful to include text

from answers in the classification. When the user is tagging a question, there are no answers available yet. The text from the answers is not helpful for classifying the questions, since the words are too different. We evaluated both the kNN approach and the Rocchio approach automatically using the 111629 questions that had user-assigned tags, comparing the predicted tags with the manually assigned tags. We used the well-known leave-one-out cross-validation scheme for evaluation. After preprocessing we encountered the fact that some questions were not assigned any tokens. These questions were, however, left in the corpus and thus affect the overall performance of our algorithm negatively.

Methodology It is important to note that the tags assigned by users should not be regarded as a gold standard. Tags are not drawn from an ontology, taxonomy or controlled vocabulary, but are free text entered by users and thus prone to spelling mistakes. Also, inexperienced users tend to assign either no tags at all, only very few tags or they tag inconsistently. Given the large number of users, we also expect that users use different synonyms to denote the same concept. Due to these ambiguities and inconsistencies we expect that the accuracy of any automated approach is considerably lower than its true usefulness.

To overcome this problem in our empirical evaluation, we distributed questionnaires and had test persons check the plausibility of tags suggested by our semi-automated approach. To reduce the workload for the test persons and because it outperformed the kNN classifier in the automated tests, we decided to test only the Rocchio-style approach. For comparison, we also had the test persons check the precision of the user-assigned tags, since we assumed many nonsensical or inconsistent tags among them. Every test person was given one or two chunks of 100 out of a random sample of 200 questions that were either machine-tagged or hand-tagged. Every question was checked by four persons to average out disagreement about the sensibility of tags.

Automatic Evaluation For the classification results evaluated against the user-assigned tags, we report precision, recall and accuracy. Precision is defined as the number of predicted tags that also occur in the set of manually assigned tags divided by the number of predicted tags that were considered. We report precision only for the top three predicted tags. Since most questions are only tagged with up to three tags, it does not make sense to report precision when allowing for more predicted tags. Recall is defined as the number of tags in the set of user-assigned tags that also occur in the set of predicted tags divided by the number of user-assigned tags. We report recall for the top ten predicted tags. With this definition of precision and recall we follow the generally accepted definition in information retrieval and machine learning. Furthermore, we report the fraction of questions where there is at least one overlap between the assigned and predicted tags as accuracy. We regard it is already useful, if only some or even one sensible suggestion is among the top suggested tags, since the user can quickly browse through a short list of suggestions and select or deselect tags. We expect that accuracy and recall will go up if we include more suggested tags.

Top n tags	1	2	3	4	**5**	6	7	8	9	10
kNN Precision	0.26	0.24	0.20	—	—	—	—	—	—	—
kNN Recall	0.26	0.23	0.21	0.23	**0.24**	0.25	0.26	0.27	0.27	0.28
kNN Accuracy	0.23	0.29	0.32	0.34	**0.35**	0.36	0.37	0.38	0.39	0.39
Rocchio Precision	0.32	0.31	0.27	—	—	—	—	—	—	—
Rocchio Recall	0.32	0.30	0.28	0.31	**0.33**	0.35	0.37	0.38	0.39	0.40
Rocchio Accuracy	0.28	0.36	0.41	0.43	**0.45**	0.47	0.48	0.49	0.49	0.50

Table 1. Results of the automatic evaluation

Table 1 shows the empirical results for precision, recall and accuracy for the two different proposed methods. For recall and accuracy, we highlight the value at five suggested tags, because we believe that a user will not accept a longer list. As Miller pointed out in [18], most people can process five items at once. We measured the classification time per instance for both approaches on an Intel Core 2 machine with 1.86 GHz and 1 GB RAM. As expected, Rocchio classification was much faster than kNN. The classification time for each instance was 155 ms for kNN and 57 ms for Rocchio.

Manual Evaluation As expected, we could observe that there was a big disagreement among the test persons and the users who originally tagged the questions as well as between the test persons themselves. For the manual evaluation, we checked only the Rocchio classifier because it performed better in the automatic test. As explained above, the total 200 questions that were evaluated were split in two sets of 100 questions, yielding four different questionnaires (two for the original user-assigned tags and two for machine-annotated tags) and each chunk of 100 questions was checked by four persons. Each test person was checking at most two sets of questions. To highlight the huge difference of the several test persons, we report the individual results in the table below. For the human-annotated tags, we evaluated precision, defined as the number of useful tags divided by the total number of assigned tags. For the machine-assigned tags, we report accuracy as well, with the same definition of accuracy as in the automatic test. Questions that had no assigned tags were ignored for evaluating accuracy. Among the 200 randomly selected question were 6 that had no assigned tags, leading to 194 questions evaluated for accuracy.

For all manual tests, we evaluated the algorithms with five suggested tags only. We believe that in a real-world semi-automated setting, we cannot assume that an inexperienced user is willing to look at more than five tags. The questions that were manually tagged had mostly three tags each, some of them only two and very few questions had more than three tags.

As expected, there was a large disagreement between different persons both on the human-annotated tags as well as the machine-annotated tags (see tables 2). It is interesting to note when looking at the second set of questions, that, although the human annotations on this set of 100 questions were rated worse than those from the first set, the tags suggested by our algorithm were

Test	TP	TP+FP	avg. Prec.
assigned tags	1535	1856	0.83
suggested tags	1866	3360	0.56

Test	Person 1	Person 2	Person 3	Person 4
Set 1, assigned tags, prec.	0.89	0.89	0.93	0.96
Set 2, assigned tags, prec.	0.52	0.73	0.73	0.87
Set 1, suggested tags, prec.	0.41	0.52	0.53	0.71
Set 2, suggested tags, prec.	0.51	0.54	0.59	0.65
Set 1, accuracy	0.84	0.84	0.86	0.87
Set 2, accuracy	0.87	0.87	0.91	0.91

Table 2. Results of the manual evaluation

on average rated even slightly better. Keeping in mind that we envision a semi-automated scenario with human intervention, we see this as a confirmation that automatically suggested tags can help to improve the quality of tagging.

When looking at macro-averaged precision, it is obvious that a classification system is still not good enough for fully automated tagging. However, it is important to note that even the human-annotated questions were rated far below 100% correct by the test persons. More than half of the suggested tags were rated as useful by the test persons. We believe that this is certainly good enough for a semi-automated scenario, were users are presented a small number of tags to choose from. In absolute numbers, interestingly, the automatic classifier produced more helpful tags than were assigned by users, even compared to the number of all user-assigned tags, not just the ones perceived as helpful by the test persons. We believe that this confirms our hypothesis that users will assign more tags when they are supported by a suggestion system. However, this can only be finally answered with a user study done with a live system.

Finally, the high accuracy of the classifier (with accuracy being defined as in section 2.4) underlines our conclusion that semi-automated tagging is good enough to be implemented in a production environment. In almost nine out of ten cases there was at least one helpful tag among the suggestions.

3 Tag Merging

3.1 Problem Formulation

When considering a social tagging system with a potentially high number of users, one of the most common problems is inconsistent tagging. One common facet is that users tend to misspell tags . While we can guide the user towards consistent tagging by suggesting tags, a complementary approach is to clean existing tag clouds after the annotation phase by identifying tags that should be merged. This merging can be based both on string similarity to detect misspellings or based on a thesaurus to detect synonyms. The first approach is well known as duplicate detection or record linkage. Yet, little attention has been

devoted to using duplicate detection techniques for improving the quality of tag clouds. Unlike semi-automated annotation, tag duplicate detection must have a precision that is high enough to be run unsupervised, with only little manual correction. An interactive review is infeasible when merging possibly several thousand misspelled tags. Therefore, the duplicate detection algorithm should be biased towards high precision.

3.2 Parameter Tuning

The obvious approach to address the tag duplicate detection problem is to use string similarity metrics. Preliminary experiments, however, showed that the performance of using a single string distance metric is not sufficient to be employed in an automatic setting in terms of precision. Therefore, we decided to combine multiple similarity measures and use a machine learning algorithm to fine-tune the exact setting. A similar approach has been used by Bilenko and Mooney [2].

We use a two-step approach. First, we check whether a tag should be considered for merging. Second, if the tag is suitable, we use an off-the-shelf spell-checking tool to identify possible tags that are candidates for merging. To be able to use a standard spell checker for our purpose, we do not use a natural language dictionary but the list of all tags in the system instead. Therefore, suggestions by the spell checker are tags that are textually similar. While an efficient implementation of a spell checker will return a result very quickly, these results are not accurate enough for a fully automated merging process. To increase precision, we perform further checks. Pairwise calculation of all of these features for all tags would be computationally too expensive and therefore prohibitive.

For the first step, checking whether a tag should be considered for merging, we used a number of features such as frequency, number of related tags, string length and the number of tokens (a tag can consist of multiple words). Tags are called related tags, when they are used together to annotate the same content. Related tags of second order are tags that are related to the same other tag (e.g. if there are relationship between "house" and "door" and between "building" and "door", then "house" and "building" have a second order relationship). Frequency is defined as the number of times the tag has been used. For the second step, checking whether two tags should be merged, we considered additional features such as Levenshtein edit distance, Jaro-Winkler string similarity, Monge-Elkan string similarity and Smith-Waterman string similarity. The relationship strength is defined as the number of co-occurrences. The relationship strength of second order is defined as the sum of the strength of the connecting first order relations.

To tune the thresholds for the merging system we created training sets with the features explained above and exported them into ARFF format for processing with WEKA [24]. Because of the desired area of use and the simplicity of implementation of the classifier, learners that output rules seemed an obvious choice. We experimented with different rule learning and decision tree algorithms

including RIPPER and C4.5 as implemented in WEKA.[4] Since our goal is to implement an automatic, unsupervised tag merger, we consider precision on the class of merged tags as more important than recall and accuracy. A classifier with high precision leads to a merger that will make few mistakes at the price of missing some tags that should be merged. To bias the learning algorithm towards precision on one class, we experimented with biased classifiers from the implementation of the Triskel algorithm [12]. Biasing methods include under-sampling (randomly dropping training instances from one class) and over-weighting (assigning a higher weight to instances from one class). It has been shown that even dropping as much as 90% of instances leads to a good classifier with high precision. After looking at the resulting models, it turned out that, as expected, some of the features were redundant.

3.3 Evaluation

To evaluate the tag merger, we used the same dataset as described in section 2, with 49836 distinct tags in the database. Running the merging algorithm generated 4320 sets of merged tags with 10245 tags in total, yielding an average 2.4 tags per set. This means that after merging, about one fifth of tags are part of a set of merged tags, and when only keeping one tag per set, the amount of tags available will be reduced by around 11.9%.

To measure the precision of the merger, we examined a sample of 100 tag sets containing 248 distinct tags. Of these 100 sets, 3 contained one tag that was not correctly merged, 1 set contained two tags that did not fit and in 2 cases it could be considered questionable whether or not the tags should be merged. This leads to an average precision of 94% resp. 96% (depending on whether the two tags in doubt are considered correctly merged). When putting the errors in relation to individual tags instead of sets (keeping in mind that a set can contain more than two tags), precision is approx. 97.98% resp. 97.18%.

We believe that the precision of this approach is high enough to allow elimination of misspelled tags or merging of tags in their singular or plural form in a fully automated setting. Even with very conservative settings yielding a high precision, there is already a significant amount of tags that can be merged or discarded.

4 Conclusion

4.1 Summary

In this paper, we have made three contributions: first, we have proposed five approaches to address some of the problems when moving from a folksonomy towards an ontology. We argue that this move brings benefits, although we have to keep in mind that heavy ontologies tend to become too complicated for the broad mass of inexperienced users. Therefore, we advocate for (semi-) automated

[4] The WEKA implementations of RIPPER and C4.5 are called JRip and J4.8

measures to guide users towards more structured tagging. We regard this as a first step on the move from Web 2.0 to the Semantic Web.

Second, we have presented a classification algorithm that has been shown to perform well in terms of accuracy even when using very short text snippets. We have shown that a drastically reduced dimensionality of the text corpus can be achieved when using part-of-speech tagging. We conclude from our empirical study based on a dataset with questions from the Lycos iQ web site that the performance of our classifier is good enough to be employed in a semi-automatic setting and that it scales well enough to be used in a production system with a large number of instances.

Third, as a complementary measure to the proposed semi-automated tagging, we have shortly described an algorithm for automatically merging misspelled or similar tags and have shown that its precision is high enough for unsupervised operation while still eliminating a high number of tags. We believe that both ad-hoc interactive tagging suggestions as well as post-hoc offline merging are effective approaches to improve the quality of a folksonomy as a prerequisite towards moving to more structured semantic networks and ontologies.

4.2 Generalisation

In recent work that is beyond the scope of this paper we have tested our text classification system on the well-known Reuters dataset. We have adapted our algorithm to output a true multi-value classification instead of a ranked list. We introduce a new method for multi-value-classification that is related to stacking. The performance of our algorithm is comparable to the older results presented in [6], although we use only the headlines of the articles as opposed to the full text. In [6], a macro-averaged accuracy for the top 10 classes of 63.7% was reported for the Rocchio classification algorithm in a one-classifier-per-class scheme. We achieved a 67.1% accuracy using our stacking approach. Due to the different focus of these experiments and space restrictions, we will present these results in a separate publication. Even though we focused on Rocchio and kNN classification due to performance requirements related to interactive use, using SVMs should be further evaluated.

Second, we applied our tag merging algorithm to the task of aligning headlines of movie reviews, news articles and movie titles from the cinema programme. The dataset was taken from the Lycos movie portal site. The setup of these experiments was slightly different: a pre-selection using a fast implementation of a spell checking algorithm was not needed due to the smaller dataset size. We removed domain-specific stop-words. Due to the different nature of the dataset, co-occurrence information was not available. As an additional distance metric, a soft-token TFIDF metric was used. As in the experiments described here, we used a rule learner to identify combination rules for the different similarity metrics and to learn a threshold. Additionally, we experimented with an SVM classifier. In these experiments we could achieve a precision of almost 100%.

4.3 Future Work

One of the next big challenges will be to align folksonomies with existing ontologies. As mentioned in section 1.2, we are currently investigating using web search to identify relationships between persons. In future work, we want to investigate how ratings by users can be used to identify high-quality tags. These tags could then be presented to a group of moderators or administrators to decide whether or not they should be added as concepts into a more stable ontology.

One of the clear hurdles towards a widespread adoption of ontologies have been usability problems related to visualising and interacting with such complex datasets. Specifically, we hope to guide the user towards real semantic tagging without alienating them. This process of gradually increasing ontology complexity has been described in literature as ontology evolution.

4.4 Acknowledgements

The research presented in this paper was partially funded by the German Federal Ministry of Economy and Technology (BMWi) under grant number 01MQ07008. The authors are solely responsible for the contents of this work.

We thank our colleagues at Lycos Europe who helped us with the manual evaluation of our tagging system.

References

1. Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. Dbpedia: A nucleus for a web of open data. In *The Semantic Web*, volume 4825 of *Lecture Notes in Computer Science*, pages 722–735, Berlin/Heidelberg, 2008. Springer.
2. Mikhail Bilenko and Raymond J. Mooney. Learning to combine trained distance metrics for duplicate detection in databases. Technical report, Dept. of Computer Science, University of Texas, Austin, Texas, USA, February 2002.
3. Philipp Cimiano, Günter Ladwig, and Steffen Staab. Gimme' the context: Context-driven automatic semantic annotation with c-pankow. In *Proc. of the 14th Int. World Wide Web Conference*, 2005.
4. Fabio Ciravegna, Sam Chapman, Alexiei Dingli, and Yorick Wilks. Learning to harvest information for the semantic web. In *Proceedings of the 1st European Semantic Web Symposium*, Heraklion, Greec, May 2004.
5. Stephen Dill, Nadav Eiron, David Gibson, Daniel Gruhl, R. Guha, Anant Jhingran, Tapas Kanungi, Sridhar Rajagopalan, Andrew Tomkins, John Tomlin, and Jason Zien. Semtag and seeker: Bootstrapping the semantic web via automated semantic annotation. In *Proc. of the 12th Int. World Wide Web Conference*, 2003.
6. Susan Dumais, John Platt, David Heckerman, and Mehran Sahami. Inductive learning algorithms and representations for text categorization. In *CIKM '98: Proceedings of the seventh international conference on Information and knowledge management*, New York, NY, USA, 1998. ACM.
7. Aidan Finn and Nicholas Kushmerick. Multi-level boundary classification for information extraction. In *Proc. European Conf. on Machine Learning*. Springer, 2004.

8. Gernot Gräfe, Christian Maaß, and Andreas Heß. Alternative searching services: Seven theses on the importance of social bookmarking. In *Proc. of the 1st Conf. on Social Semantic Web*, 2007.
9. Tom R. Gruber. A translation approach to portable ontologies. *Knowledge Acquisition*, 5(2):199–220, 1993.
10. Marieke Guy and Emma Tonkin. Folksonomies – tidying up tags? *D-Lib Magazine*, 12(1), January 2006.
11. M. Hepp, D. Bachlechner, and K. Siorpaes. Ontowiki: Community-driven ontology engineering and ontology usage based on wikis. In *Proceedings of the 2006 international symposium on Wikis*, 2006.
12. Andreas Heß, Rinat Khoussainov, and Nicholas Kushmerick. Ensemble learning with biased classifiers: The triskel algorithm. In *Proceedings of the 6th International Workshop on Multiple Classifier Systems*, Monterey Bay, California, USA, 2005.
13. Andreas Hotho, Robert Jäschke, Christoph Schmitz, and Gerd Stumme. Emergent semantics in bibsonomy. In Christian Hochberger and Rdiger Liskowsky, editors, *GI Jahrestagung*, volume 94 of *LNI*, pages 305–312, 2006.
14. Nicholas Kushmerick. *Wrapper induction for information extraction.* PhD thesis, University of Washington, 1997.
15. K. Lee. What goes around comes around: An analysis of del.icio.us as social space. In *Proc. of the 20th conf. on Computer supported cooperative work*, 2006.
16. Gang Luo, Chunqiang Tang, and Ying li Tian. Answering relationship queries on the web. In *Proc. of the 16th Int. World Wide Web Conference*, New York, NY, USA, 2007. ACM.
17. Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval.* Cambridge University Press, 2008.
18. G. A. Miller. The magical number seven, plus or minus two: Some limits on our capacity for processing information. *The Psychological Review*, 1(63):81–97, 1956.
19. Gilad Mishne. Autotag: a collaborative approach to automated tag assignment for weblog posts. In *Proc. of the 15th Int. World Wide Web Conference*, New York, NY, USA, 2006. ACM Press.
20. J. J. Rocchio. *The SMART Retrieval System: Experiments in Automatic Document Processing*, chapter Relevance feedback in information retrieval, pages 313–323. Prentice-Hall, Englewood Cliffs, NJ, USA, 1971.
21. Clay Shirky. Ontology is overrated: Categories, media & community. http://shirky.com/writings/ontology_overrated.html, 2006.
22. Sanjay C. Sood, Sra H. Owsley, Kristian J. Hammond, and Larry Birnbaum. Tagassist: Automatic tag suggestion for blog posts. In *Proceedings of the International Conference on Weblogs and Social Media*, Boulder, Colorado, USA, March 2007.
23. Grigorios Tsoumakas and Ioannis Katakis. Multi-label classification: An overview. *International Journal of Data Warehousing and Mining*, 3(3):1–13, 2007.
24. I. H. Witten and Eibe Frank. *Data Mining: Practical Machine Learning Tools with Java Implementations.* Morgan Kaufmann, San Francisco, 1999.