# A Comparison of content-based Tag Recommendations in Folksonomy Systems

Jens Illig,[1] Andreas Hotho,[1] Robert Jäschke,[1,2] Gerd Stumme[1,2]

[1] Knowledge & Data Engineering Group, Department of Mathematics and Computer Science, University of Kassel, Wilhelmshöher Allee 73, 34121 Kassel, Germany
http://www.kde.cs.uni-kassel.de/
[2] Research Center L3S, Appelstraße 9a, 30167 Hannover, Germany
http://www.l3s.de/

**Abstract.** Recommendation algorithms and multi-class classifiers can support users of social bookmarking systems in assigning tags to their bookmarks. Content based recommenders are the usual approach for facing the cold start problem, i. e., when a bookmark is uploaded for the first time and no information from other users can be exploited. In this paper, we evaluate several recommendation algorithms in a cold-start scenario on a large real-world dataset.

## 1 Introduction

Social bookmarking systems allow web surfers to store and manage their bookmarks on a central server and not as usual within the browser. They allow thus to access bookmarks simultaneously from different computers and to share them with other users. The user has the possibility to assign freely chosen keywords, so-called *tags* to each resource, which can be used to structure and retrieve the stored bookmarks. To support the user in tagging, different types of recommendation algorithms are typically utilized by bookmarking systems.

The recommendation of tags can also be considered as a classification problem, since we can consider each tag as the name of a class. More precisely, we talk about a *multi-label classification* problem [25], since users typically assign more than one tag to a resource. The number of classes is typically very high as folksonomy users are allowed to choose from as many different tags as they like.

There are two typical approaches to the recommendation problem: content-based approaches and collaborative filtering approaches [3]. While the former rely solely on the content of the documents, the latter take into account the behavior of similar users. Social bookmarking systems are an ideal scenario for the collaborative filtering approach, as the similarity of users can be measured by comparing their tagging behavior. Nevertheless the so-called *cold start problem* also occurs in social bookmarking systems: When a resource is tagged for the first time by some user, all other users – and in particular those who are similar to him – do not yield any recommendation about which tags to use. Therefore, content-based recommendations also have their use in social bookmarking systems.

In this paper, we study different content-based recommenders, and compare them on a real-world dataset – a crawl of the delicious bookmarking system.[3] The main con-

[3] http://delicious.com/

tribution is a comparison of state of the art recommenders, the adaption of classifiers to this problem and a demonstration that content based recommenders are able to generalize and to make predictions for new web pages. The paper complements our work on collaborative filtering approaches [14]. A more detailed discussion of its findings can be found in the bachelor thesis [13] of Jens Illig.

The paper is organized as follows. In Section 2, we introduce folksonomies, the underlying data structure of social bookmarking systems. In Section 3, we discuss related work. Section 4 introduces the problem definition and describes the classifiers that we used. Section 5 describes the data set that we used, and the preprocessing that we performed. We discuss our findings in Section 6 and future work in Section 7.

## 2 Social Resource Sharing and Folksonomies

The central data structure of a social bookmarking system is a *folksonomy*. It consists of the assignments of tags to resources by some users. The following definition, taken from [12], formalizes this idea:[4]

**Definition 1 (Folksonomy).** *A folksonomy is a tuple $\mathbb{F} := (U, T, R, Y)$ where*

- *$U$ is a finite set of users,*
- *$T$ is a finite set of tags,*
- *$R$ is a finite set of resources, and*
- *$Y \subseteq U \times T \times R$ is a ternary relation between users, tags, and resources. An element $(u, t, r)$ of $Y$ is called a* tag assignment (TAS) *and represents the fact that user $u$ has assigned tag $t$ to resource $r$.*

*The set of tags that user $u$ has assigned to resource $r$ is given by $T_{ur} := \{t \in T \mid (u, t, r) \in Y\}$. If $T_{ur}$ is non-empty, then we call the tuple $(u, T_{ur}, r)$ the* post *of user $u$ for resource $r$.*

Note that the set $T$ of tags may grow over time, as there are no pre-defined catchwords – the user is free to come up with arbitrary new tags. A resource is usually labeled by multiple users and tags may be assigned multiple times to the same resource by different users.

For content-based recommendations, we will abstract from the user dimension. Therefore, we introduce the set of *binary tag assignments (BTAS)* as projection $I$ of $Y$ on the tag and resource dimensions: $I := \{(t, r) \in T \times R \mid \exists u \in U : (u, t, r) \in Y)\}$. If $T_r := \{t \in T \mid (t, r) \in I\}$ is non-empty, then we call the tuple $(T_r, r)$ the *bpost* for resource $r$.

## 3 Related Work

General overviews on the rather young area of folksonomy systems and their strengths and weaknesses are given in [11,18,19]. In [20], Mika defines a model of semantic-social networks for extracting lightweight ontologies from delicious. Recently, work on

[4] In [12], we have additionally introduced a user-specific sub-tag/super-tag relation, which we will ignore for the purpose of this paper.

more specialized topics, such as structure mining on folksonomies – e. g. to visualize trends [8] and patterns [23] in users' tagging behavior – as well as ranking of folksonomy contents [12], analyzing the semiotic dynamics of the tagging vocabulary [5], or the dynamics and semantics [10] have been presented.

The literature concerning the problem of tag recommendations in folksonomies is still sparse. The existent approaches usually lay in the collaborative filtering and information retrieval areas. In [21], [4], and [14], algorithms for tag recommendations are devised based on content-based filtering techniques. Xu et al. [29] introduce a collaborative tag suggestion approach based on the HITS algorithm [16]. A goodness measure for tags, derived from collective user authorities, is iteratively adjusted by a reward-penalty algorithm. Benz et al. [2] introduce a collaborative approach for bookmark classification based on a combination of nearest-neighbor-classifiers. There, a keyword recommender plays the role of a collaborative tag recommender, but it is just a component of the overall algorithm, and therefore there is no information about its effectiveness alone. Basile et al. [1] suggest an architecture for an intelligent recommender tag system. In [9,28,27], the problem of tag-aware resource recommendations is investigated. The standard tag recommenders, in practice, are services that provide the most-popular tags used for a particular resource. This is usually done by means of tag clouds where the most frequent used tags are depicted in a larger font or otherwise emphasized.

First work which utilized machine learning algorithms to predict tags based on the content is reported in [25]. The reported results for four real world dataset are very promising but limited to only two models, a new gaussian process and an SVM model. Results for a vector space model and a poisson mixture model are reported in [26]. The results are similar to those we report here for other machine learning methods.

Most recently, the ECML/PKDD 2008 Discovery Challenge[5] has addressed the problem of tag recommendations in folksonomies. Most of them relies on a combination of good preprocessing, some external knowledge sources and a good heuristic to choose the right set of tags. No machine learning approach was used.

## 4 Tag Recommendations as Text Classification Problem

### 4.1 Definition of the Problem

In [14], we have studied tag recommendations based on a collaborative filtering approach. But in a dynamic setting, such as our web bookmarking scenario, new web pages show up frequently. When a new page is bookmarked for the first time, the only information about it is its full text. Our aim is to learn tag recommendations that are based on this information.

We formalize the problem as follows. Let $\mathbb{F} := (U, T, R, Y)$ be a folksonomy, where the set $R$ of resources consists of web pages. The web pages are modeled by the *bag of words* approach, i. e., a mapping $\mathrm{vec}\colon R \rightarrow \mathbb{R}^V$, where $V$ is the set of all[6] words occurring in at least one document, and where $\mathrm{vec}(r)_v$ is the number of occurrences of word $v$ on web page $r$. We applied tf-idf weighting to that mapping.[7]

[5] http://www.kde.cs.uni-kassel.de/ws/rsdc08/ [6] In this paper, we did not apply stopword removal. [7] We also made the same classification experiments without tf-idf weighting but the best results of every classifier family were achieved with tf-idf.

For the evaluation, we assume that the folksonomy $\mathbb{F}$ is split into a training and a test set, i. e., into:[8]

$$\mathbb{F}_{\text{train}} = (U_{train}, T_{train}, R_{train}, Y_{train}) \quad \text{and} \quad \mathbb{F}_{\text{test}} = (U_{test}, T_{test}, R_{test}, Y_{test})$$

The *problem of learning tag recommendations*, consists in finding, based on the information in $\mathbb{F}_{\text{train}}$ and for given $n \in \mathbb{N}$, a function $\varphi_n \colon \mathbb{R}^V \to \mathfrak{P}_n(T)$,[9] such that, for all resources $r$ in $\mathbb{F}_{\text{test}}$, $\varphi_n(\text{vec}(r))$ is a good approximation for the tags of $r$. As usual, we will measure the quality of the approximation with precision and recall, see Section 6.1.

## 4.2 Classifiers

In order to solve the problem of finding a concrete mapping $\varphi_n$, we applied different machine learning algorithms, which are suitable for the text classification task (cf.[24]). In the experiments, we compared the following models: SVM, multinominal naïve Bayes, Rocchio, $k$-NN, and – as a simple baseline – the most popular tags for the document. All models provide at the end a function $\breve{\Phi}_t^{\neg t} \colon \mathbb{R}^V \to \mathbb{R}$ that is returning, for $\vec{x} \in \mathbb{R}^V$, a confidence value $\breve{\Phi}_t^{\neg t}(\vec{x})$ describing how confident the model is in assigning tag $t$ to a resource $r \in R$ with $\text{vec}(r) = \vec{x}$. The recommendation $\varphi_n(r)$ then consists of those $n$ tags $t \in T$ having the highest values $\breve{\Phi}_t^{\neg t}(\text{vec}(r))$.

The functions $\breve{\Phi}_t^{\neg t}$ are either computed directly – this approach is called $t$-vs-$\neg t$ or one-vs-all – or calculated from multiple confidence values of pairwise tag comparisons $\breve{\Phi}_x^y$ where $\breve{\Phi}_x^y(\text{vec}(r))$ is the confidence in the decision to prefer tag $x$ instead of tag $y$ for resource $r$. The latter approach is called one-vs-one. For all learning algorithms except $k$-Nearest-Neighbor where only one-vs-all has been applied, we experimented both with one-vs-all and one-vs-one.

For one-vs-one, we evaluated two different variants for calculating a single confidence function $\breve{\Phi}_t^{\neg t}$ from all confidence functions $\{\breve{\Phi}_x^y \mid x \in T_{\text{train}} \wedge y \in T_{\text{train}} \wedge x \neq y \wedge (x = t \vee y = t)\}$. The first uses simple boolean vote adding and requires hard classifications for every tag-vs-tag pair to increase a vote counter for the winning tag of the pair. Confidence threshold zero has been used to get this hard classification which is motivated by the fact that most of the tested classifiers are directed to output confidence values with positive or negative values for indicating preference of $tag$ in favor of $\neg tag$ respectively tag $x$ in favor of tag $y$. A confidence value exactly equal to zero leads to no vote for any of the two tags in the one-vs-one pair.
The other tested variant of defining $\breve{\Phi}_t^{\neg t}$ uses confidence adding:

$$\breve{\Phi}_t^{\neg t} \colon \mathbb{R}^V \to \mathbb{R}; \quad \vec{x} \mapsto \sum_{t' \in T_{\text{train}} \setminus \{t\}} \breve{\Phi}_t^{t'}(\vec{x}) - \breve{\Phi}_{t'}^{t}(\vec{x}) \tag{1}$$

The algorithms follow the same principle for computing the functions $\breve{\Phi}_t^{\neg t}$ and the functions $\breve{\Phi}_x^y$: Let 0 and 1 stand for $\neg t$ and $t$, resp., in the first case, and for tag $x$ and tag $y$, resp., in the second case. Given a set $\mathbb{V}_{\text{train}} = \{\text{vec}(r) \mid r \in R_{\text{train}}\}$ and a function $\Phi_1^0 \colon \mathbb{V}_{\text{train}} \to \{0, 1\}$, all of these algorithms find a function $\breve{\Phi}_1^0 \colon \mathbb{R}^V \to \mathbb{R}$ which maps

[8] The specific splitting approach that we used for this paper is described in Section 5.2.
[9] $\mathfrak{P}_n(T)$ stands for the set of all subsets of $T$ with exactly $n$ elements.

to real valued confidence values indicating how much more suitable decision 1 is in favor of decision 0 for a feature vector in $\mathbb{R}^V$ regarding an internal model learned from training examples.

**SVM.** Support Vector Machines are classifiers that separate the feature hyperspace of some dimension $|V|$ into two subspaces divided by a $|V| - 1$ dimensional hyperplane. Thereby SVMs also try to find a hyperplane position that provides a broad 'safety' space around the hyperplane instead of simply focussing on a small training error rate.

As used for example in [22], two parameters, $C^+ \in \mathbb{R}$ and $C^- \in \mathbb{R}$ define the relative importance of consistency with positive and negative training examples against safety space maximization. For the experiments with the SVM machine learning method, a marginally modified implementation of the linear C-SVM algorithm from the library libSVM [7] has been used that, instead of hard classifications, outputs its internal hyperplane distance as confidence values. We experimented both with the default setting $C = C^+ = C^- = 1$ and a second variant using

$$C^- = \frac{|\{r \in R_{\mathrm{train}} \mid \mathit{\Phi}_1^0(\mathrm{vec}(r)) = 0\}|}{2 \cdot |\{r \in R_{\mathrm{train}} \mid \mathit{\Phi}_1^0(\mathrm{vec}(r)) = 1\}|} \quad \text{together with} \quad C^+ = 2 \cdot {C^-}^2$$

This asymmetric setting (which is marked as C= $+/-$ in the evaluation section) is motivated by the observation that a negative resource/tag example can either be a 'real' negative example (i. e., the tag indeed does not fit to the resource), or a 'missed' positive example (i. e., the tag semantically belongs to the resource, but has not yet been assigned explicitly to it by any of the users of the system). Thus, the cost of misclassifying a positive training example ($C^+$) should be higher than the cost of misclassifying a negative example. However, setting $C^+$ too high in relation to $C^-$ may lead to a trivial positive classifier. The above given settings of $C^-$ and $C^+$ have been determined on the basis of multiple small manually constructed two-dimensional test datasets. Experiments have been conducted with and without scaling all document feature vectors to an Euclidean length of one before training and classification (denoted by *lnorm* and *nolnorm*, resp., in the evaluation section).

**Multinomial Naïve Bayes.** This classification applied to tag categorization calculates a probability estimate $P(t|r)$ for the observation of tag $t$ given an observation of a resource $r$. We used the log odds ratio based on a multinomial model with document model based parameter estimation as described in [15], which leads to [10]

$$\breve{\mathit{\Phi}}_t^{\neg t}(\mathrm{vec}(r)) = log\left(\frac{P(t|r)}{P(\neg t|r)}\right) = log\left(\prod_{v \in r}\left(\frac{P(v|t)}{P(v|\neg t)}\right)^{\mathrm{vec}(r)_v} \cdot \frac{P(t)}{P(\neg t)}\right) \quad (2)$$

$$\text{with} \quad P(v|t) = \sum_{r' \in R_{\mathrm{train}}} P(r'|t) \cdot P(v|r') \ . \quad (3)$$

We estimated $P(r'|t)$, $P(v|r')$ and $P(t)$ as well as $\neg t$ variants thereof directly from the relative TAS and term occurrence frequencies in the training corpus. To avoid

[10] We use $v \in r$ here for $v \in \{v' \mid v' \in V \wedge \mathrm{vec}(r)_{v'} > 0\}$.

$P(v|t) = 0$ as a factor in the right term of Equation 2, a virtual post $p^\star = \{u^\star\} \times T_{\text{train}} \times \{r^\star\}$ has been added to the training dataset. $r^\star$ is made up of one occurrence of every feature known from the training dataset plus a virtual wildcard feature. During classification, each new feature not known from the training dataset has been treated equally to that wildcard feature.

**Rocchio.** This centroid based method builds class representation vectors that are compared to resource representation vectors in order to find some similarity measure as the confidence output value. As presented for example in [24], we calculated positive and negative centroid vectors for the training classes 0 and 1 as follows

$$\vec{c}^{\,1} = \frac{1}{|R^1_{\text{train}}|} \sum_{r_{\text{train}} \in R^1_{\text{train}}} \text{vec}(r_{\text{train}}) \qquad \vec{c}^{\,0} = \frac{1}{|R^0_{\text{train}}|} \sum_{r_{\text{train}} \in R^0_{\text{train}}} \text{vec}(r_{\text{train}})$$

With these centroids we define $\breve{\Phi}$ as follows

$$\breve{\Phi}^0_1(\text{vec}(r)) = cos\left(\sphericalangle\left(\beta\frac{\vec{c}^{\,1}}{\|\vec{c}^{\,1}\|} - \gamma\frac{\vec{c}^{\,0}}{\|\vec{c}^{\,0}\|}, \text{vec}(r)\right)\right)$$

Classifier setups have been evaluated with $\beta = 1$ in combinations with both $\gamma = 0$ and $\gamma = 1$. Additionally we experimented with TAS weighted centroids, but yielded slightly lower effectiveness. Furthermore, our experiments with Euclidean distance always led to effectiveness below the baseline.

***k*-NN.** We have run the $k$-Nearest-Neighbor method considering the 30 nearest neighbor documents and using a confidence calculation scheme taken from [24], that is

$$\breve{\Phi}^0_1(\text{vec}(r)) = \sum_{r_{\text{train}} \in MostSim_k} f_{\text{sim}}\left(\text{vec}(r), \text{vec}(r_{\text{train}})\right) \cdot \Theta^0_1(r_{\text{train}})$$

$$\Theta^0_1(r_{\text{train}}) = \begin{cases} 1 & \text{, if } \Phi^0_1(\text{vec}(r_{\text{train}})) = 1 \\ 0 & \text{, otherwise} \end{cases}$$

Here, $MostSim_k \subseteq R_{\text{train}}$ is the set of those training instances that are among the $k$ most similar instances compared by similarity measure $\text{sim}\colon \mathbb{R}^V \times \mathbb{R}^V \to \mathbb{R}$ to the argument instance $r$ which is to be classified. We used $\text{sim}(x, y) = cos\left(\sphericalangle(x, y)\right)$. For the similarity weighting function $f_{\text{sim}}$, both $f_{\text{sim}}(x, y) = 1$ and $f_{\text{sim}}(x, y) = sim(x, y)$ have been evaluated. Additionally, we experimented with an alternative definition of $\Theta^0_1(r_{\text{train}})$ which also takes into account how many users assigned a tag to a resource in the training set:

$$\Theta^{\neg t}_t(r_{\text{train}}) = \begin{cases} \log\left(|\{u \in U_{\text{train}} \mid (u, t, r_{\text{train}}) \in Y_{\text{train}}\}| + 2\right) & \text{, if } \Phi^{\neg t}_t = t \\ -\log\left(|TAS^{\neg t}_{\text{train}}(r_{\text{train}})|\right) & \text{, if } \Phi^{\neg t}_t = \neg t \text{ and } |TAS^{\neg t}_{\text{train}}(r_{\text{train}})| \geq 1 \\ 0 & \text{, otherwise} \end{cases}$$

where $\quad TAS^{\neg t}_{\text{train}}(r_{\text{train}}) = \{(u, t') \in U_{\text{train}} \times (T_{\text{train}} \backslash \{t\}) \mid (u, t', r_{\text{train}}) \in Y_{\text{train}}\}$ .

The logarithm is used for damping and $+2$ is used to slightly linearize the logarithm in order to weight positive neighbors strongly even with few TAS.

## 5 Evaluation Setting

The dataset used for the experiments is a crawl of the social bookmarking system delicious downloaded between 2005-07-27 and 2005-07-30 [12]. It consists of $75,242$ users, $533,191$ tags and $3,158,297$ resources, related by in total $17,362,212$ tag assignments. The full text of all $3,158,297$ resources had also been downloaded in 2005. Unfortunately, the protocol response headers of the resource downloads were lost. For that reason it was at first unclear how many resources were error code pages and which resources were correctly transferred resources. There was also no information about the MIME type of the resources, the encoding, or the language in case of text resources.

### 5.1 Preprocessing

Based on MIME type detections of the magic byte sequence algorithm from the "data and metadata getting" Java framework *Aperture*[11] in version 1.0.1-beta, all resources for which the detected MIME type neither started with "text" nor contained the substring "html" have been filtered out. In order to escape all character set problematics, the document corpus has been restricted to 7 bit ASCII encoded documents which have been detected by application of the *jchardet*[12] library, which is a Java port of the Mozilla universal charset detector [17]. All pages estimated being erroneous in terms of a non HTTP 2xx response have been pruned. Such documents were identified by a SVM trained by a set of $1,271$ successfully and $1,000$ unsuccessfully re-crawled resources. A ten fold cross-validation of this classifier showed an eleven point average precision of $0.96$.

Primarily because of the tokenization problem with natural text of some languages, but also with respect to possible comparable future stemming experiments on the same dataset, only English text documents were evaluated. Language guessing was done by making use of the n-gram method [6] applied via the character based part of the *ngramj* library[13]. Whenever documents contained explicit information about their language, we doubled the score of that language.

The pruning steps (pruning of error pages, non-text/html text, non-English documents and non-7-bit ASCII) reduced our initial folksonomy. We removed then all users and all resources that were no longer related to a resource. We obtained in total $65,177$ users, $299,305$ tags, and $1,113,405$ resources.

Figure 1 shows the frequency distribution of the tags. Each cross is representing one tag. The right-most cross (which is located on the $x$-axis) says that there is exactly one ($= 10^0$) tag that occurs in $35,307$ BTAS, while the left-most cross (which is located on the $y$-axis) says that there are $158,183$ different tags that occur in only one BTAS each. Since such rare tags are very difficult to predict, and since we had a variety of algorithms and parameter settings that we wanted to evaluate, we had to reduce the data further. Hence, we restricted the set of tags to the 15 most popular tags, in order to reduce the complexity of the learning problem.[14] The remaining folksonomy $\mathbb{F}$ consists

[11] http://aperture.sourceforge.net/ [12] http://jchardet.sourceforge.net/

[13] http://ngramj.sourceforge.net/ [14] Faced with the limitation of our computing machinery (an Opteron PC with $8 \times 2$ GHz and $32$ GB main memory), we had to decide whether to include more different algorithms, or to run a more extensive comparison of fewer algorithms. We decided to go for the former.

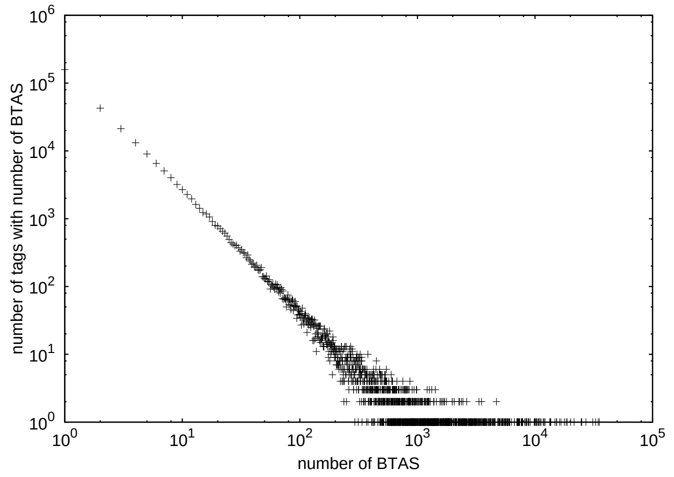

**Fig. 1.** Tag frequencies for all resources. The fact that the data points almost form a line hints at the presence of a so-called *power law* distribution, which is typical for many human-driven activities.

of $65,177$ users, $15$ tags, and $1,113,405$ resources. Users and resources that are not related to any of the 15 most frequent tags have not been deleted, as they were used as negative training data.

The remaining preprocessing steps generate the vector space representation $\mathrm{vec}\colon R \to \mathbb{R}^V$ of the full text of the web pages. For (X)HTML documents, a parser has been used that passes through all non-markup as long as it is located inside of one of the HTML-tags `head` or `body` and outside of all the HTML-tags `embed, object, style, applet,` and `script`. The parser has been configured to filter out documents containing the `frameset` HTML-tag.

Two types of document features have been extracted from text documents during the build of bag of words feature vector representations. The first type of feature is tokenized text with terms and character sequences. A tokenizer has been implemented which distinguishes between numbers, terms, special character sequences, and single interesting special characters. All of these tokens are found independently of one another, but because of the definitions of the distinguished token types, no token occurrence is accounted more than once. Single character occurrences may be accounted in multiple features. All tokens were lowercased before frequency counting. For the other feature type, occurrences of the HTML-tags `img, a, code, p, object, applet, embed, form, cite, dfn, q, samp` have been counted separately as features. We did not apply stopword removal.

Then the vector space representation of the documents has been built, as described in Section 4.1.

### 5.2 Training and Test Datasets

We follow the typical evaluation setting for supervised learning tasks by splitting the available data, i. e., the folksonomy $\mathbb{F}$ that resulted from the preprocessing as described above, into a training and a test data set. The split is based on the date of the posts. All posts between 2003-10-01 and 2004-08-26 have been used for the training dataset $\mathbb{F}_{\text{train}}$, resulting in $4,236$ users,

For the set of test documents, we considered all $10,602$ documents that occurred in posts between 2004-08-27 and the end of 2004-09-05. From these, we removed all $2,417$ documents which also occurred in posts from the training set. By removing all documents that are in both the training and the test dataset, we avoid the problem of evaluating our approach on already seen data, which would bias the evaluation. All TAS after 2004-08-27 (including those after 2004-09-05) referring to the remaining $8,185$ test documents have been used to find a testset of BTAS. (By not limiting to posts before 2004-09-05, we extend the set of BTAS and can thus use the maximal available information for the evaluation.) From the remaining documents, we removed all documents that were not tagged by any of the $15$ most frequent tags. As an additional attempt to reduce the problem of suitable recommended tags that have not been assigned in our dataset, we limited the set to only those documents with at least ten TAS in the whole dataset. Again we removed unconnected users. The resulting test set $\mathbb{F}_{\text{test}}$ contains $40,632$ users,15 tags, and $1,926$ resources.

## 6 Experiments

### 6.1 Evaluation Settings

We evaluated all the recommenders on the test dataset $\mathbb{F}_{\text{test}}$. Since all recommendations were non-personalised, we projected out the user dimension of $\mathbb{F}_{\text{test}}$ – i. e., we considered the set $I_{\text{test}} \subseteq T_{\text{test}} \times R_{\text{test}}$ of BTAS only.

For evaluating recommender $\varphi$, we computed, for each resource $r$ in $R_{\text{test}}$ and for each $i$ between 1 and 5, a recommendation $\varphi_i(r)$, recommending thus between one and five tags. For each of these combinations, precision and recall were computed:

$$\text{precision}(\varphi_i, r) = \frac{|\varphi_i(r) \cap T_r|}{|\varphi_i(r)|} \qquad \text{recall}(\varphi_i, r) = \frac{|\varphi_i(r) \cap T_r|}{|T_r|}$$

For each recommender and for each $i = 1, \ldots, 5$, we averaged precision and recall over all resources in $R_{\text{test}}$:

$$\text{precision}(\varphi_i) = \frac{1}{|R_{\text{test}}|} \sum_{r \in R_{\text{test}}} \text{precision}(\varphi_i, r)$$

$$\text{recall}(\varphi_i) = \frac{1}{|R_{\text{test}}|} \sum_{r \in R_{\text{test}}} \text{recall}(\varphi_i, r)$$

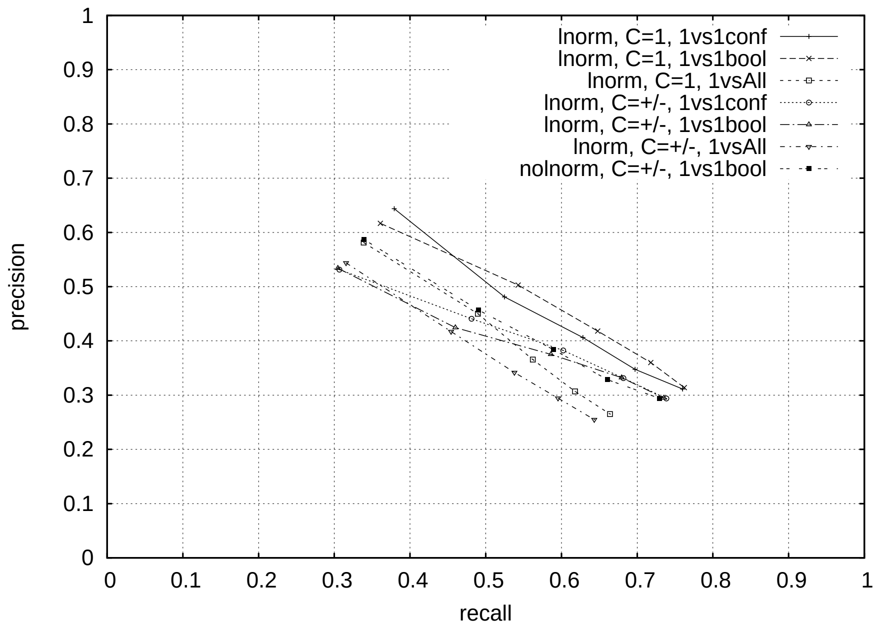

**Fig. 2.** Tf-idf precision and recall for different parameter settings of the SVM, averaged over all test documents not occuring in the training set that have at least 10 TAS in the whole dataset.

## 6.2 Comparison of the Classifiers

All classifiers were evaluated with several parameter settings, which, due to space restrictions, cannot be presented all. For more details, see the bachelor thesis [13] of Jens Illig.

Figure 2 shows the results for some settings of the SVM. The five recommendations $\varphi_1, \ldots, \varphi_5$ of each setting are plotted together in one curve. The left-most node of each curve represents the one-element-recommendation $\varphi_1$, while the right-most node represents the five-element-recommendation $\varphi_5$. As one can see, all curves are monotonically decreasing. This shows that, for all recommenders, recall is growing with an increasing number of recommendations while precision is falling.

The figure shows that the best settings for the SVM are those with C=1 parameterization. A possible explanation is that our dynamically calculated parameterization (called "C+/-" Figure 2) with its higher C values tends to overfit. This is supported by our observation that in another evaluation that is based solely on repeatedly posted documents, these classifiers show higher effectiveness than the corresponding C=1 variants. With the better working C=1 SVM configuration, Figure 2 also shows that the boolean adding one-vs-one variant is most effective at higher recall levels, while, with confidence adding, the first item can be recommended more precisely. This might be explained by real-valued confidence values of the confidence adding classifier variant where, in contrast, boolean adding uses only integer vote counts as confidence values,

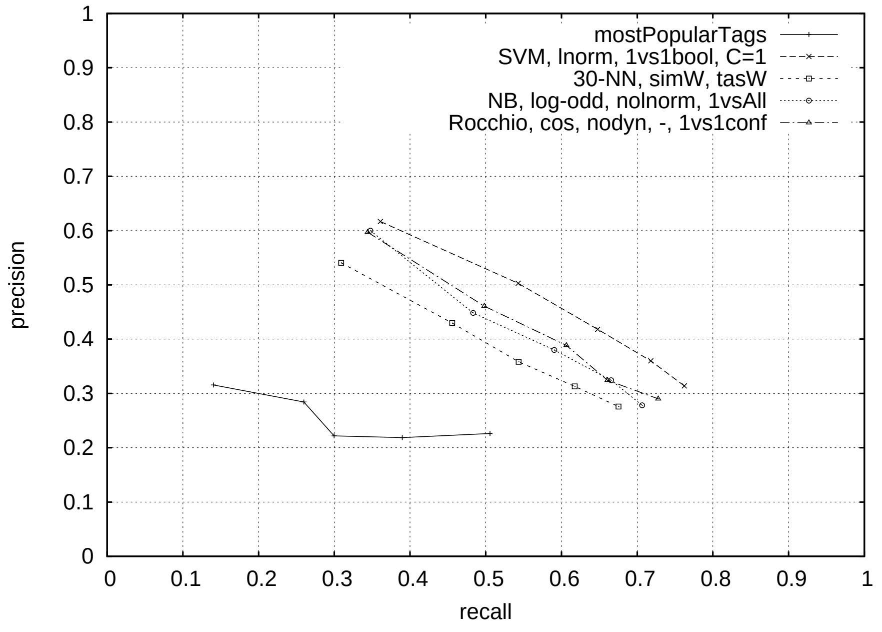

**Fig. 3.** Precision and recall of the best recommenders of different classifier types, averaged over all test documents that have at least 10 TAS in the whole dataset. All use tf-idf values.

which, as we observed, often leads to many tag suggestions with equal confidence output so that these cannot be ordered any further. Without length-normalization, SVM effectiveness is in most cases lower, especially for one-vs-all classifiers. The best non length-normalized variant is boolean adding one-vs-one with “C+/-”, but it only has around $0.02$ more precision than the C=1 variant at similar recall levels.

An overall comparison of all approaches is shown in Figure 3. For sake of readability, we did again not display all parameter settings, but only one or two of those that performed best for each classifier type. In the diagram, one can see that the SVM with one-vs-one learning is clearly more effective than the other classifiers. One-vs-one is also the best choice for the Rocchio method. Thereby, a confidence adding variant without TAS weighted centroids turned out to be most effective. However, our experiments showed that the worst cosine based Rocchio classifier is only about $0.04$ precision score points less effective at similar recall levels. Most clearly, those variants with $\gamma = 0$ were the less effective among them. Another well functioning classifier is log-odds ratio multinomial Naïve Bayes. For that classifier type, length normalization has turned out to be counterproductive for tag recommendation. 30-NN is clearly less effective. The best 30-NN variants use our TAS weighting scheme, which seems to increase precision by ca. 0.04. Similarity weighting also slightly increased precision. As expected, simple most popular tag recommendation is obviously less effective than almost all content based methods – only the highly ineffective Rocchio methods with Euclidean distance (not displayed in Figure 3) are worse.

## 7 Conclusion and Outlook

In this paper, we evaluated the effectiveness of multiple text classification methods and variants applied to a scenario that is compatible with the common text classification evaluation practice of disjoint training and test scenarios but still represents a realistic and pure cold start tag recommender evaluation scenario. Thereby, we identified a problem in the open world characteristic of the dataset and developed an evaluation scheme that addresses it.

Some algorithms have been slightly modified in various ways to make use of tag assignment frequencies by multiple users. Improvements by these extensions have been detected for the case of a TAS weighted 30-Nearest-Neighbors algorithm. Nevertheless, we found that an one-vs-one SVM variant on length normalized document feature vectors is the most effective of all evaluated classifiers. We could show that folksonomy tag assignments can be learned by application of machine learning techniques to address the cold start problem of collaborative recommender systems.

In the future, our experiments can be extended to other classifier algorithms, like for example boosting, decision trees, and rule based learners. Also transductive approaches seem promising in terms of the open world problem. Other possible extensions include stemming, term space reduction, different feature reweighting methods, and classification of documents in multiple languages.

Another open task is to evaluate and compare the effectiveness of content based and collaborative approaches (on a test set of already posted resources). The next step is then to develop combined approaches that rely on both the high effectiveness of collaborative methods on documents with known tag assignments and the strengths of content based approaches to overcome the cold start problem.

*Acknowledgement.* Part of this research was funded by the European Commission in the projects Nepomuk (FP6–027705) and TAGora (FP6–IST5–34721).

## References

1. Pierpaolo Basile, Domenico Gendarmi, Filippo Lanubile, and Giovanni Semeraro. Recommending smart tags in a social bookmarking system. In *Bridging the Gap between Semantic Web and Web 2.0 (SemNet 2007)*, pages 22–29, 2007.
2. D. Benz, K. Tso, and L. Schmidt-Thieme. Automatic bookmark classification: A collaborative approach. In *Proceedings of the Second Workshop on Innovations in Web Infrastructure (IWI 2006)*, Edinburgh, Scotland, 2006.
3. Robin Burke. Hybrid recommender systems, survey and experiments. *User Modeling and User Adapted Interaction*, 12(4):331–370, 2002.
4. Andrew Byde, Hui Wan, and Steve Cayzer. Personalized tag recommendations via tagging and content-based similarity metrics. In *Proceedings of the International Conference on Weblogs and Social Media*, Boulder, Colorado, USA, March 2007.

5. Ciro Cattuto, Vittorio Loreto, and Luciano Pietronero. Collaborative tagging and semiotic dynamics, May 2006. http://arxiv.org/abs/cs/0605015.
6. William B. Cavnar and John M. Trenkle. N-gram-based text categorization. In *Proceedings of SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval*, pages 161–175, Las Vegas, US, 1994.
7. Chih-Chung Chang and Chih-Jen Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.
8. M. Dubinko, R. Kumar, J. Magnani, J. Novak, P. Raghavan, and A. Tomkins. Visualizing tags over time. In *Proc. of the 15th International WWW Conference*, Edinburgh, Scotland, 2006.
9. Claudiu S. Firan, Wolfgang Nejdl, and Raluca Paiu. The benefit of using tag-based profiles. In *5th Latin American Web Congress, October 31 - November 2 2007, Santiago de Chile*, 2007.
10. H. Halpin, V. Robu, and H. Shepard. The dynamics and semantics of collaborative tagging. In *Proceedings of the 1st Semantic Authoring and Annotation Workshop (SAAW'06)*, Atlanta, Georgia, USA, 2006.
11. Tony Hammond, Timo Hannay, Ben Lund, and Joanna Scott. Social Bookmarking Tools (I): A General Review. *D-Lib Magazine*, 11(4), April 2005.
12. Andreas Hotho, Robert Jäschke, Christoph Schmitz, and Gerd Stumme. Information retrieval in folksonomies: Search and ranking. In York Sure and John Domingue, editors, *The Semantic Web: Research and Applications*, volume 4011 of *Lecture Notes in Computer Science*, pages 411–426, Heidelberg, June 2006. Springer.
13. Jens Illig. Machine learnability analysis of textclassifications in a social bookmarking folksonomy. Bachelor thesis, University of Kassel, Kassel, 2008.
14. Robert Jäschke, Leandro Marinho, Andreas Hotho, Lars Schmidt-Thieme, and Gerd Stumme. Tag recommendations in social bookmarking systems. *AI Communications*, 21(4):231–247, 2008.
15. S. Kim, H. Rim, D. Yook, and H. Lim. Effective methods for improving naive bayes text classifiers, 2002.
16. Jon M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632, 1999.
17. Shanjian Li and Katsuhiko Momoi. A composite approach to language/encoding detection. In *19th International Unicode Conference*, San Jose, California, USA, 2001.
18. Ben Lund, Tony Hammond, Martin Flack, and Timo Hannay. Social Bookmarking Tools (II): A Case Study - Connotea. *D-Lib Magazine*, 11(4), April 2005.
19. Adam Mathes. Folksonomies – Cooperative Classification and Communication Through Shared Metadata, December 2004. http://www.adammathes.com/academic/computer-mediated-communication/folksonomies.html.
20. Peter Mika. Ontologies Are Us: A Unified Model of Social Networks and Semantics. In Yolanda Gil, Enrico Motta, V. Richard Benjamins, and Mark A. Musen, editors, *ISWC 2005*, volume 3729 of *LNCS*, pages 522–536, Berlin Heidelberg, November 2005. Springer-Verlag.
21. Gilad Mishne. Autotag: a collaborative approach to automated tag assignment for weblog posts. In *WWW '06: Proceedings of the 15th international conference on World Wide Web*, pages 953–954, New York, NY, USA, 2006. ACM Press.
22. Katharina Morik, Peter Brockhausen, and Thorsten Joachims. Combining statistical learning with a knowledge-based approach - a case study in intensive care monitoring. In Ivan Bratko and Saso Dzeroski, editors, *Proceedings of the 16th International Conference on Machine Learning (ICML 1999), June 27-30, 1999, Bled, Slovenia*, pages 268–277. Morgan-Kaufman Publishers, San Francisco, CA, USA, 1999.

23. Christoph Schmitz, Andreas Hotho, Robert Jäschke, and Gerd Stumme. Mining association rules in folksonomies. In V. Batagelj, H.-H. Bock, A. Ferligoj, and A. Žiberna, editors, *Data Science and Classification: Proc. of the 10th IFCS Conf.*, Studies in Classification, Data Analysis, and Knowledge Organization, pages 261–270, Berlin, Heidelberg, 2006. Springer.
24. F. Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1):1–47, 2002.
25. Yang Song, Lu Zhang, and C. Lee Giles. A sparse gaussian processes classification framework for fast tag suggestions. In *CIKM '08: Proceeding of the 17th ACM conference on Information and knowledge mining*, pages 93–102, New York, NY, USA, 2008. ACM.
26. Yang Song, Ziming Zhuang, Huajing Li, Qiankun Zhao, Jia Li, Wang-Chien Lee, and C. Lee Giles. Real-time automatic tag recommendation. In *SIGIR '08: Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 515–522, New York, NY, USA, 2008. ACM.
27. Karen Tso-Sutter, Leandro Balby Marinho, and Lars Schmidt-Thieme. Tag-aware recommender systems by fusion of collaborative filtering algorithms. In *Proceedings of 23rd Annual ACM Symposium on Applied Computing (SAC'08)*, Edinburgh, Scotland, 2007.
28. Yanfei Xu, Liang Zhang, and Wei Liu. Cubic analysis of social bookmarking for personalized recommendation. *Frontiers of WWW Research and Development - APWeb 2006*, pages 733–738, 2006.
29. Zhichen Xu, Yun Fu, Jianchang Mao, and Difu Su. Towards the semantic web: Collaborative tag suggestions. In *Proceedings of the Collaborative Web Tagging Workshop at the WWW 2006*, Edinburgh, Scotland, 2006.