A novel supervised learning algorithm and its use for spam detection in social bookmarking systems

Anestis Gkanogiannis
Athens University of Economics and Business
9 PUBLICATIONS **64** CITATIONS
SEE PROFILE

Theodore Kalamboukis
Athens University of Economics and Business
34 PUBLICATIONS **370** CITATIONS
SEE PROFILE

ECML PKDD Discovery Challenge 2008

(RSDC'08)

International Workshop at
the European Conference on Machine Learning and
Principles and Practice of Knowledge Discovery in Databases
in Antwerp, Belgium, September 15th, 2008.

Preface

Since 1999 the ECML/PKDD embraces the tradition of organizing a Discovery Challenge, allowing researchers to develop and test algorithms for novel and real world datasets. This year's Discovery Challenge[1] presents a dataset from the field of social bookmarking to deal with two important tasks in this area: the detection of spam and the recommendation of tags. The results submitted by the challenge's participants are presented at an ECML/PKDD workshop on September 15th, 2008, in Antwerp.

The dataset provided has been created using data of the social bookmark and publication sharing system BibSonomy[2], operated by the organizers of this challenge. The training data was released on May 5th 2008, the test data on July 30th. The participants had time until August 1st to submit their results. This gave researchers 12 weeks time to tune their algorithms on a complete snapshot of a real world folksonomy dataset and 48 hours to compute results on the test data.

Spam Detection in Social Bookmarking Systems. With the growing popularity of social bookmarking systems, spammers use this service as a playground for their activities. Usually, they pursue two goals when placing links in the system: Attracting people to advertising sites and increasing the PageRank of their sites by collecting links in as many popular Web 2.0 sites as possible. A high PageRank leads to an increase of visibility in Google and other search engines. Common spam counter-measures such as captchas do not sufficiently prevent spammers from misusing the system. In our system, we were able to collect data of more than 2,000 active users and (despite the implementation of captchas for the registration of new users) more than 25,000 spammers by manually labelling spammers and non-spammers in the system. The challenge's dataset consists of these users and their posts. This includes all public information such as the URL, the description and all tags of the post. The goal of this challenge is to learn a model which predicts whether a user is a spammer or not. In order to detect spammers in a running system as early as possible, the model should make accurate predictions about users even if they only have few posts.

Tag Recommendation in Social Bookmarking Systems. To support the user during the tagging process and to facilitate the tagging, BibSonomy includes a tag recommender. When a user finds an interesting web page (or publication) and posts it to BibSonomy, the system offers up to ten recommended tags on the posting page. The goal of the recommendation task is to learn a model which effectively predicts the keywords a user has in mind when describing a web page (or publication).

[1] `http://www.kde.cs.uni-kassel.de/ws/rsdc08/`

[2] `http://www.bibsonomy.org`

Results. More than 150 participants registered for the mailing list which enabled them to look at the dataset. At the end, we received 18 submissions — 13 for the spam detection task and 5 for the tag recommendation task. Thirteen participants additionally submitted a paper — 11 of those were accepted and can be found in the proceedings at hand.

The 13 submitted solutions of the spam competition were evaluated by computing the AUC value [1]. The winning team has an AUC value of 0.98. Only two results were below the random baseline, which shows the high quality of the submissions. The proposed approaches varied between those heavily reliant on feature engineering and approaches comparing many different machine learning methods (among them kNN, SVM, Neural Networks, Naive Bayes Classifier or Regression Models).

The winners, A. Gkanogiannis and T. Kalamboukis from Athens University, use a text classification approach with a classifier which refines the solution similar to approaches known from the area of relevance feedback. They consider all posts of a user as a unified item or as a kind of "text document", which is further preprocessed and used in the model. The second team, J.F. Chevalier and P. Gramme (Vadis Consulting) base their approach on a clever methodology to extract features combined with a ridge regression method. C. Kim and K.-B. Hwang from Soongsil University are third ranked by using a naive bayes classifier on a selected set of tags. The selection process is driven by mutual information and a restriction of tags to known tags from the test dataset.

A combination of a language model and kNN is used by Bogers and Bosch, links of a co-occurrence network of tags and resources form the basis in the work of Krestel and Chen, and text clustering is the underlying technique to get an extended feature set combined with a text classification approach in Kyriakopoulou and Kalamboukis. Madkour et. al. investigate the applicability of kNN, Gaussian process, SVM, Neural Networks and NN combined with SVM for the spam prediction task and Neubauer and Obermayer use co-occurrence, network and text features to set up an SVM model. All the approaches demonstrate the applicability of machine learning methods to solve the spam prediction task.

Unfortunately, we received only three submissions for the tag recommender challenge. The winners with an F-Measure of 0.19 are M. Tatu and colleagues from Lymba Corporation. They use a natural language approach to generate tag recommendations. The approach includes an extensive preprocessing to clean the data. The good results mainly stem from an extension of the folksonomy data with conceptual information from Wordnet and from further external resources. M. Lipczak (Dalhousie University) is second. He developed a three step approach which utilizes words from the title expanded by a folksonomy driven lexicon, personalized by the tags of the posting user. Katakis et. al. from the Aristotle University of Thessaloniki come in third by considering the recommender task as a multilabel text classification problem with tags as categories.

As the topic of this year's challenge is related to the topic of the workshop: "Wikis, Blogs, Bookmarking Tools — Mining the Web 2.0 Workshop", we think that both events will benefit from each other. By collocating the challenge workshop and the Mining the Web 2.0 Workshop we combine different contributions of the same community and hope to enable fruitful discussions of results, challenges and ideas in this field.

We thank all participants of the challenge for their contributions and the organizers of the ECML/PKDD 2008 conference for their support. We are looking forward to a very exciting and interesting workshop.

Kassel, August 2008

Andreas Hotho, Beate Krause, Dominik Benz, Robert Jäschke

References

1. T. Fawcett. ROC Graphs: Notes and Practical Considerations for Researchers. Technical report, HP Laboratories, 2004.

Table of Contents

Using Language Models for Spam Detection in Social Bookmarking

Toine Bogers and Antal van den Bosch

ILK, Tilburg University
P.O. Box 90153, 5000 LE
Tilburg, The Netherlands
{A.M.Bogers,Antal.vdnBosch}@uvt.nl

Abstract. This paper describes our approach to the spam detection task of the 2008 ECML/PKDD Discovery Challenge. Our approach focuses on the use of language models and is based on the intuitive notion that similar users and posts tend to use the same language. We compare using language models at two different levels of granularity: at the level of individual posts, and at an aggregated level for each user separately. To detect spam users in the system, we let the users and posts that are most similar to incoming users and their posts determine the spam status of those new users. We first rank all users in the system by KL-divergence of the language models of their posts—separately and combined into user profiles—and the language model of the new post or user. We then look at the spam labels assigned to the most similar users in the system to predict a spam label for the new user. We evaluate on a snapshot of the social bookmarking system BibSonomy made available for the Discovery Challenge. Our approach achieved an AUC score of 0.9784 on an internal validation set and an AUC score of 0.9364 on the official test set of the Discovery Challenge.

Key words: Social bookmarking, language modeling, spam detection, BibSonomy

1 Introduction

A prominent feature of the Web 2.0 paradigm is a shift in information access from local and solitary, to global and collaborative. Instead of storing, managing, and accessing personal information on only one specific computer or browser, personal information management and access has been moving more and more to the Web. Social bookmarking websites are clear cases in point: instead of keeping a local copy of pointers to favorite URLs, users can instead store and access their bookmarks online through a Web interface. The underlying application then makes all stored information sharable among users, allowing for improved searching and generating recommendations between users with similar interests.

Any system that relies on such user-generated content, however, is vulnerable to spam in one form or another. Indeed, many other electronic systems that allow users to store, share, and find online resources have also come under attack from spamming attempts in recent years. Search engines, for instance, suffer increasingly from so-called *spamdexing* attempts with content especially created to trick search engines into giving certain pages a higher ranking for than they deserve [1]. Spam comments are also becoming an increasingly bigger problem for websites that allow users to react to content, like blogs and video and photo sharing websites [2].

Social websites and social bookmarking services have been becoming an increasingly popular part of the Web, but their focus on user-generated content also makes them vulnerable to spam, threatening their openness, interactivity, and usefulness [3]. Motivation for spamming can range from advertising and self-promotion to disruption and disparagement of competitors. Spamming is economically viable because the barrier for entry into the abused systems is generally low and because it requires virtually no operating costs beyond the management of the automatic spamming software. In addition, it is often difficult to hold spammers accountable for their behavior.

Spam for social bookmarking is a growing problem and this has been acknowledged by making it one of the tasks of the 2008 ECML/PKDD Discovery Challenge, along with tag recommendation. In this paper, we focus on the spam detection task alone. Our approach to spam detection is based on the intuitive notion that spam users will use different language than 'legitimate' users when posting resources to a social bookmarking system. We detect new spam users in the system by first ranking all the old users in the system by the KL-divergence of the language models of their posts—separately and combined into user profiles—and the language model of the new user or post. We then look at the spam labels assigned to the most similar users in the system to predict a spam label for the new user.

The paper is structured as follows. We start off by reviewing the related work in the next section, followed by a description of the task and the data set, our preprocessing steps, and our evaluation setup in Section 3. In Section 4, we discuss our approach to the spam detection task. Our results are presented in Section 5. We discuss our findings and conclude in Section 6.

2 Related Work

Spam issues in social bookmarking services have received relatively little attention so far. Heymann et al. (2007) examined the relationship between spam and social bookmarking in detail and classified the anti-spam strategies commonly in practice into three different categories: *prevention*, *detection*, and *demotion* [3]. *Prevention-based* approaches are aimed at making it difficult to contribute spam content to the social bookmarking system by restricting certain types of access through the interface (such as CAPTCHAs) or through usage limits (such as post or tagging quota).

Spam detection methods try to identify likely spam either manually or automatically, and then act upon this identification by either deleting the spam content or visibly marking it as such for the user [3]. To our knowledge, the only published effort of automatic spam detection for social bookmarking comes from Krause et al. (2008) who investigated the usefulness of different machine learning algorithms and features to automatically identify spam [4]. They tested their algorithms on a data dump of the BibSonomy system. This data set was not the same as the one used in the 2008 Discovery Challenge and contained many features not available for the Discovery Challenge task.

Demotion-based strategies, finally, focus on reducing the prominence of content likely to be spam. Rank-based methods, for instance, try to produce orderings of the system's content that are both more accurate and more resistant to spam [3]. A demotion-based strategy for combating spam is described by Heymann et al. (2007) and described in more detail in Koutrika et al. (2007). They constructed a simplified model of tagging behavior in a social bookmarking system and compared different ranking methods for tag-based browsing. They investigated the influence of various factors on these rankings, such as the proportion and behavior of spam users and tagging quota [5], and found that ranking methods that take user similarity into account are more resistant to manipulation.

If we cast our nets a bit wider than just social bookmarking, we can find more anti-spam approaches in related fields, such as blogs. Mishne et al. (2005) were among the first to address the problem of spam comments in blogs and used language model disagreement between the blog post itself, the comments, and any pages linked to from the comments to identify possible spam comments [2]. In 2006, the TREC Blog Track also paid attention the problem of blog spam [6].

Finally, the data set for the 2008 Discovery Challenge is based on the BibSonomy social bookmarking service and, in addition to spam detection and tag recommendation, more research has been done using this system. See [4] for a short overview of the related work.

3 Methodology

3.1 Task description

We include a brief description of the spam detection task and the data in this section to allow this paper to be self-contained. The goal of the spam detection task of the Discovery Challenge was to automatically detect spam users in the provided snapshot of BibSonomy. The goal was to learn a model that can predict whether a user is a spammer or not. An added requirement was that the model should make good predictions for initial posts made by new users, in order to detect spammers as early as possible. This decision to identify spam at the user level—instead of at the post level—means that all of a user's posts are automatically labelled as spam. This decision was justified earlier[1] in Krause et al. (2008) by the observation that users with malicious intent often attempt to hide their motivations with non-spam posts [4]. In addition, Krause et al. also cite workload reduction as a reason for the decision to classify at the user level.

[1] Krause et al. are also the organizers of the 2008 Discovery Challenge, hence the same justification applies. Unfortunately, it is not clear if the results reported in their 2008 paper were achieved on the same data set as the one made available for the Discovery Challenge.

3.2 Data

For the spam detection task a snapshot was made available of the BibSonomy system as a MySQL dump, which consisted of all resources posted to BibSonomy between its inception and March 31, 2008. Two types of resources are present in the data set: bookmarks and BibTeX records. The training data set contained flags that identify users as spammers or non-spammers. The Discovery Challenge organizers were able to collect data of more than 2,400 active users and more than 29,000 spammers by manually labeling users. These labels were included in the data set for training and tuning parameters. Table 1 shows some simple statistics of the data set and illustrates the skewness present in the data set, both in terms of spammers and 'legitimate' users, and looking at the strong preference for bookmarks among spammers and BibTeX among legitimate users.

Table 1. Statistics of the BibSonomy data set.

	count
resources	14,074,956
bookmark, spam	13,257,519
bookmark, clean	596,073
BibTeX, spam	1,240
BibTeX, clean	220,124
users	31,715
spam	29,248
clean	2,467
average posts/user	59.8
spam	55.6
clean	108.9
tags	424,963
spam	69,902
clean	379,888
average tags/post	7.5
spam	8.2
clean	3.0

As mentioned before, two types of resources can be posted: bookmarks and BibTeX records, the latter with a magnitude more metadata available. In our approach we decided to treat BibTeX records and bookmarks the same and thus use the same format to represent them both. We represented all resource metadata in an TREC-style SGML format using 4 fields: <TITLE>, <DESCRIPTION>, <TAGS>, and <URL>. For the bookmarks, the title information was taken from the book_description field, whereas the title field was used for the BibTeX records. The <DESCRIPTION> field was filled with the book_extended field for bookmarks, whereas the following fields were used for the BibTeX records: journal, booktitle, howPublished, publisher, organization, description, annote, author, editor, bibtexAbstract, address, school, series, and institution. For both resource types all tags were added to the <TAGS> field. The URLs extracted from the book_url and url fields were pre-processed before they were used: punctuation was replaced by whitespace and common prefixes and suffixes like www, http://, and .com were removed. Figure 1 shows an example of an instance of our XML representation.

```
<DOC>
  <DOCNO> 694792 </DOCNO>
  <TITLE>
    When Can We Call a System Self-Organizing
  </TITLE>
  <DESCRIPTION>
    ECAL Carlos Gershenson and Francis Heylighen
  </DESCRIPTION>
  <TAGS>
    search agents ir todo
  </TAGS>
  <URL>
    springerlink metapress openurl asp genre article issn 0302 9743
    volume 2801 spage 606
  </URL>
</DOC>
```

Fig. 1. An example of one of the posts (#694792) in our SGML representation.

Other than the data present in the provided data set, we did not use any other, external information, such as, for instance, the PageRank of the bookmarked Web page.

3.3 Evaluation

To evaluate our different approaches and optimized parameters, we divided the data set up into a training set of 80% of the users and a validation set of the remaining 20%. We evaluated our approaches on this validation set using the standard measures of AUC (area under the ROC curve) and F-score, the harmonic mean of precision and recall, with β set to 1. We optimized k using AUC rather than F-score, as AUC is less sensitive to class skew than F-score [7], and the data is rather skewed with 12 spam users for every clean user. For the final predictions on the official test set we used all of the original data as training material.

4 Spam Detection

4.1 Language Models for Spam Detection

Our approach to spam detection is based on the intuitive notion that spam users will use different language than legitimate users when posting resources to a social bookmarking system. By comparing the language models of posts made by spammers and posts made by legitimate users, we can use the divergence between the models as a measure of (dis)similarity. After we have identified the *k* most similar posts or users using language modeling, we classify new users as spam users or genuine users by scoring these new users by how many spam posts and how many clean posts were found to be similar to it.

Language models [8] are a class of stochastic *n*-gram models, generally used to measure a degree of surprise in encountering a certain new span of text, given a training set of text. The core of most language models is a simple *n*-gram word prediction kernel that, based on a context of two or three previous words, generates a probability distribution of the next words to come. Strong agreement between the expected probabilities and actually occurring words (expressed in perplexity scores or divergence metrics) can be taken as indications that the new text comes from the same source as the original training text. Language models are an essential component in speech recognition [9] and statistical machine translation [10], and are also an important model in information retrieval [11]. In the latter context, separate language models are built for each document, and finding related documents to queries is transformed into ranking documents by the likelihood, estimated through their language model, that each of them generated the query.

In generating document language models, there is a range of options on the granularity level of what span of text to consider a document. At the most detailed level, we can construct a language model for each individual post, match these to the incoming posts, and use the known spam status of the best-matching posts already in the system to generate a prediction for the incoming posts or users. We can also take a higher-level perspective and collate all of a user's posts together to form large documents that could be considered 'user profiles', and generate language models of these individual user profiles. Incoming posts or users can then be matched against the language models of spammers and clean users to classify them as being more similar to one or the other category. Figure 2 shows how these two levels of language models relate to one another.

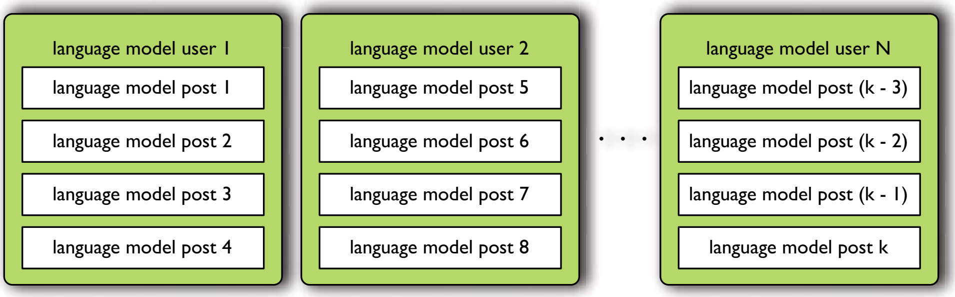


Fig. 2. Two types of language models: the models of the individual posts and the models of the user profiles.

A third option—at an even higher level of granularity—would be to only consider two language models: one of all spam posts and one of all clean posts. However, we believe this to be too coarse-grained for accurate prediction, so we did not pursue this further. Another extension to our approach would have been to use language models for the target Web pages or documents such as proposed by [2]. However, it is far from trivial to obtain the full text of all the source documents linked to by the BibTeX posts. Crawling all the target Web pages of the 2.2 million bookmark posts is impractical as well. Furthermore, we suspect that incorporating language models from all externally linked Web pages and documents would slow down a real-time spam filtering system to an undesirable degree.

We used the Kullback-Leiber divergence metric to measure the similarity between the language models. The KL-divergence measures the difference between two probability distributions Θ_1, Θ_2 is

$$KL(\Theta_1||\Theta_2) = \sum_w p(w|\Theta_1) \log \frac{p(w|\Theta_1)}{p(w|\Theta_2)} \tag{1}$$

where $p(w|\Theta_1)$ is the probability of observing the word w according to the model Θ_1 [2, 8].

The Indri toolkit[2] implements different retrieval methods based on language modeling. We used this toolkit to perform our experiments and construct and compare the language models of the posts and user profiles. The language models we used are maximum likelihood estimates of the unigram occurrence probabilities. We used Jelinek-Mercer smoothing to smooth our language models, which interpolates the language model of a post or user profile with the language model of background corpus, which in our case is the training collection of posts or user profiles. We chose Jelinek-Mercer smoothing because it has been shown to work better for verbose queries than other smoothing methods such as Dirichlet smoothing [12].

We performed two different sets of experiments. First, we compared the language models of the user profiles in our validation set with the language models of the profiles in our training set. For each test user profile we obtained a ranked list of best-matching training users. In addition, we did the same at the post level by comparing the test post language models with the language models of the training posts. Here, ranked lists of best-matching posts were obtained for each test post. These similarity rankings were normalized, and used as input for the spam classification step described in the next subsection.

For both the user and the post level we used all of the available fields—title, description, tags, and tokenized URL—to generate the language models of the posts and user profiles in our training collection. For the new posts and user profiles in our validation and test sets, however, we experimented with selecting only single fields to see what contribution each field could make to the spam detection process. This means that we have four extra sets of representations of the incoming validation and test documents, each with information from only one of the fields, bringing our total of representations for each of the two levels to five.

[2] Available at http://www.lemurproject.org

4.2 Spam Classification

After we calculated the language models for all posts and user profiles, we obtained the normalized ranking of all training documents, relative to each test post or user profile. For each of the best-matching training documents, we used the manually assigned spam labels of 0 or 1 to generate a single spam score for the new user. The simplest method of calculating such a score would be to output the spam label of the top-matching document. A more elegant option would be to take the most common spam label among the top k hits. However, we settled on calculating a weighted average of the similarity scores multiplied by the spam labels, as preliminary experiments showed this to outperform the other options. In the rare case that no matching documents could be retrieved, we resorted to assigning a default label of no spam (0) for these 0.7% of test users, as in our training set 84.2% of these unmatched users were not spammers. For post-level classification, this meant we obtained these weighted average spam scores on a per-incoming-post basis. To arrive at user-level spam scores, we then matched each incoming post to a user and calculate the average per-post score for each user.

One question remains: how many of the top matching results should be used to predict the spam score? In this, our approach is similar to a k-nearest neighbor classifier, where the number of best-matching neighbors k determines the prediction quality. Using too many neighbors might smooth the pool from which to draw the predictions too much in the direction of the majority class, while not considering enough neighbors might result in basing too many decisions on accidental similarities. We optimized the optimal value for k for all of the variants separately on the AUC scores.

5 Results

Table 2 lists the results of our different spam detection approaches. At the user level, the validation set representation where we only used the tags to construct our language models surprisingly outperformed all other approaches and representations, including the one with metadata from all fields. It achieved an AUC score of 0.9784 and the second highest F-score of all user-level representations at 0.9767. Our submission to the Discovery Challenge task was therefore made using this approach. The second best approach compared the language models of user profiles that used all metadata fields and achieved an 0.9688 AUC score. Overall, using the user-level language models outperformed the post-level language models.

Figures 3 and 4 shows the ROC curves for the 10 different combinations of fields and matching level. One surprising difference between the post-level and the user-level experiments is that at the user level the representation with only the tags works best, while it performs worst at the post level. Another interesting difference between post- and user-level experiments is the difference in the optimal number of nearest neighbors k. Matching users appears to require a considerably greater number of neighbors than arriving at a spam classification using only individual posts' language models.

Table 2. Spam detection results of the two approaches on the validation set. The optimal neighborhood sizes *k* were optimized on AUC scores. Best scores for each metric and level are printed in bold.

Level	Fields	Precision	Recall	F-score	AUC	k
user-level	all fields	**0.9659**	**0.9986**	**0.9820**	0.9688	180
	title	0.9534	0.9909	0.9718	0.9308	140
	description	0.9543	0.9976	0.9755	0.9228	95
	tags	0.9580	0.9961	0.9767	**0.9784**	195
	url	0.9502	0.9311	0.9406	0.8478	450
post-level	all fields	0.9735	**0.9950**	**0.9842**	**0.9571**	50
	title	0.9664	0.9815	0.9739	0.9149	50
	description	0.9707	0.9416	0.9559	0.8874	75
	tags	0.9804	0.7448	0.8465	0.7700	10
	url	**0.9773**	0.8940	0.9338	0.8730	15

Our submitted run on the test set provided by the Discovery Challenge used only the tags of each user to compare the language models with *k* set to 195. Classifying the incoming users as spammers or clean users achieved an AUC score of 0.9364. Precision was 0.9846, recall 0.9748, and the F-score 0.9797.

6 Discussion & Conclusions

In this paper we presented our language modeling approach to the spam detection task of the 2008 Discovery Challenge. We start by using language models to identify the best-matching posts or user profiles for incoming users and posts. We then look at the spam status of those best-matching neighbors and use them to guide our spam classification. Our results indicate that our language modeling approach to spam detection in social bookmarking systems shows promising results. This confirms the findings of [2], who applied a similar two-stage process using language modeling to detecting blog spam, albeit on a much smaller scale.

We experimented with matching language models at two different levels of granularity and found that, in general, matching at the user-level gave the best results. This was to be expected as the spam labels for the users in the data set were judged and assigned at the user-level. This means that the misleading, 'genuine' posts of spam users were automatically flagged as spam, thereby introducing more noise for the post-level matching than for the user-level matching.

The best performance at the user level was achieved by matching user-level language models using only the tags of the incoming users' posts. This is in line with the findings of [4], where the features related to the usage and content of tags were also found to be among the most important. Interestingly enough, matching posts only the incoming posts' tags resulted in the worst performance of all post-level runs. We can think of two likely explanations for this. The first is that the post-level approach is more likely to suffer from incoming posts without any assigned

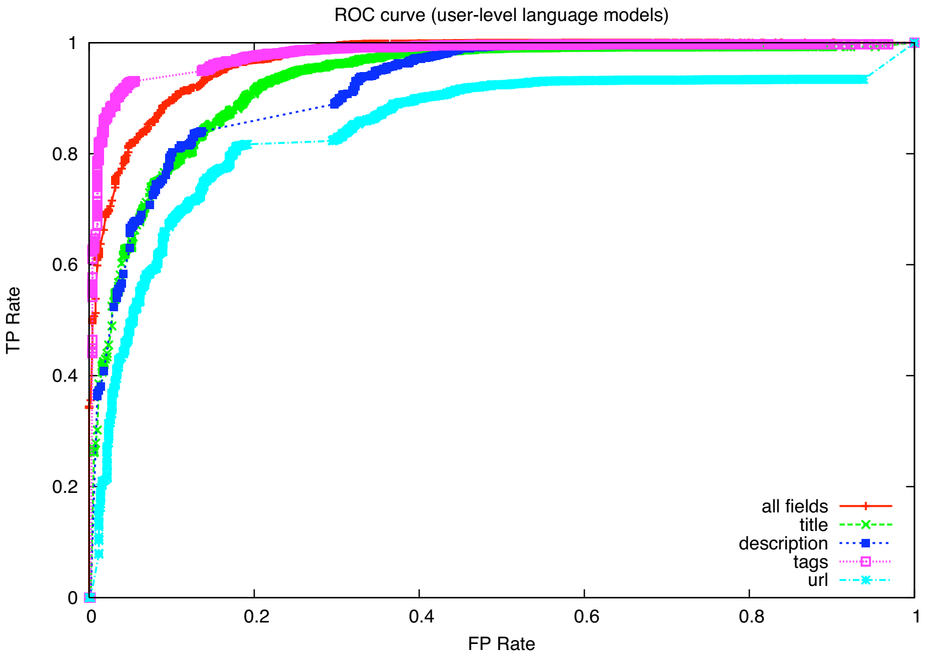


Fig. 3. ROC curve at the user level

tags than the user-level approach is. Although 99.95% of all posts in the data set have valid tags[3], this also means that it is possible for incoming posts to have no tags. Without any tags as metadata, our approach cannot find any matching posts in the system. At the user level, this is even less likely to happen: only 0.009% of all users never assign any tags. However, this might still be a valid reason to use all metadata fields for the user-level approach: with all available metadata we can increase coverage, because empty posts are not allowed by any social bookmarking system.

The second reason illustrates a possible limitation of our approach at the same time: spammers will change their behavior over time and might have done so in the time period the test set originates from. By generating metadata with a similar language model to the clean posts in the system, spammers could make it more complicated for our approach to distinguish between themselves and genuine users. However, this also makes it more difficult for the spammers themselves: it is very hard for a spammer to post resources to a social bookmarking system that will be both similar to existing posts and to the language of the spam entry. In addition, such behavior could easily be countered by extending our method to include the language models of the target resources or by including other features such as the PageRank of bookmarked pages. Extending our approach in such a way is

[3] Valid meaning with a tag other than `system:unfiled`.

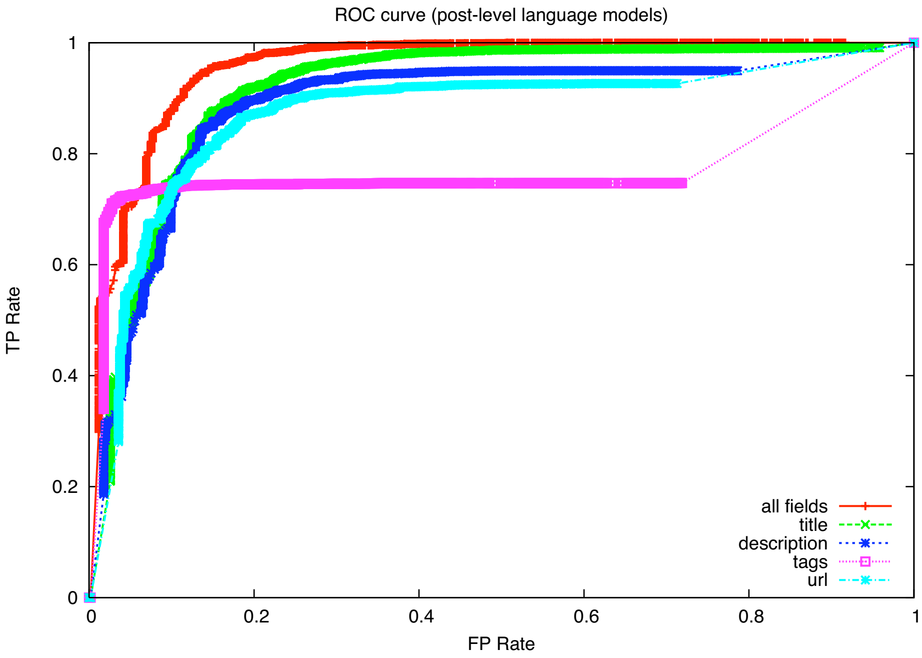


Fig. 4. ROC curve at the post level

one of the possible avenues for future work. Another would be to also restrict the language models of the training set to only certain fields and seeing how this influences performance. Finally, we would also like to test our approach on another social bookmarking system to see how our algorithms carry over to other systems.

One particular advantage of our approach is that it could be implemented with limited effort on top of an existing social bookmarking search engine. After any standard retrieval runs, the top k matching results can then be used to generate the spam classification, only requiring a lookup of predetermined spam labels.

As a final note we wish to briefly describe some of our experiences with applying language models to the other Discovery Challenge task of tag recommendation. By using a similar two-stage approach of first identifying similar posts and then aggregating the most popular tags associated with those best-matching posts, we were only able to achieve a maximum F-score of around 0.10. This clearly illustrates that an approach of finding the system-wide best matching posts for tag recommendation is not a good approach. We believe the reason for this to be that tagging, unlike spamming, is a much more personal activity: the tags another person assigned to the same resource need not necessarily be the tags a new user would apply. For spam detection our method only needs to assign one out of two possible labels to a new user, instead of picking 10 correct tags from a set of hundreds of thousands of possible tags for a new post. We therefore believe our language modeling approach to be better suited to the spam detection than to the tag recommendation.

Acknowledgments

The work described in this paper was funded by SenterNovem / the Dutch Ministry of Economics Affairs as part of the IOP-MMI À Propos project, and by the Netherlands Organization for Scientific Research as part of the NWO Vernieuwingsimpuls program.

Bibliography

[1] Gyöngyi, Z., Garcia-Molina, H.: Web Spam Taxonomy. In: AIRWeb '05: Proceedings of the 1st International Workshop on Adversarial Information Retrieval on the Web, Chiba, Japan (May 2005) 39–47

[2] Mishne, G., Carmel, D., Lempel, R.: Blocking Blog Spam with Language Model Disagreement. In: AIRWeb '05: Proceedings of the 1st International Workshop on Adversarial Information Retrieval on the Web, New York, NY, USA, ACM (2005) 1–6

[3] Heymann, P., Koutrika, G., Garcia-Molina, H.: Fighting Spam on Social Web Sites: A Survey of Approaches and Future Challenges. IEEE Internet Computing **11**(6) (2007) 36–45

[4] Krause, B., Hotho, A., Stumme, G.: The Anti-Social Tagger - Detecting Spam in Social Bookmarking Systems. In: AIRWeb '08: Proceedings of the 4th International Workshop on Adversarial Information Retrieval on the Web. (2008)

[5] Koutrika, G., Effendi, F.A., Gyöngyi, Z., Heymann, P., Garcia-Molina, H.: Combating Spam in Tagging Systems. In: AIRWeb '07: Proceedings of the 3rd International Workshop on Adversarial Information Retrieval on the Web, New York, NY, USA, ACM (2007) 57–64

[6] Ounis, I., de Rijke, M., McDonald, C., Mishne, G., Soboroff, I.: Overview of the TREC 2006 Blog Track. In: TREC 2006 Working Notes. (2006)

[7] Fawcett, T.: ROC Graphs: Notes and Practical Considerations for Researchers. Machine Learning **31** (2004)

[8] Manning, C.D., Schütze, H.: Foundations of Statistical Natural Language Processing. MIT Press, Cambridge, MA (1999)

[9] Jelinek, F.: Self-organized Language Modeling for Speech Recognition. Readings in Speech Recognition (1990) 450–506

[10] Brown, P.F., Cocke, J., Della Pietra, S.A., Della Pietra, V.J., Jelinek, F., Lafferty, J., Mercer, R.L., Roossin, P.S.: A Statistical Approach to Machine Translation. Computational Linguistics **16**(2) (1990) 79–85

[11] Ponte, J.M., Croft, W.B.: A Language Modeling Approach to Information Retrieval. In: SIGIR '98: Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, New York, NY, ACM Press (1998) 275–281

[12] Zhai, C., Lafferty, J.: A Study of Smoothing Methods for Language Models Applied to Information Retrieval. ACM Transactions on Information Systems **22**(2) (2004) 179–214

A novel supervised learning algorithm and its use for Spam Detection in Social Bookmarking Systems

Anestis Gkanogiannis and Theodore Kalamboukis

Department of Informatics
Athens University of Economics and Business, Athens, Greece
utumno@aueb.gr tzk@aueb.gr

Abstract. A novel fast and accurate supervised learning algorithm is proposed as a general text classification algorithm for linearly separated data. The strategy of the algorithm takes advantage of the training errors to successively refine an initial classifier. Experimental evaluation of the proposed algorithm on standard text collections, show that results compared favorably to those from state of the art algorithms such as SVMs. Experiments conducted on the datasets provided in the framework of the ECDL/PKDD 2008 Challenge for Spam Detection in Social Bookmarking Systems, demonstrate the effectiveness of the proposed algorithm.

1 Introduction

Text categorization is the process of making binary decisions about related or non-related documents to a given set of predefined thematic topics or categories. The task is an important component in many information management organizations. In our participation on the ECML/PKDD challenge 2008 we have not deviate from the classical supervised text classification paradigm.

Support vector machines (SVMs) [1] have shown the best performance for text classification tasks. They are accurate, robust, and quick when applied to test instances. Their only drawback is their training complexity and memory requirements. In what follows we present an algorithm, which overcomes both these problems. The strategy of the proposed algorithm takes advantage of the training errors (misclassified examples) to successively refine an initial classifier. This refinement takes the form of an iterative Rocchio-like relevance feedback-learning technique to adjust the centroid vectors of the categories, in order to maximize the performance of the classifier. Our learning algorithm runs on batch mode using all the training data at each iteration step.

Rocchio's relevance feedback technique [2] is a query modification process that has been extensively investigated in the literature and used in information retrieval. Relevance feedback improves the query with terms that are considered relevant to the information neek. This is done iteratively either manually or automatically by selecting a predefined set of the top retrieved documents as relevant (pseudo-relevance feedback). The aim is to find the optimum query, that is a query that maximizes the similarity with relevant documents while minimizing similarity with non-relevant documents. If $R(\bar{R})$ is the set of relevant (non-relevant) documents, then we wish to find, Q_{opt}, such that:

$$Q_{opt} = argmax_q \;\; sim(q, R) - sim(q, \bar{R}) \tag{1}$$

where $sim(q, R)$ denotes the similarity of the query to the set of relevant documents. Using as similarity measure the *cosine* formula, we get that:

$$\vec{Q}_{opt} = \frac{1}{|R|} \underset{d_j \in R}{} \vec{d}_j - \frac{1}{|\bar{R}|} \underset{d_j \in \bar{R}}{} \vec{d}_j = \vec{C}_R - \vec{C}_{\bar{R}} \tag{2}$$

Thus, the optimum query is a vector defined by the difference of the centroids of the relevant and non-relevant documents, as it is shown in Figure 1a. In other words Q_{opt} defines a hyperplane, $h : \vec{Q}_{opt} \cdot \vec{x} - \theta = 0$ that separates the sets, R and $\bar{R}$, for an appropriate value of θ.

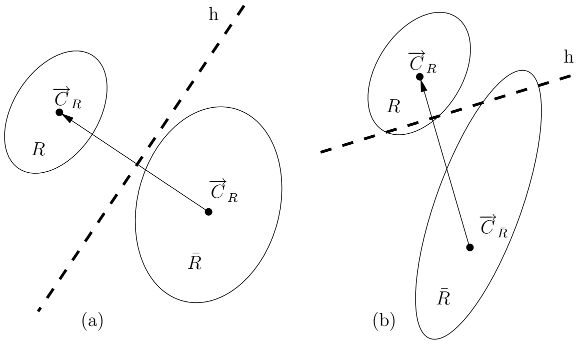


Fig. 1. Vector $C_R - C_{\bar{R}}$ is not always the optimum as it is the case in (1b)

This is not, however, always the case as it is shown in Figure 1b, where the query defined by equation (2) is not optimum although the sets R, $\bar{R}$ are linearly separable and as a consequense there is a hyperplane that separates them. Algorithmically the relevance feedback process has been implemented by updating the query iteratively according to the following equation:

$$\vec{Q}_{i+1} = \kappa \vec{Q}_i + \frac{\lambda}{|R|} \underset{d_j \in R}{} \vec{d}_j - \frac{\mu}{|\bar{R}|} \underset{d_j \in \bar{R}}{} \vec{d}_j \tag{3}$$

for a given initial query vector, $\vec{Q}_0$, where the constants κ, λ, μ are control parameters defined empirically. From (3) follows that the optimal query is, in general, a linear combination of the relevant and non-relevant documents. In the following we shall use the described Rocchio's feedback technique in constructing a classifier for a pair of linearly separable sets. The algorithm starts with an initial classifier, defined in (2), which is improved iteratively applying the relevance feedback technique on the misclassified examples (Figure 1b). In the next paragraph we describe the algorithm in more detail.

The rest of the paper is organized as follows. Section 2 provides a short description of the proposed algorithm. Section 3 present briefly the task of spam detection in social bookmarking systems and the preprocessing of the data. Section 4 presents the experimental results and finally in section 5 we conclude on the results and the advantages of the proposed algorithm.

2 The Learning Algorithm

We briefly describe here a modification of an algorithm [3], which has been tested, on several standard text collections describing a consistent behavior over all these collections with a performance comparable to SVMs, a state of the art classification algorithm. The algorithm constructs a common tangent hyperplane for the sets R,$\bar{R}$ such that these sets lie on opposite sides of the hyperplane.

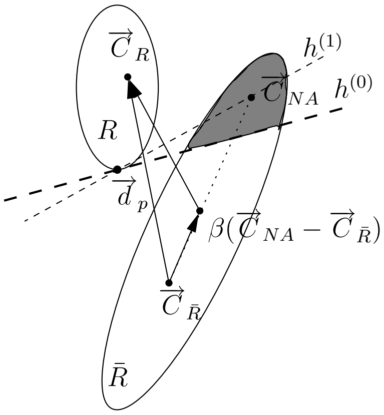

$\overrightarrow{C}_R$
R
$h^{(1)}$
$\overrightarrow{C}_{NA}$
$h^{(0)}$
$\overrightarrow{d}_p$
$\beta(\overrightarrow{C}_{NA} - \overrightarrow{C}_{\bar{R}})$
$\overrightarrow{C}_{\bar{R}}$
$\bar{R}$

Fig. 2. Rotation of hyperplane $h^{(0)}$ towards the misclassified examples

Initialization of the algorithm:

- Select initial vector $\overrightarrow{W}^{(0)} = \overrightarrow{C}_R^{(0)} - \overrightarrow{C}_{\bar{R}}^{(0)}$
- Calculate $s_j = \overrightarrow{W}^{(0)} \cdot \overrightarrow{d}_j$, $\forall\ d_j \in R \cup \bar{R}$
- Find s_p such that $s_p = min(\overrightarrow{W}^{(0)} \cdot \overrightarrow{d}_j)$, $\forall d_j \in R$

The hyperplane defined by $h^{(0)} : \overrightarrow{W}^{(0)} \cdot \overrightarrow{x} - \theta = 0$, with $\theta = s_p = \overrightarrow{W}^{(0)} \cdot \overrightarrow{d}_p$ by construction is vertical to the vector $\overrightarrow{W}^{(0)}$ and $\overrightarrow{d}_p$ lies on it (θ was defined at the value of $recall = 1$).

By construction all the relevant documents lie on the same side of $h^{(0)}$, the one pointed by its normal vector $\overrightarrow{W}^{(0)}$. If it happens all the non-relevant examples to lie on the other side of $h^{(0)}$, then $h^{(0)}$ is a separating hyperplane and the algorithm stops. However this is not generally the case, as it is shown in Figure 2, where the hyperplane $h^{(0)}$ defined by the above process, intersects the set of non-relevant documents and the examples in the gray area are misclassified. In this case we rotate the hyperplane $h^{(0)}$ towards the misclassified examples (Negative Accepted (NA) examples) until all the negative examples lie on the same side of the hyperplane.

Rotation of $h^{(0)}$ towards the misclassified examples: The rotation of the hyperplane is performed stepwise. At each step we determine the misclassified, NA, training examples by the current classifier, construct their centroid vector, $\overrightarrow{C}_{NA}$, and then rotate the hyperplane forcing it to pass through the points $\overrightarrow{d}_p$ and $\overrightarrow{C}_{NA}$. This is equivalent to the process of moving $\overrightarrow{C}_{\bar{R}}$ towards the misclassified examples $\overrightarrow{C}_{NA}$ by adding the vector $\beta(\overrightarrow{C}_{NA} - \overrightarrow{C}_{\bar{R}})$, i.e. $\overrightarrow{C}_{\bar{R}} \leftarrow \overrightarrow{C}_{\bar{R}} + \beta(\overrightarrow{C}_{NA} - \overrightarrow{C}_{\bar{R}})$, such that the plane defined by $\overrightarrow{W}^{(1)} = \overrightarrow{C}_R - (\overrightarrow{C}_{\bar{R}} + \beta(\overrightarrow{C}_{NA} - \overrightarrow{C}_{\bar{R}}))$ passes through the points $\overrightarrow{d}_p$ and $\overrightarrow{C}_{NA}$. With little algebra we estimate the value of the parameter β by:

$$\beta = \frac{\overrightarrow{W}^{(0)} \cdot (\overrightarrow{C}_{NA} - \overrightarrow{d}_p)}{(\overrightarrow{C}_{NA} - \overrightarrow{C}_{\bar{R}}) \cdot (\overrightarrow{C}_{NA} - \overrightarrow{d}_p)} \tag{4}$$

This rotation however may cause examples in the set R to be misclassified (PR, the set of Positive Rejected examples). Thus the process is repeated now on the set R by moving $\overrightarrow{C}_R$ towards the centroid of the misclassified examples $\overrightarrow{C}_{PR}$, i.e. $\overrightarrow{C}_R \leftarrow \overrightarrow{C}_R + \alpha(\overrightarrow{C}_{PR} - \overrightarrow{C}_R)$, such that the plane defined by $\overrightarrow{W}^{(2)} = (\overrightarrow{C}_R + \alpha(\overrightarrow{C}_{PR} - \overrightarrow{C}_R)) - \overrightarrow{C}_{\bar{R}}$, passes through the points $\overrightarrow{C}_{PR}$ and $\overrightarrow{C}_{NA}$. Similarly we find that the parameter α is determined by:

$$\alpha = -\frac{\overrightarrow{W}^{(1)} \cdot (\overrightarrow{C}_{PR} - \overrightarrow{C}_{NA})}{(\overrightarrow{C}_{PR} - \overrightarrow{C}_R) \cdot (\overrightarrow{C}_{PR} - \overrightarrow{C}_{NA})} \tag{5}$$

This alternation of the rotation towards either the NA or the PR continues until $|NA| = |PR| = 0$ or the number of iteration exceeds a predetermined value. This process converges to a common tangent hyperplane that leaves the sets R and $\bar{R}$ on opposite sides of the hyperplane.

3 Task Description

This year's challenge deals with two tasks about a new area called social bookmarking. Internet sites of this type offer to users the ability to share with each other material such as links to web pages, scientific publications and etc. The first task of this year's challenge deals with spam detection in such systems, whereas the second one deals with tag recommendations.

Spammers have already discovered that social bookmarking sites provide them with a large and continuously growing pool of potential customers. In fact it is much more attractive a spam post in a social bookmarking site than an annoying e-mail in someone's e-mail box. This task tries to identify such spam posts, using previously manually assigned as spam or not data from a well known social bookmarking site. The second task is about assisting the users when posting a new post, suggesting them with the appropriate tags that should accompany their post. The same non spam data used for the first task is used for training models for the second task. We have only submitted a solution for the first task, so in the rest of this paper we will refer to that task of the challenge.

The evaluation of the submissions was performed using the correct user labels as spammers or not and the participants ranked lists, with the Area Under the ROC (Receiver Operating Characteristics) Curve as an evaluation measure.

3.1 Data Description

The data used to train the model for the spam task, was taken by a well known social bookmarking site. Users of this site post their favorite links or publications, along with tags they assign to them. A snapshot was taken and all these posts where manually assigned as spam or not spam. More precisely each user of the site was labeled as spammer or not based on his posts. The final raw data where organized in 7 text files. 3 of them (tas, bookmark, bibtex) contain the data of spammers, 3 (tas_spam, bookmark_spam, bibtex_spam) of them contain the data of not spammers and the final file (user_spam) contains the true labels of the users as spammers or not.

Each post of a user may be a link to a web page (bookmark) or a link to a publication (bibtex). Files bookmark_spam and bookmark contain the bookmark posts of spammer and not spammer users, files bibtex_spam and bibtex contain the publication posts respectively and finally the files tas_spam and tas contain the posts of the users along with the assigned tags and references to the appropriate bookmark or bibtex entry.

In the first step of the preprocessing stage of the data we had to organize the raw text in a user basis. This means that for each user, spammer or not, we unified all of his posts, bibtex or bookmark. Each user was identified by a unique id and after this step for each user id we had a text fragment representing all of his posts, which means a text containing all the tags, all the bookmark posts and all the bibtex posts.

The second step of the preprocessing stage removes any unnecessary text, like common or stop words, performs stemming, using Porter's stemmer, and finally extracts the features of the dataset and produces the vectors of the users.

The final train dataset contains 31,715 user vectors (29,248 for spammers and 2,467 for not spammers). We found 244 spam and 84,298 not spam unique bibtex entries, 1,626,560 spam and 176,147 not spam unique bookmark entries.

The dimensionality of the vector space was 480,062, which means that the final datasets contains 480,062 unique features or words. Finally each vector was normalized to unity length.

4 Experimental Results

Using the user vectors defined from the training dataset, a model of our algorithm was trained. In order to evaluate the performance and due to the luck of any test dataset, the initial train set was randomly divided into two equal subsets. The first was used for training the model and the second for testing.

This training dataset contains 14,624 spam user vectors and 1,234 non spam user vectors. The evaluation dataset contains 14,624 spam user vectors and 1,233 non spam user vectors.

After training the model and applying it to the evaluation data, it correctly classified as spammers 14,403 users (TP, True Positives), correctly classified as not spammers 884 users (TN, True Negatives), incorrectly classified as spammers 349 users (FP, False Positives) and incorrectly classified as not spammers 221 users (FN, False Negatives). This means a F1 measure of 98.06%. Using the provided by the organizers of the challenge script for calculating the AUC value, a 96.20% AUC value was estimated. Figure 3 shows the ROC curve generated using the script we mentioned.

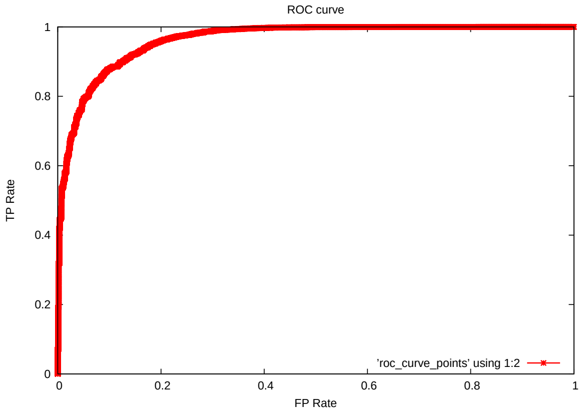


Fig. 3. ROC curve of the training set of the challenge

4.1 Test Dataset

The format of the test dataset was the same as the train dataset. After applying the same processing as in the train dataset case, the test dataset consisted of 7,205 user vectors.

Training the model with the full train dataset (31,715 user vectors), and applying it to the unlabelled test dataset, a list of the 7,205 users along with the determined by the model confidence values was submitted to the challenge.

By the end of the challenge, the true user labels where known and we where able to determine an AUC value of 97.96% and produced the ROC curve for the test dataset shown in the figure 4 below.

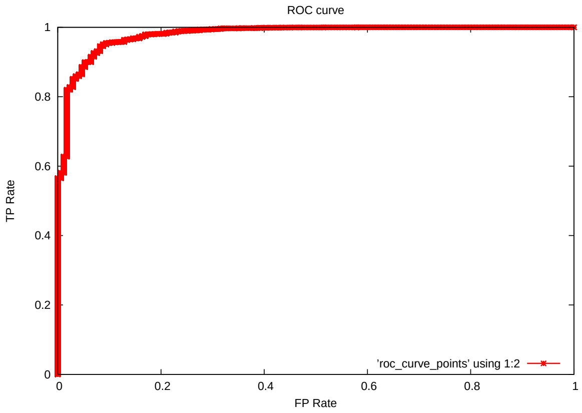


Fig. 4. The ROC curve of the testing set of the challenge

For a comparison of our method, we ran the challenge task with SVMs [4]. As it is reported in [5] the use of linear SVMs is both accurate and fast (to train and to use), so SVM's results were obtained using the SVMLight software package [6], with the default parameter values and a default linear kernel. Using the above mentioned script the AUC value was determined at 97.55%. We observe that our method outperforms SVMs both in accuracy and efficiency. Although the difference in performance is not significant, the training time of our algorithm took half the time of that of SVMs (both methods, where executed on the same machine). However the programming code of our algorithm in its current version is not optimized, and a new version is under development which will dramatically reduce the training time.

5 Concluding Remarks

In concluding, we have briefly described a novel algorithm for text categorization that is accurate and fast and we have demonstrated its performance in the ECML challenge of this year. The algorithm has been also tested on several standard text collections and showed a robust and comparable or even better performance compared with SVMs. As we have already mentioned the proposed algorithm overcomes both drawbacks of SVMs both in the training complexity and memory requirements. Definitely there are many details and extensions to the algorithm which are currently under investigation and will be published in due course.

References

1. Joachims, T.: Text categorization with support vector machines: learning with many relevant features (1998)
2. Rocchio, J.J.: Relevance feedback in information retrieval. In Salton, G., ed.: The SMART retrieval system: experiments in automatic document processing. Prentice-Hall, Englewood Cliffs, USA (1971) 313–323
3. Gkanogiannis, A., Kalampoukis, T.: An algorithm for text categorization. In: 31st ACM International Conference on Research and Development in Information Retrieval SIGIR-2008. (2008) 869–870
4. Cristianini, N., Shawe-Taylor, J.: An Introduction To Support Vector Machines (and other kernel-based learning methods). Cambridge University Press (2000)
5. Dumais, S., Platt, J., Heckerman, D., Sahami, M.: Inductive learning algorithms and representations for text categorization (1998)
6. Joachims, T.: Making large-scale support vector machine learning practical. In: Advances in Kernel Methods: Support Vector Machines. (1998)

RANK for spam detection
ECML - Discovery Challenge

Vadis Consulting – J.-F. Chevalier & P. Gramme

Vadis Consulting SA/NV., Allée de la recherche/ Researchdreeft 65
1070 Brussels (Anderlecht), Belgium
www.vadis.com
bjfc@vadis.com, bpgr@vadis.com

Abstract. This submission is aimed to benchmark of Vadis methodology in the context of spam detection. The work that has been done to provide these results can be separated in two different tasks: data preparation and modelisation .

Keywords: Very large scale problem, LARS, Variable recoding, Ridge regression, Features selection.

1 Introduction

Our approach can be summarized in four steps:

- Produce variables into a "single view", containing one record per user ;
- Split the users into two segments, according to the type of content posted ;
- Bin variables and recode them according to the percentage of targets in each bin ;
- Apply LARS algorithm & backward cross-validation to perform linear regression on these recoded variables.

The first and the most time consuming task was to derive variables from the initial files. The un-homogeneity of the data led us to split it into two segments. Once we achieved this goal, we used our generic tool to build models. After analysis and fine tuning, we ended up with a model ready to be applied on the test set.

2 Data Preparation

The goal of data preparation is to build a number of variables describing each user. There is practically no limitation in the number of variables created since our modelisation tool RANK can cope with very large data set, with a very high number of columns.

Cleaning of text fields

The provided data included a number of fields consisting of text entered by the users: tags, description of web pages or articles, etc. Before computing information, we performed some cleaning of these fields:

- Put to lower case
- Remove special characters and count them

- Cut tags into words (using spaces, hyphens and punctuation as separators)
- Try to correct typos: each tag is compared to the 1000 most popular tags. If it “close” to any popular tag, it is replaced by that tag. Otherwise, it is left as is.
- Replace tag starting with a number by either “replace_of_year" or "replace_of_number”.

Measuring the information of text fields

The goal of this measure is to estimate the rarity of a document within a corpus. In this case, the document is the value of some text variable for some user or resource, and the corpus consists of all the values taken by this text variable for all users or resources.

The information of a text field is defined as the sum of the information of all its words. The information of a single word is the inverse of its log-frequency – i.e. divide the total number of words in the considered text field (across all users) by the total number of occurrences of the word, and take the logarithm of the quotient.

Variables describing tags

Several variables were produced describing the tags posted by a user. These variables concern the tag itself, not the resource pointed by the tag. They include among others

- The number of tags of a user which contain a given special character
- The total, average, minimum and maximum number of tags that the user posted per resource
- The total, average, minimum and maximum length of the tags posted by a user
- The total, average, minimum and maximum information (see above) of the tags posted by a user.

Manual aggregation of top words

The 1000 most frequent tags (after cleaning) were manually grouped into 10 categories. These categories were afterwards used for computing some variables:

- The main category used by the user
- The total number of categories used
- The number and proportion of the user’s tags in each category
- The set of all categories used at least once by the user (appended in order to form a string variable)

Variables describing resources

The resources (both URLs and BibTEX entries) pointed out by a user were described using the following variables:

- The number of resources bookmarked by the user
- The information of different fields describing the resource (url, url_hash, description and extended_description for bookmarks, and description for BibTEX entries). The per-user sum, average, minimum and maximum information is then computed for every of those fields.

3 Modelisation

Segmentation

Users can be divided into 3 segments: users with no BibTEX entry, users with no bookmark entry, and users with both BibTEX and bookmarks. The following table shows the proportion of spammers in each segment.

Has bookmark	Has BibTEX	Nb of users	% spammers
1	0	30386	95.9 %
0	1	682	3.8 %
1	1	647	14.2 %

The strong differences in the proportion of targets shown in this table suggests to separate users having BibTEX (and possibly bookmarks too) from users having no BibTEX. We thus performed two different models.

Since our modelisation tool, RANK, expect to predict the modality which is less represented, the prediction tasks were set up so as, for the BibTEX, predicting the spammers, and for the non-BibTEX, prediction of the non-spammers.

RANK

RANK is a predictive modeling tool designed by analysts for the analyst. As a result, it combines powerful techniques and modeling experience.

It is the first tool that automates many steps of the CRISP DM methodology (http://www.crisp-dm.org/) for building models.

RANK is built to allow an analyst to quickly build models on huge data sets, and have all elements to control the model choices and its quality, in order to focus his attention on the most important part of the modeling process: data quality, overfitting, stability and robustness. Using RANK, the analyst will get support for many modeling phases: audit, variable recoding, variable selection, robustness improvement, result analysis and industrialization.

Using ridge regression [2] on a linearized space, RANK combines the robustness of the linear models and the performance of a tidily controlled non-linear approach.

Variable recoding

Non linear Recoding

The recoding of variables is an extremely important and time-consuming step in the modeling process. Analysts know that the quality of a model can be heavily influenced by this phase. This is why RANK has been extensively developed on this step to ensure best model performance. RANK allows the user to specify which type of recoding he/she wants to test, and RANK will just do it, and select the best recoding scheme for each variable.

The types of recoding are the following:

- Nominal variables – Modalities will be converted to a numeric value that is related to its relation with the target density. This recoding is known under the name "weight of evidence recoding" [3]. Modalities can also be recoded using dummy variables.
- Numerical variable – There are two possibilities: simple normalization of the variables or binning of the variable using a proprietary algorithm ('intelligent quantiles') and then treated as nominal variables with order. The intelligent quantiles analyses the distribution of a variable in order to identify most relevant quantiles, identifying 'plateau' and jumps in the distribution. In this mode, jumps also produce dummy variables.

The recoding performed by RANK has two major effects:

- The first one is to get rid of the problem of non-normal distributions that should be a basic assumption when using regression models. The recoding will remove the dissymmetry and make the data more suitable for regression models.
- The second effect is that the recoding allows RANK to spot non-linear relationships of a variable with the target, thus improving the expression power of the model.

Modality grouping

RANK automatically analyzes all variables along their cardinality. If a nominal variable has many modalities, RANK will group them in a way that each grouped modality becomes significant. For example, if Zip code with 30.000 modalities is used, only the modalities that are significant will be left as they are. The others will be grouped in a default modality. The grouping will preserve the order relationship in a variable if any. For example, for an ordinal variable like 'number of sms sent', RANK will group only modalities that are adjacent, and will possibly create many grouped modalities

Missing Values

RANK treats missing values in a very careful way. Depending on the type of variable, RANK will recode missing values in a way such that its effect on the computed score is null. This ensures that the model focuses only on relevant information for the prediction.

Variable selection

LARS Forward

When the number of variables is high (> 500), a first variable selection made using the Least Angle Regression (LARS) [1]. LARS is an embedded technique which simultaneously estimates the parameters of a linear regression and selects the most relevant variables. It is only used here for variable selection as the regression coefficients will be re-estimated later on using a ridge regression. In RANK, a variant of LARS called, *LARS with Lasso modification* [1], is implemented. Interestingly, this method computes the parameters of the Lasso regression, *i.e.* a linear regression with an upper bound on the L1 norm of the vector of coefficients. Using the L1 norm enforces the sparseness of coefficients leading to effective variable shrinkage. Importantly, the LARS procedure returns *all* the Lasso solutions in a single run, *i.e.* the coefficients for any (positive) value of the upper bound. To do so, LARS operates iteratively. At each iteration, a new variable is selected and a step is taken in the direction equi-angular to the columns of the data matrix corresponding to the currently selected variables. Doing so allows one to progressively minimize the residual error of the model while spreading uniformly its variance over all the selected variables. The algorithm is iterated until the relative residual error of consecutive iterations falls below a user-defined threshold. Note that, even if LARS works in a forward fashion, it has the ability to take backward steps by removing variables becoming useless at some stage. The algorithm results in variables pre-selection that will, afterwards, be validated by the backward pruning.

This mode can be applied on data sets involving more than 200.000 variables.

Lift optimized Backward
The backward pruning in RANK can either start with all the variables or with the pre-selection returned by the LARS. In both cases, it iteratively eliminates variables when their removal does not influence the quality of the prediction more than a prescribed threshold.

Using cross-validation, it will end with a variables selection that maximizes the area under the lift curve.

Robust Regression

Ridge regression
The regression engine of RANK uses the so-called 'Ridge' regression [2]. This technology allows improving the robustness of the models as well as improving the usage of nominal variables with a lot of modalities, like zip codes.

Cross-Validation & Bootstrap
RANK extensively uses cross-validation technique when building a model: to assert which recoding is best, to select best variables, and to evaluate the ridge regression constant. This is extremely useful when the target density is very low, which is 90% the case in real life projects like churn prediction (0.7 % per month), cross- and up-selling (0.3% of our clients possess this product) or fraud detection (0.02% of all cases). Cross-validation and bootstrap is not only relevant for building a robust model, it is also important for the analyst to observe the volatility of the model quality.

Probability Estimation
The output of RANK is not just a score. It also gives for each record the best estimation of the response probability, based on the model score function and the *a priori* probability of the target in the data file.

4 Results

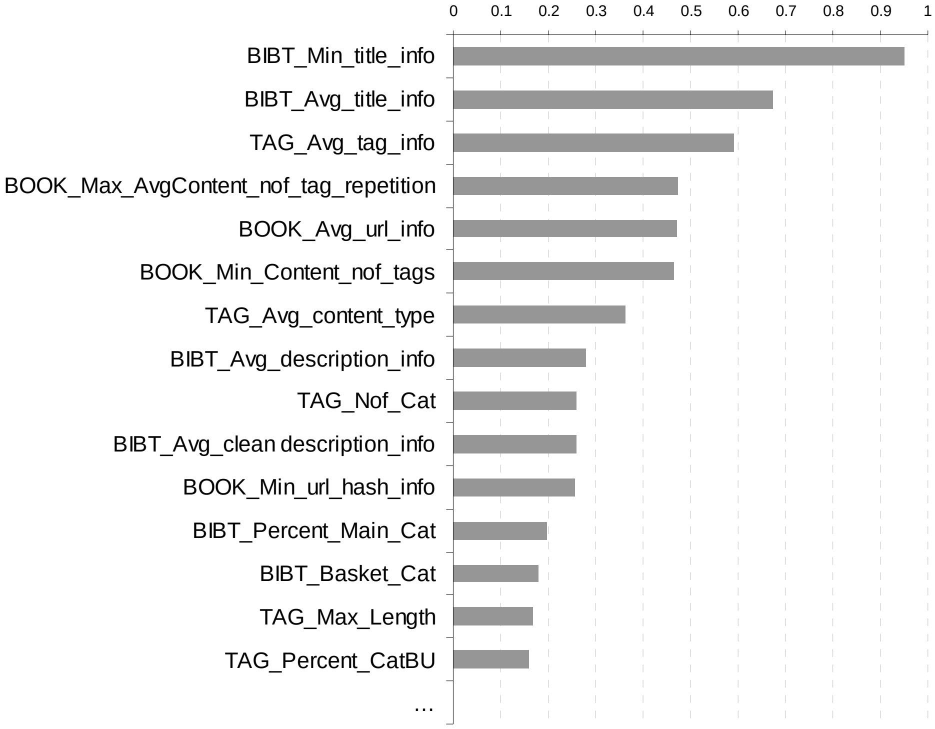
Selected variables

This section lists the top variables for each model. For some variables, it also contains graphics showing the relation between the variable distribution and the target.

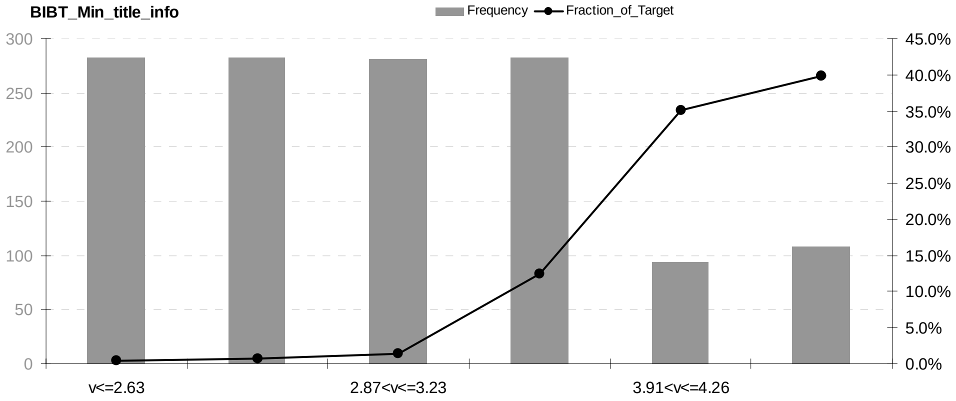
We usually rank the variables according to their importance. The importance is the loss (in percentage) of lift quality that we observe if we remove the variable.

Model for BibTEX users
In this model, the target is spam user. We have 1,329 users and 118 target (8.88%). Our model is composed of 53 variables; most of them are measuring information of a text field.

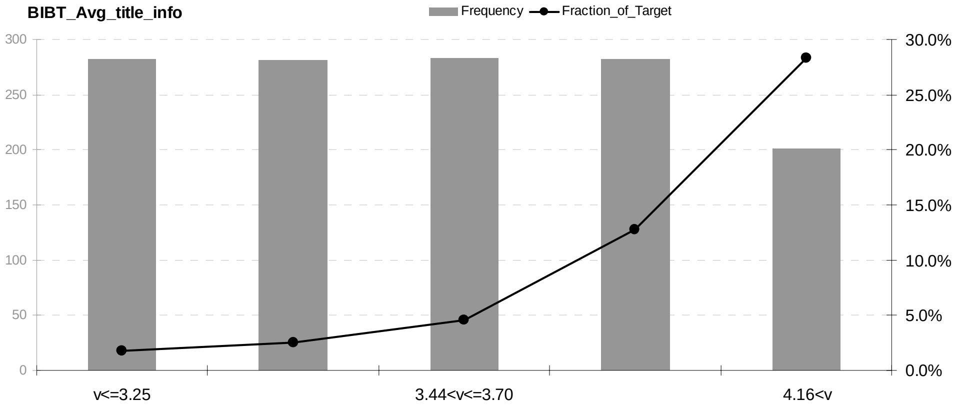
Fig. 1. Most important variables: Top 15 for BibTEXusers.



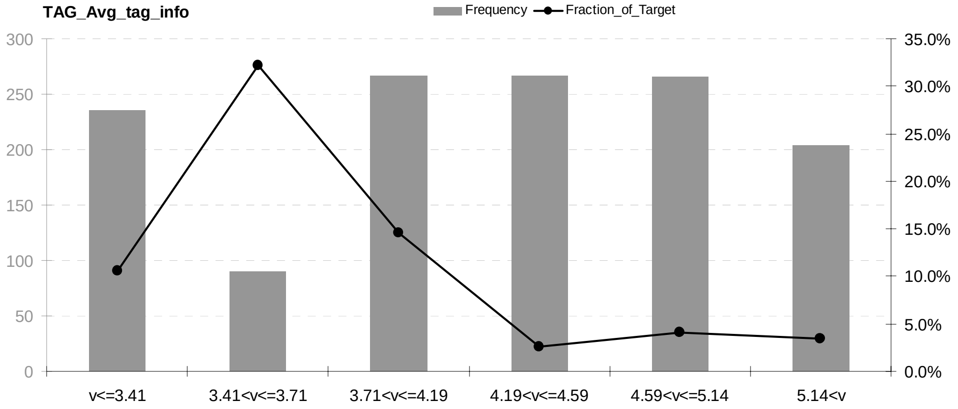
- **BIBT_Min_title_info**: minimum information contained among the titles of the BibTEX posted by the user.



- **BIBT_Avg_title_info**: average information contained among the titles of the BibTEX posted by the user.



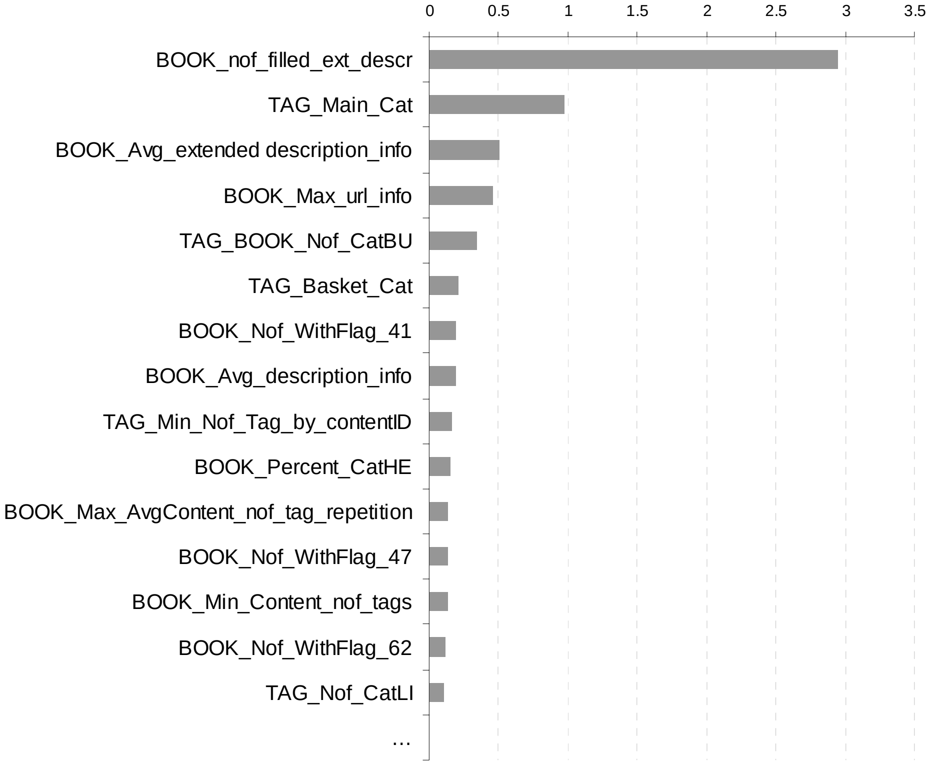
- **TAG_Avg_tag_info:** average information contained among all the tags posted by the user.



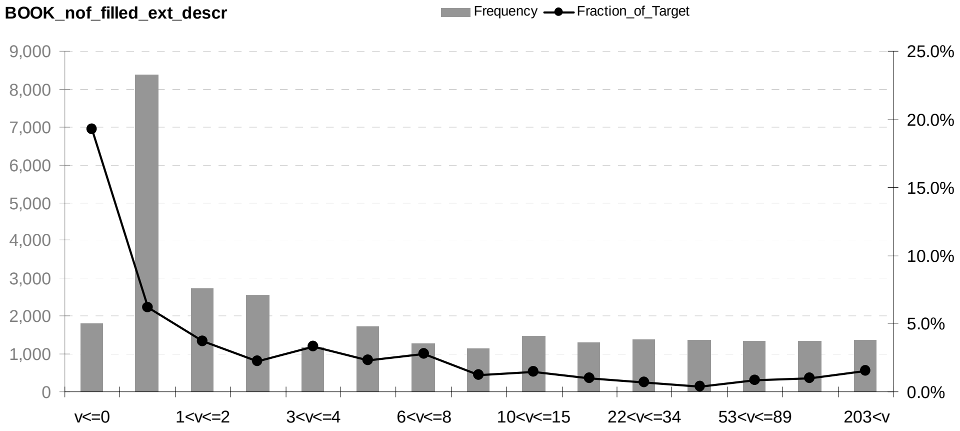
Model for non-BibTEX users

For this model, a target is a non spam user. We have 30,386 users with no BibTEX, 1,256 of them are not spammer (4.13%). In our model, we end up with 70 variables. The figure below shows the 15 most important variables.

Fig. 2. Most important variables: Top 15 for non-BibTEXusers.

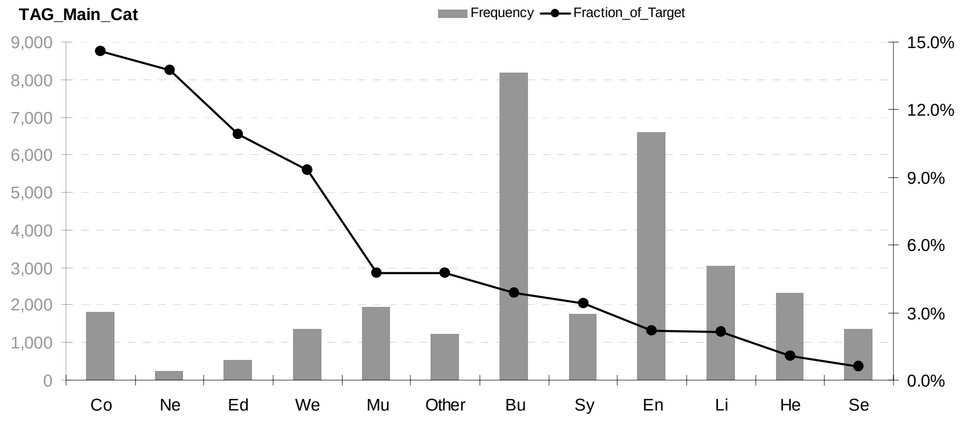


- **BOOK_Nof_Filled_ext_desc**. This first variable counts the number of filled extended_description.

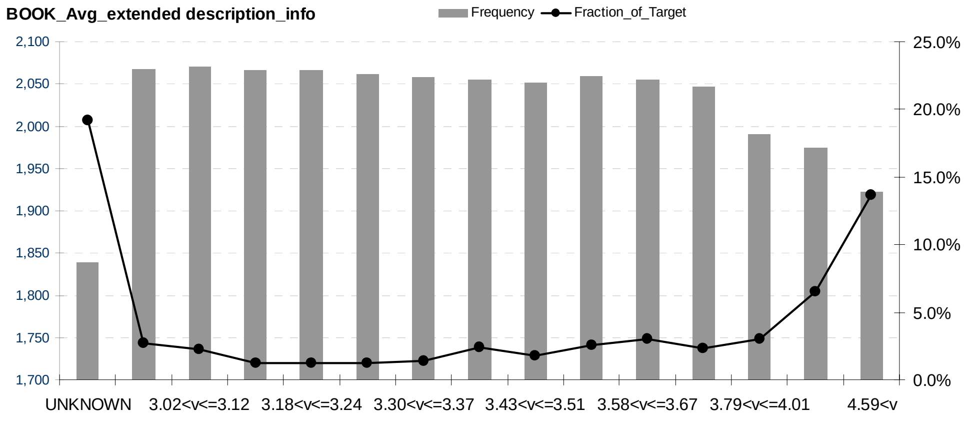


- **TAG_Main_Cat**. This variable shows the main category of the user tags. The most interesting categories are:

- Co = Computer (e.g.: software, program,...)
- Ne = News (e. g.: information, news,...)
- Ed = Education (e.g.: exercise, student,....)
- Li = Link word (e.g.: you, from,...)
- He = Health (e. g.: acne, treatment,...)
- Se = Sex (e. g.: lesbians, xxx,...)



- **BOOK_Avg_Extended_description_info**: This computes the average information in extended_description among all content ids of the user.

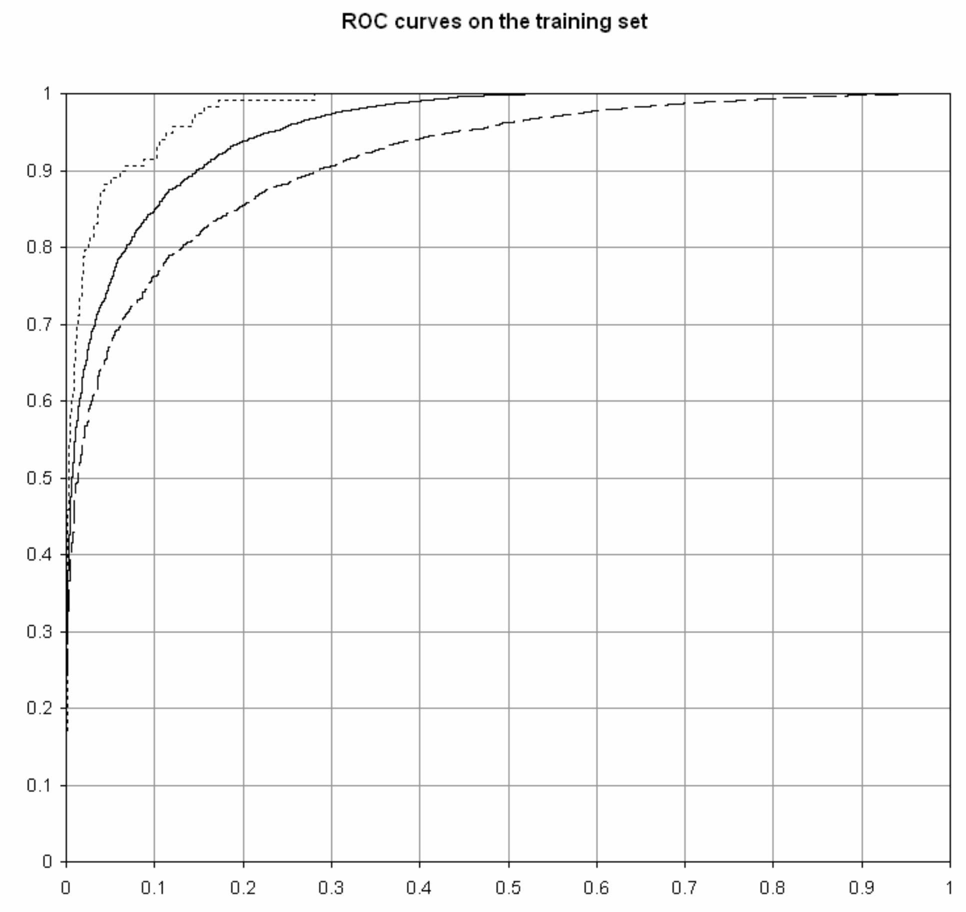


ROC curve

Training set

The following figure shows the ROC curves achieved on the training set for the two models (users with or without BibTEX), and for their combination. All these curves represent the performance for spam users prediction, hence the probability of the non-BibTEX model has been reverted. The area under the curve is 0.9792 for the BibTEX model, 0.9151 for the NoBibTex model, and 0.9556 for the global model.

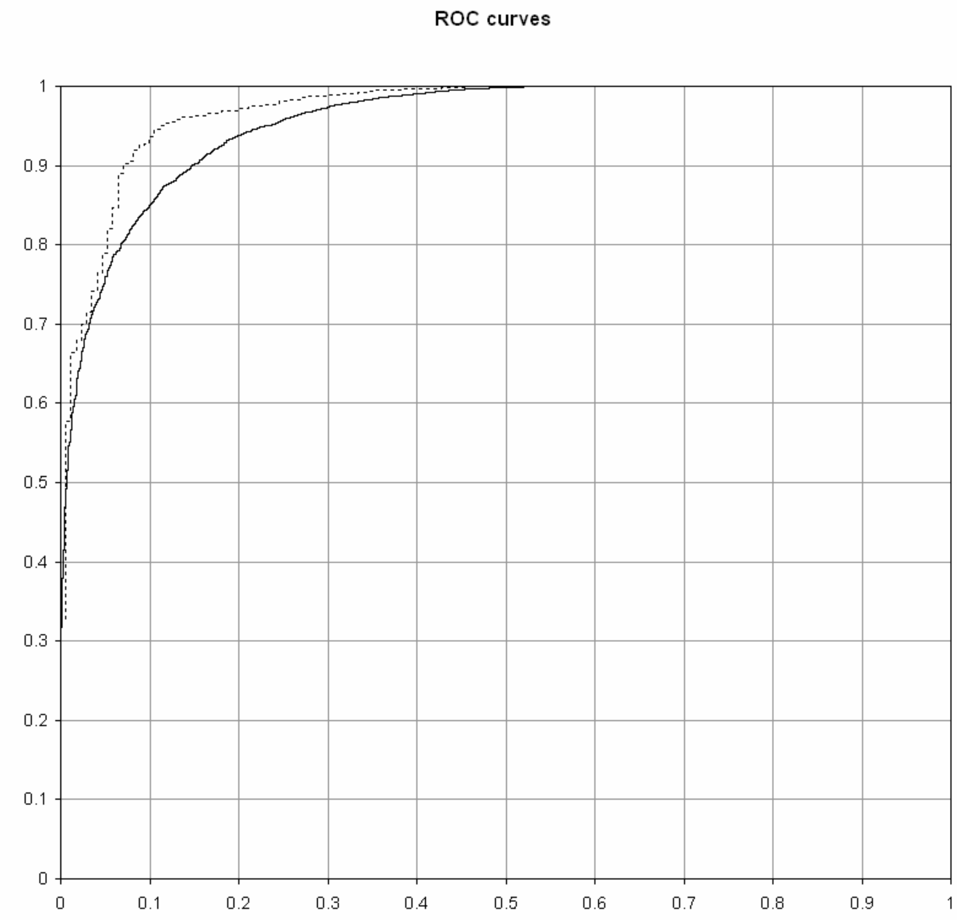
Fig. 3. ROC curve of the models on the build set. The dotted line represents the ROC curve of the BibTEX users model, the dashed line represents the ROC for the non-BibTEX users model, and the plain line shows the ROC we obtain when we combine the two models (global model).



Test set

Finally, the following pictures compares the performance of the model on the training and test sets. For the test set , the area achieved under the ROC is 0.9703. This is higher than the build! This is probably due to the fact that we have more BibTEX users in the test set.

Fig. 4. ROC curve of the global model. The plain line represents the ROC curve on the build set whereas the dotted line represents the ROC on the evaluation test set.



5 References

1. B. Efron, T. Hastie, I Johnstone and R. Tibshirani. Least Angle Regression, *The Annals of statistics* 2004, Vol 32, No 2, 407-499.
2. Hoerl, A. E. and Kennard, R. (1970). Ridge regression: biased estimation for nonorthogonal problems, *Technometrics* 12: 55-67.
3. Smith EP, Lipkovich I, Ye K. Weight of Evidence (WOE): Quantitative estimation of probability of impact. Blacksburg, VA: Virginia Tech, Department of Statistics; 2002.

Naive Bayes Classifier Learning with Feature Selection for Spam Detection in Social Bookmarking

Chanju Kim and Kyu-Baek Hwang

School of Computing, Soongsil University, Seoul 156-743, Korea
cjkim@ml.ssu.ac.kr, kbhwang@ssu.ac.kr

Abstract. Social bookmarking systems such as BibSonomy and del.icio.us have become increasingly popular with the prevalent use of internet. These systems provide powerful infrastructure solutions for semantic annotation and information sharing, promoting diverse kinds of internet-based activities, e.g., web exploration, creating and joining web-based communities, and buying recently published volumes. This usefulness is also gaining the attention of malicious users a.k.a. spammers. For instance, these spammers abuse social bookmarking systems for biasing web search results and advertising improperly. Manual spam detection is not scalable due to the vast amount of related information. In this paper, we propose a machine learning-based approach to automatic spam detection. In specific, a set of relevant features, i.e., the number of posts and posted tags for each user are extracted from training data. The extracted tags are sorted by mutual information. Then, the tags, having high mutual information value and used in test data, are chosen for the classification task. In our experiments, naive Bayes classifiers with varying numbers of selected features were learned from a subset of the given training dataset and evaluated on a separate validation set for finding the optimal parameter setting. Finally, the learned results from the entire training dataset with the best setting was applied to the real test dataset for the challenge.

1 Introduction

The performance of social bookmarking systems [4] can be severely degraded by malicious users, i.e., spammers. The spammers post irrelevant and misleading information for their private benefit. The irrelevant and misleading information in public repositories not only makes the repositories untrustful but also wasting their resources in processing the vast amount of unnecessary information. Thus, it is crucial to discriminate spamming from proper posting for the success of social bookmarking systems.

Several challenges exist in this spam filtering problem. One is the enormous amount of data. The number of users of a social bookmarking system usually amounts to several tens of thousands. Also, the number of posts could amount to several millions. To make matters worse, the given data for spam filtering could be highly skewed. For instance, the ratio between spammers and active users is about one to twelve in the given training dataset for the first task of ECML PKDD Discovery Challenge 2008 (`http://www.kde.cs.uni-kassel.de/ws/rsdc08/`). Another difficulty arises from the fact that the characteristics of the spam filtering data are gradually changing as time goes by. In other words, the distribution of a feature variable might be largely different according to the time period over which it is estimated.

We addressed the above problems using naive Bayes classifiers learning with an enhanced feature selection method. More specifically, tags for spam detection are chosen based on their mutual information value as well as their usage in test period. In our experiments, the suggested feature selection method was shown to generally outperform the conventional feature selection method solely based on mutual information.

The paper is organized as follows. In Section 2, we describe the given task and explain the proposed method for tackling the problem. The experimental results for parameter setting and the performance of the proposed approach on the test dataset is given in Section 3. Finally, conclusions are drawn in Section 4.

2 The Method

The given task is to discriminate spammers from active-users (non-spammers) based on their posting information such as the tags, the dates, the urls, the descriptions, and the related information to the published volumes. The training dataset consists of 31,715 users (including both spammers and active users) and was gathered during the period from January 1989 through March 2008. Among the 31,715 users, 2,467 are active users and the others are spammers. The dataset is comprised of seven tables, i.e., tas, tas_spam, bookmark, bookmark_spam, bibtex, bibtex_spam, and user.

We formulated the given task as a supervised learning problem in which each user corresponds to a data example and the target variable denotes whether the user is spammer or not. As feature variables, the number of bookmark postings, the number of bibtex postings, and the tags were deployed. Here, the tag variable denotes whether a user have ever posted a specific tag or not.

Because there exist a tremendous amount of posted tags (more than 425,000), an appropriate number of tags should be selected for avoiding the overfitting problem and reducing the computational cost. We harnessed the mutual information [1] for tag selection. The mutual information between a tag and the target variable is calculated as follows.

$$I(Tag_i; Target) = \mathop{}_{tag_i,target} \hat{P}(Tag_i, Target) \log \frac{\hat{P}(Tag_i, Target)}{\hat{P}(Tag_i) \cdot \hat{P}(Target)}, \quad (1)$$

where Tag_i is a binary variable denoting whether the ith tag is used or not, $Target$ corresponds to the target variable, and $\hat{P}(\cdot)$ denotes the probability value estimated from a given dataset. Here, the summation is taken over all possible configurations of the two variables, Tag_i and $Target$.

As a classifer, the naive Bayes classifier [5] was adopted because of its computational efficiency as well as its optimality for classification tasks even when the conditional independence assumption is invalid [2]. Actually, we have also tried other famous classification methods including artificial neural networks, support vector machines, tree-augmented naive Bayes classifiers, and decision trees. Their performance in our problem setting was much worse than that of the naive Bayes classifier although the experimental results are not shown here. The naive Bayes classifier in our problem setting is simply formulated by the following equation.

$$\begin{aligned} P(Target|F_1, F_2, ..., F_n) &= \frac{P(Target) \cdot P(F_1, F_2, ..., F_n|Target)}{P(F_1, F_2, ..., F_n)} \\ &= \frac{P(Target) \cdot P(F_1|Target) \cdot P(F_2|Target) \cdot ... \cdot P(F_n|Target)}{P(F_1, F_2, ..., F_n)}, \end{aligned} \quad (2)$$

where F_i corresponds to the ith feature variable. The feature variables include Tag_i defined as in Equation (1), the number of bookmark postings, and the number of bibtex postings. Tag_i's are binary. Other two feature variables were discretized by the supervised discretization method of Weka [6]. The method is based on the approach proposed by [3].

One of the challenges in spam detection lies in the fact that the tag usage pattern is continuously changing. For example, some tags chosen from a training dataset by mutual information might not exist in a separate test dataset. In this case, such tags cannot tell a test example is spammer or not because they have never been used in the test dataset. To mitigate this problem, we propose an enhanced feature selection method, considering both the mutual information and whether the tag is used in the test period as follows.

1. Remove the tags which do not exist in the test dataset.

2. Select a pre-specified number of tags from the remaining tags according to their mutual information values.

3 Experimental Evaluation

To evaluate the proposed approach and find the optimal parameter value[1], we reserved the data examples from the latest two months from the given training dataset. Hence, the training period is from January 1989 to January 2008 and the validation period is from February to March of the same year. The numbers of postings during these periods are shown in Table 1. The numbers of spammers and active users[2] during the same periods are shown in Table 2.

In order to empirically find the optimal number of tags for classification, we experimented with varying numbers of selected tags from 100 to 3,000. We also

[1] Here, the parameter value denotes the number of tags.

[2] It should be noted here that some users exist in both the training dataset and the validation dataset.

Table 1. The number of postings in the given training dataset.

	Non-spam	Spam	Total
Training period	260,271	1,264,539	1,524,820
Validation period	8,421	362,266	370,687
Total	268,692	1,626,805	1,895,497

Table 2. The number of users in the given training dataset.

	Active users	Spammers	Total
Training period	2,466	29,248	31,714
Validation period	656	10,610	11,266

compared the conventional mutual information-based feature selection method with the proposed one. The experimental results are shown in Fig. 1.

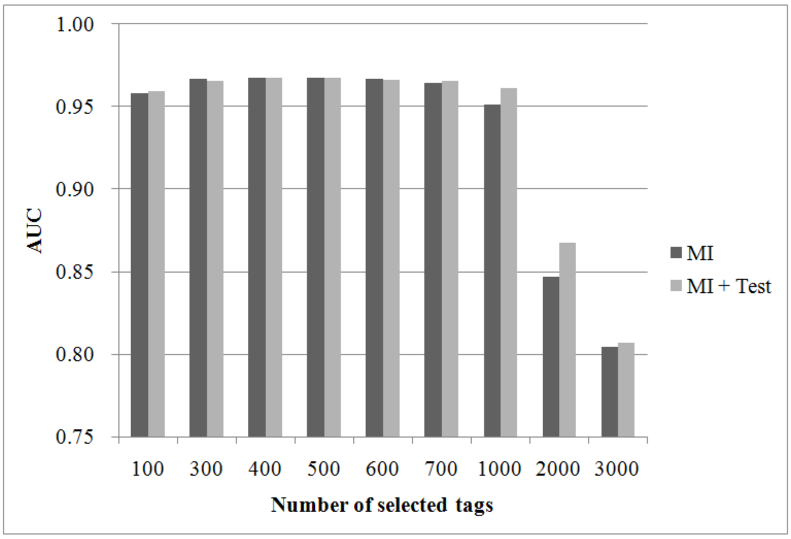


Fig. 1. Comparison of the feature selection methods with varying numbers of selected tags. MI: the conventional method. MI + Test: the proposed method.

From the results, we can observe that the proposed method improves the performance of naive Bayes classifiers in general. Also, the classification per-

Table 3. Comparison of the feature selection methods when the number of selected tags is less than 1,000. MI: the conventional method. MI + Test: the proposed method. Here, the performance is measured by the AUC.

# of Tags	100	300	400	500	600	700
MI	0.95793	**0.96709**	**0.96771**	0.96769	**0.96689**	0.96445
MI + Test	**0.95911**	0.96580	0.96739	**0.96775**	0.96598	**0.96590**

formance decreases as the number of selected tags exceeds 1,000. One possible explanation for this phenomenon is that the large number of selected tags causes the overfitting problem. In fact, the number of extreme prediction values, i.e., zero and one, increases as the number of tags grows.

The detailed comparison of the feature selection methods when the number of selected tags is less than 1,000 is given in Table 3. In this case, the classification performance obtained by the proposed feature selection method is similar to that by the conventional one. We conjecture that this is because the tags with very high mutual information are not so much different in our training and validation datasets.

We applied our spam detection method to predicting spammers in the final test dataset. The final results are shown in Fig. 2.

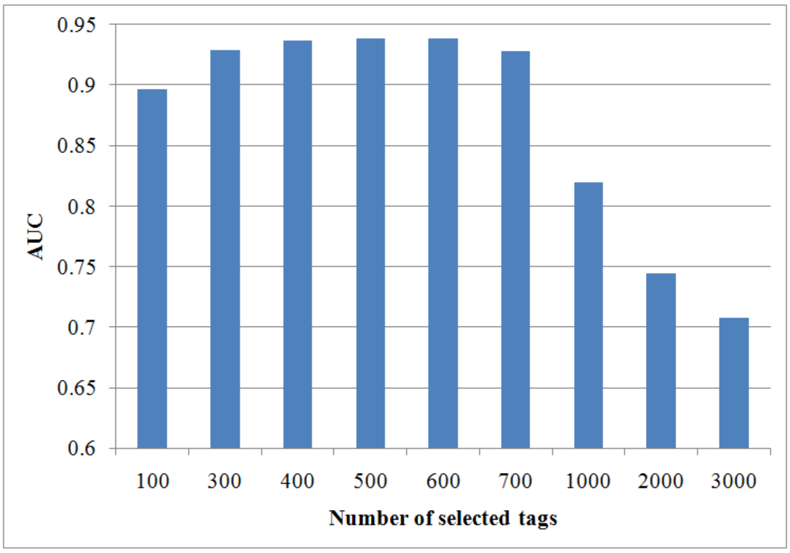


Fig. 2. Classification performance of the proposed method on the real test dataset with varying numbers of selected tags.

Interestingly, the optimal number of selected tags from Table 3 and Fig. 2 is the same. In both cases, the best classification performance was obtained by considering 500 tags. The best result in Fig. 2 is 0.93899.

4 Conclusions

We described a machine learning-based approach for spam detection. As a classifier, the naive Bayes classifier was employed because of its simplicity and efficiency. The number of bookmark postings, the number of bibtex postings, and the tags were considered as feature variables for the classification. For the tag selection, mutual information as well as the term's usage in test period were taken into account. The proposed feature selection method was shown to outperform the conventional mutual information-based approach in general. The number of selected tags was empirically optimized through the validation experiments. Through the experiments on the final test dataset, we have shown that our empirical choice of the optimal number of selected tags from the training dataset was meaningful. One of the directions for future work would be to combine the interrelationship between tags into our approach for more enhanced classification performance.

Acknowledgements

This work was supported in part by the Seoul Development Institute through Seoul R&BD Program (GS070167C093111) and in part by the Ministry of Knowledge Economy of Korea through Ubiquitous Computing and Network (UCN) Project (21st Century Frontier R&D Program).

References

1. Cover, T.M., Thomas, J.A.: Elements of Information Theory. Wiley-Interscience (1991)
2. Domingos, P., Pazzani, M.: On the optimality of the simple Bayesian classifier under zero-one loss. Machine Learning 29(2/3), 103–130 (1997)
3. Fayyad, U.M., Irani, K.B.: Multi-interval discretization of continuous-valued attributes for classification learning. In: 13th International Joint Conference on Artificial Intelligence, pp. 1022–1027. (1993)
4. Heymann, P., Koutrika, G., Garcia-Molina, H.: Can social bookmarking improve web search? In: First ACM International Conference on Web Search and Data Mining. (2008)
5. Langley, P., Iba, W., Thompson, K.: An analysis of Bayesian classifiers. In: Tenth National Conference on Artificial Intelligence, pp. 223–228. (1992)
6. Witten, I.H., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques, 2nd Edition. Morgan Kaufmann, San Francisco (2005)

Using Co-occurence of Tags and Resources to Identify Spammers

Ralf Krestel and Ling Chen

L3S Research Center
Universität Hannover, Germany
krestel|lchen@L3S.de

Abstract. Today, more and more social networking websites support *collaborative tagging*, which allows users to annotate resources (e.g., video clips, blog posts, and bookmarks) on the web. Due to its increasing popularity, however, spammers started to target this new type of service and generate misleading tags either to increase the visibility of some resources or simply to confuse users. Consequently, the performance of applications built upon tag data, such as recovery and discovery of web resources, can be limited. In this paper, we propose an algorithm to identify spammers from the collaborating systems by employing a *spam score propagating* technique. The three dimensional relationship among users, tags and web resources is firstly represented by a graph structure. A set of seed nodes, where each node represents a user, are then selected and assigned values to indicate whether the corresponding users are spammers or not. The initial values are propagated through the graph to infer the status of the remaining users. Our experimental results demonstrate the effectiveness of this approach in identify tag spammers.

1 Introduction

With the recent rise of Web 2.0 technologies, many social media applications like *Flickr*, *Del.ici.ous*, and *Last.fm* provide features which allow users to assign tags [1] to a piece of information such as a picture, blog entry, video clip etc. Web users from different backgrounds tag (annotate) resources on the Web at an incredible speed, which results in large volume of tag data obtainable from the Web today. The hidden value of tag data has been explored in many applications. For example, Tso-Sutter et al [2] incorporated tags into collaborative filtering algorithms to enhance recommendation accuracy. In [3], the authors discussed using tags to lighten the limitation of the amount and quality of anchor text to improve enterprise search. The usage of tags in Web search has also been investigated in Bao et al [4].

One notable reason which supports the increasing popularity of collaborative tagging is that users are permitted to enter tags without any constraints. Consequently, spammers can easily take advantage of this new service to generate misleading tags to increase the visibility of some resources or simply to confuse users. *Therefore, identifying spammers from collaborating systems is an important problem so that top-quality tag data can be generated by removing those supplied by spammers.* Some research effort has been exerted to target this problem. For example, Koutrika et al [5] proposed to combat tag spam by ranking the results returned from a query tag, based on the co-occurrence frequency between the tag and each resource. Their approach is specially designed for tag based search, while our research objective is more general so that the results can be used in not only tag based search but also other applications of collaborative systems.

In our approach, we firstly construct a graph which models users as nodes and three types of relationship between users as edges. Particularly, we consider the following types of relationship between users: common tags supplied by users, common resources annotated by users and common tag-resource pairs used by users. We then select a set of seed nodes whose corresponding users are manually assessed as spammers or not. The identity of the remaining nodes/users are computed by propagating the status of seed nodes through the graph. The effectiveness of our approach is demonstrated on the bibsonomy data set[1].

The rest of this paper is organized as follows. We discuss the background knowledge by reviewing related work in Section 2. In Section 3, we describe the approach which propagates the identity of seed users through the graph. The evaluation results conducted on the bibsonomy data set are presented and analyzed in Section 4. Finally, Section 5 concludes this paper with some summary remarks and future work discussions.

2 Related Work

In this section, we review related work in two areas, collaborative tagging systems and spam detection.

A collaborative tagging system allows users of a web site to freely attach to a particular resource arbitrary tags which, in the opinion of the user, are somehow associated with the resource in question. The commonly noted structure of collaborative filtering systems is a tripartite model consisting of users, tags and resources. This model is developed as a theoretical extension of the bipartite structure of ontologies with an added "social dimension" in [6]. The dynamics of collaborative systems are examined in [7] using the tag data at the bookmarking site Del.ici.ous. According to this work, tag distributions tend to stabilize over time. Halpin et al. confirm these results in [8] and show additionally that tags follow a power law distribution. Considering the structure and stable dynamics of collaborative tagging systems, it seems likely that tag data would be a reliable source of semantic information reflecting the cultural consensus of a particular system's users. As a result, various applications of tag data have been researched. Mika [6] investigates the automatic extraction of ontological relationships from

[1] http://www.kde.cs.uni-kassel.de/ws/rsdc08/dataset.html

tag data and proposes the use of such emergent ontologies to improve currently existing ontologies which are less capable of responding to ontological evolution. Dmitriev et al. [3] explore the use of "annotations" for enterprise search to compensate for the lack of sufficient anchor text in intranet environments. In [4], tag data is exploited for the purpose of web search through the use of two tag based algorithms: one exploiting similarity between tag data and search queries, and the other utilizes tagging frequencies to determine the quality of web pages. Tso et al [2] incorporate the tag data into the collaborative filtering systems. Berendt and Hanser [9] demonstrate the benefits of using tag data for weblog classification by treating it as content instead of meta data. For searching and ranking within tagging systems, A. Hotho et al [10] propose the *FolkRank* algorithm which extends the seminal PageRank approach. In particular, they model the structure of the folksonomy as a graph, where nodes represent users, tags and resources, and edges represent the assignment relationship between users and tags, users and resources, tags and resources.

Everywhere in the internet where information is exchanged, malicious individuals try to take advantage of the information exchange structure and use it for their own benefit. The largest amount of spam and historically the first field where spam was generated is the electronic communication system (e-mail). Afterwards, various internet applications were attacked by spammers such as search engine spam, blog spam, wiki spam etc, which triggered numerous research efforts in spam combating. For example, TrustRank [11] separates spam pages from non-spam pages based on the intuition that trustworthy pages usually link to also trustworthy pages and so on. They select a seed set of highly trusted pages first and then propagate the trust score of seed pages by following the links from these pages through the Web. A survey of approaches fighting spam on social web sites can be found in [12]. Comparing to spam detection from other web applications, studies on detecting spam from collaborative tagging systems are very limited. Koutrika et al [5] propose to combat spam in the particular situation when users query for resources annotated with certain tags. Their method ranks a resource higher if more users annotated it with the queried tags, based on the assumption that tag spam may not be used by the majority. As mentioned before, our work is different in the way that our approach is not designed for a particular application. Consequently, the output of our algorithm — a set of identified tag spammers — can be used by any application based on tags. Xu et al [13] assign authority scores to users, and measure the goodness of each tag with respect to a resource by the sum of the authority scores of all users who have tagged the resource with the tag. Then, the authority scores of users are computed via an iterative algorithm similar to HITs [14]. Contrasting to their approach which iteratively computes authority scores for users and tag-resource pairs, we iteratively update scores for users only. Moreover, our approach is more flexible in the way that multiple relationship, such as co-tag, co-resource and co-tag-resource, can be taken into account, rather than considering only the tag-resource pairs shared by users.

3 Finding Malicious Users

Identifying malicious users (spammers) in a tagging environment with thousands of participants and millions of tag assignments can be done by exploiting the *wisdom of the crowds* [15]. If many known spammers use a certain tag for a certain resource, it might indicate that other users having the same tag assignment are also spammers. In our approach we use an algorithm similar to TrustRank [11] to propagate a spammer score through a graph with each node representing a user. As in TrustRank, we need a set of seed nodes which were manually assessed. For the competition, the training data was used as the seed set.

3.1 Problem Specification

Let $\mathcal{U}$ be a set of users of a collaborating system, $\mathcal{T}$ be a set of tags, and $\mathcal{R}$ be a set of resources. We define the functions $getT(u)$ and $getR(u)$ to retrieve the set of tags and resources assigned by user u respectively. In addition, we define the function $getTR(u)$ to return the set of tag-resource pairs used by user u. For example, $getTR(u) = \{t_m r_n\}$ indicates that the user u assigned the tag t_m to the resource r_n.

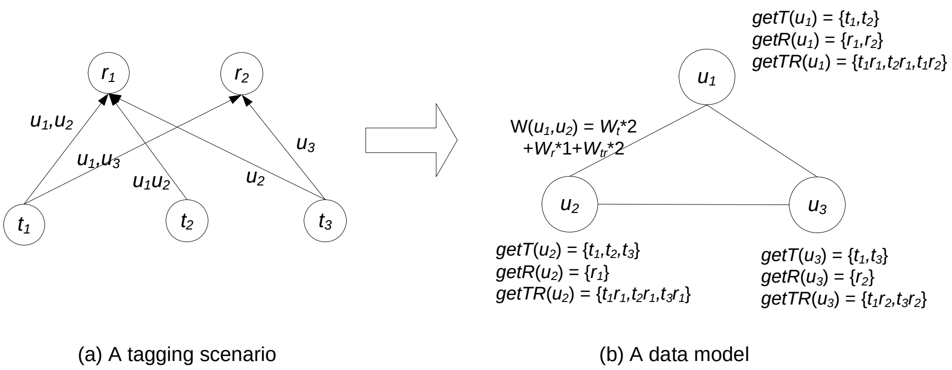
Our goal is to find a function $S(u_i), u_i \in \mathcal{U}$, which assigns a score to each user u_i such that the higher the value of $S(u_i)$, the higher the probability that u_i is a spammer. The value of $S(u_i)$ ranges in $[-1, 1]$ (the reason why negative values are involved will be explained later in Section **??**).

3.2 Tagging System Model

Given a set of data including users $\mathcal{U}$, tags $\mathcal{T}$ and resources $\mathcal{R}$, we model the data as a bidirected weighted graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, where $\mathcal{V}$ is a set of vertices with each $v \in \mathcal{V}$ represents a $u \in \mathcal{U}$. $\mathcal{E}$ is a set of edges such that each edge (v_i, v_j) indicates that the two corresponding users u_i and u_j used at least one common tag or resource. That is, $|getR(u_i) \cap getR(u_j) \cup getT(u_i) \cap getT(u_j)| \geq 1$.

Additionally, we associate a weight to each edge so that the weight of an edge depends on the number of shared tags and resources of the end nodes of the edge: $W(v_i, v_j) = W(u_i, u_j) = (|getT(u_i) \cap getT(u_j)| \times W_t) + (|getR(u_i) \cap getR(u_j)| \times W_r) + (|getTR(u_i) \cap getTR(u_j)| \times W_{tr})$. $W_i, i \in \{r, t, tr\}$ represent static weighting factors to pay tribute to the different degrees of proximity depending on whether they are sharing the same tags t, resources r or even the same tag assignments tr.

In Figure 1 (a), we present a very simple tagging scenario: Suppose we have three users $\mathcal{U} = \{u_1, u_2, u_3\}$, three different tags $\mathcal{T} = \{t_1, t_2, t_3\}$ and two resources $\mathcal{R} = \{r_1, r_2\}$. Each user has annotated the resources with certain tags. For example, the leftmost link in Figure 1 (a) indicates that both users u_1 and u_2 have supplied the tag t_1 with the resource r_1. Based on the tag assignments in this figure, the corresponding data model can be created as Figure 1 (b). Three nodes, representing the three users, are connected with each other according


r_1
r_2
u_1,u_2
u_1,u_3
u_1u_2
u_3
u_2
t_1
t_2
t_3
$getT(u_1) = \{t_1,t_2\}$
$getR(u_1) = \{r_1,r_2\}$
$getTR(u_1) = \{t_1r_1,t_2r_1,t_1r_2\}$
u_1
$W(u_1,u_2) = W_t*2$
$+W_r*1+W_{tr}*2$
u_2
u_3
$getT(u_2) = \{t_1,t_2,t_3\}$
$getR(u_2) = \{r_1\}$
$getTR(u_2) = \{t_1r_1,t_2r_1,t_3r_1\}$
$getT(u_3) = \{t_1,t_3\}$
$getR(u_3) = \{r_2\}$
$getTR(u_3) = \{t_1r_2,t_3r_2\}$

(a) A tagging scenario

(b) A data model

Fig. 1. A tagging scenario and its data model

to common tags/resources/tag-resources pairs. The results of the three functions related to a user, $getT(u_i), getR(u_i), getTR(u_i)$, are shown in the figure as well. Then, based on the tags, resources, and tag-resources used by a user, the weight of an edge connecting two users can be computed. For example, as shown in the figure, the weight of the edge between u_1 and u_3 is calculated as $W(u_1, u_2) = W_t * 2 + W_r * 1 + W_{tr} * 2$, since the two users shared two tags, one resource and two tag-resource pairs.

Based on this graph model, we introduce a right stochastic transition matrix T, which is defined as:

$$T(i,j) = \begin{matrix} 0 & \text{if } (v_i, v_j) \notin \mathcal{E} \\ \frac{W(v_i,v_j)}{\sum_{v_k \in \mathcal{V}} W(v_i,v_k)} & \text{if } (v_i, v_j) \in \mathcal{E} \end{matrix}$$

Suppose W_t, W_r and W_{tr} are set as 1. Figure 2 shows the adjacency matrix and the transition matrix for the example in Figure 1. Note that, the adjacency matrix is symmetric since the graph model is bidirected, while the transition matrix is asymmetric.

	v_1	v_2	v_3
v_1		5	3
v_2	5		2
v_3	3	2	

$$T = \begin{pmatrix} 0 & \frac{5}{8} & \frac{3}{8} \\ \frac{5}{7} & 0 & \frac{2}{7} \\ \frac{3}{5} & \frac{2}{5} & 0 \end{pmatrix}$$

Fig. 2. Adjacency (left) and transition (right) matrixes of the example in Figure 1.

3.3 Spammer Score Propagation

In our approach, the spammer score for each user, $S(u)$, is computed similarly to TrustRank [11], which itself is based on PageRank [16]. The TrustRank employs the formula as follows:

$$\text{t-rank}_{i+1} = \alpha \cdot T \cdot \text{t-rank}_i + (1 - \alpha) \cdot \mathsf{d}, \tag{1}$$

with transition matrix T, a weighting factor α and the manually assessed seed vector d. We use this formula to propagate initial spammer scores of seed users through the graph. In addition to TrustRank which propagates only trust information, we adopt the distrust propagation idea described in [17] to allow the propagation of scores for not only good users but also explicitly bad users (spammers). Consequently, we extend the manual seed set assessment to include both good users and spammers. We populate the initial vector d with:

$$\mathsf{d}(u_i) = \begin{array}{ll} \mathrm{O}(u_i) & \text{if } u_i \in SEED \\ 0 & \text{if } u_i \notin SEED \end{array} \tag{2}$$

where $\mathrm{O}(u_i) \in \{-1, 0, 1\}$ is the oracle function which assigns initial score 1 to non-spammers, -1 to spammers and 0 to the rest. $SEED \subseteq \mathcal{U}$ is a set of seed nodes, which for the competition was the provided set users in the training data.

Consider the running example shown in Figures 1 and 2, the results of our approach (i.e. spammer score for each user) after 10 iterations are shown in Figure 3, where v_1 and v_3 are selected as seed nodes and the decay factor α is set as 0.5.

$$\text{spammer-score}_{i+1} = 0.5 \cdot \begin{pmatrix} 0 & \frac{5}{8} & \frac{3}{8} \\ \frac{5}{7} & 0 & \frac{2}{7} \\ \frac{3}{5} & \frac{2}{5} & 0 \end{pmatrix} \cdot \text{spammer-score}_i + (1 - 0.5) \cdot \begin{pmatrix} 1 \\ 0 \\ -1 \end{pmatrix}$$

$i = 10$	v_1	v_2	v_3
spammer-score(v_x)	0.38621816	-0.42241633	0.03619808

Fig. 3. Spammer score computation and results for the example in Figure 1.

4 Evaluation

Evaluation was performed on the competition's data set. Due to the time constrains, we were not able to do extensive evaluation, investigating the results for different parameter settings, or do an in-depth analysis of the submitted results. Since we were only allowed to submit one run, we will try to provide more results for the final paper.

4.1 Data Set

The data set comes from bibsonomy[2] and was manually extended with spammer information[3]. Table 1 shows the properties of the provided training and test data set. The data consists mainly of tagged bookmarks rather than tagged bibtex entries (2%) and bookmark tag assignments are more likely to be spam compared to bibtex tag assignments (90.229% vs. 0.264% in the training data).

Table 1. Training and test data sets

	Training Data Set	**Test Data Set**
Users	31,715	7,205
Spammers	29,248	7,034
Tag Assignments	14,074,725	2,743,743
Tag Assignments from Spammers	13,258,759	2,612,634

4.2 Results

We evaluated different configurations. Firstly, we only considered co-occurence of tag-resource pairs between users. That means, only if two users assigned the same tag to a certain resource we created an edge in the graph for these two users. Secondly, we added resource co-occurence edges to the graph. Still ongoing are evaluations for other configurations like including tag co-occurence between users. The confusion matrices for the first two configurations can be seen in Table 2.

Table 2. Confusion matrices for different configurations

Only Tag-Resource Pairs Co-occurence			
True Positives:	**6085**	True Negatives	**13**
False Positives:	**158**	False Neagatives	**949**
Tag-Resource Pairs and Resources Co-occurence			
True Positives:	**6202**	True Negatives	**2**
False Positives:	**169**	False Neagatives	**832**

Table 3 shows *accuracy* and *ROC AUC* values. Since we only assigned boolean values to users the ROC curve is not very interesting and we ommit it here.

[2] http://www.bibsonomy.org

[3] http://www.kde.cs.uni-kassel.de/ws/rsdc08/dataset.html

Table 3. Accuracy and ROC AUC for different configurations

Strategy	Accuracy	ROC AUC
Tag-Resource Pairs	86.11%	0.4469
T-R Pairs, Resources	84.64%	0.4707

5 Conclusions and Future Work

In this paper, we mainly describe an approach to identify spammers from collaborative tagging systems. The basic idea follows the seminal PageRank approach. The specific feature which distinguishes our approach from existing work is the data structure we employ. In particular, we explicitly model users of collaborative systems as nodes in a graph, since our objective is to detect suspicious spammers. An edge is then created between two users if they co-used a resource, a tag and/or a tag-resource pair. After manually assessing a set of seed users, the scores indicating whether they are spammers or not are propagated through the graph. The intuitive is that nonspammers may annotate resources with similar tags, while spammers may have similar interests in particular resources and/or tags. Consequently, as another feature of our approach, we propagate the scores of not only nonspammers but also spammers. The experimental results on the challenge data demonstrate the effectiveness of our approach.

For future work we want to combine our link-based algorithm with a content-based approach. The benefits could be twofold: Firstly we could use the content-based approach to automatically generate the seed set, and secondly we could adjust the weights for propagation based on the content analysis. To improve our link-based algorithm we try to find more connections between users to minimize the number of unreachable partitions in the graph. The assignments of probabilities to users instead of boolean values could also comprise some potential for improvement. For real world applications, the question of seed set selection poses another interesting task which needs to be solved.

6 Acknowledgements

This work is supported by the EU project IST 45035 - Platform for searcH of Audiovisual Resources across Online Spaces (PHAROS).

References

1. Marlow, C., Naaman, M., Boyd, D., Davis, M.: Ht06, tagging paper, taxonomy, flickr, academic article, to read. In Wiil, U.K., Nürnberg, P.J., Rubart, J., eds.: Hypertext, ACM (2006) 31–40
2. Tso-Sutter, K.H.L., Marinho, L.B., Schmidt-Thieme, L.: Tag-aware recommender systems by fusion of collaborative filtering algorithms. In Wainwright, R.L., Haddad, H., eds.: SAC, ACM (2008) 1995–1999

3. Dmitriev, P.A., Eiron, N., Fontoura, M., Shekita, E.J.: Using annotations in enterprise search. In Carr, L., Roure, D.D., Iyengar, A., Goble, C.A., Dahlin, M., eds.: WWW, ACM (2006) 811–817
4. Bao, S., Xue, G.R., Wu, X., Yu, Y., Fei, B., Su, Z.: Optimizing web search using social annotations. [18] 501–510
5. Koutrika, G., Effendi, F., Gyöngyi, Z., Heymann, P., Garcia-Molina, H.: Combating spam in tagging systems. In: AIRWeb. (2007)
6. Mika, P.: Ontologies are us: A unified model of social networks and semantics. In Gil, Y., Motta, E., Benjamins, V.R., Musen, M.A., eds.: International Semantic Web Conference. Volume 3729 of Lecture Notes in Computer Science., Springer (2005) 522–536
7. Golder, S.A., Huberman, B.A.: The structure of collaborative tagging systems. CoRR **abs/cs/0508082** (2005)
8. Halpin, H., Robu, V., Shepherd, H.: The complex dynamics of collaborative tagging. [18] 211–220
9. Berendt, B., Hanser, C.: Tags are not metadata, but just more content - to some people. In: ICWSM. (2007)
10. Hotho, A., Jäschke, R., Schmitz, C., Stumme, G.: Information retrieval in folksonomies: Search and ranking. In Sure, Y., Domingue, J., eds.: ESWC. Volume 4011 of Lecture Notes in Computer Science., Springer (2006) 411–426
11. Gyöngyi, Z., Garcia-Molina, H., Pedersen, J.O.: Combating web spam with trustrank. In Nascimento, M.A., Özsu, M.T., Kossmann, D., Miller, R.J., Blakeley, J.A., Schiefer, K.B., eds.: VLDB, Morgan Kaufmann (2004) 576–587
12. Heymann, P., Koutrika, G., Garcia-Molina, H.: Fighting spam on social web sites: A survey of approaches and future challenges. IEEE Internet Computing **11**(6) (2007) 36–45
13. Xu, Z., Fu, Y., Mao, J., Su, D.: Towards the semantic web: Collaborative tag suggestions. In: WWW2006: Proceedings of the Collaborative Web Tagging Workshop, Edinburgh, Scotland (2006)
14. Kleinberg, J.M.: Authoritative sources in a hyperlinked environment. J. ACM **46**(5) (1999) 604–632
15. Surowiecki, J.: The Wisdom of Crowds. Anchor (August 2005)
16. Page, L., Brin, S., Motwani, R., Winograd, T.: The pagerank citation ranking: Bringing order to the web. techreport (1998)
17. Wu, B., Goel, V., Davison, B.D.: Propagating trust and distrust to demote web spam. In Finin, T., Kagal, L., Olmedilla, D., eds.: MTW. Volume 190 of CEUR Workshop Proceedings., CEUR-WS.org (2006)
18. Williamson, C.L., Zurko, M.E., Patel-Schneider, P.F., Shenoy, P.J., eds.: Proceedings of the 16th International Conference on World Wide Web, WWW 2007, Banff, Alberta, Canada, May 8-12, 2007. In Williamson, C.L., Zurko, M.E., Patel-Schneider, P.F., Shenoy, P.J., eds.: WWW, ACM (2007)

Combining Clustering with Classification for Spam Detection in Social Bookmarking Systems

Antonia Kyriakopoulou and Theodore Kalamboukis

Department of Informatics,
Athens University of Economics and Business,
76 Patission St., Athens, GR 104.34
{tonia.tzk}@aueb.gr
http://pages.cs.aueb.gr/ipl/ir/

Abstract. *This paper addresses the problem of learning to classify texts by exploiting information derived from both training and testing sets. To accomplish this, clustering is used as a complementary step to text classification, and is applied not only to the training set but also to the testing set. This approach allows us to study the location of the testing examples and the structure of the whole dataset, which is not possible for an inductive learner. The incorporation of this knowledge to the feature representation of the texts is expected to boost the performance of a SVM/TSVM classifier. Experiments conducted on tasks and datasets provided in the framework of the ECML/PKDD 2008 Challenge Discovery on spam detection on social bookmarking systems, demonstrate the effectiveness of our approach. The experiments show substantial improvements on classification performance.*

Key words: classification, clustering, spam detection

1 Introduction

Text classification and clustering have been the focus of critical research in the areas of machine learning and artificial intelligence. In the literature, these two streams flow independently of each other, despite their akin close conceptual and practical relations. However, there are several important research issues encapsulated into text classification tasks and the role of clustering in support of these tasks is also of great significance.

A standard research issue for text classification is the creation of compact representations of the feature space and the discovery of the complex relationships that exist between features, documents and classes. In this vein, an important area of research where clustering is used to aid text classification is the area of dimensionality reduction. Clustering is used as a *feature compression and/or extraction method*: features are clustered into groups based on selected clustering criteria. Feature clustering methods create new, reduced-size event spaces by joining similar features into groups. They define a similarity measure between features, and collapse similar features into single events that no longer distinguish among their constituent features. Typically, the parameters of the cluster become the weighted average of the parameters of its constituent features. Two types of clustering have been identified: i) one-way clustering, i.e. feature clustering based on the distributions of features in the documents or classes [2],[16] and ii) co-clustering, i.e. clustering both features and documents [4].

A second research area of text classification where clustering has a lot to offer, is the area of *semi-supervised learning.* Training data contain both labelled and unlabelled examples. Obtaining a fully labelled training set is a difficult task; labelling is usually done using human expertise, which is expensive, time consuming, and error prone. Obtaining unlabelled data is much easier since it involves collecting data that are known to belong to one of the classes without having to label them. Clustering is used as a method to extract information from the unlabelled data in order to boost the classification task. In particularly, clustering is used: i) to create a training set from the unlabelled data [5], ii) to augment the training set with new documents from the unlabelled data [18], [19], iii) to augment the dataset with new features [13], [9], [10], and iv) to co-train a classifier [14], [11].

Finally, *clustering in large-scale classification problems* is another major research area in text classification. A considerable amount of work is done on using clustering to reduce the training time of a classifier when dealing with large data sets. In particular, while SVM classifiers (see [3] for a tutorial) have proved to be a great success in many areas, their training time is at least $O(N^2)$ for training data of size N, which makes them non favourable for large data sets. The same problem applies to other classifiers as well. In this vein, clustering is used as a down-sampling pre-process to classification, in order to reduce the size of the training set resulting in a reduced dimensionality and a smaller, less complex classification problem, easier and quicker to solve [17], [1]. However, it should be noted that dimensionality reduction is not accomplished directly using clustering as a feature reduction technique as discussed earlier, but rather in an indirect way through the removal of training examples that are most probably not useful to the classification task and the selection of the most representative redundant training set. In most of the cases this involves the collaboration of both clustering and classification techniques.

For a detailed review and interpretation of the role of clustering in different fields of text classification see [12].

In this paper, we deal with the text classification aided by clustering scenario and apply it to the problem of spam detection in social resource sharing systems. Social resource sharing systems are web–based systems that allow users to upload their resources, and to label them with arbitrary words, so–called *tags.* The systems can be distinguished according to what kind of resources are supported. The system under investigation is called BibSonomy[1] and it is a social bookmark and publication sharing system that allows sharing bookmarks and

[1] http://www.bibsonomy.org

BibTex entries simultaneously. A formal description of the underlying structure which is called *folksonomy* is given in [7].

The paper is organized as follows: next section presents the algorithm. Section 3 presents the empirical evaluation. We conclude by pointing out open issues and limitations of the algorithm presented.

2 The Algorithm–Using Clustering For Text Classification

Consider a k–class categorization problem, ($k = 1$ in the case of spam detection on social bookmarking systems), with a labeled training sample $Tr = \{(\boldsymbol{x_1}, y_1), \ldots, (\boldsymbol{x_l}, y_l)\}$ of feature vectors $\boldsymbol{x} \in \mathcal{R}^n$ and corresponding labels $y_i \in \{1, \ldots, k\}$, and an unlabeled testing sample $Te = \ \boldsymbol{x_1^*}, \ldots, \boldsymbol{x_m^*}\ $ of feature vectors. The features are valued using the *TF*IDF* weighting scheme [15], defined by

$$W(f_i) = TF(f_i, \boldsymbol{x}) * IDF(f_i) \tag{1}$$

where the *term frequency* of feature f_i, $TF(f_i, \boldsymbol{x})$, is the number of its occurences in document $\boldsymbol{x}$, the *inverse document frequency* is

$$IDF(f_i) = \log_2 \quad \frac{|D|}{DF(f_i)} \tag{2}$$

where $|D| = |Tr \cup Te|$ is the number of documents in the dataset, and the *document frequency*, $DF(f_i)$, is the number of documents that contain f_i at least ones. All feature vectors are normalized to unit length.

The algorithm consists of the following three steps:

- *Clustering step:* to cluster both the training and testing set.
- *Expansion step:* to augment the dataset with *meta*–features originated from the clustering step.
- *Classification step:* to train a classifier with the expanded dataset.

2.1 Clustering Step

For the clustering step of the algorithm, we need to define the desired number of clusters into which the dataset should be clustered. Results from experiments [10] conducted on three widely used corpora (Reuters, 20Newsgroup, and WebKB) have shown an increase of performance of classification when the number of clusters is equal to the number of the predefined classes. In traditional classification tasks it can be assumed that the classes correspond to topics, and there is a one-to-one correspondence between the topic and the class under which the data are classified. Moreover, the examples of a class are clustered together which is logical since they share the same word distribution. So we can assume that there is a one-to-one correspondence between classes, topics and clusters, and use this information to define the desired number of clusters. In spam detection on social bookmarking systems we can't make such safe assumptions. Spam user posts can deal with many different topics, there is a one-to-many correspondence between the class *spam* and the topics of the posts that fall under it. So, the number of topics can't be determined beforehand. Hense, the number of clusters to select is two: one cluster with the spam posts and one cluster with the non-spam.

The $CLUTO^{TM}$ Clustering Toolkit [8] is used and a divisive clustering algorithm with repeated bisections is selected for clustering *both* the training and testing sets. In this method, the disired k–way clustering solution is computed by performing a sequence of $k-1$ repeated bisections. The dataset is first clustered into two groups, then one of these groups is selected and disected further. This process continuous until the desired k number of clusters is found. During each step, the cluster is bisected so that the resulting k–way clustering solution optimizes the internal criterion function

$$\max \overset{k}{\underset{g=1}{\quad}} \overline{\underset{u,v\in S_g}{\quad\quad} sim(u,v)} \tag{3}$$

where S_g is the set of documents assigned to the g^{th} cluster, u and v represent two documents, and $sim(u,v)$ is the similarity between two documents. The generated set of clusters $G = \{G_1, G_2, \ldots, G_k\}$ consists of k non–overlapping clusters.

2.2 Expansion Step

In the expansion step, each cluster in G contributes one *meta*–feature to the feature space of the training and testing sets, i.e. k *meta*–features are created. The weight of these *meta*–features is computed applying the *TF*IDF* weighting scheme to the clusters. We consider that all the documents in a cluster G_j share the same *meta*–feature mf_j whose *frequency* within a document $\boldsymbol{x}$ of the cluster equals to one, $TF(mf_j, \boldsymbol{x}) = 1$, its *document frequency* equals to the size of the cluster, $DF(mf_j) = |G_j|$, and its *inverse document frequency* is $IDF(mf_j) = \log_2 \ \frac{|D|}{|G_j|}$. Then by properly adjusting Equation 1 the weight of mf_j is defined by

$$W(mf_j) = \log_2 \ \frac{|D|}{|G_j|} \tag{4}$$

2.3 Classification Step

Finally, in the classification step the SVM$^{\text{light}}$ implementation of SVMs and TSVMs is used. A binary classifier is constructed for the *expanded* dataset, a linear kernel is used and the weight C of the slack variables is set to default.

3 A Performance Study

3.1 Experiment Settings

The empirical evaluation is done in two tasks created and published in the framework of ECML/PKDD 2008 Discovery Challenge[2].

- Task A deals with spam detection in social bookmarking systems. The goal of this task is to learn a model which predicts whether a user is a spammer or not. In order to detect spammers as early as possible, the model should make good predictions for a user when he submits his first post.
- In Task B the aim is to support the user during the tagging process and to facilitate the tagging. BibSonomy includes a tag recommender. This means that when a user finds an interesting bibtex or bookmark and posts it to BibSonomy, the system offers up to ten recommended tags on the posting page. The goal is to learn a model which effectively predicts the tags a user will use to describe his post.

The dataset provided consists of data, in the form of posts, collected from 2.638 active non-spam users and 36.282 spam users by manually labeling spammers and non–spammers. It was devided in a training set which was provided at the beginning of the competition, and a testing set which was released 48 hours before the deadline. Users' posts are either bibtex or bookmarks. They include all public information such as the url, the description, the title and the user defined tags.

The evaluation criterion prescribed by the competition is the AUC value.

3.2 Results

Several experiments were conducted during the contest on the given training set as well as after the publication of the true classification labels of the testing set. In this paper only the results from experiments on the whole dataset are presented. In all the experiments, the given dataset was pre-processed as follows. First, for each user, all his posts, BibTex and bookmarks, were considered as one record (feature vector), i.e. there were no multiple records per user as in the original dataset. Then, different versions of this dataset were created. One containing all public information including tags and urls, a second containing all public information except from tags and urls, a third without the tags, and a fourth without the urls. The reason for the different versions created was to examine the impact of using tags and urls in the spam detection process. As stated in [6] tags are considered as one of the ways that a spam user can use to corrupt a bookmarking system. The more often a web page is submitted and tagged, the better chance it has of being found. Spam users bookmark the same web page multiple times and tag each page of their web site using a lot of popular tags.

[2] Additional information can be found in http://www.kde.cs.uni-kassel.de/ws/rsdc08/

According to this fact, we can hypothesize that a url is a serious indicator of a spam user, whereas a tag is a deceptive one.

Also, we wanted to examine the effect of applying stemming and stopword removal mechanisms to the dataset. A series of experiments was conducted in this basis too. It should be noted that numbers, words with length less than two and punctuation marks where discarded for all datasets. Finally, the *TF*IDF* weighting scheme is applied and all users' vectors are normalized to unit length.

The following experiment scenarios were conducted:

- *case1*: the dataset is used without the tags and urls, whereas stemming and stopword removal are applied .
- *case2*: the dataset is used without the tags and urls, and stemming and stopword removal are not applied.
- *case3*: the dataset is used with the tags and urls, and stemming and stopword removal are applied.
- *case4*: the dataset is used with the tags and urls, whereas stemming and stopword removal are not applied.
- *case5*: the dataset is used without the tags, with the urls, whereas stemming and stopword removal are applied.
- *case6*: the dataset is used without the tags, with the urls, and stemming and stopword removal are not applied.
- *case7*: the dataset is used with the tags, without the urls, whereas stemming and stopword removal are applied.
- *case8*: the dataset is used with the tags, without the urls, and stemming and stopword removal are not applied.

We applied our algorithm only on Task A. The results of these experiments are presented in Table 1. To provide a baseline for comparison, results from the standard SVM and transductive SVM (TSVM) classifiers are also presented. C-SVM and C-TSVM correspond to the perfomance of an SVM/TSVM classifier when it is used in conjuction with clustering according to our algorithm.

Table 1. Results for Task A.

	SVM	C-SVM	TSVM	C-TSVM
Case1	82.03	82.72	89.15	89.17
Case2	82.33	82.92	89.76	90.10
Case3	84.68	85.86	93.34	93.64
Case4	85.26	86.42	93.04	94.54
Case5	84.79	85.61	90.05	92.95
Case6	85.00	86.28	90.93	91.84
Case7	84.69	85.54	90.02	92.44
Case8	83.77	83.78	90.04	91.65

First of all, we can see that the C-SVM and C-TSVM classifiers perform better than the standard SVM and TSVM classifiers. In all cases, the C-TSVM classifier has the best performance over the rest of the classifiers. The best result of 94.54% is obtained when the tags and urls are both included in the dataset and stemming and stopword removal are not applied (case 4). In almost all cases, the application of stemming and stopword removal deteriorates performance. The inclusion or not of tags and urls in the dataset also affects performance. When tags and urls are both included in the dataset as in cases 3 and 4, the classification performance is higher in contrast with the cases 1 and 2 where tags and urls are excluded. Also, when tags and urls are used in turn as in cases 5 to 8, we can see that the performance is better when urls are used whereas tags are not used in the dataset. The results confirm the original hypothesis that stated that a url is a serious indicator of a spam user, whereas a tag is a deceptive one.

4 Conclusions

We presented empirical results on datasets given in the framework of the ECML/ PKDD Discovery Challenge 2008 on spam detection in social bookmarking systems. On all experiments conducted, the clustering approach combined with a SVM/ TSVM classifier showed improvements over the use of a standard SVM/TSVM classifier on its own. The applycation of stemming and stopword removal mechanisms, revealed a deterioration in performance and their use is not encouraged in this context. Also, the results confirm the hypothesis that urls can be considered as a serious indicator of spam users, whereas tags can be deceptive.

One limitation of our algorithm is that when a new user makes his first post, the same procedure of clustering, *meta*–feature addition, and classification, should be applied again for the whole dataset, a rather time consuming, and computationally expensive process. A suggestion would be to use incremental clustering instead of the static clustering algorithm used now. Incremental clustering is a method that deals with the problem of updating clusters without frequently performing complete reclustering. This would be a more suitable way for maintaining clusters in the typical, dynamic environment of spam detection.

Another issue about our algorithm is its rather naive approach to clustering that may not capture all the *meta*–information possible hidden in the dataset. More sophisticated clustering methods have been proposed in the literature that focus on incorporating prior knowledge into the clustering process; conceptual clustering, topic–driven clustering, just to name a few. These methods are based in the idea that it is possible to use explicitly available domain knowledge to constrain or guide the clustering process. In our case, the class labels of the training set can constitute the domain knowledge and be used as guidance to a clustering algorithm. This way, it can be reassured that the created clusters will reflect the major concepts included in the corpus.

Other issues that can be further researched include the estimation and statistical basis of the optimum number of clusters and *meta*–features to be used.

References

1. Awad, M., Khan, L., Bastani, F., Yen, I. L.: An effective support vector machines (SVMs) performance using hierarchical clustering. In: 16^{th} IEEE International Conference on Tools with Artificial Intelligence, pp. 663–667, (2004)
2. Baker, L. D., McCallum, A.: Distributional Clustering of Words for Text Classification. In: ACM SIGIR 1998, pp. 96–103 (1998)
3. Burges, J. C.: A Tutorial on Support Vector Machines for Pattern Recognition. In: Knowledge Discovery and Data Mining, vol. 2, pp. 121–167, (1998)
4. Dhillon, I. S., Mallela, S., Kumar, R.: A Divisive Information–Theoretic Feature Clustering Algorithm for Text Classification. Journal of Machine Learning Research, 3, pp. 1265–1287 (2003)
5. Fung, G., Mangasarian, O. L.: Semi–Supervised Support Vector Machines for Unlabeled Data Classification. Technical report (2001)
6. Hammond, T., Hannay, T., Lund, B., Scott, J.: Social Bookmarking Tools (I): A General Review. D-Lib Magazine 11, Nr. 4, (2005)
7. Hotho, A., Jaschke, R., Schmitz, C., Stumme, G.: BibSonomy: A Social Bookmark and Publication Sharing System. In: 1^{th} Conceptual Structures Tool Interoperability Workshop at the 14th International Conference on Conceptual Structures. Aalborg: Aalborg Universitetsforlag, pp. 87–102 (2006)
8. Karypis, G.: CLUTO a clustering toolkit.Technical report (2002)
9. Kyriakopoulou, A., Kalamboukis, T.: Text classification using clustering. In: ECML-PKDD Discovery Challenge Workshop (2006)
10. Kyriakopoulou, A., Kalamboukis, T.: Using clustering to enhance text classification. In SIGIR 2007, 30^{th} Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp805–806 (2007)
11. Kyriakopoulou, A.: Using Clustering and Co–Training to Boost Classification Performance. In: ICTAI (2). IEEE Computer Society, pp. 325–330 (2007)
12. Kyriakopoulou, A.: Text Classification Aided by Clustering: a Literature Review. I-Tech Education and Publishing KG, Vienna, Austria (2008)
13. Raskutti, B., Ferra, H. L., Kowalczyk, A.: Combining clustering and co–training to enhance text classification using unlabelled data. In: KDD ACM 2000, pp. 620–625 (2000)
14. Raskutti, B., Ferra, H. L., Kowalczyk, A.: Using Unlabelled Data for Text Classification through Addition of Cluster Parameters. In: 19^{th} International Conference for Machine Learning (ICML 2002), pp. 114-121 (2002)
15. Salton, G., McGill, M. J.: Introduction to Modern Information Retrieval. McGraw Hill (1983)
16. Slonim, N., Tishby, N.: The power of word clustering for text classification. In: European Colloquium on IR Research, ECIR 2001, (2001)
17. Yu, H., Yang, J., Han, J.: Classifying large data sets using SVMs with hierarchical clusters. In: 9^{th} ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 306–315, Washington, DC, USA, ACM (2003)
18. Zeng, H. J., Wang, X. H., Chen, Z., Lu, H., Ma, W. Y.: CBC: Clustering Based Text Classification Requiring Minimal Labeled Data. In ICDM 2003, pp. 443–450, IEEE Computer Society (2003)
19. Zhou, D., Bousquet, O., Lal, T. N., Weston, J., Scholkopf, B.: Learning with Local and Global Consistency. In: NIPS 2003, MIT Press (2003)

Using Semantic Features to Detect Spamming in Social Bookmarking Systems

Amgad Madkour[1] , Tarek Hefni[2], Ahmed Hefny[1] , Khaled S. Refaat[1]

Human Language Technologies Group
IBM Cairo Technology Development Center[1]
P.O.Box 166 El-Ahram, Giza, Egypt
{amadkour,ahefny,ksaeed}@eg.ibm.com[1] , t_hefny@aucegypt.edu[2]

Abstract. Collaborative software is gaining pace as a vital means of information sharing between users. This paper discusses one of the key challenges that affect such systems which is identifying spammers. We discuss potential features that describe the system's users and illustrate how we can use those features in order to determine potential spamming users through various machine learning models.

1 Introduction

Spamming is a crucial challenge that affects both users and services providers. Users are faced with spamming obstacles during activities such as web-based searching. This often occurs when spammers use techniques or keywords to increase the rank of their websites. This in turn overshadows more relevant pages that users might be actually interested in. A famous form of spamming is that a spammer might create a website with keywords most relevant to a common accessory that a user maybe interested in. Other victims of spamming are email service providers. Email providers use a lot of spam detection techniques in order to minimize spam emails sent to their users.

Spam detection is faced with great challenges and thus various techniques have been deployed. Some techniques are based on detecting the most common keywords or key phrases that are frequently used in most spamming emails. Other techniques are based on learning general patterns that spammers tend to use to advertise their material. Some of them rely mostly on manually constructed pattern matching rules that need to be tuned to each user. A greater challenge is that the characteristics of spam change over time which makes maintaining the rules a daunting task. Other systems employ machine learning techniques which allow the system to automatically learn to separate spam from other messages. Classification techniques, based on different features that describe a user and posts, are used in order to differentiate between spamming and non-spamming users.

[2] Tarek Hefni is an undergraduate student at the American University in Cairo. At the time of this work, he was an intern with IBM Cairo Technology Development Center R&D Team.

In this work we target a specific type of collaborative social software systems that suffers from spamming which is social bookmarking[12]. Its main focus is collecting the user online bookmarks which the user generates. Other similar systems are used to store user-generated scientific bibliographies. The main element in such systems is a post. It's composed of a user, a resource and a number of tags annotating it. Systems such as social bookmarking allow users to upload their resources, and label them with tags. The systems differ depending on the type of resource being shared[7]. On most systems, users are described by their user ID and tags may be arbitrary strings. Users are allowed to copy the resources tagged by other users. They are also allowed to view information such as who tagged what resource and so forth. The power of such a system is that users are able to share or browse other users' shared data.

The set of tag-resource relations is referred to as *folksonomy*[1]. It stands for conceptual structures created by people. The collection of all user assignments is called a user *personomy.* A collection of all personomies in turn constitutes the folksonomy. There is widespread use of these systems due to the presence of sharing mechanisms between users which enable breaking the knowledge acquisition problem users face.

In this paper, we discuss various features that can be used to identify spammers that target such systems. The main motivation of our work is to capture the necessary features that discriminate spammers from non-spammers. Our second motivation is to determine what an efficient classification model would be like.

2 Related Work

Li and Hsieh [11] proposed a group-based anti-spam framework investigating the clustering structures of spammers based on spam traffic collected at a domain mail server. Their study showed that the relationship among spammers demonstrates highly clustering structures based on URL grouping.

Krause and Schmitz [9] proposed a social bookmarking setting to identify spammers using features based on the topological, semantic and profile-based information. They used different classification techniques to evaluate their proposed features. Their best results were achieved using SVM scoring a roc area of 0.936.

Koutrica et al [8] introduced a framework for modeling tagging systems and user tagging behavior. They proposed a tagging system where malicious tags and malicious user behaviors are well-defined, and described and studied a variety of query schemes and moderator strategies for tag spam detection.

Androutsopoulos et al [3] explored the idea that a Naive Bayesian classifier could be used to filter spam mail. They also investigated the effect of some parameters on the performance of the filter such as attribute-set size, training-corpus size, lemmatization, and stop-lists. They discovered that the filter has a stable significant positive contribution with additional safety nets like resending private messages but is not viable when blocked messages are deleted.

Beil et al [5] introduced an approach which uses frequent item (term) sets for text clustering. They used algorithms for association rule mining to discover such frequent sets. They presented two algorithms for frequent term-based text clustering: FTC and HFTC where the first creates flat clustering and the second allows hierarchical clustering respectively.

Wetzker et al [12] analyzed 150 million bookmarks. They showed how bookmarks are vulnerable to spamming and how to limit such vulnerability to avoid affecting the analysis process.

Hotho et al [2] specified a formal model for folksonomies and described their system: BibSonomy, used for sharing both bookmarks and publication references in a personal library. They showed that BibSonomy is valuable for researchers because of the fact that it combines both bookmarks and publication entries.

Gomes and Cazita [10] provided a characterization of spam traffic using workload variation, density, inter-arrival time distribution, email size distribution, temporal locality, etc., compared with non-spam emails. They showed that non-spam email transmissions are typically driven by bilateral social relationship while spam transmissions are usually unilateral actions based on the spammers' will to reach a large number of recipients.

3 Semantic Features

Semantic features refer to annotations that are derived from the content of the resources. We are motivated to show the value of using semantic features as a means of detecting spammers.

For the first set of features, we used those mentioned by Krause et al. [9]. They proposed a set of features that were extracted from a dataset with comparable characteristics to the one under investigation. Our contribution is the usage and creation of semantic features that could contribute to the classification accuracy.

The first feature [9] deals with counting the number of tags of the user which contain 'group=public'. This feature measures the amount of tags that are publicly shared with the social bookmarking community. This is used by spammers in order to increase the exposure of spammed material to the public.

For the second and third features [9], we counted the number of resources that are common between the current user and non-spammers in the training set and the number of resources that are common between the user and spammers in the training set, respectively. Common resources could be shared bookmarks or shared BibSonomy items. Those two features allow measuring a ratio between resources that the current user shares with the spammers and non-spammers community.

Following the same concept for the fourth and fifth features [9], we counted the number of tags that are common between the users and non-spammers in the training set as well as the number of tags that are common between the users and spammers in the training set. It is important to note that the difference between those two features and the previous two is that a resource could be assigned to more than one tag.

Similarly for the sixth and seventh features [9], we counted the number of resource-tag couples that are common between the user and non-spammers in the training set and the number of resource-tag couples that are common between the user and spammers in the training set. This gives an indication about the ratio of resources-tags for spammers and non spammers.

For the eighth, ninth and tenth features [9], we calculated the co-occurrences of some of the previously mentioned features. The eighth feature is a calculation of the co-occurrence generated when we compute the ratio between the second and third features. The ninth feature is a calculation of the ratio between the fourth and fifth features. The tenth feature is a calculation of the ratio between the sixth and seventh features. All the co-occurrences features are based on the assumption that spammers share the same vocabulary as non-spammers [9].

We are motivated to capture the keywords that always co-occur with spammed material [9]. Those keywords are referred to as black-listed words. The black-listed words are a list of words that generally occur in spammed material, such as emails or websites, with high frequency. We developed a weighted version of that list in order to use it for our proposed features. Using the training set, we gave each word a score which represents the frequency between word repetitions by spammers divided by frequency between word repetition by non-spammer. We used the same technique for both tags and descriptions.

For the eleventh feature, we calculated the total score of each word in the description of every bookmark for each user divided by the number of bookmarks for the user. For the twelfth feature, we calculated the total score of each word in the tags of every bookmark for each user. Those two features allow us to capture the frequency of black-listed keywords within resources such as bookmarks and tag names.

For our last feature, we counted how many times a tag was repeated with other tags for the same resource by non-spammers. We created what we call a tag-pair which consists of the two tokens inside the tag. The total score of the feature is calculated as follows: for each tag-pair of each resource, if the user used a defined tag-pair, we add one divided by the tag-pair score, otherwise, we add three.

4 Dataset

We use the dataset provided for the ECML PKDD Discovery Challenge 2008. The dataset consists of users and their posts. The information includes all public information such as the URL, the description and all tags of the post. The training data was composed of 22,200 patterns and the testing set was composed of 9,959 which included 741 non-spamming users. In this paper, we report the results obtained by using the testing set provided at training time.

5 Evaluation

The problem will be evaluated using AUC (Area Under ROC Curve). AUC estimates the probability that a randomly chosen positive instance will be ranked higher than a randomly chosen negative instance [7]. It shows the relative tradeoffs between benefits (true positives rates) and costs (false positives rates)[13].

6 Experimental Setup

We experimented with five models: K-Nearest Neighbor Regression, Gaussian Processes [6], Support Vector Machine (SVM), Neural Networks and Ensemble Learning using both SVM and Neural Networks. We used Rapid Miner[1] as our machine learning toolkit.

6.1 K-Nearest Neighbor Regression

The following table shows the AUC values obtained using different values of k. A low value of k results in an input-sensitive model while a high value of k results in a smoothing model.

Length Scale	**AUC**
3	0.863233176
5	0.87998662
7	0.886212826
13	0.901224341
15	0.900489829
17	0.902808017
19	0.904737342
30	0.908758165
35	0.90963878
40	0.909391328
50	0.909341884

Table 1. K-Nearest Neighbor Results

The best AUC value (0. 90963878) is achieved at k = 35. It is worth noting that both Gaussian Processes and KNN regression performed best at a point of notably high smoothing (low variance).

[1] http://rapid-i.com

6.2 Gaussian Processes

We used Gaussian Processes[6] with Radial Basis Function (RBF) kernel while restricting the maximum number of basis functions to 100. This restriction significantly reduced training time without any notable change to the AUC value (compared to 1000 basis functions). The following table shows the obtained AUC values on the validation set for different values of the Length-Scale of the RBF kernel. A low value of L results in an input-sensitive (high variance) model while a high value of L results in a smoothing model.

Length Scale	**AUC**
3	0.770784148624904
10	0.781429115194297
20	0.84661283935238
40	0.921302082732583
80	0.932317150535772
81	0.929417857595996
82	0.929390174977221
83	0.929390174977221
85	0.926760095505033
90	0.923650491558724
100	0.915941266710499
150	0.886343702780949

Table 2. Gaussian Processes Results

The best AUC value (0.932317150535772) is achieved at length scale of 80, a relatively high value.

6.3 Support Vector Machines

As for the SVM, we used cost-based learning [14][15][4]. Assigning a cost of 7 for labeling a user as non-spammer produced highest results. Accordingly, we reached the following figures:

Kernel	**Gamma/Pol.Degree**	**AUC**
RBF	1	0.9501623
RBF	1.2	0.9521352
RBF	2	0.9509834
RBF	7	0.9487790
Polynomial	9	0.9519381

Table 3. Support Vector Machine Results

To improve results, we employed a cascading scheme in which an SVM model (of configuration 2) is used to classify data patterns and another SVM model is used to reclassify difficult patterns. A difficult pattern is a pattern that produced an output in the range between -1 and 1 in the first SVM model, meaning that there is too little confidence to classify it. The second model is trained using the difficult points in the training set; we classified our training set using the first model and extracted the difficult points which we used to train the second model. Adding the second model with parameters different than the first model would assure that the difficult points that are misclassified by the first model are classified correctly by the second model. The table below shows the parameter configurations we tried for the second model (cost value is the same) and the AUC achieved by each.

Kernel	**Gamma/Pol.Degree**	**AUC**
RBF	1.22	0.9526184
RBF	1	0.9525310
RBF	0.9	0.9487790
Polynomial	9	0.9518192

Table 4. Support Vector Machine Results with Cascading Schema

It was observed that using one classifier misclassified 350 users of the 9950 users of the test set.

6.4 Neural Networks

We used a Neural Network Model with learning rate 0.6 and momentum 0.3 for 9000 epoch. The network contained one hidden layer of thirty neurons with a sigmoid activation function. The model resulted in an AUC of 0.9394533. Decreasing the learning rate to 0.4 decreased the AUC to 0.9238232.

6.5 Combined NN and SVM

Finally, we attempted an ensemble learning scheme where we averaged the outputs of the best two SVM models and the best Neural Network model. This produced an AUC of 0.9425323421.

Using the proposed features of [9] to train our models with the first ten features, we achieved an AUC of 0.941243347. Adding the eleventh feature alone (weighted black list feature) resulted in an AUC of 0.950537472. By adding the final feature, we reach our best model giving an AUC value of 0.9526184.

7 Conclusion

In this paper we discussed the semantic features that can be used to detect spammers in a social bookmarking system. Our proposed features demonstrate improved results compared to the ones proposed by [9] on the competition test set taking into consideration the comparable dataset. The paper also discussed the results obtained by training various classifiers. In addition, this paper demonstrated how the cascading scheme model provided better results and partially tackled the border-line of classification that [9] mentioned. We used the Area Under ROC Curve method to evaluate our results and the best result obtained was 0.9526.

References

1. A Capocci, G Caldarelli. Folksonomies and clustering in the collaborative system CiteULike.
2. A Hotho, R Jaschke, C Schmitz, G Stumme. BibSonomy: A Social Bookmark and Publication Sharing System.
3. Androutsopoulos. An Evaluation of Naive Bayesian Anti-Spam Filtering. Proceedings of the workshop on Machine Learning in the New Information Age, G. Potamias, V. Moustakis and M. van Someren (eds.), 11th European Conference on Machine Learning, Barcelona, Spain, pp. 9-17, 2000.
4. B. Scholkopf, A. J. Smola. Learning with Kernels. The MIT Press Cambridge, Massachusetts London, England 2002.
5. Beil et al. Frequent Term-Based Text Clustering. SIGKDD 02 Edmonton, Alberta, Canada
6. Carl Edward Rasmussen, Chris Williams. Gaussian Processes for Machine Learning. the MIT Press, 2006
7. E Santos-Neto, M Ripeanu, A Iamnitchi. Tracking Usage in Collaborative Tagging Communities.
8. G. Koutrika, F. A. Effendi, Z. Gyongyi, P. Heymann, and H. Garcia-Molina.: Combating spam in tagging systems. In Proc. AIRWeb , pages 57 New York, NY, USA, 2007 ACM.
9. Krause, Beate. Schmitz, Christoph. Hotho, Andreas. Stumme, Gerd. The Anti-Social Tagger - Detecting Spam in Social Bookmarking Systems. AIRWeb '08, April 22, 2008 Beijing, China.
10. L. H. Gomes, C. Cazita. Characterizing a Spam Traffic. In the proceeding of IMC 04, Oct. 2004.
11. Li, Hsieh. An Empirical Study of Clustering Behavior of Spammers and Group-based Anti-Spam Strategies. CEAS 2006 Third Conference on Email and Anti-Spam, July 27-28, 2006, Mountain View, California USA.
12. R Wetzker, C Zimmermann, C Bauckhage3. Analyzing Social Bookmarking Systems: del.icio.us Cookbook. July 10th, 2008.
13. T. Fawcett. An Introduction to ROC Analysis. Pattern Recogn. Lett., 27(8)861 2006.
14. Yoav Freund, Robert E. Schapire. Experiments with a new boosting algorithm. In Machine Learning: Proceedings of the Thirteenth International Conference, pages 148156, 1996.
15. Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. Journal of Computer and System Sciences, 55(1):119139, 1997.

Predicting Tag Spam Examining Cooccurrences, Network Structures and URL Components

Nicolas Neubauer and Klaus Obermayer

Neural Information Processing Group, Technische Universität Berlin,
neubauer|oby@cs.tu-berlin.de

Abstract. The task of the ECML/PKDD Discovery Challenge 2008 is to identify spammers in a social bookmarking system. We classify users using three different types of features, based on cooccurences, network properties and url parts. Cooccurrence features are based on the assumption that users associated with similar documents and tags as spammers are likely to be spammers themselves. Network-based features work on a collective scale, assuming common behavioural patterns which can be identified in the graph structures created by tagging activities. Finally, a text classification on the URLs' components identifies frequent terms in spam URLs. With these features, we train an SVM for classification. Our submission run, combining all three classes of features, performed worse than expected from previous tests. With the wisdom of hindsight, we find an optimal choice of features is to leave out network features entirely but to strengthen URL classification. This is, however, a side effect of wrong assumptions about the test set; network features, used alone, still yield positive results. As network features do not depend on the presence of labeled users, they should be further explored to identify structural properties of tag spam even when no ground truth exists.

1 Introduction

Social bookmarking systems have come of age. One of the less pleasant indicators of this development is the arrival of the spammers. This year's ECML/PKDD Discovery Challenge has provided researchers with data from users of a social bookmarking site (`www.bibsonomy.org`), marked as spammers or non-spammers, along with the documents they bookmarked and the tags they used. The goal was to learn a model to distinguish spammers from regular users in an unknown test dataset. We present our approach and the results on the dataset.

So far, not much research has been conducted on spam in social bookmarking sites. [4, 7] have, e.g., simulated the impact of certain spamming practices on the overall properties of a social bookmarking dataset in dependence of a number of key parameters. In [2], spam is mentioned briefly as it causes a deviation from an otherwise smooth strength distribution of a tag network. Most related to our current task is however [8], in which the organizers of this workshop performed experiments on an earlier version of the current dataset. Our work is similar in that we create user

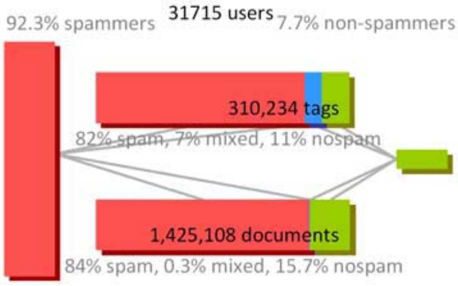


Fig. 1. Distribution of spam in the different sets

features on which we train a classifier. However, we explore other features: We vary the exploitation of cooccurrence patterns already used in [8], but also introduce features based on network analysis, and perform a text classification on the url components of the bookmarks.

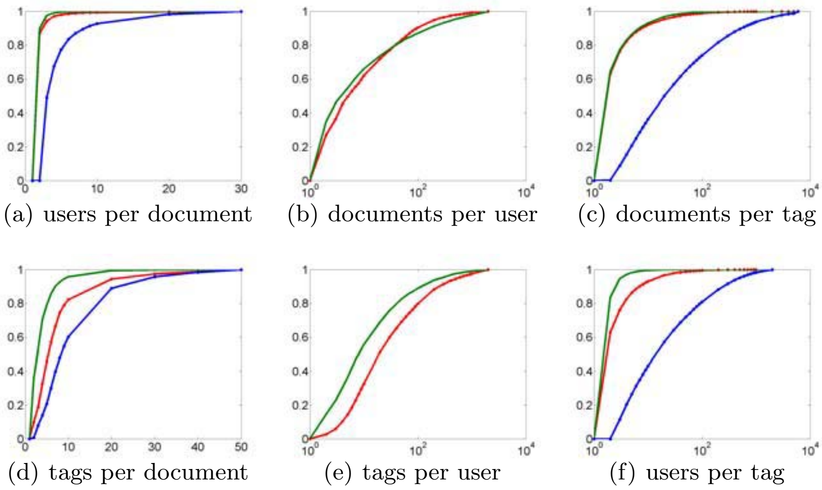
2 Dataset

2.1 Basic Properties

The training dataset provides 14,074,956 triples $E = (d, u, t) \in D \times U \times T$, where D is the set of 1,425,108 documents, U is the set of 31,715 users, and T is the set of 310,234 tags. If we interpret D, U and T as nodes and E as edges, this defines a 3-partite 3-hypergraph. Documents can either be bibliographic references or WWW bookmarks and come with associated metadata like URL, title etc. Users are simply presented as IDs and labeled as spammers or non-spammers. Tags are strings. The merged training and test dataset consists of 16,818,699 triples, 1,574,963 documents, 38,920 users and 396,474 tags. Of the additional 7,205 users, only 171, i.e. 2.37% are regular users.

Figure 1 shows the distribution of spam and non-spam elements in the training dataset. The most striking fact is that 29,248 of 31,715 users are spammers, i.e., only 7.78% are legitimate users. Most tags and documents are used by spammers or non-spammers exclusively and can thus be regarded as spam or non-spam as well. Overlap between use by spammers and non-spammers is very rare in documents, but more frequent among tags. This indicates two different incentives for spammers: They may post spam bookmarks under frequently used tags, such that other users browsing the repository are led to their pages. Posting bookmarks under other, sometimes randomly created tags, probably serves the purpose of creating links from reputated sites (the bookmarking site) to the spammed page, trying to trick search engines into improving the spammed page's rank.

Figure 2 shows the distribution of connections (in the training set) between elements of the different types as the accumulative probability distribution of elements of one type having x connections to elements of any other type. We can see that



(a) users per document (b) documents per user (c) documents per tag

(d) tags per document (e) tags per user (f) users per tag

Fig. 2. Accumulative distribution of number of connected elements for spam(red), non-spam(green) and mixed(blue) elements

- Around 90% of all documents are only bookmarked by one person
- The numbers of tags per document, tags per user and users per tag differ visibly for spam and non-spam elements.
- Documents and tags used by both spammers and non-spammers (blue in the graphs) tend to have strongly different characteristics.

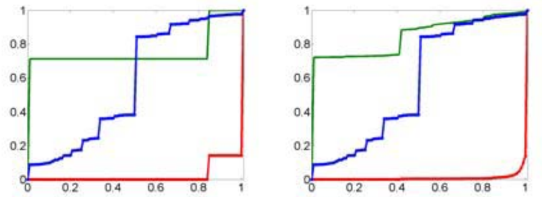
2.2 Creating a probe dataset

Many features we will present in the following use cooccurrence information. With such features, it is important to exclude the cooccurrence information for the users we want to test them on. We therefore split the training dataset into five subsets, each containing the nth fifth of spam and the nth fifth of non-spam users (see Figure 7(a)). Most of our analysis used cooccurrence information from the first four datasets, and evaluated on the fifth ("probe") one. We thereby simulate predicting future users knowing roughly four times as many users from the past.

3 Features

3.1 Cooccurence Features

Distributing Spaminess Let a tag's or a document's spaminess be the frequentist probability that a user using/tagging that element is a spammer. We formalize these notions for documents; they are equivalently definied for tags. Let $U_+(d)$, $U_-(d)$ and $U_?(d)$ be the set of spam, non-spam and unknown users who tagged a document. Then we can define a



(a) simple distribution (b) iterated distribution

Fig. 3. Accumulative document spaminess after spaminess distribution for spam(red), non-spam(green) and mixed(blue) documents

document's spaminess $s(d)$ as well as a certainty $c(d)$ for that measure, based on the fraction of known users, as

$$s(d) = \frac{|U_+(d)|}{|U_+(d)| + |U_-(d)|}, \quad c(d) = \frac{|U_+(d)| + |U_-(d)|}{|U_+(d)| + |U_-(d)| + |U_?(d)|}.$$

If there are no known users for a given element, the certainty is set to 0, and spaminess to the average of all documents. Now, an unknown user's spaminess can be computed as

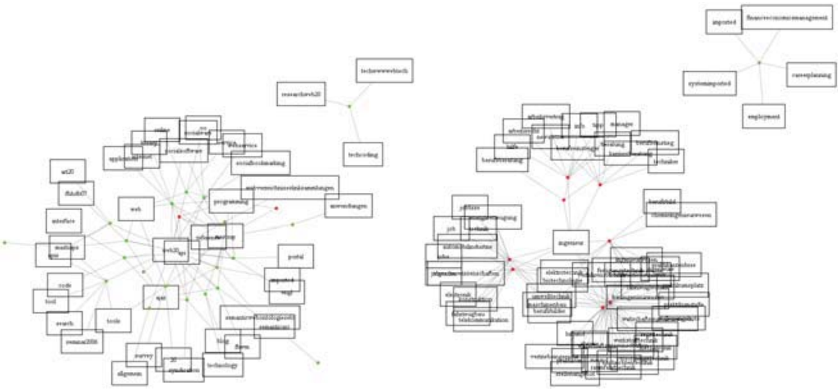
$$s(u) = \frac{{}_{d \in D(u)} s(d)c(d)}{{}_{d \in D(u)} c(d)}, D(u) = \{d \in D : \exists t \in T : (d, u, t) \in E\}$$

This can be interpreted as a graph traversal: Starting from the unknown user u, we choose a document randomly, weighted by its certainty. Then we choose a random known user associated with that document. $s(u)$ is the probability that this user is a spammer.
See Figure 3(a) for the distribution of the resulting values. Non-spam documents have either a value of 0 or, if unknown, the average (around .8), whereas spam documents have a value of either the average or 1. We see that a significant part of both spam and non-spam documents receive the default average value and can thus not be used to classify users. This is due to the high fraction of documents with very few uses (see Figure 2(a)).

Iterated Spaminess Distribution Even if we cannot tell a document's spaminess by the users that have tagged it, we might learn something from the tags it was tagged with. In the same manner, we can estimate a tag's spaminess by the documents it was used for. In order to use this information we compute a second feature for each element, its iterated spaminess s_i and the according certainty c_i. We initialize s_i and c_i with the original values s and c, and then compute

$$c_i(d) = \frac{{}_{t \in T(d)} c_i(t)}{|T(d)|}, T(d) = \{t \in T : \exists u \in U : (d, u, t) \in E\}$$

$$s_i(d) = \frac{{}_{t \in T(d)} : s_i(t)c_i(t)}{c_i(d)}$$



www.programmableweb.com www.ingenieurkarriere.de

Fig. 4. Induced bipartite graph E_D for two documents. Dots are spammers (red) and non-spammers (green)

This process is repeated iteratively for all tags and documents with certainty 0 until no such elements remain or no further information can be spread. Figure 3(b) shows the smoothing effect of this transformation.

Cosine Similarity As introduced in [8], a simple way to exploit coocurence information is to create a similarity function between users based on their used tags, and predict a user's spaminess as the sum of the known users' spaminess weighted by that similarity. We followed that approach by creating tag (and document) vectors for each user such that each component corresponds to a given tag, and the value of that component would be 1 if the user used that tag. Then, the cosine between the two tag vectors serves as a similarity function.

3.2 URL Classification

A central aspect in deciding whether a given bookmark is spam or not should be its content. While we cannot download all bookmarked URLs, we estimate their content by analyzing the terms used in the URLs themselves. We split each URL by the dashes, remove dots, colons, and frequent elements like "http", "www" or "html", and create a feature vector containing a tfidf representation of its terms. We then use the feature vectors of all URLs with spaminess 0 (as computed using the first four fifths of the training set during testing, and using the whole training set for the final prediction) as negative samples, and an equal number of URLs with spaminess 1 as positive samples. After training a linear SVM[3], we classify all URLs with a spaminess between 0 or 1 (urlspam(d)), skipping those URLs which do not contain any known URL part. This yields us with a new user features

$$\mathrm{url}(u) = \frac{{}_{d\in D_{\mathrm{url}}(u)}\,\mathrm{urlspam}(d)}{|D_{\mathrm{url}}(u)|},$$

$$D_{\text{url}}(u) = \{d \in D : (\exists t : (d, u, t) \in E) \wedge \text{urlspam}(d) \text{ is defined}\}$$

In what turned out to be a major wrong design decision, we smoothened this feature by blending it with a user's document spaminess $s(u)$:

$$\text{url}_{\text{smooth}}(u) = \frac{c_{\text{url}}(u)\text{url}(u) + c_{\text{spaminess}}(u)s(u)}{c_{\text{url}}(u) + c_{\text{spaminess}}(u)},$$

where $c_{\text{url}(u)}$ is the fraction of u's documents that have a url prediction and $c_{\text{spaminess}}(u)$ is the average spaminess certainty $c(d)$ for all u's documents.

3.3 Induced Graphs

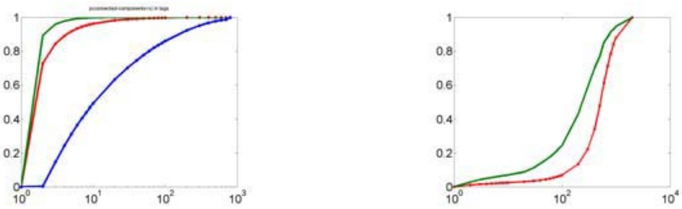
Last, we aimed to translate network features into useful user features. To examine only relevant portions of the overall graph structure, we define "induced graphs" as the bipartite graphs gained by fixing an element from one of the three sets, and connecting those elements from the other two sets that are connected via the fixed element. For example, we might fix a document and then examine all users and tags associated with it, with edges connecting users with the tags they used for the document. More formally, we define the induced graphs by its edges E_D, E_U and E_T obtained by fixing a particular document, user, or tag as

- $E_D(d') = \{(u, t) \in U \times T \mid \exists(d', u, t) \in E\}$
- $E_U(u') = \{(d, t) \in D \times T \mid \exists(d, u', t) \in E\}$
- $E_T(t') = \{(d, u) \in D \times U \mid \exists(d, u, t') \in E\}$

Refer to Figure 4 for two examples of bipartite graphs induced by a (mostly) non-spam- and spam document, respectively, rendered using the GraphViz package[1]. We hope these examples convey the intuition that graphs created by spamming should have different properties than those created by legitimate activity. We used the NetworkX package[2] to obtain various properties of the induced graphs and created user features either by using the resulting values directly (for E_U), or by averaging over the values of all associated elements (for E_D and E_T).

Connected components A connected component of a graph G is a set of nodes such that any node can be reached from any other node in that component by travelling along the edges in G. If a graph is disjoint, the number of connected components describes the number of disconnected subgraphs (both examples in Figure 4 show networks with two connected components). See Figure 5 for the distributions of connnected components across tag-induced graphs and the resulting user features.
We also obtained the number of strongly connected components in each graph (except in those cases where one of the sets of elements was bigger than 1000, due to time reasons). Strongly connected components are components of the graph in which all nodes are connected to each other.

[1] `www.graphviz.org`
[2] `networkx.lanl.org`



(a) Distribution in tag-induced graphs

(b) Distribution of resulting user features

Fig. 5. Connected components

Characteristic Path Length The characteristic path length is the average shortest path distance between two arbitrary nodes in the graph. Again, we computed this number for all graphs in which no element set exceeded 1000 elements.

Degree Ratios Finally, we observed the ratio between the average degrees of each element set, normalized by the size of the other set. For a user-induced graph, this would mean

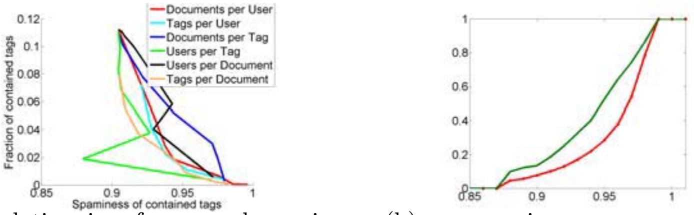
$$d_{d,t}(u) = \frac{\text{avg degree}(D(u))}{|T(u)|} \cdot \frac{\text{avg degree}(T(u))}{|D(u)|},$$

where avg degree(S) is the average degree of all components in S.

3.4 K-Cores

A k-core[9] is a subgraph of a graph G containing all elements that are connected to at least k elements which are also in the core. k-Cores have been used, for example, for the efficient decomposition of large networks[1]. In [5], a k-core of a tagging dataset is used for testing methods requiring a highly connected graph. We extend the definition of a k-core on non-partitioned graphs to a K-core, where $K \in \mathbf{N}^{(n,n)}$ for n-partite graphs such that $K(x,x) = 0 \;\forall\; x \in \{1,\ldots,n\}$. Each entry $K(i,j)$ indicates the minimum number of connections that elements from set i need to have to elements from set j. For the given case of a 3-partite graph with element types (documents, users, tags), we get a $\begin{matrix} & a & b \\ c & & d \\ e & f & \end{matrix}$-Core indicating that each document needs to be connected to a users and b tags, each user needs to be connected to c documents and d tags, and each tag needs to be connected to e documents and f users. Six-dimensional constraints allow for a wide variety of K-cores to be examined. We examine two different types of $K - cores$, regular K-Cores $K_r(a)$ which raise all constraints in parallel, and singular K-Cores $K_s^{n,m}(a)$, raising the single constraint at position (n,m):

$$K_r(a) = \begin{matrix} & a & a \\ a & & a \\ a & a & \end{matrix}, \qquad K_s^{1,2}(a) = \begin{matrix} & a & 2 \\ 2 & & 2 \\ 2 & 2 & \end{matrix}.$$



(a) relative size of cores and spaminess of their members

(b) avg. maximum core spaminess over users' tags

Fig. 6. Cores

We compute $K_r(a)$ for $a \in \{2, 3, 4, 5, 10, 15, 20\}$, and all six $K_s(a)$ for $a \in \{3, 5, 10, 20, 50, 100, 500, 1000, 2000\}$.
Figure 6(a) shows the development of the different singular cores as we increase a: The higher the constraint, the less elements in a core (plotted on the y-axis as the fraction of the total set of tags). However, the spaminess of the remaining elements, in general, increases. For example, the core "documents per tag" ($K_s^{3,1}$), for a certain value of a, contains around 3% of all tags, and those tags are much more likely to be spammy (~.97) than average (~.86). We determine, for each element, the maximum average spaminess (again, computed by the previously available data) of over all cores it is contained in. Figure 6(b) shows the distribution of the corresponding user feature, averaging over users' tags' maximum core spaminess.

3.5 Other Features

Connection Strength For each tag/document pair, we counted the number of users that used it together. We noticed that high values tend to imply spaminess. Therefore, we produced two user features measuring the averages of each document's a) average and b) maximum connection strength .

Counting Finally, we observed the average number of user per document (again averaged to the single user), the average ratio of tags and users per document, and simply the number of entries per user.

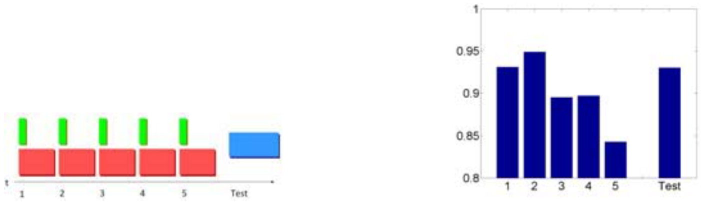
4 Experiments & Results

4.1 Single Features

A detailed list of the features generated from each group is presented in Table 1. It documents the AUC values for each feature when used alone as predictor, both on the probe and the test dataset.

Table 1. Features by group, and AUC value if used as single predictor

			Used by		
Feature name	**AUC(val.)**	**AUC(test)**	**S**	**N**	**O**
Cooccurence Features					
Avg. Spaminess					
Documents	0.676	0.689	s		o
Certainty	0.435	0.413	s		o
Std.Dev	0.445	0.463	s		o
Tags	0.839	0.926	s		o
Certainty	0.552	0.529	s		o
Std.Dev	0.321	0.276	s		o
Avg. Spaminess (iterated)					
Documents	0.813	0.900	s		o
Std.Dev	0.401	0.382	s		o
Tags	0.842	0.918	s		o
Std.Dev	0.327	0.255	s		o
User Cosine Similarity					
Documents	0.554	0.533	s		o
Tags	0.823	0.887	s		o
Avg. Document * Avg. Tag Spaminess					
Normal(tags) * Iterated(documents)	0.843	0.929			
Features of Induced Graphs					
Degree ratios					
Avg (deg(user)/deg(document)) per tag	0.518	0.565	s	n	
Avg (deg(user)/deg(tag)) per document	0.669	0.660	s	n	
deg(docs)/deg(tag)	0.683	0.674	s	n	
Characteristic Path Length					
User	0.607	0.579	s	n	
Avg. per tag	0.375	0.358	s	n	
Avg. per document	0.659	0.658	s	n	
Connected components					
#connected components/#entries by user	0.328	0.350	s	n	
Avg. #connected components per tag	0.681	0.704	s	n	
Strongly connected components			s	n	
Avg per tag	0.551	0.575	s	n	
Avg per document	0.389	0.398	s	n	
User	0.469	0.499	s	n	
Other Features					
Connection strength					
Avg (Avg. connection strength per document)	0.560	0.576	s	n	
Avg (Max. connection strength per document)	0.549	0.559	s	n	
Spaminess of highest containing core					
User	0.500	0.500	s	n	
Avg. per document	0.500	0.532	s	n	
Avg. per tag	0.622	0.641	s	n	
URL classification					
Smoothed avg. URL prediction per document	0.765	0.712	s		
Avg. URL prediction per document	0.814	0.787			o
Counting features					
Avg. #users per document	0.503	0.515	s	n	
Avg. #tags/#users per document	0.674	0.660	s	n	
# entries	0.642	0.627	s	n	o



(a) Splitting training data into 5 probe datasets

(b) AUC value of tag * iter. doc. spaminess on probe and test data

Fig. 7. Construction of probe datasets and performance of spaminess predictors

Cooccurrence Features We find that the average spaminess, particularly of tags, is a strong predictor of a user's spaminess. Iterating spaminess distribution helps to increase the expressiveness of document spaminess. The product of tag spaminess (see Figure 7(b) for performance on the different probe datasets and on the test dataset) and iterated document spaminess performs better than our final predictor on the test set.

URL prediction Apart from cooccurrence features, the url predictor is the strongest single predictor of spaminess. In particular, smoothing does not seem necessary.

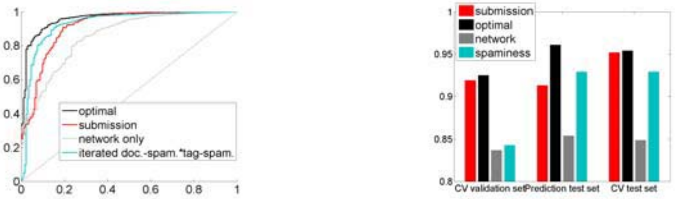
Induced Graphs Many of the features describing the statistical properties of induced graphs seem to indicate a tendency towards spaminess or non-spaminess, but no single feature is useful as a stand-alone predictor. The degree ratio between documents and tags in user-induced graphs, and the number of connected components in tag-induced graphs seem to be the most relevant single predictors.

K-cores K-Cores only turned out useful for tags on the probe and the test set, while using them on other portions of the training set (not shown here) yielded better results; it seems that the "younger" members of the graphs (which we predict here) are not connected strongly enough to show up in expressive cores.

Other features The ratio of tags per user, for documents, turns out to be a useful measure, as motivated by the graphs shown in Figure 4. Also, the simple number of entries per user provides a tendency towards spaminess.

4.2 Classifying Feature Vectors

Training a classifier With a given set of features for the probe dataset, we trained a Support Vector Machine using SVMLight[6] using a five-fold cross-validation. Across various situations, we found a polynomial kernel of degree 6 with a balancing factor of 0.077 (the fraction of non-spammers in the training dataset) to be best. We train our classifier on the normalized features generated for the probe set and use it to predict the corresponding feature vectors of the test set.



(a) ROC curves on test set

(b) AUC values for cross-validation on the probe set, prediction on the test dataset, and cross-validation on the test dataset

Fig. 8. Final performance values

Results See Figure 8 for the final results of our predictors. We obtained an AUC of 0.913 on the test set with our submission run, using basically all the features introduced earlier (see column "S" in Table 1). Unable to resist the temptations of hindsight, we tried several other configurations of features, finding that leaving out network features entirely and using the unsmoothed URL prediction (see column "O") yielded the best result of 0.961. We also examined the performance of network-based features alone (column "N"), yielding an AUC of 0.854. Finally, we added to the overview the performance of using, without an additional classifier, the product of the tag spaminess and the iterated document spaminess (0.929).

5 Conclusions & Outlook

The most striking result of the presented experiments is that, at the end of the day, our constructed classifier performs worse than a simple product of two of the features used, tag and iterated document spaminess. What happened? If we regard Figure 8(b), we may get an idea: The spaminess features perform a lot worse (almost .1 AUC) on the probe set than on the test set. The trained classifier weighs the features accordingly and is thus not able to benefit from the improved spaminess features in the test set. The quality of the network measures remains stable from probe to test set, and so does the submitted classifier. Leaving out network properties during training (classifier "optimal") forces the classifier to weigh cooccurrence features more strongly, and thus the increased quality of those features is used. Performing a cross-validation on the test set (last block), both classifiers correctly identify the strength of the of the spaminess and perform about equally; whatever difference remains is due to the weakening smoothing used for the URL classifier. A conclusion of these results is that our creation of a probe dataset was faulty. As Figure 7(b) shows, cooccurrence feature performance deteriorates as we go from predicting the oldest (1) to the latest (5) users. We chose the last fifth of the users as our probe dataset, assuming that predicting the test set would be most similar to predicting the latest users

in the training set. This is obviously not the case,– the question why it isn't remains open for now and should be further explored.
We do find the network features promising, as they produce insight into the structural properties of spamming behaviours. In contrast to cooccurrence or URL features, no labeled users are needed for their construction. This could prove valuable when examining new datasets for which no labels have been created yet.

6 Acknowledgments

The first author is funded through a scholarship of the Integrated Graduate Program "Human-Centric Communication" and partially supported via the EU NoE P2P Tagged Media (PeTaMedia). We thank Matei Leventer, funded via DFG grant no. OB 102/10-2 (Learning Agents for Text Classification) for implementing the URL-based predictor, and Frank Schumann for invaluable feedback on earlier versions of this paper.

References

1. J. I. Alvarez-Hamelin, L. Dall'Asta, A. Barrat, and A. Vespignani. k-core decomposition: a tool for the analysis of large scale internet graphs, 2005.
2. C. Cattuto, C. Schmitz, A. Baldassarri, V. D. P. Servedio, V. Loreto, A. Hotho, M. Grahl, and G. Stumme. Network properties of folksonomies. *AI Communications Journal, Special Issue on "Network Analysis in Natural Sciences and Engineering"*, 20(4):245–262, 2007.
3. Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 2008. To appear.
4. P. Heymann, G. Koutrika, and H. Garcia-Molina. Fighting spam on social web sites: A survey of approaches and future challenges. *IEEE Internet Computing*, 11(6):36–45, 2007.
5. A. Hotho, R. Jäschke, C. Schmitz, and G. Stumme. Information retrieval in folksonomies: Search and ranking. In *Proceedings of the 3rd European Semantic Web Conference*, volume 4011 of *LNCS*, pages 411–426, Budva, Montenegro, June 2006. Springer.
6. T. Joachims. Making large-scale support vector machine learning practical. In *Advances in Kernel Methods: Support Vector Machines.*
7. G. Koutrika, F. Adjie Effendi, Z. Gyöngyi, P. Heymann, and H. Garcia-Molina. Combating spam in tagging systems. In *AIRWeb '07: Proceedings of the 3rd international workshop on Adversarial information retrieval on the web*, pages 57–64, New York, NY, USA, 2007. ACM.
8. B. Krause, A. Hotho, and G. Stumme. The anti-social tagger - detecting spam in social bookmarking systems. In *Proc. of the Fourth International Workshop on Adversarial Information Retrieval on the Web*, 2008.
9. S. B. Seidman. Network structure and minimum degree. *Social Networks*, 5:269–287, 1983.

Multilabel Text Classification for Automated Tag Suggestion

Ioannis Katakis, Grigorios Tsoumakas, and Ioannis Vlahavas

Department of Informatics,
Aristotle University of Thessaloniki,
54124 Thessaloniki, Greece
{katak,greg,vlahavas}@csd.auth.gr

Abstract. The increased popularity of tagging during the last few years can be mainly attributed to its embracing by most of the recently thriving user-centric content publishing and management Web 2.0 applications. However, tagging systems have some limitations that have led researchers to develop methods that assist users in the tagging process, by automatically suggesting an appropriate set of tags. We have tried to model the automated tag suggestion problem as a multilabel text classification task in order to participate in the ECML/PKDD 2008 Discovery Challenge.

1 Introduction

Tagging can be defined as the process of assigning short textual descriptions or key-words (called tags) to information objects. It is a simple approach to information organization that was regularly practiced over the last decades. Scientific publications for example, are often accompanied by a list of keywords that are either freely entered or selected from an ontology (e.g. ACM Computing Classification) by their authors.

The increased popularity of tagging during the last few years can be mainly attributed to its embracing by most of the recently thriving user-centric content publishing and management Web applications (also known as Web 2.0 applications), such as wikis, web logs (blogs), and resource sharing systems, as one of the main means for the organization of their content.

Within most of these Web 2.0 applications, tagging is characterized by an additional social dimension, as the tagging process involves multiple users attaching freely selected tags to shared content (collaborative tagging).

The simplicity and popularity of collaborative tagging as an information organization approach comes at the expense of several limitations [1]. Firstly, people choose tags based on their personal opinions, their knowledge background and their preferences. Furthermore, users may be describing the same object based on different granularity. This creates a noisy tag space and thus makes it harder to find material tagged by other users. Secondly, people may use polysemous words (a word that has many related senses) in order to tag the web resources. The lack of semantic distinction in tags can lead to inappropriate connections between items. Another problem is that different tags, which are either synonymous or have closely related meaning increase data redundancy, leading to reduced recall of information. Last, but not least, people tend to assign a very small number of tags to an object.

All these limitations have led researchers to develop methods that assist users in the tagging process, by automatically suggesting an appropriate rich set of tags, in order to avoid the aforementioned obstacles. Related work in the field involve collaborative filtering [2], graph based [2] and text mining [3, 4] approaches. In this paper we view this problem from a different perspective, modeling it as a multilabel text classification task.

The rest of the paper is structured as follows. In the next section we provide some background information on the problem of multilabel classification. After that, we briefly describe the task of the discovery challenge that we have participated in. In Section 4 we present the datasets and comment on some of their main characteristics. In Section 5 we describe the proposed recommender that we evaluate in Section 6. Finally, Section 7 concludes this work.

2 Multilabel Classification

Traditional *single-label* classification is concerned with learning from a set of examples that are associated with a single label λ from a set of disjoint labels L, $|L| > 1$. If $|L| = 2$, then the learning task is called *binary* classification (or *filtering* in the case of textual and web data), while if $|L| > 2$, then it is called *multi-class* classification. In *multilabel* classification, the examples are associated with a set of labels $Y \subseteq L$.

Multilabel classification is a challenging research problem that emerges in several modern applications such as music categorization [5, 6], protein function classification [7–10] and semantic classification of images [11, 12]. In the past, multilabel classification has mainly engaged the attention of researchers working on text categorization [13–15], as each member of a document collection usually belongs to more than one semantic category.

Multilabel classification methods can be categorized into two different groups [16]: i) *problem transformation* methods, and ii) *algorithm adaptation* methods. The first group of methods are algorithm independent. They transform the multilabel classification task into one or more single-label classification, regression or label ranking tasks. The second group of methods extend specific learning algorithms in order to handle multilabel data directly.

The most widely-used problem transformation method, called Binary Relevance (BR Learning), considers the prediction of each label as an independent binary classification task. It learns one binary classifier $h_\lambda : X \to \{\neg\lambda, \lambda\}$ for each different label $\lambda \in L$. It transforms the original data set into $|L|$ data sets D_λ that contain all examples of the original data set, labeled as λ if the labels of the original example contained λ and as $\neg\lambda$ otherwise. It is the same solution used in order to deal with a multi-class problem using a binary classifier, commonly referred to as one-against-all or one-versus-rest.

3 Task Description

We have participated in the second task "Tag Recommendation in Social Bookmark Systems". Bibsonomy[1] is a social bookmarking and publication-sharing system. A user may store and organize Bookmarks (web pages) and BibTeX entries. The main tool provided for content management in BibSonomy is tagging. Users can freely assign tags to Bookmark or BibTeX items when they submit them to the system. This task of the competition requires the development of recommender system for BibSonomy. The recommender should efficiently propose a relevant set of tags to the user when he/she submits a new item (Bookmark or BibTeX) into the system. The organizers of the competition made available a training set including examples of users assigning tags to Bookmark and BibTeX items. A new, unseen, test set will be provided in order to evaluate candidate recommenders. The decisions of each system will be compared with the true tags and the average f-measure will be calculated.

Let D be an evaluation data set, consisting of $|D|$ examples (x_i, Y_i), $i = 1..|D|, Y_i \subseteq L$. Let h be a recommender and $Z_i = h(x_i)$ be the set of labels predicted by h for example x_i. The Precision, Recall and F-measure for the recommender h on test dataset D is calculated as follows.

$$Precision(h, D) = \frac{1}{|D|} \sum_{i=1}^{|D|} \frac{|Y_i \cap Z_i|}{|Z_i|} \qquad Recall(h, D) = \frac{1}{|D|} \sum_{i=1}^{|D|} \frac{|Y_i \cap Z_i|}{|Y_i|}$$

$$F(h, D) = \frac{1}{|D|} \sum_{i=1}^{|D|} \frac{2|Y_i \cap Z_i|}{|Z_i| + |Y_i|}$$

4 Data Analysis and Preprocessing

Three training files were provided for the tag recommendation task namely `tas`, `bookmark` and `bibtex`.

- `tas` file: contains the tags that a particular user has assigned to a particular item.
- `bookmark` file: contains metadata for bookmark items like the URL of the web page, a description of the web page, etc.
- `bibtex` file: contains metadata for the bibtex items like the title of the paper, the authors, etc.

In Table 1 you can see the attributes of all three training files.

[1] BibSonomy - http://www.bibsonomy.org

Table 1. Attributes of the three files

File	Attributes
tas	`user, tag, content id, content type, date`
bookmark	`content id, url hash, url, description, extended description, date`
bibtex	`content id, journal volume, chapter, edition, month, day, booktitle, howPublished, institution, organization, publisher, address, school, series, bibtexKey, url, type, description, annote, note, pages, bKey, number, crossref, misc, bibtexAbstract, simhash0, simhash1, simhash2, entrytype, title, author, editor year`

Note that in `bookmark` and `bibtex` files the same resource (i.e. web page or BibTeX entry) may appear several times, one for every user submitted the web page or BibTeX item. Different users might add different meta-data and, of course, different tags into a resource. A BibteX item is identified by its unique `simhash1` attribute and a Bookmark item by its unique `url_hash` attribute. The `content_id` field links the three tables and is unique for a <user,resource> pair.

In order to evaluate the proposed approach we have divided the available files into train and test. We have kept the 80% of the `tas` file for training and the rest for testing. The corresponding `bookmark` and `bibtex` train and test files were created based on the `tas` file using the `content_id` identifier.

Some interesting statistics that we obtained from the data and exploited in our method are presented below:

- In the initial `tas` file there are 816197 records, corresponding to single tags assigned by a specific user into a resource.
- There are 268692 posts in the `tas` file (tag-*set* assignments from a particular user to a specific resource).
- There are 176141 bookmark posts.
- 156054 unique bookmark resources (web pages) in the `bookmark` file distinguished by the `url_hash` attribute.
- There are 92544 bibtex posts.
- 71704 unique bibtex items in the `bibtex` file distinguished by the `simhash1` attribute.
- Only 18192 of the above bibtex items contained abstract descriptions.

After we split the original data into training and test files the following statistics were calculated.

- Only 8.55% of the bookmark items in the test set also exist in the training set.
- Only 9.77% of the bibtex items in the test set also exist in the training set.
- 65.69% of the bookmark users in the test set also exist in the training set.
- 21.89% of the bibtex users in the test set also exist in the training set.
- The average number of tags assigned by a user to a single bookmark item in the test set is 2.76.

– The average number of tags assigned by a user to a single bibtex item in the test set is 3.25.

5 Proposed Recommender

Recommendations are required for every S_i <user,item> pair in the `TestTasFile`. In other words, we want to predict what tags a particular user would assign to this particular item. Therefore, it is important to note that the recommendations should be personalized. Another important observation that arises from the statistics mentioned in the previous section, is that items will probably not appear in the test set but there is an important possibility that the users may reappear. Hence, the tag recommender should be able to exploit prior knowledge about the item or the user but simultaneously be able to make recommendation for unseen users and items. We tried to fulfill these requirements with our tag recommender.

Our recommender works as follows (see Figure 1). The system checks if the item (Bookmark or Bibtex) exists in the training set. If this is the case then the (N) most popular tags for the item are suggested. If the item appears for the first time then the system examines if the user has appeared before. If the user is found, then the most popular tags for the user is the output of the recommender. If neither the item nor the user have appeared before then the multilabel text classifier is called to assign a relevant set of tags.

The classifier is taking into consideration the text representation of the item. This can be the content and the title of the web page or the title and the abstract of the bibtex item. The classifier as implemented in our framework takes three parameters (see Figure 2) in order to classify an item. The first parameter and main input is the text representation of the object. For the bookmark items we obtained the `description`, `extended description` and content of the web page. For the bibtex items, we kept the `journal`, `booktitle`, `bitexAbstract` and `title` attributes. The second parameter is the maximum number of recommendations (M) that the classifier will produce. However, the third parameter (θ) will force the classifier to only recommend labels (tags) that is confident enough.

We have used the Binary Relevance (BR) classifier from the Mulan[2] package. We have selected the BR classifier basically because it is a simple classifier that scales linearly with the number of classes in a multilabel classification dataset. The base learner used with BR was a naive Bayes classifier. We have set up one classifier for the Bookmark items and one for the Bibtex Items.

In order to train the classifiers we had to convert the original data into ARFF (Weka [17]) format. However, in order to decrease the dimensionality of the problem, we kept only words with a minimum frequency $f_{w(min)}$ and tags with minimum frequency of appearence $f_{t(min)}$. Therefore, in order to produce datasets for the classifiers of reasonable sizes we have set $f^1_{w(min)} = 3000$ and

[2] Mulan - **Mu**lti **La**bel Classificatio**n**, (http://mlkd.csd.auth.gr/multilabel.html)

```
Data: The training set including a TasTrainFile, a BookTrainFile and a BibTrainFile.
Input: The post $S_i(< user, item >)$ pair from the TasTestFile
Output: The prediction $P = \{t_1, t_2, \ldots, t_n\}$ of the system, $t_i \epsilon T$, where $T$ is the set of all available tags
initialize $N_1$,$N_2$;
initialize $\theta_1$,$\theta_2$;
initialize $M_1$,$M_2$;
for All $S_i$ in TestTASFile do
    if $S_i$.item is Bookmark then
        if $S_i$.item appears in BookmarkTrainFile then
            $P \leftarrow N_1$ most popular tags for $S_i$.item;
            if $P = \emptyset$ then
                $P \leftarrow$ bookClassifier($S_i$.item.getText(),$\theta_1$,$M_1$);
        else
            if $S_i$.user appears in TasTrainFile then
                $P \leftarrow N_1$ most popular tags for $S_i$.user;
                if $P = \emptyset$ then
                    $P \leftarrow$ bookClassifier($S_i$.item.getText(),$\theta_1$,$M_1$);
            else
                $P \leftarrow$ bookClassifier($S_i$.item.getText(),$\theta_1$,$M_1$);
                if $P = \emptyset$ then
                    $P \leftarrow N_1$ most popular tags in BookmarkTrainFile
    if $S_i$.item is Bibtex then
        if $S_i$.item appears in BibtexTrainFile then
            $P \leftarrow N_2$ most popular tags for $S_i$.item;
            if $P = \emptyset$ then
                $P \leftarrow$ bibClassifier($S_i$.item.getText(),$\theta_2$,$M_2$);
        else
            if $S_i$.user appears in TasTrainFile then
                $P \leftarrow N_2$ most popular tags for $S_i$.user;
                if $P = \emptyset$ then
                    $P \leftarrow$ bibClassifier($S_i$.item.getText(),$\theta_2$,$M_2$);
            else
                $P \leftarrow$ bibClassifier($S_i$.item.getText(),$\theta_2$,$M_2$);
                if $P = \emptyset$ then
                    $P \leftarrow N_2$ most popular tags in BibtexTrainFile
```

Fig. 1. Pseudocode of the proposed tag recommender

```
Input: $S_i$: item to be classified, M: number of max recommendations, $\theta$:
       confidence threshold
Output: The prediction $P = \{t_1, t_2, \ldots, t_n\}$ of the system, $t_i \epsilon T$, where $T$ is the
        set of all available tags
$P \leftarrow \emptyset$;
C $\leftarrow$ Classifier.getConfidences(T, $S_i$.item.getText());
R $\leftarrow$ rank C in descending order;
for $(i = 0;\ i < M;\ i{+}{+})$ do
    if $R_i > \theta$ then
        $P = P \quad R_i$
return $P$
```

Fig. 2. Multilabel text classification at the proposed recommender

$f^1_{t(min)} = 300$ for the bookmark file and $f^2_{w(min)} = 100$ and $f^2_{t(min)} = 50$ for the bibtex file. These setting led to a bookmark arff file of 208 tags and 2150 words and a bibtex file of 159 tags and 1836. Both datasets are available on-line at: http://mlkd.csd.auth.gr/multilabel.html.

6 Evaluation

We used the f-measure as discussed in section 3 in order to evaluate the framework and tune the parameters. Although we have tried various alternative settings, we have not conducted an exhaustive study for parameter settings. Some of the results obtained are presented in Table 2.

Table 2. F-measure values obtained for various parameter settings.

Parameters			**F-measure**		
θ	**M**	**N**	**All**	**Book**	**Bib**
0.0	10	10	0.0716	0.0782	0.0633
0.0	5	5	0.0848	0.0940	0.0736
0.0	1	1	0.0700	0.0904	0.0453
0.9	10	10	0.0713	0.0752	0.066
0.9	3	3	0.0847	0.0940	0.0734
0.9	10	3	**0.0852**	**0.0942**	**0.0740**

We observe that the best overall results are achieved when $\theta = 0.9$, $M = 10$, $N = 3$ [3]. Note that this is a setting providing 3 recommendations which is close to the average number of tags assigned by the users, as observed in

[3] In order to simplify the selection of parameter values we set $\theta = \theta_1 = \theta_2$, $M = M_1 = M_2$ and $N = N_1 = N_2$.

section 4. There was a slight improvement to these results when we used the classifier to make predictions when the most popular tag set was empty (see Figure 1), for example because of the removal of some tags from the training set. A further small improvement was achieved when we used the most popular tags in bookmarks and bibtex respectively when the classifier predictions where empty. The final f-measures achieved were 0.0856, 0.0942, 0.0751 respectively.

7 Conclusions

We have tried to utilize a multilabel classification algorithm in order to build an automated tag recommender for bibsonomy. Results show that tag recommendation is indeed a challenging and interesting problem for the data mining and machine learning community. Having more time we would like to test more multilabel classification algorithms and apply multilabel feature selection.

Acknowledgements

This work was partially supported by a PENED program (EPAN M.8.3.1, No. 03Δ73), jointly funded by the European Union and the Greek Government (General Secretariat of Research and Technology/GSRT).

References

1. Marchetti, A., Tesconi, M., Ronzano, F.: Semkey: A semantic collaborative tagging system. (2007)
2. Jäschke, R., Marinho, L.B., Hotho, A., Schmidt-Thieme, L., Stumme, G.: Tag recommendations in folksonomies. In Kok, J.N., Koronacki, J., de Ma'ntaras, R.L., Matwin, S., Mladenic, D., Skowron, A., eds.: Knowledge Discovery in Databases: PKDD 2007, 11th European Conference on Principles and Practice of Knowledge Discovery in Databases. Volume 4702 of Lecture Notes in Computer Science., Berlin, Heidelberg, Springer (2007) 506–514
3. Chirita, P.A., Costache, S., Nejdl, W., Handschuh, S.: P-tag: large scale automatic generation of personalized annotation tags for the web. In: WWW '07: Proceedings of the 16th international conference on World Wide Web, New York, NY, USA, ACM (2007) 845–854
4. Sood, S., Hammond, K., Owsley, S., Birnbaum, L.: TagAssist: Automatic Tag Suggestion for Blog Posts. In: Proceedings of the International Conference on Weblogs and Social Media (ICWSM 2007). (2007)
5. Li, T., Ogihara, M.: Detecting emotion in music. In: Proceedings of the International Symposium on Music Information Retrieval, Washington D.C., USA (2003) 239–240
6. Trohidis, K., Tsoumakas, G., Kalliris, G., Vlahavas, I.: Multilabel classification of music into emotions. In: Proceedings of the 9th International Conference on Music Information Retrieval (ISMIR). (2008)
7. Clare, A., King, R.: Knowledge discovery in multi-label phenotype data. In: Proceedings of the 5th European Conference on Principles of Data Mining and Knowledge Discovery (PKDD 2001), Freiburg, Germany (2001) 42–53

8. Diplaris, S., Tsoumakas, G., Mitkas, P., Vlahavas, I.: Protein classification with multiple algorithms. In: Proceedings of the 10th Panhellenic Conference on Informatics (PCI 2005), Volos, Greece (November 2005) 448–456
9. Roth, V., Fischer, B.: Improved functional prediction of proteins by learning kernel combinations in multilabel settings. In: Proceeding of 2006 Workshop on Probabilistic Modeling and Machine Learning in Structural and Systems Biology (PMSB 2006), Tuusula, Finland (2006)
10. Zhang, M.L., Zhou, Z.H.: Multi-label neural networks with applications to functional genomics and text categorization. IEEE Transactions on Knowledge and Data Engineering **18**(10) (2006) 1338–1351
11. Boutell, M., Luo, J., Shen, X., Brown, C.: Learning multi-label scene classification. Pattern Recognition **37**(9) (2004) 1757–1771
12. Kang, F., Jin, R., Sukthankar, R.: Correlated label propagation with application to multi-label learning. In: CVPR '06: Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, New York City, NY, USA, IEEE Computer Society (2006) 1719–1726
13. Yang, Y.: An evaluation of statistical approaches to text categorization. Journal of Information Retrieval **1** (1999) 67–88
14. McCallum, A.: Multi-label text classification with a mixture model trained by em. In: Proceedings of the AAAI' 99 Workshop on Text Learning. (1999)
15. Schapire, R.E. Singer, Y.: Boostexter: a boosting-based system for text categorization. Machine Learning **39**(2/3) (2000) 135–168
16. Tsoumakas, G., Katakis, I.: Multi-label classification: An overview. International Journal of Data Warehousing and Mining **3**(3) (2007) 1–13
17. Witten, I.H., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques, Second Edition (Morgan Kaufmann Series in Data Management Systems). Morgan Kaufmann (June 2005)

Tag Recommendation for Folksonomies Oriented towards Individual Users

Marek Lipczak

Faculty of Computer Science, Dalhousie University, Halifax, Canada, B3H 1W5
lipczak@cs.dal.ca

Abstract. Tagging has become a standard way of organizing information on the Web, particularly in folksonomies – data repositories freely created by communities of users. A few tags attached to each resource create a bridge between heterogeneous data and users accustomed to keyword-based search and browsing. To establish this connection, tagging requires users to manually define tags for each resource they enter to the system. This potentially time-consuming step can be eased by tag recommender systems, which propose terms that users may choose to use as tags. This paper suggests and evaluates potential sources of recommended tags, focusing on folksonomies oriented towards individual users. These suggestions are used to propose a three-step tag recommendation system. Basic tags are extracted from the resource title. In the next step, the set of potential recommendations is extended by related tags proposed by a lexicon based on co-occurrences of tags within resource's posts. Finally, tags are filtered by the user's personomy – a set of tags previously used by the user.

1 Introduction

Folksonomy services allow users to store and share various types of Internet resources. The content of folksonomies is completely defined by communities of their users. Large number of creators and resources push the folksonomies from the traditional hierarchical data structure design based on directories created by system editors (e.g., Open Directory Project[1]) to tag-based taxonomies defined jointly by service users (e.g., BibSonomy[2], del.icio.us[3], Flickr[4], Technorati[5]).While adding a resource to the system, users are asked to define a set of tags – keywords which describe it and relate it to other resources gathered in the system. To ease this process, some folksonomy services recommend a set of potentially matching tags. Proposing a tag recommendation system was a task of ECML PKDD discovery challenge 2008[6]. This paper presents a tag recommendation system submitted to the challenge.

[1] http://www.dmoz.org/about.html

[2] http://bibsonomy.org/help/about/

[3] http://del.icio.us/about/

[4] http://flickr.com/about/

[5] http://technorati.com/about/

[6] http://www.kde.cs.uni-kassel.de/ws/rsdc08/

The formal definition of folksonomy can be found in [6]. A folksonomy is a collection of resources entered by users in posts. Each *post* consists of a resource and a set of tags attached to it by a user. Generally, the resource is specific to the user who added it to the system. However, for some types of resources (e.g., bookmarks) identical resources can be added to the system by different users. In the latter case, by the *set of resource tags* we denote all tags attached to a given resource by various users.

Folksonomies can be classified into two types based on the objective of the tagging process. The first type, represented by BibSonomy and del.icio.us, treats resources (e.g., personal bookmarks) as an individual property of a user. Here, the aim of tags is to create a repository tailored to individual user interests. In this paper, this type is referred to as *folksonomies oriented towards individual users.* The second type of folksonomies, represented by Flickr and Technorati, is a shared repository of public resources (e.g., blog entries). In this case tags are added keeping in mind a broad audience that in the future would like to search for the resource. In this paper, this type is referred to as *folksonomies oriented towards broad audience.* As the reason of tagging a resource is fundamentally different, we may expect that a tag recommendation system that suits one folksonomy type would be inappropriate for the other. This paper focuses on the first type, proposing a tag recommender for individual users.

2 Related work

The attention of researchers is mostly directed to tag recommendation systems for broad audience folksonomies. TagAssist [12] is a system designed to recommend tags of blog posts. The recommendation is built on tags previously attached to similar resources. Earlier, meaning disambiguation is performed based on co-occurrence of tags in the complete repository. Co-occurrence of tags was also used by Sigurbjörnsson and van Zwol [11] to propose tags that complement user-defined tags of photographs in Flickr.

The problem of tag recommendation in folksonomies oriented towards individual users was addressed by Jäschke et al. [7]. They compared a number of recommendation techniques including collaborative filtering, PageRank, and its modification suited for folksonomies – FolkRank. The evaluation showed that the FolkRank based recommender outperforms other approaches; however, the tests were performed on a dense core of folksonomy, thus might be not representative.

Most of the tag recommendation systems are based on the tags that are already present in the system. An exception from this rule is the system presented by Lee and Chun [9]. The system recommends tags retrieved from the content of a blog, using artificial neural network. The network is trained based on statistical information about word frequencies and lexical information about word semantics extracted from WordNet.

Schmitz et al. [10] proposed association rule mining as a technique that might be useful in the tag recommendation process. The intuition behind this concept was also used in the system presented by this paper.

3 Examined dataset

All presented experiments and the evaluation of proposed tag recommendation system were performed on a snapshot of BibSonomy [5] containing $2,570$ users, $242,175$ resources and $274,139$ posts (after preprocessing). The snapshot was provided by the organizers of the ECML PKDD discovery challenge 2008. The preprocessing phase included removing useless tags (e.g., "system:unfiled"), changing all letters to lower case and removing non-alphabetical and non-numerical characters from tags.

The statistical characteristics of folksonomies have been an object of many research publications [2, 3, 8, 11]. In the following sections I present experiments particularly important from the perspective of the tag recommendation task.

3.1 General characteristics

The frequency distribution of tags from the Bibsonomy snapshot shows that mid- and low-frequency tags follow Zipf's distribution (Fig. 1). Zipf's distribution does not hold for high-frequency tags. The frequency distribution of tags from Flickr, which represents folksonomies oriented towards broad audience shows important differences [11]. Flickr's low-frequency tags does not follow Zipf's distribution. A possible explanation of this fact is a smaller number of user specific tags in comparison to folksonomies oriented towards individual users. In addition, Flickr's high-frequency tags follows Zipf's distribution and are too general to be used as recommendation. The list of the most frequent tags from Bibsonomy ("software", "web20", "tools", "web", "blog") shows that tag recommenders for folksonomies oriented towards individual users should not ignore high-frequency terms.

The difference between two folksonomy types may have impact on the efficiency of applied tag recommendation methods. A commonly used collaborative filtering approach is based on the intuition that the best recommendation consists of tags attached to the resource by people similar to the user. This approach proved its quality in many recommendation systems; however, the intuition behind it can be deceiving. Folksonomies like BibSonomy or del.icio.us are mainly designed as a collection of repositories of individual users. By adding posts, each user defines his/her own set of used tags – personomy [6], which describes the resources from a user's point of view. As a result, users addressing similar resources do not have to use similar tags, and similar personomies do not have to be associated with similarity in tagged resources. In fact, there is no such correlation in the processed BibSonomy snapshot. The cosine similarity between users calculated based on tags seems to be uncorrelated with that calculated based on resources (Fig. 2). In this situation recommending tags assigned to a resource by similar users (collaborative filtering) should give similar results as recommending the tags frequently attached to the resource by any user. This conclusion seems to be confirmed by the experiment presented by Jäschke et al. [7]. Minding the limitations of the collaborative approach I decided to focus on a tag space that is directly related to a post.

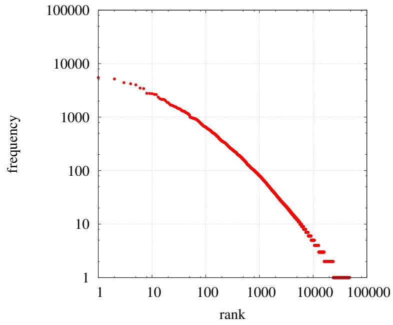


Fig. 1. The overall frequency distribution of tags (after preprocessing and removing posts classified as imported).

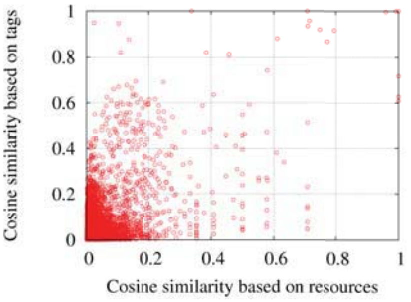


Fig. 2. Cosine similarity between each pair of users calculated based on tags (tf-idf weights) and resources (binary weights). The two values seem to be independent.

3.2 Characteristics based on individual posts

Considering only the direct surrounding of the post, the potential tag recommendations can be obtained from the resource itself, the set of tags attached to the resource in previous posts, or the set of tags that were already used by the user (user's personomy). Exploiting tags from the resource depends on the folksonomy character. In BibSonomy the resource can be a bibtex entry or a web-page bookmark. The first contains bibliographic information about a research publication including its title and abstract. The second contains web-page title and URL. Preliminary experiments showed that using title words as tags outperforms the results of abstracts and URLs. The latter two contain lesser amount of correct tags. The title is the only element that joins both resource types and it is common in other folksonomies, which are its additional advantages. I decided to use the title as the representation of resource.

To evaluate the three potential sources of tag recommendations, namely words from the resource title, resource tags and user's personomy, I checked for each post if its tags can be found in any of these sources associated with all other posts in the folksonomy. The quality of sources was measured by precision (i.e., number of correct tags retrieved divided by the total number of retrieved tags) and recall (i.e., number of correct tags retrieved divided by the total number of correct tags). These are standard information retrieval metrics [4]. The value of recall was averaged over all tested posts. The averaged recall informs us how many correct tags can be found in a source. The value of precision was averaged only over posts, for which the source returned any tags. Precision averaged this way is the ratio of correct tags among all tags retrieved. In addition, I present the total number of potential tags obtained from the sources, and the number of correct tags among them (Fig. 3).

User's personomy is the richest source of correct tag recommendations. For the tested BibSonomy snapshot it gave access to 90% of tags from test posts. On

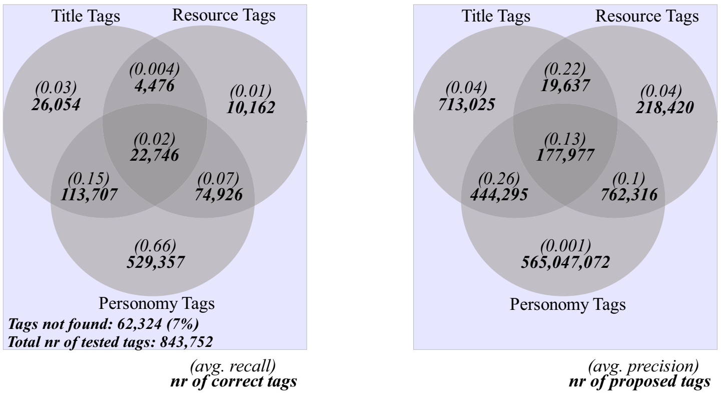


Fig. 3. Venn diagrams presenting average recall, plus the number of correct tags found in three potential sources of tags (left) and average precision, plus the total number of tags retrieved from these sources (right).

the other hand, correct tags from personomy are accompanied by a large number of incorrect tags (precision around 0.001). Compared to tags retrieved from personomy, the recommendation based on resource title is much more precise; however, the number of correct tags found this way is lower. In addition, most of these tags can be also found in the user's personomy. Finally, both recall and precision values show that resource tags are not a good source of potential tag recommendations. The character of each tag recommendation source and their potential usability in tag recommendation system are discussed in the following sections.

Resource title Resource title appears to be the most robust source of tag recommendations. Among all posts in processed BibSonomy snapshot only 51 resource titles were unable to produce any tags (no letters or numbers in the title). In addition, among all discussed sources the title seems to be the most strongly related to the resource. The drawback of this source is low recall which makes the title inappropriate as a stand-alone tag recommender. The title is a simplified natural language sentence, which should be cleaned of words with no informative value (e.g., stopwords).

Resource tags Tags assigned to the resource by other folksonomy users are not a good source of tag recommendations. One of the reasons is the sparsity of data; 92% of resources were added to the system only once. This fact significantly limits the possible recall of this source of tags. The other issue is the personal character of posts (discussed in section 3.1), which hurts the precision of retrieved tags. The variety of tags attached by users creates, however, another application of resource tag sets. Mining relations between tags attached to the same resource can result in a simplified semantic lexicon. The lexicon would not give us the information about the character of relation, but given a tag, the lexicon can point out related tags which are also potential recomendations. The lexicon consists of general relations between tags and can be used independently of the resources. This fact reduces the negative impact of data sparsity. In addition, it is suited for a particular folksonomy and it can capture specific relations between its tags.

Personomy tags Building his/her personomy the user is interested in representing his/her interests using a limited number of tags. The same tag will be attached to resources fitting a particular interest, for example, all articles related to user's master thesis will be tagged by the same keyword. In addition, users are likely to stick with one lexical form of a word or expression, for example using constantly singular or plural form of a noun (e.g., "*publication*" or "*publications*"). These are the reasons why we are likely to find a lot of good recommendations among user's tags. The problem is that the choice of the lexical form or the word that describes the interest is completely up to the user. For the given example of resources related to master thesis the tag may be "*masterthesis*", "*msc*", "*thesis*", "*work*", or any other that according to user's opinion conveys the information.

To describe the resource more accurately users pick additional tags, specific not only to the user, but also to the resources. This is likely the cause of a large number of low-frequency tags (see section 3.1) and complicates the process of retrieving potential recommendations from personomy.

4 Tag recommendation system

The tag recommendation system (Algorithm 1), described in this section, is based on observations from the presented statistical experiments. The system is built of three steps. The first step produces tags from resource title words and assigns a score that represents their usefulness for previously tagged resources. The second step uses the resource tag based lexicon to propose tags related to tags taken from the title. The third step checks the tags proposed by the lexicon against user's personomy. The tags recommended to the user are a union of most promising tags produced in step one and three. The following sections give the detailed description of each step.

Extraction of title based tags The resource title is divided into words, which are then cleaned of non-alphabetical and non-numerical characters. The system assigns a score to each word, which represents the probability of being chosen as a tag – number of times being chosen as a tag divided by the number of occurrences. If the word occurred in the titles of previously entered resources less than 100 times its probability of being a correct tag is set to 0.1 which is an empirically estimated value for low-frequency tags. The probability score is introduced to reduce the impact of stopwords. It is important to notice that the standard stopwords list, which is often used in information retrieval systems, is

Algorithm 1: Tag recommendation system

```
Data: a resource $p_{res}$ and user $u$
Result: a set of recommended tags $T_{Recommendation}$, a tag consist of a keyword
        (w) and recommendation score (score)
begin
    /*Step 1 – Extraction of title based tags*/
    $W_{Title} \longleftarrow extractTitleWords(p_{res})$
    $T_{Title} \longleftarrow \emptyset$
    foreach $w \in W_{Title}$ do
        $T_{Title}$ add $makeTag(w, getPriorUsefullness(w))$
    /*Step 2 – Retrieval of tags related to title*/
    foreach $t \in T_{Title}$ do
        $T_{t\ Related} \longleftarrow \emptyset$
        $T_{t\ RelTags} \longleftarrow \emptyset$// related tags from Tag-to-Tag lexicon
        $T_{t\ RelTitle} \longleftarrow \emptyset$// related tags from Title-to-Tag lexicon
        foreach $r \in getRelated(l_{TagToTag}, t)$ do
            $T_{tRelTags}$ add $makeTag(r.w, t.score * getRelScore(l_{TagToTag}, t, r))$
        foreach $r \in getRelated(l_{TitleToTag}, t)$ do
            $T_{t\ RelTitle}$ add $makeTag(r.w, t.score * getRelScore(l_{TitleToTag}, t, r))$
        $T_{t\ RelTags} \longleftarrow limitSize(T_{tRelTags}, 20)$
        $T_{t\ RelTitle} \longleftarrow limitSize(T_{tRelTitle}, 20)$
        $T_{t\ Related} \longleftarrow unionProb(T_{t\ RelTags}, T_{t\ RelTitle})$
    $T_{Related} \longleftarrow unionProb(T_{t_1\ Related}, \ldots, T_{t_n\ Related})$
    /*Step 3 – Personomy based filtering*/
    $P \longleftarrow getPersonomy(u)$
    foreach $t \in T_{Related}$ do
        $T_{t\ RelPersonomy} \longleftarrow \emptyset$// tags retrieved from user's personomy
        if $t \in P$ then
            foreach $r \in P$ do
                $T_{t\ RelPersonomy}$ add $makeTag(r.w, t.score * getRelScore(P, t, r))$
    $T_{RelPersonomy} \longleftarrow unionProb(T_{t_1\ RelPersonomy}, \ldots, T_{t_n\ RelPersonomy})$
    $T_{RelPersonomy} \longleftarrow normalizeScores(limitSize(T_{RelPersonomy}, 10))$
    $T_{Title} \longleftarrow normalizeScores(T_{Title})$
    $T_{Recommendation} \longleftarrow limitSize(unionProb(T_{Title}, T_{RelPersonomy}), 10)$
end
```

not sufficient here, because we have to deal with titles in various languages and stopwords specific for the folksonomy (e.g., word "*page*" is frequent in web-page titles, but it is rarely used as a tag).

Retrieval of tags related to title The most important element of this step is the definition of the lexicon. It can be built based on two types of relations. As introduced in section 3.2, the lexicon can be built based on tags attached to the same resource, which are considered as related. The calculation of the factor that represents the relation strength can be solved based on various approaches (e.g., association rule mining). In the presented system the score for a tag t_1 is the number of its co-occurrences with another tag t_2 among all resources, divided by the total number of occurrences of tag t_1. The score is analogous to the confidence score (Eq. 1) in association rule mining [1].

$$confidence(t_1,\ t_2) = \frac{support(\{t_1 \cap t_2\})}{support(\{t_1\})} \tag{1}$$

Considering title words as the source of tags we can think of the second type of the lexicon representing relations between the words extracted from resource title and resource tags. The method of construction is analogical to the previous lexicon, the only difference is that tag t_1 is drawn from the title not the resource tags.

Both lexicons present different perspective of tag relations and give silightly different results (Table 1). The latter approach seems to be more adequate to the input tags; however, it is biased for general words that are often used in the title. For this type of words the related tags given by the second lexicon are simply the most frequently used tags (Table 2). To avoid the need of disambiguation between words more appropriate for either of the lexicons I decided to join the list of related tags produced by both of them (limited to twenty tags). The scores of tags that were present in both lists are summed as they were probabilities of two independent events.

This step is performed independently for each tag extracted from the title. Based on the lexicon the list of related tags with scores defining the strength of relation is retrieved. Finally, the lists are joined. Scores of multiple occurrences of identical tags are summed as they were independent probabilistic events, where the probability is defined by the relation score. Tags related to a word that is not likely to become a tag (e.g., "*page*") are also not good candidates for recommendation. These are very general terms which are hard to connect with any concept. This is the reason why before joining the relation score is multiplied by the title tag score computed in the previous step.

	Tag-to-Tag lex.		Title-to-Tag lex.	
	occurrence: 317		occurrence: 204	
	Tag	Score	Tag	Score
1.	semantics	1.000	semantics	0.392
2.	semanticweb	0.306	semanticweb	0.348
3.	ontology	0.177	semantic	0.313
4.	semantic	0.167	folksonomy	0.215
5.	semweb	0.158	tagging	0.196

Table 1. Top 5 tags related to "**semantics**" according to two types of lexicon.

	Tag-to-Tag lex.		Title-to-Tag lex.	
	occurrence: 53		occurrence: 2439	
	Tag	Score	Tag	Score
1.	home	1.000	software	0.081
2.	page	0.113	tools	0.073
3.	software	0.094	computing	0.064
4.	server	0.075	java	0.059
5.	photos	0.056	opensource	0.051

Table 2. Top 5 tags related to "**home**" according to two types of lexicon.

Personomy based filtering The set of tags retrieved in the second step is likely to consist of a lot of correct recommendations. However, low precision caused by the size of the set, makes its usefulness low. The last processing step is used to filter the tags that are most likely to be chosen by a user. Checking the tags against the user's personomy allows the system to choose lexical forms preferred by user (e.g., "semantics" instead of "semantic"). In addition, the personomy gives access to user specific tags (e.g., "masterthesis"). The retrieval of related tags is done analogously to the lexicon based approach used in the second step. The strength of relation is calculated based on Eq. 1; however, now the set of resources is limited to user's own posts. It is important to notice that this approach gives access not only to tags that are explicitly found in the personomy, but also to tags that co-occurred with them in user's posts. Subsequently, the scores are multiplied by the relation score of the base tag, which was calculated in the second step. Again the scores are calculated for each base tag separately and then the lists of results are joined, summing scores of multiple occurrences of the same tag in probabilistic way. The list of tags proposed as a recommendation is limited to the ten tags with the highest score.

As mentioned in section 3.2, the objective of some tags is to describe the resource, not to relate it to user's interests. To give the user access to recommendation of such tags, the system recommends also the tags retrieved from the title in the first step. As scores defined in first and third step are not comparable, I decided to normalize the scores in both lists, to make the sum of scores in each list equal to one. After normalization the lists are joined, again using the probabilistic sum and limiting the final list to ten tags.

5 Evaluation

This section presents the results of the off-line system evaluation based on the available BibSonomy snapshot. The used evaluation approach assumed that all and only relevant tags were given by the user. Although this method simplifies the problem it is robust and objective. The used quality metrics were recall and precision, commonly used in recommender system evaluations [4].

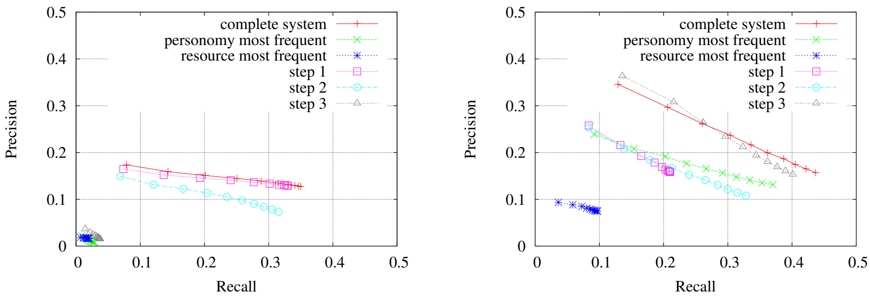
Methodology The commonly used evaluation approach is to keep strict division between training and testing set. This approach was used by the organizers of the ECML PKDD discovery challenge 2008. It allowed the organizers to keep the list of correct tags in secret during the contest. However, assuming that a user provides all and only relevant tags in a post, tag recommendation becomes a specific problem in which the complete feedback about the quality of recommendation is entered to the system with each post. In such case, we should consider incremental way of evaluation in which each tested post trains the system with tags provided by the user. The paper presents both evaluation approaches. The first experiment followed strictly the approach proposed by the organizers of the ECML PKDD discovery challenge 2008 – $59,542$ newest posts were used as test set. In the second experiment, in addition to incremental training, I decided to reduce the impact of posts imported from an external repository (e.g., web browser), by not testing the system on groups of user's posts with the same timestamp. This limited the number of test posts to $7,133$. Imported posts have their tags assigned automatically. In real use a tag recommender is not used for the imported tags, therefore it should not be tested by them.

To give more insights about the system its final recommendation is presented together with tags produced in each of its three steps. The first step, that simply proposes words from the title as tags, can be considered as a baseline system. Additional baseline systems presented are the recommenders that proposes most frequent tags from user's personomy and resource tags. As each approach returns ranked list of tags it is possible to freely limit the number of recommended tags. The plots (Fig. 4) present consecutive results for the top n tags, where $1 \leq n \leq 10$.

Results The first experiment shows low quality of personomy based recommendation, represented by the third step of the system and the baseline system which proposes the most frequent user's tags (Fig. 4(a)). This unexpected situation is caused by the evaluation approach used in this experiment (strict division between training and testing set). Among $59,542$ tested posts only $16,169$ (27%) were entered by users who have their previous posts in the training set. For the rest of tested posts the personomy based recommenders could not propose any tags. Clearly such large percentage of "first-time" users is not possible in reality, thus this evaluation approach seems to underestimate the score of personomy based recommenders. The results are also strongly biased by the choice of test posts. Especially not representative is a single user that is responsible for 65% of all test posts. His/Her posts are likely to be imported from an external repository. The tags in these posts look like being mechanically extracted from the article content, which supports the recall result of the title based recommender (first step of the system). The overall result of the system is therefore completely determined by the tags proposed in the first step, that was not meant to be the main element of the system.

The second evaluation approach, in which tested posts were used to train the system, solves the problem of extraordinarily large number of "first-time" users (3% of test posts). For this evaluation method personomy based recommender (the third step) outperforms title based solutions (Fig. 4(b)). Low and slowly decreasing (with increasing number of recommended tags) precision of the baseline approach shows that most frequent tags from personomy are not necessary a good recommendation. These results confirm previous experiments, which showed that personomy is the richest, but noisiest source of tags. The title also confirmed its usefullness as a source of tags. At some point increasing the number of recommended tags does not improve precision and recall of the first step – the number of words in the title hardly ever reaches 10. The results of the second step are consistent with the first step for the top tags. Tags tend to have high self-relation score which makes title base tags likely to be high in the ranking produced by the second step. The results of the last baseline system, the most frequent tags from the resource, confirmed that data sparsity greatly reduces the usefullness of recommenders based on resource tags.

Considering only the top tags, the third step of the system seems to be more precise than the overall system. It shows that the ranking method used to join the results of first and third step should be improved. The advantage of joining the results of these two steps is visible in the total recall of the overall system (0.44 compared to 0.4 for the third step).



(a) Evaluation approach proposed by the organizers of the ECML PKDD discovery challenge 2008.

(b) Training set is iteratively enriched by tested posts. Posts classified as imported are not tested.

Fig. 4. Recall and precision of tag recommendation system compared to results of its three steps, and two baseline systems: most frequent personomy and resource tags.

6 Conclusions and future work

The key conclusion of the presented experiments is that, when recommending tags in folksonomies oriented towards individual users we should not rely only on tags previously attached to given resource. Sparsity of data and individuality of users greatly reduce their usefulness. Looking for potential tags we should focus on the direct surrounding of the post, as in this type of folksonomies a collaborative filtering approach may be deceiving. The presented tag recommendation system tries to follow these directions starting with tags from the most robust source – resource title, and expanding them by the richest source – user's personomy.

The introduced three steps of tag recommender system can be used as a basis of more sophisticated approaches. The element that potentially is the most promising area of improvements is the folksonomy-based lexicon. In my future research I plan to experiment with mining the relations between tags using techniques from Data Mining (e.g., association rule mining) and Information Retrieval (e.g., PageRank algorithm). Specific characteristics of the dataset used in system evaluation decrease its reliability. To confirm the results I plan to repeat the experiments on different folksonomies (e.g., del.icio.us). However, large variance of results of two evaluation approaches points to the need for a unified evaluation method that is representative of the real applications of tag recommenders, before new experiments are made.

References

1. Rakesh Agrawal, Tomasz Imieliński, and Arun Swami. Mining association rules between sets of items in large databases. *SIGMOD Rec.*, 22(2):207–216, 1993.
2. Ralitsa Angelova, Marek Lipczak, Evangelos Milios, and Pawel Pralat. Characterizing a social bookmarking and tagging network. In *Proc. of the ECAI 2008 Workshop on Mining Social Data (MSoDa)*, pages 21–26, 2008.
3. Ciro Cattuto, Christoph Schmitz, Andrea Baldassarri, Vito D. P. Servedio, Vittorio Loreto, Andreas Hotho, Miranda Grahl, and Gerd Stumme. Network properties of folksonomies. *AI Commun.*, 20(4):245–262, 2007.
4. Jonathan L. Herlocker, Joseph A. Konstan, Loren G. Terveen, and John T. Riedl. Evaluating collaborative filtering recommender systems. *ACM Trans. Inf. Syst.*, 22(1):5–53, 2004.
5. Andreas Hotho, Robert Jäschke, Christoph Schmitz, and Gerd Stumme. BibSonomy: A social bookmark and publication sharing system. In *Proc. the First Conceptual Structures Tool Interoperability Workshop at the 14th Int. Conf. on Conceptual Structures*, pages 87–102, Aalborg, 2006. Aalborg Universitetsforlag.
6. Andreas Hotho, Robert Jäschke, Christoph Schmitz, and Gerd Stumme. Trend detection in folksonomies. In *Proc. First International Conference on Semantics And Digital Media Technology (SAMT)*, volume 4306 of *LNCS*, pages 56–70, Heidelberg, dec 2006. Springer.
7. Robert Jäschke, Leandro Balby Marinho, Andreas Hotho, Lars Schmidt-Thieme, and Gerd Stumme. Tag recommendations in folksonomies. In *Knowledge Discovery in Databases: PKDD 2007, 11th European Conference on Principles and Practice of Knowledge Discovery in Databases, Warsaw, Poland, September 17-21, 2007, Proceedings*, volume 4702 of *LNCS*, pages 506–514. Springer, 2007.
8. Beate Krause, Robert Jäschke, Andreas Hotho, and Gerd Stumme. Logsonomy - social information retrieval with logdata. In *HT '08: Proc. the 19th ACM conf. on Hypertext and hypermedia*, pages 157–166, New York, NY, USA, 2008. ACM.
9. Sigma On Kee Lee and Andy Hon Wai Chun. Automatic tag recommendation for the web 2.0 blogosphere using collaborative tagging and hybrid ANN semantic structures. In *ACOS'07: Proc. the 6th Conf. on WSEAS Int. Conf. on Applied Computer Science*, pages 88–93, Stevens Point, Wisconsin, USA, 2007. WSEAS.
10. Christoph Schmitz, Andreas Hotho, Robert Jäschke, and Gerd Stumme. Mining association rules in folksonomies. In *Data Science and Classification (Proc. IFCS 2006 Conference)*, pages 261–270, Ljubljana, July 2006. Springer.
11. Börkur Sigurbjörnsson and Roelof van Zwol. Flickr tag recommendation based on collective knowledge. In *WWW '08: Proc. the 17th international conference on World Wide Web*, pages 327–336, New York, NY, USA, 2008. ACM.
12. S.C. Sood, K.J. Hammond, S.H. Owsley, and L. Birnbaum. TagAssist: Automatic tag suggestion for blog posts. In *Proc. the International Conference on Weblogs and Social Media (ICWSM 2007)*, 2007.

RSDC'08: Tag Recommendations using Bookmark Content

Marta Tatu, Munirathnam Srikanth, and Thomas D'Silva

Lymba Corporation, Richardson, TX, 75080, USA
marta,srikanth,tsilva@lymba.com

Abstract. A variety of factors contribute to a tag being assigned by a user to a document that he or she bookmarked. Textual information present in a URL's title, a user's description of a document, or a bibtex field associated with a scientific publication are sources for automatically recommending tags relevant for a given bookmark. Lymba's submission for the RSDC'08 Tag Recommendation task uses document and user models derived from the textual content associated with URLs and publications by social bookmarking tool users. This paper describes our natural language understanding approach for producing tag recommendations and provides some initial results of our internal evaluation.

1 Introduction

Social bookmarking tools enable people to label content of interest with descriptions from their own vocabulary which facilitate easy recall. When shared in a collaborative setting, the bookmark labels enable users to discover new content and other users with similar or shared interests. While user interests and their vocabulary play a significant part in what is assigned as a label to a given document, the content of the document is the driver of the bookmarking event and the basis of the assigned labels and hence it provides automated systems important clues about the tags that can be assigned to a given document. Among other advantages, automatic tag recommendations help bookmarking systems manage the tag space and direct it towards the standardization of the labels that users assign to documents in order to describe them.

For the RSDC'08 Tag Recommendation task, we used the textual content associated with bookmarks to model documents (web pages and publications) and users based on their tagging and suggest tags for new bookmarks. A combination of statistical and semantic features are used to build document and user models. Section 2 presents the different "spaces" in which documents, bookmarks and users can be modeled and the process that generates these models. The different methods used to recommend tags for bookmarks are discussed in Section 3 followed by a description of the data processing that was performed on the RSDC'08 training and test data sets to build a representation of bookmarks, documents and users. The results of our experiments and our observations follow.

1.1 Previous Work

Automatic labeling of documents based on predefined categories has been explored in document classification systems. The labels or categories are modeled based on the terms extracted from content or document metadata. In social tag prediction, the set of labels is not fixed and the explicit association of users to documents adds additional dimensions to the classification or prediction problem. User's preferences for tags based on their interests, workflow, etc. play an essential part in the tags associated with content. Tags like *myown* are relevant to a particular user and do not add value in collaborative settings.

A number of approaches to social tag prediction have used information retrieval methods to identify similar documents and recommend their tags for a given document. AutoTag [1] suggests tags to weblogs based on the tags associated with other similar weblogs in a given collection. While this approach depends on the content of the document, no new tag (from the content) is suggested. Some approaches used the collaborative or social aspect of the data to recommend tags based on user similarity. Recently, Heymann et al. [2] use a large dataset of del.icio.us[1] links to evaluate the effectiveness of using text and inlinks information to model and predict tags for documents. A bag of words model with TFIDF weighting of terms is used to represent the features and solve tag prediction as a text classification problem. We use textual content associated with bookmarks to model users and documents and suggest tags not only from the existing tag space (generated from the training data), but also from the textual content associated with bookmarks.

2 Document and User Models

A bookmark b_i is identified by the triple $(u_i, d_i, \{t_{ij}\}_j)$ corresponding to "user u_i bookmarked document d_i by assigned it the set of tags $\{t_{ij}\}_j$". A bookmark b_i can be associated with its corresponding textual metadata. For instance, the metadata of a web document d_i bookmarked by a user u_i can include the actual URL, its title and any user-given description. For scientific publications, this metadata can include the title and author of the paper, the journal where the paper was published, etc. In addition to metadata information provided by users for their bookmarks, the textual content of bookmarked documents (d_i) can be exploited. For scientific publications, this is the textual content of the paper.

[1]

Our system uses the textual content associated with bookmarks and documents to model users and the documents they bookmark.

Given a bookmark that associates document d_0 with user u_0, the tag recommendation problem can be solved by estimating the likelihood $P(t_i|d_0, u_0)$ of a tag t_i being assigned to the bookmark (u_0, d_0). The recommended tags can exist in the tag space derived from the training data. However, concepts derived from the content associated with both d_0 and u_0 can also be suggested as tag recommendations. Joint modeling of documents and users in a common feature space provides the desired tag ranking for our recommendation system. For the RSDC'08 data, the textual content of the metadata provided for each (user,document) pair is used to represent each bookmark. Therefore, the representation of a document d_0 becomes the union of the representations of all its bookmarks $\{(u_i, d_0)\}_i$ (a document is modeled using the descriptions given to the document by all users that bookmarked the document). Similar combinations of bookmark representations provide the features needed to create user models.

For the RSDC'08 Discovery Challenge, we used and evaluated different feature representations depending on the data type and the adopted tag recommendation approach. A suite of natural language processing tools were used to understand the textual content associated with each bookmark (u_i, d_i). We extracted important concepts (nouns, adjectives and named entities) from the textual metadata associated with each bookmark and used semantic analysis to generate normalized versions of the concepts. The normalization process makes use of different lexico-semantic resources, e.g., WordNet[2] to stem the concepts and link synonyms. For instance, the concepts *European Union*, *EU*, and *European Community* (in all their case variations) are normalized to the same concept *european_union*. By representing bookmarks, documents and users in the *concept space* created by the concepts identified in all textual metadata as collections of normalized concepts – each associated with a corresponding weight, our tag recommendation system is able to suggest as tag recommendations concepts that do not belong to the existing tag space.

In order to derive a common feature space to compare documents, users, bookmarks and tags, we developed a conflation method for grouping tags into semantically related groups of tags referred to as *conflated tags*. This process of tag normalization takes care of spelling mistakes, abbreviations, joined concepts and provides a common representation for synonyms. For instance, *weblog* is one of the conflated tags derived for the RSDC'08 data. It conflates the following list of space-separated tags: *blog Blog weblog blogs Weblog weblogs blog, BLOGS Blogs Weblogs bloga bloging blogs, weblogs, Blog. we-*

[2] http://wordnet.princeton.edu

blogs_weblog_blogs_blog weblog_blog to_blog webology bl;ogs Blogs! ??blog blogr Blogs, BLOG Blog”> Blog: blogblogs. The set of conflated tags define a *tag space* that can be used to model bookmarks, documents and users. For this purpose, we use the mapping between normalized concepts and conflated tags provided by the underlying ontology (WordNet) to create a bookmark, document and user representation within the existing tag space.

We note that our mechanism for normalizing concepts and conflating tags was developed for the English language and has not been customized to handle multi-lingual data. This is reflected in our tagging results for non-English content and tags.

3 Recommending Tags for Bookmarks

Given the bookmark, document and user representations detailed in Section 2, let us consider the following notations: for a bookmark $b_i = (u_i, d_i, \{t_{ij}\}_j)$

- $TH(b_i)$ = the set of tag conflations derived for $\{t_{ij}\}_j$ – the tags assigned by user u_i to document d_i; $TH(b_i) \subseteq TagSpace$,
- $TC(b_i) = b_i$’s representation in the concept space (the set of normalized concepts evoked by b_i’s textual content); $TC(b_i) \subseteq ConceptSpace$, and
- $TT(b_i) = b_i$’s representation in the tag space (the set of conflated tags evoked by b_i’s concepts); $TT(b_i) = TC(b_i)$’s projection to the $TagSpace$; $TT(b_i) \subseteq TagSpace$.

For a document $d_0, TH(d_0) = \cup_i TH(b_i), TC(d_0) = \cup_i TC(b_i)$, and $TT(d_0) = \cup_i TT(b_i)$ where $b_i = (u_i, d_0, \{t_{ij}\}_j)$ is a bookmark provided in the training data. Similar definitions can be derived for a user u_0. We note that TC and TT are independent of the tags that user u_i assigns to document d_i. They are solely based on the textual content that the user associates with the document. Let us also denote $TT(b_i) \cup TH(b_i)$ by $THT(b_i)$ – the set of tags that can be associated with bookmark b_i. It includes the tags assigned by humans as well as tags derived from the textual content of the bookmarked document. $THT(b_i) \subseteq TagSpace$.

3.1 Recommending Existing Tags

Given the models defined in Section 2 and the notations introduced above, tag recommendations derived from the existing tag space for a given bookmark (u_0, d_0) are generated by $[THT(d_0) \cap THT(u_0)] \cup TT(d_0, u_0)$. These are existing tags evoked by the textual content of both d_0 and u_0. They may include tags previously assigned to d_0 by other users (if d_0 is part of the training data,

$TH(d_0) \neq \emptyset$) or by u_0 to other documents (if u_0 is part of the training data, $TH(u_0) \neq \emptyset$).

3.2 Suggesting New Tags

In order to recommend tags that do not currently exist in the tag space, we make use of the document and user representations in the concept space. Therefore, for a given bookmark (u_0, d_0), recommendations are generated using $[THT(d_0) \cap THT(u_0)] \cup TC(d_0, u_0)$. Lymba's submission for the RSDC'08 Challenge made use of this tag recommendation mechanism.

4 Data Preparation and Processing

4.1 Experimental Data

For the RSDC'08 Challenge, we received bookmarking data from BibSonomy[3], a web-based social bookmarking system that enables users to tag web documents as well as bibtex entries of scientific publications. Brief statistics of the data can be found in Table 1.

Table 1. RSDC'08 bookmarking data

	Bookmarks	Publications
Training	176,147	92,545
Average no. of tags	3.38	2.37
Test	16,195	43,348
Average no. of tags	2.12	2.27

Each bookmark was described by its URL (e.g., *http://www.bibsonomy.org/*), a description of the URL (e.g., *BibSonomy::home*) that usually maps to its title and an extended description of the bookmark (e.g., *BibSonomy is a system for sharing bookmarks and lists of literature.*). We note that the user that is bookmarking a URL has complete control over the bookmark's descriptions.

Each bookmarked publication is associated with values of bibtex fields such as *title*, *author*, *booktitle*, *journal*, *series*, *editor*, *publisher*, *volume*, *number*, *pages*, etc. In addition to this information, the *entrytype*, *bibtexKey*, *bibtexabstract*, and *URL* can be specified. The user can also input their own description of the publication. Miscellaneous information is collected in the *misc* field. This may include user comments, non-standard bibtex fields, e.g., *isbn* (1532-0626

[3] http://www.bibsonomy.org

(print), 1532-0634 (electronic)), *doi* (10.1002/cpe.607), and *bibdate* (Mon Feb 25 14:51:24 MST 2002). For instance, one of the publications bookmarked by user *26* is described by the information shown in Table 2.

Table 2. Bookmarked publication in BibSonomy

Title	Temporal and real-time databases: a survey
Author	G. Ozsoyoglu and R. T. Snodgrass
Journal	Knowledge and Data Engineering, IEEE Transactions on
Volume	7
Number	4
Pages	513–532
Year	1995
EntryType	article
BibtexKey	ozso95
BibtexAbstract	A temporal database contains time-varying data. In a real-time database transactions have deadlines or timing constraints. In this paper we review the substantial research in these two previously separate areas. First we characterize the time domain; then we investigate temporal and real-time data models. We evaluate temporal and real-time query languages along several dimensions. We examine temporal and real-time DBMS implementation. Finally, we summarize major research accomplishments to date and list several unanswered research questions
URL	http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=404027
Misc	- I'm not clear why temporal databases are used. Does Amazon use them? I suspect they just annotate the row with a timestamp as needed. - paper surveys variety of research, quite extensive. Tries to combine temporal dbs with real-time dbs.
Description	sdasda

4.2 Data Normalization

Bookmarks are described by their URL. Thus, we planned to use the *url_hash* information associated with each bookmark to identify the unique list of bookmarked webpages. However, multiple URLs can pinpoint to the same webpage. Table 3 lists the different URLs found in the training data that identify the *BibSonomy* main webpage. Therefore, our data preparation process included an URL normalization step that unified the bookmarked URLs.

As a result of this step, several bookmarks were merged into a single bookmark to which we assigned the descriptions of the given bookmarks. For instance, seven URLs bookmarked by user *1339* point to the same webpage (Table 4). Clearly, user *1339* intended to bookmark different sections of the webpage (he also bookmarked the entire page), however, using existing social bookmarking tools, users cannot explicitly bookmark portions of documents. In Ta-

Table 3. URLs describing BibSonomy

http://www.bibsonomy.org/
http://www.bibsonomy.org
http://bibsonomy.org/
http://bibsonomy.org

Table 4. Duplicate bookmarks were merged

User Id	URL	Tag(s)
1339	http://www.gehspace.com/arte26a30.htm#26	arte
1339	http://www.gehspace.com/arte26a30.htm#27	arte
1339	http://www.gehspace.com/arte26a30.htm#28	arte
1339	http://www.gehspace.com/arte26a30.htm#inicio	arte
1339	http://www.gehspace.com/arte26a30.htm#29	arte
1339	http://www.gehspace.com/arte26a30.htm#30	arte
1339	http://www.gehspace.com/arte26a30.htm	arte
1339	http://www.gehspace.com/arte26a30.htm	arte

ble 4's last row, we display the bookmark that unifies the seven (URL,tags) pairs.

A similar unification process was performed on the set of bookmarked publications. For publications, we used the *simhash1* field information. In Table 5, we display one example.

Table 5. Duplicate publications were merged

User Id	Publication	Tag(s)
82	Title: *How triadic diagrams represent conceptual structures*, Author: *Klaus Biedermann*, Series: *LNAI*	triadic formal concept lattices analysis fba begriffsanalyse forschungsgruppe tu ag1 formale darmstadt fca
82	Title: *How Triadic Diagrams Represent Conceptual Structures*, Author: *K. Biedermann*, Series: *Lecture Notes in Computer Science*	FCA OntologyHandbook
82	Title: *How Triadic Diagrams Represent Conceptual Structures*, Author: *K. Biedermann*, Series: *LNAI*	FCA OntologyHandbook
82	Title: *How triadic diagrams represent conceptual structures*, Author: *Klaus Biedermann*, Series: *LNAI, Lecture Notes in Computer Science*	triadic formal concept lattices analysis fba begriffsanalyse forschungsgruppe tu ag1 formale darmstadt fca ontology-handbook

A second preprocessing stage was performed for all publications whose *misc* field was comprised of (field,value) pairs. A manual review of all bibtex fields stored in *misc* by one of the authors of this paper resulted in a map that augments the existing information stored for a publication. For instance, the values stored within *misc* for the field name *englishtitle* were added to the *title* information. Similarly, the *confname* values augmented the *booktitle* information. However, not all bibtex fields stored within *misc* could be mapped to already existing columns. Therefore, new fields were generated and populated with the corresponding values as stored in the *misc* field. Among the new columns, we have *classification, contents, keywords, review, subject, topic,* and others.

Given that we had this rich meta information for only some of the publications, we attempted to fill the missing information by exploiting the data that digital libraries such as *ieeexplore.ieee.org, portal.acm.org, or sciencedirect.com* provide for the publications that they host. Thus, for any publication whose URL pointed to one of these websites, we downloaded the html files, parse them, extracted any piece of information that belonged to the publication's metadata and stored them in the corresponding bibtex fields. We note that the html files downloaded from these resources have a uniform structure and can be easily parsed.

An additional data preparation step was required for the RSDC'08 data. Most of the textual information that is associated with scientific publications contains LaTeX markings. By removing these markings, each publication was associated with plain text that can be accurately processed by our natural language processing tools.

4.3 Data Processing

As mentioned in Section 2, each bookmark is associated with the textual content of the metadata that accompanies it and certain concepts are derived from each piece of information. However, the importance of the identified concepts differs from column to column. For instance, the concepts derived from a publication's *address* field are less important than one ones derived from the *journal* information which are less important than the *title* concepts when it comes to the concept's likelihood to become a tag assigned to the document. Therefore, when we aggregated the concept information for a bookmark for the purpose of deriving the bookmark's representation in the concept space ($TC(b_i)$), we used various field weights that denoted the importance of the field. For Lymba's RSDC'08 submission we used a set of heuristics to assign weights to the different fields that describe the URLs and the publications. However, we plan to learn these weights from the training data within a machine learning framework.

5 Experiments and Observations

For the RSDC'08 challenge, we normalized the test data according to the processing described in Section 4.2. For the testing bookmarks/publications that mapped to bookmarks existing in the training data, we returned the tags assigned to these bookmarks according to the training data. 602 bookmarks/publications were tagged by this process with a maximum average F-measure of 96.95% (at top 1) and a minimum average F-measure of 96.70% (at top 2).

5.1 Results

In Table 6, we show the performance of the system on the RSDC'08 testing data. The highest F-measure (21.33%) is achieved for the top 5 tag recommendations with the highest precision at top 1 and highest recall at top 10. Our measures of the system's performance on the two types of bookmarks (URLs and publications) – also shown in Table 6 – releaved a significant difference between the quality of the tag recommendations made for a publication (highest F-measure – at top 5 – is 27%) and the quality of the recommendations made for a bookmark (highest F-measure : 7%).

Table 6. Performance on the test data (average recall, precision and f-measure over the testing bookmarks/publications for the top N tag recommendations – $N \in \{1, \ldots, 10\}$). Performance numbers are shown for the entire test data, only for bookmarks, and only for publications.

Top N	Recall	Precision	F-measure
1	0.2062 : 0.0700 / 0.2572	0.2062 : 0.0700 / 0.2572	0.2062 : 0.0700 / 0.2572
2	0.2089 : 0.0656 / 0.2625	0.1892 : 0.0629 / 0.2365	0.1986 : 0.0643 / 0.2488
3	0.2515 : 0.0665 / 0.3207	0.1749 : 0.0586 / 0.2185	0.2063 : 0.0623 / 0.2599
4	0.3005 : 0.0684 / 0.3873	0.1635 : 0.0537 / 0.2046	0.2118 : 0.0601 / 0.2678
5	0.3468 : 0.0717 / 0.4497	0.1540 : 0.0502 / 0.1929	0.2133 : 0.0591 / 0.2700
6	0.3894 : 0.0745 / 0.5072	0.1460 : 0.0469 / 0.1830	0.2123 : 0.0575 / 0.2690
7	0.4266 : 0.0792 / 0.5565	0.1389 : 0.0457 / 0.1737	0.2096 : 0.0579 / 0.2648
8	0.4592 : 0.0816 / 0.6004	0.1322 : 0.0437 / 0.1653	0.2053 : 0.0569 / 0.2592
9	0.4865 : 0.0844 / 0.6369	0.1257 : 0.0424 / 0.1569	0.1998 : 0.0564 / 0.2517
10	0.5073 : 0.0860 / 0.6649	0.1193 : 0.0412 / 0.1485	0.1932 : 0.0557 / 0.2428

A closer look at the set of 16,194 bookmarks given in the test dataset revealed 9,183 bookmarks were assigned a single tag (*indexforum*) which was not among the top 10 tag recommendations returned by our system. These bookmarks are various articles from *http://forum.index.hu*. Table 7 displays three of these bookmarks. The tag assigned to these URLs can be derived from their common URL. However, our system recommends the concepts it identified in the bookmark's content without joining them. For instance, our tag recommendations for the second URL shown in Table 7 are *tft, monitorok, forum, article, la, index, 9156398, hu,* and *78013830*.

Table 7. Quality of out-of-tag-space recommendations and in-tag-space recommendations measured on the test data

URL	Title
http://forum.index.hu/Article/showArticle?t=9096213&la=73045864	tomor, szines programozasi nyelv
http://forum.index.hu/Article/showArticle?t=9156398&la=78013830	TFT monitorok
http://forum.index.hu/Article/showArticle?t=9013231&la=34610778	Temetni jottem a Linuxot, nem dicserni!

We evaluated the system's performance on the test data from which we eliminated the 9,183 URLs from *http://forum.index.hu*. The performance, as it can be seen in Table 8, improves by 4% (F-measure of top 5 tag recommendations). Our immediate plans are to expand our system to include joined concepts in the tag recommendations that it makes.

Table 8. System performance on the test data excluding the *http://forum.index.hu* bookmarks

Top N	Recall	Precision	F-measure
1	0.2062 / 0.2439	0.2062 / 0.2439	0.2062 / 0.2439
2	0.2089 / 0.2470	0.1892 / 0.2238	0.1986 / 0.2348
3	0.2515 / 0.2974	0.1749 / 0.2069	0.2063 / 0.2440
4	0.3005 / 0.3554	0.1635 / 0.1934	0.2118 / 0.2505
5	0.3468 / 0.4101	0.1540 / 0.1822	0.2133 / 0.2523
6	0.3894 / 0.4605	0.1460 / 0.1726	0.2123 / 0.2511
7	0.4266 / 0.5045	0.1389 / 0.1642	0.2096 / 0.2478
8	0.4592 / 0.5431	0.1322 / 0.1563	0.2053 / 0.2428
9	0.4865 / 0.5754	0.1257 / 0.1486	0.1998 / 0.2363
10	0.5073 / 0.6000	0.1193 / 0.1411	0.1932 / 0.2285

The difference in the system's performance on the bookmark data as opposed to the test publications (Table 6) may also be cause by the difference in textual content that the system associated with each bookmark/publication. The bibtex entries were much richer in textual content when compared with the URL descriptions.

5.2 Impact of Tag Space Expansion

Our approach to tag recommendations (described in Section 3) enables us to limit our suggestions to the existing tag space as well as expand the set of tag recommendations to include concepts the were not previously assigned as tags. In this section, we show the impact that the process of recommending out-of-tag-space concepts had on the performance of our system. In Table 9, we show the performance of the system when we limited the tag recommendations to the tag space generated based on the training data. The highest F-measure, 10.04% – achieved for the top 4 tag recommendations, is significantly smaller when compared with the system's F-measure when it recommends concepts that do not exist in the current tag space. The understanding of the bookmark/publication's content (the textual information that accompanies the URL/publication) doubles the quality of the tag recommendations.

Table 9. Quality of out-of-tag-space recommendations and in-tag-space recommendations measured on the test data

Top N	Recall	Precision	F-measure
1	0.2062 / 0.0927	0.2062 / 0.0927	0.2062 / 0.0927
2	0.2089 / 0.0935	0.1892 / 0.0868	0.1986 / 0.0900
3	0.2515 / 0.1137	0.1749 / 0.0842	0.2063 / 0.0968
4	0.3005 / 0.1345	0.1635 / 0.0801	0.2118 / 0.1004
5	0.3468 / 0.1498	0.1540 / 0.0748	0.2133 / 0.0998
6	0.3894 / 0.1600	0.1460 / 0.0693	0.2123 / 0.0967
7	0.4266 / 0.1660	0.1389 / 0.0639	0.2096 / 0.0922
8	0.4592 / 0.1696	0.1322 / 0.0593	0.2053 / 0.0879
9	0.4865 / 0.1718	0.1257 / 0.0557	0.1998 / 0.0842
10	0.5073 / 0.1732	0.1193 / 0.0531	0.1932 / 0.0813

6 Conclusion

In this paper, we briefly describe Lymba's approach for generating tag recommendations for bookmarks/publications. We exploit the textual content that can be associated with bookmarks, documents and users and generate models within the concept space or the existing tag space. Using these rich models, our tag recommendation system is able to suggest various concepts as tags for a given bookmark. The quality of the tag recommendations generated from concept space (without being constrainted to the tag space produced by the training data) exceeds by far the appropriateness of the tags suggested from the current tag space.

Further analysis is required to identify which features contribute and by how much, towards a particular tag. The semantic relation between the tags in the tag space can be exploited to obtain better weighting of tags and improved models for documents and users.

Our future work will focus also on designing an “adaptive tag recommendation” system which uses the chronology of the bookmarking events to grow the tag space and learn from all previously stored bookmarks.

References

1. Mishne, G.: Autotag:a collaborative approach to automated tag assignment to weblog posts. In: Proceedings of WWW’06, Edinburgh, Scotland (2006)
2. Heymann, P., Ramage, D., Garcia-Molina, H.: Social tag prediction. In: Proceedings of SIGIR’08, Singapore (2008)

Author Index

Keyword Index