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

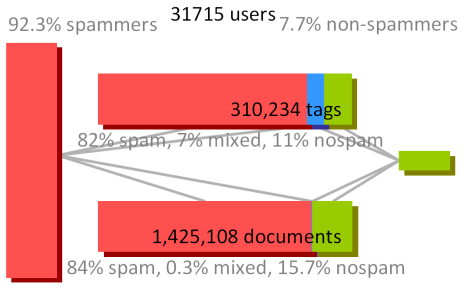


Fig. 1. Distribution of spam in the different sets

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.

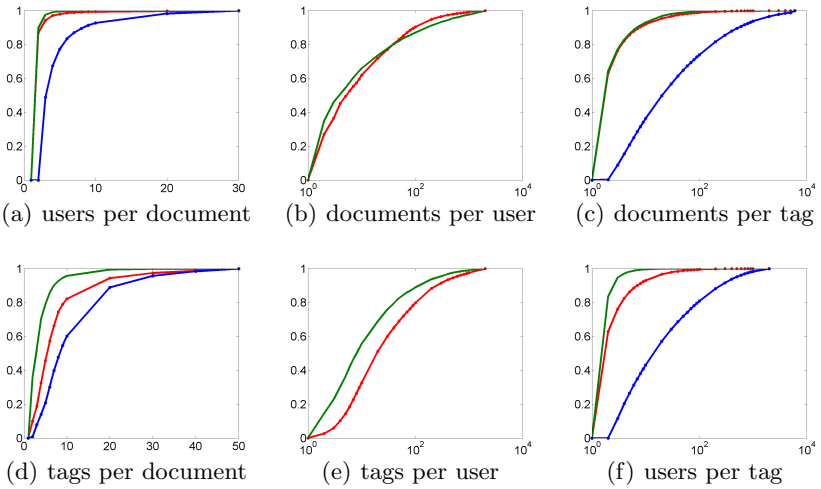
2 Dataset

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.

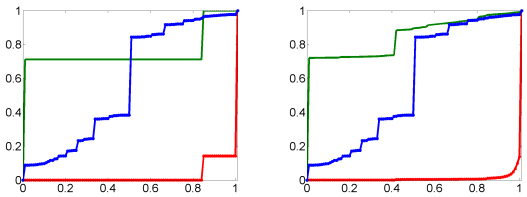
2.2 Creating a probe dataset

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

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

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

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

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

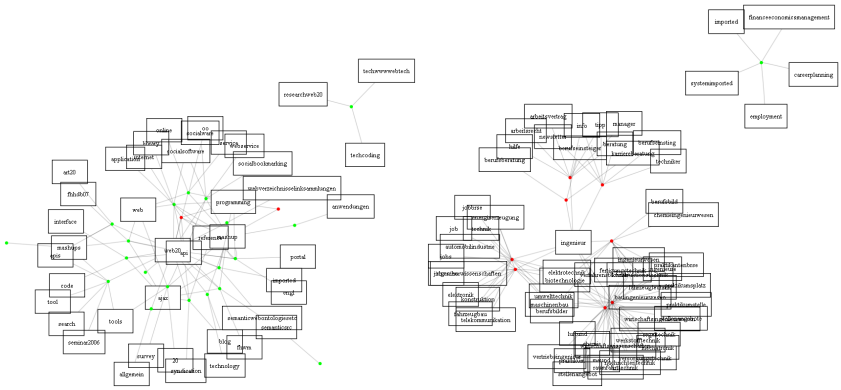


Fig. 4. Induced bipartite graph E_D for two documents. Dots are spammers (red) and non-spammers (green)

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 cooccurence 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

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

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

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

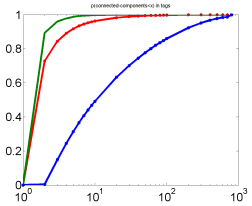
Last, we aimed to translate network features into useful user features. To examine only relevant portions of the overall graph structure, we define "induced graphs" as the bipartite graphs gained by fixing an element from one of the three sets, and connecting those elements from the other two sets that 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

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

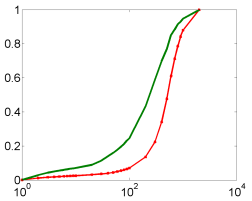
Refer to Figure 4 for two examples of bipartite graphs induced by a (mostly) non-spam- and spam document, respectively, rendered using the GraphViz package[1]. We hope these examples convey the intuition that graphs created by spamming should have different properties than those created by legitimate 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

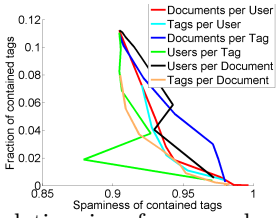
$$d_{d,t}(u) = \frac{\text{avg degree}(D(u))}{|T(u)|} \cdot \frac{\text{avg degree}(T(u))}{|D(u)|},$$

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

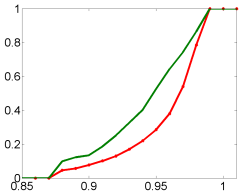
3.4 K-Cores

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 $\begin{pmatrix} \text{a} & \text{b} \\ \text{c} & \text{d} \\ \text{e} & \text{f} \end{pmatrix}$-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):

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



(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

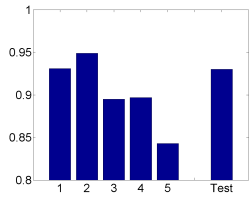
4.1 Single Features

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

Feature name	AUC(val.)	AUC(test)	Used by 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.

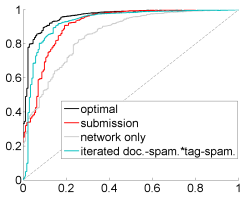
Induced Graphs Many of the features describing the statistical properties of induced graphs seem to indicate a tendency towards spaminess or non-spaminess, but no single feature is useful as a stand-alone predictor. The degree ratio between documents and tags in user-induced graphs, and the number of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

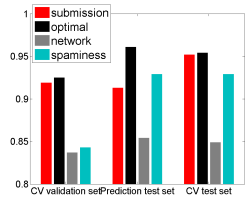
Other features The ratio of tags per user, for documents, turns out to be a useful measure, as motivated by the graphs shown in Figure 4. Also, the simple number of entries per user provides a tendency towards spaminess.

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.