Comment spam classification in blogs through comment analysis and comment-blog post relationships

1 AUTHOR:

Ashwin Rajadesingan
Arizona State University

3 PUBLICATIONS   **5** CITATIONS

SEE PROFILE

Comment Spam Classification in Blogs through Comment Analysis and Comment-Blog Post Relationships

Ashwin Rajadesingan and Anand Mahendran

VIT University
School of Computer Science and Engineering
Vellore, TN, India-632014
{ashwinr2008,manand}@vit.ac.in

Abstract. *Spamming* refers to the process of providing unwanted and irrelevant information to the users. It is a widespread phenomenon that is often noticed in e-mails, instant messages, blogs and forums. In our paper, we consider the problem of spamming in blogs. In blogs, spammers usually target commenting systems which are provided by the authors to facilitate interaction with the readers. Unfortunately, spammers abuse these commenting systems by posting irrelevant and unsolicited content in the form of spam comments. Thus, we propose a novel methodology to classify comments into spam and non-spam using previously-undescribed features including certain blog post-comment relationships. Experiments conducted using our methodology produced a spam detection accuracy of 94.82% with a precision of 96.50% and a recall of 95.80%.

Keywords: spam detection, comment spam, blog spam, text mining.

1 Introduction

Over the past decade, blogs have gained immense popularity in the Internet. A *blog* or *weblog* is an online journal or diary that usually allows interactions between the author and the readers through comments. According to WordPress[1], a blog publishing platform, their users alone produce an average of 500,000 new posts and 400,000 new comments everyday. Unfortunately, with such huge volumes of traffic in a largely unmoderated space, blogs have become a target for spammers. Spammers have expanded from spamming traditional e-mail and messaging systems to social networks, blogs, forums etc. Akismet[2], a plug-in for comment and trackback spam detection, has detected over 25 billion spams over the past four years in Wordpress blogs. Thus, with such high levels of spam in blogs, it is imperative that we constantly devise newer strategies to combat them.

The different types of spam in blogs include splogs, comment spam, trackback spam etc[3]. In this paper, we restrict our scope to detecting comment spam which is the most common type of blog spam. Studies indicate that 81% of blogs

have commenting systems[4] which allow the author to interact with the readers. A typical commenting system contains text fields for commenter's name, homepage url, e-mail address and a text area for typing comments. These commenting systems are exploited unethically by spammers who post advertisements, irrelevant links and malware in the text area as spam comments. These comments are generated through automated applications called *bots* which repetitively post irrelevant and often malicious content as comments.

Spam e-mail detection methodologies used in detecting spam comments have been reasonably successful[5] but cannot be viewed as a full-fledged solution to the comment spam problem. Their fallacies may be attributed to the inherent difference in the features of spam comments and spam e-mails. While the purpose of a spam e-mail is to coax the recipient into interacting with the solicited website, the purpose of a spam comment is to improve the search engine rankings of the advertised website. Also, unlike spam e-mails, spam comments (as shown in Fig. 2,3,4) are optimized according to the ranking algorithms of search engines such as Google through *Search Engine Optimization* (SEO) techniques. For example, the Google search engine employs its PageRanking algorithm to rank websites based on the weighted sum of their incoming links[6]. Thus, spammers use a SEO technique called *link building*[7] which involves repeatedly posting links in blogs, forums etc., to increase the incoming links and thereby, improving the search rankings.

Currently, blog owners and blogging platforms such as WordPress have adopted certain techniques to reduce comment spam. Some blog owners choose to manually monitor and moderate comments. While this process may be effective in removing spam completely, it is laborious and unfeasible especially if the blog attracts a large amount of traffic. Also, some blog owners disallow multiple postings of the same comments in their blogs. This approach prevents some but not all spam comments from being posted. Another approach is to prevent comment spam by distinguishing automated spamming bots from genuine commenters using CAPTCHAs[8]. *CAPTCHAs* are puzzles that usually involve recognizing letters or numbers from cluttered images that are difficult for bots to automatically identify. However, research has proved that this method is not foolproof and that it can be broken[9]. Yet another approach is to attach a "nofollow" link attribute to the commenting systems[10]. The "nofollow" attribute directs the search engine crawlers not to follow the links posted in comments. Thus, these links do not contribute to the page rank of the linked page during search queries. Unfortunately, spammers continue to spam even "nofollow"-attributed commenting systems as experiments conducted by SEO communities show that the links posted in such commenting systems are still followed by some crawlers[11].

The rest of the paper is organized as follows. In section 2, we discuss the past works related to comment spam. In section 3, we describe the dataset used for the validation of our methodology. In section 4, we identify and describe features required for our proposed methodology. In section 5, we provide a mathematical model that combines the features extracted in section 4. In section 6, we describe our experimental setup and the results obtained on applying our methodology.

In section 7, we conclude and explore the scope for further research on comment spam.

2   Related Work

Spam detection has been an active research area over the past decades with considerable work done primarily on email spam[12][13][14]. However, specific research on comment spam started only in 2005 and has yet to gain much prominence.

In 2005, Mishne et al[15] used probabilistic language models to detect spam comments. The difference in language models (calculated using a smoothed KL-Divergence) of the blog post, its comments and the pages that were linked by the comments were in the spam detection process. A major drawback of this method is that spam classification is solely based on comparing language models. Thus, spam comments that have language models similar to that of the blog post may pass the spam filters without detection. In 2006, Han et al[16] proposed a collaborative filtering method for detecting spam links in blog comments. In their method, blog owners manually identify and share spam links through a trusted network of blogs called *trustroll)* to aid in spam detection. This approach can be applied only to user-hosted blogs (eg. Wordpress) and not developer-hosted blogs (eg. Blogger) as blog owners do not have the facility to create custom trustrolls in developer-hosted blogs. Also in 2006, Wong et al.[17] proposed a collaborative security system to detect spam comments. Their system automatically identified spam comments and constructed signatures which were distributed to a set of peers to assist in their spam detection process. In their system, for each spam comment detected, a signature is created and stored in a database before it is distributed to its peers in the network. This methodology is difficult to put into practical use because the database size and the network traffic increase with increase in spam comments. In 2007, Cormack et al[5] worked on spam filtering for short messages such as comments by analyzing and evaluating the available filtering systems such as Bogofilter, OSBF-Lua etc. They focused their analysis purely on comments and did not correlate the comments with their corresponding blog posts. In 2009, Bhattarai et al[18] performed content analysis of spam comments to identify features such as number of word duplications, stop words ratio etc., which were used to train classifiers for spam detection. They obtained an accuracy of 86% in detecting spam comments using their approach.

Previous works on spam comment detection relied on methodologies using language models, collaborative approaches and content analysis techniques. However, these approaches did not taken into extensive consideration, the meta-data in blogs such as time of posting, name of commenters etc., and focused purely on the text content of the blog post and its comments. To the best of our knowledge, our methodology is *the first to integrate features based on meta-data describing both comments and their relationship with blog posts for detecting spam comments.* In our work, we propose a novel methodology which combines the results from content analysis of comments and blog post-comment relationships to train

classifiers such as Naive Bayes, Support Vector Machines (SVM) etc., to detect spam comments. Our approach is more robust and accurate when compared to previous works as the comments are classified not only based on their properties but also based on their correlation with the blog posts.

3   Dataset

We use a blog corpus compiled by Mishne et al[15] for evaluating our methodology. This corpus contains 50 random blog posts with 1024 comments. The number of comments per blog post range from 3 to 96 and the average length of the comments is 41 words. The blogs and the comments are predominantly in English (over 90%). These comments were classified by human evaluators into spam and non-spam. The corpus contains 332 non-spam comments and 692 spam comments (about 67%). This is a realistic representation of the percentage of spam comments in the blogosphere and is in accordance with values obtained from recent observations[19]. All examples featured in this paper have been extracted from this corpus.

4   Feature Selection

Spam comments have certain defining features which distinguish them from non-spam comments. We analyzed comments and identified six such features which can be used to train classifiers in detecting spam comments. In this process, we used Beautiful Soup, a HTML parser library[20] and NLTK library[21] (Natural Language Toolkit) for extracting and evaluating blog posts along with their corresponding comments.

4.1   Features Based on Comment Analysis

The following features are based purely on the the properties of comments. Here, we analyze the content and the meta-data related to the comments in order to identify features that aid in the spam detection process.
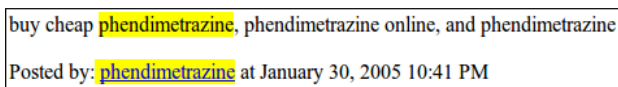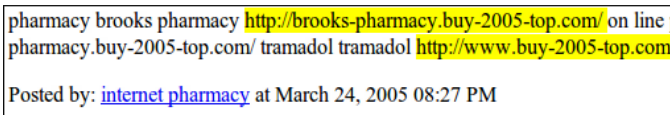
buy cheap phendimetrazine, phendimetrazine online, and phendimetrazine

Posted by: phendimetrazine at January 30, 2005 10:41 PM

Fig. 1. An example of a spam comment containing the commenter's name

Presence of References to Own Name. Commenting systems always provide a separate name field in which the commenter may input his/her name. Thus, genuine commenters generally never find the need to post their names in the comment body. However, spammers post multiple copies of keywords as names in both the name field and the comment body of commenting systems in order to increase the keyword density which improves search rankings[22]. In the corpus, it is observed that 93.17% of comments referring to the commenter's own name are spam comments. Thus, the *presence of references to own name* is used as a feature in comment spam detection. In Fig. 1, we observe such a spam comment where "phendimetrazine" is present in both the name field and the comment body.

pharmacy brooks pharmacy http://brooks-pharmacy.buy-2005-top.com/ on line
pharmacy.buy-2005-top.com/ tramadol tramadol http://www.buy-2005-top.com

Posted by: internet pharmacy at March 24, 2005 08:27 PM

Fig. 2. An example of a spam comment containing homepage links

Presence of Homepage Links. Usually, links present in non-spam comments direct the user to a specific inner page rather than the homepage or the subdomain homepage of a website. This is because genuine commenters provide topic-specific information which is available on these inner pages. On the other hand, spammers try to increase the search rankings of their entire website by post links directing to both the homepage and the inner pages of their website. In the corpus, we find that 91.05% of comments containing links directing to homepages are classified as spam. Hence, the *presence of homepage links* is used to distinguish spam from non-spam comments. Figure 2 shows a part of a spam comment containing links to homepages.

Presence of Dictionary Words in Name Field. Spammers often input keywords in the name field and website links in the comment body of commenting systems to improve the search ranking of their website for the inputted keywords. These keywords are usually dictionary words such as "shopping", "business" etc. Using the pyEnchant programming module[23], we observe that 84.34% of comments having dictionary words in their name field are classified as spam in the corpus. Thus, the *presence of dictionary words in name field* is an effective feature in the spam detection process. Figure 2 shows a part of a spam comment which contains "internet pharmacy" (both dictionary words) in the name field.

4.2   Features Based on Comment-Blog Post Relationships

The following features highlight the properties that link blog posts and comments. These features are especially effective in detecting spam comments as the correlation between blog posts and spam comments are generally very weak.
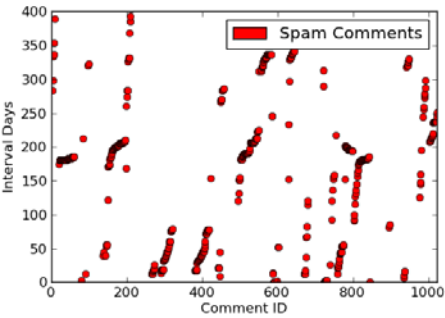
Fig. 3. Graph shows the distribution of spam comments with respect to the time interval between blog and comment posting
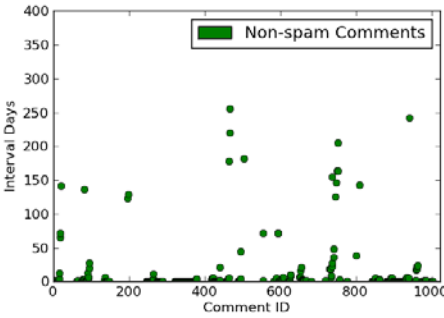
Fig. 4. Graph shows the distribution of non-spam comments with respect to the time interval between blog and comment posting

Time Interval between the Dates of Blog Posting and Comment Posting. As content in blogs is chronologically ordered, we can analyze the time interval between the dates of blog posting and comment posting. It is observed that as this time interval increases, the possibility of a comment being spam also increases. This measure is quite intuitive, for example, if a comment is posted, say, two years after the blog post was posted, the comment is most likely to be spam. We plotted two graphs (Fig. 3 and Fig. 4) showing the distribution of spam and non-spam comments in the corpus based on time interval. In both graphs (Fig. 3 and Fig. 4), "Interval Days" refers to the time interval between the dates of blog posting and comment posting (in days) and "Comment Id" refers to a unique number identifying each comment in the corpus. We observe that most non-spam comments are posted close to the blog post publishing date whereas spam comments have a wider distribution. This difference may be used in the distinguishing spam and non-spam comments and thus, *time interval between the dates of blog posting and comment posting* is a valuable feature which can be utilized in spam detection.

Presence of References to Blog Post Author. The "Comments" section in blogs serves as a discussion platform for commenters and blog post authors. The comments are usually directed at the author or at other commenters by
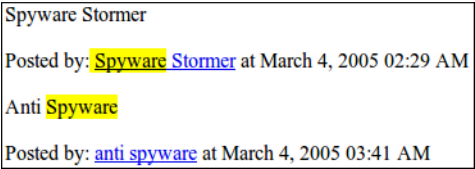

Fig. 5. An example of a spam comment containing another commenter's name

usually referring to their names. In the corpus, it is observed that 91.67% of comments containing references to authors are non-spam comments. Hence, the *presence of references to blog post authors* is a useful feature to differentiate spam and non-spam comments. However, the presence of references to other commenters is not used as a differentiating feature because such references can be forged by posting multiple comments, with atleast one comment containing the other commenter's name. For example, in Fig. 5, the first comment is authored by "Spyware Stormer" and the following comment contains "Spyware", which is incidentally the first name of the previous commenter. Thus, an algorithm using the presence of references to other commenters as a feature would wrongly classify the two comments as non-spam.
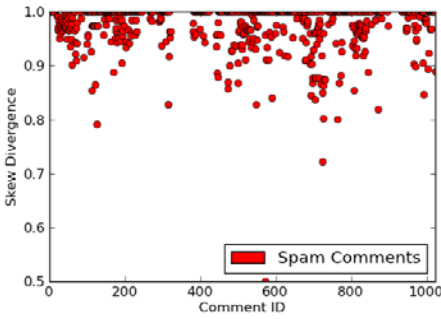

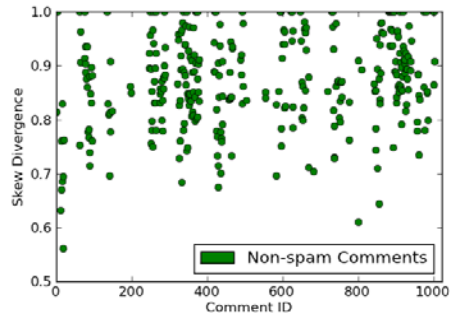**Fig. 6.** Graph shows the distribution of spam comments with respect to skew divergence

Fig. 7. Graph shows the distribution of non-spam comments with respect to skew divergence

Skew Divergence. As spam comments differ greatly in the language used when compared to their corresponding blog post, we compare the language model of the comment with that of the blog post. A language model is a probability distribution over the word sequences present in the text. In our approach, we calculate the language model of the blog post and the blog comments using maximum likelihood estimations and then the skew divergence[24], which is the difference between the two models, is calculated.The skew divergence is asymmetric and is a modification of the Kullback-Leibler divergence (KL-Divergence). The skew divergence S_α between two language models l_1 and l_2,is given by:

$$S_\alpha(l_1 \mid\mid l_2) = KL(l_2 \mid\mid \alpha l_1 + (1-\alpha)l_2) \tag{1}$$

where

$$KL(l_1 \mid\mid l_2) = \sum_y l_1(y)(\log l_1(y) - \log l_2(y)) \tag{2}$$

Here, $KL(l_1 \mid\mid l_2)$ is the KL-divergence of language models l_1 and l_2, y represents each word in l_1 and α is the skew divergence constant. It can be seen that

the skew divergence is a KL-divergence with l_1 smoothed using l_2 according to α. It has been observed and proved by Lee et al[24] that higher values of α tends to produce better results. Thus, we choose α=0.99 for our calculations. As skew divergence is asymmetric, we calculate both $S(l_{post} \parallel l_{comment})$ and $S(l_{comment} \parallel l_{post})$ and find their mean $\overline{S}$ as follows:

$$\overline{S} = \frac{S_{0.99}(l_{post} \parallel l_{comment}) \; + \; S_{0.99}(l_{comment} \parallel l_{post})}{2} \tag{3}$$

$\overline{S}$ is normalized and plotted in two graphs (Fig. 6 and Fig. 7) which shows the distribution of the spam and non-spam comments based on their normalized mean skew divergence. In the two graphs (Fig. 6 and Fig. 7), "Skew Divergence" refers to the divergence values and "Comment Id" is a unique number identifying each comment in the corpus. From the graphs, we observe that most spam comments have a higher skew divergence when compared to non-spam comments. This observation concurs with our intuitive understanding that language models of spam comments differ greatly from that of blog posts. Hence, the difference in *skew divergence* values of spam and non-spam comments aids in detecting spam comments.

5   Combining Features

The features described in section 4 perform poorly in the spam detection process when taken into consideration individually, but when these features are combined to train a classifier, they perform comment spam detection accurately. Before calculating and combining feature values, appropriate preprocessing is performed on all the blog posts and comments in the dataset. Preprocessing includes stemming, removing stop words, punctuation etc., which improve the overall accuracy of our process. After preprocessing, values for all the features are calculated and are used to represent the comments.

Our approach can be mathematically defined as follows:

Let us assume that each comment instance is a point in an instance space. All comments can be described by the six features mentioned in section 4. These features have domains D_i (i=1 to i=6) as shown in table 1.

Table 1. Domain Details

Features	Domain Name	Domain
Presence of references to own name	D_1	Boolean
Presence of homepage links	D_2	Boolean
Presence of dictionary words in name field	D_3	Boolean
Time interval between the dates of blog posting and comment posting	D_4	Continuous
Presence pf references to blog post authors	D_5	Boolean
Skew Divergence	D_6	Continuous

As shown in table 1, time interval and skew divergence are in continuous domain, while the other features take boolean values (true represents the presence of that feature in the comment while false represents otherwise).

Thus, each comment instance C in the corpus can be represented as:

$$C \; = \; D_1 \times D_2 \times D_3 \times D_4 \times D_5 \times D_6 \tag{4}$$

These comment instances, along with their manual classifications (spam or not spam) are used to train learning algorithms to detect spam comments.

6   Experimental Setup and Results

The spam detection problem is essentially a binary-text classification problem (classification into spam and non-spam). In our methodology, we train classifiers such Nave Bayes, Support Vector Machines (SVM) etc., to obtain a model which is then tested for accuracy. We use different classifiers in order to analyze and evaluate the classifier that is most accurate for our classification problem.

Firstly, a dataset containing the six feature values and the manual classification for each comment in the corpus is compiled. Then, we use a ten-fold cross validation process[25] to test and evaluate the classifications made by the classifier. Here, the compiled data set is divided into 10 equal parts. The classifier is trained and tested 10 times where each time, a different part of the dataset is the testing set while the remaining parts are combined to form the training set. This process helps avoid the possibility of overfitting[26] and gives an accurate estimation of the accuracy of our classifier. The accuracy of the classifier is determined to be the total number of correct classifications divided by the total number of classifications made by the classifier. The results for different classifiers are shown in table 2:

Table 2. Results

Classifying algorithms	Accuracy	Precision	Recall
Naive Bayes Classifier	94.04%	95.92%	95.23%
Support Vector Machines (SVM)	92.57%	91.62%	97.97%
Logistic Regression	92.96%	94.92%	94.65%
Decision Trees (C4.5)	94.82%	96.50%	95.80%

From table 2, we observe that all learning algorithms perform very well with the extracted features values. We observe that SVM has the highest recall value but its precision and accuracy is less than that of some of the other classifying algorithms. Decision trees give the highest overall accuracy of 94.82% along with a precision of 96.50% and a recall of 95.80%. The accuracy obtained is 8.82% higher than the accuracy obtained by Bhattarai et al.[18] Also, the accuracy

obtained is much higher when compared to the 83% accuracy obtained by Mishne et al.[15] using the same spam corpus. Since, the relative cost of misclassifying a legitimate comment as spam is very high when compared to misclassifying a spam comment as legitimate, our focus has been to obtain a high precision in our system. Thus, with a precision of 96.50% obtained using decision trees, we believe that we have devised an excellent spam detection methodology with very high precision.

7   Conclusion and Future Work

From our results, we observe that the spam detection accuracy is vastly improved if both comment analysis and blog post-comment relationships are considered during the spam detection process. In our approach, spam comments need to closely mimic non-spam comments not only in their own properties but also in their relationships with the blog posts in order to deceive the classifier. Thus, our approach disencourages spammers by making spamming more computationally expensive as spammers would need to post comments customized according to the blog post content. Also, we believe that our methodology is relatively language independent as most of the features mentioned in section 4 are not language dependent (such as date of comment posting, author's and commenter's names etc.).But, as the size of the corpus is small, we consider our results as a proof-of-concept and a base for further experimentation. In the future, we look to improve and expand the blog spam corpus. Also, we wish to include more features (such as those chosen by Bhattarai et al[18] and test the level of language independence of our methodology. We would also like to incorporate a collaborative spam detection module for better efficiency. Another possible extension of our work would be to use WordNet[27] to identify similar words present in comments and blog posts.

References

1. Wordpress,   http://web.archive.org/web/20110307112536/http://en.wordpress.com/stats/ (retrieved in 2011)
2. Akismet, http://web.archive.org/web/20110523025730/http://blog.akismet.com/2011/04/08/25-billion-pieces-of-spam/ (retrieved in 2011)
3. Thomason, A.: Blog spam: A review. In: Fourth Conference on Email and Anti-Spam (CEAS) (2007)
4. Sobel, J.: State of the blogosphere (2010), http://web.archive.org/web/20110325150629/http://technorati.com/blogging/art-icle/state-of-the-blogosphere-2010-introduction/ (retrieved in 2011)
5. Cormack, G.V., Hidalgo, J.M.G., Sanz, E.P.: Spam filtering for short messages. In: ACM Sixteenth Conference on Information and Knowledge Management, pp. 313–320 (2007)
6. Page, L., Brin, S., Motwani, R., Winograd, T.: The pagerank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab (1999)

7. Malaga, A.R.: Search engine optimization black and white hat approaches. In: Zelkowitz, M.V. (ed.) Advances in Computers: Improving the Web. Advances in Computers, vol. 78, ch. 1, pp. 1–39. Elsevier (2010)

8. von Ahn, L., Blum, M., Hopper, N.J., Langford, J.: Captcha: Using Hard ai Problems for Security. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 294–311. Springer, Heidelberg (2003)

9. Mori, G., Malik, J.: Recognizing objects in adversarial clutter: Breaking a visual captcha. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 1, pp. 134–141 (2003)

10. GoogleBlog, `http://web.archive.org/web/20110716122606/http://googleblog.blog-spot.com/2005/01/preventing-comment-spam.html` (retrieved in 2011)

11. ThoughtMechanics: Does nofollow attribute work? google says yes, studies say otherwise,
`http://web.archive.org/web/20110104152458/http://www.thoughtmechanics.com/does-nofollow-attribute-work-google-says-yes-studies-say-otherwise/`
(retrieved in 2011)

12. Drucker, H., Wu, D., Vapnik, V.: Support vector machines for spam categorization. IEEE Transactions on Neural Networks 10, 1048–1054 (1999)

13. Androutsopoulos, I., Koutsias, J., Chandrinos, K.V., Paliouras, G., Spyropoulos, C.D.: An evaluation of naive bayesian anti-spam filtering. Computing Research Repository (2000)

14. Androutsopoulos, I., Koutsias, J., Chandrinos, K.V., Spyropoulos, C.D.: An experimental comparison of naive bayesian and keyword-based anti-spam filtering with personal e-mail messages. In: Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2000, pp. 160–167. ACM, New York (2000)

15. Mishne, G., Carmel, D., Lempel, R.: Blocking blog spam with language model disagreement. In: First International Workshop on Adversarial Information Retrieval on the Web, pp. 1–6 (2005)

16. Han, S., Ahn, Y-Y., Moon, S., Jeong, H.: Collaborative blog spam filtering using adaptive percolation search. In: World Wide Web Workshop 2006, Edinburgh, UK (2006)

17. Wong, B., Locasto, M., Keromytis, A.: Palprotect: A collaborative security approach to comment spam. In: Information Assurance Workshop, pp. 170–175. IEEE (2006)

18. Bhattarai, A., Rus, V., Dasgupta, D.: Characterizing comment spam in the blogosphere through content analysis. In: IEEE Symposium on Computational Intelligence in Cyber Security, CICS 2009, pp. 37–44 (2009)

19. Akismet, `http://web.archive.org/web/20110106110340/http://blog.akismet.com/-2005/10/29/rising-percentage/` (retrieved in 2011)

20. Richardson, L.: Beautiful Soup Documentation (2007)

21. Loper, E., Bird, S.: Nltk: The natural language toolkit. In: Proceedings of the ACL Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics. Association for Computational Linguistics, Philadelphia (2002)

22. Sedigh, A.K., Roudaki, M.: Identification of the dynamics of the google ranking algorithm. In: 13th IFAC Symposium on System Identification (2003)

23. PyEnchant, `http://packages.python.org/pyenchant/` (retrieved in 2011)

24. Lee, L.: On the effectiveness of the skew divergence for statistical language analysis. Artificial Intelligence and Statistics, 65–72 (2001)
25. Kohavi, R.: A study of cross-validation and bootstrap for accuracy estimation and model selection. In: International Joint Conference on Artificial Intelligence, pp. 1137–1143. Morgan Kaufmann (1995)
26. Hawkins, D.M.: The problem of overfitting. Journal of Chemical Information and Computer Sciences 44, 1–12 (2004)
27. Fellbaum, C.: WordNet: An Electronic Lexical Database. Bradford Books (1998)