Detecting Wikipedia Vandalism using WikiTrust[⋆]

Lab Report for PAN at CLEF 2010

B. Thomas Adler[1], Luca de Alfaro[2], and Ian Pye[3]

[1] thumper@alumni.caltech.edu, Fujitsu Labs of America, Inc.
[2] luca@dealfaro.com, Google, Inc. and UC Santa Cruz (on leave)
[3] ipye@cs.ucsc.edu, CloudFlare, Inc.

Abstract WikiTrust is a reputation system for Wikipedia authors and content. WikiTrust computes three main quantities: *edit quality, author reputation,* and *content reputation.* The edit quality measures how well each edit, that is, each change introduced in a revision, is preserved in subsequent revisions. Authors who perform good quality edits gain reputation, and text which is revised by several high-reputation authors gains reputation. Since vandalism on the Wikipedia is usually performed by anonymous or new users (not least because long-time vandals end up banned), and is usually reverted in a reasonably short span of time, edit quality, author reputation, and content reputation are obvious candidates as features to identify vandalism on the Wikipedia. Indeed, using the full set of features computed by WikiTrust, we have been able to construct classifiers that identify vandalism with a recall of 83.5%, a precision of 48.5%, and a false positive rate of 8%, for an area under the ROC curve of 93.4%. If we limit ourselves to the set of features available at the time an edit is made (when the edit quality is still unknown), the classifier achieves a recall of 77.1%, a precision of 36.9%, and a false positive rate of 12.2%, for an area under the ROC curve of 90.4%.
Using these classifiers, we have implemented a simple Web API that provides the vandalism estimate for every revision of the English Wikipedia. The API can be used both to identify vandalism that needs to be reverted, and to select high-quality, non-vandalized recent revisions of any given Wikipedia article. These recent high-quality revisions can be included in static snapshots of the Wikipedia, or they can be used whenever tolerance to vandalism is low (as in a school setting, or whenever the material is widely disseminated).

1 Introduction

Between 4% and 6% of Wikipedia edits are considered vandalism [15]. Although most vandalism is promptly reverted, vandalism on Wikipedia still has negative consequences. First, locating and reverting vandalism ends up consuming the scarce time of Wikipedia editors. Second, the presence of vandalism on the Wikipedia lessens its perceived quality and can be an obstacle to the wider use of its content. For instance, in spite of

[⋆] The authors like to sign in alphabetical order; the order of the authors may not necessarily reflect the relative sizes of the contributions. This research has been in part supported by the Institute for Scalable Scientific Data Management, an educational collaboration between LANL and the University of California Santa Cruz.

Wikipedia being an outstanding educational resource, its use in schools is hampered by the risk of exposing students to inappropriate or incorrect material inserted by vandals. Third, any static compilation of Wikipedia articles, such as those produced by the Wikipedia 1.0 project[13], is liable to contain a non-negligible amount of vandalized revisions. The presence of vandalism in the compilations reduces their appeal to schools and other environments, where their static, no-surprise nature would otherwise be most appreciated. As the compilations are static, and published on media such as DVD disks or USB keys, the only way to remedy the vandalism is to publish new compilations — incurring both significant cost and risking the inclusion of new vandalism.

Automated tools help reduce the impact of vandalism on the Wikipedia by identifying vandalized revisions, both facilitating the work of editors and allowing the automated exclusion of vandalized revisions from static compilations and other settings where they are deemed particularly undesirable [10,11,5,8,4,12]. Corresponding to these two use cases — helping editors, and filtering revisions — we distinguish two aspects of the Wikipedia vandalism detection problem:

- *Zero-delay vandalism detection.* The goal of the automated tool is to identify vandalism as soon as it is inserted. Thus, to identify vandalism the tool can make use only of features available at the time the edit takes place: in particular, no features that can be acquired *in the future* of the edit can be used. This type of vandalism detection is most useful to help the work of editors, who can be altered to the vandalism and take appropriate action in timely fashion. Zero-delay vandalism detection was the focus of the Task 2 PAN 2010 workshop evaluation.
- *Historical vandalism detection.* The goal of the automated tool is to find vandalized revisions wherever they may occur in the revision history of Wikipedia articles. This type of vandalism detection is most useful when filtering vandalism out of the revisions that are displayed to visitors, as in the Flagged Revisions project [14], or included in a static compilation.

The goal of this work was to build, and evaluate the performance of a vandalism detection tool that relies on the features computed by WikiTrust.[4] WikiTrust is a reputation system for Wikipedia authors and content, based on the algorithmic analysis of the evolution of Wikipedia content [1,2]. WikiTrust computes the quality of each revision, according to how much of the change introduced by the revision is preserved in subsequent revisions. This computation involves the comparison of each revision with both previous and subsequent revisions. It results in a quality index comprised between -1, for revisions that are entirely reverted, and $+1$, for revisions whose contribution is kept unchanged. Authors gain or lose reputation according to the quality of the revisions they make. WikiTrust then uses author reputation to compute the reputation of the text comprising each revision, at the granularity of the individual word, according to how well the text has been revised by high-reputation authors. In particular, after each revision, the text that has been inserted or heavily modified has a small amount of reputation, in proportion to the reputation of the revision's author, while the text that has been left unchanged has gained a small amount of reputation, again in proportion to the reputation of the revision's author.

[4] `http://www.wikitrust.net/`

We decided to base our vandalism detection tool on a simple and efficient architecture. WikiTrust stores the information about revision quality, author reputation, and text reputation in database tables that complement the standard database tables used by the Mediawiki software to implement the Wikipedia. To classify a revision as a *vandalism* or *regular* revision, our tool reads information from the WikiTrust and Mediawiki database tables about the revision and its author, and feeds this information to a decision-tree classifier which produces the desired output. We relied on the machine-learning toolset Weka to train and evaluate a classifier [7]. Our decision to rely only on information readily available in the Mediawiki and WikiTrust database tables has enabled us to produce an efficient Web-based API for our classifier: given the revision id of a Wikipedia revision, the API performs the database lookups and return the classification of the revision in milliseconds. This makes our classifier well-suited to the real-time identification of vandalism and filtering of revisions up to the scale of the English Wikipedia.

Since vandalism tends to be performed by anonymous or novice authors, who have little or no reputation, and since vandalism tends to be reverted promptly, corresponding to revisions of low quality as measured by WikiTrust, we expected our tool to perform fairly well at historical vandalism detection. Indeed, when evaluated on the PAN 2010 Wikipedia vandalism corpus [9], our tool was able to achieve a recall of 83.5% of vandalism, with a precision of 48.5% and a false positive rate of 8.2%, corresponding to an area under the ROC curve [6] of 93.4%. To our surprise, our tool performed reasonably well even at the task of zero-delay vandalism detection, achieving a recall of 82.8% with a precision of 28.6% and false positive rate of 14.4%, leading to an area under the ROC curve of 90.9% (these results are summarized in Table 1). The surprise is due to the fact that, in evaluating the performance for zero-delay vandalism, we have had to exclude the two most potent classification features we had: revision quality and user reputation. We had to discard the revision quality feature because it is based on a comparison between the given revision and *future* revisions, and these future revisions are of course not available when a revision is inserted.

On the other hand, the author reputation feature is available at the time a revision is made, and it would be usable in any real use of our tools. Unfortunately, we had to exclude this from the evaluation performed for the PAN 2010 Workshop, due to the time lag between the revisions being used for the evaluation, and the evaluation itself. The problem is that WikiTrust keeps track only of the current value of user reputation. At the time the PAN 2010 Workshop Task 2 evaluation took place, the values of user reputation in the WikiTrust database reflected author reputation as of May 2010. The PAN 2010 Task 2 evaluation dataset was instead based on revisions that had been entered in November or December 2009. Thus, the author reputation values available to us were in the future of the revisions to be evaluated, and we deemed them unsuitable for the evaluation of the performance of zero-delay vandalism detection. Undoubtedly, the performance we report for the zero-delay tool is lower than the real performance we can achieve by including also the user reputation feature.

2 Features and Classification

The WikiTrust vandalism detection tool follows a standard two-phase machine learning architecture, consisting of a feature-extraction component followed by a classifier.

2.1 Features

In selecting the features to feed to the classifier, we have limited our consideration to the features that can be readily derived from the information available in the database tables used by WikiTrust, or by the Mediawiki software that serves the Wikipedia. This constraint was imposed so that the resulting tool could work on-line, in real-time, providing vandalism detection for any Wikipedia revision in a fraction of a second. As the WikiTrust database tables replicate some of the information present in the Mediawiki database tables, in practice we could derive all features from the WikiTrust tables alone: this enabled us to implement the vandalism detection tool as a self-contained web API on top of the WikiTrust database at UC Santa Cruz.

We describe below the features we extracted. We annotate with "H" the features that were extracted for use by the historical classifier, and we annotate with "Z" those that were extracted for use by the zero-delay classifier; we also indicate in brackets the feature name used by the classifier. Not all features we extracted for use by a classifier ended up being used: many were discarded by the classifier training process, as they were of too little significance to be worth using.

– **Author reputation [Reputation] (H).** Author reputation is an obvious feature to use, since vandalism tends to be performed predominantly by anonymous or novice users, both of which have reputation 0 in the system. In the WikiTrust vandalism detection tool, this feature is included both for zero-delay and for historical vandalism detection: author reputation is in fact available at any time for any user. However, for the purposes of the PAN 2010 Workshop evaluation, we have had to forego this feature, due to the time lag between the revisions, entered in November-December 2009, and the values of reputation available to us, updated as of May 2010.
– **Author is anonymous [Anon] (H,Z).** In addition to author reputation, we also considered the fact whether the author was anonymous or not. Interestingly, whenever author reputation was included as a feature, the feature stating whether the author was anonymous or not was not used by the classifier. Evidently, knowing that a revision was authored by a low-reputation author was enough information: whether the author was anonymous, or a novice, did not seem to matter.
– **Time interval to the previous revision [Logtime_prev] (H,Z), time interval to the next revision [Logtime_next] (H).** We provided as features the quantities $\log(1+t)$, where t is amount of time from the preceding revision, or to the following revision. We thought this feature might be useful, as spam is usually reverted promptly. Indeed, the Logtime_next feature was used, but with a very low threshold of 2.74, corresponding to a delay of only a dozen seconds between a revision and the next one.
– **Hour of day when revision was created [Hour_of_day] (H,Z).** We observed a correlation between the probability of vandalism, and the hour of the day at which

the revision was created (timing signals have been used in a more sophisticated way for vandalism detection in [12]). The classifier did not use this feature: either it was unable to exploit it, or the information it contained was subsumed by that contained in other, more significant features.

- **Minimum revision quality [Min_quality] (H).** In WikiTrust, every revision r is judged with respect to several past and future revisions. In detail, the quality $q(r \mid r^-, r^+)$ of r with respect to a past revision r^- and a future revision r^+ is defined by
$$q(r \mid r^-, r^+) = \frac{d(r^-, r^+) - d(r, r^+)}{d(r^-, r)}.$$
where $d(r, r')$ represents the edit distance between r and r' (for the details on this edit distance, see [1]). To understand this formula, it is useful to consider it from the point of view of the author A^+ of the future revision r^+. From the point of view of A^+, the distance $d(r^-, r^+) - d(r, r^+)$ represents how much closer to A^+'s work the revision has become, and thus, it measures the improvement done by r upon r^-. The amount $d(r^-, r)$ measures the amount of change done by introducing r. Thus, $q(r \mid r^-, r^+)$ is a measure of the improvement, divided by the total change: it is equal to -1 for entirely reverted revisions (where $r^- = r^+$), and to +1 if the change introduced by r with respect to r^- is perfectly preserved in r^+.
Every revision is evaluated with respect to up to 6 past and 6 future revisions [3]. The minimum revision quality is the minimum quality computed with respect to all past and future revisions considered. A low value for the minimum revision quality indicates that at least one future author has reverted, in part or entirely, the edit that led to the revision. Minimum revision quality was the most influential feature for detecting vandalism in the historical vandalism detection tool.
- **Total weight of judges [Judge_weight] (H).** Not all triples (r^-, r, r^+) used to compute the quality of revision r are given the same weight. The higher the reputation rep(A^+) of the author A^+ of r^+, the higher the weight we give to the computed quality $q(r \mid r^-, r^+)$. Additionally, if r^+ is very different from both r^- and r, then the computed quality is given less weight, as it it difficult to compute what fraction of the change from r^- to r has been preserved in r^+. Thus, we give to each judging triple (r^-, r, r^+) the weight
$$\exp\left(-\frac{\min(d(r^-, r^+), d(r, r^+))}{3 \cdot (1 + d(r^-, r))}\right) \cdot \log(1 + \text{rep}(A^+)).$$
The total weight of the judges is the total weight of all triples used to judge the revision r. This feature was not used by any classifier.
- **Average revision quality [Avg_quality] (H).** In addition to the minimum revision quality mentioned above, we have also considered the *average* quality of a revision, with respect to the past and future revisions with which it has been compared, weighed as above. In cases in which the minimum revision quality was above the -0.662 threshold, the average quality was a strong signal, with a discrimination threshold of 0.156.
- **Maximum dissent [Max_dissent] (H).** The maximum dissent of a revision measures how close the average revision quality is to the minimum revision quality. This feature turned out to be useful in the classifier.

– **Delta [Delta] (H, Z).** This feature measures the edit distance $d(r, r^{-})$ between a revision and the previous one. This feature was used by the classifiers, mainly to treat very small edits in a more lenient way than longer ones.
– **Revision comment length [Comment_len] (H,Z).** The length of the comment is another feature we considered, as we assumed that vandalism tended to be associated with short comments. The classifier made use of this feature only for the zero-delay detection, and even there the feature did not carry much weight.
– **Next revision comment length [Next_comment_len] (H).** We also considered as a feature the length of the comment of the revision following the revision to classify. Somewhat to our surpise, this feature turned out to be useful: if the next comment was longer than 110 characters, this made it slightly more likely that the revision under consideration was vandalism.
– **Next comment mentioned a revert [Next_comment_revert] (H).** We considered whether the comment of the next revision mentioned a revert or undo. We expected this to be an important feature: after all, most editors label in such a way the corrective actions they take in presence of vandalism. However, our classifier did not make use of this feature: the features of minimum and average revision quality turned out to be much more reliable and less noisy.
– **Previous text trust histogram [P_prev_hist0 ... P_prev_hist9] (H,Z).** Whenever a revision is created, WikiTrust computes the *reputation* of each word of the revision, where the reputation is an integer in the interval $0, \ldots, 9$ [2]. The reputation of a word indicates how well the word has been revised by reputable authors; in particular, words that have been just entered or displaced by authors without reputation (including both novice and anonymous authors) are assigned a reputation of 0. When the revision is created, WikiTrust also computes a 10-column histogram detailing how many words of the revision have each of the 10 possible reputation values, and stores the histogram in the database, in an entry associated with the revision. We renormalized the histogram, so that the columns summed to 1, and we used the renormalized value of each column as a feature. This turned out to be an important feature in the historical vandalism detection tool, and even more so in the zero-delay detection tool. We tried many different renormalizations for the histograms, such as ensuring that the columns sum to 1 (as in this case), or taking the logarithms of every column value (as in the histogram difference, explained later). The different normalizations led to essentially the same classifier performance.
– **Current text trust histogram [Hist0 ... Hist9] (H,Z).** The current value of the text trust histogram was also provided as a feature, in this case without any renormalization. This feature turned out to be useful in most models, and in particular, in the models for zero-delay vandalism detection.
– **Histogram difference [L_delta_hist0 ... L_delta_hist9] (H, Z).** For each possible text trust value $i \in \{0, \ldots, 9\}$, we also included a measure of $\log(1 + |h(i) - h^{-}(i)|) \cdot \mathrm{sign}(h(i) - h^{-}(i))$, where h is the text trust histogram for the current revision, and h^{-} is the text trust histogram for the previous revision. This feature turned out to be useful in both the zero-delay and the historical vandalism tools: an increase in the number of words with reputation 0 was associated with vandalism.

In the historical vandalism detection tool, we found that the behavior of the classifier essentially depended on two strong features: [Min_quality] and [Reputation]. The addition of other features increased performance somewhat, but the set of other features considered was not particularly critical. In the zero-delay tool, on the other hand, the features related to the trust histogram of words played an important role, since they were used in part as proxies for the user reputation feature, which could not be used.

2.2 The Classifier

We based our vandalism tools on standard classifiers, and precisely, on the alternating decision tree classifier available as part of Weka [7]. We experimented with various classifiers provided as part of Weka, and the alternating decision tree classifier (ADTree) was the one that at the same time peformed best, and let to the classification models that were the simplest, and the easiest to implement in a web-based API. Since vandalism is relatively rare, we wished to achieve high recall of vandalism even at the expenses of precision. To this end, we trained the classifier using a cost matrix that specified that the cost of misclassifying a vandalism as a regular revision was β times as large as the cost of misclassifying a regular revision as vandalism. After various experiments, we settled on very small decision trees, consisting of only 10 or 20 nodes. In the PAN 2010 submission, we used $\beta = 10$ and a 20-node classifier. As we will see, the 20-node classifier used in the submission is no better than the simpler 10-node classifier we will present in detail. Again, we attribute this to the predominance of a few, very strong features. The classifier for the submission was trained with the following command:

```
weka.classifiers.meta.CostSensitiveClassifier
   -cost-matrix "[0.0 1.0; 10.0 0.0]" -S 1
   -W weka.classifiers.trees.ADTree -- -B 20 -E -3
```

3 Results

The performance of the classifier is summarized in Table 1. In the table, we specify the number of nodes used in the classifier, and the cost-factor β used to prioritize recall agaist precision. The models with $\beta = 50$ can be used for the problem of selecting a recent, non-vandalized revision of a Wikipedia article, an application where it is more important to reject vandalism, than to avoid false positives. The resulting classifier achieves 92.4% rejection of vandalism, while exhibiting a false positive rate of only 19.8%. As a selection tool for the best revision of an article has the luxury of selecting the *best* among available revisions, we believe the real-world vandalism rejection rate would in fact exceed 92.4%. The models used by the 10-node historical and zero-delay classifiers are reported in Tables 2 and 3; these models offer an insight into the working of the classifier, and on the relative importance of the features in each case. In the tables, the nodes of the decision tree are labeled by integers, and the indentation used denotes the level of the node in the tree.

4 Conclusions

Our approach to the vandalism task is orthogonal to the usual heuristics and natural language processing techniques that are suggested when the topic of Wikipedia arises.

Classifier	Type	Nodes	β	Dataset	Recall	Precision	False Pos.	ROC area
H10b20	Historical	10	20	Training	0.903	0.430	0.078	0.956
H10b20	Historical	10	20	Evaluation	0.835	0.485	0.082	0.934
H20b50	Historical	20	50	Training	0.950	0.276	0.163	0.957
H20b50	Historical	20	50	Evaluation	0.924	0.302	0.198	0.937
Z10b20	Zero-Delay	10	20	Training	0.883	0.286	0.144	0.930
Z10b20	Zero-Delay	10	20	Evaluation	0.828	0.308	0.173	0.909
Z20b10	Zero-Delay	20	10	Training	0.837	0.357	0.098	0.931
Z20b10	Zero-Delay	20	10	Evaluation	0.771	0.369	0.122	0.904

Table 1. Performance summary of the historical and zero-delay vandalism tools, evaluated on the training dataset (via 10-fold cross validation), and on the PAN 2010 evaluation dataset. The classifier used for the PAN 2010 submission is Z20b10.

We build atop WikiTrust [1], a system for computing user and text reputations by using the *implicit* feedback of later editors to compute a quality score for every edit. This has the interesting property of adapting with the community to changing notions of acceptable behavior, which we use to construct user reputation. Utilizing features that included data from the *future* of an edit made notable improvements in our ability to classify the edits either as *vandalism* or *regular*, from an AUC value of 0.909 for our zero-delay tool, to an AUC value of 0.934 for our historical analysis tool.

An open question is whether the information present in our feature set already overlaps other solutions, or adds to the picture. For instance, it would be relatively easy to add NLP analysis at the point where the actual differences of an edit is computed. As part of our WikiTrust project, we are already able to dynamically download revisions and evaluate them. To encourage further research extending the ideas we propose here, we have integrated our results from this work into the live system and made two web-based APIs available on an experimental basis.

The first web API implements our final model and presents the caller with a single numerical result indicating the probability that the named revision is vandalism. For a revision <*id*>, the call

```
http://en.collaborativetrust.com/WikiTrust/RemoteAPI?
method=quality&revid=<id>
```

returns the vandalism estimate for the revision, using the WikiTrust historical vandalism detection tool.

Our second web API will be of greater interest to vandalism detection researchers: we are making available the features for each revision, returned as a JSON[5] result. It uses the same URL structure as the first API, but uses a `method` value of "rawquality" to indicate that the request is for the feature set:

```
http://en.collaborativetrust.com/WikiTrust/RemoteAPI?
method=rawquality&revid=<id>
```

[5] `http://www.json.org`

```
: 0.134
|  (1)Min_quality < -0.662: 0.891
|  |  (3)L_delta_hist0 < 0.347: -0.974
|  |  (3)L_delta_hist0 >= 0.347: 0.151
|  |  (4)Max_dissent < 0.171: -1.329
|  |  (4)Max_dissent >= 0.171: 0.086
|  |  |  (10)Next_comment_len < 110.5: -0.288
|  |  |  (10)Next_comment_len >= 110.5: 0.169
|  (1)Min_quality >= -0.662: -1.203
|  (2)Reputation < 0.049: 0.358
|  (2)Reputation >= 0.049: -1.012
|  |  (6)P_prev_hist5 < 0.01: 0.482
|  |  (6)P_prev_hist5 >= 0.01: -0.376
|  |  |  (7)Avg_quality < 0.156: 0.5
|  |  |  (7)Avg_quality >= 0.156: -2.625
|  |  |  (9)L_delta_hist2 < 0.347: -0.757
|  |  |  (9)L_delta_hist2 >= 0.347: 1.193
|  (5)Logtime_next < 2.74: 1.188
|  (5)Logtime_next >= 2.74: 0.045
|  |  (8)Delta < 3.741: -0.255
|  |  (8)Delta >= 3.741: 0.168
Legend: -ve = False, +ve = True
```

Table 2. H10b20 classifier used by the WikiTrust historical vandalism detection tool.

```
: 0.134
|  (1)L_delta_hist0 < 0.347: -1.018
|  |  (7)Hist0 < 0.5: -0.113
|  |  (7)Hist0 >= 0.5: 0.528
|  (1)L_delta_hist0 >= 0.347: 0.766
|  |  (3)L_delta_hist3 < 0.347: 0.026
|  |  |  (8)L_delta_hist4 < 0.347: 0.1
|  |  |  (8)L_delta_hist4 >= 0.347: -0.751
|  |  (3)L_delta_hist3 >= 0.347: -0.962
|  |  (6)P_prev_hist0 < 0.004: 0.094
|  |  (6)P_prev_hist0 >= 0.004: -0.493
|  (2)Anon = False: -0.576
|  (2)Anon = True: 0.312
|  (4)P_prev_hist9 < 0.115: -0.333
|  (4)P_prev_hist9 >= 0.115: 0.182
|  |  (9)Hist7 < 1.5: 1.217
|  |  (9)Hist7 >= 1.5: -0.029
|  (5)Delta < 2.901: -0.251
|  (5)Delta >= 2.901: 0.182
|  (10)Comment_len < 18.5: 0.123
|  (10)Comment_len >= 18.5: -0.229
Legend: -ve = False, +ve = True
```

Table 3. Z10b20 classifier used by the WikiTrust zero-delay vandalism detection tool.

The resulting JSON output is a hash table of key/value pairs, naming the feature and the corresponding value for that feature.

These web APIs will be available as time and resources allow us to maintain this service. We hope that the community will make use of these services and extend this work easily with their own ideas.

References

1. Adler, B., de Alfaro, L.: A content-driven reputation system for the Wikipedia. In: Proc. of the 16th Intl. World Wide Web Conf. (WWW 2007). ACM Press (2007)
2. Adler, B., Chatterjee, K., de Alfaro, L., Faella, M., Pye, I., Raman, V.: Assigning trust to Wikipedia content. In: WikiSym 2008: International Symposium on Wikis. ACM Press (2008)
3. Chatterjee, K., de Alfaro, L., Pye, I.: Robust content-driven reputation. In: Proceedings of AISec 08: First ACM Workshop of AISec. ACM Press (2008)
4. Chin, S.C., Street, W., Srinivasan, P., Eichmann, D.: Detecting Wikipedia vandalism with active learning and statistical language models. In: WICOW '10: Proceedings of the Fourth Workshop on Information Credibility on the Web (2010)
5. Druck, G., Miklau, G., McCallum, A.: Learning to predict the quality of contributions to wikipedia. In: WikiAI'08: Proceedings of the Workshop on Wikipedia and Artificial Intelligence: An Evolving Synergy. pp. 7–12. AAAI Press (2008)
6. Fawcett, T.: ROC graphs: Notes and practical considerations for researchers. Machine Learning 31, 1–38 (2004)
7. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.: The WEKA data mining software: An update. SIGKDD Explorations 11(1) (2009)
8. Itakura, K., Clarke, C.: Using dynamic Markov compression to detect vandalism in the Wikipedia. In: SIGIR'09: Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval. pp. 822–823. ACM Press (2009)
9. Potthast, M.: Crowdsourcing a Wikipedia vandalism corpus. In: Proc. of the 33rd Intl. ACM SIGIR Conf. (SIGIR 2010). ACM Press (2010)
10. Potthast, M., Stein, B., Gerling, R.: Automatic vandalism detection in Wikipedia. In: ECIR'08: Proceedings of the 30th European Conference on IR Research. LNCS, vol. 4956, pp. 663–668. Springer-Verlag (2008)
11. Smets, K., Goethals, B., Verdonk, B.: Automatic vandalism detection in Wikipedia: Towards a machine learning approach. In: WikiAI'08: Proceedings of the Workshop on Wikipedia and Artificial Intelligence: An Evolving Synergy. pp. 43–48. AAAI Press (2008)
12. West, A., Kannan, S., Lee, I.: Detecting Wikipedia vandalism via spatio-temporal analysis of revision metadata. In: EUROSEC'10: Proceedings of the Third European Workshop on System Security. pp. 22–28 (2010)
13. Wikipedia: Version 1.0 editorial team (2010), `http://en.wikipedia.org/wiki/Wikipedia:1`, [Online; accessed 15-July-2010]
14. Wikipedia:flagged revisions (2010), `http://en.wikipedia.org/wiki/Wikipedia:FLAGGED`, [Online; accessed 15-July-2010]
15. Wikiproject vandalism study #1 (2006), `http://en.wikipedia.org/wiki/Wikipedia:WikiProject_Vandalism_studies/Study1`, [Online; accessed 15-July-2010]