Vandalism Detection in Wikipedia: A High-Performing, Feature–Rich Model and its Reduction Through Lasso

Sara Javanmardi
School of Informatics & Computer Science
University of California, Irvine
sjavanma@ics.uci.edu

David W. McDonald
The Information School
University of Washington
dwmc@uw.edu

Cristina V. Lopes
School of Informatics & Computer Science
University of California, Irvine
lopes@ics.uci.edu

ABSTRACT

User generated content (UGC) constitutes a significant fraction of the Web. However, some wiiki–based sites, such as Wikipedia, are so popular that they have become a favorite target of spammers and other vandals. In such popular sites, human vigilance is not enough to combat vandalism, and tools that detect possible vandalism and poor-quality contributions become a necessity. The application of machine learning techniques holds promise for developing efficient online algorithms for better tools to assist users in vandalism detection. We describe an efficient and accurate classifier that performs vandalism detection in UGC sites. We show the results of our classifier in the PAN Wikipedia dataset. We explore the effectiveness of a combination of 66 individual features that produce an AUC of 0.9553 on a test dataset – the best result to our knowledge. Using Lasso optimization we then reduce our feature–rich model to a much smaller and more efficient model of 28 features that performs almost as well – the drop in AUC being only 0.005. We describe how this approach can be generalized to other user generated content systems and describe several applications of this classifier to help users identify potential vandalism.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous

Keywords

Wikipedia, Vandalism Detection, Lasso, Random Forests

1. INTRODUCTION

Wiki technology enabled the emergence of Web sites whose content relies entirely on users' contributions. Such content is usually called User Generated Content (UGC). Wikipedia is one example of such a Web site, and the flagship of UGC. Within UGC systems, vandalism is a well-known problem: individuals sometimes delete or distort the information with malicious intentions; commercial entities sometimes use the popularity of these sites to sell or advertise their products in text disguised as UGC; etc. These various forms of vandalism can occur in many ways with many different motivations. User vigilance can revert vandalism, to some extent, but vandalism can be both massive and subtle, making manual detection a difficult task. Tools that can effectively detect possible vandalism and alert users are important for maintaining value and trust in a UGC resource. These tools serve as a force multiplier making the efforts of individuals who work to maintain UGC more efficient and effective as they focus their energy in places where vandalism is highly likely and subtle.

In this work we describe a low–cost and highly accurate vandalism detection model which makes it practical for real–time applications. While our more general focus is UGC, we develop the model based on a Wikipedia corpus from the "Uncovering Plagiarism, Authorship, and Social Software Misuse (PAN)" workshop. This workshop has been developing corpora and testing algorithms head–to–head since 2007 and thus provides data and benchmarks for comparing our results. By aggregating several features used in the PAN competition and identifying a number of new features, we train a classifier that performs better than most prior results in the PAN competition. Further, we try to compress the model by eliminating redundant features. We do this by estimating the contribution of features to the vandalism detection model based on a MapReduce paradigm, which makes our approach both efficient and scalable. While our specific example is from Wikipedia, the most significant features from the model can be more generally applied to many forms of UGC.

This paper is structured in the following way. We begin with a review of the relevant work on vandalism mitigation from both a user and a technical perspective. Through this we identify a number of persistent challenges for users as well as sets of features that are commonly used to develop technical solutions for vandalism detection. In subsequent sections we elaborate a relevant set of features, use those features to develop a machine classifier that is effective at predicting vandalism. We then apply the Lasso technique to reduce the number of features to a minimum set that achieves high classification performance. In concluding, we argue that the minimum set of features are more generally applicable to UGC systems, and describe some approaches for applying the resulting classifier to help users identify possible vandalism.

WikiSym'11 October 3–5, 2011, Mountain View, CA, USA.

2. BACKGROUND

Vandalism detection has been a concern for Wikipedia since its inception. Vandalism in Wikipedia is defined as any addition, removal, or change of content in a deliberate attempt to compromise the integrity of Wikipedia. According to this broad definition, vandalism can include spamming, lobbying and destroying the edits of others. Wikipedia relies mostly on its human editors and administrators to fight vandalism. But the scale of Wikipedia makes locating all vandalism very time consuming. Tools such as Vandal Fighter, Huggle, and Twinkle are used to monitor recent changes in articles and revert changes deemed vandalism [7]. In the following we describe two broad approaches to vandal fighting, the user approach with relies mostly on tools to assist user detection and the more automated approaches that generally rely on bot or other algorithms.

Viegas *et. al.* [20] conducted some early work on the types of vandalism found in Wikipeida. They used a visualization technique called "history flow" to see the various ways pages were edited and changed over time. In considering these changes they identified five types of vandalism: Mass Deletion, Offensive Copy, Phony Copy, Phony Redirection, and Idiosyncratic Copy. They also analyzed the time for repair of vandalism, which they termed "survival time". They found that the median survival time for some vandalism is quite short, on the order of minutes. However, they noted that the mean time is skewed long, on the order of days or weeks, depending on the type of vandalism. This means there is some vandalism that goes undetected for long periods of time.

In followup work, Priedhorsky *et. al.* [15], considered the impact of a piece of vandalism. That is, if some vandalism lasts on a site for days, how likely is it that a user might stumble across that and be misinformed or otherwise get a wrong impression about the quality of the entire content based on a vandalized page. They developed a model based on page viewing behaviors and vandalism persistence. Their model correlates closely with the empirical results from Viegas *et. al.* [20] and further illustrates that about 11% of vandalism would last beyond 100 page views, with a small fraction of a percent lasting beyond 1000 page views. Further, Priedhorsky *et. al.* [15] also considered the types of vandalism and how likely users were to agree on the types of vandalism. They began with the vandalism types from [20] with some small refinements and identified two additional types: Misinformation, and Partial Delete. They identified the most frequent categories of vandalism as Nonsense (Phony Copy) 53%, Offensive 28%, Misinformation 20% and Partial Delete 14%. However, they also noted that the rate of agreement among their users for some categories is somewhat low with Misinformation having the lowest level of agreement for these top four categories. This means that some of the most subtle vandalism, in the form of Misinformation, is even hard for users to agree upon.

In recent work, Geiger & Ribes [7] studied the process of editors who participate in Recent Changes Patrolling using Huggle. Their study raises some interesting and relevant issues about the work to remove vandalism and how it is performed, illustrating how Wikipedians consider the activities of others. In the case of the decision to ban a vandal, the effort is to understand whether the activities are intentionally designed to corrupt content or wreak havoc on the community itself. This work illustrates that there is a fair amount of vandalism which is somewhat 'routine' but that some is still difficult to detect even by people who are practiced at looking for vandalism. Another interesting insight is that most tools that support vandalism rollback do not yet categorize or provide a prediction rating for the edit being viewed. Most tools simply show the edit, the prior version, and the IP or username of the editor who made it.

The second approach relies on automated bot and algorithms. Researchers have been very supportive of this approach, so we focus on the prior work that is related to how we have also approached the problem. Since 2007 automated bots have been widely used to fight vandalism in Wikipedia. The most prominent of them are ClueBot and VoABotII. Like many vandalism detection tools, they use lists of regular expressions and consult databases with blocked users or IP addresses to keep legitimate edits free of vandalism. The major drawback of these approaches is that most bots utilize static lists of obscenities and grammatical rules that are hard to maintain and, with some creativity, can be easily thwarted. A study on performance analysis of these bots in Wikipedia shows that they can detect about 30% of the instances of vandalism [16].

Several machine learning approaches have recently been proposed that would improve vandalism detection [16, 14, 9, 5]. Comparing the performance of these approaches is difficult because of several shortcomings in early vandalism detection corpora. These early corpora were too small, they failed to account for the true distribution of vandalism among all edits, and the hand labeling of the examples had not been double–checked by different annotators. These shortcomings were resolved by the creation of a large–scale corpus for the PAN 2010 competition consisting of a week's worth of Wikipedia edits (PAN–WVC–10) [13].

The PAN 2010 corpus is comprised of $32,452$ edits on $28,468$ different articles. It was annotated by 753 annotators recruited from Amazon's Mechanical Turk, who cast more than $190,000$ votes so that each edit has been reviewed by at least three of them. The annotator agreement was analyzed in order to determine whether each edit is a regular edit or vandalism, with $2,391$ deemed to be vandalism. The corpus is split into a training set and a test set, which have $15,000$ and $18,000$ edits, respectively [11].

A survey of detecting approaches Potthast & Stein [11] shows that about 50 features were used by the 12 different teams. Features are categorized into two broad groups: (1) edit textual features; (2) edit meta information features. Edit textual features are extracted based on the text of the edit. Some features in this category are adopted from previous work on spam detection in emails or blogs. For example, "Longest character sequence" and "upper case to low case char ratio" are known to be important features for spam detection in emails [8]. The first winner of the PAN competition mainly used features in this category.

Edit meta information mainly contains two types of features: user features and comment features. Most teams used comment features, but only two teams extensively relied on user features. User features are extracted based on the edit patterns done by the user. Comment features are extracted based on the comment related to an edit.

Traditional spam detection systems for emails and blogs mainly rely mainly on textual features based on the content. There is no notion of history logs in these UGC platforms. However, in wikis we have history revisions which make them a valuable source for feature extraction. Interestingly, two

teams captured this difference and extracted some user features based on the history revisions. This helped them place second and third in the competition.

Other approaches rely more heavily on a model of user reputation. Adler *et.al.* [1] used WikiTrust to estimate user reputation. In their system, users gain reputation when their edits are preserved and they lose reputation when their edits are reverted or undone [2].

In our previous work, we used user reputation features based on a reputation management system we developed [10]. Compared to [2], our model is simpler and more efficient. One reason is that our model is only based on the stability of inserts. In [2], stability of deletes and reverts are also considered. A detailed comparison between these two approaches is presented in [10].

Potthast *et.al.* [11] combined the predictions submitted by the top eight teams and developed a meta classifier based on the predictions. To learn the meta classifier, they used random forest. This improved the classification performance (ROC–AUC) of the top team significantly, from 0.91580 to 0.9569. Using a similar approach, Adler *et.al* [17] developed a meta classifier based on the predictions of three different classifiers. To evaluate the meta classifier, they merged train set and test set and reported ROC–AUC based on 10–fold cross validation. Hence, their results are not comparable with PAN competition results and our results.

In general, meta classifiers work at a macro level by aggregating results from different classifiers. In addition, this makes them even more complex and less practical for real–time applications. In contrast, in this study we work at the level of individual features and focus on building accurate classifiers with minimum set of features. We show that the classification performance of the compact classifier is comparable to the meta classifier developed in [11].

3. FEATURE EXTRACTION

To extract features, we mine the entire English Wikipedia history dump, released on Jan, 2010. Totally, 41 edits in PAN corpus are missing in the dump. We use crawler4j[1] to extract the data of these missing edits. We also use other Wikipedia SQL dumps to extract users with special access rights such as administrators and bureaucrats.

Using all this data, we extract 66 features. This feature set includes most of the features used in the PAN competition by different teams [11]. In addition, we introduce some new features. Table 1 shows the features along with their definitions (each row may represent more than one feature).

Similar to [11] we separate edit textual features and edit meta data features. Since the edit meta data features contains both user and comment features, we consider them as different groups. In addition, we add language model features as a new group. These features capture topical relevance and are estimated based on the language model of the newly submitted revision and the background. The effectiveness of using these features for vandalism detection has been studied in [5, 12]. We categorize the features into four groups (see Table 1):

- **User Features**: In this work we introduce 12 features for each user including statistical and aggregate features. We calculate these features by mining history revisions up to time T [10]. For the purpose of this study, we consider T as 2009–11–18 which is the timestamp of the earliest edit log in the PAN corpus. Hence, all features in this category are based solely on history data.

- **Textual Features**: we have 30 features in this category. Most of the features in the category are adopted from [19]. We calculate the value of the features based on the inserted content. In addition, in this work, we calculate the value of the features also based on the deleted content. To distinguish these features we use "Ins" and "Del" prefix throughout this paper. For example, *Vulgarism* shows the frequency of vulgar words. We expect insertion of vulgar words to be a signal for vandalism. Conversely, we expect deletion of such words to be a signal for legitimate edits aiming at removing vandalism.

- **Meta Data Features**: we have 22 features in this category. Most features are extracted from the comments associated with the edits. For example, we have similar textual features to the previous category but here we extract them based on the comment. Because the descriptions are similar to their peer textual features, we do not define them in Table 1. These features are specified by *.

  In addition to these, we introduce some new features that we extract from the automatic comments generated by Wikipedia. These comments specify which section of the article has been edited. We extract unigram, bigrams, and trigrams from these types of comments. We use feature selection on the PAN train set to extract the important ones. For example, the short time interval between the old and the new revisions might be an indicator of vandalism.

- **Language Model Features**:
  In this category we have 3 features which calculate the *Kullback–Leibler* distance (KLD) between two unigram language models. We calculate KLD between the previous and the new revision [5]. We introduce two more features: the KLD between the inserted content and the previous revision. Similarly, we calculate the KLD between the deleted content and the previous revision. We suspect that, sometimes vandalism comes with some unexpected words so we expect to see sharp changes in the distance. Conversely, deleting unexpected words can be an indicator of legitimate edits.

4. LEARNING VANDALISM DETECTION MODEL

We consider vandalism detection as a binary classification problem. We map each edit in PAN corpus into a feature vector and learn the labels by mapping feature vectors onto {0,1}, where 0 denotes legitimate edit, and 1 vandalistic edit. To learn a classifier and tune its free parameters, we use the PAN train set and leave the test set untouched for final evaluation.

Statistical analysis of Wikipedia edits show that roughly 7% of edits are vandalistic [10], which is consistent with the vandalism ratio in the PAN corpus. Given this, we need to use machine learning algorithms that are robust enough to

[1]http://code.google.com/p/crawler4j/

Table 1: List of features. Asterisked features are extracted based on both the edit content and its comment.

Feature	Description
DSR, DDSR, Rep	Aggregated features representing a user's reputation
Ins Words	Total number of words inserted by a user
Del Words	Total number of words deleted by a user
Lost Words	Total number of deleted words from a user
Ins Revision	Total number of revisions a user has done insertion in
Del Revision	Total number of revisions a user has done deletion in
Ins Page	Total number of pages a user has done insertion in
Del Page	Total number of pages a user has done deletion in
User Type	User has some special rights, such an admin, a bot, or a bureaucrat
User Page	User has a user page in Wikipedia
Ins Size	Number of inserted words
Del Size	Number of deleted words
Revision Size	Size difference ratio between the old and the new revision.
Blanking	The whole article has been deleted
Internal Links	Number of links added to Wikipedia articles
External Links	Number of added external links
Word Repetitions	Length of the longest word
Char Repetitions	Length of the longest repeated char sequence
Compressibility	Compression rate of the edit differences.
Capitalization*	Ratio of upper case chars to lower case chars
Capitalization All*	Ratio of upper case chars to all chars
Digits*	Ratio of digits to all letters
Special Chars*	Ratio of non-alphanumeric chars to all chars
Diversity*	Length of all inserted lines to the (1 / number of different chars)
Inserted Words*	Average term frequency of inserted words
Vulgarism*	Frequency of vulgar words
Bias*	Frequency (impact) of biased words
Sex*	Frequency (impact) of sex related words
Spam*	Frequency (impact) of spam related words
Pronouns*	Frequency (impact) of personal pronouns
WP*	Frequency (impact) of mark up related words
Special Words*	Aggregation of vulgarism, bias, sex, spam, pronouns, and WP ratios
Time Diff	Time interval between the submission of the old and the new revision
Category	If the automatic comment contains "category"
Early Years	If the automatic comment contains "early years"
Copyedit	If the automatic comment contains "copyedit"
Personal Life	If the automatic comment contains "personal life"
Revert	If the automatic comment contains "revert"
Revision Ordinal	Ordinal of the submitted revision
Length	Length of the comment
Reverted	if the MD5 digest of new revisions is the same as one of the old ones in window size of 10
KL Distance	Kullback–Leibler distance between the old revision and the new revision
KL Distance Ins	Kullback–Leibler distance between the inserted words and the new revision
KL Distance Del	Kullback–Leibler distance between the deleted words and the new revision

handle class imbalance problems. In addition, we need to use an evaluation metric that is insensitive to imbalanced data. Similar to the evaluation metric used for PAN competition, we use AUC, area under the ROC curve.

We use different binary classification algorithms such as Logistic Regression, Naive Bayes, and Random Forest [3]. In all of our experiments, random forest results in the best classification performance. Random forests are known to be tolerant to imbalanced datasets. In addition, they are a suitable option for the datasets with missing data [3]. The PAN data set is an imbalanced dataset and for some features, like user group features have the problem of missing data. For 4% of users we do not have any information. Thus, in this paper we report the results based on random forest classifiers.

To learn a random forest classifier, we need to tune two free parameters: the number of trees in the model and the number of features selected to split each node. Our experiments show that classification performance is sensitive to the former but not to the latter. This result is consistent with Breiman's observation [3] on the insensitivity of random forests to the number of features selected in each split.

To tune the number of trees, we partition the train set into three folds and use 3–fold cross validation. Using three folds allows us to keep a relatively decent number of vandalized cases in each training sets (around 600). To find the optimal value for the number of trees, we need to sweep a large range of numbers. Hence, we need to design an efficient process for this purpose.

For each fold, we create a pool of $N = 10,000$ trees, each trained on a random sample of the training data in that fold. Then we use this pool for creating random forests of different sizes. For example, to create a random forest with 20 trees, we randomly select 20 trees from this pool of N trees. However, since this random selection can be done in $C(N, 20)$ different ways, each combination may result in a different AUC. We repeat the random selection of trees $r = 50$ times and we report the mean and variance of the $F \times r$ results (where F is the number of folds).

The advantage of this approach is that we can calculate the mean and variance of AUC very efficiently for forests with different sizes without the need to train a huge number of trees independently. Otherwise, to report the mean and variance of AUC for random forests of size $k = 1$ *to* T, we would need to train $r+2\times r+3\times r+...+T\times r = r*T(T+1)/2$ trees for each fold, which is 10^8 trees. Using this approach we only need to train N trees per fold (in our experiments we used $N = 5 \times T$).

Figure 1 shows the mean of AUC as a function of number of trees in the model. As more trees are added to the model, mean of AUC increases and the variance decreases. The mean of AUC does not improve significantly after having 500 trees in the forest but the variance keeps decreasing. It should be emphasized that models with smaller variance are more stable and therefore more predictable in test environments. Although more trees may result in slightly higher AUC values, we decide to set the number of trees as 1000 to have a balance between classification performance and model complexity. More complex models with more trees would require more time for prediction in real–time. Given this, the AUC on the train set based on 3–fold cross validation is 0.9739 ± 0.0024. The AUC value on the PAN test set is **0.9553**.

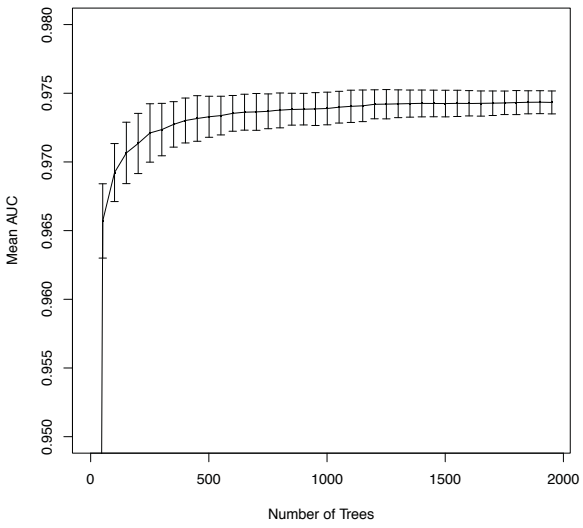

Figure 1: The effect of number of trees on AUC mean and standard deviation

The result reported here is significantly higher than the best AUC reported in the PAN competition, 0.9218 [11].

5. LOW–COST VANDALISM DETECTION

The classifier mentioned in the previous section makes its decision based on 66 features in four different groups. However, computing and updating each of these features imposes some off–line or run–time cost. For example, computing and updating features in the user group requires the tracking of all edits done by each individual. Maintaining a system that updates data for computing these features would come at some cost for the wiki. Some other features like textual features are computed after submission of a new edit and the vandalism detection system should be able to compute them in real–time to decide whether it is legitimate or vandalistic.

In this section, we report the results of our experiments in finding a minimum set of features whose classification performance is like the one with 66 features. In other words, we try to detect and eliminate redundant features. There are two types of redundant features: (a) features that are not informative and do not help in discriminating legitimate and vandalistic content; (b) features correlated with some other features so that once one of them is selected, adding others does not bring any new discriminating power to the model.

In order to detect and eliminate redundant features, we perform two sets of experiments. First, in Section 5.1 we study the contribution of each group of features as a whole unit to examine if any of the four groups can be eliminated without a significant drop in AUC. Then in Section 5.2 we study the contribution of each feature individually and use the results for eliminating redundant features.

Given the number of experiments needed for this study, we use the Amazon MapReduce cluster to run them in parallel. In our implementation, each mapper receives specific config information and trains a classifier for that config. Then

Table 2: The drop in mean AUC when we eliminate a feature group. Results on train set are for 3–fold cross validation.

Dropped Group	Train set	Test set
User	−3.8%	−6.6%
Textual	−1.8%	−1.5%
Meta data	−0.4%	−0.3%
Language Model	−0.2%	−0.3%

Table 3: The mean AUC when we only use features in one group.

Selected Group	Train set	Test set
User	0.9399	0.9225
Textual	0.9001	0.8400
Meta data	0.7019	0.6924
LM	0.7271	0.6986
Baseline (all groups)	0.9739	0.9553

reducers aggregate the AUC results for different folds and report the mean AUC for different configs.

5.1 Eliminating Groups of Features

For each group of features, we train a classifier without the features of that group. This will show us the drop in AUC when this group is ignored. Table 2 shows the results. According to these results we can conclude that removing features in the User group and the Textual group results in a significant drop in AUC, while the drop for the Meta data and the LM group is much less.

Based on these results we cannot infer that features in the Meta data and LM group are not informative. The only conclusion is that having both User and Textual features in our feature set, adding Meta data and LM features do not add significant new discriminating power to the model. Interestingly, Table 3 supports this conclusion. When we only use Meta data or LM features the AUC value is much higher that the AUC of a random classifier (0.50). Another interesting result is that when we only keep features of one group, the User group results in the highest AUC (0.9399) .

The results of this study might not seem consistent with what was reported in [17]. Here we show the importance of user features, while in [17], authors have concluded that there is much less need to care about the past performance of the users and therefore that user features do not contribute in any significant way to the classification performance. We see two reasons for this difference. First, we calculate user features based on different approaches; so our feature sets are not the same. Secondly, the authors in [17] have used meta classifiers to measure the contributions of groups of features to the classification performance.

It is important to note that a meta classifier works at the macro level. In other words, it sends the same test instance to a set of classifiers and then aggregates their decisions (*e.g*, using a weighted average). In many problems this approach produces more accurate results [4] because now the final decision is made based on the votes casted by a diverse set of classifiers. However, training different classifiers on different groups of features and then combining them with a meta classifier, as suggested in [17], may not necessarily result in an optimal classification performance. The reason for this is that, by limiting each single classifier to a subset of features, we may limit its expressiveness, thereby making it weaker. A combination of such classifiers may only result in a sub–optimal performance; but a classifier that has access to all of the features in different groups, has the chance of making its decision based on a combination of features in different groups. For example, a decision tree based classifier may grow branches in which both user reputation features and textual features are used. Therefore, it can exploit all of the useful information embedded in these features to enhance the result.

5.2 Eliminating Individual Features

In Section 5.1 we showed that all the four groups have some informative features. In this section, we attempt to find the smallest feature set whose AUC is comparable to the AUC of a classifier with 66 features. To this aim we do feature selection.

Traditional feature selection algorithms like Information Gain or Chi–Square evaluate features independently in order to estimate their importance. However, we need an algorithm that considers correlations between features and is able to detect and eliminate redundant features effectively. We use Lasso (Least Absolute Shrinkage and Selection Operator) [18] for this purpose. Lasso fits a regularized regression model to features in such a way that the final model is a sparse solution in the feature space. Thus, the weight of redundant features in the final model would be zero. This means that we can remove these features from the model with no significant change in the classification performance. We use the Logistic Regression Lasso implemented in R glmnet package [6].

Lasso has a regularizer parameter, λ, that offers a tradeoff between sparsity of the model and its classification performance. Lower values for λ result in more relaxation of the regularization constraint which allows more features to have non–zero weights.

Table 4 shows a correlation between the selected features and different values of λ. For $\lambda = 0.0716$, only one feature is selected. This means that, according to Lasso, if we want to decide the legitimacy of an edit based on only one feature, "Ins Special Words" would be our best choice. As we decrease the value of λ, more features are selected according to their importance. The second most important feature is "DDSR". The last column in Table 4 shows the value of the AUC on the PAN test set for classifiers trained on the selected features. As the number of selected features increases, we see a higher value for AUC but the cost of computing and updating the features would also increase.

We use 3–fold cross validation on the PAN train set to pick the largest value for λ, where the drop in AUC is not statistically significant. The result was $\lambda = 0.0030$ which leads to selection of 28 features. Table 4 shows a list of the selected features. The AUC for this feature set on the PAN test set is 0.9505. This feature set only includes less than half of the original features but the drop in AUC is only 0.005.

In this sparse feature set we have features from all groups.

Table 4: Feature Selection using Lasso. Parameter λ determines a tradeoff between number of selected features and AUC of the classifier. Smaller values of λ allow more features be selected and result in models with higher performance.

λ	Selected Features	AUC on the PAN test set
0.0716	Ins Special Words	0.5974
0.0594	DDSR, Ins Special Words	0.8614
0.0373	DDSR, Rep, Ins Special Words	0.8965
0.0340	DDSR, Rep, User Page, Ins Digits, Ins Special Words	0.9074
0.0310	DDSR, Rep, User Page, Ins Digits, Ins Vulgarism, Ins Special Words	0.9090
0.0257	DDSR, User Page, KLDNew2OLD, Ins Digits, Ins Vulgarism, Ins Special Words	0.9197
⋮	⋮	⋮
0.0030	DDSR, Del Words, User Type, User Page, Copyedit, Personal Life, Revision Ordinal, Comment Length, KLDNew2OLD, Blanking, Ins Internal Link, Ins External Link, Longest Inserted Word, Ins Longest Character Sequence, Ins Compressibility, Ins Capitalization, Ins Digits, Ins Special Chars, Ins Vulgarism, Ins Bias, Ins Sex, Ins Pronouns, Ins WP, Ins Special Words, Del Bias, Ins Digits, Comment Special Chars Comment Spam	0.9505
⋮	⋮	⋮

For example, "DDSR", "Del Words", "User Type", and "User Page" are selected from the user group. If we follow the Lasso path, features get added and eliminated as λ decreases. For example, "Rep" is selected as the third important feature, but because of its correlation with other selected features it is eliminated from the feature set later. Interestingly, "Rep" is computationally more expensive than "DDSR".

We have 17 features from the Textual group. These features are also widely used for spam detection in other domains such as emails or blogs [8]. For example, "Ins Longest Character Sequence" shows whether a user has inserted a long sequence of the same character which can be a good indicator of vandalism. There are also some textual features which are unique to Wikipedia. For example, "Del bias" shows that the user has deleted words which represent bias and therefore is a good indicator of legitimate edit.

We have 6 features selected from the Meta data group. For example, "Comment Length" or "Comment Special Chars" are selected as important features. "Personal Life" is another important feature. It shows whether the edit is made in the "Personal Life" section of a biography article. We have observed that this section of biography articles is more often vandalized and therefore this feature can be an important signal. This observation is consistent with Wikipedia statistics which show a high vandalism ratio in biography articles. Given that Wikipedia automatically adds the name of an edited section to the comment associated to each edit, we can extract this feature from comments.

The only feature that is selected from the Language Model group is "KLDNew2OLD". The goal of this feature is to detect sharp linguistic distance between the new and the previous revision which can be an important signal for vandalism detection.

6. DISCUSSION & CONCLUSION

In the preceding sections we have described an approach to the important task of detecting vandalism in User Generated Content (UGC) systems. Our specific example comes from Wikipedia, but the technical approach can be applied more generally to many forms of UGC.

Using a validated corpus of Wikipedia edits from the PAN competition we developed binary classifiers using random forests. Random forests are strong indicators of missing and unbalanced data which is a common characteristic for datasets from wikis and other forms of UGC. We compressed the learning model using a Lasso statistical model shrinkage technique. We found a smaller feature set, for which the computing and classifying is inexpensive and thus very practical for the online/real–time application of the classifier.

In fact, it is through this smaller, compact set of features that we claim a broader application to a wider range of UGC systems. User reputation features in the form of DDSR and special characters in the edit are some of the strongest features and are easily detected in other UGC systems. The usage of special characters has been widely practiced in spam detection in emails or blog comments. DDSR measures the survivability of the content contributed by users. This survivability is interpretable in other domains. For example, to measure it in Twitter we can see how often and how fast a tweet gets re–tweeted. Similarly, in Facebook, we can look at patterns of sharing and propagation.

UGC systems can apply vandalism detection to help users understand which content may be problematic. Colorizing text is a common interface mechanism to indicate trust. With our compact model, we know which features contribute most to the prediction of vandalism and could colorize or annotate the content to indicate the strength of the prediction.

Another variant of this technique is to help end–users detect and remove vandalism. Few user tools for vandalism removal provide any suggestion or prediction of that is or

is not vandalism. This is not a complete oversight on the part of the tool designers. Some of the predictive models are difficult to maintain and the predictions cannot be computed in real–time, as the tool loads a specific contribution. The model that we develop could meet these requirements, but admittedly, we have not tested this in an actual vandal fighting tool. Instead, we have left that for some future work.

A third application of this technique is to use it as the basis for an awareness tool or event notification tool. Most wikis support some form of a watch list. When a user puts a specific wiki page on her watch list she is indicating interest in that page. When that page is edited or changed, the system sends email to the user about the edit. This is not a problem if a user is interested in a few pages, but if she were interested in many pages this might result in an unmanagable flood of email. An enhanced watch list mechanism could use our model and a user specified vandalism threshold to control notifications. That is, for some pages, a user might want to see every change (low vandalism threshold), while for others she might want to see only changes that cross a high threshold of predicted vandalism. This would allow a user to monitor a much broader span of pages while focusing her attention on pages or on activities that are of more interest to her.

Vandalism detection is a difficult problem. Technical approaches like ours are making progress. However, some user studies attempting to classify vandalism have illustrated that even among users there are types of vandalism that are open to interpretation [15, 20]. In future work we plan to explore the types of vandalism as a multi–label classification problem. We have used Amazon Mechanical Turk to begin re–labeling the PAN corpus using categories derived from [15] and [20]. Our plan is to move beyond a standard binary classification (vandalism/legitimate) for each content edit to explore where users agree or disagree on vandalism and technical approaches that could help users understand how and why content might, or might not be, vandalism.

Vandalism will continue to be a challenge for all types of UGC systems. What constitutes vandalism will always be something "in the eye of the beholder". As such, the community of users contributing to UGC systems will always be the final arbiters of what should stay and what should go. Tools based on models like ours need to be timely and provide a clear and rational explanation for why a prediction is being made, so that the user can make the best decision possible with the information we provide.

7. ACKNOWLEDGEMENT

The authors would like to thank Prof. Robert Tibshirani for his comments on using Lasso, Thomas Debeauvais for his help on running experiments, Santiago M. Mola–Velasco for his feedback on implementing textual features, and Martin Potthast for his support. In addition, the authors would like to thank Amazon for a research grant that allowed us to use their MapReduce cluster. This work has been also partially supported by NSF grant OCI–074806.

8. REFERENCES

[1] B. Adler, L. de Alfaro, and I. Pve. Detecting wikipedia vandalism using wikitrust. Technical report, PAN lab report, CLEF (Conference on Multilingual and Multimodal Information Access Evaluation), 2010.

[2] B. T. Adler and L. de Alfaro. A content-driven reputation system for the wikipedia. In *WWW '07: Proceedings of the 16th International Conference on World Wide Web*, pages 261–270, New York, NY, USA, 2007. ACM.

[3] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[4] R. Caruana and A. Niculescu-Mizil. An empirical comparison of supervised learning algorithms. In *Proceedings of the 23rd international conference on Machine learning*, ICML '06, pages 161–168, New York, NY, USA, 2006. ACM.

[5] S. chi Chin, P. Srinivasan, W. N. Street, and D. Eichmann. Detecting wikipedia vandalism with active learning and statistical language models. In *Fourth Workshop on Information Credibility on the Web (WICOW 2010)*, 2010.

[6] J. H. Friedman, T. Hastie, and R. Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33(1):1–22, 2010.

[7] R. S. Geiger and D. Ribes. The work of sustaining order in wikipedia: the banning of a vandal. In *Proceedings of the 2010 ACM conference on Computer supported cooperative work*, CSCW '10, pages 117–126, New York, NY, USA, 2010. ACM.

[8] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., New York, NY, USA, 2001.

[9] K. Y. Itakura and C. L. A. Clarke. Using dynamic markov compression to detect vandalism in the wikipedia. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '09, pages 822–823, New York, NY, USA, 2009. ACM.

[10] S. Javanmardi, C. Lopes, and P. Baldi. Modeling user reputation in wikipedia. *Journal of Statistical Analysis and Data Mining*, 3(2):126–139, 2010.

[11] T. H. Martin Potthast, Benno Stein. Overview of the 1st international competition on wikipedia. In *CLEF'2010*, September 2010.

[12] G. Mishne, D. Carmel, and R. Lempel. Blocking blog spam with language model disagreement. In *AIRWeb*, pages 1–6, 2005.

[13] M. Potthast. Crowdsourcing a wikipedia vandalism corpus. In *Proceeding of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '10, pages 789–790, New York, NY, USA, 2010. ACM.

[14] M. Potthast, B. Stein, and R. Gerling. Automatic vandalism detection in wikipedia. In C. Macdonald, I. Ounis, V. Plachouras, I. Ruthven, and R. W. White, editors, *ECIR*, volume 4956 of *Lecture Notes in Computer Science*, pages 663–668. Springer, 2008.

[15] R. Priedhorsky, J. Chen, S. Lam, K. Panciera, L. Terveen, and J. Riedl. Creating, destroying, and restoring value in wikipedia. In *GROUP '07: Proceedings of the 2007 International ACM Conference on Supporting group work*, pages 259–268, New York, NY, USA, 2007. ACM.

[16] K. Smets, B. Goethals, and B. Verdonk. Automatic

vandalism detection in wikipedia: Towards a machine learning approach. In *Proceedings of the Association for the Advancement of Artificial Intelligence (AAAI) Workshop on Wikipedia and Artificial Intelligence: An Evolving Synergy (WikiAI08)*, pages 43–48. AAAI Press, 2008.

[17] B. T.Adler, L. de Alfaro, S. M.Mola-Velasco, P. Rosso, and A. G. West. Wikipedia vandalism detection: Combining natural language, metadata, and reputation features. In *Proceedings of Computational Linguistics and Intelligent Text Processing(CICLing'11)*, pages 266–276, 2011.

[18] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, Series B*, 58:267–288, 1994.

[19] S. M. M. Velasco. Wikipedia vandalism detection through machine learning:feature review and new proposals. Technical report, PAN lab report, CLEF (Conference on Multilingual and Multimodal Information Access Evaluation), 2010.

[20] F. B. Viegas, M. Wattenberg, and K. Dave. Studying cooperation and conflict between authors with history flow visualizations. In *CHI '04: Proceedings of the SIGCHI Conference on Human factors in computing systems*, pages 575–582, New York, NY, USA, 2004. ACM.