Detecting Wikipedia Vandalism

Tony Jin, Lynnelle Ye, Hanzhi Zhu

1 Introduction

Since its inception in 2001, Wikipedia has become the largest encyclopedia ever created in human history. With over 4 million articles in the English edition alone, it has become the highest-traffic educational website on the Internet. It receives over 100,000 edits per day, which can be daunting for human editors to monitor for vandalism, spam, or other inappropriate content. While there are existing vandalism reversion bots, they are generally hard-coded and may not be efficient enough at detecting vandalism. Types of vandalism include insertion of obscenities or personal attacks, deletion of valid content, and intentional introduction of incorrect facts (which can be difficult even for a human to detect).

We will experiment with using machine learning techniques to create a vandalism detection bot. We will consider features such as character frequencies, word attributes, attributes of the comment associated with the revision, and the history and attributes of the editor. We will attempt to perform logistic regression and Naive Bayes on these features, and we will also consider training an SVM with them.

2 Literature review

Automatic detection of Wikipedia vandalism has attracted significant attention in the literature, though generally at a fairly preliminary level. Naive Bayes and SVMs have been applied with some success in [1], [2], [5], and [6]. Potthast, Stein, and Gerling in [5] and Mola-Velasco in [6] have compiled lists of features that are likely to be useful. We have combined ideas from their lists below.

- **Character frequencies.** Character distributions which deviate from expectation may indicate the insertion of nonsense. Long sequences of identical characters and character strings with unusual compressibility are similarly indicative. Other potential features include the proportion of upper-case characters (vandals are more likely to use all-capitals or all-lowercase), the proportion of digits or non-alphanumeric characters, character diversity in general, and so on.

- **Word attributes.** Unusually long words may signal nonsense, or unusual word lengths in general. The frequency of obscene words in the edit is also an obvious choice of feature, though it may be necessary here to distinguish between frequency and impact, the latter being the percent by which the edit increases the proportion of obscenity in the entire article. This is just because some Wikipedia articles, for example the ones about the words themselves, may legitimately require use of obscenity in edits.

  In a similar vein, one may wish to count first and second pronouns, colloquialisms, misspellings, and wiki formatting elements (which vandals are unlikely to use).

  More generally, comparing the words in the revision to those in the old version and to those in the article overall is likely to be useful, since the insertion of a large number of words which do not appear elsewhere in a long, well-established article suggests that the new content is irrelevant.

- **Overall edit attributes.** This includes, say, the change in size between the old and new texts. For example, large deletions are often malicious ones.

- **Comment attributes.** Long comments tend to be associated with regular editing, though short or empty comments are also common practice with well-intentioned edits. It is also possible to apply the same process to comments as to edit text, looking for nonsense or obscenity, though these are probably much weaker features in this context, since there is no real incentive for vandals to insert junk text into comments that are not part of the page.

- **Editor attributes.** Anonymity is a strong indicator of vandalism. One may also be interested in counting the number of past edits by the same user, and perhaps measuring the quality of past edits by how often they were quickly reverted.

We will not attempt to use all of these features, since some should fall out of our algorithms, some are highly correlated to the point of redundancy, and some are difficult to obtain or analyze.

3 The data

3.1 Source

We draw our data from PAN-WVC-10, created by Potthast ([4]). It consists of 32452 revisions on 28468 Wikipedia articles, each flagged as well-intentioned or malicious by a majority of voters on Amazon's Mechanical Turk. The malicious edits make up 2394 of the revisions. This corpus is slightly too large for us and its very unbalanced distribution also makes evaluating accuracy more difficult, so we pull out two sub-datasets for some smaller experiments. The first sub-dataset consists of 700 vandalizing and 700 non-vandalizing edits which are scrambled randomly before being split into training and testing sets. The second sub-dataset consists of all 2394 vandalizing edits together with a matching 2394 non-vandalizing edits which are scrambled randomly before being split into training and testing sets. We will refer to these as edits-1400 and edits-4788.

While the fact that edits-1400 and edits-4788 contain a much different distribution of vandalism from the real-life situation could harm the performance of our classifier, in practice our algorithms seem to behave similarly no matter what the distribution is, as long as it is the same between the training and testing sets.

3.2 Processing

The corpus records the Wikipedia page from before and after each revision. For each revision, we convert all words into lowercase letters and store for each word the difference between the number of times it appears in the new version from the number of times it appears in the old version. We will sometimes refer to this as a "frequency count" of the word in the document, but it is important to remember that it can be negative if the number of appearances of the word is reduced by the edit.

We also compute the following four non-word features: the length of the edit comment, the amount by which the edit changed the length of the article (this can be positive or negative), whether the editor was anonymous, and the fraction of those words added by the edit which already appeared in the original article.

4 The experiments

We now describe the algorithms we implemented and their results. It is important to note that all our experiments on edits-1400 were performed under seven-fold hold-out cross-validation, with 200 examples per fold. All our experiments on edits-4788 and the full dataset were performed under eight-fold hold-out cross-validation, with 300 examples per fold in the case of edits-4788 and roughly 4500 in the case of the full dataset.

4.1 Feature selection

Although we could use the entire vocabulary for the word features, this results in too much data and the words which are very infrequent do not improve the classifier. We thus wish to only look at words which appear with some moderately high frequency. We ended up with three ways of ranking the words. First, we simply took the most frequent words in all of the edits, both vandalizing and constructive, and removed stopwords (the top ten words). We call this feature group A. Second, we took the top 1000 most frequent words in each class, and only used as features those which appear in only one of the top thousand lists. We call this feature group B. Third, we took those words with the highest positive difference in frequency between vandalizing and constructive edits. We call this feature group C.

4.2 Naive Bayes implementation and results

We performed a standard implementation of Naive Bayes on our training set. To prepare the data for Naive Bayes, we extracted feature group B and replaced all negative frequency counts with a count of 0. Though this is of course a significant convenience, we believe it is also theoretically justified. This is simply because we do not generally expect the content of the old version of a document to inform our judgment about the legitimacy of a particular edit *on its own*, independently from how it compares to the new version. There is no particular reason a certain word should be removed far more often by vandals than non-vandals, while there are many reasons it might be added far more often by vandals than non-vandals. (Actually, this is probably not strictly true. It is likely that the removal of some words corresponds to politically- or ideologically-motivated vandalism. But we will assume that this is a fairly rare or at least localized occurrence which can be addressed through other means.)

Tabel 1 gives the performance of Naive Bayes under eight-fold cross-validation using various numbers of the top word features. Note that as the number of features increases, accuracy and recall increase somewhat while precision essentially doesn't change. Of course, the most important point to note here is that these results are all much better than chance.

Number of features	Overall accuracy	Precision	Recall
100	0.6696	0.8001	0.4368
200	0.6904	0.8083	0.4843
450	0.7092	0.8093	0.5339
900	0.7137	0.8027	0.5548

Table 1: Performance of Naive Bayes with various levels of feature selection.

4.3 Support Vector Machine implementation and results

We ran LIBLINEAR on several different versions of our training set, with wildly varying and sometimes rather strange results. On edits-1400, with no preprocessing to remove negative frequencies (since it is no difficulty for the SVM to handle them and they cannot hurt), we get very reasonable accuracies in the $60-70\%$ range, increasing somewhat as the number of features increases, as shown in Figure 1. The features are drawn from feature group C.

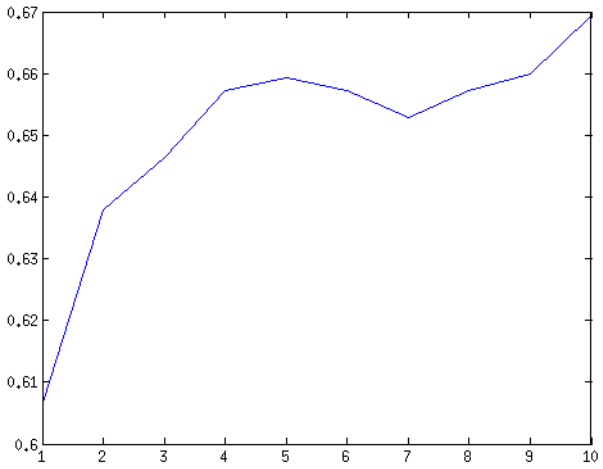

Figure 1: Accuracy vs. number of features for LIBLINEAR. The horizontal axis is in hundreds of features used.

The precision-recall curve for varying numbers of features is shown in Figure 2. Note that in general,

LIBLINEAR precision is much higher than recall, with numbers around 80% and 30% respectively. This is generally good news in our context, because as part of an army of vandal-fighting bots and humans, it is far more important for a particular bot to avoid deleting legitimate edits than for it to find all the illegitimate ones.

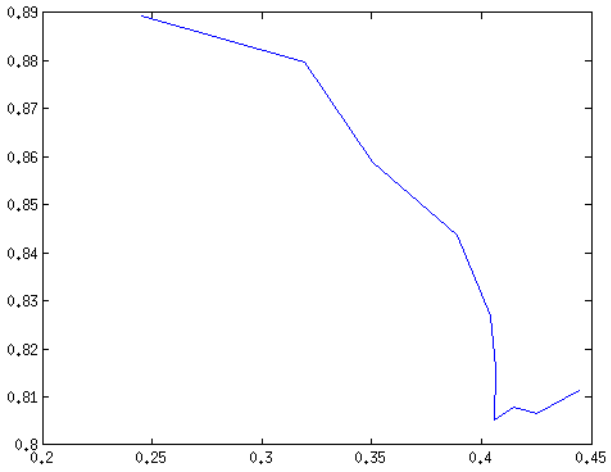

Figure 2: Precision-recall curve for LIBLINEAR, with the number of features ranging from 100 to 1000.

Unfortunately, we were not able to replicate these results on edits-4788. Table 2 shows the behavior of LIBLINEAR on edits-4788.

Number of features	Overall accuracy	Precision	Recall
100	6117	0.6651	0.523
200	0.62	0.6476	0.4909
450	0.6038	0.584	0.7796
900	0.6521	0.6836	0.6107
1000	0.68	0.6744	0.7073

Table 2: Performance of LIBLINEAR with various levels of feature selection.

We see that while LIBLINEAR continues to do significantly better than chance in all cases, its specific performance as the number of features change is a little mysterious.

We also tried attaching the four non-word features described in Section 3.2 to the training data for LIBLINEAR, together with our 1000 word features. The result on edits-4788 was an accuracy of 69.87% with 68.95% precision and 72.37% recall. This is a little better than the numbers in Tabel 2, but not very much.

4.4 Logistic regression implementation and results

Finally, we implemented logistic regression using solely the four non-word features described in Section 3.2, reasoning that the dimensional blow-up caused by including the words themselves would slow down the algorithm to impractical levels. Surprisingly, this gave us our most consistently good results. On edits-4788, the regression had an accuracy rate of 79.42% with a precision of 75.94% and a recall of 85.92%. The success of logistic regression can be attributed to the usefulness of the features in predicting vandalism, in particular whether the user is anonymous (logged-in users are highly unlikely to be vandals), the length of the comment (many vandals do not both filling in a comment), and the change in document size (removal of large portions of text is probably vandalism).

4.5 Tests on the entire corpus

We then ran Naive Bayes and SVM on the entire corpus with eight-fold cross-validation. Naive Bayes was run on the non-negative frequency counts only. The SVM was run on four different matrices, depending on whether the frequency counts included negative numbers or not, and whether the matrix included the extracted features or was limited to just the frequency counts. The results are summarized below:

Algorithm	Accuracy	Precision	Recall
Naive Bayes	0.9407	0.7336	0.3118
SVM (w/o negative, w/o features)	0.5345	0.1000	0.6645
SVM (w/ negative, w/o features)	0.6934	0.1524	0.6701
SVM (w/o negative, w/ features)	0.9069	0.4332	0.4046
SVM (w/ negative, w/ features)	0.8915	0.3266	0.4048

Table 3: Performance of various algorithms on the entire corpus.

Accuracy, precision, recall are averaged over the 8 folds. It appears that Naive Bayes performs better than the SVM. A possible reason for this is that only 7% of the corpus is vandalism, so the prior in Naive Bayes makes false positives less likely. Additionally, running the SVM over large amounts of data may result in overfitting, as the decision boundary attempts to accomodate every single point.

5 Discussion

We found that Naive Bayes, LIBLINEAR, and logistic regression all classify edits as well-intentioned or malicious with success probability much better than chance on the various subsets of the PAN-WVC-10 corpus we considered. Logistic regression on our four non-word features had the highest overall accuracy rate, but was impractically slow even with a stripped-down dataset and could probably not be used in practice. Whether Naive Bayes or LIBLINEAR did better depended on the specific makeup of the training and testing data. LIBLINEAR did well on a small balanced dataset of 1400 edits, with a predictable small loss in precision and noticeable gain in recall as the number of features went up, but behaved more strangely (though still much better than chance) on larger datasets, perhaps because of overfitting. Naive Bayes remained reasonably well-behaved on all scales of data.

References

[1] Belani, Amit. "Vandalism detection in Wikipedia: a bag-of-words classifier approach." arXiv preprint arXiv:1001.0700 (2010).

[2] Chin, Si-Chi, W. Nick Street, Padmini Srinivasan, and David Eichmann. "Detecting Wikipedia vandalism with active learning and statistical language models." Proceedings of the 4th workshop on Information credibility. ACM, 2010.

[3] Mola-Velasco, Santiago M. "Wikipedia vandalism detection." Proceedings of the 20th international conference companion on World wide web. ACM, 2011.

[4] Potthast, Martin. Crowdsourcing a Wikipedia Vandalism Corpus. In Hsin-Hsi Chen, Efthimis N. Efthimiadis, Jaques Savoy, Fabio Crestani, and Stéphane Marchand-Maillet, editors, 33rd International ACM Conference on Research and Development in Information Retrieval (SIGIR 10), pages 789-790, July 2010. ACM. ISBN 978-1-4503-0153-4.

[5] Potthast, Martin, Benno Stein, and Robert Gerling. "Automatic vandalism detection in Wikipedia." Advances in Information Retrieval (2008): 663-668.

[6] Velasco, Santiago M. Mola. "Wikipedia vandalism detection through machine learning: Feature review and new proposals." Lab Report for PAN-CLEF 2010 (2010).