Novel Balanced Feature Representation for Wikipedia Vandalism Detection Task*

Lab Report for PAN at CLEF 2010

István Hegedűs[1], Róbert Ormándi[1], Richárd Farkas[1], and Márk Jelasity[2]

[1] University of Szeged, Hungary
[2] University of Szeged and Hungarian Academy of Sciences, Hungary
{ihegedus,ormandi,rfarkas,jelasity}@inf.u-szeged.hu

Abstract In online communities, like Wikipedia, where content edition is available for every visitor users who deliberately make incorrect, vandal comments are sure to turn up. In this paper we propose a strong feature set and a method that can handle this problem and automatically decide whether an edit is a vandal contribution or not. We present a new feature set that is a balanced and extended version of the well known Vector Space Model (VSM) and show that this representation outperforms the original VSM and its attribute selected version as well. Moreover, we describe other features that we used in our vandalism detection system and a parameter estimation method for a weighted voting metaclassifier.

1 Introduction

Nowadays, Wikipedia is one of the most relevant sources of encyclopedic knowledge. Although it usually provides high quality, relevant information, users more and more often obtain articles which contain false data or even spammed content. Detecting of this type of content manually is a time-consuming and maybe impossible task due to e.g. the size of Wikipedia. For this reasons, it is crucial to support the task of ”keeping Wikipedia clear“ by automatic or semi-automatic methods. For automatic detection we can use approaches from the field of Natural Language Processing (NLP) since this problem – in an abstract form – can be viewed as a text categorization task where each Wikipedia edit has to be classified as a regular edit, or as a vandalized edit.

The problem of text categorization [9] is one of the most important problems of NLP. We can find proposals for solving the classification of query request results into relevant or irrelevant categories from the early 60’s, where a Naïve Bayes based training method was used [6]. Later, more sophisticated methods were proposed, which performs very well in the task of text categorization [5,10]. As a feature representation, the bag-of-word model or vector space model (VSM) was proposed, which is a common but pretty strong, finite dimensional numeric representation of any textual data [8,7,2].

* M. Jelasity was supported by the Bolyai Scholarship of the Hungarian Academy of Sciences. This work was partially supported by the Future and Emerging Technologies programme FP7-COSI-ICT of the European Commission through project QLectives (grant no.: 231200). This work was supported in part by the NKTH grant of the Jedlik Ányos R&D Programme (project codename TEXTREND) of the Hungarian government.

In this paper we propose a common feature set for the task, which contains basic, word based features and complex bag-of-word based, optimized features as well. After, we introduce a voting based classifier method. We propose a method that helps fine tune the parameters of this meta-classifier avoiding the overfitting during the model building phase. Basically, this study is an overview of our system, which was applied in the PAN 2010 Wikipedia Vandalism Detection shared task [1].

2 Machine learning based vandalism detection approach

To solve the Wikipedia vandalism detection problem we decided to propose an inferring method, which is as automatic as possible. This method follows the traditional NLP approach: first, we extract features from the published training set, then, we apply statistical learning algorithms to produce a model which can be used for automatically labeling the evaluation set. The main idea behind our features was to try to capture the vandalism class as much as possible, since the regular one is too general. The description of the features is the following:

- *BalancedVSM (BVSM)*
  This feature is a specialization of the VSM where the vector of a certain document contains only 0 or 1 values for each dimension. In our case, we used 4 different values as vector elements:
  - when the edit does not contain the word, then the value is *n*,
  - when the word is in an added sequence, then the value is *a*,
  - when the word is in a removed sequence, then the value is *d*,
  - when the word is in a changed sequence, then the vale is *c*.

  As it is well known using this type of VSM representation is not very successful [3] due to the fact that the dimension of this representation is 47 324. And so we had to apply a dimension reduction method, which is based on the InfoGain [3] score. Our preliminary observations showed that choosing the top 100 attributes results in better representation.
  Since the distribution of the regular and vandalism samples is totally unbalanced ($\sim$93.86% regular samples), the above described attribute extraction step over-represents the words from the regular edits. Having seen this problem we *balanced* the VSM representation: initially, we selected all the samples which were classified as vandalism. Then, we iteratively added to this set randomly selected, regular samples of the same quantity. Next, we performed the previously described dimensional reduction step and we stored and summed up the InfoGain scores of the top 100 attributes. Finally, we selected the top 100 attributes based on the aggregated scores and used them as Balanced SVM attributes.
- *CharacterStatistic*
  This feature family involves two different attributes: the upper case letter and the non-letter character occurrences divided by the number of characters respectively.
- *RepeatedCharSequences*
  One of the signs of vandalism is when somebody just repeats a sort string e.g. ”asdasdasdasdasd“. For this reason, we scanned the modifications and the comments to find short and frequent repeats.

– *ValidWordRatio*
  In these two attributes, we used dictionaries to provide the feature values. We used a simple English language dictionary and another that contains pejorative English expressions. Finally, the feature values are the number of the word occurrences in the dictionaries normalized by the word occurrences in the given edit.
– *CommentStatistic* (non-edit based feature)
  Commenting on the modification is made available for each user who edits Wikipedia pages. The possible feature values are:
  – *deleted* if the comment of the edit was deleted,
  – *comment* if the user has written into the comment field,
  – *nothing* in any other cases.
– *UserNameOrIP* (non-edit based feature)
  The user who edits Wikipedia can either register and choose a nickname or not register and use his IP number. So we added a feature that describes whether a user is registered or not.

Based on the previously defined features, we built several models applying different learning algorithms. These algorithms are quite common, and their implementations are available from several sources. We used the WEKA [4] implementations in our experiments. The algorithms used and their short descriptions are the following:

– *ZeroR*: the most frequent class classifier.
– *NaïveBayes* : Bayes' theorem based classifier.
– *J48*: one of the decision tree learning methods.
– *LogReg*: the maximum likelihood based Logistic Regression method.
– *SMO*: an implementation of the SVM classifier.
– *WeightedVotingMetaclassifier*: This classifier combines several underlying classifier algorithms based on a weighted aggregation. This algorithm is the only one which is not an official WEKA algorithm since we had to develop it as a WEKA extension.

3 Evaluation

In this section we describe the evaluations we made and present our final result measured on the released evaluation set. Since we knew the correct class labels for the training set only, we could only use this information for evaluation. We used the AUC score as evaluation metric.

3.1 Evaluation of different VSMs

In our first evaluation, we investigated the representation power of different VSMs. The overall results can be seen in Figure 1. Here, we evaluated several classifiers on the Normal VSM feature set, which is a simple VSM representation of the edits. The second feature set is an Attribute Selected VSM, where we retained the top 100 InfoGain scored VSM features. As one can see this is a much stronger feature set, but by using the Balanced Attribute Selected VSM, we can achieve a higher 10-fold AUC score in the case of each classifier. For this reason, we chose this representation as the base VSM representation in later evaluations.

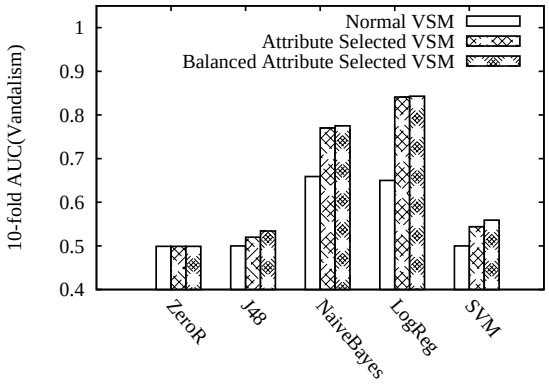

Figure 1. AUC results of different VSM feature sets

3.2 Results of the Different Feature Sets

Our second evaluation focused on the examination of the relation of the defined feature sets to the chosen training algorithms. We defined some feature sets as the subsets of the previously defined feature ranges and performed some learning-evaluation phases applying different training algorithms. The results are summarized in Table 1.

Table 1. Overall results measured on different feature sets (the top two results are highlighted)

Features	ZeroR	J48	NaïveBayes	LogReg	SMO
BVSM	0.4990	0.5230	**0.7220**	**0.8130**	0.5590
BVSM, stop	0.4990	**0.8680**	0.7750	**0.8430**	0.5430
All features, stop	0.4990	0.8280	**0.8830**	**0.8870**	0.5820

In Table 1, the semantics of the feature set labels is the following: ”BSVM“ means the Balanced VSM representation; ”BSVM, stop“ is the same as the previous except that we added a stop world list, which ignores the meaningless words; ”All features, stop“ represents the feature set, where we used all the above defined features and in the case of BVSM the stop world list was also used.

The most reasonable models are the probability based ones, however in the case of the last feature set (All features, stop) the J48 algorithm, which is a clearly discriminative model based approach, also shows pretty good AUC result.

In the case of the last feature set, the fact that one of the discriminative model could achieve a similar result than the probability based approaches indicated that this feature set is quite stable. In our further experiments we used this feature set.

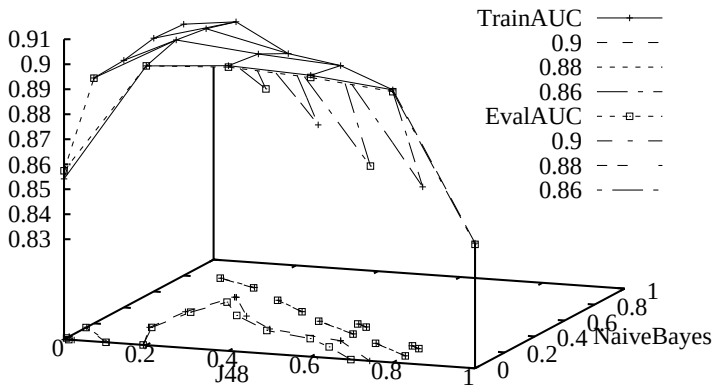

Figure 2. Parameter defined AUC surface on optimization and evaluation set

From the fact that these algorithms work in a completely different way, we assumed that perhaps the algorithms, which were based on different approaches, made different errors. From this naturally comes that idea that we should try to combine the three best algorithms namely the J48, the NaïveBayes and the LogReg, and built the previously introduced Weighted Voting Metaclassifier on the top of these three algorithms. The only questions here are *How should we determine the weights of the underlying classifiers?* and *Are the optimal weights found in the training set optimal on the evaluation set as well?*

3.3 Voting Based Classification and Parameter Tuning

We decided that we use the 10-fold AUC scores as the optimality measurement of a weight setting. To check the validity of our optimization process, we splitted the training set into an optimization and an evaluation set by the ratio of 4:1. We performed a 10-fold cross validation based optimization of the parameters on the optimization set and we checked whether this selection is optimal on the evaluation set or not. We performed this optimization in the whole parameter space. The summary of our optimization can be seen in Figure 2. In this figure, the x-axis and y-axis represent the weight of the J48 and Naïve Bayes classifiers respectively. Since the weights must be normalized, the weight of the LogReg model can be calculated as $(1-$weight of J48 $-$ weight of NaïveBayes $)$.

As one can see, the results of the evaluation set and the results of the optimization sets correlate (the two surfaces are almost the same). So we can say that these optimiza-

tion criteria are valid and we found that the optimal weighting of the algorithms is the following $(J48 : 0.3; NaiveBayes : 0.09; Logistic : 0.61)$.

The achieved AUC 10-fold cross validation based score of the optimally weighted metaclassifier is **0.9129**, which is significantly higher than the best score in Table 1. Thus, we used this combined Weighted Voting Metaclassifier model (, which learned on the full train set and used the weights presented above) for making our final predictions on the official evaluation set. Our result makes **0.87669** error score on the official evaluation set of vandalism detection task.

4 Conclusions

Our experiments in the field of detecting vandalism in Wikipedia edits indicated that we should participate in the Wikipedia Vandalism Detection Shared Task. Although our solution made an average performance on the challenge, we feel that our work has a strong contribution. This contribution is twofold. First, we developed a strong feature representation for the task, which can be built in a fully automatic manner and some of these features are pretty complex e.g. the Balanced VSM representation, which is a novel extension of the basic VSM representation and is suitable for learning tasks where the class labels have a highly unbalanced distribution. Second, we successfully combined classification methods in a weighted manner, where the weights had been optimized as hyperparameters.

References

1. Pan 2010 lab, task 2: Wikipedia vandalism detection, http://www.uni-weimar.de/medien/webis/research/workshopseries/pan-10/task2-vandalism-detection.html
2. Belani, A.: Vandalism detection in wikipedia: a bag-of-words classifier approach. CoRR abs/1001.0700 (2010)
3. Bishop, C.M.: Pattern Recognition and Machine Learning (Information Science and Statistics). Springer-Verlag New York, Inc., Secaucus, NJ, USA (2006)
4. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The weka data mining software: an update. SIGKDD Explor. Newsl. 11(1), 10–18 (2009)
5. Joachims, T.: Text categorization with support vector machines: learning with many relevant features. In: Nédellec, C., Rouveirol, C. (eds.) Proceedings of ECML-98, 10th European Conference on Machine Learning. pp. 137–142. Springer, Heidelberg et al. (1998), http://citeseer.ist.psu.edu/joachims97text.html
6. Maron, M.E., Kuhns, J.L.: On relevance, probabilistic indexing and information retrieval. J. ACM 7(3), 216–244 (1960)
7. McCallum, A., Nigam, K.: A comparison of event models for naive Bayes text classification. In: Learning for Text Categorization: Papers from the 1998 AAAI Workshop. vol. 752, pp. 41–48 (1998), http://www.kamalnigam.com/papers/multinomial-aaaiws98.pdf
8. Salton, G., Wong, A., Yang, C.S.: A vector space model for automatic indexing. Commun. ACM 18(11), 613–620 (1975)
9. Sebastiani, F.: Machine learning in automated text categorization. ACM Comput. Surv. 34(1), 1–47 (2002)
10. Zhang, T., J. Oles, F.: Text categorization based on regularized linear classification methods. Inf. Retr. 4(1), 5–31 (2001)