Improving KNN-based e-mail classification into folders generating class-balanced datasets.

Pablo Bermejo, Jose A. Gamez, Jose M. Puerta
Intelligent Systems and Data Mining group
Computing Systems Department
Universidad de Castilla-La Mancha,Spain
{pbermejo,jgamez,jpuerta,}@dsi.uclm.es

Roberto Uribe
Computing Engineer Department
Universidad de Magallanes, Chile
Database Group (UART)
U. Nacional de la Patagonia Austral, Argentina
ruribe@ona.fi.umag.cl

Abstract

In this paper we deal with an e-mail classification problem known as e-mail foldering, which consists on the classification of incoming mail into the different folders previously created by the user. This task has received less attention in the literature than spam filtering and is quite complex due to the (usually large) cardinality (number of folders) and lack of balance (documents per class) of the class variable. On the other hand, proximity based algorithms have been used in a wide range of fields since decades ago. One of the main drawbacks of these classifiers, known as lazy classifiers, is their computational load due to their need to compute the distance of a new sample to each point in the vectorial space to decide which class it belongs to. This is why most of the developed techniques for these classifiers consist on edition and condensation of the training set. In this work we make an approach to the problem of e-mail classification into folders. It is suggested a new algorithm based on neighbourgood called Gaussian Balanced K-NN, which does not edit nor condense the database but samples a whole new training set from the marginal gaussian distributions of the initial set. This algorithm lets choose the computational load of the classifier and also balances the training set, alleviating the same problems that edition and condensation techniques try to solve. **Keywords:** Balanced, class, distance, classification.

1 Introduction

One of the most common tasks in text-mining is classification [16], where the goal is to decide which class a given document belongs to among a set of classes. Apart of the needed preprocessing in any data mining task, in this case we need to perform some extra preprocessing in order to transform the unstructured text document into a data structure susceptible of being used as input by a learning algorithm. Concretely, we seek for a bi-dimensional table with rows representing documents (e-mails) and columns representing predictive attributes or terms[1]. This task has been widely studied for its applications to spam filtering, but it has not been so studied the classification of e-mails into folders defined by the user [1, 3, 2].
From the available dictionary of attributes, it is possible to construct or sample new documents for those cases in which classes distributions are not balanced over the whole set of documents. In this work the authors sample new documents to get a more optimal training dataset to perform e-mail classification [1, 15]. This work has two goals:

[1] In text mining applications, most of the attributes are words or tokens appearing in the documents, and in general they are referred to as *terms*

L. Magdalena, M. Ojeda-Aciego, J.L. Verdegay (eds): Proceedings of IPMU'08, pp. 529–536
Torremolinos (Málaga), June 22–27, 2008

1. Balance the number of representative documents for each class.
2. Introduce a new lazy classifier K-NN [12] here called *Gaussian Balanced K-NN*, which aims to sample a new set of highly predictive prototypes.

Besides this introduction, next section explains more in depth the task of data mining and the particular problem of e-mail classification. Section 3 shows a brief review of classifiers based on proximity. In Section 4 we introduce the *Gaussian Balanced K-NN* classifier. Next, in Section 5 we compare it against classifiers usually considered for this task: *NBM, K-NN* and *SVM*. Finally, conclusions are shown and some possible future work is suggested.

2 Text Data Mining and E-mail Classification

In this section we formally state the problem of e-mail foldering, and briefly describe the preprocessing carried out and the validation used.

This work focuses on automatic classification of mails, regarding them as a set of documents and without considering any special feature. Formally, the problem can be defined as giving a set of mails $C_{train} = \{(d_1, l_1), \ldots, (d_{|D|}, l_n)\}$ obtain a classifier $c : D \rightarrow L$, where:
- $d_i \in D$ is the document which corresponds to the ith mail of the given set of documents (mails) D,
- $l_i \in L$ is a folder containing several documents,
- $L = \{l_1, \ldots, l_{|L|}\}$ is the set of possible folders (i.e. the class variable).

the main differences between standard classification and text classification are: the need of preprocessing the unstructured documents in order to get a standard data mining dataset (bi-dimensional table) and the usually large number of features (attributes) and large cardinality of classes in the resulting dataset. In this work we focus on *bag-of-words* model, that is, a document (mail) is regarded as a set of words or terms without any kind of structure. For the selection of the documents and terms (i.e., the vocabulary V) used in our study we have followed the preprocessing described in [1]:

- Documents: Non-topical folders (inbox, sents, trash, etc.) and folders with only one or two mails are not considered.
- Terms: We only consider words as predictive attributes (MIME attachments are removed) and no distinction is made respect to where the word appears (e-mail header or body). Stop-words and words appearing only once are removed.
- Class: The folder hierarchy has been flattened and each one of the resulting folders constitutes a class label or state.

Our datasets can be observed as bi-dimensional matrices $M[\mathsf{numDocs}, \mathsf{numTerms}]$, where $M[i, j] = M_{ij}$ is a real number representing (some transformation of) the frequency of appearance of term j in document i.

As reported in [1] using training/test splits done at random for validation of e-mail classification is not appropriate because e-mail datasets depends on time, and so random splits may create unnatural dependencies. Because of this fact Bekkerman et al (2005) proposed a new validation scheme they called: *Incremental time-based split validation*. It consists on ordering mails based upon its *timestamp* field, and then training with the first t mails and testing using next t. After that, training is performed with first $2t$ mails and testing with next t, and that way until it is trained with the first $(K-1)t$ mails and tested with the remaining ones, K being the number of time splits the total amount of mails is divided into, and t being the number or mails in each time split. Finally, the accuracy averaged over the $K-1$ test sets used is reported.

3 Classification algorithms based on neighbourhood

Classification algorithms based on proximity [8] are theoretically simple and have a long background in supervised classification. Documents whose class is known are represented in a vectorial space, where each coordinate i corresponds to the *i-th* attribute used in the representation of the documents. The value M[i,j] represents its frequency in such document or the *tf*idf* value (which takes into account that a high frequency in a document is important but this importance is not such if the same attribute also has high frequency in the rest of documents).
These algorithms are called lazy because there not exists a training step to build a model which will be used to classify the documents. So each time a new document is to be assigned a class it will be necessary, in the most general and basic case, to compare the distance of this document to all the other documents in the database. Thus, the NN(*Nearest Neighbor*) algorithm would assign to this document the label of the nearest document to it in the vectorial space.
From this NN algorithm, several algorithms based on neighbourgood have been developed. The straight evolution from NN is the K-NN algorithm, in which the assigned class is the one which gets more votes among the K nearest neighbors. This way it is avoided erroneous assignments in those cases where the closest document does not belong to the correct class the new document belongs to (spatial intrusion reduction or noise). Among many variants of K-NN we can summarize:

- *Different weights for attributes* [7]. When computing the distance between two documents the value of each dimension of the documents is used. Since dimensions correspond to attributes used to represent the document, it seems a good idea to give more weight to those attributes more relevant than others. A way to decide this importance might be a previous selection or ranking of the set of attributes.

- *Use of a voting threshold.* It can be stablished a minimum number of votes that a class should receive before assigning it to the document being classified. If no class receives votes over this threshold the document remains unclassified.

- *Average distance.* K nearest neighbors are not used to vote, but it is computed the average distance of the document being classified to each of the K neighbors belonging to the same class. Thus the assigned class is the one which minimizes the average.

- *Use of centroids.* The K-NCN [18] algorithm selects, among all documents in the training set, the document or centroid representative of each class. Then, distances are not computed to each training document but to each centroid. This approach drastically reduces the computational load, but it makes the centroid selection a very important phase for a successfull classification.

There exist two key decisions to make when designing a classification algorithm based on proximity: the representation of dimensions (attributes) in samples and the distance metric used to compute distances between two samples. Respect to the distance metric, the most common are those belonging to the *Minkowski* distances family [6] (as the Euclidian Distance).
Since classification algorithms based on proximity have a high computational load, one of the main threads of investigation for these classifiers is related to the reduction of the training set size [8, 18]. The resulting subset is expected to be more representative and useful than the original training set. These tasks can be divided in two groups: edition and condensation tasks [9]. Edition techniques are designed to find and delete missclassified documents or documents whose class is erroneous. On the other hand, condensation techniques try to select an optimum training subset. There exists a third kind of techniques, called *generation or replacement of prototypes* [19], which does

not seem to have been studied as much as *edition* and *condensation*. One of the former studies of replacement techniques appears in [5], where two nearest neighbor samples of the same class are mixed if the accuracy in the training set is not decreased. This lets reduce the number of samples and creates new ones with probably a better predictive power. Some changes to this method are in [11], where clusters are mixed instead of samples, and in [21] where vector quantization is used due to its property of data compression previous to its sending through a channel. This VQ-NN takes as input the training set and returns a reduced and differente set which will be used as training set. The main problem of this method is that it does not provide the option to choose the number of prototypes to be created, but it seems to perform better than several algorithms based on neighbourhood which use edition or condensation techniques.

The standard similarity metric in text is the cosine value of such angle (used in our suggested algorithm):

$$sim(u,v) = \frac{\vec{u} \cdot \vec{v}}{|\vec{u}| \times |\vec{v}|} = \frac{\sum_{i=1}^{n} t_{iu} \times t_{iv}}{\sqrt{\sum_{i=1}^{n} t_{iu}^2 \times \sum_{i=1}^{n} t_{iv}^2}}$$

With n the number of dimensions (attributes) of each vector and t_{iu} the value of dimension i in vector (document) u.
In the results shown below, our suggested algorithm Gaussian Balanced K-NN represents documents using *tf*idf* values, this is that because in different experiments we found out that this metric performs much better than a frequency-based representation; and this is also the case for the Support Vector Machines (SVM) classifier [13, 14] with which some comparisons are made in Section 5.3. However, Naive Bayes Multinomial [17] and K-NN classifiers showed in our experiments that frequencies perform better. In this work we only show the results with the most appropiate representation metric.

4 Gaussian Balanced K-NN

The algorithm we propose in this work is based on the original K-NN algorithm with the following modifications:

1. The training set (set of documents to compute distances) is sampled from the truncated Gaussian distribution of the original training set.
2. A balancing of classes is made. This is acheived by sampling the same number of instances for each class from the learnt Gaussian distribution.
3. No vectorial distance metric is used but the cosine distance.

When the used database to perform classification has not been designed on porpose for this task but it comes from real life sources, a common problem is the lack of balance in the number of documents belonging to each class. In classifiers as Naive Bayes this lack of balance might derive in some overfitting of learnt parameters, and in non parametric classifiers as NN it might lead to some *invasion* in the vectorial space which avoids the correct classification for documents whose true class appears just a few times in the training set. Gaussian Balanced K-NN needs to receive as input the number P of documents to sample by class from the gaussian distribution computed from the training set. Since documents are represented by a vector of attributes, *mean* and *deviation* values are computed from the values of each attribute from documents belonging to the same class. This distribution must be truncated for values below 0 since negative values for frequencies or tf*idf would not make sense.
Besides the balancing of classes, our suggested algorithm tries to exhibit the advantages of the three improvements for neighbourhood algorithms mentioned in Section 3:

1. *Edition.* Wrong classifications and missclassification errors are expected to be alleviated since the gaussian distribution is learnt form all the documents of each

class, so a few errors in classification should not affect the final distribution.

2. *Condensation.* Low predictive documents influence is also expected to be alleviated by the Gaussian distribution. And the computational load can be selected since it depends on the number P of documents or prototypes to be sampled for each class.

3. *Generation (replacement) of prototypes.* This is quite obvious since our algorithm samples P instances (see Alorithm 1) for each class.

In Algorithm 1 it is shown the classification procedure for our suggested Gaussian Balanced K-NN algorithm.

```
    P ←prototypes to sample per class;
    K ←neighbors whose vote will count;
    X ←number of executions;
    E ←original training set;
    E' ← ∅;
    R ← ∅;
 1  for(c = 0; c < NumOfClasses; c++)
 2    for(a = 0; a < NumOfAttrib; a++)
 3      μ_ca ← mean of attrib. a in class c
 4      σ_ca ← standard deviation of a in class c
 5    endfor
 6  endfor
 7  for(c = 0; c < NumOfClasses; c++)
 8    for(p = 0; p < P; p++)
 9      E' ← E' ∪ new sampled prototype
10              from N(μ_c, σ_c)
11    endfor
12  endfor
13  R ← Run K-NN using E'
14  return R
```

Algorithm 1. Gaussian Balanced K-NN.

5 Experiments

5.1 Database and tools

As in [1, 15] we have used datasets corresponding to seven users from the ENRON corpus (mail from these users and a temporal line in increasing order can be downloaded from http://www.cs.umass.edu/∼ronb). The downloaded data has been preprocessed according to the process described in Section 2. To do this we have coded our own program in Java that interacts with *Lucene* information retrieval API (http://lucene.apache.org/who.html) and outputs a sparse matrix $M[\mathsf{numDocs}, \mathsf{numTerms}]$ codified as a *.arff* file, i.e., a file following the input format for the WEKA data mining suite [20]. Table 2 describes the obtained datasets. The last column of Table 2 shows the average dispersion (percentage of total number of instances) of all the classes for each user and the standard deviation. The standard deviation can be seen as a representation of how unbalanced the classes of a user are.

Table 1: Instances, Classes, Attributes and Dispersion Average of classes in datasets

Table 2: Datasets statistics

	#I	#C	#A.	%D($\mu \pm \sigma$)
lokay-m	2489	11	18903	9.09±12.71
sanders-r	1188	30	14463	3.33±6.36
beck-s	1971	101	13856	0.99±1.25
williams-w3	2769	18	10799	5.56±13.70
farmer-d	3672	25	18525	4.00±6.96
kaminski-v	4477	41	25307	2.44±3.17
kitchen-l	4015	47	34762	2.13±3.06

5.2 Classifiers and Evaluation

The Gaussian Balanced K-NN classifier has been compared with a Random Balanced K-NN classifier besides three other classifiers: IBk (Weka implementation of K-NN using euclidian distance), Naive Bayes (its multinomial version for text) and the most recommended classifier in literature for text classification, SVM. The Random Balanced K-NN (RB K-NN) classifier has been tested to check if the results obtained by Gaussian Balanced K-NN are due to the learnt distribution or just to the fact of creating the same number of documents per class. For RB K-NN the distribution for an attribute given a class is uniform in a continuos space from 0 to the highest value (in the training set) of such attribute given the class. IBk (classifier based on neighbourhood) and NBM implementations in Weka have been used in our experiments. With regard to SVM, we have followed the most suggested configuration in literature for text classification: linear kernel and attributes representation using tf*idf values normalized by the cosine function. For

SVM we have used WLSVM [10] which is an implementation of LibSVM [4] running under Weka.
NBM was tested using both frequencies and tf*idf representations, resulting frequencies in better accuracies, so results shown in Table 3 correspond to that representation.
IBk has been tested with $k = 1, k = 10, k = 20$ and $k = 30$, performing better for $k = 1$. In almost all cases, frequencies representation performs better than tf*idf, so Table 3 shows the results for this configuration.
Best results for Gaussian Balanced K-NN (GB K-NN)are obtained with tf*idf representation, $K = 15$ and $P = 30$ (see Algorithm 1). Since the final training set will be different in each execution of the algorithm due to the stochastic nature of the algorithm and that the sampling is performed from a probability distribution, it is also important to decide the number of X times the algorithm is to be run to compute mean values. This will also alleviate the fact that a distribution might have been learnt from just a few documents belonging to the same class. In this case, differences among executions are quite low so we have found quite reasonable to set $X = 5$. The parameters in RB K-NN are the same for GB K-NN.
The distance metric used in our suggested algorithm is the cosine value on the similarity function of two vectors (see Section 3). Validation has been performed following the procedure described in Section 2, with $t = 100$ as in [1]. Experiments have been run without any feature selection; that is, all the available attributes for each user have been used to represent their e-mails.

5.3 Results

Learning a conditional gaussian distribution of attributes given class shows to perform quite better than a uniform (random) distribution (See Table 3). Besides, our suggested Gaussian Balanced K-NN algorithm achieves better results than other algorithm of the family, IBk, for all tested values for k and for all users. Regarding Naive Bayes Multinomial, GB K-NN performs better in four of the seven tested users. Moreover, difference in accuracies is higher when GB K-NN is the winner than in the other cases. Finally, when testing with the most suggested classifier in literature (SVM) we find that GB K-NN beats it for three of the seven users (NBM nor IBk beats it in any case). A point quite interesting is that our proposal of class balancing performs better (in RB and GB K-NN) for a user when the results in the other classifiers are quite low; it is in theses cases also when the user has a high number of classes (See Table 2) and a low dispersion average. We can conclude that the class balancing improves the accuracy results, as the kind of distribution learnt to sample the new prototypes also proves to be appropiate. Besides, GB K-NN overcomes on average the SVM classifer, proving this way the high predictive power of our proposed algorithm.

Table 3: Accuracies for RB K-NN, GB K-NN, IBk, NBM and SVM.

	RB	**GB**	**IBk**	**NBM**	**SVM**
lokay-m	65.07	72.58	57.42	74.07	76.95
sanders-r	75.72	**77.27**	46.89	43.12	55.02
beck-s	46.37	**48.87**	18.16	26.08	33.52
williams-w3	67.53	78.53	84.60	84.85	88.20
farmer-d	44.11	66.52	65.26	67.22	72.34
kaminski-v	46.88	**53.58**	18.65	37.49	44.28
kitchen-l	41.82	*48.31*	29.45	32.37	52.32
MEAN	55.36	**63.67**	45.78	52.12	60.38

Since RB and GB K-NN classifiers have shown to perform quite good (Gaussian version better than Random version), NBM and SVM have also been tested using the balanced training set from the gaussian distribution.

Table 4: Accuracies for Gaussian Balanced NBM and SVM

	GB NBM	**GB SVM**
lokay-m	70.83	80.17
sanders-r	72.26	72.39
beck-s	46.16	46.54
williams-w3	84.31	88.93
farmer-d	66.84	75.84
kaminski-v	51.93	56.36
kitchen-l	38.61	54.88
MEAN	61.56	67.87

As it can bee seen in Table 4, class balancing also helps to improve classification accuracy for NBM and SVM. Paired signed ranking

wilcoxon tests have been performed with alpha=0.05 between the same user through the $X = 5$ accuracies obtained after $X = 5$ executions of the algorithm. Results of these tests show, for each user, that GB SVM is statiscally different than GB K-NN. GB NBM is also statiscally different than GB K-NN for all users except *farmer-d*. So it can be concluded that Gaussian Balanced SVM performs better than Gaussian Balanced K-NN in five of seven cases and Gaussian Balanced NBM does in just one case. To check if averages from GB SVM and NBM are statistcally different from GB K-NN, the same test has been run using as paired data the results shown in Table 4 and the GB column in Table 3. It cannot be concluded whether GB SVM (p-value=0.1094) or GB NBM (p-vaule=0.2969) perform better than GB K-NN since no statistic difference can be acknowledged after running the tests.

6 Conclusions and Future Work

It has been tackled the not very widely studied problem of e-mail classification into folders (e-mail foldering) using 2 non parametric classifiers (GB K-NN, IBk) and 2 parametric classifiers (NBM,SVM). SVM has come down to be the best text classifier (with tf*idf and linear kernel), but our GB K-NN proposal overcomes it on average and performes much better than Naive Bayes Multinomial and also better than a classifer based on neighbourhood: IBk, for different values ok k. As mentioned in Section 5.3 these good accuracies are due to class balancing and also to the appropiate distribution learnt from the original training set. Besides, higher differences between GB K-NN and NBM and SVM are achieved when these last two classifiers perform worse and the user has a greater number of classes. When NBM and SVM are balanced, they show a great increase in their accuracy getting GB SVM to perform better than GB K-NN.

Our balancing proposal provides the option of exactly selecting the computing load (*condensation*) thanks to the possibility of choosing the number of prototypes to sample per class. Besides, regarding GB K-NN, using the learnt probability distribution implicitly performs the *edition* task, thus alleviating the influence of documents misclassified or incorrectly classified in the original training set.

As future work, it could be tested some ideas found in the literature for other *Nearest Neighbor* algorithms, as weights for attributes or documents when computing the distance metric between documents. It could also be used some techniques for feature ranking and thus fix different weights for attributes. Besides, regarding e-mails as structured data (Body, To, From tags) could lead to better results.

Acknowledgements

This work has been supported by the JCCM under projects PBI-05-022/PCI-08-0048, MEC under project TIN1504-06204-C03-03 and the FEDER funds.

References

[1] R. Bekkerman, A. McCallum, and G. Huang. Automatic categorization of email into folders: Bechmark experiments on enron and sri corpora. Technical report, Department of Computer Science. University of Massachusetts, Amherst., 2005.

[2] P. Bermejo, J. A. Gámez, and J. M. Puerta. Attribute construction for e-mail foldering by using wrappered forward greedy search. In *9th International Conference on Enterprise Information Systems*, pages 247–252, 2007.

[3] P. Bermejo, J. A. Gámez, and J. M. Puerta. Construcción de atributos: un caso de estudio en clasificación de correo-e. In *V Congreso Espaol de Metaheurísticas, Algoritmos Evolutivos y Bioinspirados*, pages 555–562, 2007.

[4] C. Chang and C. Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at www.csie.ntu.edu.tw/∼cjlin/libsvm.

[5] C. L. Chang. Finding prototypes for nearest neighbor classifiers. *IEEE Trans-*

actions on Computers, 23:1179–1184, 1974.

[6] E. Chávez, G. Navarro, R. Baeza-Yates, and J. L. Marroqun. Searching in metric spaces. In *ACM Computing Surveys*, pages 273–321, 2001.

[7] S. Cost and S. Salzberg. A weighted nearest neighbor algorithm for learning with symbolic features. *Machine Learning*, 10:57–78, 1993.

[8] B. Dasarathy. Nearest neighbor (NN) norms: NN pattern recognition classification techniques. *IEEE Computer Society Press*, 1991.

[9] P. A. Devijver, J. Kittler, and J. Kittler. *Pattern Recognition: A Statistical Approach*. Prentice Hall, 1982.

[10] Y. El-Manzalawy and V. Honavar. *WLSVM: Integrating LibSVM into Weka Environment*, 2005. Software available at www.cs.iastate.edu/∼yasser/wlsvm.

[11] F. J. Ferri, R. A. Mollineda, and E. Vidal. An experimental comparison between consistency-based and adaptive prototype replacement schemes. In *ICPR '02: Proceedings of the 16 th International Conference on Pattern Recognition (ICPR'02)*, volume 3, page 30041, 2002.

[12] E. Fix and J. L. H. Jr. Discriminatory analysis, nonparametric discrimination. Technical report, USAF scholl of Aviation Medicine, Randof field, Project 21-49-004, Rept 4, 1951.

[13] T. Joachims. Text categorization with support vector machines: learning with many relevant features. Technical Report LS-8 Report 23, University of Dortmund, 1998.

[14] T. Joachims. *Learning to classify text using support vector machines*. Kluwer Academic Publishers, 2002.

[15] B. Klimt and Y. Yang. The ENRON corpus: a new dataset for email classification research. In *15th European Conference on Machine Learning*, pages 217–226, 2004.

[16] D. Lewis. *Representation and learning in information retrieval*. PhD thesis, Department of Computer Science, University of Massachusetts, 1992.

[17] A. McCallum and N. K. A comparison of event models for naive bayes text classification. In *AAAI/ICML-98 Workshop on Learning for Text Categorization*, pages 41–48, 1998.

[18] J. S. Sánchez, F. Pla, and F. J. Ferri. Prototype selection for the nearest neighbour rule through proximity graphs. *Pattern Recogn. Lett.*, 18:507–513, 1997.

[19] G. Toussaint. Proximity graphs for nearest neighbor decision rules: Recent progress. In *34th Symposium on Computing and Statistics*, 2002.

[20] I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques (Second Edition)*. Morgan Kaufmann, 2005.

[21] Q. Xie, C. Laszlo, and R. Ward. Vector quantization technique for nonparametric classifier design. *IEEE Trans. Pattern Anal. Machine Intell*, 15:1326–1330, 1993.