ATTRIBUTE CONSTRUCTION FOR E-MAIL FOLDERING BY USING WRAPPERED FORWARD GREEDY SEARCH

Pablo Bermejo, José A. Gámez and José M. Puerta
Computing Systems Department. Intelligent Systems and Data Mining research group (SIMD)
University of Castilla-La Mancha, Albacete, Spain

Keywords: Text mining, e-mail foldering, attribute construction, X-of-N attribute, wrapper approach, forward search.

Abstract: E-mail classification is one of the outstanding tasks in text mining, however most of the efforts in this topic have been devoted to the detection of spam or junk e-mail, that is, a classification problem with only two possible classes: spam and not-spam. In this paper we deal with a different 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 and is quite complex due to the (usually large) cardinality of the class variable (the number of folders). In this paper we try to improve the classification accuracy by looking for new attributes derived from the existing ones by using a data-driven approach. The attribute is constructed by taking into account the type of classifier to be used later and following a wrapper approach guided by a forward greedy search. The experiments carried out show that in all the cases the accuracy of the classifier is improved when the new attribute is added to the original ones.

1 INTRODUCTION

One of the most common tasks in text-mining is classification (Lewis, 1992), 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 classifier learning algorithm. Concretely, we seek for a bi-dimensional table with rows representing documents (e-mails) and columns representing predictive attributes or terms[1]. The selection of a good set of terms to describe the documents is of great importance in order to achieve a good classification rate (Bekkerman et al., 2005). The quality of the selected subset of terms depends on the importance of each attribute respect to its class, dependences among the included terms and existence of negative attributes (those which reduce the correct classification rate). Once the initial set of terms has been obtained, commonly based upon information retrieval techniques (Salton and Buckley, 1987), we can try to improve the resulting subset in two *supervised* (i.e., class dependent) ways:

- Attribute selection. This is a task (Liu et al., 2002) widely studied in data mining, consisting basically on reducing the set of available features (attributes) through the selection of the most important ones.
- Attribute construction. Sometimes it is possible to obtain higher quality attributes from those available (e.g. area from length and width). This task is called attribute construction (Larsen et al., 2002).

In this work we deal with the second option, specifically applied to a particular task inside text classification: classification of e-mail into folders (Bekkerman et al., 2005; Klimt and Yang, 2004). E-mail classification has been widely studied applying it to classification or filtering of junk mail (or *spam*), but its application to (semi)automatic classification of mail into folders (*e-mail foldering*) defined by user has not been so deeply investigated (Bekkerman et al., 2005).

[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*

In this paper we propose to use attribute construction in order to improve the quality of the classifiers learnt from them. In this sense the goal of the work is to construct a single attribute that collects possible dependences among those used as building blocks, and whose use improves the accuracy of the classification process. Thus, our main contribution is the design of such a new derived attribute and of the search algorithms used to look for it.

The work is structured in the following sections: Section 2 details the model used to classify e-mails into folders; next in Section 3 we introduce the concept of attribute construction and the type of attribute we construct; in Section 4 we describe the search algorithms proposed to look for good derived attributes; finally, in the last two sections we present the experiments carried out and our analysis and conclusions.

2 TEXT (E-MAIL) CLASSIFICATION

The main differences between standard classification and classification of text 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 or attributes in the dataset. In this work we focus on models of *bag-of-words*. In these models, a document text is regarded as a set of words or terms without any kind of structure. Therefore, these models are based upon the fact that a document can be represented with a selection of terms from a dictionary or vocabulary V. The identification process of the set of terms (V) can be performed in two stages: (1) deleting irrelevant attributes or *stop-words*; and (2) selection of a set of *most relevant* terms for the classification task. From this point our dataset can be observed as a bi-dimensional matrix $M[\mathsf{numDocs}, \mathsf{numTerms}]$, where M_{ij} is a real number representing (some transformation of) the frequency of appearance of term j in document i.

Our interest in 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 established from a set of mails $C_{train} = \{(d_1, l_1), \ldots, (d_{|D|}, l_n)\}$, such that $d_i \in D$ is the document which corresponds to the ith mail of the set of documents or mails D, l_j corresponds to the folder that contains it and $L = \{l_1, \ldots, l_{|L|}\}$ is the set of possible folders. The goal is to build a classifier $c : D \rightarrow L$. In this work we will focus on the probabilistic classifier model Nave Bayes Multinomial (NBM) to solve the mail classification problem. In (McCallum and Nigam, 1998), NBM and NB Binomial are compared resulting in NBM to achieve a better performance when vocabulary size is not small. Our main goal is the search of new features from the available terms to improve the correct classification of new mails and, therefore, our effort is performed previously (or simultaneously) to the application of NBM, and in general, this task is valid independently of the classifier used.

2.1 Nave Bayes Multinomial

Formally a NBM classifier assumes independence among terms once the class they belong to is known. Besides, this model lets us regard not only those terms appearing in each document but also the frequency of appearance of each term. This is important, because we can presume that a high appearance frequency increases the probability of belonging to a particular class. In the model here considered, the class a document belongs to is decided by calculating the class which maximises the Bayes rule (eq. 1), computing the conditional probability as shown in equation 2:

$$P(c_j|d_i) = \frac{P(c_j)P(d_i|c_j)}{P(d_i)} \tag{1}$$

$$P(d_i|c_j) = P(|d_i|)|d_i|! \prod_{t=1}^{|V|} \frac{P(w_t|c_j)^{N_{it}}}{N_{it}!} \tag{2}$$

N_{it} being the amount of times that term w_t appears in document d_i, $|V|$ the size of our vocabulary, $|D|$ the amount of documents to classify and $|d_i|$ the length of document i. Equation 2 assumes independence among terms. The independence assumption among parameters is a problem itself since it is not realistic in real databases. Besides, this assumption gets even more troublesome when using the multinomial model (Lewis, 1998) because it assumes not just independency among different terms but also among various occurrences of the same term; this is alleviated by the type of new attribute that we propose in this work (see Section 3.1).

3 ATTRIBUTE CONSTRUCTION

As it has been commented above, the quality of the available attributes is decisive to reach a good accuracy classification rate. From this idea, several techniques (Hu, 1998a) have been developed for the construction of new attributes through the application of some operations over the available ones. Attribute construction is most important when working with real databases, since they have not been created thinking about their application to data mining tasks and thus it is possible they do not contain attributes meaningful enough for a beneficial use (Freitas, 2001).

In attribute construction the main goal is to get a new attribute which represents the regularities of our database in a simpler way to detect them and thus making the classification task easier (Otero et al., 2003). In this paper we aim to design a single new constructed attribute highly optimised for the classification process and inspired in the X-of-N methodology by using algorithms that according to the taxonomies about attribute construction described in (Hu, 1998a) and (Hu, 1998b) fall in the categories of *combination* and *data driven*, because we seek to identify dependences between the original attributes by interacting with the classifier (NBM) to be used later.

3.1 Attribute X-of-N

A X-of-N attribute (Zheng, 1995) can be viewed as a logical formula, concretely a disjunction of clauses (attribute-value pairs) and its value for a given instance is calculated by counting the number of true clauses when using that instance as input for the X-of-N formula. In a X-of-N attribute N refers to the number of (different) attributes included in it. Of course, the attribute takes values in the set $\{0,\ldots,N\}$.

For instance, if we have a dataset with 5 predictive attributes (A,B,C,D,E) and we make $N = 3$, then $X-of-\{A=a_1,\ C>=17.2,\ E=true\}$ could be a derived attribute from A, C and E. Now, if we have the two following instances

id	A	B	C	D	E
r1	a_1	0	19.45	34	false
r2	a_2	0	15.3	26	true

then it is easy to see that $X-of-\{A=a_1,\ C>=17.2,\ E=true\}(r1)=2$ and $X-of-\{A=a_1,\ C>=17.2,\ E=true\}(r2)=1$.

As we can observe from the example, comparison operators ($\{=,\leq,\geq,>,<,\text{etc.}\}$) are usually considered to build the clauses or attribute-value pairs.

In our problem, all the attributes are continuous representing frequencies of appearance, thus each attribute-value pair could be written as $A_i \leq t$, t being a threshold. The main disadvantage of this approach is the complexity of the searching process because we have to identify which attributes are included in the constructed one and also the specific threshold for each attribute, leading to a very large search space. An alternative to this representation is to restrict our attribute-value pairs to take only two ways: $A_i > 0$ or $A_i = 0$. This representation leads to a simpler search space and also allows us to explicitly represent inclusion and exclusions of terms in our constructed attribute. On the other hand, this representation corresponds to the boolean model in which we lose the information about frequencies. Finally, and trying to seek for a compromise between the search process and the expressiveness of the constructed attribute, we propose to consider only[2] $A_i > 0$ attribute-value pairs but modifying the evaluation process of the constructed X-of-N attribute in order to take into account the frequencies. Thus, if M_i represents an instance (a row in our dataset), we compute:

$$X-of-\{A_{j_1}>0,\ldots,A_{j_r}>0\}(M_i)=\sum_{k=1}^{r} M_{ij_k}.$$

In this way we take into account the frequencies which is really interesting having in mind the classification model (NBM) to be used and, simultaneously we simplify the search process to the point of simply looking for the attributes to be included in the constructed attribute. This idea, as far as we know, has not been proposed in the literature.

4 X-OF-N SEARCH ALGORITHM

In the previous sections we have identified relevant aspects for the design of the search algorithm:

- We look for a *single* constructed attribute.
- If $\{A_1,\ldots,A_n\}$ is the set of available attributes, them the search space is $\{0,1\}^n$, constrained to the fact that the number of 1's in a potential solution must be at most N.
- We use a wrapper approach based on the NBM classifier. That is, each time a potential solution has to be evaluated, a classifier has to be trained and evaluated.

Because of these conditions and because this is a preliminary approach we consider it appropriate to use a greedy algorithm to deal with our problem. Greedy search is a kind of metaheuristics inside a global metaheuristics set known as Constructive Metaheuristics. Greedy search is based upon constructing the solution step by step, choosing each time the best member of the solution and if needed we can stop at any time returning the current solution. On the other hand this behaviour leads to find just a local optimal because components have been chosen in a local manner without any perspective of the whole search space, however its performance tends to be quite high because it needs lower computation times (than stochastic methods) and besides the solutions it returns might be close to the global optimum.

[2]Initially we considered also $A_i = 0$ attribute-value pairs but we discard them because their use lead to a lower performance

4.1 Forward Search

Forward search is probably one of the most used techniques in variable selection, and due to the way in which we have designed our X-of-N algorithm, it seems appropriate to use this type of greedy search. Our algorithm will test at each step all the available attributes in our dataset to add a new condition to the X-of-N attribute, then it will choose the attribute which increases the correct classification rate the most. The search will stop when no improvement is found. Algorithm 1 shows the pseudocode of forward search.

```
In   A (attributes; $\{A_1,\ldots,A_n\}$); M (the dataset)
Out  The attribute X-of-N constructed
1    B ← A; X-of-N ← $\emptyset$;
2    bestacc ← NBM(A,M); $go \leftarrow true$
3    While ($|B| > 0$ and $go$) do {
4       $go \leftarrow false$; $best \leftarrow null$
5       C ← B
6       for(i=1; i < |C|; i++) {
7          acc ← NBM($\mathbf{A} \cup$ (X-of-N + $\{C_i\}$), M)
8          if (acc > bestacc)
9             best ← i; bestacc ← acc
10      }
11      if (best ≠ null) {
12         X-of-N += $\{C_{best}\}$
13         B -= $\{best\}$; $go \leftarrow true$
14      }
15   }
16   return X-of-N
```

Algorithm 1: X-of-N forward search.

At each iteration of the algorithm, the for loop runs over all possible candidates (**C**) in order to choose the best one to be included in the X-of-N attribute. In algorithm 1 the set of candidates is equals to all the available attributes not yet included in the target attribute X-of-N. This means to learn and evaluate n classifiers at the first iteration, $n-1$ at the second iteration, etc., which can be prohibitive if n is large, as it is usually the case. Because of this we introduce two different modifications:

• Let $\{X_{k_1},\ldots,X_{k_r}\}$ be the set of attributes already included in the target attribute X-of-N, then instead of search over all the remaining attributes, we only look for attributes that appear with any of the already included attributes in some documents. This *heuristic* seems to be reasonable because we are trying to catch dependences between the variables included in the X-of-N attribute. Thus, we replace step 5 in Algorithm 1 by:

$$\mathbf{C} \leftarrow \{B_j \mid \exists_{i,l=1..r}\ s.t.\ M[d_i,B_j]\cdot M[d_i,X_{k_l}] > 0\ \}$$

where $M[d_i,B_j]$ refers to the frequency of attribute B_j in document d_i.

• Our second simplification or heuristics is even more greedy, because we limit the candidates to those terms who share any document with the *last* attribute (X_l) entered in X-of-N, thus:

$$\mathbf{C} \leftarrow \{B_j \mid \exists_i\ s.t.\ M[d_i,B_j]\cdot M[d_i,X_l] > 0\ \}.$$

In this way we expect to make our algorithm faster but without decreasing significantly the accuracy.

5 EXPERIMENTS

5.1 Test Suite

The database used in our experiments is the Enron Corpus, the same used in (Bekkerman et al., 2005; Klimt and Yang, 2004). Mail from 7 users of this corpus and a temporal line in increasing order can be downloaded from http://www.cs.umass.edu/∼ronb.

As far as we know, the most in deep study carried out on this corpus can be found in (Bekkerman et al., 2005), where mails of seven users were selected for the classification study. We have carried out the same preprocessing process that in (Bekkerman et al., 2005); that is, stop-words deletion, removal of folders with 2 or less mails and we have also flattened folder levels to just one level. To do this preprocessing we have coded our own program that reads directories of text files and, by using the *Lucene* information retrieval API (http://lucene.apache.org/who.html), processes them and produces as output an *arff* file[3], that is, a file following the input format for the WEKA data mining workbench (Witten and Frank, 2005).

We have implemented the algorithms using LiO (Mateo and de la Ossa, 2006), a library of metaheuristics developed in Java by our research group which lets the programmer use or extend a great amount of metaheuristics. The wrapper approach is carried out by calling the NBM included in the WEKA API, but using the *time-based split evaluation* proposed in (Bekkerman et al., 2005). This validation method 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 finally trained with the first $(K-1)t$ mails and it its tested with the remaining ones. Being K the number of time splits the total amount of mails is divided into, and t the number or mails in each time split; in our case we have set t=100.

5.2 Results

We have selected the same users as in (Bekkerman et al., 2005), whose corresponding datasets are de-

[3]Using a sparse representation

scribed in Table 1. As we can see there is a great variability both in the number of instances and classes, so we can expect to have really different classification rates for them[4]. The results of NBM over these datasets are shown in column 1 (*original*) of Table 2 when about 14000 attributes were used as input. We also check the performance of NBM when some *easy* attribute selection is carried out, concretely we build 14 new datasets by projecting the original 7 datasets over the 100 and 1000 first attributes ranked by using mutual information. As we can see in Table 2 with this type of attribute selection the accuracy of the classifier decreases (on average) 2.5% to the original dataset when using the first 100 attributes, while it increases (on average) 3.3% when using the 1000 first attributes. In the following we experiment with 100 and 1000 attributes projection in order to analyse the impact of adding the X-of-N attribute to them.

Table 1: Instances and Classes for each user.

	Instances	Classes
beck-s	1971	101
farmer-d	3672	25
kaminski-v	447	41
kitchen-l	4015	47
lokay-m	2489	11
sanders-r	1188	30
williams-w3	2769	18

Table 2: Classification rates for each user.

	Original	100 At.	1000 At.
beck-s	33,945	26,131	31,546
farmer-d	65,893	69,742	71,045
kaminski-v	39,593	35,889	43,559
kitchen-l	36,618	32,200	32,759
lokay-m	72,099	62,974	74,181
sanders-r	46,956	49,051	62,698
williams-w3	77,915	80,732	80,438

Tables 3 and 5 show the results of using the X-of-N attribute during the classification process. The tables show the accuracy improvement (%) when the new attribute is used as well as the number of attributes (i.e. the value of *N*) included in the constructed attribute and the columns represent the algorithms we are going to test. Let us to analyse these results:

• With respect to the improvement achieved we can observe it shows a great variability over the different users, but on the average the accuracy improves 1.5% in the 100-attributes case and 1% in the 1000-attributes case. Although these figures are not very impressive from an absolute point of view, we have to take into account our starting point, that is, the (usually) low percentage of success when using the original set of attributes.

[4]In fact, as pointed in (Klimt and Yang, 2004; Brutlag and Meek, 2000) performance among different users varies much more than variation between different classifiers

Table 3: Improvement (and value of *N*) for each type of candidate set when using 100 attributes as input.

	Last attrib		X-of-N attribs.		All attribs.	
beck-s	2,24	(15)	2,31	(13)	2,31	(13)
farmer-d	0,71	(8)	0,71	(8)	0,71	(8)
kaminski-v	1,04	(8)	1,02	(8)	1,02	(9)
kitchen-l	1,33	(8)	1,37	(9)	1,37	(9)
lokay-m	1,19	(7)	1,19	(7)	1,19	(7)
sanders-r	3,31	(9)	3,31	(9)	3,31	(9)
williams-w3	1,03	(4)	1,03	(4)	1,03	(4)

• With respect to the three versions of our greedy search we can see that accuracies are similar in the 100-attributes data set, being a possible explanation that these 100 attributes are so important and so they appear in most of the documents. In the 1000-attributes case the extremely greedy behaviour of using as candidate only those attributes sharing at least a document with the last attribute added to X-of-N performs slightly worse than the other two versions. However, from the results it seems reasonable to use as candidate only those attributes sharing docs with the current X-of-N because the result is almost the same as when using all the attributes as candidates, being the amount of CPU time clearly lower in this case as shown in Table 4 for the 1000-attributes case.

• With respect to the number of attributes included in the X-of-N attribute, it varies depending on the case (user) and also on the type of candidate list used during the forward search. Thus, in the 100-attributes case the figures are quite similar for the three methods, and *N* is relatively small what yields to interpretable attributes that usually include the name or surname of the sender, the e-mail address, etc. More variability is observed in the 1000-attributes case, both among users and methods. In this case the simpler version adds few attributes while the other two versions add significantly more variables, leading to less interpretable results. We think that if interpretability is a goal, then an upper bound (i.e. N=7 as used by other authors) should be fixed, because in this way we get simpler X-of-N attributes but maintaining similar rates of improvement in accuracy.

6 CONCLUSION

In this paper we have proposed to look for a single X-of-N attribute in e-mail foldering. The attribute has been designed according to the target task, and search

Table 4: Greedy execution time (hours).

	Last attrib.	All attrib.	No candid.
beck-s	2,14054	9,98000	18,80485
farmer-d	5,988998	16,00856	21,52744
kaminski-v	7,60864	27,47413	36,43047
kitchen-l	24,11993	60,77219	66,94359
lokay-m	4,60200	4,46918	10,06760
sanders-r	1,24783	1,70260	2,07056
williams-w3	6,20118	10,23979	11,26965

Table 5: Improvement (and value of N) for each type of candidate set when using 1000 attributes as input.

	Last attrib		X-of-N attribs.		All attribs.	
beck-s	1,56	(9)	2,01	(18)	2,03	(20)
farmer-d	0,40	(5)	0,51	(10)	0,47	(12)
kaminski-v	1,15	(11)	1,36	(25)	1,37	(26)
kitchen-l	0,59	(7)	0,76	(18)	0,80	(21)
lokay-m	0,47	(5)	0,51	(5)	0,52	(10)
sanders-r	1,08	(4)	1,08	(4)	1,08	(4)
williams-w3	0,58	(3)	0,73	(6)	0,73	(6)

methods to look for it have also been designed and tested. The experiments carried out show that the use of the new attribute is beneficial with respect to the classifier accuracy, and also that in many cases it is interpretable. Besides, its construction process is not classifier-specific. With respect to the search methods we can say that the one considering as candidates to be included in X-of-N only those attributes sharing docs with the current X-of-N, exhibits the best tradeoff between CPU requirements and accuracy improvement. For the future we plan to go deeper in this study (different designs for X-of-N and different search methods) and also to consider the inclusion of more than one X-of-N attributes or the combination of attribute selection and construction instead of performing them in a two-stage process as in the current work.

ACKNOWLEDGEMENTS

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

REFERENCES

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

Brutlag, J. D. and Meek, C. (2000). Challenges of the email domain for text classification. In *ICML '00: Proceedings of the Seventeenth International Conference on Machine Learning*.

Freitas, A. A. (2001). Understanding the crucial role of attributeinteraction in data mining. *Artif. Intell. Rev.*, 16:177–199.

Hu, Y.-J. (1998a). *Constructive induction: covering attribute spectrum In Feature Extraction, Construction and Selection: a data mining perspective*. Kluwer.

Hu, Y.-J. (1998b). A genetic programming approach to constructive induction. In *3rd Anual Genetic Programming Conference*.

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

Larsen, O., Freitas, A., and Nievola, J. (2002). Constructing x-of-n attributes with a genetic algorithm. In *Proc Genetic and Evolutionary Computation Conf (GECCO-2002)*.

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

Lewis, D. D. (1998). Naive (Bayes) at forty: The independence assumption in information retrieval. In *Proceedings of ECML-98, 10th European Conference on Machine Learning*, number 1398, pages 4–15, Chemnitz, DE. Springer Verlag, Heidelberg, DE.

Liu, H., Motoda, H., and Yu, L. (2002). Feature selection with selective sampling. In *Nineteenth International Conference on Machine Learning*, pages 395 – 402.

Mateo, J. L. and de la Ossa, L. (2006). Lio: an easy and flexible library of metaheuristics. Technical report, Departamento de Sistemas Informticos, Escuela Politécnica Superior de Albacete, Universidad de Castilla-La Mancha.

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

Otero, F., Silva, M., Freitas, A., and NIevola, J. (2003). Genetic programming for attribute construction in data mining. In *Genetic Programming: Proc. 6th European Conference (EuroGP-2003)*.

Salton, G. and Buckley, C. (1987). Term weighting approaches in automatic text retrieval. Technical report, Cornell University.

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

Zheng, Z. (1995). Constructing nominal x-of-n attributes. In *International Joint Conference on Artificial Intelligence (IJCAI-05)*. Morgan Kaufmann.