Probabilistic Methods for Structured Document Classification at INEX'07

Luis M. de Campos, Juan M. Fernández-Luna, Juan F. Huete,
and Alfonso E. Romero

Departamento de Ciencias de la Computación e Inteligencia Artificial
E.T.S.I. Informática y de Telecomunicación, Universidad de Granada,
18071 – Granada, Spain
{lci,jmfluna,jhg,aeromero}@decsai.ugr.es

Abstract. This paper exposes the results of our participation in the Document Mining track at INEX'07. We have focused on the task of classification of XML documents. Our approach to deal with structured document representations uses classification methods for plain text, applied to flattened versions of the documents, where some of their structural properties have been translated to plain text. We have explored several options to convert structured documents into flat documents, in combination with two probabilistic methods for text categorization. The main conclusion of our experiments is that taking advantage of document structure to improve classification results is a difficult task.

1 Introduction

This is the first year that members of the research group "Uncertainty Treatment in Artificial Intelligence" at the University of Granada submit runs to the Document Mining track of INEX. As we had previous experience in automatic classification, particularly in learning Bayesian network classifiers [1,3], we have limited our participation only to the task of text categorization.

The proposed methodology does not use text classification algorithms specifically designed to manage and exploit structured document representations. Instead, we use algorithms that apply to flat documents and do not take structure into consideration at all. What we want to test is whether these methods can be used, in combination with some simple techniques to transform document structure into a modified flat document representation having additional characteristics (new or transformed features, different frequencies,...), in order to improve the classification results obtained by the same methods but using purely flat document representations.

The rest of the paper is organized in the following way: in Section 2 we describe the probabilistic flat text classifiers we shall use. Section 3 gives details of the different approaches to map structured documents into flat ones. Section 4 is focused on the experimental results. Finally, Section 5 contains the concluding remarks and some proposals for future work.

N. Fuhr et al. (Eds.): INEX 2007, LNCS 4862, pp. 195–206, 2008.

2 Probabilistic Methods for Flat Text Classification

In this section we are going to explain the two methods for non-structured (flat) text classification that we are going to use in combination with several methods for managing structured documents. One of them is the well-known naive Bayes classifier, whereas the other is a new method, based on a restricted type of Bayesian network.

The classical probabilistic approach to text classification may be stated as follows: We have a class variable C taking values in the set $\{c_1, c_2, \ldots, c_n\}$ and, given a document d_j to be classified, the posterior probability of each class, $p(c_i|d_j)$, is computed according to the Bayes formula:

$$p(c_i|d_j) = \frac{p(c_i)p(d_j|c_i)}{p(d_j)} \propto p(c_i)p(d_j|c_i) \tag{1}$$

and the document is assigned to the class having the greatest posterior probability, i.e.

$$c^*(d_j) = \arg\max_{c_i}\{p(c_i)p(d_j|c_i)\}$$

Then the problem is how to estimate the probabilities $p(c_i)$ and $p(d_j|c_i)$.

2.1 The Naive Bayes Classifier

The naive Bayes classifier is the simplest probabilistic classification model that, despite its strong and often unrealistic assumptions, performs frequently surprisingly well. It assumes that all the attribute variables are conditionally independent of each other given the class variable. In fact, the naive Bayes classifier can be considered as a Bayesian network-based classifier, where the network structure contains only arcs from the class variable to the attribute variables, as shown in Figure 1. In the context of text classification, there exist two different models called naive Bayes, the multivariate Bernouilli naive Bayes model [4,5,9] and the multinomial naive Bayes model [6,7]. In this paper we are going to use the multinomial model.

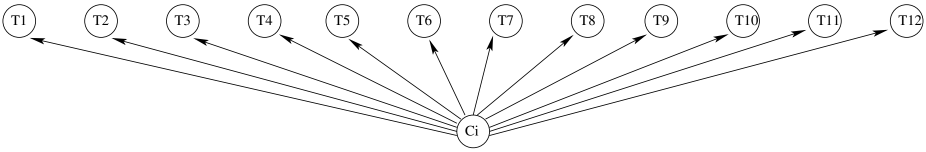

Fig. 1. The naive Bayes classifier

In this model a document is an ordered sequence of words or terms drawn from the same vocabulary, and the naive Bayes assumption here means that the occurrences of the terms in a document are conditionally independent given the class, and the *positions* of these terms in the document are also independent given the class. Thus, each document d_j is drawn from a multinomial distribution of words with as many independent trials as the length of d_j. Then,

$$p(d_j|c_i) = p(|d_j|)\frac{|d_j|!}{\prod_{t_k \in d_j} n_{jk}!}\prod_{t_k \in d_j} p(t_k|c_i)^{n_{jk}} \tag{2}$$

where t_k are the distinct words in d_j, n_{jk} is the number of times the word t_k appears in the document d_j and $|d_j| = \sum_{t_k \in d_j} n_{jk}$ is the number of words in d_j. As $p(|d_j|)\frac{|d_j|!}{\prod_{t_k \in d_j} n_{jk}!}$ does not depend on the class, we can omit it from the computations, so that we only need to calculate

$$p(d_j|c_i) \propto \prod_{t_k \in d_j} p(t_k|c_i)^{n_{jk}} \tag{3}$$

The estimation of the term probabilities given the class, $p(t_k|c_i)$, is usually carried out by means of the Laplace estimation:

$$p(t_k|c_i) = \frac{N_{ik} + 1}{N_{i\bullet} + M} \tag{4}$$

where N_{ik} is the number of times the term t_k appears in documents of class c_i, $N_{i\bullet}$ is the total number of words in documents of class c_i and M is the size of the vocabulary (i.e. the number of distinct words in the documents of the training set).

The estimation of the prior probabilities of the classes, $p(c_i)$, is usually done by maximum likelihood, i.e.:

$$p(c_i) = \frac{N_{i,doc}}{N_{doc}} \tag{5}$$

where N_{doc} is the number of documents in the training set and $N_{i,doc}$ is the number of documents in the training set which are assigned to class c_i.

In our case we have used this multinomial naive Bayes model but, instead of considering only one class variable C having n values, we decompose the problem using n binary class variables C_i taking their values in the sets $\{c_i, \overline{c}_i\}$. This is a quite common transformation in text classification [10], especially for multilabel problems, where a document may be associated to several classes. In this case we build n naive Bayes classifiers, each one giving a posterior probability $p_i(c_i|d_j)$ for each document. As in the Wikipedia XML Corpus each document may be assigned to only one class, we select the class $c^*(d_j)$ such that $c^*(d_j) = \arg\max_{c_i}\{p_i(c_i|d_j)\}$. Notice that in this case, as the term $p_i(d_j)$ in the expression $p_i(c_i|d_j) = p_i(d_j|c_i)p_i(c_i)/p_i(d_j)$ is not necessarily the same for all the class values, we need to compute it explicitly through

$$p_i(d_j) = p_i(d_j|c_i)p_i(c_i) + p_i(d_j|\overline{c}_i)(1 - p_i(c_i))$$

This means that we have also to compute $p_i(d_j|\overline{c}_i)$. This value is estimated using the corresponding counterparts of eqs. (3) and (4), where

$$p(t_k|\overline{c}_i) = \frac{N_{\bullet k} - N_{ik} + 1}{N - N_{i\bullet} + M} \tag{6}$$

$N_{\bullet k}$ is the numbers of times that the term t_k appears in the training documents and N is the total number of words in the training documents.

2.2 The OR Gate Bayesian Network Classifier

The second classification method for flat documents that we are going to use is based on a Bayesian network with the following topology: Each term t_k appearing in the training documents (or a subset of these terms in the case of using some method for feature selection) is associated to a binary variable T_k taking its values in the set $\{t_k, \overline{t}_k\}$, which in turn is represented in the network by the corresponding node. There are also n binary variables C_i taking their values in the sets $\{c_i, \overline{c}_i\}$ (as in the previous binary version of the naive Bayes model) and the corresponding class nodes. The network structure is fixed, having an arc going from each term node T_k to the class node C_i if the term t_k appears in training documents which are of class c_i. In this way we have a network topology with two layers, where the term nodes are the "causes" and the class nodes are the "effects". An example of this network topology is displayed in Figure 2.

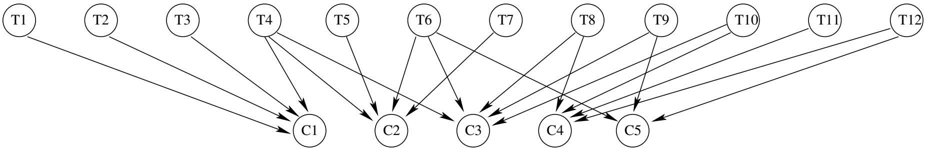

Fig. 2. The OR gate classifier

The quantitative information associated to this network are the conditional probabilities $p(C_i|pa(C_i))$, where $Pa(C_i)$ is the set of parents of node C_i in the network (i.e. the set of terms appearing in documents of class c_i) and $pa(C_i)$ is any configuration of the parent set (any assignment of values to the variables in this set). As the number of configurations is exponential with the size of the parent set, we use a canonical model to define these probabilities, which reduce the number of required numerical values from exponential to linear size. More precisely, we use a noisy OR Gate model [8].

The conditional probabilities in a noisy OR gate are defined in the following way:

$$p(c_i|pa(C_i)) = 1 - \prod_{T_k \in R(pa(C_i))} (1 - w(T_k, C_i)) \; , \; p(\overline{c}_i|pa(C_i)) = 1 - p(c_i|pa(C_i)) \, . \tag{7}$$

where $R(pa(C_i)) = \{T_k \in Pa(C_i) \,|\, t_k \in pa(C_i)\}$, i.e. $R(pa(C_i))$ is the subset of parents of C_i which are instantiated to its t_k value in the configuration $pa(C_i)$. $w(T_k, C_i)$ is a weight representing the probability that the occurrence of the "cause" T_k alone (T_k being instantiated to t_k and all the other parents T_h instantiated to $\bar{t}_h$) makes the "effect" true (i.e., forces class c_i to occur).

Once the weights $w(T_k, C_i)$ have been estimated, and given a document d_j to be classified, we instantiate in the network each of the variables T_k corresponding to the terms appearing in d_j to the value t_k (i.e. $p(t_k|d_j) = 1$ if $t_k \in d_j$), and all the other variables T_h (those associated to terms that do not appear in d_j) to the value $\bar{t}_h$ (i.e. $p(t_h|d_j) = 0 \; \forall t_h \notin d_j$). Then, we compute for each class node C_i the posterior probabilities $p(c_i|d_j)$. As in the case of the naive Bayes model, we assign to d_j the class having the greatest posterior probability.

The combination of network topology and numerical values represented by OR gates allows us to compute very efficiently and in an exact way the posterior probabilities:

$$p(c_i|d_j) = 1 - \prod_{T_k \in Pa(C_i)} (1 - w(T_k, C_i) \times p(t_k|d_j)) = 1 - \prod_{T_k \in Pa(C_i) \cap d_j} (1 - w(T_k, C_i)) \tag{8}$$

In order to take into account the number of times a word t_k occurs in a document d_j, n_{jk}, we replicate each node T_k n_{jk} times, so that the posterior probabilities then become

$$p(c_i|d_j) = 1 - \prod_{T_k \in Pa(C_i) \cap d_j} (1 - w(T_k, C_i))^{n_{jk}} \ . \tag{9}$$

The estimation of the weights in the OR gates, $w(T_k, C_i)$, can be done in several ways. The simplest one is to estimate $w(T_k, C_i)$ as $p(c_i|t_k)$, the conditional probability of class c_i given that the term t_k is present. We can do it by maximum likelihood:

$$w(T_k, C_i) = \frac{N_{ik}}{N_{\bullet k}} \tag{10}$$

Another, more accurate way of estimating $w(T_k, C_i)$ is directly as $p(c_i|t_k, \bar{t}_h \; \forall T_h \in Pa(C_i),\, T_h \neq T_k)$. However, this probability cannot be reliably estimated, so that we are going to compute an approximation in the following way[1]:

$$p(c_i|t_k, \bar{t}_h \; \forall h \neq k) \approx p(c_i|t_k) \prod_{h \neq k} \frac{p(c_i|\bar{t}_h)}{p(c_i)} \tag{11}$$

The values of $p(c_i|t_k)$ and $p(c_i|\bar{t}_h)/p(c_i)$ in eq. (12) are also estimated using maximum likelihood. Then, the weights $w(T_k, C_i)$ are in this case:

$$w(T_k, C_i) = \frac{N_{ik}}{N_{\bullet k}} \times \prod_{h \neq k} \frac{(N_{i\bullet} - N_{ih})N}{(N - N_{\bullet h})N_{i\bullet}} \tag{12}$$

[1] This approximation results from assuming a conditional independence statement similar to that of the naive Bayes classifier, namely $p(t_k, \bar{t}_h \, \forall h \neq k|c_i) \approx p(t_k|c_i) \prod_{h \neq k} p(\bar{t}_h|c_i)$.

3 Document Representation

In this section we deal with the problem of document representation. As we have seen before, we are using flat-document classifiers for this track, so we need methods to translate structural properties to plain text document.

Because these methods are independent of the classifier used, it is possible to make all possible combinations of classifiers and transformation methods, wich gives us a large amount of categorization procedures.

We shall use the small XML document (the beginning of "Don Quijote") displayed in Figure 3 to illustrate the proposed transformations. Next we explain the different approaches to map structural documents into flat ones.

```
<book>
 <title>El ingenioso hidalgo Don Quijote de la Mancha</title>
 <author>Miguel de Cervantes Saavedra</author>
 <contents>
   <chapter>Uno</chapter>
    <text>En un lugar de La Mancha de cuyo nombre no quiero
    acordarme...</text>
 </contents>
</book>
```

Fig. 3. "Don Quijote", XML fragment used to illustrate the different transformations

3.1 Method 1: "Only Text"

This is the naive approach. It consists in removing all the structural marks from the XML file, obtaining a plain text file. Used with the previous example, we obtain the document displayed in Figure 4:

```
El ingenioso hidalgo Don Quijote de la Mancha Miguel de Cervantes
Saavedra Uno En un lugar de La Mancha de cuyo nombre no quiero
acordarme...
```

Fig. 4. "Don Quijote", with the "only text" approach

This method should be taken as a *baseline*, as we are losing all the structural information. We would like to improve its classification accuracy by using more advanced representations.

3.2 Method 2: "Adding"

This method adds structural features to the document, different from the textual features. That is to say, structural marks are introduced into the document as if they were "additional terms". We can consider structural marks in an atomic way, or in the context of the other marks where they are contained (i.e. using part of the path from the mark being considered to the root element, until a certain depth level). Using the previous example, the `text` mark can be considered standalone ("adding_1", with depth = 1), `contents_text` ("adding_2", depth = 2) or `book_contents_text` ("adding_0", maximum depth value, the complete path to the root mark).

We show in Figure 5 the transformed flat document of the example document using "adding" with depth = 2. Leading underscores are used to distinguish between textual terms and terms representing structural marks:

```
_book _book_title El ingenioso hidalgo Don Quijote de la Mancha
_book_author Miguel de Cervantes Saavedra
_book_contents _contents_chapter Uno _contents_text En un lugar
de La Mancha de cuyo nombre no quiero acordarme...
```

Fig. 5. "Don Quijote", with the "adding_2" approach

3.3 Method 3: "Tagging"

This approach is the same as the one described in [2], also named "tagging". It considers that two appearances of a term are different if it appears inside two different structural marks. To modelize this, terms are "tagged" with a representation of the structural mark they appear in. This can be easily simulated prepending a preffix to the term, representing its container. We can also experiment at different depth levels, as we did in the method "adding".

Data preprocesed with this method can be very sparse, and very large lexicon could be built from medium sized collections. For our example document this method, with depth = 1, obtains the flat document displayed in Figure 6.

```
title_El title_ingenioso title_hidalgo title_Don title_Quijote
title_de title_la title_Mancha author_Miguel author_de
author_Cervantes author_Saavedra chapter_Uno text_En text_un
text_lugar text_de text_La text_Mancha text_de text_cuyo
text_nombre text_no text_quiero text_acordarme...
```

Fig. 6. "Don Quijote", with the "tagging_1" approach

3.4 Method 4: "No Text"

This method tries to unveil the categorization power using only structural units, processed in the same way as in the "adding" method. Roughly speaking, it is equivalent to "adding" and then removing textual terms. In Figure 7 we can see the "notext_0" processing of the previous example.

```
_book _book_title _book_author _book_contents
_book_contents_chapter _book_contents_text
```

Fig. 7. "Don Quijote", with the "notext_0" approach

3.5 Method 5: "Text Replication"

The previous methods deal with a structured collection, having no previous knowledge about it. That is to say, they do not take into account the kind of mark, in order to select one action or another. This approach assigns an integer value to each mark, proportional to its informative content for categorization (the higher the value, the more informative). This value is used to *replicate* terms, multiplying their frequencies in a mark by that factor. Notice that only values for structural marks directly containing terms must be supplied.

In the previous example, suppose we assign the following set of replication values:

```
title: 1, author: 0, chapter: 0; text: 2
```

Notice that a value equal to 0 indicates that the terms in that mark will be removed. The resulting text is displayed in Figure 8.

```
El ingenioso hidalgo Don Quijote de la Mancha En En un un lugar
lugar de de La La Mancha Mancha de de cuyo cuyo nombre nombre no
no quiero quiero acordarme acordarme...
```

Fig. 8. "Don Quijote", with the "replication" approach, using values proposed before

This method is very flexible, and it generalizes several ones, as the "only text" approach (one may select 1 for all the replication values). The method consisting of just selecting text from certain marks can be simulated here using 1 and 0 replication values if the text within a given mark is to be considered or not, respectively.

The main drawback of "text replication" is that we need some experience with the collection, in order to build the table of replication values before processing the files.

4 Experimentation

Previous to the production runs, and in order to select the best combinations of classifiers and representations, we have carried out some experiments using only the training set, by means of cross-validation (dividing the training set into 5 parts). The selected evaluation measures are the microaverage and macroaverage breakeven point (BEP) (for *soft categorization*) and microaverage and macroaverage F1 (for *hard categorization*) [10]. In every case, the "only text" representation will be used as a baseline to compare results among different alternatives.

Table 1 displays the replication values used in the experiments with the "text replication" approach, for the different tags. Tags with unspecified replication values are always set to 1.

We have also carried out experiments with some feature/term selection methods. For the naive Bayes model we used a simple method that removes all the terms that appear in less that a specified number of documents. For the OR gate model we used a local selection method (different terms may be selected for different class values) based on computing the mutual information measure between each term and each class variable C_i.

Table 1. Replication values used in the experiments

Tag	id=2	id=3	id=4	id=5	id=8	id=11
conversionwarning	0	0	0	0	0	0
emph2	2	3	4	5	10	30
emph3	2	3	4	5	10	30
name	2	3	4	5	20	100
title	2	3	4	5	20	50
caption	2	3	4	5	10	10
collectionlink	2	3	4	5	10	10
languagelink	0	0	0	0	0	0
template	0	0	0	0	0	0

The results of this preliminary experimentation are displayed in Table 2. In this table, "OR Gate (ML)" means the OR gate classifier using eq. (10); "OR Gate (AP)" is the OR gate classifier using eq. (12); "$\geq i$ docs." means using term selection, where only terms that appear in more than or equal to i documents are selected; "MI" means local term selection using mutual information.

The best classifier for the four performance measures is the OR Gate classifier using the weights in eq. (12); it gets the best results with the "only text" approach, together with a very light term selection method. The simpler version of this OR Gate classifier (the one using maximum likelihood) obtains quite poor results, except if we use a much more aggresive term selection method based on mutual information.

It is a clear fact that the "replication" approach helps the naive Bayes classifier. One of the main drawbacks of this classifier are the generally bad results obtained in macro measures (this is probably due to the nature of the classifier, that benefits the classes with higher number of training examples). This drawback can be alleviated by using a replication approach with moderate replication values.

On the other hand, the "adding" and "tagging" methods do not seem to give good results in combination with any of these probabilistic classifiers. The runs with the "notext" approach were also really disappointing and they are not listed here.

4.1 Official Runs

Finally, we decided to submit to the Document Mining track the five runs described in Table 3. The evaluation measures of the official runs are the microaverage and macroaverage recall (which coincide in this case with the microaverage

Table 2. Results of the preliminary experimentation with the training set using 5-fold cross-validation

Classifier	Representation	Term Selec.	micro BEP	macro BEP	micro F1	macro F1
Naïve Bayes	Only text	None	0.76160	0.58608	0.78139	0.64324
Naïve Bayes	Only text	≥ 2 docs.	0.72269	0.67379	0.77576	0.69309
Naïve Bayes	Only text	≥ 3 docs.	0.69753	0.67467	0.76191	0.68856
Naïve Bayes	Adding_1	None	0.75829	0.56165	0.76668	0.58591
Naïve Bayes	Adding_1	≥ 3 docs.	0.68505	0.66215	0.74650	0.65390
Naïve Bayes	Adding_2	None	0.73885	0.55134	0.74413	0.54971
Naïve Bayes	Adding_2	≥ 3 docs.	0.66851	0.62747	0.71242	0.59286
Naïve Bayes	Adding_3	None	0.71756	0.53322	0.72571	0.51125
Naïve Bayes	Adding_3	≥ 3 docs.	0.64985	0.59896	0.68079	0.53859
Naïve Bayes	Tagging_1	None	0.72745	0.49530	0.72999	0.50925
Naïve Bayes	Tagging_1	≥ 3 docs.	0.65519	0.60254	0.71755	0.60594
Naïve Bayes	Replic. (id=2)	None	0.76005	0.64491	0.78233	0.66635
Naïve Bayes	Replic. (id=2)	≥ 2 docs.	0.71270	0.68386	0.61321	0.73780
Naïve Bayes	Replic. (id=2)	≥ 3 docs.	0.70916	0.68793	0.73270	0.65697
Naïve Bayes	Replic. (id=3)	None	0.75809	0.67327	0.77622	0.67101
Naïve Bayes	Replic. (id=4)	None	0.75921	0.69176	0.76968	0.67013
Naïve Bayes	Replic. (id=5)	None	0.75976	0.70045	0.76216	0.66412
Naïve Bayes	Replic. (id=8)	None	0.74406	0.69865	0.72728	0.61602
Naïve Bayes	Replic. (id=11)	None	0.72722	0.67965	0.71422	0.60451
OR Gate (ML)	Only text	None	0.37784	0.38222	0.59111	0.37818
OR Gate (ML)	Only text	MI	0.74014	0.72816	0.74003	0.68430
OR Gate (AP)	Only text	None	0.79160	0.76946	0.79160	0.74922
OR Gate (AP)	Only text	≥ 3 docs.	0.77916	0.78025	0.77916	0.73544
OR Gate (AP)	**Only text**	**≥ 2 docs.**	**0.79253**	**0.78135**	**0.79253**	**0.75300**
OR Gate (ML)	Adding_1	None	0.40503	0.43058	0.58777	0.39361
OR Gate (ML)	Adding_1	≥ 3 docs.	0.39141	0.41191	0.57809	0.36936
OR Gate (ML)	Adding_1	MI	0.69944	0.72460	0.69943	0.58835
OR Gate (ML)	Adding_2	None	0.40573	0.43335	0.58908	0.39841
OR Gate (ML)	Adding_2	≥ 3 docs.	0.39204	0.41490	0.57951	0.37346
OR Gate (ML)	Adding_2	MI	0.65642	0.70755	0.65642	0.52611
OR Gate (ML)	Notext_2	None	0.40507	0.42914	0.48818	0.38736
OR Gate (ML)	Tagging_1	None	0.37859	0.40726	0.57274	0.35418
OR Gate (ML)	Tagging_1	≥ 3 docs.	0.36871	0.38475	0.56030	0.32546
OR Gate (ML)	Tagging_1	MI	0.59754	0.67800	0.59754	0.39141
OR Gate (AP)	Tagging_1	None	0.73784	0.74066	0.73789	0.70121
OR Gate (ML)	Replic. (id=2)	MI	0.74434	0.73908	0.74432	0.66995
OR Gate (AP)	Replic. (id=2)	None	0.78042	0.76158	0.78042	0.73768
OR Gate (ML)	Replic. (id=3)	MI	0.74612	0.74275	0.74608	0.67249
OR Gate (AP)	Replic. (id=3)	None	0.78127	0.76095	0.78127	0.73756
OR Gate (ML)	Replic. (id=4)	MI	0.74815	0.74623	0.74813	0.67357
OR Gate (AP)	Replic. (id=4)	None	0.78059	0.75971	0.78059	0.73511
OR Gate (ML)	Replic. (id=5)	MI	0.74918	0.74643	0.74916	0.67498
OR Gate (AP)	Replic. (id=5)	None	0.77977	0.75833	0.77978	0.73245
OR Gate (ML)	Replic. (id=8)	MI	0.75059	0.75254	0.75059	0.66702
OR Gate (AP)	Replic. (id=8)	None	0.77270	0.74943	0.77270	0.72186
OR Gate (ML)	Replic. (id=11)	MI	0.72656	0.71326	0.72656	0.64101
OR Gate (AP)	Replic. (id=11)	None	0.73041	0.70260	0.73041	0.66733

and macroaverage F1 mesaures, because only one class may be assigned to each document), whose values are also displayed in Table 3.

Notice that the relative ordering among these classifiers is the same than in the previous table (OR Gate AP > NB > OR Gate ML), and the final evaluation measures are close to the previously presented estimators.

Table 3. Submitted runs

Classifier	Representation	Term Selec.	micro Recall	macro Recall
Naïve Bayes	Only text	None	0.77630	0.58536
Naïve Bayes	Replic. (id=2)	None	0.78107	0.63730
OR Gate (ML)	Replic. (id=5)	MI	0.75354	0.61298
OR Gate (ML)	Replic. (id=8)	MI	0.75097	0.61973
OR Gate (AP)	**Only text**	**≥ 2**	**0.78998**	**0.76054**

5 Concluding Remarks

Our participation in the XML Document Mining track of the INEX 2007 Workshop has been described in this work. This is the first year that we apply for this track but, despite the low number of participants in the Categorization approach, our participation was remarkable. The main relevant results presented here are the following:

- We have described a new approach for flat document classification, the so called "OR Gate classifier", with two different variants: ML estimation, and a more accurate approximation of the required conditional probabilities.
- We have shown different methods of representing structured documents as plain text ones. We must also recall that some of them, particularly the replication method, are new.
- According to the results, we found that we could improve categorization of structured documents using a multinomial naive Bayes classifier, which is widely known and is included in almost every text-mining software package, in combination with the replication method.

On the other hand, the present paper raises the following questions, which can be stated as future lines of research:

- How are the results of our models compared with a SVM (Support Vector Machine) using only the text of the documents?
- Can the naive Bayes classifier be improved more using a more sophisticated feature selection method?
- Having in mind that the replication approach is the one that has given the best results, what are the optimum replication parameters that can be used in Wikipedia? In other words, what marks are more informative and how much?

- Is there a way to make a representation of the structure of documents that could be used to improve the results of the OR Gate classifier (specially in its more promising AP version)?
- Do the "adding", "tagging" and "no text" approaches help other categorization methods, like, for instance, Rocchio or SVMs?

Managing structure in this problem has been revealed as a difficult task. Besides, it is not really clear if the structure can make a good improvement of categorization results. So, we hope to start answering the previous questions in future editions of this track.

Acknowledgments. This work has been jointly supported by the Spanish Consejería de Innovación, Ciencia y Empresa de la Junta de Andalucía, Ministerio de Educación y Ciencia and the research programme Consolider Ingenio 2010, under projects TIC-276, TIN2005-02516 and CSD2007-00018, respectively.

References

1. Acid, S., de Campos, L.M., Castellano, J.G.: Learning Bayesian network classifiers: searching in a space of acyclic partially directed graphs. Machine Learning 59(3), 213–235 (2005)
2. Bratko, A., Filipic, B.: Exploiting structural information for semi-structured document categorization. Information Processing and Management 42(3), 679–694 (2006)
3. de Campos, L.M., Huete, J.F.: A new approach for learning belief networks using independence criteria. International Journal of Approximate Reasoning 24(1), 11–37 (2000)
4. Koller, D., Sahami, M.: Hierarchically classifying documents using very few words. In: Proceedings of the Fourteenth International Conference on Machine Learning, pp. 170–178 (1997)
5. Larkey, L.S., Croft, W.B.: Combining classifiers in text categorization. In: Proceedings of the 19th International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 289–297 (1996)
6. Lewis, D., Gale, W.: A sequential algorithm for training text classifiers. In: Proceedings of the 17th International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 3–12 (1994)
7. McCallum, A., Nigam, K.: A Comparison of event models for Naive Bayes text classification. In: AAAI/ICML Workshop on Learning for Text Categorization, pp. 137–142. AAAI Press, Menlo Park (1998)
8. Pearl, J.: Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan and Kaufmann, San Mateo (1988)
9. Robertson, S.E., Sparck-Jones, K.: Relevance weighting of search terms. Journal of the American Society for Information Science 27, 129–146 (1976)
10. Sebastiani, F.: Machine Learning in automated text categorization. ACM Computing Surveys 34, 1–47 (2002)