Journal of Theoretical and Applied Information Technology
10th September 2013. Vol. 55 No.1
© 2005 - 2013 JATIT & LLS. All rights reserved·
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

WEB CLASSIFICATION APPROACH USING REDUCED VECTOR REPRESENTATION MODEL BASED ON HTML TAGS

[1]ABDELBADIE BELMOUHCINE, [2]ABDELLAH IDRISSI, [3]MOHAMMED BENKHALIFA
Computer Science Laboratory, Computer Science Department, Faculty of Sciences, Mohammed V Agdal University, Rabat, Morocco
E-mail: [1]belmouhcine@gmail.com, [2]aidrissi@fsr.ac.ma, [3]khalifa@fsr.ac.ma

ABSTRACT

Automatic web page classification plays an essential role in information retrieval, web mining and web semantics applications. Web pages have special characteristics (such as HTML tags, hyperlinks, etc.…) that make their classification different from standard text categorization. Thus, when applied to web data, traditional text classifiers do not usually produce promising results. In this paper, we propose an approach which categorizes web pages by exploiting plain text and text contained in HTML tags. Our method operates in two steps. In step 1, we use Support Vector Machine classifier (SVM) to generate, for each target web page (page to classify), reduced vector representation based on plain text and text from HTML tags. In Step 2, we submit this vector representation to Naive Bayes (NB) algorithm to determine the final class for the target page. We conducted our experiments on two large datasets of pages from ODP (Open Directory Project) and WebKB (Web Knowledge Base), which are accidentally discovered to suffer from a lot of missing HTML tags. The results prove that NB classifier, supported by our model and using HTML tags content combined with plain text, (1) performs significantly better than NB classifier using text alone in terms of both Micro-F1 and Macro-F1 measures and even with the presence of missing HTML tags, (2) performs consistently with respect to category distribution and (3) outperforms NB classifier, using text alone, simply with the use of very basic handling techniques of missing HTML tags.

Keywords: *HTML Tags, Naive Bayes, Semantic Web, SVM, Web Mining, Web Page Classification*

1. INTRODUCTION

With the fast growth of World Wide Web size, the task of locating relevant information on the web becomes much more complex. The web contains billions of web pages making the need for performant automatic web page classification techniques more noticeable. Web page classification creates new problematic research issues due to rich information enclosed within hypertext. Standard text classifiers usually work on structured data collections but are not concerned by special characteristics possessed by web pages like HTML tags, hyperlinks, etc.….. Research demonstrates that integrating hypertext information along with the content of web pages [1], [2], [3] can provide a rich source of information for web classification.

On the other hand, there are web pages that contain HTML tags treating related but different subjects, which do not necessarily indicate the major class of those web pages. Also, there are some web developers who define HTML tags contents which are irrelevant to categories of web pages so that these later can be top listed by search engines. This poses a serious challenge to classifiers with respect to discriminating informative tags from noisy tags. For instance, a course home page can have course information in title (<title> tag), refers to course teacher(s) home pages in anchors (<a> tag) and talks about students in paragraphs (<p> tag). In this case, <a> and <p> tags provide no indication about web page category, whereas <title> tag seems to be more informative. Given this, the question of how to effectively employ HTML tags contents becomes very crucial to web classification.

In this paper, we address the problem of how HTML tags contents can be utilized to enhance web page categorization. We propose HTML tags content based web page classification approach that proceeds in two phases. In the first phase, we construct seven virtual documents for each target page. We create the first virtual document using web page's text content alone and each one of the six others using text extracted from each of the following HTML tags: <title>, <a>, <strong>, <em>, <h1> and <h2>. Then, we use SVM [4] classifier to obtain a class for each of the seven virtual documents. Next, we represent each web page by a vector (called "Topics" Vector) composed of the seven classes obtained, by SVM. In phase 2, we submit our vector to Naive Bayes [5] classifier to find the final class for the web page. The main contributions of this work can be summarized as follows:

We suggest an approach using 2 classifications: one to better represent the target web page, and another for its final classification. This approach is distinguished by: (1) new technique that uses SVM classifiers to create reduced and more refined vector representation of the target web page. The creation of this vector is based mainly on HTML tags information without any weighting schema for these tags. (2) The use of NB model (for final classification) where the naive part, namely the independence assumption of topics vector components, is experimentally verified thanks to the very small dimension of the representation vector.

We test our approach on two different web corpora (The ODP [6] and WebKB [7]). We accidentally find out that both datasets suffer from the presence of a lot of missing HTML tags. Thus, we consider employing simple handling methods of missing HTML tags for NB classification based on our model.

Our study shows that combining plain text with HTML tags information for NB classification founded on our proposed model, results in: (1) Better performance than using plain text alone, even with existence of missing tags, (2) consistent performance behavior with regard to class distribution and (3) higher performance than using text alone with just very simple HTML tags imputing techniques.

The remainder of the paper is organized as follows. In section 2, we review recent work on using metadata and hypertext information to improve web page classification. In section 3, we present our approach in details. In section 4, we show the experimental setting adopted. In section 5, we present and discuss obtained results. Then, we conclude our work and cite some of our future perspectives.

2. RELATED WORK

Web pages features have been used by several research works as a rich source of information to boost hypertext classification performance.

Fang et al. [8] discussed web page classification using hypertext features such as the text included in the web page, title, headings, URL and anchor text. They investigated five different classification methods based on SVM that use individual or combined features. They showed that combining these features improves classification performance. They also proposed a voting technique that further improves the performance compared to individual classifiers. He and Liu [9] proposed a new approach of web classification using the output from an MFICA (improved feature selection using Independent Component Analysis) as the input for NB classifier. They experimentally demonstrated that their method provides acceptable classification accuracy. Youquan et al. [3] contributed to NB algorithm improvement by considering semi structure nature of web pages. They split HTML tags into "Supervise tags" groups to supervise classification. They introduced new weighting strategy based on these "supervise tags" groups. They concluded that NB classifier performance benefits from their weighting policy. Sun et al. [10] employed SVM algorithm to classify web pages using their content and context. They showed that combination of text with <title> and <a> HTML tags helps improving classification performance, as opposed to the use of text alone. They also proved that this combination assists SVM to perform very well compared to FOIL-PILFS algorithm. Li et al. [11] introduced a new method that calculates the word's mixed weight by the information of HTML tags features, and then mines the classification rules based on the mixed weight to classify web pages. They showed that their approach allows better performance than traditional associated classification methods. Fermandez et al. [12] use NB classifier for a new enriched web page representation. They tried six term weighting functions. Some of them use only text of web pages and are based on frequencies. Others combine several criteria including information from HTML tags. They also compare two different models for Bayesian prior probabilities: the event and the Gaussian model, applied to representations based on frequencies and relevance, respectively. Their experiments showed that, in general, the Gaussian model obtains better results than even model when enriched representation is considered. Ardö and Golub [13] used different methods to derive significance indicators assigned to different web page elements: title, headings, metadata and text. They showed that the best results are obtained when all the elements are considered in the classification process. They also demonstrated that the combination way of these indicators is not very important. Kwon and Lee [14] supplemented KNN approach with a feature selection method and a term weighting scheme using markup tags. They also reformed inter document similarity measure used in the vector space model. They showed that their method improves the performance of KNN classification. Böttcher et al. [15] presented an approach that uses weighting of HTML tags for separating relevant information in hypertext documents from the noise. They evaluated their method on web page classification. They proved that their technique significantly helps classification.

All research work cited above introduce very interesting research ideas on hypertext classification. This paper proposes two stage classification approach. It employs SVM classifiers for web page reduced vector representation. Then, it applies Naive Bayes classifier on this vector representation to find the final class.

3. PROPOSED APPROACH

Web pages authors usually embed very important text within HTML tags such as <title> and </title>, <h1> and </h1> etc.... On the other hand, these web pages have a complex structure where plain text and HTML tags contents may talk about related but different subjects. Even though, these later may not be the same, they all inform to the final page with various contribution degrees. Our approach is motivated by simply considering these possibly different subjects towards final classification of the target page. This motivation is driven by two curiosities about these subjects: (1) how they can be exploited to best represent the target web page (2) how they can be made optimally informative to the right class of the target page. In this work, we construct our topics using plain text part and text from each of the six following HTML tags: <title>, <a>, <strong>, <em>, <h1> and <h2>.

Our technique operates in two stages. First, we construct, for each target page, seven virtual documents (VDs) where the first document is based on plain text alone, and the six others are created from text extracted respectively from <title>, <a>, <strong>, <em>, <h1> and <h2>. Then, we separately submit these VDs to SVM classifier to obtain seven classes (called "Topics"). These topics, which are not necessarily the same, form a vector ("Topics" Vector) that represents the target page. Second, we supply our "Topics" vector to NB classifier in order to compute the final class.

3.1 "Topics" Vector Representation Using Plain Text And HTML Tags

As mentioned earlier, we create seven virtual documents for the target web page. One VD contains terms extracted from plain textual part. Each one, of the six other VDs, is created from text included in one pair of the six HTML tags (<title></title>; <a></a>; <strong></strong>; <em></em>; <h1></h1> and <h2></h2>). We define our VDs as follows:

Let's denote: P: target web page and t: term of P.

$$VD_X(P) = \{t | t \in P.X\}$$
$$VD_T(P) = \{t | t \in P.title\}$$
$$VD_A(P) = \{t | t \in P.a\}$$
$$VD_S(P) = \{t | t \in P.strong\}$$
$$VD_E(P) = \{t | t \in P.em\}$$
$$VD_{H1}(P) = \{t | t \in P.h1\}$$
$$VD_{H2}(P) = \{t | t \in P.h2\}$$

Where P.X is the plain text of P and P.title, P.a, P.strong, P.em, P.h1 and P.h2, are the texts, in P, included respectively in tags <title>, <a>, <strong>, <em>, <h1> and <h2>.

To find the most adapted algorithm to classify each VD, we experimentally consider several techniques (like K-nearest neighbors algorithm (KNN) [16]). We opted for SVM algorithm that demonstrates high performance in such type of classification [8], [10], [17]. We applied SVM classification for each VD representation of the target web page as follows (table 1):

Table 1: "Topics" created using SVM classifier

VD Representation	SVM classifier	Class obtained (Topic)
$VD_X(P)$	SVM(X)	c_X
$VD_T(P)$	SVM(T)	c_T
$VD_A(P)$	SVM(A)	c_A
$VD_S(P)$	SVM(S)	c_S
$VD_E(P)$	SVM(E)	c_E
$VD_{H1}(P)$	SVM(H1)	c_{H1}
$VD_{H2}(P)$	SVM(H2)	c_{H2}

Finally, we generate "TOPICS" vector from the seven classes obtained previously:

$$V_{Topics} = \left(c_{X,} c_T, c_A, c_S, c_E, c_{H1}, c_{H2}\right)$$

3.2 Naive Bayes Classification

In this step, we use Naive Bayes (NB) classifier to compute the final class of the target page represented by "TOPICS" vector (V_{Topics}). NB is a simple and effective text classification algorithm which has been shown to perform very well in practice [18], [19]. Figure 1 provides a summary of our approach.

4. EXPERIMENTS

In this section, we present the experiments that we have conducted in order to validate the advantage of combining HTML tags text and plain text in web page classification task. We describe the experimental design adopted, i.e. the preprocessing techniques utilized, the classifiers applied, the datasets employed, the performance evaluation metrics used and the results obtained based on those measures with the discussion.

4.1 Experimental Setup

4.1.1 Pre-processing

In first stage of our approach, we separately applied a number of preprocessing techniques to each of the seven virtual documents of the target web page. These techniques aim at cleaning and normalizing the raw text contained in these virtual documents. In tokenization step, we converted all words tokens to lower cases to avoid the problem of case differences. We removed from words some special characters, punctuation marks, scripts, styles, mimes headings and numbers. We also eliminated stop words and terms that occur less than twice. As stemming process, we applied Porter method [20]. After preprocessing stage, we proceed to the dictionary building. In our case, we consider web pages as bags of words. Therefore, our dictionary consists of words resulting from pre-processing. Then, we represent our web pages by the use of the conventional Vector Space Model [21]. We map each virtual document onto its vector $VD_j = \left(n_{1j}, n_{2j}, ..., n_{mj}\right)$; where n_{ij} denotes the weight of term i in virtual document j. We adopted the Inverse Document Frequency (IDF) [22] based weighting model to obtain the weights:

$$n_{ij} = IDF_i = \log(\frac{N}{df_i})$$

Where N is the total number of target web pages and df_i is the number of virtual documents where term i appears.

In second stage, we did not perform any preprocessing for "TOPICS" vectors representing our target web pages. We systematically submitted these vectors to NB classifier.

4.1.2 Classifiers used

We employed two different classifiers, namely Support Vector Machines [4] for step 1 and Naive Bayes [5] for step 2. We provide a succinct description for each of these algorithms below.

4.1.2.1 Support vector machine (SVM)

SVM is a powerful learning technique that has been successfully applied to text classification [17]. It is based on the Structured Risk Maximization theory, which aims at minimizing the generalization error instead of the experimental error on training data alone. Multiple versions of SVM have been described in [23]. In this work, we used Sequential Minimal Optimization version which was developed in [24], [25]. We empirically adjust C parameter, which defines tolerance degree to errors, to the value $C=2^{-4}$ maximizing both Micro-F1 and Macro-F1 measures on datasets used. On the other hand, we use a linear kernel, which proves to be efficient for text categorization where we have high feature vector dimension [17]. This is the case in both ODP and WebKB datasets which count more than 28678 terms and 17215 terms respectively.

4.1.2.2 Naive Bayes (NB)

NB is a simple and very popular text classification algorithm [18], [19]. Its basic principle is to use the joint probabilities of features and categories to estimate the probabilities of categories given a document. In this work, we applied Bayes's rule to use the joint probabilities of "Topics" (provided by SVM classifiers) and categories to estimate the probabilities of category given a web page.

Having a web page P_i represented by a topics vector

$$V_{i_{Topics}} = (c_{i1}, c_{i2}, \ldots, c_{i7}),\ where\ c_{ik} = \begin{cases} c_{iX}\ if\ k=1 \\ c_{iT}\ if\ k=2 \\ c_{iA}\ if\ k=3 \\ c_{iS}\ if\ k=4 \\ c_{iE}\ if\ k=5 \\ c_{iH1}\ if\ k=6 \\ c_{iH2}\ if\ k=7 \end{cases}$$

Journal of Theoretical and Applied Information Technology
10th September 2013. Vol. 55 No.1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

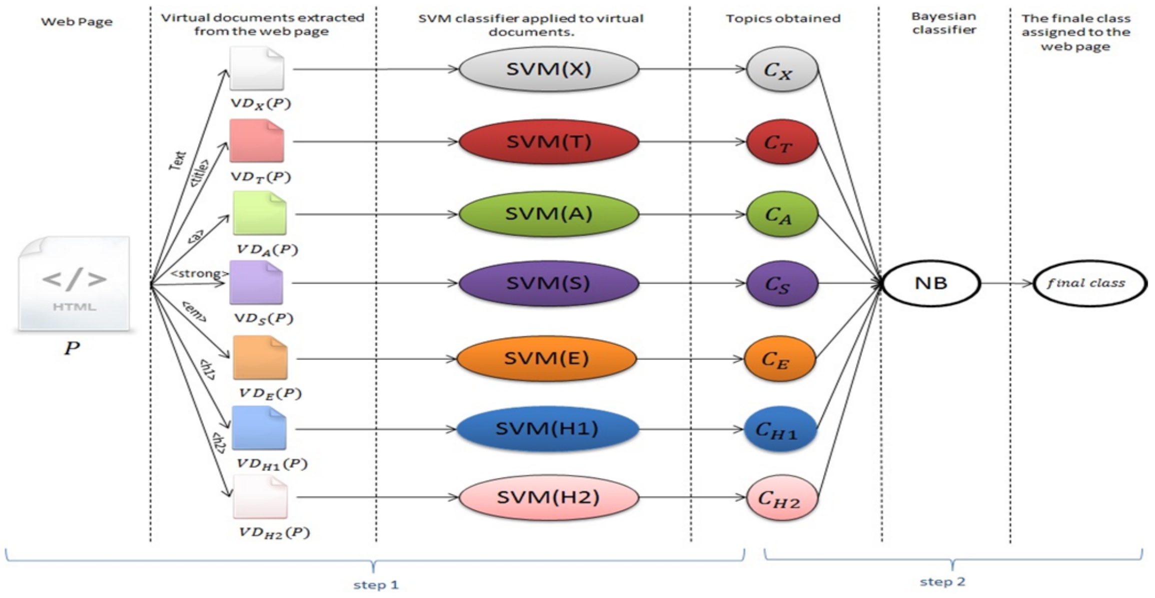

Figure 1: The summary of the proposed approach

These probabilities can be formalized as follows:

$$P\left(c_j|V_{i_{Topics}}\right)=\frac{P(c_j)\prod_{k=1}^{7}P\left(c_{ik}|c_j\right)}{\sum_{r=1}^{|C|}P(c_r)\prod_{k=1}^{7}P\left(c_{ik}|c_r\right)}$$

c_j refers to the category c_j.

$V_{i_{Topics}}$ is the topics vector representing the web page P_i.

c_{ik} is the kth component of $V_{i_{Topics}}$.

$|C|$ is the total number of classes.

To estimate posterior probabilities, we used the m-estimate model [26].

$$P\left(c_{ik}|c_j\right)=\frac{(\text{The number of topics vectors of the category } c_j \text{ with } c_{ik})+1}{(\text{The total number of topics vectors belonging to the category } c_j)+7}$$

$$P\left(c_j\right)=\frac{\text{The number of topics vectors belonging to the category } c_j}{\text{The total number of topics vectors}}$$

Note that we used $m=7$ and $p=\frac{1}{7}$ as prior probability.

In this work, reduced dimension of topics vector (7 components) made it possible to experimentally verify Bayesian model's naive part, which is the assumption of topics independence. Thus, we experimentally show that the conditional probability of each topic given a category is independent from the conditional probabilities of the other 6 topics given that category. For each data collection used and for each category, we consider 21 pairs (i.e. C_7^2) of topics. We proved experimentally that: For each pair of topics $(c_{ik},c_{ik'})$ where k=1, 2,…, 7 and k'=1, 2,…, 7 such that k≠k' and for each category c_j we have :

$$P\left(c_{ik},c_{ik'}|c_j\right)\approx P\left(c_{ik}|c_j\right)\times P\left(c_{ik'}|c_j\right)$$

The experimental differences between these probabilities are on average between 0.002 and 0.008 for WebKb and between 0.001 and 0.002 for ODP.

4.1.3 Datasets

In this article, we used 2 different data collections to evaluate our approach. The first dataset is taken from the Open Directory Project (ODP) and the second is extracted from the "Web Knowledge Base" (WebKB).

The ODP [6] is a huge repository containing around 4.6 million URLs and 765,282 categories and subcategories [27]. We extracted our dataset from the category "Top/Health/Conditions_and_Diseases", which counts 21 subcategories with 6 of them, sufficiently populated for our experiments. The 15 others were treated as subcategory "Others". In this work, we conducted our experiments on the 6 most populated subcategories and the subcategory "Others". Hence, the seven categories considered, are:

- Cancer
- Cardiovascular_Disorders
- Immune_Disorders

- Infectious_Diseases
- Musculoskeletal_Disorders
- Neurological_Disorders
- Other

The WebKB [7] is a collection of web pages about computer science departments of different American universities (Such as Cornell, Texas, Washington, Wisconsin and others). It contains 8282 web pages and counts 7 categories: Student (1641 pages), Faculty (1124 pages), Course (930 pages), Project (504 pages), Department (182 pages), Staff (137 pages) and Other (3764 pages). The category “Other” contains pages instances of other categories. To avoid any possible bias to classifiers, we replace this category’s content with “Staff “and “Department” categories. In this paper, we consider 5 categories for our experiments: the 4 most populated categories “Student”, “Faculty”, “Project“, “Course” and “Other” category, which was replaced by the 2 least populated categories “department” and “Staff”. Therefore, the categories used here are:

- Student
- Faculty
- Course
- Project
- Other

Our approach proceeds in 2 stages, where we apply 2 learning classifiers that need training and test datasets. The output “Topics” vector of the first stage is used as an input for the second stage. Consequently, we carefully prepare a data partitioning protocol for each data collection. We divide, each data corpus, into 3 disjoint datasets as follows:

1. Dataset (Tr-set1) to train SVM classifiers.
2. Dataset (Test-set1) to test SVM classifiers in order to generate V_{Topics} vectors, which are employed as training set (denoted as VTopics-Tr) for NB classifier.
3. Dataset (Test-set2) to test our entire approach: Test-set2 is used to test SVM in order to create V_{Topics} vectors. These later are utilized as test set (VTopics-test) for NB to generate final classes. Figure 2 outlines our data partitioning workflow.

For both datasets, all experiments are conducted using cross validation [28] to reduce the uncertainty of data split between training and test data. These cross validations are performed according to our partitioning protocol described above. Hence, for WebKB, we used leave one university out cross validation. We employed respectively 2 universities, 1 university and 1 university for Tr-set1, Test-set1 and Test-set2 datasets. For ODP, we used 10 fold cross validation. We split our data into 10 arbitrary equal subsets. For each run, we utilize respectively 8 subsets, 1 subset and 1 subset for Tr-set1, Test-set1 and Test-set2 datasets.

4.1.4 Missing HTML tags treatment

In general, the presence of missing values in a dataset may affect the performance of a classifier constructed using that dataset as a training sample [29]. In this work, we observe that in both datasets and for all categories web pages have missing HTML tags with different proportions. Table 2 provides for each dataset the averages of missingness ratios of HTML tags in web pages.

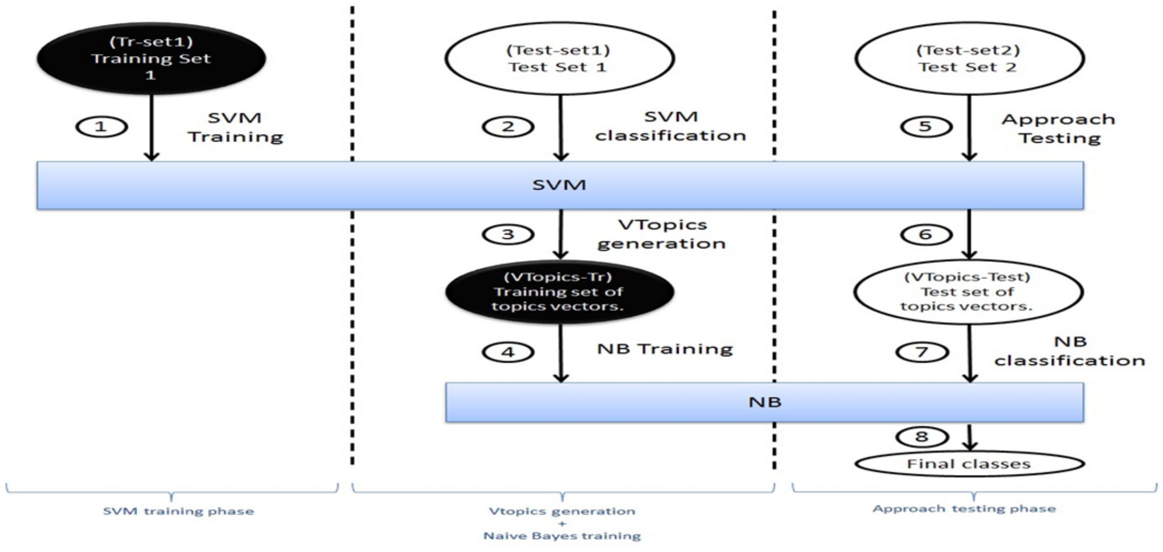

Figure 2: Data partitioning workflow

Table 2: The Averages Of Missingness Ratios Of HTML Tags In Both Datasets

Dataset	<title>	<a>	<strong>	<em>	<h1>	<h2>
ODP	6%	14%	77%	90%	85%	85%
WebKB	11%	8%	94%	75%	39%	42%

Acuña and Rodriguez [29] state that if the rate of missing values is relatively high, then any kind of interpretation on classification may be severely impacted. On the other hand, Farhangfar et al. state in [30] that Naive Bayes can be a resistant classifier to missing data. In this paper, we presume that handling method of missing HTML tags is worth considering. This presumption is motivated mainly by the checking of the coherence of our proposed model with respect to NB classification performance in the presence of missing data. In this work, we use three basic techniques to treat missing HTML tags. We employ 2 imputation methods that replace missing tags with estimated values based on information available in web pages. Each one of them deals with a different case of HTML tags missingness. Also, we use another method that just ignores missing tags. These 3 techniques are described below:

1. NB(SVM+Tags)-Text: It imputes missing HTML tags during the preprocessing. Its rationale is that the plain text, of a web page, is most likely to be the best informative part to the correct class for the missing tag in that web page. Thus, when a tag is initially missing, in a web page, it is replaced by plain text content. Then, the web page part (which relates to missed tag) is preprocessed using plain text in order to create the corresponding virtual document. It is very important to note that, this technique has the main advantage of being independent of the learning algorithm since it imputes HTML tags during preprocessing step.
2. NB(SVM+Tags)-Text+Max: It replaces missing HTML tags after the preprocessing by operating on V_{Topics} vector. It is based on the fact that the class with the highest frequency (among six other topics in vector V_{Topics}) is most probable to be the best estimator of missing topic (class) (due to missing tag). Therefore, when an HTML tag initially exists, in web page, but disappears after preprocessing (due to low frequencies of its terms) there will be no virtual document that relates to this tag. Then, the obtained V_{Topics} vector will have a missing topic that corresponds to that tag. This missing topic is replaced by the topic (among other six classes) with maximum frequency within V_{Topics}. It is essential to mention that this case rarely happens for both datasets used.
3. NB(SVM+Tags)-NoRepalcement: It does not replace any missing HTML tag in any of the 2 cases described above. It just ignores it. Consequently, we submit, to NB classifier, our V_{Topics} vector with a missing component (topic) that relates to that missing tag.

4.1.5 Evaluation measures

To evaluate the performance of both classifiers, we employ the standard metrics: recall, precision and F1, which are commonly used in the classification task. Recall is defined to be the ratio of correct assignments by the system divided by the total correct assignments. Precision is the proportion of correct assignments by the system within the total number of the system's assignments. F1, introduced by Van Rijsbergen [31] is the equally weighted average of recall and precision as stated below:

$$\mathrm{F1} = \frac{2 \times \mathrm{Precision} \times \mathrm{Recall}}{\mathrm{Precision} + \mathrm{Recall}}$$

To measure global averages across multiple categories, we used two averaging techniques: Micro-averaging and Macro-averaging. The former, which has been widely used in cross-method comparisons, assigns equal weight to every document. The latter, which has been used in some cases [32], gives equal weight to every category, regardless of its frequency. The micro-averaged scores are dominated by the classifier performance on most populated categories, whereas the macro-averaged scores are influenced more by rare categories. In our experiments, both kinds of scores are used to test the effectiveness of classification.

Table 3: Micro and macro F1 performance averages

	Macro-F1 measure	Micro-F1 measure
NB(Tags+SVM)-Text+Max	0.617	0.666
NB(Tags+SVM)-Text	0.617	0.666
NB (Tags+ SVM)-NoReplacement	0.596	0.665
NB (TextOnly)	0.419	0.427

(a) Results on ODP dataset

	Macro-F1 measure	Micro-F1 measure
NB(Tags+SVM)-Text+Max	0.839	0.873
NB(Tags+SVM)-Text	0.84	0.874
NB (Tags+ SVM) - NoReplacement	0.821	0.865
NB (TextOnly)	0.687	0.75

(b) Results on WebKB dataset

Table 4: Performance summary of different NB classifiers

	Cancer			Cardiovascular_Disorders			Immune_Disorders			Infectious_Diseases			Musculoskeletal_Disorders			Neurological_Disorders		
	P	R	F-1	P	R	F-1	P	R	F-1	P	R	F-1	P	R	F-1	P	R	F-1
NB(Tags+SVM)-Text+Max	0.89	0.847	**0.866**	0.685	0.63	**0.639**	0.659	0.637	**0.636**	0.7	0.49	**0.505**	0.544	0.425	**0.457**	0.634	0.585	**0.599**
NB(Tags+SVM)-Text	0.89	0.847	**0.866**	0.685	0.63	**0.638**	0.66	0.64	**0.638**	0.698	0.489	**0.503**	0.544	0.425	**0.457**	0.633	0.586	**0.599**
NB (Tags+ SVM) - No Replacement	0.903	0.848	**0.873**	0.72	0.586	**0.622**	0.758	0.565	**0.632**	0.739	0.416	**0.463**	0.652	0.317	**0.396**	0.635	0.574	**0.592**
NB(TextOnly)	0.636	0.579	**0.581**	0.701	0.375	**0.466**	0.143	0.668	**0.229**	0.573	0.536	**0.509**	0.378	0.289	**0.317**	0.528	0.354	**0.411**

(a) Results on ODP dataset

	Course			Faculty			Project			Student		
	P	R	F-1	P	R	F-1	P	R	F-1	P	R	F-1
NB(Tags+SVM)-Text+Max	0.955	0.954	**0.952**	0.694	0.809	**0.745**	0.8	0.74	**0.764**	0.902	0.889	**0.895**
NB(Tags+SVM)-Text	0.955	0.954	**0.951**	0.692	0.827	**0.752**	0.789	0.74	**0.76**	0.906	0.886	**0.895**
NB(Tags+ SVM)-NoReplacement	0.95	0.941	**0.942**	0.722	0.745	**0.729**	0.772	0.72	**0.723**	0.851	0.932	**0.889**
NB (TextOnly)	0.921	0.836	**0.874**	0.516	0.61	**0.556**	0.396	0.725	**0.507**	0.863	0.766	**0.811**

(b) Results on WebKB dataset

4.2 Results And Discussion

In this section, we show the experimental results obtained on both datasets. In these results, we discuss 5 points relating to NB classification (supported by our approach) contrasted to NB classification based on text alone. The first point pertains to global performance comparisons. The 2nd and 3rd points concern performance variation with regard to distribution of classes. Finally, the 4th and 5th points relate to performance behavior with respect to missing tags handling.

- For both ODP and WebKB datasets, as shown in tables 3 and 4, the 3 NB classifiers based on our proposed model (NB(SVM+Tags)-Text; NB(SVM+Tags)-Text+Max; NB(SVM+Tags)-NoReplacement) outperform NB(TextOnly) for all evaluation measures used. Table3 shows that any of the F1 micro average scores and any of the F1 macro average scores of the 3 NB classifiers are respectively higher than the F1 micro average score and the F1 macro average score of NB(TextOnly). Also, table 4 confirms that, for any category, the F1 score of any of the 3 NB classifiers is bigger than the F1 score of NB(TextOnly). This observation reinforces the fact that introducing HTML tags with plain text using our reduced vector representation model helps NB classifier improve performance compared to using text alone. Exceptionally, for Infectious_Diseases (which has very small number of positive training data "291 web pages"), the F1 score of NB(TextOnly) (0.509) is higher than every one of the F1 scores of NB(SVM+Tags)-NoReplacement (0.463), NB(SVM+Tags)-Text (0.503) and NB(SVM+Tags)-Text+Max (0.505). This may be explained by the presence of (1) missing HTML tags in this category (<title>: 8%; <a>: 12%; <strong>: 68%; <em>: 79%; <h1>: 86% and <h2>: 80%), and (2) noise in the category vocabulary, especially for <title>, which is the most present tag for this category. Also, it is worth mentioning that imputing missing HTML tags helps NB classifier based on our proposed model (i.e. NB(SVM+Tags)-Text; and NB(SVM+Tags)-Text+Max) have their F1 performances very close to F1 performance of NB(TextOnly).

Table 5: Averages of F1-performance improvement percentages over NB(TextOnly)

	Categories	NB(SVM+Tags)-Text+Max	NB(SVM+Tags)-Text	NB(SVM+Tags)-NoReplacement	NB(TextOnly)	APIP* over NB(TextOnly)
Common categories	Cancer	0.866	0.866	0.873	0.581	**49.45 (%)**
	Cardiovascular_Disorders	0.639	0.638	0.622	0.466	**35.84 (%)**
	Neurological_Disorders	0.599	0.599	0.592	0.411	**45.17 (%)**
Rare categories	Immune_Disorders	0.636	0.638	0.632	0.229	**177.44 (%)**
	Infectious_Diseases	0.505	0.503	0.463	0.509	**-3.67 (%)**
	Musculoskeletal_Disorders	0.457	0.457	0.396	0.317	**37.75 (%)**

(a) Results on ODP dataset

	Categories	NB(SVM+Tags)-Text+Max	NB(SVM+Tags)-Text	NB(SVM+Tags)-NoReplacement	NB(TextOnly)	APIP*over NB(TextOnly)
Common categories	Course	0.952	0.951	0.942	0.874	**8.50 (%)**
	Faculty	0.745	0.752	0.729	0.556	**33.45 (%)**
	Student	0.895	0.895	0.889	0.811	**10.11 (%)**
Rare categories	Project	0.764	0.76	0.723	0.507	**47.73 (%)**

(b) Results on WebKB dataset

**APIP= Average of Performance Improvement Percentages*

- From table 5, we observe that our proposed model aids NB classifier achieve, on the average, higher F1-performance improvement percentages for rare categories than for common categories for both datasets. In ODP, the average of the F1- performance improvement percentages (70.51 (%)) for rare categories (i.e. Immune_Disorders, infectious_Diseases and Musculoskeletal_Disorders) is larger than the corresponding average (43.49 (%)) for common categories (i.e. Neurological_Disorders, Cancer and cardiovascular_Disorders). Similarly in WebKB, the average of the F1-performance improvement percentages (47.73 (%)) for rare categories (There is only one rare category: i.e. Project) is greater than the equivalent average (17.36 (%)) for common categories (Student, Faculty and course). One possible reason may lie in the fact that globally common categories suffer from much more missing HTML tags than rare categories. This suggests that our proposed model, which uses HTML tags with text, helps NB classifier better compensate the lack of training information in initial data within infrequent categories than within frequent categories.

performance remains consistent with respect to category Distribution.

- From table 7, we notice that the averages of performance improvement percentages (for F1 micro and F1 macro averages) of the 2 NB classifiers (i.e. NB(SVM+Tags)-Text and NB(SVM+Tags)-Text+Max) over NB(Text Only) are significantly interesting for each dataset. This shows that very simple imputation methods (used in this paper) seem to be significantly efficient to estimate missing tags for NB classifier based on our proposed schema. Thus, our proposed approach to NB classifier suitably operates with very basic missing data handling techniques, which are suggested in this paper.

Table 6: Averages of F1 performances with respect to category distribution

	NB(SVM+Tags)-Text+Max	NB(SVM+Tags)-Text	NB(SVM+Tags)-NoReplacement
Common categories	0.701	0.701	0.696
Rare categories	0.533	0.533	0.497

(a) Results on ODP dataset

	NB(SVM+Tags)-Text+Max	NB(SVM+Tags)-Text	NB(SVM+Tags)-NoReplacement
Common categories	0.864	0.866	0.853
Rare categories	0.764	0.76	0.723

(b) Results on WebKB dataset

- For both datasets, as shown in table 6, the 3 NB classifiers NB(SVM+Tags)-Text, NB(SVM+Tags)-Text+Max and NB(SVM+Tags)-NoReplacement show higher performances average for common categories than for rare categories. For ODP, the 3 NB classifiers achieve an average F1 performance score of 0.699 on frequent classes (i.e. Neurological_Diorders, Cancer and Cardiovascular_Disorders). Whereas, they only show an average F1 score of 0.521 on non-frequent classes (i.e. Immune_Disorders, Infectious_Disorders, and Musculoskeletal_Disorders). For WebKB, the 3 NB classifiers achieve an average F1 performance score of 0.861 on common classes (i.e. Course, Faculty and Student), and an average F1 score of 0.749 on rare classes (i.e. Project). This shows that our proposed model allows NB classifier performance remains consistent with respect to category Distribution.

- From table 3 and 4, we note that NB(SVM+Tags)-NoReplacement achieves better results, in terms of all evaluation measures used, than NB(TextOnly) for both datasets. On one hand, with the presence of missing HTML tags reaching relatively high averages of missingness rates (as shown in table 2). On the other hand, without using any missing data handling approach, NB classifier (based on our model) succeeds to outperform NB(TextOnly). This illustrates that our approach permits NB classifier to be resistant to missing data. Nevertheless, this observation does not apply to Infectious_Disorders category in ODP dataset due to reasons stated earlier in the first point of this section.

Table 7. Averages of F1-performance improvement percentages over NB(TextOnly)

	NB(SVM+Tags)-Text+Max	NB(SVM+Tags)-Text	AIPP* over NB(TextOnly)
Macro F1 measure	0.617	0.617	**47.26 (%)**
Micro F1 measure	0.666	0.666	**55.97 (%)**

(a) Results on ODP dataset

	NB(SVM+Tags)-Text+Max	NB(SVM+Tags)-Text	AIPP* over NB(TextOnly)
Macro F1 measure	0.84	0.839	**22.20%**
Micro F1 measure	0.874	0.873	**16.47%**

(b) Results on ODP dataset

**APIP= Average of Performance Improvement Percentages*

5. CONCLUSION AND FUTURE PERSPECTIVES

This paper has proposed a new method to NB web classification. This approach uses a new reduced vector representation model, based on HTML tags and plain text, which is created by SVM classifier. The experimental results show that, under appropriate empirically setting, our schema provides significant improvements for NB web classification. Our major findings include:

- Introducing HTML tags combined with plain text using our reduced representation vector model assists NB classification improve its performance compared to using plain text alone.
- Our proposed schema helps NB classifier make, on the average, higher performance improvement ratio for rare categories than for common categories.
- The performance of NB classifier, supported by our approach, keeps the same behavior with respect to category distribution as the performance of a standard learning classifier. This latter supplies better performance for frequent classes than for infrequent classes.
- Very basic missing data handling techniques, which are proposed in this paper, work efficiently enough with NB classifier based on our model.
- The model we suggested to NB classifier allows this later to be resistant to missing data.

While the framework of the experimental results presented is necessarily restrained to our datasets, we hope that our analysis will assist future research into hypertext classification. This assistance pertains to: (1) Supplying some ideas about exploring different ways to design and refine web page representation vector model, including hypertext information so that it better suits web classifiers. (2) How can web classifiers be built or combined to take profit of this model.

While our results are promising, there are still many improvements to be made. In the future, we plan to use information provided by the neighborhood and Hyperlinks along with our model for web page classification. We will also intend to explore, from HTML tags, more useful semantic information for web page classification.

REFERENCES

[1] D. Shen, J.-T. Sun, Q. Yang, and Z. Chen, "A comparison of implicit and explicit links for web page classification," in *Proceedings of the 15th international conference on World Wide Web*, New York, NY, USA, 2006, pp. 643–650.

[2] R. Ghani, S. Slattery, and Y. Yang, "Hypertext Categorization using Hyperlink Patterns and Meta Data," in *Proceedings of the Eighteenth International Conference on Machine Learning*, San Francisco, CA, USA, 2001, pp. 178–185.

[3] H. Youquan, X. Jianfang, and X. Cheng, "An improved Naive Bayesian algorithm for Web page text classification," in *2011 Eighth International Conference on Fuzzy Systems and Knowledge Discovery (FSKD)*, 2011, vol. 3, pp. 1765 –1768.

[4] C. Cortes and V. Vapnik, "Support-Vector Networks," *Mach Learn*, vol. 20. no. 3, pp. 273–297, Sep. 1995.

[5] G. H. John and P. Langley, "Estimating continuous distributions in Bayesian classifiers," in *Proceedings of the Eleventh conference on Uncertainty in artificial intelligence*, San Francisco, CA, USA, 1995, pp. 338–345.

[6] "ODP - Open Directory Project." [Online]. Available: http://www.dmoz.org/. [Accessed: 24-Feb-2013].

[7] "The 4 Universities Data Set." [Online]. Available: http://www.cs.cmu.edu/afs/cs.cmu.edu/project/theo-20/www/data/. [Accessed: 24-Feb-2013].

[8] R. Fang, A. Mikroyannidis, and B. Theodoulidis, "A Voting Method for the Classification of Web Pages," in *Proceedings of the 2006 IEEE/WIC/ACM international conference on Web Intelligence and Intelligent Agent Technology*, Washington, DC, USA, 2006, pp. 610–613.

[9] Z. He and Z. Liu, "A Novel Approach to Naive Bayes Web Page Automatic Classification," in *Fifth International Conference on Fuzzy Systems and Knowledge Discovery, 2008. FSKD '08*, 2008, vol. 2, pp. 361–365.

[10] A. Sun, E.-P. Lim, and W.-K. Ng, "Web classification using support vector machine," in *Proceedings of the 4th international workshop on Web information and data management*, New York, NY, USA, 2002, pp. 96–99.

[11] X. Li, J. Lan, and H. Shi, "Associative web document classification based on word mixed weight," in *2010 3rd IEEE International Conference on Computer Science and*

Information Technology (ICCSIT), 2010. vol. 3, pp. 578–581.

[12] V. F. Fernandez, S. M. Herranz, R. M. Unanue, and A. C. Rubio, "Naive Bayes Web Page Classification with HTML Mark-Up Enrichment," in *International Multi-Conference on Computing in the Global Information Technology, 2006. ICCGI '06*, 2006, pp. 48–48.

[13] K. Golub and A. Ardö, "Importance of HTML structural elements and metadata in automated subject classification," in *Proceedings of the 9th European conference on Research and Advanced Technology for Digital Libraries*, Berlin, Heidelberg, 2005, pp. 368–378.

[14] O.-W. Kwon and J.-H. Lee, "Web page classification based on k-nearest neighbor approach," in *Proceedings of the fifth international workshop on on Information retrieval with Asian languages*, New York, NY, USA, 2000. pp. 9–15.

[15] S. Böttcher, L. Werner, and R. Beckmann, "Enhanced Information Retrieval by Using HTML Tags," in *DMIN'05*, 2005, pp. 24–29.

[16] T. Cover and P. Hart, "Nearest neighbor pattern classification," *Ieee Trans. Inf. Theory*, vol. 13, no. 1, pp. 21–27, 1967.

[17] T. Joachims, "Text categorization with Support Vector Machines: Learning with many relevant features," in *Machine Learning: ECML-98*, C. Nédellec and C. Rouveirol, Eds. Springer Berlin Heidelberg, 1998, pp. 137–142.

[18] A. McCallum and K. Nigam, *A comparison of event models for Naive Bayes text classification*. 1998.

[19] T. M. Mitchell, *Machine Learning*, 1st ed. McGraw-Hill Science/Engineering/Math, 1997.

[20] M. F. Porter, "Readings in information retrieval," K. Sparck Jones and P. Willett, Eds. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1997, pp. 313–316.

[21] G. Salton and M. J. McGill, *Introduction to Modern Information Retrieval*. New York, NY, USA: McGraw-Hill, Inc., 1986.

[22] K. S. Jones, "A statistical interpretation of term specificity and its application in retrieval," *J. Doc.*, vol. 28, pp. 11–21, 1972.

[23] C.-J. Lin, "Asymptotic convergence of an SMO algorithm without any assumptions," *Ieee Trans. Neural Networks*, vol. 13, no. 1, pp. 248–250. 2002.

[24] J. C. Platt, "Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines," ADVANCES IN KERNEL METHODS - SUPPORT VECTOR LEARNING, 1998.

[25] S. S. Keerthi, S. K. Shevade, C. Bhattacharyya, and K. R. K. Murthy, "Improvements to Platt's SMO Algorithm for SVM Classifier Design," *Neural Comput.*, vol. 13, no. 3, pp. 637–649, Mar. 2001.

[26] B. Cestnik, "Estimating probabilities: A crucial task in machine learning," presented at the Proc. of the European Conference on Artificial Intelligenc, 1990. pp. 147–149.

[27] L. Henderson, "Automated Text Classification in the DMOZ Hierarchy." 06-Nov-2009.

[28] F. Mosteller and J. W. Tukey, "Data Analysis, Including Statistics," in *Handbook of Social Psychology (G. Lindzey and E. Aronson, eds.)*, 2nd ed., vol. 2, Addison-Wesley, Reading, MA, pp. 80–203.

[29] E. Acuña and C. Rodriguez, "The Treatment of Missing Values and its Effect on Classifier Accuracy," in *Classification, Clustering, and Data Mining Applications*, D. D. Banks, D. F. R. McMorris, D. P. Arabie, and P. D. W. Gaul, Eds. Springer Berlin Heidelberg, 2004, pp. 639–647.

[30] A. Farhangfar, L. Kurgan, and J. Dy, "Impact of imputation of missing values on classification error for discrete data," *Pattern Recognit.*, vol. 41, no. 12, pp. 3692–3705, Dec. 2008.

[31] C. J. V. Rijsbergen, *Information Retrieval*, 2nd ed. Butterworth-Heinemann, 1979.

[32] Y. Yang and X. Liu, "A re-examination of text categorization methods," in *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, New York, NY, USA, 1999, pp. 42–49.