Effective Probabilistic Model for Webpage Classification

Hammad Haleem[1], Niyas C[2], Akshay Kumar[3] & Faiyaz Ahmad[4]

1, 2,3 & 4 Department of Computer Engineering, Faculty of Engineering and Technology,
Jamia Millia Islamia
New Delhi 110025, INDIA

Abstract—**World Wide Web (www) is a large repository of information which contains a plethora of information in the form of web documents. Information stored in web is increasing at a very rapid rate and people rely more and more on internet for acquiring information. Internet World Stats reveal that world internet usage has increased by 480% within the period 2000-2011. This exponential growth of the web has made it a difficult task to organize data and to find it. If we categorize data on the internet, it would be easier to find relevant piece of information quickly and conveniently. There are some popular web directories projects like yahoo directory and Mozilla directory in which web pages are organized according to their categories. According to a recent survey, it has been estimated that about 584 million websites are currently hosted on the internet. But these internet directories have only a tiny fraction of websites listed with them. The proper classification has made these directories popular among web users. However these web directories make use of human effort for classifying web pages and also only 2.5% of available webpages are included in these directories. Rapid growth of web has made it increasingly difficult to classify web pages manually, mainly due to the fact that manually or semi-automatic classification of website is a tedious and costly affair. Because of this reason web page classification using machine learning algorithms has become a major research topic in these days. A number of algorithms have been proposed for the classification of web sites by analyzing its features. In this paper we will introduce a fast, effective, probabilistic classification model with a good accuracy based on machine learning and data mining techniques for the automated classification of webpages into different categories based on their textual content.**

Index Terms: Content classification, Machine learning, Naïve Bayesian, Web-mining, Probabilistic models, Webpage classification.

I. INTRODUCTION

THE World Wide Web (WWW) started in the year 1991 and has shown a rapid growth within last two decades. It is estimated that today more than 3.5 billion people are using internet. The number of people using the Internet is rapidly increasing. Internet World Statistics reveals that the world Internet usage growth has increased by 480.4% during the period 2000-2011 [1]. It was also observed that more than 35% data available in whole world is stored in internet. According to Netcraft's January 2012 survey, more than 584 million web sites exist on the internet and out of which, nearly 175.2 million are active. Today there are several different tools available for an average internet user to locate and identify relevant information on the internet. These tools can be broadly classified as 1.Crawler based Search Engines (SE) e.g., Google, Bing, Yahoo, Duck-Duck-Go etc., 2.Meta Search engines e.g. Metacrawler, Clusty etc. and 3.Subject Directories like DMOZ (Directory Mozilla), Librarians Internet Index (LII) etc. The crawler based search engines and subject directories maintain their own data repositories, whereas meta search engines don't maintain any such data repositories, instead they depend on indices of other Search engines and subject directories to answer user queries. The database maintained by any crawler based search engine is quite large, and have considerably larger amount of data indexed in their databases as compared with subject directories. Directory Mozilla (DMOZ) [1] i.e., dmoz.com has 93,446 editors for 1.2 million categories and has indexed 5.26 million websites, which is only 2.5% of the total active web sites available on the Internet today. Majority of the web directories are edited and maintained by human effort.. Subject directories are popular due to proper classification of data in several categories. A larger website directory could be quite helpful in improving the quality of search results, filtering web content, developing knowledge bases, building vertical (domain specific) search engines. Hence need for automating the process of classification of websites based on their content arisen recently. Manually classifying data is really expensive to scale as well as quite labor intensive and in this paper we present a technique to reduce manual effort significantly and hence this paper presents a cost effective way for categorizing data.

This paper presents a Naïve Bayesian (NB) probabilistic model for the automatic classification of webpages. Naive Bayes probabilistic model is one of the most effective and straightforward model for text document classification and has exhibited good results in previous studies conducted for data mining. The model is quite optimized and has quite effectively worked to classify websites based on their content in real-time. Also it can be scaled for large databases. The model

presented in this paper has given accuracy around 94% for the test data considered.

The rest of the paper is organized as follows. Section II reviews previous work on the machine learning, classification and probabilistic approaches. Section III and IV discusses the classification of web pages and Naïve Bayes Theorem respectively, Section V presents our approach of classifying websites based on textual content of web pages using NB technique. Section VI discusses the results of our experiment. The last section summarizes the paper and gives some directions for future research.

II. Related Work

I. In this section we briefly try to review previous works in the field of text classification with a special emphasis on classification of webpages. In the starting days, task of classification of documents was generally carried out manually by experts of that domain. But very soon, ways were identified to carry out the classification in semi-automatic or automatic ways. Now we have a lot of effective ways to carry out machine assisted classification of documents. Many techniques have been researched and worked upon lately. Some of the approaches for text-categorization include statistical and machine leaning techniques like k-Nearest Neighbor approach [2`][3][4], Bayesian probabilistic models [5][6], inductive rule learning [7], decision trees [8],[9], neural networks [10],[11] and support vector machines [12],[13]. While most of the learning methods have been applied to pure text documents, there are numerous approaches dealing with classification of web pages. Pierre [12] discusses various practical issues in automated categorization of web sites. Machine and statistical learning algorithms have also been applied for classification of web pages [13]-[17]. In order to exploit the hypertext based organization of the web page several techniques like building implicit links [17], removal of noisy hyperlinks[18], fusion of heterogeneous data[19], link and context analysis[20] and web summarization[21] are used. An effort has been made to classify web content based on hierarchical structure [22].

II. Classification of Web pages

Web page classification is different from normal text classification in some aspects. The uncontrolled nature of web content presents additional challenges to web page classification as compared to traditional text classification. The web content is semi structured and contains formatting information in form of HTML tags. A web page may contain hyperlinks to point to other pages. This interconnected nature of web pages provides features that can be of greater help in classification. In the first step of classification all HTML tags are removed from the web pages, including punctuation marks. The next step is to remove stop words as they are common to all documents and does not contribute much in classification. In most cases a stemming algorithm is applied to reduce words to their basic stem. One such frequently used stemmer is the Porter's stemming algorithm [23]. Each text document obtained by application of procedures discussed above is represented as frequency vector. Machine learning algorithms are then applied on such vectors for the purpose of training the respective classifier. The classifier is then used to test an unlabeled set of sample documents against the learnt data. In order to rank high in search engine results, site promoters pump in many relevant keywords. This additional information can also be exploited.

III. Bayes Algorithm

A. *Bayesian Classifiers:*

Bayesian classifiers are statistical classifiers which predict the class membership probabilities of tuples. It means probability of a given tuple to be in a particular class. Bayesian classifiers are based on Bayes theorem which will be explained in next section. Studies comparing classification algorithms have found that a simple Bayesian classifier known as the naive Bayesian classifier is comparable in performance with decision tree and selected neural network classifiers. Bayesian classifiers have also exhibited high accuracy and speed when applied to large databases. In theory Bayesian classifiers have the minimum error rate in comparison to all other classifiers. However, in practice this is not always the case owing to inaccuracies in the assumptions made for its use, such as class-conditional independence, and the lack of available probability data. [24]

B. *Bayes Theorem:*

This theorem was named after Thomas Bayes who did early work in probability and decision theory during the 18th century. Let X a data tuple. In Bayesian terminology, X is termed as 'evidence'. X is described by n attributes. Ie, X={x1, x2, x3, x4.....xn}. Let H be some hypothesis that the data tuple X belongs to a particular class C. For classification problems we want to find P(H|X), the probability that hypothesis H holds when attributes set/'evidence' X is given. That means we are looking for the probability that the tuple X belongs to class C, given that we know the attribute description of X. Bayes theorem helps us to determine the probability P(H/X) in terms of P(H),P(X/H) and P(X).

Bayes Theorem is:

$$P(H/X) = \frac{P(X/H)P(H)}{P(X)} \quad \text{- (1)}$$

Where P(X/H) gives the probability for tuple to occur when it is given that the hypothesis holds. P (H) is the probability for the hypothesis H to hold. And P(X) is the probability for the attribute set X to occur.

C. *Naive Bayes Classifier*

Naive Bayes Classifiers are Bayesian Classifiers which assume *class conditional independence.* I.e., it assumes that the effect of an attribute value on a given class is independent of the values of the other attributes. This assumption simplifies the calculations a lot. Naive Bayesian

classifier works as follows.

Let D be a training set of tuples and their associated class labels. Each tuple is represented by n dimensional attribute vector in the form (x_1,x_2,x_3....x_4) depicting n measurements made on the tuple from n attributes, respectively, A_1, A_2,..A_n

$$P(Ci/X) = \frac{P(X/Ci)P(Ci)}{P(X)} \quad \text{- (2)}$$

1. As P(X) will be constant for all classes, we have to consider numerator term only. If we are taking equal number of training tuples for each class, then P(C_i) will be same for all classes. In such cases we have to maximize P(X/C_i) only. Otherwise we have to maximize P(X/C_i)*P(C_i). Where P(C_i) can be calculated as follows

$$P(Ci) = \frac{\text{Number of training tuples belonging to class Ci}}{\text{Total number of training tuples}} \quad \text{- (3)}$$

2. When we have data sets with several number attributes, it is computationally expensive to calculated P(X/Ci). But when we assume *class conditional independence*, computational cost can be reduced significantly. With this assumption P(X/Ci) can be calculated as follows.

$$P(X/Ci) = \prod_{k=1}^{n} P(xk/Ci) = P(x1/Ci)xP(x2/Ci)x....xP(xn/Ci) \quad \text{- (4)}$$

If attribute A_k is categorical, then P (x_k/C_i) is given as follows

$$P(xk/Ci) = \frac{\text{Number of tuples in class Ci having value xk for attribute Ak}}{\text{Total number of tuples in class Ci}} \quad \text{- (5)}$$

If attribute x_k is continuous valued, then the following function can be used for finding P(x_k/C_i)

$$g(x,\mu,\sigma) = \frac{1}{\sqrt{2\Pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \quad \text{- (6)}$$

So that:

$$P(xk/Ci) = g(xk,\mu Ci,\sigma Ci) \quad \text{- (7)}$$

To predict the class label of X, P(X/C_i)P(C_i) is evaluated for each class C_i. The classifier predicts that the class label of tuple X is the class C_i if and only if P(C_i/X) > P(C_j/X) for 1 <= j <= m and i!=j

D. Modification on Naive Bayes for document classification:
For document classification problems calculation of P(x_k/Ci) will be modified. It is required since attributes are not either categorical or continuous valued in case of document classification problems. In document classification problem we have the relation,

$$P(xk/Ci) \propto P(Ci) \prod_{1\le k\le nd} P(tk/Ci) \quad \text{- (8)}$$

Where n_d is number of tuples in class C_i and total number of occurrences of word x_k in category C_i. P (t_k/C_i) can be calculated as follows:

$$P(tk/Ci) = \frac{Tci}{\sum_{t\prime\in V} Tct'} \quad \text{- (9)}$$

Where T_{ci} is number of occurrences of word T_k in category C_i. V is the vocabular, all words in training set.

E. Laplacian correction:
To avoid zero computational value In order to avoid the zero probability value , when a word in testing tuple do not come in training set, we use Laplacian correction. Laplacian corrected formula is Where V is number of terms in vocabulary.

$$P(tk/Ci) = \frac{Tci+1}{\sum_{t\prime\in V}(Tct'+1)} = \frac{Tci+1}{\sum_{t\prime\in V} Tct' + |V|} \quad \text{- (10)}$$

IV. Experimental Setup

This section explains the setup of the entire experiment. We collected different web pages belonging to predefined categories. Then popular data cleaning and data extraction techniques were applied to extract useful data from collection of webpages. Once the data extraction process is finished we trained our classifier according to k-fold strategy for various values of k and used to predict category of webpages in test set. All the processes are briefly discussed below.

A. Creation of Data Set
The data set is collection of webpages within predefined categories. There were 6 predefined categories. Number of document in each category, number documents used for train set and number of document used for test set is given in table 1. Most of the documents were taken from popular news networks like BBC, CNN, and Reuters since they are already classified into various categories. We designed a crawler to automatically scan these web documents and to store these documents into different local directories corresponding to their category. In this experiment we have only considered the data in English language. Since we were only interested in the text content we simply removed all multimedia contents on the web page like Videos, Images, Flash plugin content and JAVA applets. Other than these, pages with relatively less content were also removed from the collection. The dataset consisted of total of 3183 documents in six categories.

B. Cleaning HTML Documents

We used the Python3 regular expression module and BeautifulSoup module to extract the information from web pages. Beautiful Soup is a Python library designed for easy and effective scraping of webpages and provides a few simple methods and Pythonic idioms for navigating, searching, and modifying a parse tree [23]. Rather than extracting information from specific tags we removed all the HTML tags present in the document with the exception of title tag because it offers considerable amount of information regarding the content of webpage.

Table 1
Composition of testing data

Category	Total	Train	Test
Education	529	432	97
Entertainment	537	429	108
Politics	508	406	106
Religion	536	428	108
Sports	530	424	106
Technology	543	434	109
Total	**3183**	**2553**	**630**

All non ASCII characters and special symbols were removed from the data. CSS and JavaScript content present in the HTML documents were also cleaned during this phase. In many webpages images were used as buttons to be clicked in place of hyperlinks or images were used to display name of the organization. Such information is a very important feature for classification purpose, however our experiment concentrates on text based retrieval so such graphical text was ignored. The standard Stop-Word list used in Bow [26] was used along with a comprehensive Stop-word list provided In the NLTK[27][28] (Natural language processing toolkit) library in python. Along this we used the stemming and lemmatization techniques to reduce words to their least forms. We used the post tagger present in the NLTK library to individually analyze each word. Then based on the word form i.e. Noun, Adjective or Verb the wordnet lemmatizer was used to strip down all the words in their least forms. Those words which were not recognized by the Wordnet lemmatizer were processed further with the Porter Stemming [25] algorithm to generate strip down version of the word. After all the above processing the HTML document was converted into a list of words in their most basic forms.

C. Vocabulary Generation

The words that occurred commonly in most of the webpages were considered as stop words. A list of such words is present in Table2.

Table 2
Web-page specific Stop Words

login, view, browser, website, web, online, search, keyword, designed, copyright, rights, reserved, click, search, welcome, email, click, contact, developed, mail, home, page, feedback, webmaster

These common words were considered as Stop-Words and removed directly from the document. It was also observed that webmasters inflated the title, Meta description and keyword tags with multiple keywords. We normalized such repeating keywords to reduce the impact of site promotion techniques applied by webmasters. This step was performed during the cleaning phase. All the words with the occurrence less than two were removed from the Documents and also the words with length less than three were also discarded from the Data. The frequency of each of the words was stored in a Hash table along with their category, for faster and efficient access in the later stages of experiment. The keywords that appear in two sample categories are given in table 3 and table 4

Table 3
Key words Related to Educational Category

student western university offer alumni news degree view program menu read bull calendar study experience graduate visit state academics admission major meet amp business college mountain campus life sport education service inform library common event institute watch music aid recreate history faculty graham athlete finance school center Wheaton art census Wichita request depart William online office form student day scholarship

Table 4
Keywords Related to Religion Category

god worship religion Hinduism prayer will Muslim time amp before symbol john people good quote practice order oxford year king source church refer idea pope power influence relate belief include life number faith doctrine religion follow century origin word work great call accord scripture book Christian article ecclesiast reform history state universe nation yantra India Iranian Buddhism Zoroastrian group adhere culture tradition pupil jump Indian Islam Abraham create retrieve Zoroaster Mazda text evil

D. Testing and Training the Classifier

We followed the k-fold cross validation strategy (with k=5) to decide the number of training and testing examples. The documents in each category are divided into 5 equal partitions say D_1, D_2, D_3... D_5. Training and testing will be performed k times. In iteration *I*, partition Di in each category will be reserved as a training set, and the remaining partitions will be collectively used to train the model. That is, in the first iteration, subsets D_2... D_5 collectively serve as the training set to obtain a first model, which will be tested on D_1 of each category; the second iteration will be trained on sets D_1, D_2, D_3... D_5 and will be tested on D_2. And so on. Prior probability of each category will be same since we have taken equal number of documents in each category. Training set was stored in the form of hash tables of hash tables. Each hash table in first level stands for category. And hash tables in second level will be containing the frequency of words in that particular category. The posterior probability with Laplace's correction was calculated using the formula[29][30]:

$$P(w_k|c) = (nk+1) / (n + |Vocabulary|).$$

- (11)

Where N_k stands for number of occurrences of word W_k in category C, N is total number of words in given category and |vocabulary| stands for number of words in training set.
In order to classify a document say *X*, the probabilities of each word of document in a given category were calculated from hash table , then they were multiplied together .The probability values obtained for individual categories were sometimes going below the floating point limit of Python (in order of $x*10^{-300}$) . We were able to find a solution for this problem by taking the Log_2 (P_x), of the obtained probability and summing them up. The obtained number was then further multiplied by -1, to get overall positive value.

$$C=\arg\max (-\log_2 (P_c) - \sum \log_2 (P_{wk} | C)$$

- (12)

The calculated values were compared to find the minimum of all. The category with minimum log of probability was selected as the classified class for a document. The equation 12 helps to clearly understand the solution of the mentioned problem. The Naïve Bayes algorithm to test the classifier is given below:

Algorithm Probability testing

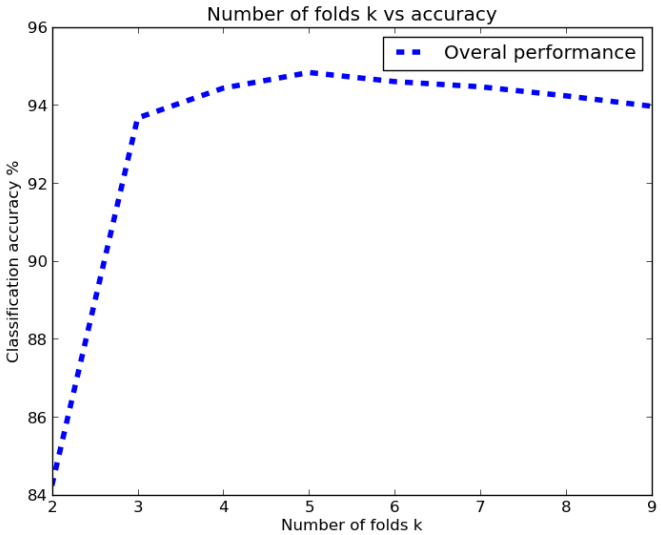
Input : D_{test} (set of all test documents) , Num_word (number of words in the training set),
D+(training set)
Output : D^- (Set of predicted documents)

```
for d in D_test  {
    Freq = Dict()
    Prob = Dict()
    Freq =find_frequency_document(d)
    Cat_prob= []

    for category in categories {
          prob[category] = 0
          for words in freq {
            if word in D+ [category]
                  prob[category]+=log(nk+1)-log(n+|vocabulary|)
            else
                  prob[category]+=log(1/(n+|vocabulary|))
                 Cat_prob.append(prob[catogegory])
          }
 d.catagory = {category for which Cat_prob[category] is
minimum}
          D-.push(d)
     }
}
```

V. Experimental Results

The classifier was executed for the given dataset with 3183 examples in six categories for various values of k. With each value of K, We were able to record corresponding accuracy. Graph1 depicts the relation between the increasing value of K and its effect on the accuracy of classification of documents for the range 2<=k<10.



Graph 1
Average accuracy vs. number of folds

Initially there is a rise in classification accuracy with increase in number of folds which can be explained in terms of better training. Then there is a slight fall in accuracy which can be explained in terms of over training of classifier. I.e. classifier is trained too much and is not able retrieve a result.
It is observed that the classifier was able to achieve maximum accuracy which is above 94% for K = 5, i.e. when $4/5^{th}$ of the documents were used as training set and rest for test set. Using our approach we were able to get a good accuracy.

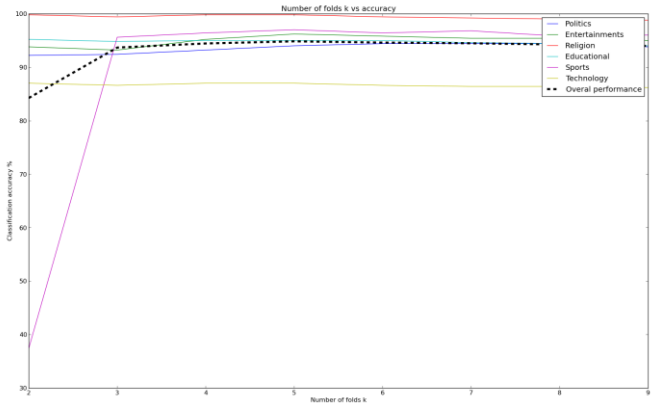
The table 4 depicts confusion matrix. A classifier is good if majority of the documents lie along the diagonal of a confusion matrix. This trend is observed in table 4. This confusion matrix can be further used to find most common classification measures like accuracy, recall, error rate, specificity and f-measures etc.

Table 4
Classification of data in various categories'

Category	Pol	Entr	Sport	Educ	Rel	Tech
Pol	470	5	10	0	14	1
Entr	7	481	6	0	6	0
Sport	11	3	485	0	1	0
Educ	17	0	0	475	6	2
Rel	1	0	0	0	499	0

Tech	64	0	0	1	0	235
Pol: Politics**, Entr:** Entertainment**, Sport:** sports **Educ:** Educational **Rel:** Religion **Tech:** Technology						

Based on each of the K value we generated confusion matrix for each of the category. We calculated the accuracy for each value of K. Graph 2 show the relation between the K value and accuracy achieved for specific category. We can see initially when the K value was lower the accuracy was lower. But as the K value increase efficiency also increased. Similar trend is also shown in the graph1. Thus we can say the classification model presented in this paper can be used effectively for real-time classification of webpage with high accuracy.



Graph2
Relation between Increasing K and accuracy of classifier

VI. Conclusion and Future Works

The naive bayes document classification algorithm discussed in this paper intelligently exploits the richness of textual content of webpage for effective classification into industry type category.

The algorithm here classifies the webpages into a very broad set of categories. Naive Bayes approach for classification of web pages based on their textual content, for six categories considered in this paper yielded above 94% accuracy. It has been observed that the classification accuracy of the classifier is proportional to number of training documents up to a limit. The results are quite encouraging. This approach could be utilized by the search engines and other online directory projects like DMOZ[31] for effective categorization of web pages to build an automated web directory based on the content available on the web page and type of organization. Although in this experiment, only nonhierarchical and distinct categories are considered the above algorithm can also be used to classify the pages into more specific categories (hierarchical classification) by changing the feature set e.g. a web site that is ecommerce may be further classified into electronics, clothes or a book selling website.

References

[1] *http://news.netcraft.com/archives/2013/08/09/august-2013-web-server-survey.html . Netcraft internet survey for august 2012*

[2] *Simple Knn approach for text classification http://www.cis.uab.edu/zhang/Spam-mining-papers/A.Simple.KNN.Algorithm.for.Text.Categorization.pdf*

[3] *Optimized approach for text classification http://www.cis.uab.edu/zhang/Spam-mining-papers/An.Optimized.Approach.for.KNN.Text.Categorization.using.P.Trees.pdf*

[4] *G. Guo, H. Wang and K. Greer, "A kNN model-based approach and its application in text categorization", Int. Conf., CICLing Springer, Seoul, Korea, 2004, pp. 559-570. 5th*

[5] *A. McCallum and K. Nigam, "A comparison of event models for Naïve Bayes text classification", in AAAI/ICML-98 Workshop on Learning for Text Categorization, 1998, pp. 41-48.*

[6] *D.D. Lewis and M. Ringuette, "A Classification of two learning algorithms for text categorization", in Proc. of 3rd Annual Symposium on Document Analysis and Information Retrieval (SDAIR'94), 1994, pp. 81-93.*

[7] *C. Apte and F. Damerau and S. M. Weiss, "Automated learning of decision rules for text categorization", ACM Trans. on Information Systems, Vol. 12, no.3, pp. 233-251, 1994.*

[8] *S. Wermter, "Neural network agents for learning semantic text classification", Information Retrieval, Vol. 3, no. 2, pp. 87 - 103, Jul 2000.*

[9] *A.S. Weigend, E.D. Weiner, and J.O. Peterson, "Exploiting hierarchy in text categorization", Information Retrieval, Vol. 1, no. 3, pp.193-216, 1999.*

[10] *E. Leopold, and J. Kindermann, "Text categorization with support vector machines. How to represent texts in input space?", Machine Learning, Vol. 46, no. 1-3, pp. 423-444, 2002*

[11] *D. Bennett and A. Demiritz, "Semi-Supervised support vector machines", Advances in Neural Information Processing Systems, Vol. 11, pp. 368-374, 1998.*

[12] *J. M. Pierre, "Practical issues for automated categorization of web sites.", in Electronic Proc. of ECDL 2000 workshop on the Semantic Web, Lisbon, Portugal, 2000.*

[13] *A. Sun, E. Lim and W. Ng, "Web classification using support vector machine", in Proc. of the 4th Int. workshop on Web information and data management, McLean, Virginia, USA, 2002, pp. 96 – 99.*

[14] *Y. Zhang and B. F. L. Xiao, "Web page classification based on a least square support vector machine with latent semantic analysis", in Proc. of the 5th Int. Conf. on Fuzzy Systems and Knowledge Discovery 2008, Vol. 2, pp. 528-532,*

[15] *O. Kwon and J. Lee, "Web page classification based on k-nearest neighbor approach", in Proc. of the 5th Int. Workshop on Information Retrieval with Asian languages, Hong Kong, China, 2000, pp. 9-15.*

[16] *S. Dehghan and A. M. Rahmani, "A classifier-CMAC neural network model for web mining", in Proc. of the IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology 2008, Vol. 1, pp. 427-431.*

[17] *S. Dou, S. Jian-Tao , Y. Qiang and C. Zheng, "A comparison of implicit and explicit lin [17] ks for web page classification", in Proc. of the 15th International Conference on World Wide Web, Edinburgh, Scotland, 2006 , pp. 643–650*

[18] *S. Zhongzhi and L. Xiaoli, "Innovating web page classification through reducing noise", Journal of Computer Science and Technology, Vol. 17, no. 1 , pp. 9–17, Jan. 2002*

[19] *Z. Xu, I. King and M. R. Lyu, "Web page classification with heterogeneous data fusion", in Proc. of the 16th International*

Conference on World Wide Web, Banff, Alberta, Canada, 2007, pp. 1171 – 1172,

[20] G. Attardi, A. Gulli, and F. Sebastiani, "Automatic web page categorization by link and context analysis", in Chris Hutchison and Gaetano Lanzarone (eds.), Proc. of THAI'99, 1999, pp. 105-119.

[21] S. Dou, C. Zheng, Y. Qiang, Z. Hua-Jun, Z. Benyu, L. Yuchang and M. Wei-Ying, "Web-page classification through summarization", in Proc. of the 27th annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval, Sheffield, United Kingdom, 2004, pp. 242 - 249.

[22] S. Dumais and H. Chen, "Hierarchical classification of web content", in Proc. of the 23rd annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval, Athens, Greece, 2000, pp. 256 - 263. [23] M.F. Porter, "An algorithm for suffix stripping", Program, Vo.14, no. 3, pp. 130-137, Jul. 1980.

[23] Python beautiful soup library. http://www.crummy.com/software/BeautifulSoup/

[24] http://tartarus.org/martin/PorterStemmer/index.html Porter Stemming algorithm, with various implementations.

[25] http://nlp.stanford.edu/IR-book/html/htmledition/naive-bayes-text-classification-1.html Stanford NLP , naïve bayes text classification

[26] The BOW or libbow C Library [Online]. Available: http://www.cs.cmu.edu/~mccallum/bow/

[27] Bird, Steven, Edward Loper and Ewan Klein (2009). Natural Language Processing with Python. O'Reilly Media Inc. Python NLTK

[28] www.matplotlib.org Python based open source mathematical analysis toolkit.

[29] T. M Mitchell. (1997). Machine Learning McGraw-Hill Companies, Inc.

[30] Data Mining concepts and techniques Han Kamber Lee Morgan Kaufman publications. 3rd Edition 2012.

[31] DMOZ open directory project. [Online]. Available: http://dmoz.org/

[32] C. J. van Rijsbergen. (1979). Information Retrieval (2nd ed.) London:Butterworths,[Online].Available:http://www.dcs.gla.ac.uk/Keith/Preface.html

[33] T. M Mitchell. (1997). Machine Learning McGraw-Hill Companies, Inc.