Effective probabilistic model for website classification

Author names 1, 2,3,4,5

Abstract—this electronic document is a "live" template. The various components of your paper [title, text, heads, etc.] are already defined on the style sheet, as illustrated by the portions given in this document. DO NOT USE SPECIAL CHARACTERS, SYMBOLS, OR MATH IN YOUR TITLE OR ABSTRACT. *(Abstract)*

Index Terms—Content classification, Machine learning, Naïve Bayesian, web-mining, probabilistic models.

I. Introduction

THE World Wide Web (WWW) started in the year 1991 and has gained large popularity day by day today more than 3.5 billion users are using internet. In the recent time we have seen internet penetration of more than 35% all over the world. Netcraft's January 2012 survey, estimates that more than 584 million web sites exist on the internet and out of which, nearly 175.2 million are active. The number of users using the Internet is rapidly increasing. Internet World Stats reveals that the world Internet usage growth has increased by 480.4% during 2000-2011. Today there are many varieties of tools available for an average internet user to locate and identify information on the internet. These tools are 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. These directories like DMOZ need to be manually edited and maintained manually. 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 web sites based on content of their webpages. This 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. The presented model has offered accuracy of approximately 92% 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 home 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

III. In this section we briefly try to review related work 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. Recently we have experienced a relatively high number of effective ways to carry out 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`][a][b], Bayesian probabilistic models [c][3]-[4], inductive rule learning [5], decision trees [4],[6], neural networks [7],[8] and support vector machines [9],[10]. While most of the learning methods have been applied to pure text documents, there are numerous publications dealing with classification of web pages. Pierre [11] discusses various practical issues in automated categorization of web sites. Machine and statistical learning algorithms have also been applied for classification of web pages [12]-[15]. In order to exploit the hypertext based organization of the web page several techniques like building implicit links [16], removal of noisy hyperlinks[17], fusion of heterogeneous data[18], link and context analysis[19] and web summarization[20] are used. An effort has been made to classify web content based on hierarchical structure [21].

IV. Classification of Web pages

Classification of web content is different in some aspects as compared with text classification. 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 consists of hyperlinks to point to other pages. This interconnected nature of web pages provides features that can be of greater help in classification. First 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 searching. 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 [22]. 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 classification mechanism of the algorithm is used to test an unlabeled sample document against the learnt data. In our approach we deal with home pages of organizational websites. A neatly developed home page of a web site is treated as an entry point for the entire web site. It represents the summary of the rest of the web site. Many URLs link to the second level pages telling more about the nature of the organization. The information contained the title, Meta keyword and Meta description and in the labels of the A HREF (anchor) tags are very important source of rich features. In order to rank high in search engine results, site promoters pump in many relevant keywords. This additional information can also be exploited. Most of the homepages are designed to fit in a single screen. The factors discussed above contribute to the expression power of the home page to identify the nature of the organization.

V. 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 a simple Bayesian classifier known as the naive Bayesian classifier is to be 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.[1]

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:

$$P(H/X) = \frac{P(X/H)P(H)}{P(X)}$$

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 Bayesian Classification

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

$$P(Ci/X) = \frac{P(X/Ci)P(Ci)}{P(X)}$$

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

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

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.

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

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

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

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

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

So that:

$$P(xk/Ci) = g(xk, \mu Ci, \sigma Ci)$$

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(xk/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,

$$P(xk/Ci) \propto P(Ci) \prod_{1\leq k\leq nd} P(tk/Ci)$$

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

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

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.

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

VI. Experimental Setup

V. Experimental Setup

This section explains the setup of the entire experiment. We collected different pages of websites pre-classified into different categories. We applied popular data cleaning and data extraction techniques to extract useful data from collection of webpages. The collected data was subjected to training and the model was tested on the remaining percentage of data. All the process is briefly discussed below.

A. Creation of Data Set

The dataset used contained various webpages generally in HTML markup format. The data belonged to six different categories as Mentioned in Table 1. In order to generate our dataset various search engines were used along with popular news networks like CNN, BBC. Most of the training data was extracted from the news articles from these popular news networks as the articles or content present on these websites is already classified manually into various categories, also the content provided is rich in English language and offers better results as compared to other sources of data. We designed a crawler to automatically scan these websites and store the already categorized data into different local directories for training and testing. In this experiment we have only considered the data in English language. Since we were only interested in the text content we simply remove any multimedia content on the web page. The content that was remove mainly consist of 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 Beautiful Soup provides a few simple methods and Pythonic idioms for navigating, searching, and modifying a parse tree [BeautifulSoup]: a toolkit for dissecting a document and extracting data 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 webpage. All non ASCII characters and any special symbols were removed from the data. Also the CSS and JavaScript 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.

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

The standard Stop-Word list used in Bow [24] was used along with a comprehensive Stop-word list provided In the NLTK (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 in the Porter Stemming 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.

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

C. Vocabulary Generation

The features that occurred commonly in most of the webpages were considered as stop features. A list of such words is present in Table2. These common words were considered as Stop-Words and remove directly from the document Common features that are part of every web site were considered as stop features (About Us, Home, All Rights Reserved, Contact Us, Feedback etc.). Such words are similar to regular stop words but specific to webpages. Such words were also considered as stop words and therefore were removed from the dataset. 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 one were removed from the Documents and also the words with length equal or less than three were also discarded from the Data. The frequency of each of the words was stored in a Hash table for faster and efficient access in the later stages of experiment.

Table 3
Educational Classifier

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 institut 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 3
Home page specific Stop Words

god worship religion Hinduism prayer will Muslim time amp before symbol john people good quote wycliff English practice order oxford year king form 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. Training the Classifier

The k-fold cross validation (with k=5) was followed to decide the number of training and testing examples. The documents in each category will be 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:

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

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.

E. Testing the Classifier

In order to classify a document say *X*, the probabilities of a given category are looked up in the hash table and multiplied together. The category producing the highest probability is the classification for document *X*. Only the words found in *X* would be looked up in the hash table. Also if a word in *X* is absent in the original vocabulary (built from training set) the word is ignored. The equation used to classify *X* is

$$C = \arg\max\ (P(c)\ \Pi\ P\ (wk|c)).$$

The Naïve Bayes algorithms to train and test the classifier are as given below:

Algorithm Probability training

Algorithm Probability testing

VII. Experimental Results

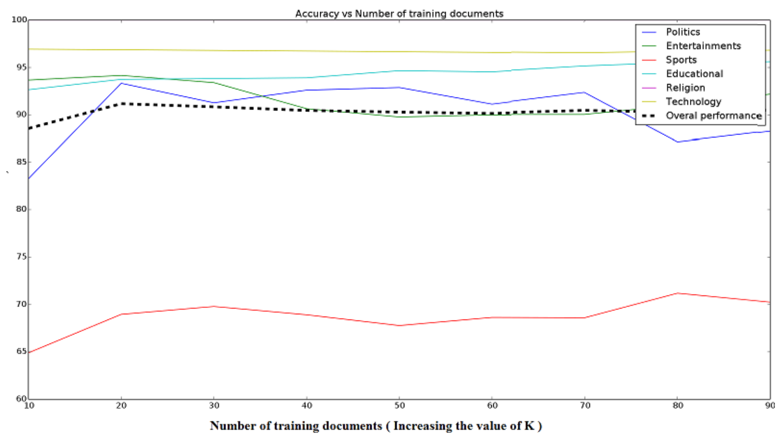
Table IV shows the results obtained when nine folds i.e., 4398 examples were used as the training set to build the Classifier and the remaining fold 489 examples were used to test the classifier for accuracy. We use recall [25], precision [25] and F-measure [25] to verify the accuracy of our classification approach. F-measure is the harmonic mean of recall and precision. Recall, Precision and F-Measure are calculated as follows:

Using the three measures, we observe that the average precision is 89.09%, average recall is 89.04%, whereas the F-Measure is 89.05%. Thus, classification of web sites is possible by examining the contents of their home pages.

Number of Training Examples and Accuracy

The classifier was subjected to training and testing in 9 steps each time increasing the input by 50 documents. Graph 1 depicts the number of training examples versus the accuracy in terms of average F-measure. The accuracy of the classifier was very poor i.e., about 45%, when only 50 documents were supplied as training data. The accuracy increases each time when the classifier is supplied with additional learning data. The classifier achieved an accuracy of 89% when nearly 450 documents were supplied as input in each category.

Thus, the accuracy of the classifier depends on the number of training documents and in order to achieve high accuracy, the classifier should be supplied with sufficiently large training documents.



Graph 1

This graph depicts the relation between the increasing values of k and the archived accuracy at that value of K.

VIII. Conclusion and Future Works

The naive bayes document classification algorithm discussed in this paper intelligently exploits the richness of features present in the home page of any website for effective classification into industry type category.

The algorithm here classifies the webpages into a very broad set of categories. Naive based approach for classification of websites based on home pages for six categories considered in this paper yielded a result of more than 92% accuracy. It has been observed that the classification accuracy of the classifier is proportional to number of training documents. The results are quite encouraging. This approach could be utilized by the search engines and other online directory projects like DMOZ for effective categorization of websites to build an automated website directory based on the content available on the website and type of organization. Although in this experiment, only nonhierarchical and distinct categories are considered. The above algorithm could 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] http://nlp.stanford.edu/IR-book/html/htmledition/naive-bayes-text-classification-1.html Stanford NLP , naïve bayes text classification

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

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

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

[6] Automated Classification of Web Sites. Proceedings of international conference of engineers and computer scientists 2012 Vol 1 , IMECS 2012 Hong Kong

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

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

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

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

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

A. http://www.cis.uab.edu/zhang/Spam-mining-papers/A.Simple.KNN.Algorithm.for.Text.Categorization.pdf
B. http://www.cis.uab.edu/zhang/Spam-mining-papers/An.Optimized.Approach.for.KNN.Text.Categorization.using.P.Trees.pdf
C. http://nlp.stanford.edu/IR-book/html/htmledition/naive-bayes-text-classification-1.html
D. http://www.dmoz.org
E. http://www.internetworldstats.com/stats.htm
F. http://en.wikipedia.org/wiki/World_Wide_Web
G. http://www.crummy.com/software/BeautifulSoup/