Hierarchical classification of web content using Naïve Bayes approach

Neetu
Research Scholar,
Central University of Himachal Pradesh,
Dharamshala, India
neetus7@gmail.com

***Abstract*— This paper explores the use of hierarchical structure to classify a heterogeneous collection of web pages. In the hierarchical classification, a model learns to distinguish a second level category from all other categories that are within the same top level. In the flat non hierarchical classification, a model distinguishes a second level category from all existing second level categories. We use Naïve Bayes classifier which has been proved to be effective for web content classification, but has not been previously explored in the case of hierarchical classification. This paper analyses the feasibility of a web page classifier which exploits the hierarchical structure of categories and studies their recall, precision and F-measure scores.**

Keywords- Machine learning, web page classification, Naïve Bayes classifier, hierarchical structure

I. INTRODUCTION

The goal of web content classification is to assign documents to predefined categories. This goal is central to tasks such as document routing, organizing documents into hierarchical catalogs or directory structures. The methods used for web content categorization differ in the type of the classifier, the technique used for training and the representation of the documents. The comparative results from the literature show that the best categorization methods differ from each other only slightly in accuracy. It is becoming very difficult to enhance performance significantly when a similar representation of topics is being used.

This paper exploits the hierarchical structure of categories to improve web content categorization performance over baseline models that ignore category structure. The classification problem is broken down into subtasks based on a category hierarchy. The use of a hierarchical decomposition of a classification problem allows for efficiencies in both learning and representation. Each sub-problem is smaller than the original problem, and it is sometimes possible to use a much smaller set of features for each [1].We have seen a wide application of problem decomposition to reduce a large problem into several smaller, easier problems. This idea applies to categorization with hierarchical structure in the classes. Using a hierarchical category structure allows us to decompose the problem: first, we determine the general topic group, then within that group, distinguishes among topics. This type of decomposition can be generalized to any number of levels.

II. RELATED WORK

The techniques for categorizing the text include statistical and machine leaning techniques like k-Nearest Neighbor approach [2], Bayesian probabilistic models [3]-[4], inductive rule learning [5], decision trees [4],[6], neural networks [7],[8] and support vector machines [9],[10]. These approaches do not explore the hierarchical structure of the categories. Much of the previous work on hierarchical methods for text classification uses the Reuters-22173 or Reuters-21578 articles. This is a rather small and tidy collection, and this alone is problematic for understanding how the approaches generalize to larger more complex internet applications. In addition, the Reuters articles are organized into 135 topical categories with no hierarchical structure [1]. Xipeng Qiu et al. [12] have proposed a variant Passive-Aggressive (PA) algorithm for hierarchical text classification with latent concepts. Ruiz and Srinivasan [13] used a hierarchical mixture of experts to classify abstracts within the MeSH sub-category Heart. The classifiers were learned at different levels of granularity, with the top-level distinctions serving as "gates" to the lower level "experts". Small advantages were found for the hierarchical approach compared with a flat approach. Larkey [14] classified patents in the Speech Signal Processing sub- category and compared hierarchical and flat approaches. She could not find any multilevel algorithm that performed significantly better than a flat one which chooses among all the speech classes.

III. WEB PAGE CLASSIFICATION

Web pages are what make up the World Wide Web. A web page is a document or information resource written usually in HTML and translated by the web browser. Web pages are formed by different kinds of information, such as videos, audios, images or other digital assets that are addressed by a common URL

(Uniform Resource Locator). All web pages contain different types of information. In order to classify them, data extracted from HTML code will be used.

The uncontrolled nature of web content poses additional challenges to web page classification as compared to traditional text classification. The web content contains formatting information in the form of HTML tags. A web page may consist of hyperlinks which point to other web pages. Data cleaning involves removal of all HTML tags, including punctuation marks from the web pages. Then the stop words are removed since they are common to all web pages and do not provide much information. Naïve Bayes machine learning algorithm is then applied for the purpose of training the classifier. The classification mechanism of Naïve Bayes classification algorithm is used to test an unlabelled test document against the labeled documents. In our approach, we have used Open Directory Project (ODP) which is available online at http://www.dmoz.org [11]. We then split the set of web pages into training and test set, as per ODP category of our interest.

IV. NAÏVE BAYES TEXT CLASSIFICATION

Let D = $\{d_1, d_2, d_3, \ldots, d_r\}$ to be a set of documents and C=$\{c_1, c_2, c_3, \ldots, c_p\}$ be set of classes. Each of the documents in D is classified into one of the classes from set C. The probability of a document d being in class c is calculated as:

$$P(c|d) \propto P(c) \prod_{1 \le k \le nd} P(w_k|c)$$

where $P(w_k|c)$ is the conditional probability of word w_k occurring in a document of class c. $P(w_k|c)$ is a measure of how much evidence is contributed by w_k that c is the correct class. P(c) is the prior probability of a document occurring in class c. $(w_1, w_2, w_3, \ldots w_{nd})$ are the terms in document d. These terms are a part of the vocabulary U we build for our classification purpose and nd is the number of terms in document d.

Our goal here is to find the best class for the document. The best class c_b is computed as:

$$c_b = \arg\max_{c \in C} P(c|d)$$

$$c_b = \arg\max_{c \in C} P(c) \prod_{1 \le k \le nd} P(w_k|c) \qquad (1)$$

Multiplying many conditional probabilities can result in a floating point underflow. In order to avoid this, we perform the computations by adding the logarithms of probabilities rather than multiplying probabilities. Log (xy) = log(x) +log(y) and the logarithm function is monotonic. So the equation (1) can be rewritten as:

$$c_b = \arg\max_{c \in C} [\log P(c) + \sum_{1 \le k \le nd} \log P(w_k|c)]$$

The class that has the highest final unnormalised log probability score is the most probable. Each conditional parameter $\log P(w_k|c)$ is a weight that indicates how good an indicator w_k is for c. log(P(c)) is a weight that indicates the relative frequency of c. The sum of log prior and term weights gives a measure of how much evidence there is for document being in the class. The prior probability of class c is given as:

$$P(c) = \frac{Nc}{N}$$

where N_c is the number of documents in class c and N is the total number of documents.

$$P(w|c) = \frac{T_{ct}}{\sum_{t' \in U} T_{ct'}} \qquad (2)$$

where T_{ct} is the number of occurrences of term w in the documents from class c. $\sum_{t' \in U} T_{ct'}$ gives the total number of terms in documents from class c. To remove zeros, Laplace smoothing is used. The equation (2) after adding Laplace correction becomes:

$$P(w|c) = \frac{T_{ct}+1}{\sum_{t' \in U} (T_{ct'}+1)} = \frac{T_{ct}+1}{\sum_{t' \in U} (T_{ct'}+|U|)}$$

V. EXPERIMENTAL SETUP

A. Datasets

The dataset is constructed by crawling the web pages that are found in the Open Directory Project (ODP). The ODP consists of web pages that preclassified into different categories. The set of web pages is split into training and a test set, per ODP category. We used 4415 pages for training and testing. Table 1 shows the number of pages in each top level category.

Table 1. Dataset

Category	**Total samples**	**Training samples**	**Testing samples**
Computer	1476	1329	147
Science	1493	1346	147
Society	1446	1303	143
Total	4415	3978	437

Table 2 shows the number of pages in the training and testing set of each second level category. Each top level category is divided into three second level categories. Dataset is formed by cleaning the web pages that includes removing their HTML tags, scripts, style sheets etc. Then, the dataset is used to train and test the classifier.

B. Developing the dataset

Our dataset consisted of web pages in HTML format. There are 3 top level categories and 9 second level categories mentioned in Table 1 and Table 2 respectively. The contents of the web pages are visually examined with the help of internet browser. Web pages belonging to the categories of our interest were stored in respective directories. Since we were interested in web page classification, web pages that were developed using technologies like Flash or other plug in applications were omitted from the dataset. Web pages built in languages other than English were also not included in the dataset. The dataset consisted of 4415 web pages in 3 top level and 9 second level categories.

Table 2. Second level of the hierarchy

Top level Category	**Second level category**	**Total samples**	**Training samples**	**Testing samples**
Computer	Hardware	491	442	49
	Software	491	442	49
	Internet	494	445	49
Science	Mathematics	498	449	49
	Biology	496	447	49
	Chemistry	499	450	49
Society	People	474	427	47
	Philosophy	474	427	47
	Politics	498	449	49
Total		4415	3978	437

C. Cleaning the Web pages

The HREF (hyperlink) label, TITLE, META description and META keyword and all BODY text of each webpage was extracted using the Jericho HTML parser. Jericho HTML parser is a java library allowing analysis and manipulation of parts of an HTML document, including server side tags, while reproducing verbatim any unrecognized or invalid HTML. It is an open source library released under both the Eclipse Public License (LGPL). It has built in functionality to extract all text from HTML markup. This library is available online at http://jerichohtml.sourceforge.net [15]. A major benefit of using this library is that the bad formatting present in some of the web pages does not affect the parsing of rest of the webpage.

Some web sites use images to display the name of the organization. This information can be very useful for the purpose of classification but since we were focused on text based classification, such graphical text was not included as a feature.

We used the standard stop word list of Bow [16]. Bow is a library of C code useful for writing statistical text analysis, language modeling and information retrieval programs. The current distribution includes the library, as well as front ends for document classification, document retrieval and document clustering. The library claims to be bug free. Bow uses a standard stop word list. Stemming algorithm such as Porter was not used because it sometimes changes the entire meaning of the word which is undesirable. E.g. In the software category, the word ‘resume’ refers to ‘resume the processing’ whereas in the ‘people’ category, the word ‘resume’ refers to the resume of a person.

D. *Generating the vocabulary*

Common words that occur in almost all the web pages were removed from the dataset. Such words were considered as stop words and added to the standard stop word list of Bow. Some of these common words are mentioned in Table 3

Table 3. Common Word List

http, www, html, org, en, Wikipedia, wiki, index, php, gov, uk, news, home, files, pdf, google, com,…

We created a vocabulary consisting of the most relevant words belonging to categories of our interest. The words that occur more than 6 times were considered as relevant words and were included in the vocabulary. Very common words and very rare words were removed from the dataset. We term this set of relevant words in the vocabulary as U.

E. *Classifier Training*

We have followed the k-fold strategy (k=10) to determine the number of training and testing samples. At the top level of the hierarchy, 3978 samples were used as training set to train the classifier and 437 samples were used as testing set to test the classifier. The prior probability of each category at the top level is 1/3 since there are 3 categories at the top level. For calculating the posterior probability, all the documents belonging to the respective category were parsed and the text of the documents were extracted and concatenated into a single file. Then, a hash table was prepared consisting of <key, value> pairs. The key here is the word occurring in the document of a particular category and value (nk) is the frequency of the word in all the documents of that category. For each category, we calculated the total number of words occurring in all documents of that category. We term it as n. The Laplace correction was calculated using the formula

$$P\,(w_k|c) = \frac{nk+1}{n+|U|}$$

The feature sets formed during the experiment for computer category and science category are given in Table 4 and 5 respectively.

Table 4. Feature set for the computer category

Mac, freeware, desktop, online, multimedia, directory, servers, programs, Microsoft, sites, tool, apps, pc, computer, image, database, …

Table 5. Feature set for the science category

Chromatography, oil, gas, electrochem, dna, molecular, bio, acids, instruments, sodium, fire, coal, refinery, distillation, calcium,…

Similarly, the classifier is trained for the categories at the second level. The feature sets for hardware, people and mathematics are mentioned in Table 6, 7 and 8 respectively.

Table 6. Feature set for the hardware category

Bit, editors, drivers, graphic, player, game, office, media, computer, antivirus, ipod, converter, company, bitdownload, filespack, …

Table 7. Feature set for the people category

Inmate, twitter, contact, prison, height, penpals, profiles, friends, posts, people, life, personals, family, name, books, groups,..

Table 8. Feature set for the mathematics category

Research, theory, lattices, calculus, mathematics, algebra, matlab, number, logic, functions, linear, calc, theorem, time, integralcalc, statistics, mathworld,…

F. *Classifier Testing*

The Naïve Bayes algorithm gives the category which produces the highest probability among the given categories. A test document T is taken and all the words occurring in T are looked up in the hash table formed during training for each category. If the test document T contains a word w which does not occur in the vocabulary U, then the word w is ignored. The classification of T in c is done using the formula: C_b= arg max $_{c\in C}$ P(c) $\prod_{1\le k\le nd} P$ $(w_k|c)$

The Naïve Bayes algorithm used to train and test the classifier is given below.

Algorithm- NB Training

```
1. Let U be the vocabulary consisting of all the words occurring in documents in the dataset.
2. For each category $c_i \in C$
    a. Let $D_i$ be the subset of documents in D belonging to category $c_i$.
    b. P $(c_i)$=| $D_i$ | / |D|
    c. Extract text from each document in $D_i$
    d. Let $T_i$ be the concatenation of all the documents of $D_i$
    e. Let $n_i$ be the total frequency of all the words in $T_i$
    f. For each word $w_i \in U$
        i. Let nk be the frequency of $w_i$ in $T_i$
        ii. Let P $(w_j|c_i)$= $\frac{nk+1}{n+|U|}$
3. Let S be the set of categories at the second level within the category $c_i$
4. For each category $c_i \in S$
    a. Let $D_i$ be the subset of documents in D belonging to category $c_i$.
    b. $P(c_i)$=| $D_i$ | / |D|
    c. Extract text from each document in $D_i$
    d. Let $T_i$ be the concatenation of all the documents of $D_i$
    e. Let $n_i$ be the total frequency of all the words in $T_i$
    f. For each word $w_i \in U$
        i. Let nk be the frequency of $w_i$ in $T_i$
        ii. Let P $(w_j|c_i)$= $\frac{nk+1}{n+|U|}$
```

Algorithm- NB Testing

```
1. Let a test document be T
2. Let total be the total number of word occurrences in T.
3. Return the category $c_t$ at the top level
   $c_t$ = arg max $_{ci \in C}P(c_i) \prod_{1\le j\le total} P$ $(w_j|c_i)$
4. For the category $c_t$ returned at the top level
    a. Let S be the set of categories at the second level within the category $c_t$
    b. Return the category $c_s$ at the second level
       $c_s$ =arg max $_{ci\in S}$ $P(c_i) \prod_{1\le j\le total} P$ $(w_j|c_i)$
```

VI. EXPERIMENTAL RESULTS

Table 9 shows the results obtained from our experiment when k-fold strategy was used for training and testing the classifier. The accuracy of our approach is verified by using recall, precision and F-measure. The equation for precision, recall and F-measure is given as follows:

$$\text{Precision} = \frac{Number\ of\ documents\ correctly\ assigned\ to\ the\ class}{Number\ of\ documents\ retrieved}$$

$$\text{Recall} = \frac{Number\ of\ documents\ correctly\ assigned\ to\ the\ class}{Total\ number\ of\ documents\ that\ have\ that\ topic}$$

$$\text{F-measure} = \frac{2\ X\ Precision\ X\ Recall}{Precision+Recall}$$

It is observed that average precision is 89.24, average recall is 89.09 and average F-measure is 89.15.

Table9. Accuracy of the classifier

Category	Precision	Recall	F-measure
Hardware	88.26	89.66	88.95
Software	89.21	88.16	88.68
Internet	86.42	87.53	86.97
Mathematics	93.23	88.53	90.71
Biology	91.24	93.56	92.38
Chemistry	88.22	87.64	88.76
People	86.21	90.16	88.14
Philosophy	90.14	89.43	89.78
Politics	88.83	88.24	88.53

Fig 1 shows how the average F-measure varies with the number of documents. The accuracy of the classifier was 48% when 50 documents were supplied for training. As the number of documents supplied for training increased, the accuracy of the classifier also increased. It touched 89% when 450 documents per second level category were used for training the classifier. So, it can be seen that the accuracy of the classifier is dependent upon the number of training documents and the classifier can achieve high accuracy when it is supplied with large training data.

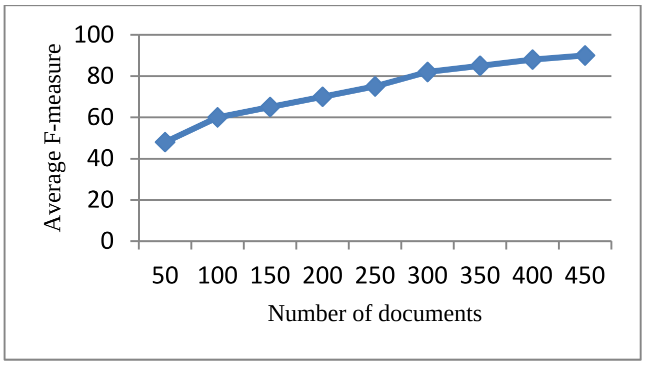

Fig 1: Number of training documents versus average F-measure

VII. CONCLUSION

This paper exploits the hierarchical structure of categories for classification. It classifies the web pages into very broad categories. Naïve Bayes approach is used for classification and it yielded accuracy. It is also seen that the accuracy of the classifier is dependent upon the number of training documents. As the number of training documents increases, the accuracy of the classifier also increases. The results are quite encouraging. Search engines can use this approach to classify the web pages and to build automated web directories.

VIII. ACKNOWLEDGEMENT

Our thanks to the experts who have contributed towards the development of the template.

IX. REFERENCES

[1] Koller, D. and Sahami, M. “Hierarchically classifying documents using very few words”. Proceedings of the Fourteenth International Conference on Machine Learning (ICML’97), pp. 170-178, 1997.
[2] Guo, G., Wang, H. and Greer, K.. “An kNN model-based approach and its application in text categorization”, 5th Int. Conf., CICLing Springer, Seoul, Korea, 2004, pp. 559-570.
[3] McCallum, A. and Nigam, K.. “A comparison of event models for Naïve Bayes text classification”, in AAAI/ICML-98 Workshop on Learning for Text Categorization, 1998, pp. 41-48.
[4] Lewis, D.D. and Ringuette, M.. “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.
[5] Dumais, S.T., Platt, J., Heckerman, D. and Sahami M.. “Inductive learning algorithms and representations for text categorization”, in Proc. of the 17th Int. Conf. on Information and Knowledge Management (CIKM’98), 1998, pp. 148-155.
[6] Apte, C. and Damerau, F. and Weiss, S.M.. “Automated learning of decision rules for text categorization”, ACM Trans. on Information Systems, Vol. 12, no.3, pp. 233-251, 1994.
[7] Wermter S. “Neural network agents for learning semantic text classification”, Information Retrieval, Vol. 3, no. 2, pp. 87 – 103, 2004.
[8] Weigend, A.S., Weiner, E.D. and Peterson, J.O. “Exploiting hierarchy in text categorization”, Information Retrieval, Vol. 1, no. 3, pp. 193- 216, 1999.

[9] Leopold, E. and Kindermann, J. .”Text categorization with support vector machines. How to represent texts in input space?”, Machine Learning, Vol. 46, no. 1-3,pp. 423-444, 2002.
[10] Bennett D. and Demiritz, A.. “Semi-Supervised support vector machines, Advances in Neural Information Processing Systems”, Vol. 11, pp. 368-374, 1998.
[11] Open Directory Project [Online]. Available: http://www.dmoz.org
[12] Qiu, Xipeng. Huang. Xuanjing, Liu, Zhao. And Zhou. Jinlong. “Hierarchical Text Classification with Latent Concepts”, Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, 2011.
[13] Ruiz, M.E. and Srinivasan, P. “Hierarchical neural networks for text categorization”. Proceedings of the 22nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR’99), 1999, pp. 281-282.
[14] Larkey, L. “Some issues in the automatic classification of U.S. patents”. In Working Notes for the AAAI-98 Workshop on Learning for Text Categorization,1998.
[15] The Jericho HTML Parser Library Version 3.2. [Online], Available: The Jericho HTML Parser Library Version 3.2, [Online]. Available : http://www.jerichohtml.sourceforge.net
[16] The BOW C Library [Online]. Available: http://www.cs.cmu.edu//~mccallum/bow/