
Automatic Web Page Classification

Yasser Ganjisaffar

84802416

yganjisa@uci.edu

1 Introduction

To facilitate user browsing of Web, some websites such as Yahoo! (<http://dir.yahoo.com>) and Open Directory Project (<http://dmoz.org>) manually maintain a hierarchical structure. While manual classification of web pages provides high accuracy, it is very expensive. To automatically include new emerging pages into these hierarchies, web page classification becomes a hot research topic in web information retrieval.

Web page classification is more than typical text based classification methods. The main reason is that Web pages have far richer structures, such as hypertext tags, metadata, hyperlinks and styles. In this project, I studied how adding these structural information can help improving traditional text classifiers. The experiments are performed on four different classifiers.

2 Document Representation

For representing documents, I used the traditional vector space model where each document is represented as a vector of term weights. More formally, given $W = \{w_1, \dots, w_k\}$ be the set of unique words in the dataset we can represent each document d as a vector $\langle d_1, \dots, d_k \rangle$ where d_i is the number of occurrences of word w_i in d .

However, the text on the pages themselves is often insufficient or irrelevant for a reliable classification. Therefore, other sources of information must be used for better classification. For example, information on the pages that contain a link to a given page are often very helpful in classification. For example, home pages of computer science departments often only consist of images with pointers to information about offered courses, students, and faculty home pages. Even if this information is contained on a single page, the words on the page itself do not provide many clues for the fact that we are dealing with the home page of a computer science department as opposed to any other page in a computer science department.

I used the following sources of information for classifying documents:

Text In the simplest case, we can consider a web page as a set of words obtained after removing HTML tags.

Text + Anchors The anchor text is the visible, clickable text in a hyperlink. Anchor text usually gives high quality information about the content of the target page. For example, department pages typically have a large number of links pointing to them

Feature Set	# Features	Features with document frequency ≥ 10
Text	53,674	%12.6
Text + Anchor	58,735	%12.1
Text + Anchor + Style	76,356	%11.8
Text + Anchor + Style + URL	87,326	%11.0

Table 1: Number of Features in different Feature Sets

that are marked with anchor texts that include phrases like “Computer Science Department”, “CS Department”, “Dept. of Computer Science”, or similar. Similarly, student home pages often contain a pointer to their advisor’s home page. Thus, faculty home pages can often be identified by the occurrence of the word “advisor” in the anchor of a link that points to them.

Therefore, the anchor words of the incoming links can be considered as additional context features. For this purpose, the anchor words of the incoming links are prefixed with *anchor:* and are added to the document words.

Text + Anchors + Styles Styles are another aspect in which Web pages are different from normal texts. Web pages can have text chunks with different font sizes. Typically text chunks with larger font sizes are more important in the page and therefore must be weighted more in the classification process. In addition, other style features (bold, italic, underline, color, etc.) can be used for distinguishing styled words from normal words. In our classification task, we add a *style:* prefix to the words that are more expressive in the web page.

Text + Anchors + Styles + URL In addition to the above features, URL of the web page can also be a valuable source of information when classifying that page. For example, one can guess that a page with URL *http://cs-www.bu.edu/faculty/heddaya* is probably a faculty’s home page. For extracting the information available in the URL of web pages, URLs are tokenized and these tokens are prefixed with a *url:* prefix and are added to the set of words of the page.

2.1 Dimensionality Reduction

Since the number of unique words in a real-word dataset can be very large, the stop-words (words with very high frequency such as ‘the’ and ‘to’) are removed and all the remaining words are stemmed using Porter [8] stemmer. But even by using these techniques the number of dimensions can be very large. For example, in the WebKB dataset, there are 53,674 distinct words. Adding anchors, styles and URL words, the number of features would increase to 87,326 features (Table 1). It’s obvious that most of these features are noisy and are not useful in classifying documents. For example, %87 of the words in the WebKB dataset are occurred in less than 10 documents (%56 of the words are occurred only in one document). It is better to remove these rare words before classification. In my experiments, I omit words that have occurred in less than 10 documents.

2.2 Normalization

Since web pages have different lengths, it is necessary to normalize their text lengths before comparing their feature vectors and making judgments on their similarities. Document vectors are normalized by being divided by their size: $d/||d||$

3 Dataset

For evaluating the proposed methods and algorithms I used the WebKB [5] dataset. This dataset contains 8,221 webpages from several universities, labeled with whether they are student, faculty, staff, department, course, project, or other pages. Figure 1 shows the distribution of documents on categories in this dataset.

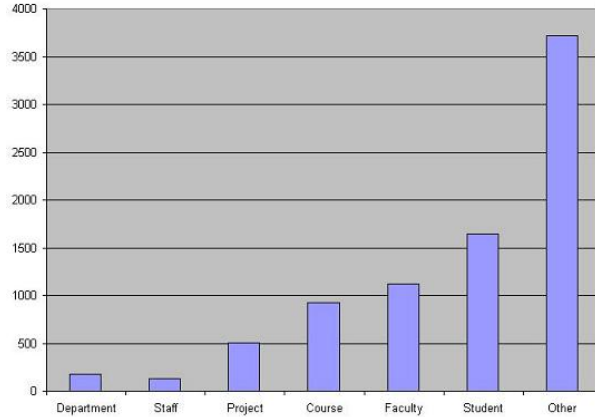


Figure 1: Distribution of documents on Categories in WebKB dataset

4 Classifiers Used

For my experiments, I used four different classifiers.

4.1 k -NN

In k -NN classifier, a document is classified by a majority vote of its neighbors, with the object being assigned to the class most common amongst its k nearest neighbors. Different measures of similarity can be used for finding the k nearest neighbors of each document. One popular measure is the cosine similarity which is the cosine of the angle between document vectors:

$$\cos \theta = \frac{d_1 \cdot d_2}{\|d_1\| \|d_2\|}$$

Since documents are normalized, this would be reduced to the dot product of the vectors ($d_1 \cdot d_2$).

4.2 Multinomial Naive Bayes

A naive Bayes classifier is a simple probabilistic classifier based on applying Bayes' theorem with strong (naive) independence assumptions. In this classifier, new examples are assigned to the class that is most likely to have generated the example. While the naive assumption is clearly false in most real-world tasks, naive Bayes often performs classification very well.

Recent approaches to text classification have used two different first-order probabilistic models for classification, both of which make the naive Bayes assumption. Some use a multi-variate Bernoulli model, that is, a Bayesian Network with no dependencies between

words and binary word features. Others use a multinomial model [4], that is, a uni-gram language model with integer word counts.

It is shown that the Multinomial model usually performs better on larger vocabulary sizes. In this model, prior probability of each class is calculated based on the fraction of the documents in the training set that belong to this class: $p(c) = \frac{D_c}{D}$. In addition, probability of each word given each class is also calculated based on the number of occurrences of word w in documents of class c to the total number of words in documents of class c : $p(w|c) = \frac{n(w,c)}{n(c)}$. But it is very common that some words have not been seen in training examples of a class. We don't want to assign zero probabilities to these words. Therefore a smoothing parameter, α , is used:

$$p(w|c) = \frac{n(w,c) + \alpha}{n(c) + |W| \times \alpha}$$

where $|W|$ is the size of the vocabulary.

Probability of document d which is a concatenation of words w_1, \dots, w_m , being generated by class c can be calculated as $p(c) \times p(w_1|c) \times \dots \times p(w_m|c)$. Since we only want to compare these probabilities and the probabilities would approach to zero for large values of m , we can compare their log values:

$$c_d = \operatorname{argmax}_c \{ \log p(c) + \sum_{w \in d} \log p(w|c) \}$$

Note that α is a parameter between 0 and 1 which can be tuned by training on the training set.

4.3 SVM

As an SVM classifier, I used *SVM^{light}* [6] package. It is an implementation of the support vector machines with a fast optimization algorithm and has built-in support for standard kernel functions. But the point is that an SVM-classifier can be used in binary classification problems, while the web page classification problem is a multi-class classification problem. For building a k -class SVM classifier, I combined k 1-versus-others classifiers. Each classifier determines whether the document is in this class or the other remaining classes with a confident degree (Larger positive values indicate document belonging to this class, while larger negative values indicate document belonging to other classes). Then for each test document it would be classified based on the degree of confidence in these classifiers (Classifier with the largest positive value determines class of document).

4.4 Decision Tree Classifier: J48

As another classifier, I was interested in using a decision tree classifier. For this purpose, I used Weka [10] which is an open source collection of data mining software written in Java. I used J48 which is Weka's implementation of the popular C4.5 decision tree algorithm [7]. This algorithm is based on the concept of *Information Entropy*. C4.5 uses the fact that each attribute of the data can be used to make a decision that splits the data into smaller subsets. It examines the normalized Information Gain (difference in entropy) that results from choosing an attribute for splitting the data. The attribute with the highest normalized information gain is the one used to make the decision. The algorithm then recurs on the smaller sublists.

		Text	Text + Anchor	Text + Anchor + Style	Text + Anchor + Style + URL
k -NN	$k = 3$	0.644	0.642	0.656	0.670
	$k = 5$	0.656	0.668	0.672	0.692
	$k = 7$	0.667	0.678	0.673	0.683
	$k = 10$	0.655	0.668	0.671	0.685
Multinomial Naive Bayes		0.642	0.683	0.671	0.727
SVM		0.792	0.803	0.816	0.832
J48		0.695	0.697	0.701	0.704

Table 2: Classification accuracies for different classifiers on different feature sets

5 Evaluations

For parsing the Web pages and extracting text and styles from them, I used Cobra [12] which is a Java HTML Renderer & Parser. The reason for choosing this parser among the various available parsers was its support for HTML 4, Javascript, and CSS 2. This means that by using this parser, in addition to extracting the text of the page, I could also find the font and style (bold, italic, underline) of every chunk of text.

For evaluating classifiers, we need to split the dataset into a training set and a test set. In my experiments, I randomly split the dataset into a training set covering %80 of documents and a test set covering %20 of documents. For more accurate results, I repeated this random selection of train and test sets several times and reported average results.

Table 2 shows average of classification accuracies—portion of correctly classified test documents—of different classifiers on different feature sets.

As can be seen in this table, SVM performs better than the other classifiers on all feature sets. In addition, most of the classifiers perform better when more features are considered for documents. Figure 2 is a better visualization of results.

Note that the α parameter of the multinomial naive bayes model is trained by splitting the training set into train–eval sets. Figure 3 is a sample training of this parameter on a log scale which shows how different values of this parameter can affect classification accuracy.

It is worth mentioning that these classifiers belong to four different categories of classifiers with very different properties. For example, k -NN is an instance based classifier, meaning that the training instances are needed in the testing phase. J48 is the slowest in the training phase while it is fastest in the testing phase. When considering the sum of training time and testing time required, Multinomial is the fastest classifier and J48 is the slowest one.

6 Related Work

In the literature, different authors have tried different classifiers to the task of web page classification. In [1], authors have used SVM for classifying web pages. In [2] authors have used PCA for feature selection and then applied Neural Networks for classification of pages. In [3] authors have proposed a summarization–based approach for reducing noise in classifying web pages.

In [11], authors have used separate features to train individual classifiers, whose outputs

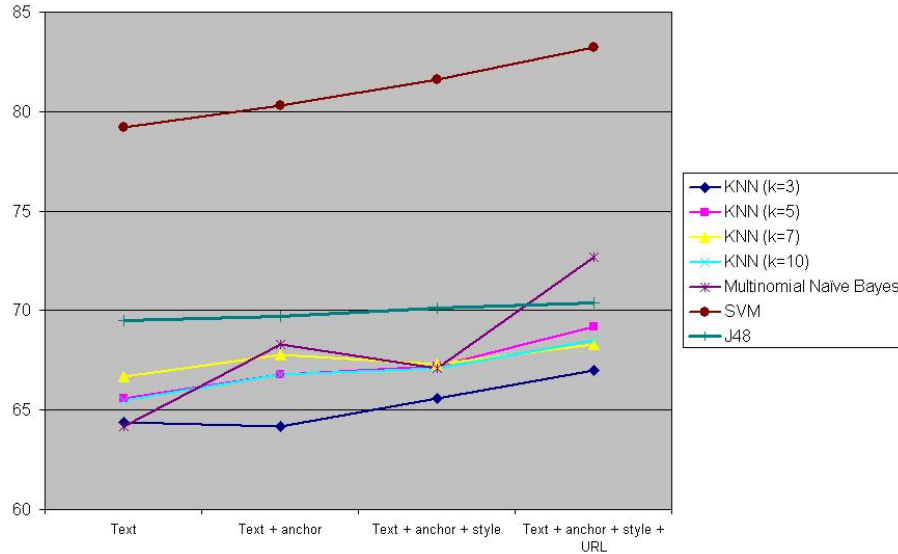


Figure 2: Trends of classification accuracies of different classifiers

are combined to give a final category for each page.

7 Conclusion

In this project, I studied the problem of automatic Web page categorization. Using experiments performed on a standard dataset, I studied how adding several features can improve the classification accuracy. I used several different classifiers on the dataset and results show that all of these classifiers perform better in the hybrid approach where all of the text, anchors, style and URL features are present. In addition, results show that on this dataset, the SVM classifier outperforms other classifiers with a good margin on all of the feature sets.

References

- [1] Sun A., Lim E., Ng W., (2002) Web Classification Using Support Vector Machines, in *Proceedings of the fourth international workshop on Web information and data management*, pp. 96–99. ACM Press.
- [2] Selamat A., Omatu S., (2004) Web page feature selection and classification using neural networks, *Information Sciences-Informatics and Computer Science*, Volume 158, Issue 1, pp. 69–88.
- [3] Shen. D., et al, (2004) Web–page Classification through Summarization, *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 242–249, ACM Press.
- [4] McCallum A., Nigam K. (1998) A Comparison of Event Models for Naive Bayes Text Classification.
- [5] World Wide Knowledge Base Project, Available online at: <http://www.cs.cmu.edu/~webkb/>
- [6] SVM–Light Support Vector Machines, <http://svmlight.joachims.org/>

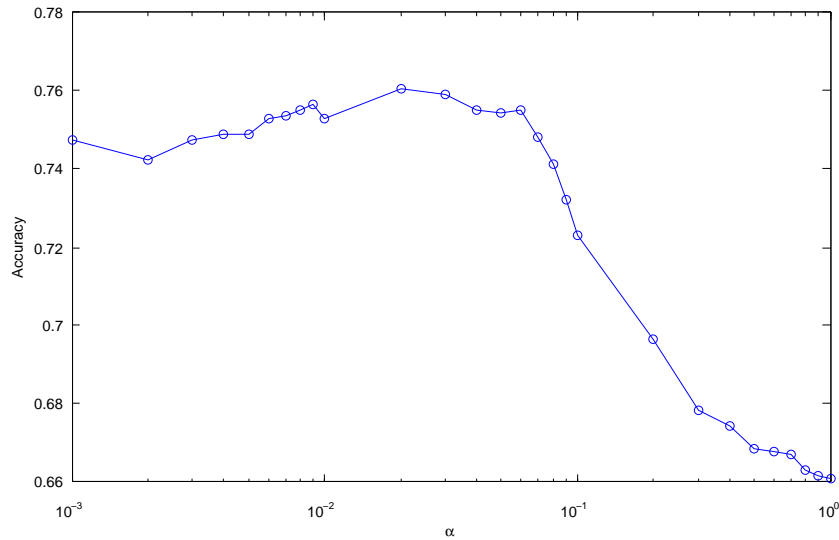


Figure 3: Training α for Multinomial Naive Bayes Model

- [7] Quinlan J. R., (1993) C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers.
- [8] Porter M. F., (1997) An Algorithm for suffix stripping, Readings in information retrieval, pp. 313–316, Morgan Kaufmann Publishers Inc.
- [9] Ghani R., et al, (2001) Hypertext Categorization using Hyperlink patterns and Meta data, In Proc. of 18th International Conference on Machine Learning, pp. 178–185, Morgan Kaufmann Publishers.
- [10] Weka 3: Data Mining with Open Source Machine Learning Software in Java, <http://www.cs.waikato.ac.nz/ml/weka/>
- [11] Bennett P., Dumais S., Horvitz E., (2002) Probabilistic combination of text classifiers using reliability indicators: Models and results, In Proc. of SIGIR'02, pp. 207–215.
- [12] Cobra: Pure Java HTML Renderer & Parser, <http://lobobrowser.org/cobra.jsp>.