Experiments in Hypertext Categorization

Houda Benbrahim and Max Bramer

Department of Computer Science and Software Engineering

University of Portsmouth, UK

{houda.benbrahim, max.bramer}@port.ac.uk

Abstract

This paper looks at (i) which extra information hidden in HTML tags and linked neighbourhood pages to take into consideration to improve the task of automatic classification of web documents and (ii) how to deal with the high level of noise in linked pages. A hypertext dataset and four well-known learning algorithms (Naïve Bayes, K-Nearest Neighbour, Support Vector Machine and C4.5) are used to exploit the enriched text representation. The results show that the clever use of the information in linked neighbourhood and HTML tags improves the accuracy of the classification algorithms.

1. Introduction

It has been estimated that the World Wide Web comprises more than 3 billion pages and is growing at a rate of 1.5 million pages a day. Faced with such a huge volume of documents, search engines become limited: too much information to look at and too much information retrieved. The organization of web documents into categories will reduce the search space of search engines and improve their retrieval performance. A recent study showed that users prefer to navigate through directories of pre-classified content and that providing a categorised view of retrieved documents enables them to find more relevant information in a shorter time. The common use of the manually constructed category hierarchies for navigation support in Yahoo and other major web portals has also demonstrated the potential value of automating the process of hypertext categorization.

Text categorization is a relatively mature area where many algorithms have been developed and experiments conducted. Classification accuracy reached 87% for some algorithms applied to known text categorization corpora (Reuters, 20_newspaper…) where the vocabulary is coherent and the authorship is high. Those same classical classifiers perform badly on samples from Yahoo! pages. This is due to the extreme diversity of web pages (such as homepages, articles…) and authorship, and to limited consistency in vocabulary.

Automated hypertext categorization poses new research challenges because of the extra information in a hypertext document. Hyperlinks, HTML tags, metadata and linked neighbourhood all provide rich information for classifying hypertext that is not available in traditional text categorization. Researchers have only recently begun to explore the issues of exploiting rich hypertext information for automated categorization.

There is a growing volume of research in the area of learning over web text documents. Since most of the documents considered are in HTML format, researchers have taken advantage of the structure of those pages in the learning process. The systems generated differ in performance because of the quantity and nature of the additional information considered.

Benbrahim and Bramer used the BankSearch dataset to study the impact of the use of metadata (page keywords and description), page title and link anchors in a web page on classification. They concluded that the use of basic text content enhanced with weighted extra information (metadata + title + link anchors) improves the performance of three different classifiers.

Oh et al. reported some observations on a collection of online Korean encyclopaedia articles. They used system-predicted categories of the linked neighbours of a test document to reinforce the classification decision on that document and obtained a 13% improvement over the baseline performance when using local text alone.

Joachims et al. also reported a study using the WebKB university corpus, focusing on Support Vector Machines with different kernel functions. Using one kernel to represent one document based on its local words, and another kernel to represent hyperlinks, they give evidence that combining the two kernels leads to better performance in two out of three classification problems.

Yang, Slattery and Ghani have defined five hypertext regularities which may hold in a particular application domain, and whose presence may significantly influence the optimal design of a classifier. The experiments were carried out on 3 datasets and 3 learning algorithms. The results showed that the naïve use of the linked pages can be more harmful than helpful when the neighbourhood is noisy, and that the use of metadata when available improves the classification accuracy.

This paper deals with web document categorization. Two issues will be considered in depth: (i) the choice of representation for documents and the extra information hidden in HTML pages and its neighbourhood that should be taken into consideration to improve the classification task, and (ii) how to filter out the noisy neighbourhood. Finally, data collected from the web will be used to evaluate the performance of different classification methods with different choices of text representation.

Document representation is described in Section 2. Some classification algorithms used for hypertext are reviewed in Section 3. Section 4 presents experiments and results, comparing different classification algorithms with different webpage representation techniques.

2. Text Representation

In order to apply machine-learning methods to document categorization, consideration first needs to be given to a representation for HTML pages. An indexing procedure that maps a text into a compact representation is applied to the dataset. The most frequently used method is a *bag-of-words* representation where all words from the set of documents under consideration are taken and no ordering of words or any structure of text is used. The words are selected to support classification under each category in turn, i.e. only those words that appear in documents in the specified category are used (the *local dictionary* approach). This means that the set of documents has a different feature representation (set of features) for each category. This approach for building the dictionary has been reported to lead to better performance. This leads to an attribute-value representation. Each distinct word corresponds to a feature, with the number of times the word occurs in the document as its value.

Based on our preliminary work, metadata (page keywords and description), page title and link anchors in a web page along with basic page content improved the accuracy of the classification task. This extra information is included in the data dictionary. Next, the words in the page's neighbours (documents pointing to, and pointed to by the target page) are included in the text representation. The blind use of the content in links may harm the classification task, this is due to the fact that many pages point to pages from different subjects, e.g., web pages of extremely diverse topics link to Yahoo! or BBC news web pages. To filter out the noisy link information, just the most similar adjacent pages to the target page are kept. The similarity of pages is determined by the cosine measure.

With the bag-of-words approach for text representation, it is possible to have tens of thousands of different words occurring in a fairly small set of documents. Using all these words is time consuming and represents a serious obstacle for a learning algorithm. Moreover many of them are not really important for the learning task and their usage can degrade the system's performance. There are many approaches to reducing the feature space dimension. The most common ones are: (i) the use of a stop list containing common English words, (ii) or the use of stemming, that is keeping the morphological root of words.

3. Classification Algorithms

(a) Naïve Bayes (NB)

Naïve Bayes (NB) is a widely used model in machine learning and text classification. The basic idea is to use the joint probabilities of words and categories in the training set of documents to estimate the probabilities of categories for an unseen document. The term 'naïve' refers to the assumption that the conditional probability of a word is independent of the conditional probabilities of other words in the same category.

A document is modelled as a set of words from the same vocabulary, *V*. For each class, C_j, and word, $w_k \in V$, the probabilities, $P(C_j)$ and $P(w_k|C_j)$ are estimated from the training data. Then the posterior probability of each class given a document, D, is computed using Bayes' rule:

$$P(C_j \mid D) = \frac{P(C_j)}{P(D)} \prod_{i=1}^{|D|} P(a_i \mid C_j)$$

where a_i is the ith word in the document, and $|D|$ is the length of the document in words. Since for any given document, the prior probability *P(D)* is a constant, this factor can be ignored if all that is desired is ranking rather than a probability estimate. A ranking is produced by sorting documents by their odds ratios, $P(C_1|D) / P(C_0|D)$, where C_1 represents the positive class and C_0 represents the negative class. An example is classified as positive if the odds are greater than 1, and negative otherwise.

(b) K-Nearest Neighbour (KNN)

K-Nearest Neighbour (KNN) is a well-known statistical approach in pattern recognition. KNN assumes that similar documents are likely to have the same class label. Given a test document, the method finds the *K* nearest neighbours among the training documents, and uses the categories of the *K* neighbours to weight the category candidates. The similarity score of each neighbour document to the test document is used as the weight of the categories of the neighbour document. If several of the K nearest neighbours share a category, then the per-neighbour weights of that category are added together, and the resulting weighted sum is used as the likelihood score of that category with respect to the test document. By sorting the scores of candidate categories, a ranked list is obtained for the test document. By thresholding on these scores, binary category assignments are obtained. The decision rule in KNN can be written as:

$$y(\vec{x}, c_j) = \sum_{\vec{d_i} \in KNN} sim(\vec{x}, \vec{d_i})\, y(\vec{d_i}, c_j) - b_j$$

where $y(\vec{d_i}, c_j) \in \{0,1\}$ is the classification for document $\vec{d_i}$ with respect to category cj (y = 1 for Yes, and y = 0 for No); $sim(\vec{x}, \vec{d_i})$ is the similarity between the test document $\vec{x}$ and the training document $\vec{d_i}$; and b_j is the category specific threshold for the binary decisions.

(c) C4.5 (Decision Tree Classification)

C4.5 is a decision tree classifier developed by Quinlan. The training algorithm builds a decision tree by recursively splitting the data set using a test of maximum gain ratio. The tree is then pruned based on an estimate of error on unseen cases. During classification, a test vector starts at the root node of the tree, testing the attribute specified by this node, then moving down the tree branch corresponding to the value of the attribute in the given example. This process is then repeated for the subtree rooted at the new node until a leaf is encountered, at which time the pattern is asserted to belong to the class named by that leaf.

(d) Support Vector Machine (SVM)

Support vector machines are based on the Structural Risk Minimization principle from computational learning theory. The idea is to find a hypothesis *h* for which we can guarantee the lowest true error. The true error of *h* is the probability that *h* will make an error on unseen and randomly selected test examples. An upper bound can be used to connect the true error of a hypothesis *h* with the error of *h* on the training set and the complexity of H (measured by VC-Dimension), the hypothesis space containing *h*. Support vector machines find the hypothesis *h*, which minimizes this bound on the true error by effectively and efficiently controlling the VC-Dimension of H. A remarkable property of SVMs is that their ability to learn can be independent of the dimensionality of the feature space. This property makes them suitable for text categorization where the dimension of the input space is very high.

4. Experiments

(a) Dataset

To test the proposed algorithms for hypertext classification, datasets were needed that reflected the properties of real world hypertext classification tasks.

The major practical problem in using web document datasets is that most of the URLs become unavailable. The well-known dataset WebKB project at CMU is outdated since most of its web pages are no longer available.

The BankSearch dataset used for the experiments comprises a set of HTML web documents. The Open Directory Project and Yahoo! categories were used to provide web pages that have already been categorized by people. The dataset considered consists of 11,000 pages distributed over 11 categories under 4 distinct themes. The dataset consists of some sets of categories that are quite distinct from each other, as well as other categories that are quite similar. Table 1 gives a summary of the dataset.

Dataset ID	Dataset Category	Associated Theme
A	Commercial Banks	Banking and Finance
B	Building Societies	Banking and Finance
C	Insurance Agencies	Banking and Finance
D	Java	Programming Languages
E	C/C++	Programming Languages
F	Visual Basic	Programming Languages
G	Astronomy	Science
H	Biology	Science
I	Soccer	Sport
J	Motor Sport	Sport
K	Sport	Sport

Table I. Dataset Summary

(b) Performance Measures

The evaluation of the different classifiers is measured using four different measures: recall (R), precision (P), accuracy (Acc), and F1 measure. These can all be defined using the 'confusion matrix' shown as Table II.

	Correct Class is C_k	*Correct Class is* $\overline{C_k}$
Assigned class is C_k	A	b
Assigned class is $\overline{C_k}$	C	d

Table II. Confusion Matrix

$$R = \frac{a}{(a+c)} \quad \textit{if (a+c) > 0 otherwise R = 1}$$

$$P = \frac{a}{(a+b)} \quad \textit{if (a+b) > 0 otherwise P = 1}$$

$$Acc = \frac{(a+d)}{n} \quad \textit{where n = a + b + c + d > 0}$$

$$F1 = \frac{2PR}{(R+P)} = \frac{2a}{(2a+b+c)} \quad \textit{if (a+c) > 0 otherwise undefined.}$$

Recall (R) is the percentage of the documents for a given category that are classified correctly. Precision (P) is the percentage of the predicted documents for a given category that are classified correctly. Accuracy (Acc) is defined as the ratio of correct classification into a category C_k.

Neither recall nor precision makes sense in isolation from the other. In fact, a trivial algorithm that assigns class C_k to all documents will have a perfect recall (100%), but an unacceptably low precision. Conversely, if a system decides not to assign any document to C_k it will have a perfect precision but a low recall. The F1 measure has been introduced to balance recall and precision by giving them equal weights.

Classifying a document involves determining whether or not it should be classified in any or potentially all of the available categories. Since the four measures are defined with respect to a given category only, the results of all the binary classification tasks (one per category) need to be averaged to give a single performance figure for a multiple class problem.

In this paper, the 'micro-averaging' method will be used to estimate the four measures for the whole category set. Micro-averaging reflects the per-document performance of a system. It is obtained by globally summing over all individual decisions and uses the global contingency table.

(c) Design of experiments

The classification algorithms NB, KNN, SVM and C4.5 were applied to the BankSearch dataset to address the different binary classification problems. The dataset was randomly split into 70% training and 30% testing.

Two local dictionaries were then built for each category and for each text representation after stop word removal (using a stop list of 512 words provided by David Lewis), with the option of stemming turned either on or off. Documents were represented by a VSM where the weights were the term frequencies in documents.

Two series of experiments were conducted. The documents are represented by (i) the basic content of HTML documents or (ii) a combination of basic HTML content, metadata, title and link anchors of the target page, along with weighted content of the similar in-coming and out-going linked pages.

The local dictionaries and the document's VSM for the second option of text representation were constructed as follows. For each target page in the dataset, the set of neighbour pages was determined. The content of all the in-coming and out-going links along with the target page was used to build the dictionaries. Then, the similarity of each target page with its neighbours was calculated to filter out the noisy links. The term weights of the target pages were adjusted so that the target page was influenced by its similar neighbours.

(d) Results and interpretation

The pages considered in this specific dataset have on average 16.4 out-going links**,** with this number varying from a maximum of 189 to a minimum of 1. This number also varies depending on the category considered. Concerning the in-coming links, the average number of pages was 7, with a maximum of 456 and a minimum of 0. Many target pages in the dataset were not pointed to by any document in the web. An interesting remark was drawn while determining the similar pages to a given target page; the average number of similar pages (including both in-coming and out-going pages) was 5, with a minimum of 0 and a maximum of 36 (those numbers vary depending on the considered category). As a result, a large number of linked pages were thrown away. This explains clearly the fact that linked neighbourhood is noisy. This filtering step was helpful in this regard.

The different algorithms result in different performance depending on the features used to represent the documents.

The set of experiments evaluates SVM, C4.5, NB and KNN, for texts represented using either (i) the basic content enhanced by the meta data, title and link anchors with stemming option turned on or off, or (ii) a combination of basic content, metadata, title and link anchors of the target page with those of its similar neighbours where extra weight was assigned to common words between the target page and its neighbours, this is done with the stemming option turned on or off.

Figure 1 (Figure 2) reports the performance accuracy (F1 measure) on the test set of SVM, C4.5, NB and KNN for the different text representation options.

Figures 1 and 2 show that the use of stemming improves the performance of all the classifiers for all the options of text representation. Stemming is helpful since it decreases the size of the feature space. Moreover, it combines the weights of the different words that share the same stem.

These figures also show that SVM outperforms all the other classifiers. This is not surprising since it has been reported that it works well in high dimensional feature space. This also explains its slight increase in performance when stemming was used. C4.5 also outperforms NB and KNN for the different text representations. The features selected by C4.5 to build the tree were meaningful in terms of class description.

Including the extra information in the filtered neighbourhood to the basic pages has improved the performance of the different classifiers. Note that if the threshold used to decide about the similarity of two pages is set too high no similar documents will be found and the performance of the classifiers, with the linked neighbourhood information taken into consideration for text representation, will be as good as that of the classifiers with the basic text content as text representation. The slight increase in each classifier's performance when the linked information is used means that all the noisy links that may harm the classification were filtered out. The threshold used in those experiments to decide about the similarity of two pages was set to 0.8. This threshold may seem too high since it may decline even useful links, but at the same time, somewhat secure since there is a low chance of including noisy links.

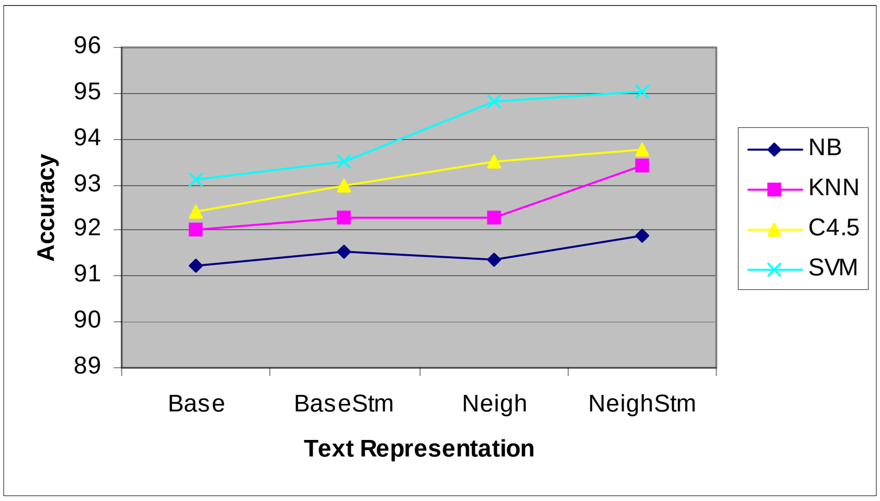

Figure 1: N.B, KNN, C4.5 and SVM accuracy for different choices of text representation

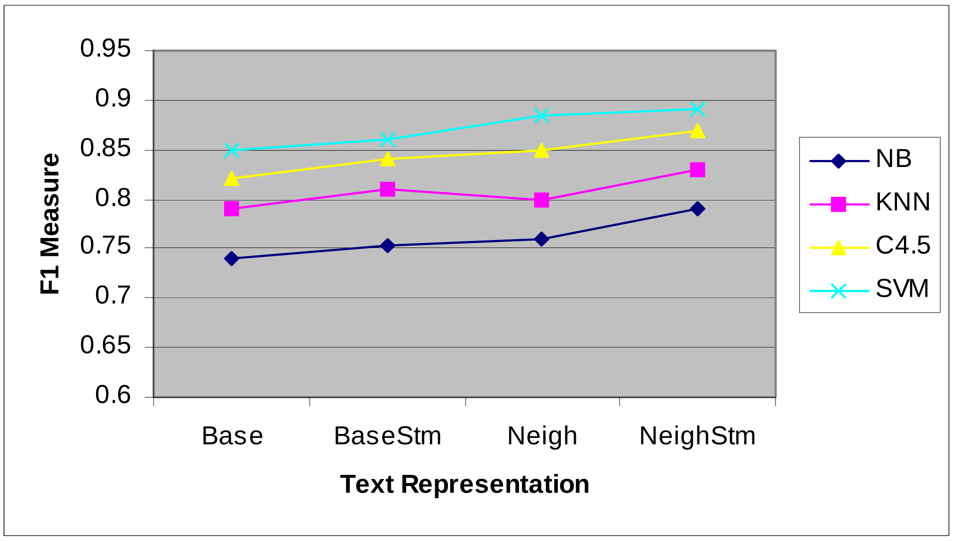

Figure 2: N.B, KNN, C4.5 and SVM F1 measure for different choices of text representation

5. Conclusions and Future Work

In summary, a number of experiments were conducted to evaluate the performance of some well-known learning algorithms on hypertext data. Different text representations have been used and evaluated. It can be concluded that the careful use of the extra information available in the linked neighbourhood increases the performance of the classifiers. The improvement was smaller than expected since the filtering was too high, and useful links might have been filtered out.

The careful use of the extra information in the linked neighbourhood of HTML pages improved the performance of the different classifiers. In future work, this extra information will be extended by less severely selecting the useful neighbour links. The class of the linked neighbourhood instead of its similarity to the target page may be used to filter out the noisy links. Experiments with different datasets should also be conducted before final conclusions are drawn.

This paper is a slightly shortened version of [1], which includes a full set of references.

[1] Benbrahim, H. and Bramer, M.A. (2004). Neighbourhood Exploitation in Hypertext Categorization. *In* Research and Development in Intelligent Systems XXI, Springer-Verlag, 2005.