Naïve Bayes vs. Decision Trees vs. Neural Networks in the Classification of Training Web Pages

Daniela XHEMALI[1], Christopher J. HINDE[2] and Roger G. STONE[3]

[1] Computer Science, Loughborough University
Loughborough, Leicestershire, LE11 3TU, UK
D.Xhemali@lboro.ac.uk

[2] Computer Science, Loughborough University
Loughborough, Leicestershire, LE11 3TU, UK
C.J.Hinde@lboro.ac.uk

[3] Computer Science, Loughborough University
Loughborough, Leicestershire, LE11 3TU, UK
R.G.Stone@lboro.ac.uk

Abstract
Web classification has been attempted through many different technologies. In this study we concentrate on the comparison of Neural Networks (NN), Naïve Bayes (NB) and Decision Tree (DT) classifiers for the automatic analysis and classification of attribute data from training course web pages. We introduce an enhanced NB classifier and run the same data sample through the DT and NN classifiers to determine the success rate of our classifier in the training courses domain. This research shows that our enhanced NB classifier not only outperforms the traditional NB classifier, but also performs similarly as good, if not better, than some more popular, rival techniques. This paper also shows that, overall, our NB classifier is the best choice for the training courses domain, achieving an impressive F-Measure value of over 97%, despite it being trained with fewer samples than any of the classification systems we have encountered.

Keywords: *Web classification, Naïve Bayesian Classifier, Decision Tree Classifier, Neural Network Classifier, Supervised learning.*

1. Introduction

Managing the vast amount of online information and classifying it into what could be relevant to our needs is an important step towards being able to use this information. Thus, it comes as no surprise that the popularity of Web Classification applies not only to the academic needs for continuous knowledge growth, but also to the needs of industry for quick, efficient solutions to information gathering and analysis in maintaining up-to-date information that is critical to the business success.

This research is part of a larger research project in collaboration with an independent brokerage organisation, Apricot Training Management (ATM), which helps other organisations to identify and analyse their training needs and recommend suitable courses for their employees. Currently, the latest prospectuses from different training providers are ordered, catalogued, shelved and the course information found is manually entered into the company's database. This is a time consuming, labour-intensive process, which does not guarantee always up-to-date results, due to the limited life expectancy of some course information such as dates and prices and other limitations in the availability of up-to-date, accurate information on websites and printed literature. The overall project is therefore to automate the process of retrieving, extracting and storing course information into the database guaranteeing it is always kept up-to-date.

The research presented in this paper is related to the information retrieval side of the project, in particular to the automatic analysis and filtering of the retrieved web pages according to their relevance. This classification process is vital to the efficiency of the overall system, as only *relevant* pages will then be considered by the extraction process, thus drastically reducing processing time & increasing accuracy.

The underlining technique used for our classifier is based on the NB algorithm, due to the independence noticed in the data corpus analysed. The traditional technique is enhanced however, to analyse not only the visible textual content of web pages, but also important web structures such as META data, TITLE and LINK information. Additionally, a 'believed probability' of features in each category is calculated to handle situations when there is little evidence about the data, particularly in the early stages of the classification process. Experiments have shown that our classifier exceeds expectations, achieving an impressive F-Measure value of over 97%.

2. Related Work

Many ideas have emerged over the years on how to achieve quality results from Web Classification systems, thus there are different approaches that can be used to a degree such as Clustering, NB and Bayesian Networks, NNs, DTs, Support Vector Machines (SVMs) etc. We decided to only concentrate on NN, DT and NB classifiers, as they proved more closely applicable to our project. Despite the benefits of other approaches, our research is in collaboration with a small organisation, thus we had to consider the organisation's hardware and software limitations before deciding on a classification technique. SVM and Clustering would be too expensive and processor intensive for the organisation, thus they were considered inappropriate for this project. The following discusses the pros and cons of NB, DTs and NNs, as well as related research works in each field.

2.1 Naïve Bayes Models

NB models are popular in machine learning applications, due to their simplicity in allowing each attribute to contribute towards the final decision equally and independently from the other attributes. This simplicity equates to computational efficiency, which makes NB techniques attractive and suitable for many domains.

However, the very same thing that makes them popular, is also the reason given by some researchers, who consider this approach to be weak. The conditional independence assumption is strong, and makes NB-based systems incapable of using two or more pieces of evidence together, however, used in appropriate domains, they offer quick training, fast data analysis and decision making, as well as straightforward interpretation of test results. There is some research ([13], [26]) trying to relax the conditional independence assumption by introducing latent variables in their tree-shaped or hierarchical NB classifiers. However, a thorough analysis of a large number of training web pages has shown us that the features used in these pages can be independently examined to compute the category for each page. Thus, the domain for our research can easily be analysed using NB classifiers, however, in order to increase the system's accuracy, the classifier has been enhanced as described in section 3. Enhancing the standard NB rule or using it in collaboration with other techniques has also been attempted by other researchers. Addin et al in [1] coupled a NB classifier with K-Means clustering to simulate damage detection in engineering materials. NBTree in [24] induced a hybrid of NB and DTs by using the Bayes rule to construct the decision tree. Other research works ([5], [23]) have modified their NB classifiers to learn from positive and unlabeled examples. Their assumption is that finding negative examples is very difficult for certain domains, particularly in the medical industry. Finding negative examples for the training courses domain, however, is not at all difficult, thus the above is not an issue for our research.

2.2 Decision Trees

Unlike NB classifiers, DT classifiers can cope with combinations of terms and can produce impressive results for some domains. However, training a DT classifier is quite complex and they can get out of hand with the number of nodes created in some cases. According to [17], with six Boolean attributes there would be need for 18,446,744,073,709,551,616 distinct nodes. Decision trees may be computationally expensive for certain domains, however, they make up for it by offering a genuine simplicity of interpreting models, and helping to consider the most important factors in a dataset first by placing them at the top of the tree.

The researchers in [7], [12], [15] all used DTs to allow for both the structure and the content of each web page to determine the category in which they belong. An accuracy of under 85% accuracy was achieved by all. This idea is very similar to our work, as our classifier also analyses both structure and content. WebClass in [12] was designed to search geographically distributed groups of people, who share common interests. WebClass modifies the standard decision tree approach by associating the tree root node with only the keywords found, depth-one nodes with descriptions and depth-two nodes with the hyperlinks found. The system however, only achieved 73% accuracy. The second version of WebClass ([2]) implemented various classification models such as: Bayes networks, DTs, K-Means clustering and SVMs in order to compare findings of WebClassII. However, findings showed that for increasing feature set sizes, the overall recall fell to just 39.75%.

2.3 Neural Networks

NNs are powerful techniques for representing complex relationships between inputs and outputs. Based on the neural structure of the brain ([17]), NNs are complicated and they can be enormous for certain domains, containing a large number of nodes and synapses. There is research that has managed to convert NNs into sets of rules in order to discover what the NN has learnt ([8], [21]), however, many other works still refer to NNs as a 'black box' approach ([18], [19]), due to the difficulty in understanding the decision making process of the NN, which can lead to not knowing if testing has succeeded.

AIRS in [4] used the knowledge acquired during the training of a NN to modify the user's query, making it possible for the adapted query to retrieve more documents than the original query. However, this process would sometimes give more importance to the knowledge 'learnt', thus change the original query until it lost its initial keywords.

Researchers in [6] and [14] proposed a term frequency method to select the feature vectors for the classification of documents using NNs. A much later research ([3]) used NNs together with an SVM for better classification performance. The content of each web page was analysed together with the content of its neighbouring pages. The resulting feature scores were also used by the SVM.

Using two powerful techniques may radically improve classification; however, this research did not combine the techniques to create a more sophisticated one. They simply used them one after the other on the same data set, which meant that the system took much longer to come up with results.

3. NB Classifier

Our system involves three main stages (Fig. 1). In stage-1, a CRAWLER was developed to find and retrieve web pages in a breadth-first search manner, carefully checking each link for format accuracies, duplication and against an automatically updatable rejection list.

In stage-2, a TRAINER was developed to analyse a list of relevant (training pages) and irrelevant links and compute probabilities about the feature-category pairs found. After each training session, features become more strongly associated with the different categories.

The training results were then used by the NB Classifier developed in stage-3, which takes into account the 'knowledge' accumulated during training and uses this to make intelligent decisions when classifying new, unseen-before web pages. The second and third stages have a very important sub-stage in common, the INDEXER. This is responsible for identifying and extracting all suitable features from each web page. The INDEXER also applies rules to reject HTML formatting and features that are ineffective in distinguishing web pages from one-another. This is achieved through sophisticated regular expressions and functions which clean, tokenise and stem the content of each page prior to the classification process. Features that are believed to be too common or too insignificant in distinguishing web pages from one another, otherwise known as *stopwords*, are also removed. Care is taken, however, to preserve the information extracted from certain Web structures such as the page TITLE, the LINK and META tag information. These are given higher weights than the rest of the text, as we believe that the information given by these structures is more closely related to the central theme of the web page.

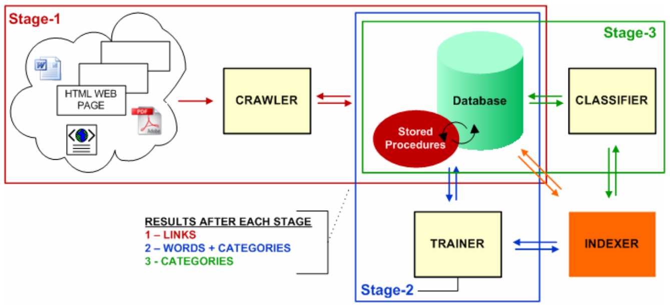

Fig. 1 System Stages

The classification algorithm is then applied to each category stored in the database. There are only two categories currently used in this research, *relevant* and *irrelevant*, however, the system is designed to work with any number of categories. This is to allow for future growth and potential changes at ATM.

Our classification algorithm is based on the NB approach. The standard Bayes rule is defined as follows:

$$\arg\max_{n}\{P(C_n|w)\} = \frac{P(w|C_n) \times P(C_n)}{P(w)} \tag{1}$$

where:

$P(C_n)$ = the prior probability of category n
w = the new web page to be classified
$P(w|C_n)$ = the conditional probability of the test page, given category n.

The *P(w)* can be disregarded, because it has the same value regardless of the category for which the calculation is carried out, and as such it will scale the end probabilities by the exact same amount, thus making no difference to the overall calculation. Also, the results of this calculation are going to be used in comparison with each other, rather than as stand-alone probabilities, thus calculating *P(w)* would be unnecessary effort. The Bayes Theorem in (eq. 1) is therefore simplified to:

$$\arg\max_{n}\{P(C_n|w)\} \propto P(w|C_n) \times P(C_n) \tag{2}$$

The algorithm used in this research is based on the mathematical manipulation of the probabilities of the top 100 most used keywords extracted from each web page. However, the separate probabilities alone would not be sufficient in classifying a new web page. The classification of each page requires the combination of the probabilities of all the separate features $\{f_1, f_2, \dots f_i, \dots, f_n\}$ into one probability, as in eq. 3:

$$\arg\max_n \{P(C_n|w)\} \propto \frac{1}{z} P(C_n) \prod_{i=1}^{n} P(f_i|C_n) \tag{3}$$

where z is a scaling factor dependent on the feature variables, which helps to normalise the data. This scaling factor was added as we found that when experimenting with smaller datasets, the feature probabilities got very small, which made the page probabilities get close to zero. This made the system unusable.

Once the probabilities for each category have been calculated, the probability values are compared to each other. The category with the highest probability, and within a predefined threshold value, is assigned to the web page being classified. All the features extracted from this page are also paired up with the resulting category and the information in the database is updated to expand the system's knowledge.

Our research adds an additional step to the traditional NB algorithm to handle situations when there is very little evidence about the data, in particular during early stages of the classification. This step calculates a Believed Probability (Believed Prob), which helps to calculate more gradual probability values for the data. An initial probability is decided for features with little information about them; in this research the probability value decided is equal to the probability of the page, given no evidence ($P(C_n)$) The calculation followed is as shown below:

$$Believed\ Prob = \frac{(bpw \times P(C_n)) + (\sum_{i=1}^{C_n} f_i \times P(f_i|C_n))}{bpw + \sum_{i=1}^{C_n} f_i} \tag{4}$$

where: *bpw* is the believed probability weight (in this case *bpw* = 1, which means that the *Believed Probability* is weighed the same as one feature).

The above eliminates the assignment of extreme probability values when little evidence exists; instead, much more realistic probability values are achieved, which has improved the overall accuracy of the classification process as shown in the Results section.

4. Results

4.1 Performance Measures

The experiments in this research are evaluated using the standard metrics of accuracy, precision, recall and f-measure for Web Classification. These were calculated using the predictive classification table, known as Confusion Matrix (Table 1).

Table 1: Confusion Matrix

		PREDICTED	
		IRRELEVANT	RELEVANT
ACTUAL	IRRELEVANT	**TN**	**FP**
	RELEVANT	**FN**	**TP**

Considering Table 1:

TN (True Negative) → Number of correct predictions that an instance is *irrelevant*
FP (False Positive) → Number of incorrect predictions that an instance is *relevant*
FN (False Negative) → Number of incorrect predictions that an instance is *irrelevant*
TP (True Positive) → Number of correct predictions that an instance is *relevant*

Accuracy – The proportion of the total number of predictions that were correct:

$$\textbf{Accuracy}\ (\%) = (TN + TP) / (TN + FN + FP + TP) \tag{5}$$

Precision – The proportion of the predicted *relevant* pages that were correct:

$$\textbf{Precision}\ (\%) = TP / (FP + TP) \tag{6}$$

Recall – The proportion of the *relevant* pages that were correctly identified

$$\textbf{Recall}\ (\%) = TP / (FN + TP) \tag{7}$$

F-Measure – Derives from precision and recall values:

$$\textbf{F-Measure}\ (\%) = (2 \times Recall \times Precision) / (Recall + Precision) \tag{8}$$

The F-Measure was used, because despite Precision and Recall being valid metrics in their own right, one can be optimised at the expense of the other ([22]). The F-Measure only produces a high result when Precision and Recall are both balanced, thus this is very significant.

A Receiver Operating Characteristic (ROC) curve analysis was also performed, as it shows the sensitivity (FN classifications) and specificity (FP classifications) of a test. The ROC curve is a comparison of two characteristics: TPR (true positive rate) and FPR (false positive rate). The TPR measures the number of *relevant* pages that were correctly identified.

$$TPR = TP / (TP + FN) \quad (9)$$

The FPR measures the number of incorrect classifications of *relevant* pages out of all *irrelevant* test pages.

$$FPR = FP / (FP + TN) \quad (10)$$

In the ROC space graph, FPR and TPR values form the x and y axes respectively. Each prediction (FPR, TPR) represents one point in the ROC space. There is a diagonal line that connects points with coordinates (0, 0) and (1, 1). This is called the “line of no-discriminations’ and all the points along this line are considered to be completely random guesses. The points above the diagonal line indicate good classification results, whereas the points below the line indicate wrong results. The best prediction (i.e. 100% sensitivity and 100% specificity), also known as ‘perfect classification’, would be at point (0, 1). Points closer to this coordinate show better classification results than other points in the ROC space.

4.2 Data Corpus

In this research, each web page is referred to as a sampling unit. Each sampling unit comprises a maximum of 100 features, which are selected after discarding much of the page content, as explained previously. The total number of unique features examined in the following experiments was 5217. The total number of sampling units used was 9436. These units were separated into two distinct sets: a training set and a test set.

The training set for the NB classifier consisted of 711 randomly selected, positive and negative examples (i.e. relevant and irrelevant sampling units). The test collection created consisted of data obtained from the remaining 8725 sampling units. The training set makes for under 10% of the size of the entire data corpus. This was intentionally decided, in order to really challenge the NB classifier. Compared to many other classification systems encountered, this is the smallest training set used.

4.3 Experimental Results

The first experiment that was carried out was to test our enhanced NB classifier against the standard naïve bayes algorithm, in order to determine whether or not the changes made to the original algorithm had enhanced the accuracy of the classifier. For this purpose, we stripped our system of the additional steps and executed both standard and enhanced NB classifiers with the above training and test data. The results showed that the enhanced NB classifier was comfortably in the lead by over 7% in both accuracy and F-Measure value.

In the second set of experiments, the sampling units analysed by the NB classifier were also run by a DT classifier and an NN classifier. The results were compared to determine which classifier is better at analysing attribute data from training web pages. The DT classifier is a ‘C’ program, based on the C4.5 algorithm in [16], written to evaluate data samples and find the main pattern(s) emerging from the data. For example, the DT classifier may conclude that all web pages containing a specific word are all relevant. More complex data samples however, may result in more complex configurations found.

The NN classifier used is also a ‘C’ program, based on the work published in [8]-[11]. MATLAB’s NN toolbox ([20]) could have also been used, however in past experiments MATLAB managed approximately 2 training epochs compared to the ‘C’ NN classifier, which achieved approximately 60,000 epochs in the same timeframe. We, therefore, abandoned MATLAB for the bespoke compiled NN system.

All three classifiers were initially trained with 105 sampling units and tested with a further 521 units, all consisting of a total of 3900 unique features. The NB classifier achieved the highest accuracy (**97.89%**), precision (**99.20%**), recall (**98.61%**) and F-Measure (**98.90%**) values, however, the DT classifier achieved the fastest execution time. The NN classifier, created with 3900 inputs, 3900 midnodes and 1 output, came last in all metrics and execution time.

For the most recent test, all classifiers were trained with 711 sampling units and they were then tested on the remaining 8725 sampling units. The NB and DT classifiers were adequately fast for exploitation and delivered good discriminations. The test results are shown in Table 2 and Table 3.

Table 2: Confusion Matrix for NB Classifier

		PREDICTED	
		IRRELEVANT	RELEVANT
ACTUAL	IRRELEVANT	TN / 876	FP / 47
	RELEVANT	FN / 372	TP / 7430

Table 3: Confusion Matrix for DT Classifier

		PREDICTED	
		IRRELEVANT	RELEVANT
ACTUAL	IRRELEVANT	TN / 794	FP / 129
	RELEVANT	FN / 320	TP / 7482

The vocabulary used in our experiments, consisting of 5217 features, was initially mapped onto a NN with 5217 inputs, one hidden layer with 5217 nodes and 1 output, in keeping with the standard configuration of a NN, where the number of midnodes is the same as the number of inputs. A fully connected network of this size would have over 27 million connections, with each connection involving weight parameters to be learnt. Our attempt at creating such a network resulted in the NN program failing to allocate the needed memory and crashing.

After paying more attention to the function complexity, we decided to change the number of midnodes to reflect this complexity. We, therefore, created a NN with 5217 inputs, 1 output and only 200 midnodes. This worked well and the resulting NN successfully established all connections. However, we realised that the NN would need to be extended (more nodes and midnodes created) to model any additional, new features, each time they are extracted from future web pages. This would potentially take the NN back to the situation where it fails to make all the required connections and this would be an unacceptable result for ATM. Technology exists for growing nodes; however, this would be complex and expensive. Furthermore, the NN took 200 minutes to train, which is much longer than the other classifiers, which took seconds for the same training sample. Therefore, we decided not to proceed with NNs any further, as they would be unsuitable for our project and other projects of this kind.

Table 4: Final Results

Classifier	Accuracy	Precision	Recall	F-Measure
NB Classifier	**95.20%**	**99.37%**	95.23%	**97.26%**
DT Classifier	94.85%	98.31%	**95.90%**	97.09%

Table 4 shows the Accuracy, Precision, Recall and F-Measure results achieved by the NB and DT classifiers, following the calculations in section 4.1. These results show that both classifiers achieve impressive results in the classification of attribute data in the training courses domain. The DT classifier outperforms the NB classifier in execution speed and Recall value (by 0.67%). However, the NB classifier achieves higher Accuracy, Precision and most importantly, overall F-Measure value, which is a very promising result.

This result is further confirmed by the comparison of the two classifiers on the ROC space (Fig.2.), where it is shown that the result set from the NB classifier falls closer to the 'perfect classification' point than the result set from the DT classifier.

Table 5: ROC Space Results

Classifier	FPR	TPR
NB Classifier	0.05092	0.95232
DT Classifier	0.13976	0.95899

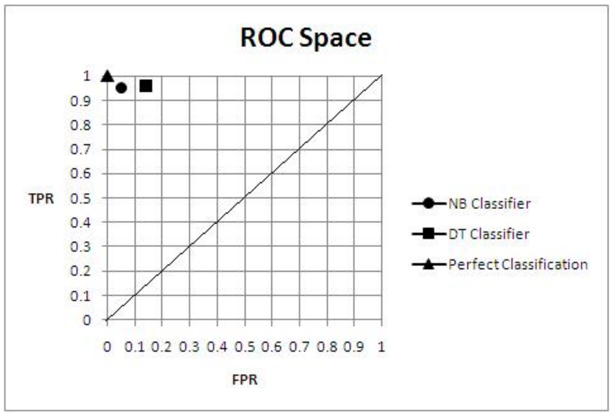

Fig. 2. ROC Space

The ROC space was created using the values in Table 5, following the calculations in equations (9) and (10).

5. Conclusions

To summarise, we succeeded in building a NB Classifier that can classify training web pages with 95.20% accuracy and an F-Measure value of over 97%. The NB approach was chosen as thorough analysis of many web pages showed independence amongst the features used. This approach was also a practical choice, because ATM, like many small companies, has limited hardware specifications available at their premises, which needed to be taken into account.

The NB approach was enhanced, however, to calculate the believed probability of features in each category. This additional step was added to handle situations when there is little evidence about the data, in particular during early stages of the classification process. Furthermore, the classification process was enhanced by taking into consideration not only the content of each web page, but also various important structures such as the page TITLE, META data and LINK information. Experiments showed that our enhancements improved the classifier by over 7%

in accuracy, in comparison with the original naïve bayes algorithm.

The NB classifier was tested against 8725 sampling units after being trained with only 711 units. This exact same sample was also analysed by a DT and a NN classifier and the results from all systems were compared to one-another. Our experiments showed that although some NN classifiers can be very accurate for some domains, they take the longest to train and have extensibility issues due to their extremely large and complex nature. It was therefore realized that NNs would be too expensive for ATM and unsuitable for handling a potentially large number of features created by the classification process.

On a more positive note, our experiments produced exciting findings for the application of the NB algorithm in the training courses domain, as the NB classifier achieved impressive results, including the highest Precision value (99.37%) and F-Measure (97.26%). Although some of the results are close to the results from the DT classifier, these experiments show that Naïve Bayes Classifiers should not be considered inferior to more complex techniques such as Decision Trees or Neural Networks. They are fast, consistent, easy to maintain and accurate in the classification of attribute data, such as the training courses domain. In one of our previous papers ([25]), we expressed our concern that many researchers go straight for the more complex approaches without trying out the simpler ones first. We hope this paper will encourage researchers to exploit the simpler techniques, as they can be, as this paper showed, more efficient and much less expensive.

The system may be improved further by reducing the number of features analysed. More research needs to be done to establish a possible cut off point for the extracted features. This may speed up the classification process as well as potentially improve the classifier further. More tests will also be done to confirm the NB classifier's success on a grander scale. In conclusion, this research has shown that the NB approach, enhanced to perform even with limited information, whilst analysing both web content and structural information, gives very promising results in the training courses domain, outperforming powerful and popular rivals such as decision trees and neural networks.

Acknowledgments

We would like to thank the whole team at ATM for the support and help they have offered us since the first day of the project. Also, thank you to both ATM and the Centre for Innovative and Collaborative Engineering (CICE) for funding our work.

References

[1] Addin, O., Sapuan, S. M., Mahdi, E., & Othman, M. "A Naive-Bayes classifier for damage detection in engineering materials", **Materials and Design**, 2007, pp. 2379-2386.
[2] Ceci, M., & Malerba, D. "Hierarchical Classification of HTML Documents with WebClassII", **Lecture Notes in Computer Science**, 2003, pp. 57-72.
[3] Chau, M., & Chen, H. "A machine learning approach to web page filtering using content and structure analysis", **Decision Support Systems**, Vol. 44, No. 2, 2007, pp. 482-494.
[4] Crestani, F. "An Adaptive Information Retrieval System Based on Neural Networks", in: **International Workshop on Artificial Neural Networks: New Trends in Neural Computation,** Vol. 686, 1993, pp. 732-737.
[5] Denis, F., Laurent, A., Gilleron, R., Tommasi, M. "Text classification and co-training from positive and unlabeled examples", in: **ICML Workshop: The Continuum from Labeled to Unlabeled Data**, 2003, pp. 80-87.
[6] Enhong, C., Shangfei, W., Zhenya, Z. & W. Xufa. "Document classification with CC4 neural network", in: **Proceedings of ICONIP,** Shanghai, China, 2001.
[7] Estruch, V., Ferri, C., Hernández-Orallo, J., & Ramírez-Quintana, M. J. "Web Categorisation Using Distance-Based Decision Trees", in: **International Workshop on Automated Specification and Verification of Web Site**, 2006, pp. 35-40.
[8] Fletcher, G.P & Hinde, C.J. "Interpretation of Neural Networks as Boolean Transfer Functions", **Knowledge-Based Systems**, Vol. 7, No. 3, 1994, 207-214.
[9] Fletcher, G.P & Hinde, C.J. "Using Neural Networks as a Tool for Constructing Rule Based Systems", **Knowledge-Based Systems**, Vol. 8, No. 4, 1995, 183-189.
[10] Fletcher, G.P & Hinde, C.J. "Producing Evidence for the Hypotheses of Large Neural Networks", **Neurocomputing**, Vol. 10, 1996, pp. 359-373.
[11] Hinde, C.J., & Fletcher, G.P., West, A.A. & Williams, D.J. "Neural Networks", **ICL Systems Journal,** Vol. 11, No. 2, 1997, pp. 244-278.
[12] Hu, W., Chang, K. & Ritter, G. "WebClass: Web Document Classification Using Modified Decision Trees", in: **38th Annual Southeast Regional Conference**, 2000, pp. 262-263.
[13] Langseth, H. & Nielsen, T. "Classification using Hierarchical Naïve Bayes models", **Machine Learning**, Vol. 63, No. 2, 2006, pp. 135-159.
[14] Liu, Z. & Zhang, Y. "A competitive neural network approach to web-page categorization", **International Journal of Uncertainty, Fuzziness & Knowledge Systems**, Vol. 9, 2001, pp. 731-741.
[15] Orallo, J. "Extending Decision Trees for Web Categorisation", in: **2nd Annual Conference of the ICT for EU-India Cross Cultural Dissemination**, 2005.
[16] Quinlan, J. R. "Improved use of continuous attributes in C4.5", **Journal of Artificial Intelligence Research**, Vol. 4, 1996, pp. 77-90.
[17] Russell, S. & Norvig, P. **Artificial Intelligence: A Modern Approach**, London: Prentice Hall, 2003.

[18] Segaran, T. **Programming Collective Intelligence**, U.S.A: O'Reilly Media Inc, 2007.
[19] Tal, B. "Neural Network - Based System of Leading Indicators", **CIBC World Markets**, 2003.
[20] TheMathsWork, http://www.mathworks.com/products/neuralnet/
[21] Towell, G. & Shavlik, J. "Extracting Refined Rules from Knowledge-Based Neural Networks", **Machine Learning**, Vol. 13, No. 1, 1993, pp. 71-101.
[22] Turney, P. "Learning to Extract Keyphrases from Text", **Technical Report ERB-1057**, Institute for Information Technology, National Research Council of Canada, 1999.
[23] Wang C., Ding C., Meraz R., Holbrook S. "PSoL: a positive sample only learning algorithm for finding non-coding RNA genes", **Bioinformatics**, Vol. 22, No. 21, 2006, pp. 2590-2596.
[24] Wang, L., Li, X., Cao, C. & Yuan, S. "Combining decision tree and Naïve Bayes for classification", **Knowledge Based Systems**, Vol. 19, 2006, pp. 511-515.
[25] Xhemali, D., Hinde, C.J. & Stone, R.G. 2007. "Embarking on a Web Extraction Project", in: **The 2007 UK Conference on Computational Intelligence**, 2007.
[26] Zhang, N. "Hierarchical latent class models for cluster analysis", in: **18th National Conference on Artificial Intelligence**, 2002, pp. 230-237.

Daniela Xhemali is an Engineering Doctorate (EngD) student at Loughborough University, UK. She received a First Class (Honours) BSc in Software Engineering from Sheffield Hallam University in 2005 and an MSc with Distinction in Engineering, Innovation and Management from Loughborough University in 2008. Daniela Xhemali has also worked in industry for two years as a Software Engineer, programming multi-user, object oriented applications, with large database backend. Her current research focuses on Web Information Retrieval and Extraction, specifically on the use of Bayes Networks, Decision Trees and Neural Networks in the classification of web pages as well as the use of Genetic Programming and Evolution in the extraction of specific web information.

Dr. Christopher J. Hinde is a Senior Lecturer at Loughborough University. He is the Programme Director of the Computer Science & Artificial Intelligence group as well as the Programme Director of the Computer Science & E-business group. Dr. Hinde is also the leader of the Intelligent and Interactive Systems Research division. His research interests include: Artificial intelligence, fuzzy reasoning, logic programming, natural language processing, neural nets etc.

Dr. Roger G. Stone is a lecturer at Loughborough University. He is DANS Coordinator and the Quality Manager at Loughborough University. Dr. Stone is also a member of the Interdisciplinary Computing Research Division. His research interests include: Web programming, web accessibility, program specification techniques, software engineering tools, compiling etc.