Experience with Learning Agents which Manage Internet-Based Information

Peter Edwards, David Bayer, Claire L. Green & Terry R. Payne *
Department of Computing Science
King's College, University of Aberdeen
Aberdeen, Scotland, AB9 2UE
{pedwards, dbayer, claire, terry}@csd.abdn.ac.uk

Abstract

To provide assistance with tasks such as retrieving USENET news articles or identifying interesting Web pages, an intelligent agent requires information about a user's interests and needs. Machine learning techniques are now being used to acquire this information. A general architecture is presented, and two approaches to learning through observation are described. An instantiation of the architecture is then evaluated.

Introduction

The recent, rapid growth of the Internet has led to enormous amounts of on-line information. However, as the volume of this information has increased, so have the problems encountered by users in dealing with it. Software agents have been proposed as a solution to this problem. They may be characterised as systems which aid and assist a user with a common task, employing some degree of learning or adaptation to improve the quality of their assistance over time.

Agents have been described as:

> "...computer programs which employ Artificial Intelligence techniques to provide assistance to a user dealing with a particular computer application..." (Maes 1994)

Many systems have been developed to deal with filtering information, such as electronic mail or USENET news articles (Sheth 1994; Payne, Edwards, & Green 1995). Other systems have been developed which actively seek out information (Voorhees 1994; Bayer 1995), and the number of such systems being developed is increasing.

If an agent is to be of assistance, it requires knowledge about the domain application and/or the user. Two approaches have traditionally been used to provide an agent with knowledge about its task domain. The first and most common method is for users to provide their own rules, e.g. by using a scripting language. Users of systems such as the Information Lens (Malone *et al.* 1987) have to define a set of rules to filter and sort incoming mail messages. Other systems rely on user-defined scripts which contain short programs, such as those employed by the Information Retrieval Agent (*IRA*) (Voorhees 1994).

The second method makes use of traditional knowledge engineering techniques to identify background knowledge about the application and the user. This technique has been applied to advisory agents, such as UCEgo (Chin 1991), which provides advice on using the UNIX operating system. Whilst this shifts the task of programming the agent from the user to the Knowledge Engineer, the agent will not be customised for a particular user. Thus, approaches such as this cannot be used for personalised tasks, such as information filtering.

An alternative solution is to build a *profile* which reflects the user's preferences when using an application, such as a World-Wide Web browser. A variety of learning mechanisms have been employed within agent systems to induce such profiles. Genetic algorithm approaches have been used for news filtering agents (Sheth 1994); symbolic rule induction algorithms such as C4.5 (Quinlan 1993) and CN2 (Clark & Niblett 1989), and instance-based approaches (Stanfill & Waltz 1986; Kibler & Aha 1987) have been used with mail and news filtering tasks (Metral 1993; Payne & Edwards 1995; Payne, Edwards, & Green 1995) and automated Web browsing (Bayer 1995; Balabanović & Yun 1995). Minimum Description Length techniques have been explored in USENET news filtering (Lang 1995), and relational learning algorithms such as FOIL (Quinlan 1990) have been applied to text categorisation (Cohen 1995). Unfortunately, it is difficult to assess the relative perfor-

* Terry R. Payne and David Bayer acknowledge financial support provided by the UK Engineering & Physical Sciences Research Council (EPSRC).

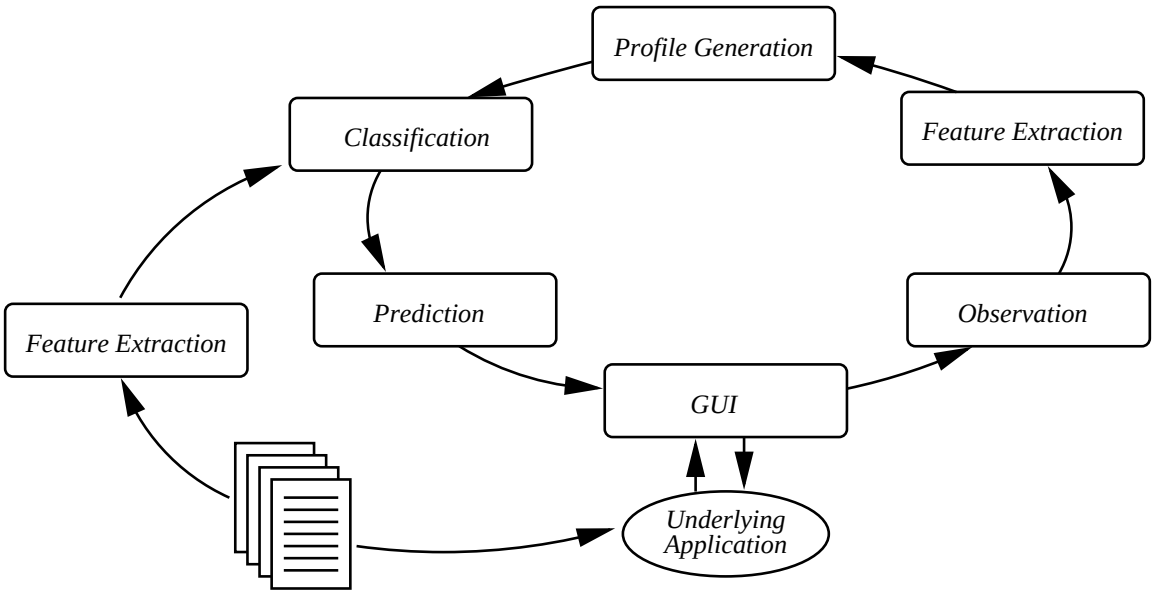

Figure 1: A Learning Interface Agent Architecture.

mance of these techniques for agent systems, due to the ad hoc nature of much of the evaluation performed. This can perhaps be explained by the lack of standard data sets within the UCI Machine Learning Data Repository (Murphy & Aha 1994). One recent trend has been the development of agent systems to act as comparative testbeds for learning techniques (Armstrong *et al.* 1995; Blum 1995; Lang 1995; Payne & Edwards 1995).

Related Work

A number of systems have been developed which combine machine learning techniques with information filtering techniques to create user profiles. Lira (Balabanović & Yun 1995) employed a *term-frequency/inverse-document frequency (tfidf)* weighting to extract a fixed number of terms from World-Wide Web pages. These were then used to construct a number of vectors which were used by a Web robot to search and score matching pages. The highest scoring pages were then presented to the user, who rated them on an 11 point scale. This feedback was then used to adjust the weights within the vectors.

Lang (1995) studied how well machine learning techniques performed compared to traditional information filtering techniques within the news-filtering system, NewsWeeder. A vector of word frequencies was first created from news articles, and then used to train a learning algorithm based on the *Minimum Description Length (MDL)* principal. The performance of this algorithm was then compared to an approach involving a *tfidf* weighting. The vector of word frequencies was weighted using this calculation. They were then grouped by classification, and the weights were averaged to form a *prototype* vector. New vectors were then compared to these prototype vectors using a cosine similarity measure. Results indicated that learning techniques can be used to identify news articles of interest, and that they have a number of benefits over the use of information retrieval techniques such as *tfidf* (see Lang (1995) for details).

A similar study has been carried out with World-Wide Web documents. WebWatcher (Armstrong *et al.* 1995) compared the use of a *tfidf* based classifier with the linear threshold algorithm, *Winnow* (Littlestone 1988) in searching for links to technical papers. Each training instance was represented by a vector of approximately 530 boolean features, taken from four fields. The first three fields contained features from the Web documents, the fourth contained words entered by the user when defining the information search goal. The vectors were used by different classification techniques to predict which links a user should follow to reach a technical paper. Four techniques were compared: *Winnow*, *tfidf*, a statistical based approach (*Wordstat*), and a random prediction generator, used to provide a baseline measure against which to compare the other methods. Results showed that the three learning methods could all be used to successfully learn search control knowledge for the Web, by observing searches performed by users.

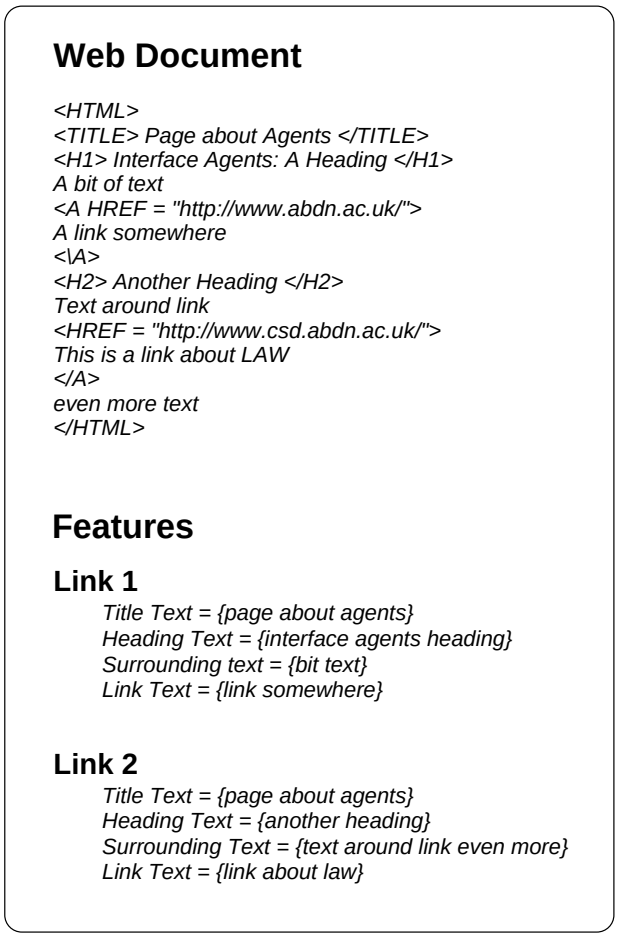

Figure 2: Extracting Features from a HTML Document.

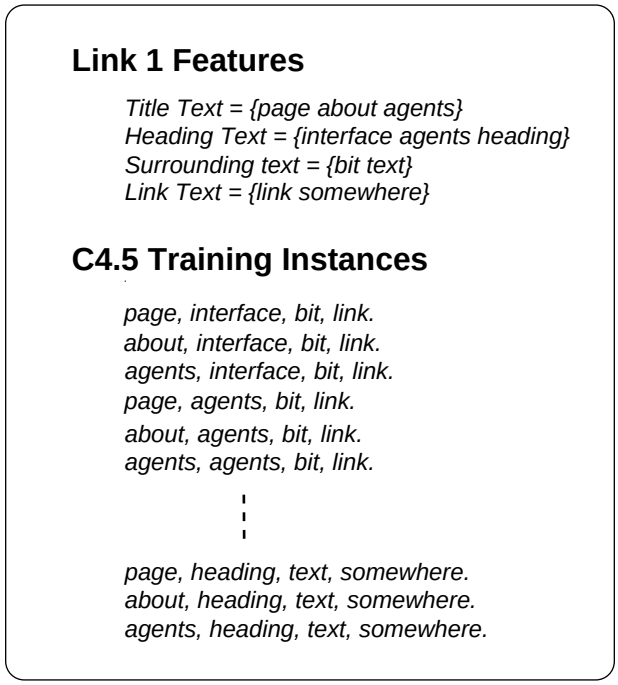

Figure 3: Generating Instances from Features Extracted from a HTML Document.

Agent Model

To explore the issues involved in employing learning mechanisms within existing software applications, the agent architecture shown in Figure 1 was developed. Agents based on this general architecture have been embedded within a number of applications including a World-Wide Web browser and a USENET news reader. The architecture can be divided into two broad areas: the *Profile Generation Phase* and the *Classification/Prediction Phase*. The Profile Generation phase is responsible for inducing the user profile, and consists of three stages: the *Observation Stage*, the *Feature Extraction Stage* and the *Profile Generation Stage*. Actions performed by the user on a document (news article, Web page, etc.) are recorded together with the text of the document. Features are extracted from these observations, and used to create a training instance. The training instances are then used to induce the user profile. As the application is used over time, the set of observations grows in size. It is periodically pruned to remove the oldest observations, so that the profile induced reflects the user's current interests.

The Classification/Prediction phase is responsible for determining the actions to be performed on new documents. It consists of a *Feature Extraction Stage*, a *Classification Stage* and a *Prediction Stage*. Features are extracted from each document, and the user profile employed to generate a classification (with an associated confidence rating). The confidence rating is used by the Prediction Stage to determine whether a prediction should be made. If the rating is lower than the *confidence threshold*, then no prediction will be made. If, however, the confidence rating is higher than this threshold, the prediction will be passed on to the application.

Feature Extraction Issues

Various methods can be used to extract features from documents. By feature we mean a word or *term*, and the field from which it was extracted within the document, such as the *Subject* field in a USENET News article, or the *Title* of a Web page (see Figure 2). All terms in each field are rated using a measure of term

significance[1] and the N highest terms then extracted from the document. Figure 2 illustrates how four features are extracted to represent each link in a sample HTML document.

The measure used to rank terms may consider each document in isolation (e.g. measuring the frequency of different terms within the document), or consider the document with respect to the whole collection of documents (*corpus*). For example, measures such as the *tfidf* weighting consider the frequency of terms across all documents. Once features have been extracted from a document, they need to be presented to the learning algorithm. One approach is to generate a vector of features to represent the document. However, such vectors can be large, as their size is dependent on the number of different features selected from the corpus. Various approaches have been proposed to limit the size of vectors produced. For example, WebWatcher considers only a subset of features from the whole corpus (Armstrong *et al.* 1995). We have investigated two approaches which avoid the use of vectors.

The first approach relies on creating a number of training instances for each observation (Figure 3). Learning algorithms such as C4.5 (Quinlan 1993) and CN2 (Clark & Niblett 1989) expect training instances which contain a single value for each attribute. By creating groups of training instances in this way, data can be generated for use by these algorithms.

$$\Delta(\mathcal{D}, x, y) = \sum_{a=1}^{A} \left(\frac{\sum_{x_v=1}^{x.a_n} \sum_{y_v=1}^{y.a_n} d(\mathcal{D}, x_v, y_v) w(\mathcal{D}, x_v)}{x.a_n \times y.a_n} \right) \quad (1)$$

$$d(\mathcal{D}, x_v, y_v) = \sum_{c \in \mathcal{C}} \left(\delta(\mathcal{D}, c, x_v) - \delta(\mathcal{D}, c, y_v) \right)^2 \quad (2)$$

$$w(\mathcal{D}, x_v) = \sqrt{\sum_{c \in \mathcal{C}} \delta(\mathcal{D}, c, x_v)^2} \quad (3)$$

The second approach involved developing a learning algorithm which can learn from features such as those illustrated in Figure 2. Instance-Based algorithms derive a classification by comparing a new instance with previously classified instances. It is possible to modify the comparison so that multiple values can be compared for each attribute. The Value-Distance Metric, used in the Memory-Based Reasoning Algorithm (Stanfill & Waltz 1986), provides a means of calculating a similarity measure (*distance*) between symbolic values. For this reason, it was modified (see Equation 1) so that multiple distances could be calculated and averaged when comparing attributes. This resulted in the algorithm IBPL1 (Payne & Edwards 1995). A similarity measure is calculated by determining the distance $d(\mathcal{D}, x_v, y_v)$ between two values for attribute a (Equation 2), and a weighting value $w(\mathcal{D}, x_v)$ for value x_v (Equation 3). Table 1 lists the notation used.

Symbol	Description
$\mathcal{D}$	Training set
$\mathcal{C}$	The set of all classes in $\mathcal{D}$
a	Attribute
A	Number of attributes
x	Unclassified instance
y	Instance in training set $\mathcal{D}$
$x.a_n$	Number of values in attribute a of x
$y.a_n$	Number of values in attribute a of y
x_v	Value considered in instance x
y_v	Value considered in instance y
$\delta(\mathcal{D}, c, x_v)$	Ratio of the number of times value x_v occurs in training instances of class c, to the number of times x_v occurs in the training set.
$\delta(\mathcal{D}, c, y_v)$	Ratio of the number of times value y_v occurs in training instances of class c, to the number of times y_v occurs in the training set.

Table 1: Notation used in Equations 1, 2 & 3

Magi - Mail Agent Interface

Magi was the first system to explore the use of the architecture shown in Figure 1, and was developed to aid a user in sorting incoming electronic mail (Payne & Edwards 1995). *Xmail*, a graphical user interface for mail was modified to record mail filing and mail deletion operations.

Two versions of Magi were constructed to explore how profiles could be induced from these observations. The first generated groups of instances for each observation. These were then used by CN2 to induce the profile. The second version used IBPL1. The performance of both methods was compared. Coverage (i.e. how many new messages could be classified) and accuracy (i.e. whether these classifications were correct) were recorded. A total of 408 mail messages were used in the evaluation, sorted into 12 mail boxes. The study found that the overall accuracy of predictions made by IBPL1 was slightly lower (57%) than those for CN2 (65%), although the results for individual mail boxes varied. It should be noted that the rules gener-

[1] Commonly occurring words such as *and, or,* etc. are removed before performing these calculations.

ated by CN2 were biased towards features containing fewer terms, as these appeared more frequently in the training data. These results are described in detail in (Payne & Edwards 1995).

IAN - Intelligent Assistant for News

IAN is a modified version of the UNA system (Green 1995) which aids a user in identifying interesting USENET news articles. As the user reads news in a modified version of the *xrn* news browser, they provide a rating on a four point scale, in order to indicate their level of interest in each article. When the user next reads news, they can instruct the system to filter out any uninteresting articles. All remaining articles are presented to the user for feedback.

The IAN test set contained 1200 news articles, split evenly across 6 different newsgroups (*alt.lefthanders*, *sci.stat.math*, *rec.food.cooking*, *rec.food.veg.cooking*, *rec.humor*, and *alt.education.research*). Two sets of tests were carried out when evaluating the performance of the system; one to identify whether a correct prediction could be made for *broad classifications*, (i.e. articles rated 1 or 2 were grouped as 'uninteresting', while articles rated 3 or 4 were grouped as 'interesting') and the other for *narrow classifications* (i.e. for a correct classification on the scale 1-4). Coverage and accuracy were calculated for each test. Experimentation showed that the accuracy of predictions using C4.5 and IBPL1 were, on average, approximately the same, with C4.5 generally performing better when predicting broad classifications (C4.5 63%, IBPL1 61%), and IBPL1 generally performing better when predicting narrow classifications (IBPL1 40%, C4.5 36%).

LAW : A Learning Apprentice for the World Wide Web

LAW (Bayer 1995) is a system that helps a user find new and interesting information on the World-Wide Web. It provides assistance in two ways: by interactively suggesting links to the user as they browse the Web; and through the use of a separate Web robot that autonomously searches for pages that might be of interest.

LAW is based on the general agent architecture shown in Figure 1. However, two different profiles are generated: the *link profile* and *page profile*. The link profile represents the type of links which the user typically explores as they browse the Web, and is used to provide interactive assistance as the user views new pages. The page profile describes the type of pages which the user finds interesting, and is used in conjunction with the link profile to control the Web robot.

Interactive Assistance

LAW provides interactive assistance as the user browses the Web, by highlighting the links which appear most interesting on each page visited. In this way, the system immediately focuses the user's attention on the salient parts of a page.

The *Chimera* Web browser has been modified so that pages can be analysed prior to being displayed. The links within each document are extracted and classified using the link profile. Those that are classified as interesting are highlighted by inserting an icon immediately prior to the link in the document. Once these modifications have been made, the page is displayed to the user.

The Web Robot

The robot is a separate application that explores the World-Wide Web using a best-first search through the links it encounters (Figure 4).

The robot is given a number of starting points by the user from which to begin exploring. It then enters the following cycle: load a page, extract and analyse the links within the page, analyse the overall content of the page. The links extracted are classified using the link profile. Those that are classified as interesting are given a numerical rating which depends on how closely they match the type of links which the user typically explores. The rating measure used is the confidence value returned by the classification engine. The rating given to each link is used to order the search, with the highest scoring links being explored first. Pages are analysed in a similar manner, using the page profile. Those classified as interesting are given a rating, and the highest rated pages are presented to the user.

Feature Extraction

The observations collected through the modified Web browser consist of HTML documents visited by the user and actions performed on these documents. The actions recorded are browser functions such as the user saving the location of a page as a bookmark, or printing a page. From this data a set of training instances must be constructed for each of the profiles.

To construct the set of instances needed for the link profile, the terms associated with each link in each document must be identified. Four distinct groups of terms can be extracted for each link: terms in the link text, terms in the text surrounding the link, terms in the heading nearest the link and terms in the title of the document. This process is depicted in Figure 2. Links explored by the user are used as training instances. Each instance is given a classification of either *interesting* or *not interesting*. If a link led to a page

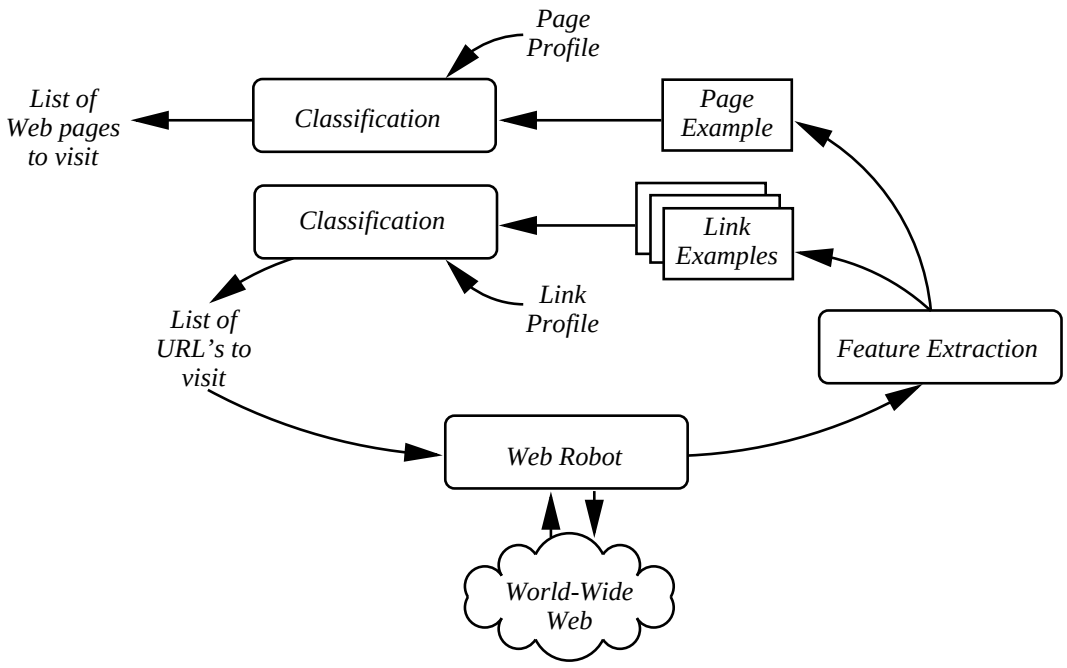

Figure 4: The Role of the Link and Page Profiles in Controlling the Web Robot.

that the user saved as a bookmark, printed, or visited frequently then it is classified as interesting, otherwise as not interesting.

The training data required for the page profile is constructed in a similar manner. An instance is created for each unique document. Four fields are used to represent the contents of a page: terms in the title of the document, terms in the headings within the document, terms in the links, and terms in the remainder of the document. An instance is classified as interesting if the page was visited frequently, saved as a bookmark or printed.

Only terms considered significant are extracted from the raw data. Initially, HTML tags, such as $< html >$ and $< p >$, and low information content words, such as *the*, *and*, etc. are removed. The remaining terms are rated using a measure of term significance. The highest rated terms are used in the instances. A number of different measures of term significance have been compared, including *term frequency*, *term frequency/inverse document frequency (tfidf)*, and *term relevance* (Salton & McGill 1983).

The *term frequency* measure (Equation 4) assumes that the importance of a term is directly proportional to the frequency with which it appears within a document.

$$Weight_{ik} = \frac{Freq_{ik}}{NoWords_i} \tag{4}$$

$Freq_{ik}$ is the frequency of term k in document i and $NoWords_i$ is the number of terms in document i.

The *tfidf* measure (Equation 5) assigns a greater significance to terms that are good discriminators between documents in a collection. The measure compares how frequently a term appears in a document against the number of other documents which contain that term. The weighting formula is as follows:

$$Weight_{ik} = Freq_{ik} \cdot [\log_2 n - \log_2 DocFreq_k + 1] \tag{5}$$

where n is the total number of documents in the collection and $DocFreq_k$ is the number of documents which term k appears in.

The *term relevance* measure (Equation 6) gives preference to terms that differentiate between classes of documents. The calculation gives preference to terms that frequently occur in one class of documents and infrequently in the rest.

$$TermRelevance_{kc} = \frac{r_{kc}/(R_c - r_{kc})}{s_{kc}/(I_c - s_{kc})} \tag{6}$$

$TermRelevance_{kc}$ is the significance weight given to a term k in a document belonging to class c. r_{kc} is the number of documents belonging to class c that contain term k. R_c is the number of documents in class c. s_{kc} is the number of documents not in class c that also contain the term k. I_c is the total number of documents not in class c.

The *term relevance* measure depends on knowledge about the classification of each document. In the training data this is known, and appropriate significance values can be calculated. A problem arises when feature extraction must be performed on new documents, as the class of the document is unknown. Consequently, it is necessary to use an alternative weighting formula for these documents. As the *term-frequency* weighting method requires no *a-priori* knowledge about the class, it is used to calculate significance values for unclassified documents.

Experimentation Methodology and Results

LAW's performance was assessed using three different data sets which were constructed using the modified browser. Each set contained approximately 120 pages relating to one particular topic area: *food*, *human rights* and *sport* (see Table 2). The documents in the *food* data set contained long lists of links. In contrast, the pages in the *human rights* data set usually consisted of large amounts of text interspersed with occasional links. The *sport* data set contained a mixture of both types of document.

Profile	Data Set	Positive Instances	Negative Instances	Total
Page	Food	73	51	124
	Human	62	55	117
	Sport	66	59	125
Link	Food	128	49	177
	Human	48	54	102
	Sport	176	122	298

Table 2: Numbers of Instances used to Evaluate Page and Link Profiles.

A number of experiments were performed with LAW to evaluate the impact of the three different weighting formulae described above. Two alternative learning strategies were also considered. The first involved the use of C4.5 to learn a profile from groups of instances (see earlier discussion). The second approach used IBPL1.

Each data set was divided into a training and a test set. Features were extracted for the training set using one of the weighting formulae. The *term-frequency* weighting method was used to extract features for the test set. Each test was repeated 25 times, and average *accuracy* and *coverage* values calculated. Accuracy results varied across the data sets. Table 3 lists the best performance figures obtained when training with 80% of the data.

Best Performance Figures for IBPL1				
Profile	Data Set	Accuracy %	Coverage %	Confidence Threshold
Page	Food	75	79	2.5
	Human	84	77	1.5
	Sport	73	78	2.0
Link	Food	83	83	2.0
	Human	70	84	1.0
	Sport	78	91	0.5

Best Performance Figures for C4.5				
Profile	Data Set	Accuracy %	Coverage %	Confidence Threshold
Page	Food	64	97	0.5
	Human	71	96	0.5
	Sport	63	99	0.5
Link	Food	83	78	0.8
	Human	68	76	0.6
	Sport	77	90	0.6

Table 3: Best Performance Figures for each Data Set.

In general, no single weighting strategy produced consistently better results than any other. Graphs 5 & 6 illustrate the performance of the three weighting strategies when used to generate a page profile. The accuracy of predictions made varies with the confidence threshold used. As the threshold increases, the number of predictions made (i.e. *coverage*) decreases, as those with a low confidence rating are rejected. This results in an overall increase in accuracy. Graphs 7 & 8 illustrate this phenomenon for both IBPL1 and C4.5.

A confidence threshold of zero was used to compare the performance of C4.5 to that of IBPL1 for both page and link profiles. This resulted in both algorithms achieving the same coverage (approximately 100%). In general, IBPL1 performs as well as, or slightly better than C4.5. Graphs 9 & 10 compare C4.5 and IBPL1 (using the *term-relevance* weighting method).

The performance of the Web robot was evaluated by performing three test runs using each of the different data sets. A typical run lasted approximately six hours, during which 700 documents were retrieved from around 200 unique hosts. The pages suggested by the robot were assessed using two criteria. Firstly, whether they were in the correct topic area, i.e. in the same domain as the data set. Secondly, if the pages were of specific interest to the user[2]. The results obtained for a single test run using the food data set can be seen in Table 4. This table shows the number

[2]This was determined by one of the authors.

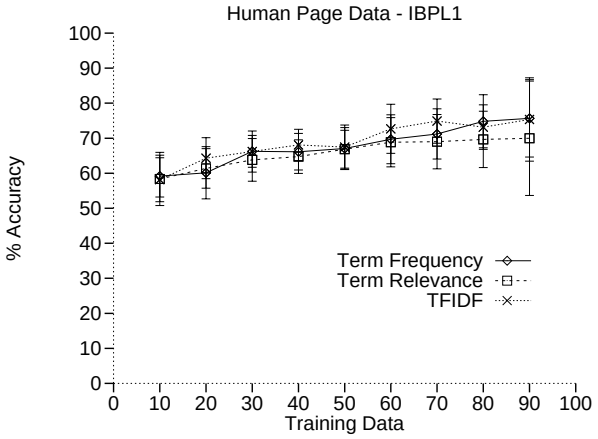

Figure 5: A Comparison of the Three Different Weighting Methods when used with IBPL1.

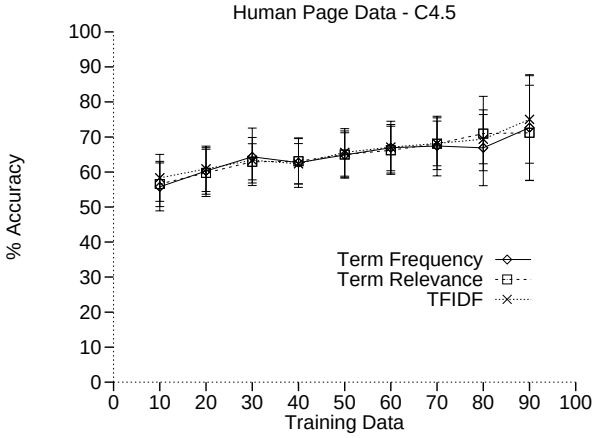

Figure 6: A Comparison of the Three Different Weighting Methods when used with C4.5.

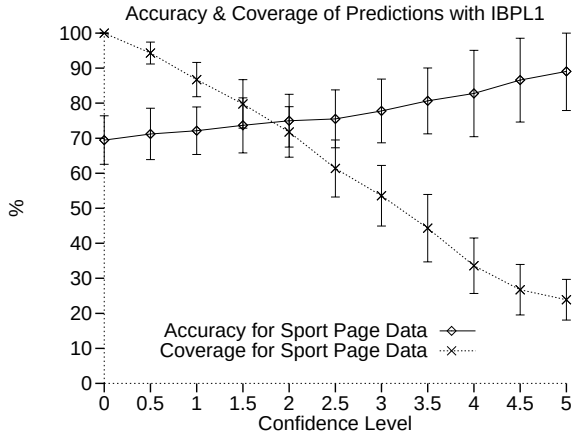

Figure 7: Coverage and Accuracy for IBPL1 as the Confidence Threshold Increases.

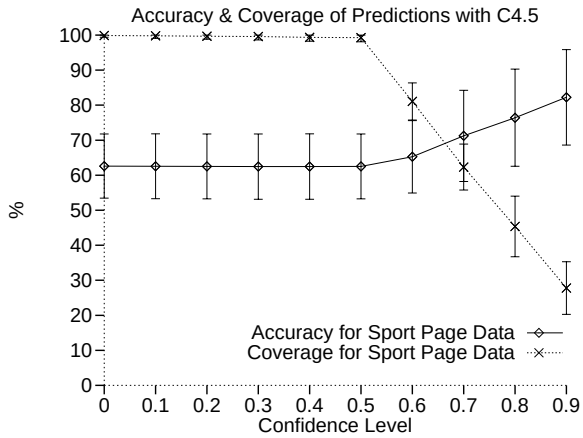

Figure 8: Coverage and Accuracy for C4.5 as the Confidence Threshold Increases.

of pages that would have been suggested if a particular confidence threshold had been set. This clearly demonstrates that the robot is able to discover interesting pages. However, its performance declines rapidly as its confidence in the suggestions drops. Similar results were obtained for the tests on the other data sets. Overall, the performance of the robot is directly related to the quality of the page and link profiles.

Discussion

A number of important conclusions have been drawn from the experiments performed with LAW. The methods used to extract features appear not to have an effect on the accuracy of predictions. This may be due to different weighting measures being used in the Profile Generation phase and Classification/Prediction phase. For this reason, alternative weighting measures need to be investigated that do not require *a-priori* knowledge of the class of the document. It would also be interesting to examine techniques that are not based solely on term frequency calculations. For example, methods which identify word associations, extract sentences or use a thesaurus to recognise related words.

The confidence threshold provides a means of improving the accuracy of predictions at the expense of the coverage. This is significant when configuring the different systems. In LAW, it is more important to suggest a small number of links that would be found interesting, than to present a larger number of uninteresting links. For systems such as IAN, where articles are presented to the user if a prediction cannot be made, selecting the appropriate threshold value is

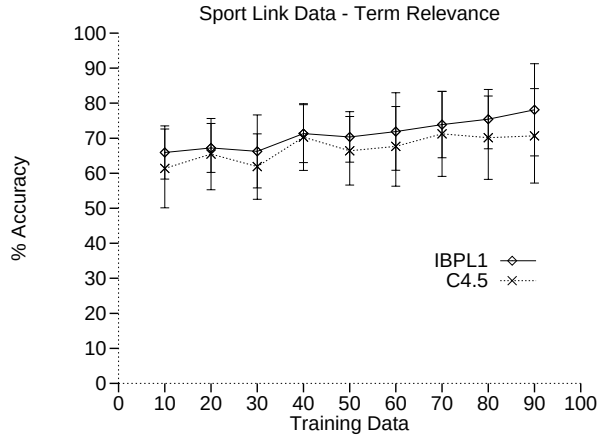

Figure 9: A Comparison of C4.5 & IBPL1 - Learning Link Profiles.

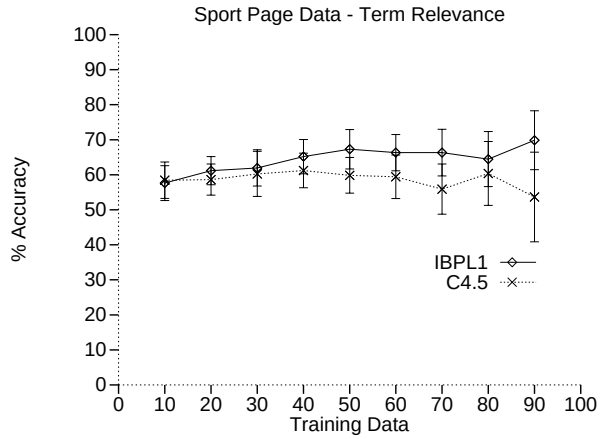

Figure 10: A Comparison of C4.5 & IBPL1 - Learning Page Profiles.

more difficult. If the threshold is lowered to increase the coverage, then the accuracy of the predictions will fall. Unless the agent is able to provide accurate and consistent advice, the user will lose trust in the resulting predictions. This accuracy/coverage tradeoff is discussed further by Armstrong *et al.* (1995).

The IBPL1 algorithm uses a novel approach to overcome the problem of learning with multi-valued attributes. However, the results presented here indicate that it offers no obvious advantages over the use of a symbolic rule induction algorithm for learning concepts for information management. Work is underway to improve the accuracy of IBPL1 for such tasks, by investigating different feature weighting mechanisms, such as those described in (Wettschereck, Aha, & Mohri 1995).

Confidence Threshold	Number of Suggested Pages	Percentage in Correct Domain	Percentage Interesting
1.97	10	90	60
0.79	30	63.33	40
0.71	50	52	36
0.45	100	41	21

Table 4: Summary of Results for a Single Test Run of the Web Robot on the Food Data Set.

Conclusions

The work described in this paper has demonstrated that the application of machine learning techniques to existing information management tools can enhance the capabilities of such software. Tools such as LAW can achieve an accuracy of 68-83% when identifying interesting links on a World-Wide Web page.

This work has highlighted a number of issues to consider when embedding learning techniques within information management applications. Firstly the accuracy/coverage tradeoff is crucial, as an agent system is more beneficial to a user if its assistance is infrequent but correct, rather than frequent but inaccurate. A second issue concerns which features to select for the learning task. Whilst a number of techniques were evaluated in LAW, no single method consistently outperformed any other. Hence, other techniques should be investigated, such as extracting n-grams or using methods from natural language processing.

The IBPL1 algorithm has so far failed to offer any significant advantages over other learning methods. Work is needed to improve the algorithm's accuracy over that achieved by other techniques.

Acknowledgements

We gratefully acknowledge the authors and maintainers of *xmail*, *xrn* and *Chimera* for allowing us to use their software. Thanks must also go to Nick Murray for allowing us to run our CPU-intensive tests on the network at the most inappropriate times.

References

Armstrong, R.; Freitag, D.; Joachims, T.; and Mitchell, T. 1995. WebWatcher: A Learning Apprentice for the World Wide Web. In *Working Notes of the AAAI Spring Symposium Series on Information Gathering from Distributed, Heterogeneous Environments.* Menlo Park, CA:AAAI.

Balabanović, M; Shoham, Y., and Yun, Y. 1995. An Adaptive Agent for Automated Web Browsing. *Jour-*

nal of Image Representation and Visual Communication 6(4).

Bayer, D. 1995. A Learning Agent for Resource Discovery on the World Wide Web. MSc Thesis, Department of Computing Science, University of Aberdeen, Scotland.

Blum, A. 1995. Empirical Support for Winnow and Weighted-Majority Based Algorithms: Results on a Calendar Scheduling Domain. In *Proceedings of the 12th International Conference on Machine Learning*, 64–72.

Chin, D. N. 1991. Intelligent Interfaces As Agents. In Sullivan, J. W., and Tyler, S. W., eds., *Intelligent User Interfaces*. New York, New York:ACM Press. 177–206.

Clark, P., and Niblett, T. 1989. The CN2 Induction Algorithm. *Machine Learning* 3:261–283.

Cohen, W. 1995. Text Categorization and Relational Learning. In *The 12th International Conference on Machine Learning*, 124–132.

Green, C. 1995. USENET News Agent. BSc Final Year Project Report, Department of Computing Science, University of Aberdeen, Scotland.

Kibler, D., and Aha, D. 1987. Learning Representative Exemplars of Concepts: An Initial Case Study. In *Proceedings of the 4th International Workshop on Machine Learning*, 24–30.

Lang, K. 1995. NewsWeeder: Learning to Filter Netnews. In *Proceedings of the 12th International Machine Learning Conference (ML95)*, 331–339. San Francisco, CA:Morgan Kaufmann.

Littlestone, N. 1988. Learning Quickly When Irrelevant Attributes Abound. *Machine Learning* 2(4):285–318.

Maes, P. 1994. Agents that Reduce Work and Information Overload. *Communications of the ACM* 37(7):30–40.

Malone, T.; Grant, K.; Turbak, F.; Brobst, S.; and Cohen, M. 1987. Intelligent Information-Sharing Systems. *Communications of the ACM* 30(5):390–402.

Metral, M. 1993. Design of a Generic Learning Interface Agent. BSc Thesis, Department of Electrical Engineering and Computer Science, MIT.

Murphy, P., and Aha, D. 1994. UCI Repository of Machine Learning Databases. Department of Information and Computer Science, University of California, Irvine, CA. [http://www.ics.uci.edu/~mlearn/MLRepository.html].

Payne, T., and Edwards, P. 1995. Interface Agents that Learn: An Investigation of Learning Issues in a Mail Agent Interface. Technical Report AUCS/TR9508, Department of Computing Science, University of Aberdeen, Scotland. Submitted to Applied Artificial Intelligence.

Payne, T. R.; Edwards, P.; and Green, C. L. 1995. Experience with Rule Induction and k-Nearest Neighbour Methods for Interface Agents that Learn. In *ML95 Workshop on Agents that Learn from Other Agents*.

Quinlan, J. 1990. Learning Logical Definitions from Relations. *Machine Learning* 5:239–266.

Quinlan, J. 1993. *C4.5 Programs for Machine Learning*. San Mateo, CA:Morgan Kaufmann.

Salton, G., and McGill, M. 1983. *Introduction to Modern Information Retrieval*. New York: McGraw-Hill.

Sheth, B. 1994. A Learning Approach to Personalized Information Filtering. Master's Thesis, Department of Electrical Engineering and Computer Science, MIT.

Stanfill, C., and Waltz, D. 1986. Toward Memory-Based Reasoning. *Communications of the ACM* 29(12):1213–1228.

Voorhees, E. 1994. Software Agents for Information Retrieval. In *Software Agents: Papers from the 1994 Spring Symposium*, 126–129. Menlo Park, CA:AAAI Press.

Wettschereck, D.; Aha, D.; and Mohri, T. 1995. A Review and Comparative Evaluation of Feature Weighting Methods for Lazy Learning Algorithms. Technical Report AIC-95-012, NRL NCARAI.