Using Machine Learning to Enhance Software Tools for Internet Information Management

Claire L Green & Peter Edwards

Department of Computing Science,
King's College, University of Aberdeen,
Aberdeen, Scotland, AB24 3UE
{claire, pedwards}@csd.abdn.ac.uk

Abstract

This paper discusses the issues involved in the application of machine learning techniques to the management of Internet-based information. We present a general architecture, and describe how this has been instantiated in several different applications. The first three of these systems provide assistance to a user sorting incoming mail, reading USENET news or identifying World-Wide Web pages of interest; the final system constructs a personalised on-line newspaper, assembled from documents gathered by a Web robot. A number of machine learning techniques have been used in the construction of these systems; some comparative results are presented.

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. Interface agents have been proposed as a solution to this problem and can be characterised as systems which "employ Artificial Intelligence techniques to provide assistance to users dealing with a particular computer application" (Maes 1994a).

In order to be able to assist the user, an agent must be provided with knowledge of its domain. Two approaches have traditionally been employed to achieve this. The first and most common approach is for users to provide the agent with rules. For example, the Oval system (Malone *et al.* 1987) employs rules to determine if a mail message is of interest, and if so, what action should be performed on it. A number of systems have employed a scripting language to allow users to specify rules. However, the overhead involved in learning and applying the scripting language may discourage non-technical users from using such a system.

The second approach involves endowing the agent with extensive domain-specific knowledge about both the application and the user. Though this approach shifts the task of programming the agent from the user to the Knowledge Engineer, the knowledge of the agent is fixed, i.e. it is difficult to customise to the user's changing preferences and habits.

A more practical approach that allows flexibility for a wide range of users is one that relies on the application of machine learning techniques. The agent is given a minimum of background knowledge, and learns appropriate behaviour from the user and perhaps other agents (Maes 1994b). The use of machine learning methods to develop a profile of user preferences allows the agent to adapt to changes in user behaviour, as well as eliminating the need for explicit programming with rules or scripts.

A common method of developing a user profile is by observing and analysing user behaviour. The profile is utilised when the agent decides what assistance it should provide, and also when determining the confidence it has in its own actions. Unless the agent can provide accurate and consistent advice, the user will lose trust in the system. The user should therefore be able to override any agent decision if necessary.

This paper describes the issues involved in the application of machine learning techniques to interface agents which manage Internet-based information. The agents described here all are based on the same basic architecture (Payne 1994) described below.

As a Graphical User Interface (GUI) is used to interact with the underlying application, observations are made of the user's behaviour. These observations, typically consisting of documents and corresponding user actions, are passed to a feature selection module, which generates training examples to be used by a learning algorithm in order to produce a user profile. Such examples typically summarise the content of a document in terms of a small number of words, selected using metrics such as frequency of occurrence. New documents are also processed by the feature selection module, and the output from this passed to the classification stage. The user profile is then employed to generate classifications for the new documents, such as

a user's interest rating in a USENET news article or a World-Wide Web page.

The agents described in this paper have all been embedded within existing, publicly available software; our aim has been to develop the software in such a way that the user experiences minimum disruption while using the agent-enhanced system. Several other interface agents have been developed in recent years to deal with Internet-based information; a selection of these systems are described in the next section.

Related Work

NewsWeeder (Lang 1995) is a news-filtering system which prioritises USENET news articles for a user using the Minimum Description Length algorithm (Rissanen 1978). The user may choose to read a newsgroup from the normal newsgroup hierarchy, or read NewsWeeder's *virtual newsgroup*. This virtual newsgroup contains a personalised list of one-line article summaries, from which the user may select a group of articles to read. NewT (News Tailor) (Sheth 1994) adopts a genetic algorithm-based approach to identify articles of interest to the user. Any number of filtering agents may exist, each agent responsible for filtering news from a specified news hierarchy. NewT filters new articles by converting them into their vector space representations (Salton & McGill 1983); and testing these against the profiles. Articles are ranked according to the closeness of the match, and the highest ranking articles are presented to the user.

WebWatcher (Armstrong *et al.* 1995) is an information search assistant for the World-Wide Web which attempts to recommend links that the user should follow. The system learns by observing the user's reaction to its advice and the eventual success or failure of the user's actions. Webhound (Lashkari) is a personalised World-Wide Web document filtering system that recommends new documents to the user based on observation. An agent window enables users to interact with their own personal Web agent. The system employs social information filtering (Maes 1994b) to recommend documents to a user by comparing materials deemed to be of interest to one user with a database of other users' preferences.

We will now describe four agents, MAGI, IAN, LAW and AARON, that have been developed to assist a user in dealing with Internet-based information.

MAGI - Mail AGent Interface

Magi aids a user in sorting incoming electronic mail (Payne 1994). By interacting with a modified version of *Xmail* the user organises their mailbox. The user's actions and the messages on which the actions were performed are stored in a *session logfile*. Features are extracted from the messages in this logfile and utilised by a machine learning algorithm to generate the user profile. Two different approaches were explored to create such profiles: rule induction and an instance-based method. The CN2 rule induction algorithm (Clark & Niblett 1989) takes a collection of training examples and induces symbolic rules which can be used to classify new, unseen examples. The IBPL instance-based algorithm (Payne & Edwards 1995) does not perform an explicit rule generation phase, rather it stores instances for use during the classification phase. The stored instances are compared with descriptions of new situations and a classification determined, based on a similarity measure between the new and old instances.

Once a profile has been generated, it is used to classify incoming mail messages, and a confidence rating is calculated for each prediction. A prediction is considered valid if its confidence rating is greater than a lower threshold value, known as the *predictive threshold*. Valid predictions are then stored by the agent for presentation to the user. When the user next uses the application, she can instruct the agent to perform its suggested actions on the messages, or can browse the predictions made by the agent. The browser displays a summary of the predicted actions and indicates those predictions with a sufficiently high confidence rating to be invoked. The user can confirm predictions with low ratings, or reject highly rated predictions, thus overriding the agent's decision.

IAN - Intelligent Assistant for News

IAN, an adaptation of UNA (Green 1995), aids a user in identifying interesting USENET news articles. An existing news browser, *xrn* was adapted in order to accomplish this (see Figure 1). As the user reads each news article, she provides a rating on a 4 point scale, in order to indicate her level of interest in the article. The interest rating is conveyed by pressing one of the rating buttons on the user interface. A rating of 1 indicates that the user found the article extremely dull or uninteresting, while a rating of 4 indicates to the agent that the user found the article highly interesting. A *Don't care* button is also provided. The feature selection module identifies fields in the articles such as the *newsgroup, subject* and *body*, and extracts values from them according to the term frequency method. This method involves removing any low entropy words, such as *the, and*, etc. and keeping a frequency count of all remaining words. The words that occur with the highest frequency are considered to be most significant, and are selected to represent the article.

As with MAGI, rule induction and instance-based

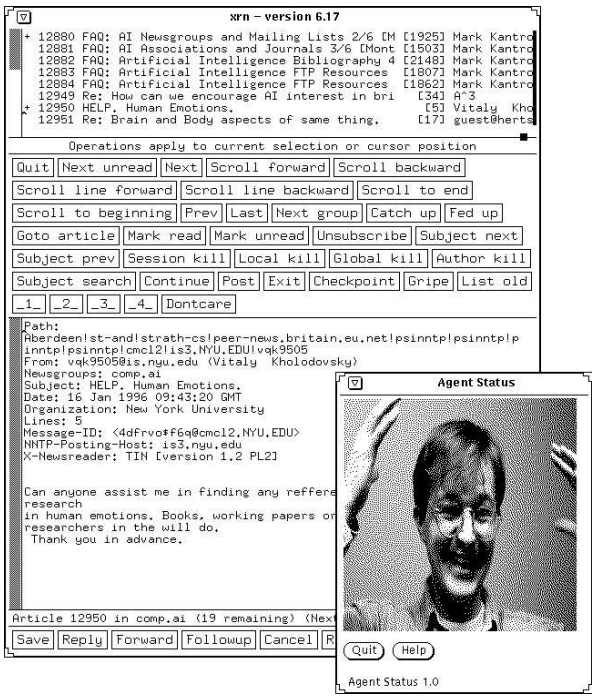

Figure 1: The IAN User Interface.

approaches for generating the user profile were compared; the rule induction algorithm in this case being C4.5 (Quinlan 1993).

Periodically (e.g. every hour) a daemon queries the news server to retrieve new articles from subscribed newsgroups. The new articles are then classified using the user profile and the results passed to the prediction stage, which generates a prediction (on the scale 1-4) if a confidence threshold is exceeded.

When the user next reads news, she can choose one of two modes: *agent* mode or *browse* mode. When in browse mode, there is no agent intervention in the presentation of articles to the user; all articles are presented, regardless of their predicted interest rating. When in agent mode, uninteresting articles (i.e. those given a rating of 1-2) are marked as having been read. Interesting articles (i.e. those given a rating of 3-4) or those for which the agent is unable to generate a prediction, are left as unread. In this way, articles believed to be of little or no interest are filtered out. An agent status window runs permanently in the background of the user's desktop (see Figure 1). This is a graphical representation of the status of the agent, indicating one of four possible states: *idle*, *learning*, *dull* or *excited*. The agent is deemed to be *idle* if no new articles have been posted since the user last read news. The agent is *learning* if a profile is being generated from user observations. The *dull* icon indicates that new articles have been posted to at least one newsgroup and that all the articles have been classified as uninteresting, whereas the *excited* icon indicates that some interesting articles have been detected.

Experimentation was carried out to compare the performance of the IBPL and C4.5 algorithms. The IAN test set consisted of 1200 news articles, split evenly across six newsgroups: *alt.lefthanders*, *alt.education.research*, *rec.food.cooking*, *rec.food.veg.cooking*, *rec.humor* and *sci.stat.math*. Each article was given an interest rating by one of the authors. For each newsgroup, a percentage of randomly selected messages were used to train the agent, and the remainder used for testing purposes to assess the classification accuracy of the learned user profile. This process was repeated for training percentages from 10% to 90% in 10% steps.

Two sets of tests were carried out: 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).

As can be seen from Figures 2 and 3, the rule induction algorithm performed better than the instance-based algorithm when predicting broad classifications, with accuracies of up to 75%. When predicting narrow classifications, however, the instance-based method outperformed the rule induction algorithm, though the results for both algorithms were disappointing, with the accuracy falling as low as 20% in some cases.

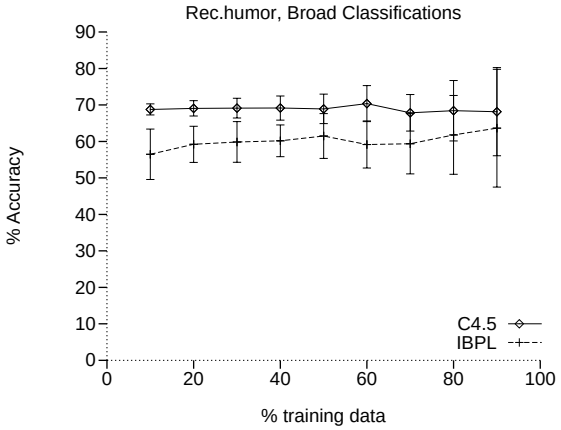

Figure 2: Comparison of the Performance of IBPL and C4.5 on Broad Classifications for IAN.

The results obtained from IAN could perhaps have been improved if an alternate feature selection method

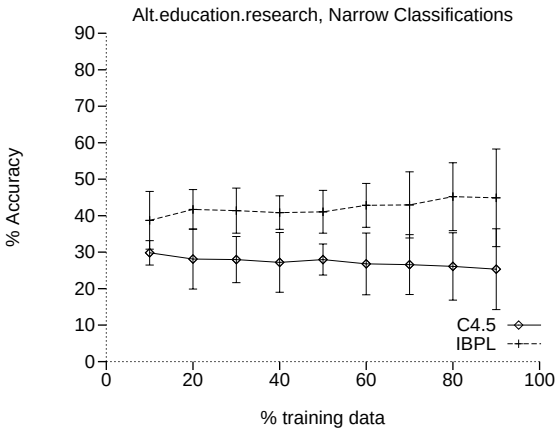

Figure 3: Comparison of the Performance of IBPL and C4.5 on Narrow Classifications for IAN.

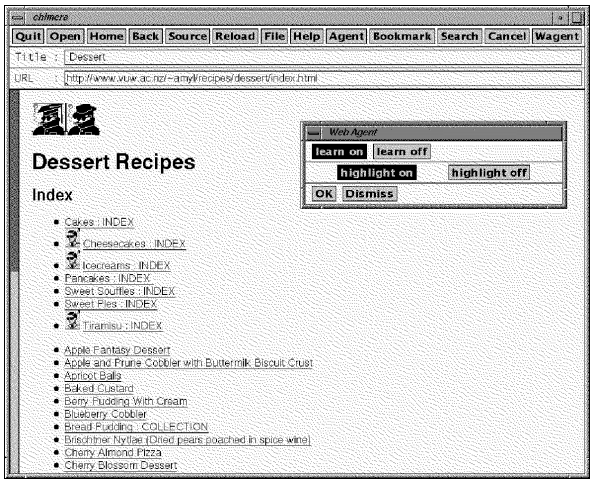

Figure 4: The LAW User Interface.

had been employed. One possible reason for the disappointing results (especially when attempting to predict narrow classifications) could be due to the fact that all USENET news articles are already classified into a subject area (e.g. *sci.stat.math*). Therefore, the most frequently occurring words in all articles (regardless of whether the user found them interesting) are similar, making it difficult for a learning algorithm to predict a rating of user interest. It is for this reason that further work on feature selection mechanisms was carried out when developing the LAW system, described in the next section.

LAW - A Learning Apprentice for the World-Wide Web

LAW (Bayer 1995) 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 attempts to find pages that might be of interest. As with IAN, an existing tool (in this case the graphical Web browser *Chimera*) was modified to incorporate the agent.

As the user browses the Web using the modified browser, data is collected about the documents viewed and any actions taken, such as whether the user saved the location of a page as a bookmark or printed a page. The user can also give direct feedback by pressing an *agent* button, which indicates that they found a page interesting. This information is used to create two profiles. The first profile represents the links which the user followed or found interesting. The second describes interesting pages. The two are referred to as the *link profile* and *page profile* respectively.

To construct the set of instances needed for the link profile, the words associated with each link in each document must be identified. Four distinct groups of words are extracted from each link: words in the link text, words in the text surrounding the link, words in the heading nearest the link, and words in the title of the document. Each link is represented as an instance made up of these four attributes and is given a classification of either interesting or uninteresting. If a link led to a page that the user saved as a bookmark, printed, or visited frequently then it is classified as interesting, otherwise it is classified as uninteresting.

The training data required for the page profile is constructed in a similar manner. An instance is created for each unique document. Four attributes are used to represent the contents of a page: words in the title of the document, words in the headings within the document, words in the links, and words in the remainder of the document. Each instance is classified as interesting or uninteresting, where interesting implies that the user gave direct feedback to the agent or that the page was visited frequently, was saved as a bookmark or printed.

Only the five most significant words are extracted from each field in the document. Low entropy words are removed, such as *the, and,* etc. and the remaining words rated using a measure of word significance. A number of different measures have been compared, including term frequency, TFIDF (term frequency versus inverse document frequency), and a measure based on term relevance (Salton & McGill 1983).

Term frequency[1] (Equation 1) is a simple measure that assumes that the importance of a word is directly

[1]This is the measure used by the MAGI and IAN systems

proportional to the frequency with which it appears within a document.

$$Weight_{ik} = \frac{Freq_{ik}}{NoWords_i} \quad (1)$$

Where $Freq_{ik}$ is the frequency of word k in document i and $NoWords_i$ is the number of words in document i.

Term frequency / inverse document frequency gives preference to words that are good discriminators between documents in a collection. The measure compares how frequently a word appears in a document against the number of other documents which contain that word. The weighting formula is as follows :

$$Weight_{ik} = Freq_{ik} \cdot [\log_2 n - \log_2 DocFreq_k + 1] \quad (2)$$

Where n is the total number of documents in the collection. $DocFreq_k$ is the number of documents in which word k appears.

Term relevance gives preference to words that differentiate the different classes of documents. The calculation gives preference to words that frequently occur in one class of documents and infrequently in the rest. The formula for this measure can be seen in Equation 3.

$$TermRelevance_{kc} = \frac{r_{kc}/(R_c - r_{kc})}{s_{kc}/(I_c - s_{kc})} \quad (3)$$

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

LAW considers two classes of documents, interesting and uninteresting, hence two values must be calculated for every word in the collection. The first value is the significance of a word in documents that are considered interesting. The second, the significance of the same word in documents that are considered uninteresting. In the training data the different classifications of the pages are known a-priori. This allows the appropriate significance values to be used. A problem arises when feature selection must be performed on new documents. Here the class of the document is not known. Consequently it is necessary to use an alternative weighting formula (e.g. term frequency) when analysing these documents.

The modified Web browser can be set in one of four modes: *inactive*, *learning*, *advising* or *learning and advising*. When the agent is in learning mode, the data needed to generate the profiles is collected by observing the user's interaction with the browser. When the agent is in advising mode, links within documents are extracted and individually classified using the link profile. The links that are classified as interesting are highlighted by inserting an icon immediately prior to the link in the document. Once all of the links have been analysed, the page is displayed to the user. An icon representing the agent mode is placed at the top of each document displayed to the user. Both the link and page profiles are used by the Web robot to decide which links to follow and which pages to present to the user. The modified Web browser can be seen in Figure 4.

The Web robot is a separate program that explores the Web using a best first search through the links encountered. The robot is given a number of starting points from which to begin its exploration. It then enters the following cycle: load a page, extract and analyse the links within the page, and assess the overall content of the page. The extracted links are classified using the link profile. Those that are classified as interesting are given a score based on the confidence returned by the classification engine in its prediction. The score given to each link is used to order the search through the links. The highest scoring links are explored first as they are most likely to lead to interesting information. The content of each page is assessed using the page profile. If it is classified as interesting it is given a score based on the confidence returned by the classification engine. The n highest scoring pages are presented to the user.

Experiments were performed on three different data sets constructed using the modified Web browser. These covered *human rights*, *sport* and *cooking*. Each set contained approximately 120 pages.

A comparison of the different feature selection techniques showed that no measure consistently produced better results. However, the different methods did sometimes outperform one another, e.g. term relevance gave better results than term frequency when used to generate a link profile from the *human rights* data set. The differing characteristics of the data sets are believed to be responsible for this effect.

The performance of the Web robot was also assessed using the three data sets. The pages suggested by the robot were analysed by first checking whether each page was in the correct domain and second, if it was of interest[2]. The results of a typical run on the *cooking* data set can be seen in Table 1. The run lasted six hours and 700 documents were loaded from 219 unique hosts. The results demonstrate that the robot

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

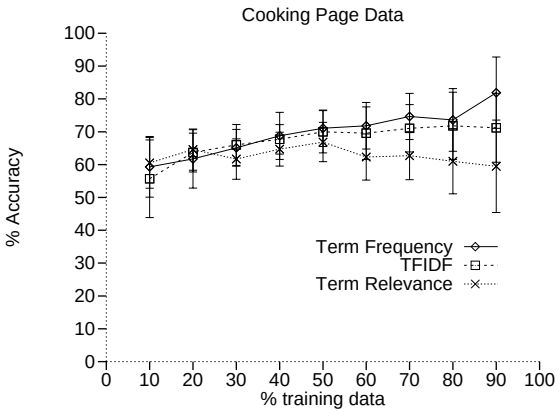

Figure 5: Comparison of Different Feature Selection Techniques for LAW.

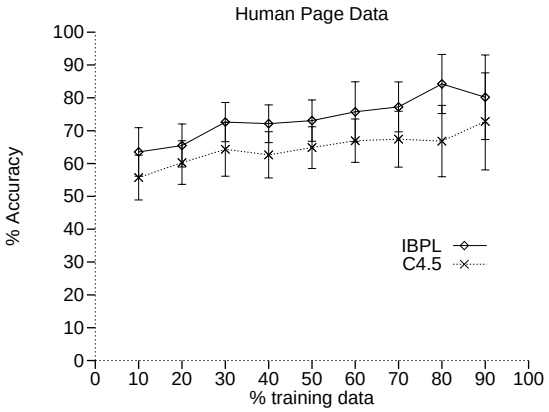

Figure 6: Comparison of Different Learning Algorithms for LAW.

was able to discover a considerable number of relevant and interesting pages. However, the accuracy of the robot's suggestions dropped rapidly as the confidence in its predictions decreased.

AARON

The AARON system (Mackenzie 1996) is a specialisation of the LAW approach described above. AARON (Automated Assistant for Retrieval of On-line Newspapers) gathers HTML documents from a number of pre-specified sites on the World-Wide Web which provide news related material. These documents are clustered into related groups, and a user profile used to identify interesting groups. A personalised newspaper is then delivered as a series of local HTML pages. The main components of the AARON system are shown in Figure 7.

A simple Web robot is employed to search sites con-

No. Links Suggested	Confidence	% in Correct Domain	% Interesting
10	1.97	90	60
30	0.79	63	40
50	0.71	52	36
100	0.45	41	21

Table 1: Results for a Single Test Run of the LAW Web Robot on the *cooking* Data Set.

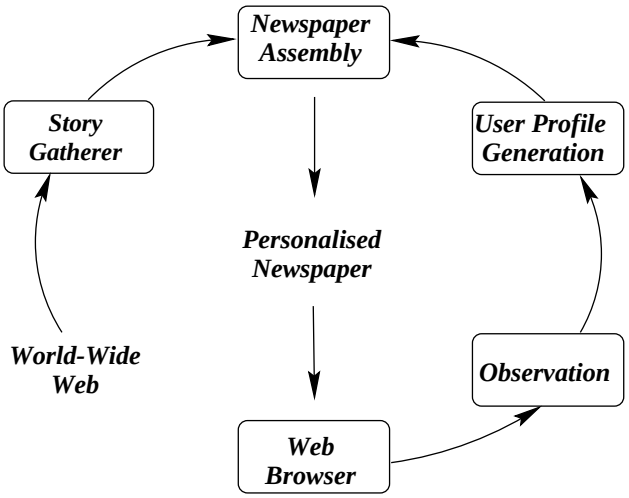

Figure 7: The AARON Architecture.

taining news material such as *http://www.telegraph.co.uk*, *http://www.asahi.com/english/english.html*, etc. Details of sites to be explored are specified by the user when configuring AARON. An exhaustive search is performed to recover all HTML documents containing textual news summaries and photographic material. A number of constraints are placed on the robot's behaviour, e.g. links leading away from news sites are not followed. Once material has been gathered by the robot, feature selection is performed on the HTML documents (using methods analogous to those used by LAW) to generate feature sets for each document. These are passed to a Bayesian clustering algorithm, AutoClass (Cheeseman & Stutz 1996) which identifies similarities between documents from different sources and groups them into clusters (typically 5-20 documents per cluster). Associated with each document is a probability score which indicates how representative the document is of the cluster to which it has been assigned.

Once documents have been clustered into groups (we will use the term *story* to refer to such groupings) the material must be processed to remove uninteresting

stories, before being organised for presentation in a newspaper-like format. The personal newspaper consists of a number of HTML pages organised hierarchically under three broad section headings: *headlines*, *interest*, *gazette*. AARON begins by evaluating each *story* cluster to determine whether it should be regarded as a headline. This is done by employing a simple heuristic which labels clusters which contain a large number of lengthy documents as headline topics. The system then considers all the remaining clusters and estimates the level of user interest in each, using the instance-based learning algorithm IBPL (Payne & Edwards 1995); each cluster is compared against a number of pre-stored instances of previously seen, interesting news stories. These instances are gathered by observing the user reading the newspaper, once it is constructed. The observation and user profile generation steps are analogous to those in the LAW system. Interesting stories are presented in the *interest* section. A small, random sample of the remaining stories (those that are neither headlines or of direct interest) are used to prepare the *gazette* section of the paper.

At present, the AARON architecture described here exists as a simple prototype. We are currently performing an extensive series of tests on the system to evaluate the effectiveness of components such as the clustering algorithm, headline identifier, and instance-based learner.

Conclusions

This paper has summarised recent work at Aberdeen on the development of agent-enhanced tools for managing Internet-based information. We have developed a variety of applications, and in the process have explored the performance of different machine learning techniques and/or feature selection methods. Results to date do not support any definitive conclusions regarding the relative merits of the techniques investigated, other than the observation that instance-based methods overall perform considerably faster than rule-induction approaches. It may be concluded from our results that more investigation of feature selection methods is needed. Since documents can be many tens of thousands of words in length, taking the n most frequently occurring words may not be the most effective way of extracting the salient features. One possible future direction may be to use text summarisation techniques to extract the key information from an article, utilising this to train the learning algorithm.

However, our work has demonstrated that it is possible to integrate agents which learn user profiles into existing Internet software. This integration can be achieved in such a way that the user is not impeded in their use of the tool and is not forced to change their manner of working. Once generated, interest profiles can be used in a variety of ways, e.g. to filter incoming information, to highlight relevant information, to guide searches, or to determine how information is presented to the user.

AARON goes beyond existing software agents for Web browsing/access by employing learning techniques not only to learn a profile of user interest (as in LAW), but also to manage the presentation of information to the user, in the form of collections of topic-related documents (*stories*). We are currently investigating techniques which integrate information retrieval and machine learning methods, in an attempt to improve the classification accuracy of learned user profiles.

Acknowledgements

Figure 1 from Payne, Edwards & Green, *IEEE Transactions on Knowledge and Data Engineering (to appear)*. Reproduced with permission. All rights reserved.

We gratefully acknowledge the authors and maintainers of *Xmail*, *xrn* and *Chimera* for allowing us to use their software. We thank Nick Murray for allowing us to run our CPU intensive tests on the network at the most inappropriate times and David Dyke for providing a number of amusing images, bringing IAN to life.

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.

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

Cheeseman, P., and Stutz, J. 1996. Bayesian Classification (AutoClass): Theory & Results. In *Advances in Knowledge Discovery and Data Mining*. AAAI/MIT Press, Menlo Park.

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

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

Lang, K. 1995. NewsWeeder: Learning to Filter Netnews. In *Proceedings of the 12th International Ma-*

chine Learning Conference (ML95), 331–339. San Francisco, CA:Morgan Kaufmann.

Lashkari, Y. The Webhound Personalized Document Filtering System. http://webhound.www.media.mit.edu/projects/webhound.

Mackenzie, R. 1996. An Automated Assistant for the Retrieval of On-line Newspapers. BSc Final Year Project Report, Department of Computing Science, University of Aberdeen, Scotland.

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

Maes, P. 1994b. Social Interface Agents: Acquiring Competence by Learning from Users and Other Agents. In *Software Agents: Papers from the 1994 AAAI Spring Symposium*, 71–78.

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

Payne, T., and Edwards, P. 1995. Interface Agents that Learn: An Investigation of Learning Issues in a Mail Agent Interface. Applied Artificial Intelligence, in press.

Payne, T. 1994. Learning Email Filtering Rules with Magi, A Mail Agent Interface. MSc Thesis, Department of Computing Science, University of Aberdeen, Scotland.

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

Rissanen, J. 1978. Modeling by Shortest Data Description. *Automatica* 14.

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.