Exploiting Learning Technologies for World Wide Web Agents

P Edwards, C L Green, P C Lockier & T C Lukins

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

Abstract

This paper illustrates how machine learning techniques can be utilised within intelligent software agents which assist users with the management of Web-based information. We discuss a number of recent activities within the *Learning Agents & Systems*[1] group at Aberdeen, namely: a generic Web browsing assistant; an agent which constructs personalised travel brochures; the application of clustering techniques to Web browser histories; and multi-agent support for adaptive user querying.

1 Introduction

The recent, rapid growth of the World Wide Web 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 one 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" [1].

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 [2] employs rules to determine if a mail message is of interest, and to decide upon the relevant action to perform. A number of systems have employed scripting languages to allow users to specify rules. However, the overhead involved in learning and applying such a language may discourage non-technical users from using these systems. 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 offers 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 [3]. 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.

This paper illustrates a number of ways in which machine learning techniques can be applied within intelligent software agents. We began our work in this area by developing a basic learning agent architecture [4]. As a user

[1] http://www.csd.abdn.ac.uk/~pedwards/res/las.html

interacts with an underlying software application, observations are made of his/her behaviour; such observations typically consist of documents (e.g. electronic mail messages, Web pages) and corresponding user actions (e.g. a numerical interest rating for a Web page). These observations are passed to a term selection module, which generates training examples to be used by a learning algorithm in order to produce a user-profile. Training examples summarise the content of a document as a collection of terms, selected using metrics such as frequency of occurrence. We have explored a variety of different learning approaches to create user-profiles, including the C4.5 [6] rule induction algorithm, and the IBPL [4] instance-based algorithm. Rule induction algorithms take a collection of training examples and induce symbolic rules which can be used to predict actions for new, unseen document instances. IBPL does not perform an explicit rule generation phase, rather it stores instances for use during the prediction phase. The stored instances are compared with descriptions of new situations and a prediction determined, based on a similarity measure between the new and old instances.

2 Related Work

Recent years have seen a rapid expansion in research into intelligent agents for information management; in this section we highlight a number of approaches which have employed machine learning techniques.

NewsWeeder [7] is a news-filtering system which prioritises USENET news articles for a user using the Minimum Description Length algorithm [8]. 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) [9] adopts a genetic algorithm-based approach to identify articles of interest to the user. A population of filtering agents exists, where each agent is responsible for filtering news from a specified news hierarchy. New articles are first converted into a vector space representation [10] before being tested against an interest profile. Articles are ranked according to the closeness of the match, and the highest ranking articles are presented to the user.

WebWatcher [11] is an information search assistant for the World Wide Web which attempts to recommend links that a 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 [12] 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 [3] 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 Web agents: LAW, ELVIS, CUSTARD and MAVA.

3 LAW - A Learning Apprentice for the World Wide Web

LAW [13] 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. An existing graphical Web browser *Chimera* was modified to incorporate the agent, which is itself based on the generic learning agent architecture discussed above.

As the user browses the Web using the modified browser (Figure 1), 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

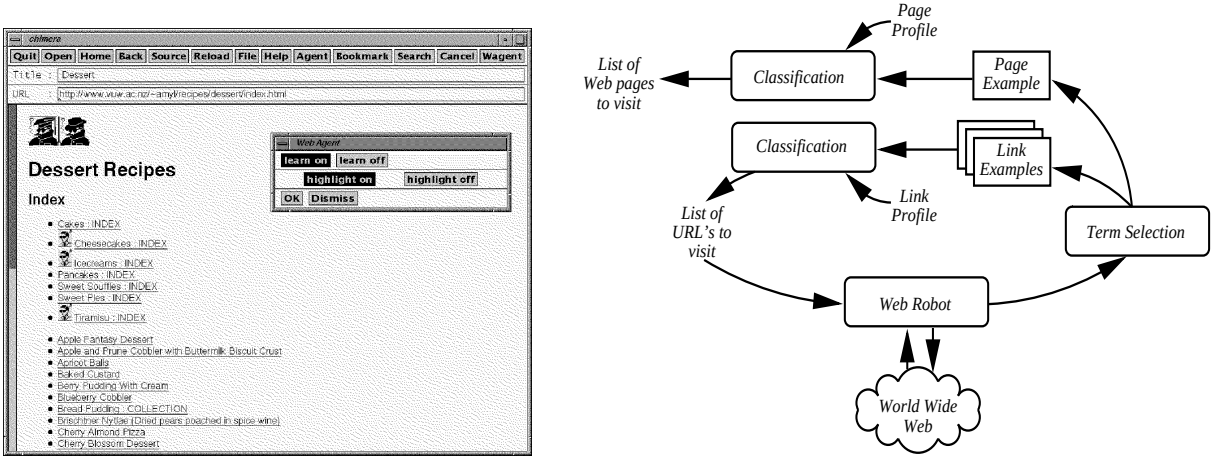

Figure 1: LAW - modified Web browser (left) & Web robot architecture (right).

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 terms associated with each link in each document must be identified. Four distinct groups of terms are extracted from each link: those in the link text, the text surrounding the link, the heading nearest the link, and 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, containing terms from the title, headings, links, and free text 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 terms are extracted from each field in the document. Low entropy terms are removed, such as *the, and,* etc. and the remaining terms rated using a measure of term 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 [10].

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.

The Web robot is a separate program that explores the Web using a best first search through the links encountered (Figure 1). 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 also given a score based on LAW's confidence in its prediction. The scores are used to order the search through the links; the highest scoring links being explored first as they are most likely to lead to interesting information. The content of each page is assessed using the page profile, and if classified as interesting is assigned a score. The n

highest scoring pages are presented to the user.

A number of experiments have been performed to evaluate the effectiveness of LAW. Space does not permit us to present results here; see [13] for full details.

4 ELVIS - Enhanced Learning Visitor Information System

The ELVIS system is a specialisation of the LAW approach described above. ELVIS gathers HTML documents from a number of sites on the World Wide Web which provide travel information, such as details of hotels, restaurants, museums, galleries, shopping, etc. A personalised brochure is delivered as an HTML page organised under six broad section headings: *accommodation*, *sight seeing*, *food & drink*, *entertainment*, *something active*, and *shopping*; an additional heading *miscellaneous* is used for other relevant material. The entire system is implemented using Java.

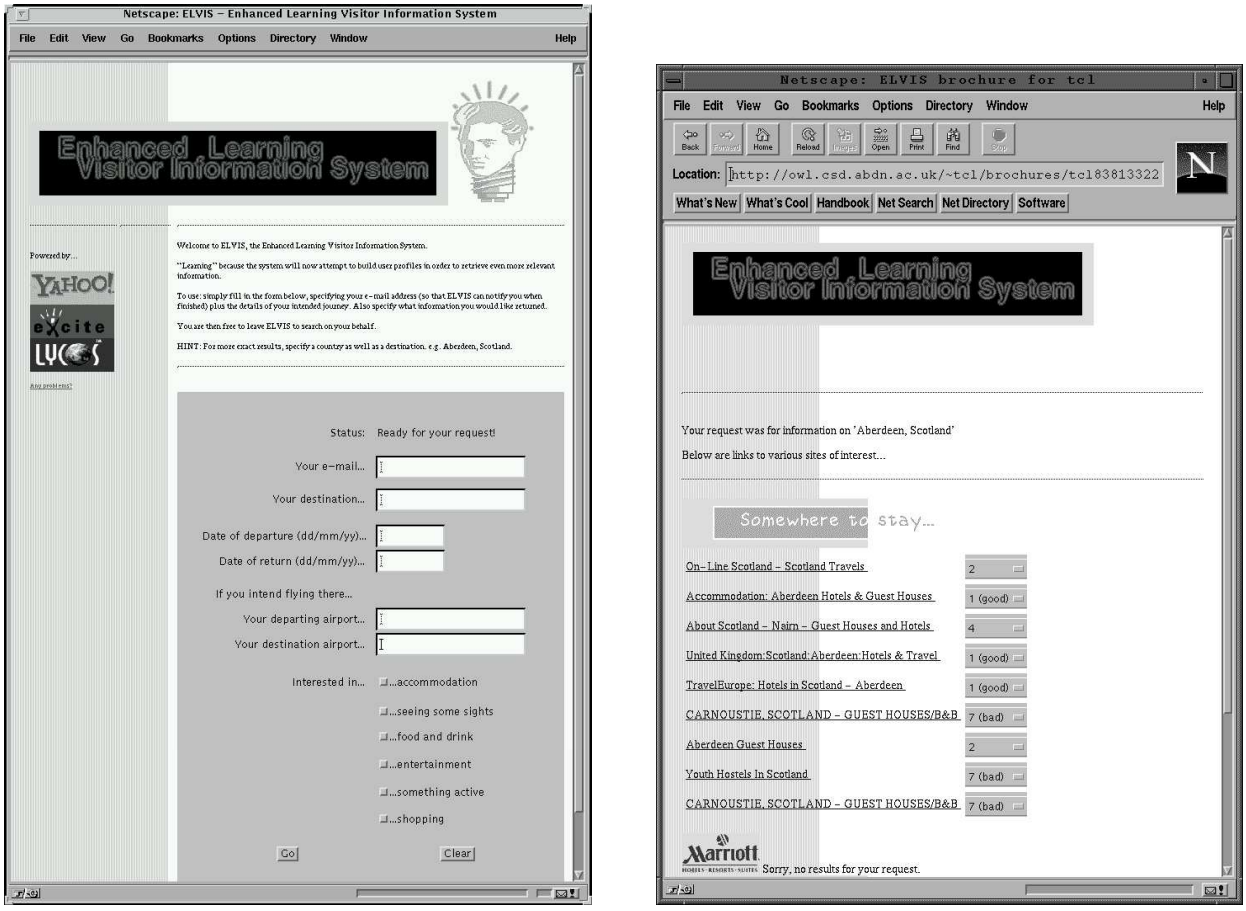

Figure 2: ELVIS - client interface (left) & sample brochure, showing user feedback (right).

A user interacts with ELVIS via a simple client applet embedded within a Web page (see Figure 2); the applet allows a travel destination and other information such as the user's electronic mail address, dates, destination airport, etc. to be specified. This information is communicated to the ELVIS server, which acknowledges its receipt. The user is then free to exit their Web browser, as the request has been entered in a queue maintained by the server. Once at the head of the queue, the destination information is combined with various travel-related keywords (e.g. *hotel*, *museum*, *gallery*) and used to form a series of requests to existing Web search engines[2].

[2] At present, ELVIS uses the Yahoo, Excite, and Lycos search engines.

These requests are handled in parallel by ELVIS and in each case, the top 10 hits from each search engine are gathered; duplicate results are removed and the resulting list of URLs passed to the relevance assessment module.

Page relevance is determined as follows: first ELVIS checks whether the page exists, then the title, headings, and free text are extracted. The resulting lists are then processed to remove low entropy terms, before being passed through an implementation of Porter's stemming algorithm [14]. A numerical assessment of relevance is computed by considering the three most frequent terms in each of the lists, and matching these against two profiles; the first profile contains general travel-related terms (such as *hotel*, *fiesta*, *festival*, *museum*, etc.) while the second contains user-specific terms (obtained via user feedback from previous brochures). Terms occurring in the generic profile are all given a numeric weighting of 1.0, while terms in the user-profile are weighted according to past feedback. The relevance assessment is computed from the scores of the matching terms, multiplied by the appropriate section weight, i.e. title (1.0), headings (0.8), free text (0.6). If the resulting value exceeds a threshold (adjusted, if necessary, to take the absence of headings, free text, etc. into account), then the page is deemed to be relevant, and is included in the brochure constructed by ELVIS.

In addition to accessing the pages identified via the search engine queries, ELVIS also contacts a number of hotel and airline reservation/information sites, e.g. http://www.marriott.com. These sites are used to provide additional information on hotel availability, flight timings, etc. Once brochure assembly is complete, ELVIS sends a message (using the mail address supplied with the original request) to advise the user of the brochure's location[3]. Part of a sample brochure appears in Figure 2.

The user is able to provide relevance feedback to the agent, by using a rating applet appearing to the right of each brochure entry; numerical ratings on a scale of 1 (good) to 7 (bad) are communicated directly to the ELVIS server by this applet. This information is used, together with the terms summarising the content of the rated document, to add new term/weight pairs to the user-profile or to refine weights associated with existing profile terms.

5 CUSTARD - Clustering User Browser Histories

Utilities such as *browser-history*[4] allow users to maintain a log (in HTML format) of every Web page they have visited; unfortunately, such logs quickly become large and difficult to navigate. We are currently exploring the use of unsupervised learning (specifically conceptual clustering [15]) in order to present such information in a way that is more meaningful to a user. This involves grouping similar documents together, and presenting the user with clusters of documents, rather than a flat list. The user can choose to view as many groups of documents as they wish, using the conceptual descriptions of the clusters as a navigation aid. These conceptual descriptions can also be treated as a concise *profile* of the user's information preferences. Such a profile could be utilised to locate documents similar to those the user has previously found to be of interest.

CUSTARD (Clustering USer Text And Refining Document descriptions) employs a basic clustering algorithm, capable of processing over 1000 HTML documents. Instances are generated by visiting each URL and extracting the most frequently occurring terms to represent the document. Once these instances have been clustered, a local Web page is produced, containing groups of original document titles and associated hyperlinks. We are currently investigating techniques which will allow the user to provide feedback to the agent so that, by refining its knowledge of the user, the agent may alter its clustering strategy to suit the user's view of the information space.

[3] All brochures reside on the same host machine as the ELVIS server.

[4] http://zenon.inria.fr/koala/colas/browser-history/

6 MAVA - Adapting Visitor Information Queries

Users wishing to interact with Web-based information resources are often provided with a form-based query interface. We are investigating how such interfaces can be augmented with intelligent agents which reason about the likely success of a user query, and if necessary, adapt the query (through the use of background knowledge) to increase its chance of success; success being measured in terms of the number of hits obtained.

MAVA (Multi-Agent Visitor Advising) employs a number of such agents to mediate access to a collection of databases containing information on hotels, restaurants and pubs in the Aberdeen area. The MAVA architecture is shown in Figure 3. The *Bellboy*, *Maitre d'* and *Landlord* agents each monitor queries submitted via a Web-based interface to their respective databases, and if the number of hits generated by a query is below a threshold value, attempt to adapt the query. Each agent employs its own specialist knowledge base which contains details of the acceptable modifications to user queries. For example, *Bellboy* is able to relax the requirement that a hotel room have a four-poster bed (if this is resulting in no matches within the hotel database), but is unable to alter a request for a double room into a request for a single.

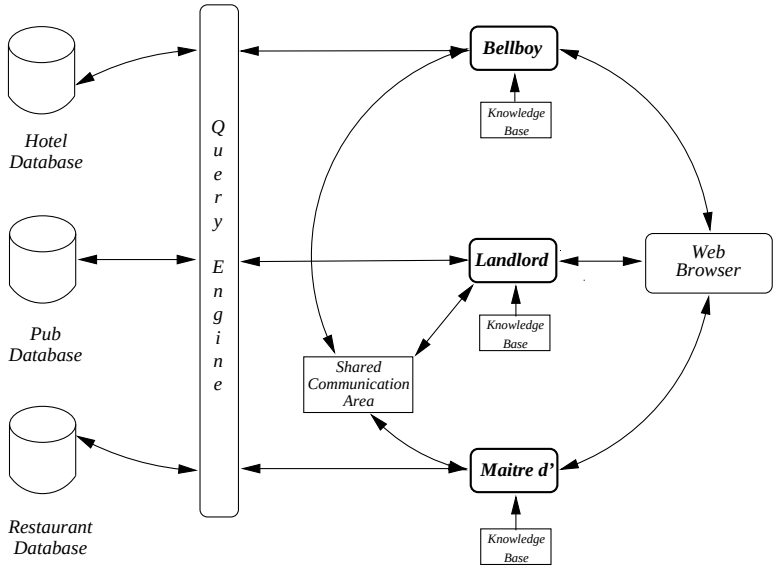

Figure 3: The MAVA architecture.

In addition to adapting user queries, each of the MAVA agents is also able to observe user activity and share meta-level information with the other agents. For example, if the *Bellboy* observes that a certain user is searching for expensive hotels in a particular geographical area (say, near the airport) it can communicate this to the *Maitre d'* and *Landlord* agents. Should the user subsequently query either of the other databases during the same session, these agents can make use of the information as additional knowledge to guide the query adaptation process.

7 Conclusions

This paper has summarised recent work at Aberdeen on the development of agent-enhanced tools for managing Web-based information. We have developed a variety of applications, and in the process have explored the performance of different machine learning techniques and/or term selection methods.

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. We are currently investigating techniques which integrate information retrieval and machine learning methods, in an attempt to improve the classification accuracy of learned profiles.

Acknowledgements

We gratefully acknowledge the authors and maintainers of *Chimera* for allowing us to use their software. We thank Nick Murray for invaluable LaTeX 2ε support; Terry Payne provided support for the LAW evaluation.

References

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

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

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

[4] T.R. Payne and P. Edwards. Interface Agents that Learn: An Investigation of Learning Issues in a Mail Agent Interface. *Applied Artificial Intelligence*, 11(1):1–32, 1997.

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

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

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

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

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

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

[11] R. Armstrong, D. Freitag, et al. WebWatcher: A Learning Apprentice for the World Wide Web. In *AAAI Spring Symposium on Information Gathering from Distributed, Heterogeneous Environments*. Menlo Park, CA:AAAI, 1995.

[12] Y. Lashkari. The Webhound Personalized Document Filtering System. http://rg.media.mit.edu/projects/webhound.

[13] P. Edwards, D. Bayer, C.L. Green, and T.R. Payne. Experience with Learning Agents which Manage Internet-Based Information. In *AAAI Spring Symposium on Machine Learning in Information Access*, pages 31–40. Menlo Park, CA:AAAI, 1996.

[14] M. F. Porter. An Algorithm for Suffix Stripping. *Program: automated library and information systems*, 14(1):130–137, 1980.

[15] D. Fisher and P. Langley. Approaches to Conceptual Clustering. In *Proceedings of the 9th International Joint Conference on AI*, pages 691–697, 1985.