Experience with Rule Induction and k-Nearest Neighbour Methods for Interface Agents that Learn

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

1 Introduction

In recent years, interface agents have been developed to assist users with various tasks such as arranging diary appointments (Mitchell, Caruana, Freitag, McDermott & Zabowski, 1994; Kozierok & Maes, 1993) and filtering incoming information (Sheth, 1994; Payne, 1994), as well as many other applications. If the agent is to be of assistance, it requires knowledge about the domain application and/or the user. Many agents employ learning techniques to acquire this knowledge. They may learn autonomously by observing the user (such as an apprentice system) or be explicitly trained.

Traditionally, two approaches have been used to provide an agent with knowledge about a task domain. The first and most common approach is for users to provide their own rules. This is used in systems such as *Oval* (Malone, Grant, Turbak, Brobst & Cohen, 1987) which determines if a mail message is of interest, and if so, what actions should be performed. Systems such as this utilise a scripting language to specify rules. However, learning and applying this scripting language may discourage non-technical users from using the system. As well as understanding exactly how they require the system to behave, a user must appreciate how the agent will perform with the rules. The user also has to be responsible for maintaining the rules over time as their interests change.

The second approach makes use of traditional knowledge engineering techniques to identify background knowledge about the application and the user (such as the UNIX consultant UCEgo (Chin, 1991)). 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 the approach cannot be used for personalised tasks such as information filtering.

The use of learning techniques to develop a *profile* of an individual user's preferences not only eliminates the need for programming rules, but allows the agent to adapt to changes. There are many ways a system can learn from the user (Gil, 1994). Approaches such as programming by demonstration provide good training examples. In comparison, apprentice systems acquire knowledge by observing and analysing the user's behaviour.

The profile should be used by the agent to decide what assistance to provide and to determine some measure of its reliability. This can be achieved by generating a *confidence rating*, which provides an indication of the agent's confidence in its predictions. Unless the advice generated is accurate, the user will fail to trust the agent. Thus the user should be able to override agent decisions if necessary. This feedback can be used to refine the profile, identifying aspects which result in poor behaviour and promoting those which improve it.

The work described here details an interface agent architecture which learns from observations, and describes how it has been applied to two different information filtering domains; that of classifying incoming mail messages (Magi) and identifying interesting USENET news articles (UNA).

2 Learning and Information Filtering

A number of applications have been developed which employ machine learning techniques to assist a user with filtering USENET news articles and email messages. Lang (1995) has developed a news-filtering system (NewsWeeder) which learns a user profile by allowing the user to rate their interest in each article on a scale of 1-5. The system currently focuses on using the content of the article to determine its relevance, which is known as *content-based filtering*. As a user reads each article, they provide a rating. Each night,

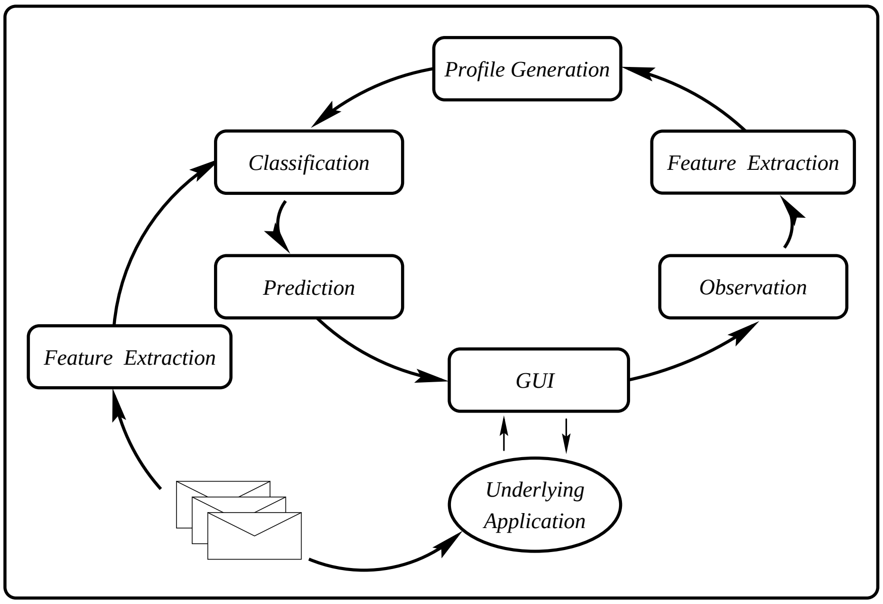

Figure 1: A Learning Interface Agent Architecture.

the system generates a new profile based on these ratings. As well as reading different newsgroups, a user can also read NewsWeeder's *virtual newsgroups*. Such a newsgroup contains a personalised, sorted list of the top rated articles determined by the agent.

So far, two different strategies have been explored to identify articles of interest. A popular technique used in Information Retrieval called *term-frequency/inverse-document frequency* weighting (TF-IDF) (Salton & McGill, 1983) has been used as a benchmark from which to compare another technique, the *Minimum Description Length* principal (MDL), described in (Lang, 1995). Both strategies rely on breaking up the article body into *tokens* and counting their occurrence to create a vector of token counts.

NewT (News Tailor) (Sheth, 1994) adopts a genetic algorithm based approach to identify articles of interest to the user. A set of profiles are applied to new articles to identify those the user would find interesting. Each profile has a rating, which measures how effective it is in identifying such articles. Genetic processes, such as *crossover* and *mutation* are used to create new profiles which may outperform current profiles, or explore new areas of interest. Profiles with a low rating are eliminated, ensuring that the number of profiles is kept to a manageable level.

NewT periodically filters new articles to determine which ones could be of interest to the user. This is done by converting articles into their vector space representation (Salton & McGill, 1983). Each article is tested against the profiles, and ranked according to the closeness of the match. The highest ranking articles are then presented to the user. The user provides positive or negative feedback as the articles are browsed, which is then reflected by changes in the rating of a profile.

A different approach has been used in the development of the mail filtering agent, Maxims (Metral, 1993). An earlier calendar management system (Kozierok & Maes, 1993) was adapted to produce a generic agent architecture which could be attached to any application. Maxims learns to prioritize, delete, forward, sort and archive mail messages on behalf of the user. The agent uses the sender and recipient fields of a message (including `cc:` recipients), and keywords from the subject field. Other information such as whether the message has been read, whether it is a reply to a previous message, etc. is also used.

As the user reads their mail and performs certain actions, the situation is memorised. When new messages arrive, the agent uses *Memory-Based Reasoning* (Stanfill & Waltz, 1986) to find the closest memorised situation, and make a recommendation to the user. Confidence ratings are generated and compared to two threshold values: a *tell-me* threshold and a *do-it* threshold. These are used by the agent to determine whether to simply recommend actions or whether to act on behalf of the user.

A caricature, in the form of a face, is used to indicate the agent's confidence in its recommendations. Different expressions are used, so the user knows if an action has been performed on their behalf, or if the agent wants to advise them. Depending on whether the user confirms or rejects the recommendation, the agent will question the user to determine what factors were important in this response. This feedback is used to adjust priority weights which affect future recommendations.

Other agents have been developed which employ machine learning techniques to assist users exploring the World-Wide Web, such as WebWatcher (Armstrong, Freitag, Joachims & Mitchell, 1995).

3 Agent Model

Our agent architecture is shown in Figure 1. A Graphical User Interface (GUI) is used by the user to interact with the underlying application. As it is used, observations are made from which the agent can induce a user profile. These observations, consisting of articles and actions performed on them, are used to generate training examples, by passing them to the feature extraction module. The training examples are then used by a learning algorithm to induce a user profile. New articles are also processed by the feature extraction module and output passed to the classification stage. The user profile is used to generate a classification such as a mail processing action (Magi), or an interest rating (UNA). The resulting classifications are evaluated by the prediction stage, and a prediction is made.

The feature extraction module identifies fields in the articles such as the *author* or the *subject*, and extracts values from them. Words are also extracted from the article body based on how frequently they occur within the text. Other information such as the length of the article can also be determined. Within an article, each field can generate a number of values. The exact format of the examples used to generate the user profile, and subsequently to make predictions on new articles depends on the nature of the learning algorithm used.

Two different learning paradigms have been explored within this architecture: a rule induction algorithm, CN2 (Clark and Niblett, 1989) and a k-nearest neighbour algorithm (k-NN) *Memory-Based Reasoning* (Stanfill & Waltz, 1986).

With CN2, many examples are generated for each article, as the algorithm expects single values for each attribute (see Figure 2). New articles are also converted into multiple examples for the classification stage. This provides a means of producing a confidence rating for each prediction made by the agent. The examples may fire different rules, leading to different classifications. The number of rules which fire for each classification are therefore summed, and a confidence rating is generated.

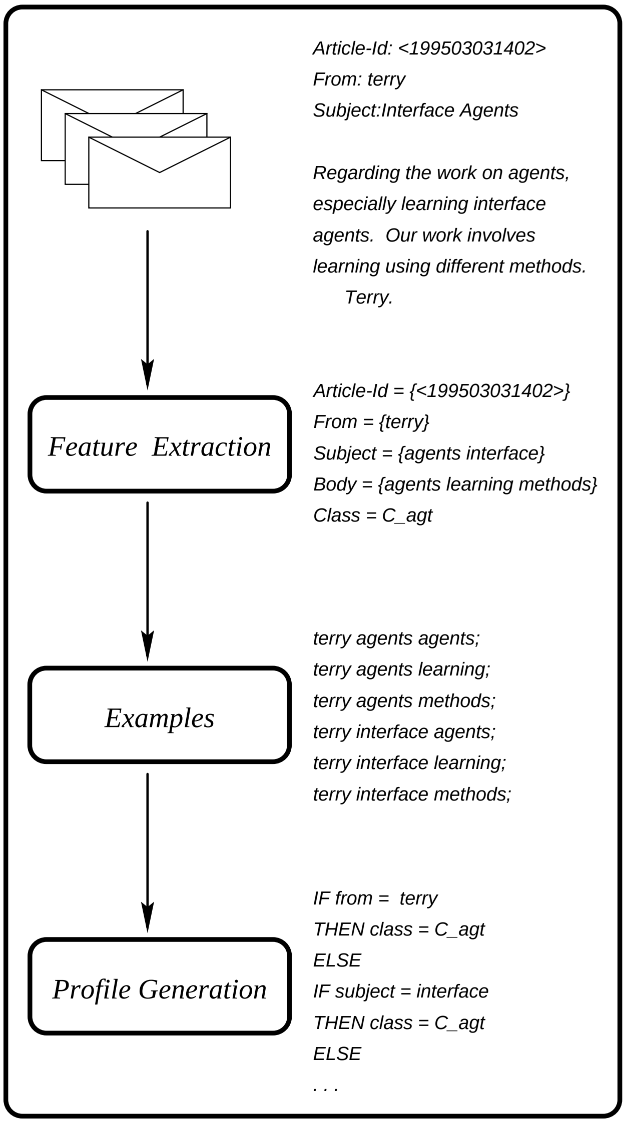

Figure 2: Generating a Profile with CN2.

The version of MBR used differs slightly from the original algorithm (Stanfill & Waltz, 1986). Instead of expecting a single value for each attribute, the modified algorithm stores sets of values (Payne & Edwards, 1995). Each article generates a single training episode, where each attribute contains a set of all the extracted values from the relevant field. MBR generates a confidence rating as part of its classification process.

As new observations are made and new training examples created, the examples are time stamped. This way the training set can be pruned with respect to time. Not only does this help reduce the number of training examples used, but also removes old examples which may become out of date as the user's behaviour changes.

The two learning approaches differ in the way new training examples are integrated into the user profile. CN2 periodically (e.g. every night) induces a new rule-set, pruning out old examples and adding new examples. Hence new examples do not affect performance until new rules are induced. MBR, however, introduces new examples into the training set as soon as they are created, and thus the effects are immediate.

We will now describe two agents, Magi and UNA which are based on this agent architecture. The use of both learning algorithms has been explored with these systems, and is described later.

4 Magi

Magi aids a user in sorting incoming electronic mail (Payne, 1994; Payne & Edwards, 1995). In essence, the system is an apprentice which autonomously observes and analyses user behaviour in dealing with mail. By interacting with a modified version of *Xmail*, a user can send and read mail messages, and organise their mail box. For each session, a *session logfile* is created which contains the user's actions and the messages on which the actions were performed.

Periodically, features are extracted from the messages in this logfile; it is these which are used to generate the user profile. Features extracted from incoming mail messages are tested by the classification engine, and a confidence rating is generated. The 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.

The user is informed if new messages have arrived when they next use the application. At this point the user can instruct the agent to perform its suggested actions on the messages, or can browse the predictions. For each type of action that can be predicted, there exists a *confidence threshold*. Only actions with ratings greater than this will be invoked. The rationale behind this is that certain predicted actions, if incorrect, can be tolerated, such as storing a message in the wrong mailbox. However, actions such as those that delete mail or forward messages to other recipients are more critical. Thus the agent requires a higher level of certainty in such predictions before performing them. This second threshold value is therefore used to determine which predictions require user confirmation before being performed.

A prediction browser allows the user to monitor actions suggested by the agent, and thus establish trust in the agent's predictive ability. 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 browse the predictions, and either confirm predictions with low ratings, or reject highly rated predictions, thus overriding the agent's decision. This feedback could be used to adjust the confidence threshold for each class of action, such as deletion, etc.[1]

5 UNA

UNA aids a user by identifying interesting USENET news articles (Green, 1995). As the user reads each news article in a modified version of the *xrn* browser, they provide a rating in order to indicate their level of interest in the article. The interest rating is an integer in the range 1-6, conveyed by pressing one of 6 buttons on the user interface (see Figure 3). A rating of 1 indicates that the user found the article extremely dull or uninteresting, whilst a rating of 6 indicates to the agent that the user found the article highly interesting. Article details and ratings are appended to a *session logfile*. When the user exits the user interface, features are extracted from these observations, and are used to generate the user profile. The user profile is utilised by the classification engine in order to classify future news articles.

Periodically (e.g. every hour) a daemon runs, which identifies the newsgroups the user is subscribed to, and queries the news server to retrieve all new articles posted to each subscribed newsgroup. Features are extracted from these new articles in the same way as for the training data. The articles are then classified and the results passed to the prediction stage, which interprets the results of classification, generating a prediction (on the scale 1-6) of the user's interest in the article.

When the user next reads news, they 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 whether the agent has judged them to be of interest or not. When in agent mode,

[1]The use of feedback to adjust the different confidence thresholds for each type of action has yet to be implemented.

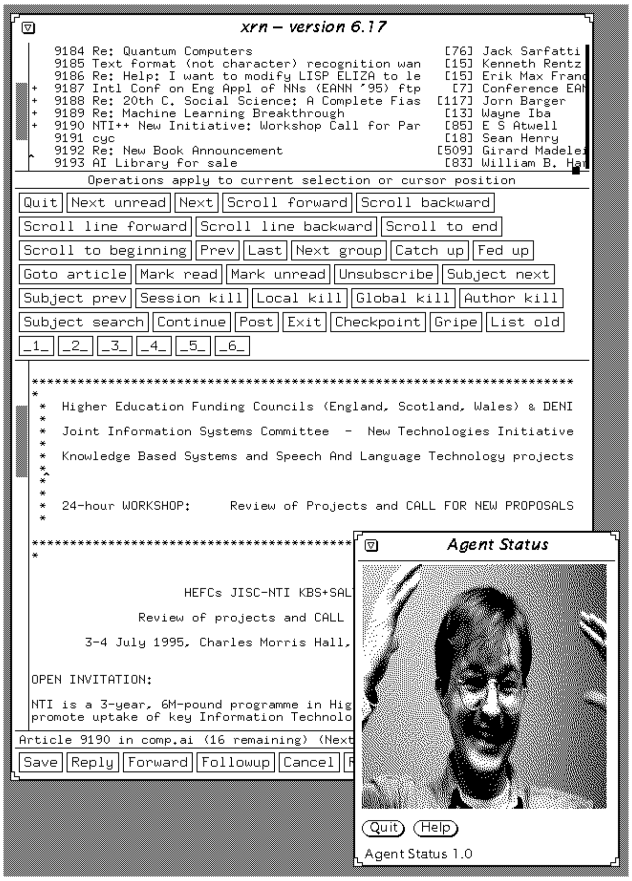

Figure 3: The UNA User Interface.

however, the agent marks any new articles that it has predicted as uninteresting (i.e. given a rating 1-3) as having already been read. All articles which have been predicted as interesting (i.e. given a rating 4-6) or those for which the agent was unable to generate a prediction, are left as unread. With this method, 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 3). This is a graphical representation of the status of the agent. The user, by simply glancing at the agent status window, can see if any new articles have been found by the daemon since the last news reading session, and if so, whether any of these articles have been classified as interesting. The agent status window can represent four states of the agent; *idle*, *learning*, *dull* or *excited*. The agent is deemed to be *idle* if the daemon is not running, and 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 the articles have been classified as uninteresting, whereas when the *excited* icon appears, some interesting articles have been detected.

6 Experimentation and Evaluation

Experimentation has been performed to compare the performance of both learning algorithms in making accurate predictions for new articles. The Magi test set consisted of 408 mail messages, sorted into 12 classifications, whereas the UNA set contained 1200 articles split evenly across 6 newsgroups. UNA was tested by rating messages as either interesting or dull, and by providing an integer rating between 1 (dull) and 6 (interesting).

Each Magi classification determined in which mailbox a message should be placed. Mailboxes such as *agents* and *cure* contained messages from a mailing list. The three mailboxes *dai, kdd* and *mead* all contained messages from a mailing list digest. The messages in these digest mailboxes contain individual messages which have been grouped together by a moderator (either manually or automatically). Thus the message body features selected represent the digest, as opposed to any individual topic within the digest. This data is described fully in (Payne & Edwards, 1995).

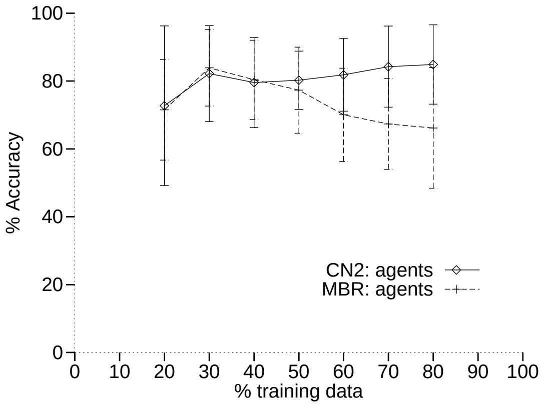

Figure 4: Accuracy of Predictions made for the *agents* Mailbox.

6.1 Magi Experimentation

The average accuracy of predictions over all mailboxes was higher for CN2 (65%) than for MBR (57%). Figures 4 and 5 show examples of the performance for individual mailboxes: the *agents* mailbox and a small digest mailing list, *dai*. For the digest mailing lists

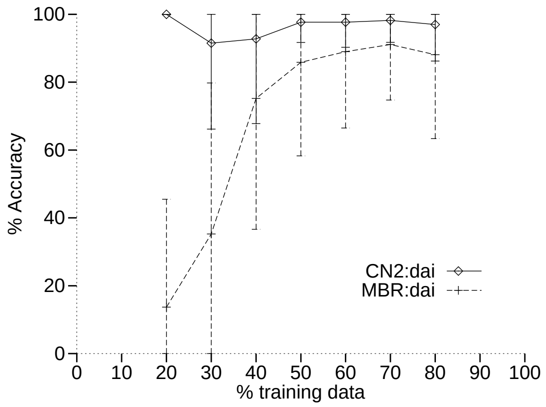

Figure 5: Accuracy of Predictions made for the *dai* Digest.

which contained large numbers of messages (27 messages in *kdd* and 90 messages in *mead*) both CN2 and MBR were able to predict messages with near 100% accuracy. Whilst CN2 produced accurate predictions for the small digest mailbox *dai*, MBR performed badly (see Figure 5). This can be explained by considering the voting strategy used by MBR, where the top k messages are considered in determining the classification. In this case, k was set to 10. Cover (1968) demonstrated that a larger value of k results in an improvement in the behaviour of large samples, at the expense of small sample behaviour.

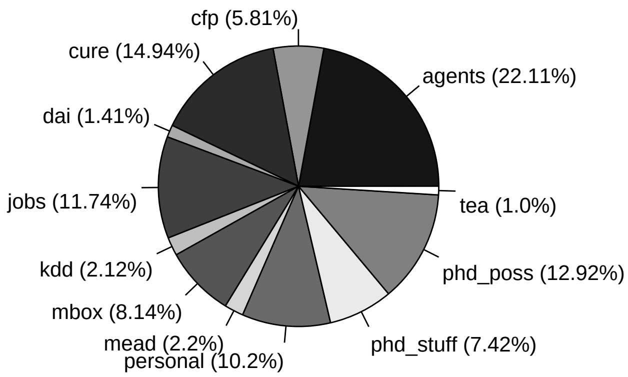

Figure 6: Percentage of Rules used in Magi for the Different Mailboxes.

Figure 6 shows the proportion of rules generated for each mailbox by CN2. The digest mailboxes rely on a very small number of rules. This is due to the *From* and *Subject* fields being similar across all messages within a digest. However, for other mailboxes which have different *From* and *Subject* features, there is a steady increase in the number of rules as the number of messages in the mailbox increases. This indicates that there are few features which are common to all messages within a given mailbox. Because of this, a larger number of rules are need to cover the larger number of less common features found in the messages.

6.2 UNA Experimentation

Experimentation with UNA concentrated on examining the feature extraction mechanism and investigating whether a hotlist of words would improve performance. The percentage of correctly predicted news articles varied between 30% and 80%, depending on the newsgroup and learning algorithm. With a broad classification (where articles are either interesting or dull), an average of 59% of articles were correctly predicted with CN2, compared to 51% when using MBR. This contrasts with using narrow classifications (6 classes, 3 positive and 3 negative) where only an average of 27% of articles were correctly predicted with CN2, and 25% with MBR. See Figures 7 and 8 for examples of UNA classifying broad and narrow classifications.

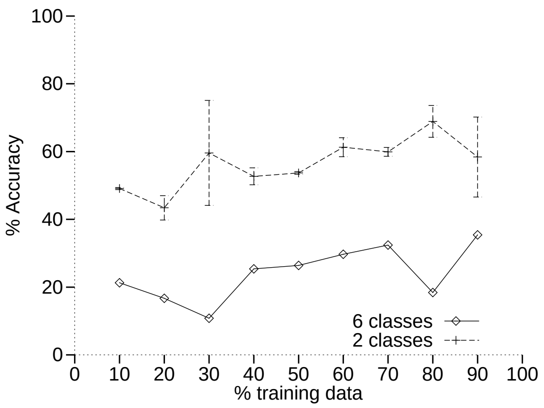

Figure 7: Narrow vs. Broad Classifications for *rec.humor* - CN2.

Experimentation was also carried out to investigate alternative methods of feature extraction. These methods consisted of extracting the contents of the *From* field, the length of the article, and also implementing a user defined hotlist of significant words to identify relevant features, as well as investigating combinations of these methods.

The results were inconclusive for all alternative methods of feature extraction, as none of the methods tested resulted in a significant improvement in accuracy. For example, for some groups the addition of a hotlist improved performance, whilst degrading it for

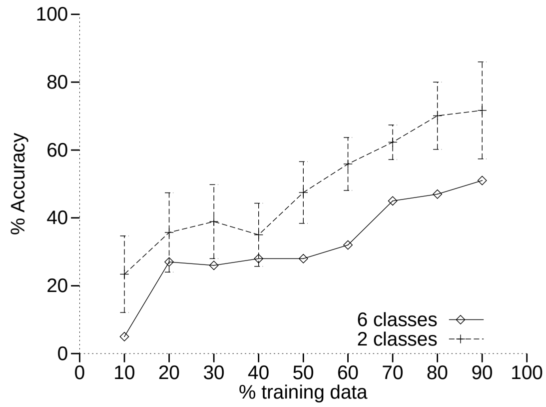

Figure 8: Narrow vs. Broad Classifications for *rec.humor* - MBR.

others. Figures 9 and 10 show the change in performance when the basic feature extraction mechanism was adapted to include a hotlist for the newsgroup *sci.psychology*.

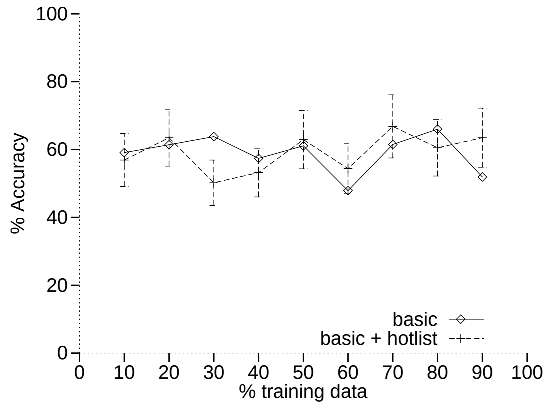

Figure 9: Effect of Hotlist on Predictive Accuracy for *sci.psychology* - CN2.

It can be seen that there is no significant increase in accuracy with the addition of a hotlist, regardless of the learning algorithm used. As the articles rated by the user are already clustered into a particular newsgroup, the frequently occurring words in one article are likely to occur in many articles across the newsgroup.

7 Discussion

Previous work in this area has concentrated on issues such as the interaction between the user and the agent (e.g. Maxims or NewT), or different classification mechanisms, as in NewsWeeder. The motivation

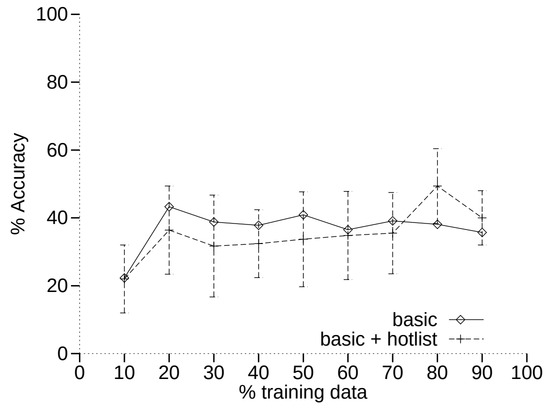

Figure 10: Effect of Hotlist on Predictive Accuracy for *sci.psychology* - MBR.

behind our agent architecture was the development of a testbed to explore different aspects of interface agent technology. For example, different learning techniques can be compared and various approaches to feature extraction can be explored.

The two agent systems described in this paper use the same feature extraction mechanism, which extracts words according to word frequency. The underlying assumption here is that words which act as good classifiers for identifying message topics appear frequently. Whilst this model appears to work for Magi, where the task is primarily that of grouping together related messages, it is unsuitable for UNA where articles have already been sorted into topics, or newsgroups. Features identified by the current feature extraction module are not ideal for determining the user's interest in an article.

The performance of UNA degrades significantly when multiple narrow classifications are used. We are currently studying this phenomenon, however as the number of classes increases, there is a greater chance of features appearing in more than one class. Algorithms such as CN2 and MBR consider each classification as distinct from the others, as a result, such features will be considered as poor classifiers.

An important difference between the two algorithms is the time taken to induce and apply user profiles to new articles. The instance based approach builds a sub-symbolic representation in the form of weights and distance metrics. Unlike rule induction in CN2, these calculations do not involve searching through a large space of possible solutions. The search performed by CN2 is compounded by the large number of features generated by the article body. It was found that tests involving CN2 took significantly (30 to 40 times) longer than tests involving MBR.

Considerations such as speed of profile induction and classification are important. In order to induce a user profile based on observations, many examples are needed, and large log files are generated. As agent technology is applied to commercial tools such as web browsers and email filters, these issues have to be considered.

8 Future Work

Our agent model is currently being applied to the task of identifying interesting information on the World-Wide Web. An agent is being developed which logs pages visited by the user. From this, a user profile is induced which can be used to assist the user in two ways. As the user examines Web pages, interesting links are highlighted. This is similar to the approach used in WebWatcher (Armstrong *et.al.* 1995). In addition, a Web search engine is being developed which uses the profile to search for links to other pages of interest. These links will then be presented to the user.

Techniques such as TF-IDF are being explored and compared to the feature extraction and learning techniques described above. We also hope to investigate the use of genetic algorithm techniques to develop user profiles and plan to evaluate such techniques for filtering USENET news, email, and locating information on the World Wide Web.

Acknowledgements

T.R.Payne acknowledges financial support provided by the UK Engineering & Physical Sciences Research Council (EPSRC). We gratefully acknowledge the authors and maintainers of *Xmail* and the authors of *xrn* for allowing us to use their software. We thank David Dyke for providing a number of amusing images, bringing UNA to life.

References

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

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

Clark, P. & Niblett, T. (1989). The CN2 Induction Algorithm. *Machine Learning.* **3**: 261-283.

Cover, T.M. (1968). Estimation by the Nearest Neighbor Rule. *IEEE Transactions on Information Theory.* **14**(1):50-55.

Gil, Y. (1994). Trainable Software Agents. *Software Agents: Papers from the 1994 Spring Symposium.* 99-102. Menlo Park, CA:AAAI Press.

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

Kozierok, R. & Maes, P. (1993). A Learning Interface Agent for Scheduling Meetings. *Proceedings of the ACM-SIGCHI International Workshop on Intelligent User Interfaces.* 81-88. New York, New York:ACM Press.

Lang, K. (1995). NewsWeeder: Learning to Filter Netnews. *To appear in Proceedings of the 12th International Machine Learning Conference (ML95).*

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

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

Mitchell, T.M., Caruana, R., Freitag, D., McDermott, J. & Zabowski, D. (1994). Experience with a Learning Personal Assistant. *Communications of the ACM.* **37**(7):81-91.

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

Payne, T. & Edwards, P. (1995). Interface Agents that Learn: An Investigation of Learning Issues in a Mail Agent Interface. *Submitted to International Journal of Human-Computer Studies.*

Salton, G. & McGill, M.J. (1983). *Introduction to Modern Information Retrieval.* New York: McGraw-Hill.

Sheth, B.D. (1994). A Learning Approach to Personalized Information Filtering. *MSc Thesis, Department of Electrical Engineering and Computer Science, MIT.*

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