A Learning Personal Agent for Text Filtering and Notification

Anandeep S. Pannu
pannu+@cs.cmu.edu

Katia Sycara
katia@cs.cmu.edu

The Robotics Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Content Areas: *software agents, information retrieval*
Word Count : 4833

A Learning Personal Agent for Text Filtering and Notification

Anandeep S. Pannu Katia Sycara
pannu+@cs.cmu.edu katia@cs.cmu.edu
The Robotics Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

The WWW is increasingly being used for announcements of important events and solicitations such as conference announcements and requests for proposals. This information is accessed by users using direct manipulation tools. The volume of this information is increasing daily and users currently must sift through large amounts of text to access relevant information. We describe a reusable agent that learns a model of the user's preferences, scouts appropriate information sources, filters the information, and notifies the user when relevant information becomes available. The agent is a Personal Assistant which operates autonomously with minimal user intervention. The agent's task is to identify conferences and request for proposals that fit a user's research interests. For this task, there is a large volume of irrelevant documents and the proportion of relevant documents is very small. It is also critical that the agent not misclassify relevant documents, if need be at the cost of misclassifying a few irrelevant documents. Information Retrieval and Neural Network techniques were utilized to learn the model of user's preferences. Readily available textual information was used for training, so that the agent's performance at startup is quite high. The agent has been evaluated through extensive experimentation under a variety of conditions. The results are analyzed and the comparative performance of the learning techniques used are discussed. An interesting result observed is that though Neural Network techniques are inferior in performance to Information Retrieval techniques for learning from text in more general text filtering tasks, in our task they had comparable performance.

1 Introduction

The WWW is increasingly being used to propagate time sensitive and important information. There are newsgroups and Web pages announcing calls for papers, requests for proposals and new publications. A user may know the location of such information but has to repeatedly access the information using direct manipulation computer tools such as Mosaic, Netscape and news-readers. Even if the access is automated, the problem of sifting through all the information remains, since the information sources have a high percentage of "noise" or irrelevant data. In this paper, we investigate how a Personal Agent could be structured to acquire a user profile which enables it to distinguish between relevant and irrelevant documents in text form on the WWW. This user profile is then used to accomplish the task of automating the retrieval and selection of documents from information sources that otherwise have to be constantly monitored by the user. In particular, our agent is learning profiles for notifying users about conference announcements and requests for proposals that match their research interests.

In contrast to environments where the goal is to avoid information overload (eg., the Learning Interface Agent [Maes and Kozierok1993] and NewsWeeder [Lang1995]) the filtering task for our agent involves judging whether an article is relevant or irrelevant to the user based on the user profile, in an environment where the prior probability of encountering a relevant document is very low compared to the probability of encountering an irrelevant document. In such an environment, it would be very frustrating and time consuming for a user to interact with an agent that starts with no knowledge but must obtain all its training examples through user feedback. Therefore, our Learning Personal Agent (LPA) gets trained from available relevant knowledge as well as from user feedback. In this way when the agent starts interacting with the user, although it needs additional training, it can be of immediate use. Since the user preference model learned by the LPA involves retrieving conference announcements and requests for proposals which are related to the research interests of the user, papers written by the users in their research areas can be used to model a priori preferences. This ensures that when the LPA starts interacting with the user it correctly classifies a large proportion of the conference announcements and requests for proposals.

Learning Personal Agents have been used for the information filtering from the WWW [Lang1995], [Armstrong *et al.*1995], [Pazzani *et al.*1995]. In case of the WebWatcher [Armstrong *et al.*1995] and the agent described in [Pazzani *et al.*1995] the agent tries to find an "interesting" link in a Web Page that has already been preselected by a user, which means that the probability of finding relevant links is relatively high. Similarly in the Newsweeder the user is subscribed to newsgroups which are of interest to the user and have a large proportion of relevant articles. In the information sources monitored by our agent the proportion of relevant documents is quite small and irrelevant documents predominate. The cost of missing a relevant document to the user in the environment of our agent is very high. Our agents predominant concern is to interpret the user profile in such a way as to not miss a relevant document. The intent of the other agents is to avoid information overload since the cost of missing a relevant news article or a link is not high, given the redundancies in articles and links.

In the rest of the paper we present the design decisions and learning methods used in our LPA. Our agent learns utilizing Information Retrieval and Neural Network techniques. The agent is evaluated through extensive experimentation under a variety of conditions.

2 The Design Characteristics of the Learning Personal Agent

Figure 1 shows an instantiation of our architecture to accomplish the task of forwarding relevant documents to the agent's *owner* (the user who has control of the Learning Personal agent). The goal of this research effort is to put in

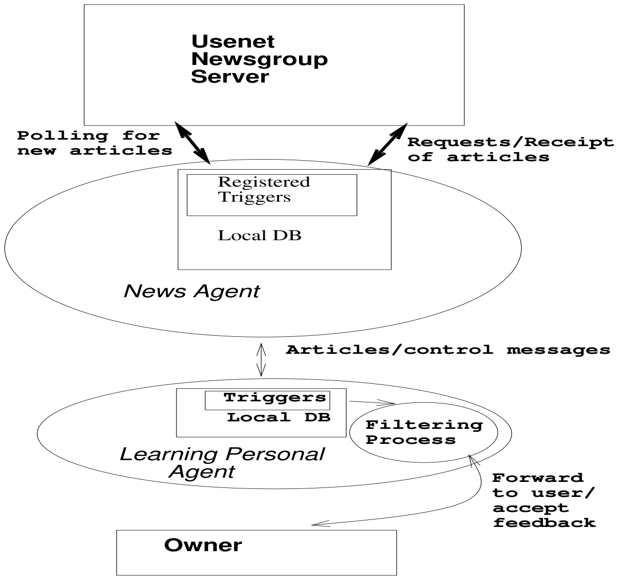

Figure 1: Learning Personal Agent and Environment

the hands of faculty members an LPA to retrieve conference announcements and requests for proposals based on their research interests. Each LPA uses the same learning methods and interface but learns a different user model. We have experimented (see Section 6) with the LPA of one user and we are currently setting up the LPA's of other users.

The News Information agent polls the newsgroups for messages. When a new message arrives it forwards the article to all the Personal agents (owned by different individuals) that have *registered* with it. Each Personal agent activates itself on receipt of the message and runs the text of the article through a *filter*. This filter is based on a model of the owner's preferences and has been learned earlier. If the learned filtering model judges the article as being relevant the article is allowed to 'filter' through to the owner. The owner is notified by e-mail or a pop-up window.

The task of retrieving relevant documents entailed:

1. building a model of the features in the text predicting the relevance of the text to the user

2. developing a learning mechanism which utilizes the feature model to infer different relevance criteria for each user

3. classifying a body of text based on the inference and making the decision as to whether to notify the user or not, based on a decision making policy.

In designing the agent some explicit design criteria were kept in mind. We needed to avoid excessive set up costs. We felt that the cost of setting up the agent for a new user should be minimal in term of the user's involvement in training the system. The agent should have the capability to track evolving user preferences without excessive user time investment. Finally it was desired that the agent not be computationally expensive when functioning autonomously and be able to delay tasks that were computationally expensive to times when system load is low.

We found that some types of filtering models were more suited to our task than others. For example *Static Filtering Models* are models that involve no learning. These could be rule based models of user preference for instance. These models had high initial set up costs because a complete set of rules need to be acquired. They would not incorporate a change in user preferences without major modification. Hence we ruled them out. *Dynamic Learning Filtering Models* could easily incorporate evolving user preferences since they are constantly updated. User feedback is incorporated while carrying out the task, and filtering and learning take place simultaneously. The WebWatcher [Armstrong *et al.*1995] and the Learning Interface Agents [Maes and Kozierok1993] construct user preference models for filtering which are Dynamic Learning Filtering Models. The owner looks at a document and decides whether it is relevant or not, giving appropriate feedback to his/her Learning Personal agent. In a Dynamic Learning Filtering system the agent modifies its model of the user's preferences based on this criteria. This would be very time consuming and frustrating if the user had to spend a lot of time sequentially classifying documents till the agent reached a reasonable level of expertise. In these systems there is no initial training set as the system acquires knowledge based solely on user feedback. This is a not a structural problem but one that is based on the operational needs of the agent. To ensure that the agent starts up with a reasonable performance level we decided to use a *Semi-Dynamic Filtering model.* These models have a training phase which is computationally intensive. Once the training phase is over the model is learned and can be used for filtering. The training phase can be repeated to learn from new instances periodically ie. learning based on user feedback. The user is asked to provide the agent with readily available relevant and irrelevant documents. For our agent, abstracts from conference, journal and workshop papers could be used to learn user research interests that are used in retrieving conference announcements and requests for proposals. The agent used these documents to acquire off-line a high degree of classification competence, so its performance when it starts interacting with the user is quite high.

3 Learning the User Profile

The learning mechanism used to construct the model of user preferences is based on using features of the text in training examples. The training examples are text files supplied by the user. For a Personal Agent that monitors the WWW for conference announcements and requests for proposals, the training examples comprise text files that contain abstracts of papers and proposals written by the owner and abstracts of papers/proposals written by persons who do not share any of the research interests of the owner. The learning mechanism uses the owner's abstracts as *positive examples* of the concept (relevant articles) to be learned and the other abstracts as *negative examples*.

Much of the data found on the Web is in unstructured text form. The agent has to alert the user to the presence of relevant documents based on the text in the document. Learning programs require that the examples be presented as a set of feature vectors. For learning from text, the features we picked to be included in the vectors were the words occurring in a document. For the purposes of this paper a word is a sequence of letters, delimited by non-letters. This is similar to NewsWeeder's [Lang1995] approach of using *tokens* which are generalized words. The granularity of feature size has been kept to word level to avoid the enormous task of developing linguistic knowledge needed for Natural Language Processing.

The assumption is that there is some underlying or "latent" structure in the pattern of word usage across documents [Foltz and Dumais1992]. The need therefore is to come up with a method to estimate the latent structure. Thus words are used as the building blocks of the structures that result in the prediction of the relevance of text to a user. Not all words that appear in a document are used as features, only the ones that are expected to contribute significantly to the estimation of the latent structure and indirectly to the classification of the document as relevant or irrelevant. Statistical criteria and heuristics are used to decide which words contribute significantly.

In the techniques described later in this paper we used the *vector space* information retrieval paradigm, where documents are represented as vectors [Salton1989]. Assume some dictionary vector $\vec{D}$ where each element d_i is a word. Each document then has a vector representation $\vec{V}$, where element v_i is the weight of the word d_i for that document. If the document does not contain d_i then $v_i = 0$. The term *feature* refers to an element of the vector V for a document and so is denoted by v_i.

4 Using an Information Retrieval Based Technique for Filtering

Once we decided on a representation of the documents as described in Section 3, one of the techniques adopted for learning a user preference model was a standard well tested technique from Information Retrieval called term frequency-inverse document frequency weighting (tf-idf) [Salton1989]. This is a simple technique that is hard to beat [Lang1994] and provides a well tested benchmark to which other learning models can be compared.

This technique relies on the occurrence properties of the terms in document collections. In the composition of written texts grammatical function words such as "and", "of" and "or" exhibit approximately equal frequencies of occurrence in all the documents of a collection and the frequencies of occurrence are high. Other words, including words that relate to document content, tend to occur with varying frequencies in the different texts of a collection. The more times a word t occurs in a document d the more likely it is that t is related to the topic of d. The number of times t occurs throughout *all* documents is called the *document frequency* of t or df_t. The larger df_t the worse t discriminates between documents. So for a given document, the relevance of the document based on a term is directly proportional to $tf_{t,d}$ (the number of times word t occurs in document d) and inversely proportional to df_t. We allocate a weight for each word encountered in a collection of documents as follows

$w(t,d) = tf_{t,d} log(|N|/df_t)$

where N is the entire set of documents. The classification power of a word in a document is indicated by this weight.

Each document is represented by a vector $\vec{V}$ as explained in the previous section. Each element v_i of this vector contains the tf-idf weight calculated as shown above for the word d_i from the dictionary $\vec{D}$.

Once each document is represented as a vector, the similarity or dissimilarity of two documents can be measured by using the cosine angle between two vectors representing the two documents. To learn a user profile a set of documents intended for training is converted into vectors and each document's classification noted. For each vector $\vec{V_i}$ classified in the same category each corresponding individual element v_{ij} is summed up. The average of each vector element is then taken forming a *prototype vector* for that category. When a new document is to be classified the "similarity" or "dissimilarity" of the tf-idf vector representation of the new document to each prototype vector is calculated, using the cosine angle measure.

The vectors used in tf-idf could potentially be the size of the total vocabulary of the documents in the training set. The large size of the vector can give rise to the problem of learning in high dimensional spaces. Heuristics are needed to cut down the size of the vector. We used an *information-based* and a *keyword based* approach to make the problem manageable.

In the information based approach we threw out from the dictionary vector $\vec{D}$ the k most common words. We used only n words from the dictionary for classification. These n words corresponded to the top ranking n vector elements in the documents vector in terms of tf-idf weight. The values of k and n for best performance were determined empirically. k values tested were 10, 20, 50, 75 and 100. n took the values 200, 1000 and 2000.

The keyword based approach uses a dictionary vector $\vec{D}$ of words that we know are important in the documents (ie. *keywords*). These words were gleaned from the Research Interests Database at Carnegie Mellon University for the conference proceedings and research proposal retrieval domain. The size of the tf-idf document vector is fixed for every document though the value of each element in the vector (ie. the tf-idf weight) differs across documents. This means that in the domain knowledge based approach we are making inferences from the distribution of a fixed set of words over a collection of documents. We report the results obtained by using this technique and compare its' performance to a Neural Network based technique in Section 6.

5 Using a Neural Network Based Technique for Filtering

As an alternative we used a Neural Network based technique to learn a model of user preferences. The learning process used the same document vector as the tf-idf process described above. The network was trained using the same positive and negative examples as the tf-idf technique. The dimensionality of the document vectors was reduced using the domain knowledge based on the approach described in section 4. The significant differences between the NN technique and the Information Retrieval based technique involve the assumptions underlying the model and the representation of the document vector.

The NN technique does not make any assumptions regarding the statistical properties of the words in the training set, unlike the tf-idf approach which weights the effect of a word in a document based on its frequency of occurrence. We use a simple three layer back-propagation network to *learn* the association between these properties given a training set of documents. These emergent associations are then used to classify new documents. Our belief was that the network would be able not only to learn the model that tf-idf (for instance) uses but also learn more associations that are implicit in the training data.

In the Information Retrieval technique the document vector is a real number vector in which the tf-idf weight of each word is represented as explained in a previous section. For the Neural Network we used a zero-one Boolean document vector, with a one indicating that a word is present in the document. So each v_i in a document vector $\vec{V}$ could either be a 0 or a 1.

To classify documents using a Neural Network a threshold value was chosen. A high output is an output value which exceeded the threshold value. Once the Neural Network is trained, a high output value during the classification of a new document indicates that the document being classified is of interest to the user and that the agent should notify the user about the arrival of the document.

6 Experimental Setup and Results

6.1 The Information Retrieval based Filtering Technique

The tf-idf technique was tested with both the information-based and domain knowledge based approach. We were able to make use of documents from two different users. For one user we obtained 179 documents that were used as positive examples and for the other user we obtained 112 documents for use as positive examples. The number of negative examples (i.e documents classified as irrelevant to the user) was also of the same order (178 and 112 respectively). The documents contained abstracts from conference, workshop and journal papers that were written by the users (positive examples) or others with different research interests (negative examples) in addition to a smaller number of calls for papers and conference announcements. It should be noted that both the users were able to provide us these documents quite easily.

The documents available for learning the user profile were not randomly selected from the same distribution as the documents that the LPA would be classifying. The positive examples in particular shared the same author and therefore were correlated with each other. The training examples thus biased the learning. They had the advantage that using them allowed the agent to acquires a degree of competence that is of immediate advantage to the user.

However the true test of the agent's performance would be when it was presented with conference announcements and request for proposals from WWW sources distinct from the training set. Therefore we used 50 documents acquired from the WWW not previously seen by the agent. This set of documents were set aside for testing and evaluation purposes and were not used for training.

The training set (the original set of documents) was used to form the *prototype vector* for each category as described earlier. The documents from the test set were used to evaluate the performance of the technique. After a document was classified by the system, user feedback was solicited. The user classified the document and the document was added to the training set. Training was done off-line with the augmented training set. The performance of the technique was evaluated after retraining.

We evaluated the performance of the two techniques using the measures of *recall* and *precision*. These are information retrieval measures rather than machine learning measures of performance. Recall is the ratio of relevant documents (i.e. positive examples) identified as relevant by the system by the number of relevant documents present. Precision is the ratio of the number of relevant documents identified as relevant, to the total number of documents presented to the system. The classification accuracy of the technique was quite good. The initial performance was high in both the information based and keyword based approaches. This performance was as high as a recall of 100% and a precision of 93% despite the training examples being picked from a biased sample. The performance also increased when additional feedback was given by the user. This is desirable performance for our agent, which can start up with an initially high performance and a biased training set, but can rid itself of the bias as the user gives feedback during regular operation of the agent.

A surprising result observed was that the information based approach performed better than the keyword based approach despite the greater domain knowledge incorporated into the latter approach.

6.2 The Neural Network based Filtering Technique

It is well known that the Neural Network's performance is sensitive to the training process used. We were careful to avoid some of the pitfalls such as *over-training*. The approach we used was to divide up all the example documents available into a training set and a coordination set in a certain proportion, randomly choosing the documents to go into each set. These sets were mutually exclusive collections of documents. The test set used to evaluate performance was the same as the one used to evaluate the Information Retrieval Approach.

The *training set* was the set of documents that were used to train the network. The document vectors were presented to the network at the input nodes after being converted to a Boolean zero-one feature vector. Each input node represent an element v_i of the document vector $\vec{V}$. The output of the network was compared to the representation of the relevance of the document called the *expected output* (1 for a relevant example and -1 for an irrelevant example). The error between the expected output for the document and the forward propagated output was then back-propagated for a number of *epochs*. In case the network output was greater than 0.5 for an expected value of 1, or less than 0.5 for an expected value of -1 the document was marked as being classified correctly. Otherwise the document was marked as being classified incorrectly. The documents in the *test set* were used to test the performance of the network previously unseen text. The performance measures were the same as the ones used in the Information Retrieval based technique. The *coordination set* is a third set that is used neither to train the network nor to evaluate its performance. The documents belonging to the coordination set were used to decide how many epochs to train the network for to give optimum performance in classification of new examples.

In order to get the maximum performance on previously unseen documents the training process needs to be carried out for a number of *epochs*. The decision of the number of epochs at which to stop training is critical to the performance of a NN. If the performance of the training set (in terms of minimizing the total error between the expected output and the network generated output on all the feature vectors in the training set) is used to decide when to stop training - there is a danger that the network will be *over-trained*. That is, the network will pick specific features of the individual examples presented to it to classify rather than features that are truly predictive of class membership. Performance (in terms of network error) of the test set cannot be used to guide training since the test set is used for evaluation.

The coordination set is not used to train the network. The documents in the coordination set were used to decide how many epochs to train the network for, to give optimum performance in classification of new examples. In every epoch after the network is trained it is used to classify the documents in the coordination set. Each document vector $\vec{V}$ is presented to the input nodes, each of which represents an element v_i of the vector. The error between the network output for the document vectors and the expected output is summed up for each document in the set. This is compared to the same quantity calculated in the previous epoch. Figure 2 shows a plot of the sum of the error between expected output for a document vector and the output from the neural network when the document vector was presented as input for all documents in the coordination set versus the number of epochs the NN has been trained for. The same plot is repeated for the documents training set. The optimum performance occurs at point A in the training. The error of the training set keeps decreasing from point A but the error of the coordination set increases. This means that training the network beyond the number of epochs at point A will result in "over-training" in which the network learns particular associations to get better performance on the training set but loses power of generalization. Thus the decision to stop training is made when the total error on the coordination set shows a rising trend ie. error in the previous epoch is less than the error in the current epoch.

The network was then trained to the point corresponding to A in Figure 2 be-

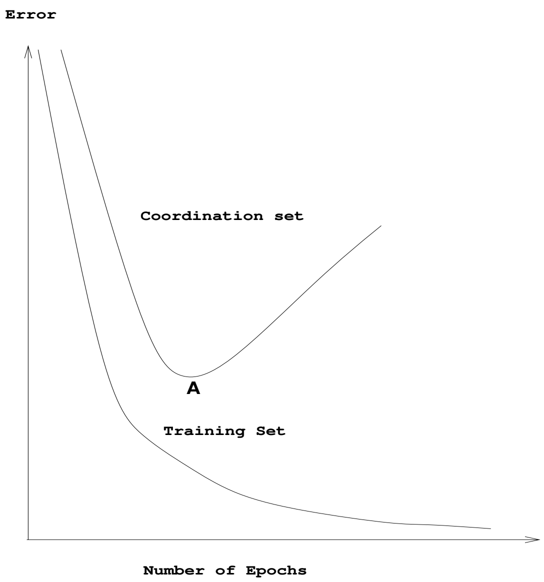

Figure 2: Error versus number of epochs for NN

fore being evaluated. Other Neural Network parameters were chosen to maximize classification accuracy on a set of documents split randomly from the training set. The coordination error decreased significantly for a number of epochs but starts stagnating before increasing. To keep the time for the experimental runs within manageable limits the NN training was terminated when the error decrease was less the $1 * 10^{-6}$ between epochs or when the trend was towards increasing error.

The results reported are for a 360 word vector of keywords, and training time was roughly comparable to the tf-idf algorithm using the keyword based approach (which resulted in a document vector length of 360 words). The training time for tf-idf was significantly longer for the longer length vectors. The vector length did not have a significant impact on the training time for the NN.

The performance of the Neural Network was worse than the performance of the Information Retrieval technique both in terms of precision and recall. The results obtained by us are not completely disappointing however. Despite the small number of training examples used (357) compared to those reported in other literature (for instance [Lang1994] which used 20000 examples), we were able to get performance approaching 60% recall and a precision of 94%. For the Cascade 2 neural network growing algorithm trained on a 1000 articles, Lang [Lang1994] reported 25% correct classification. In our case the Neural Network has approximately 3 times the performance achieved by the Cascade 2 algorithm. This is no doubt due to the fact that our text filtering task is more specialized than the task for [Lang1994]. The precision of increases with user feedback as is expected.

7 Conclusions and future work

We described a Learning Personal Assistant that learns a model of the user's preferences in order to notify a user when relevant information becomes available. The learning task entailed using Information Retrieval and Neural Network techniques to identify conferences and request for proposals that fit a user's research interest. The results reported in this paper are preliminary. The two approaches used here are both good candidates for the task of creating a user profile to filter documents. The Information Retrieval based approach is the most promising. The Neural Network based approach uses an extremely simple representation for a document (a Boolean zero-one vector). Another approach that we are experimenting with is to combine the strengths of tf-idf and the Neural Network. Currently we are setting up experiments to collect data from a number of users. We have concluded that it is possible to use a biased set of training examples, in order to get high initial performance and yet get acceptable performance when classifying documents that are not part of the biased set.

References

[Armstrong *et al.*1995] Robert Armstrong, Dayne Freitag, Thorsten Joachims, and Tom Mitchell. WebWatcher: A Learning Apprentice for the World Wide Web. In *Proceedings of AAAI Spring Symposium on Information Gathering from Heterogenous Distributed Environments*, 1995.

[Foltz and Dumais1992] P W Foltz and S T Dumais. Personalized Information delivery : An Analysis of Information Filtering Methods. *Communications of the Association for Computing Machinery*, December 1992.

[Lang1994] Ken Lang. Newsweeder: An Adaptive Multi-User Text filter. Research Summary, Carnegie Mellon University, August 1994.

[Lang1995] K Lang. Learning to Filter Netnews. In *Proceedings of the Machine Learning Conference 1995*, 1995.

[Maes and Kozierok1993] Pattie Maes and Robyn Kozierok. Learning Interface Agents. In *Proceedings of the Eleventh National Conference on AI*. AAAI, AAAI Press/MIT Press, 1993.

[Pazzani *et al.*1995] Michael Pazzani, Larry Nguyen, and Stefanus Mantik. Learning from Hotlists and Coldlists : Towards a WWW Information Filtering and Seeking Agent. In *Proceedings of the Seventh International IEEE conference on Tools with AI*. IEEE Computer Press, 1995.

[Salton1989] G Salton. *Automatic Text Processing. The Transformation, Analysis and Retrieval of Information by Computer*. Addison-Wesley, 1989.