A Hybrid User Model for News Story Classification

Daniel Billsus and Michael J. Pazzani*

Dept. of Information and Computer Science, University of California, Irvine, CA, USA

Abstract. *We present an intelligent agent designed to compile a daily news program for individual users. Based on feedback from the user, the system automatically adapts to the user's preferences and interests. In this paper we focus on the system's user modeling component. First, we motivate the use of a multi-strategy machine learning approach that allows for the induction of user models that consist of separate models for long-term and short-term interests. Second, we investigate the utility of explicitly modeling information that the system has already presented to the user. This allows us to address an important issue that has thus far received virtually no attention in the Information Retrieval community: the fact that a user's information need changes as a direct result of interaction with information. We evaluate the proposed algorithms on user data collected with a prototype of our system, and assess the individual performance contributions of both model components.*

1 Introduction

Research on intelligent information agents has recently attracted much attention. As the amount of information available on-line grows with astonishing speed, people feel overwhelmed navigating through today's information and media landscape. Information overload is no longer just a popular buzzword, but a daily reality for most of us. This leads to a clear demand for automated methods, commonly referred to as intelligent information agents, that locate and retrieve information with respect to users' individual preferences (Lang, 1995; Pazzani and Billsus, 1997; Balabanovic, 1998).

As intelligent information agents aim to automatically adapt to individual users, the development of appropriate user modeling techniques is of central importance. Algorithms for intelligent information agents typically draw on work from the Information Retrieval (IR) and machine learning communities. Both communities have previously explored the potential of established algorithms for user modeling purposes (Belkin et al. 1997; Webb 1998). However, work in this field is still in its infancy and we see "User Modeling for Intelligent Information Access" as an important area for future research.

We describe an intelligent information agent designed to compile a daily news program for individual users. Building such an agent is a challenging task, because traditional Information Retrieval approaches are not directly applicable to this problem setting. Most IR systems assume

* We thank Daimler-Benz and Sun Microsystems, Inc. for their generous support.

the user has a specific, well-defined information need. In our setting, however, this is not the case. If at all, the user's query could be phrased as: "What is new in the world that I do not yet know about, but should know?". Computing satisfactory results for such a query is non-trivial. The difficulty stems from the range of topics that could interest the user, and the user's changing interest in these topics. We must also take into account that it is the novelty of a story that makes it interesting. Even though a certain topic might match a user's interests perfectly, the user will not be interested in the story if it has been heard before. Therefore, we need to build a system that acquires a model of a user's multiple interests, is flexible enough to account for rapid interest changes, and keeps track of information the user knows.

We focus on two issues related to the automated induction of user profiles for news story classification. First, we motivate the induction of a hybrid user model that consists of separate models for a user's long-term and short-term interests. Second, we show how the user model keeps track of information that has already been presented to the user. This allows us to address an important issue that has thus far received virtually no attention in the Information Retrieval community: the fact that a user's information need changes as a direct result of interaction with information (Belkin, 1997). We evaluate the proposed algorithms on user data collected with a prototype of our system, and assess the individual performance contributions of our proposed approaches separately.

2 A Personal News Agent that Talks and Learns

In this section we present the design and architecture of an agent that is intended to become part of an intelligent, IP-enabled radio, which uses synthesized speech to read news stories to a user. While the system's speech capabilities are not the focus of this paper, we briefly discuss the system's overall functionality and intended purpose in order to motivate various design decisions. Part of our research is based on the observation that most of the work on personalized information access has focused on agents accessible through the World Wide Web. Research in this field has not yet led to the development of interfaces for software agents that do not require access to a computer workstation. However, there is a clear demand for such information systems, as demanding schedules prohibit people from continuous computer access.

The following example motivates the need for personalized information delivery outside of the World Wide Web. Users *A* and *B* spend a large portion of their day driving. They can listen to the radio in their cars, but do not have access to any medium that focuses on information specific to their interests. User *A* is primarily interested in business news in order to follow the current stock market, but he must listen to much unrelated information. To locate information, he must switch news channels and pay attention to the time at which broadcasts start. Similarly, user *B* must switch news channels to locate information on his interest in local sports.

We have implemented a Java Applet that uses Microsoft's *Agent* library to display an animated character, named *News Dude*, that reads news stories to the user. The Applet requires Microsoft's Internet Explorer 4.0 or newer and is publicly accessible at *http://www.ics.uci.edu/~dbillsus/NewsDude* (see Figure 1). Although our ultimate goal is to work towards a speech-driven agent that does not require graphical user interfaces, we use the web as a medium that allows us to make the system available to a large user base for data collection and testing purposes. Furthermore, we believe that there is a variety of useful applications for speech-driven agent technology for the web. For example, a talking news agent that reacts to voice commands could prove useful for the visually impaired.

Currently, the agent provides access to stories from six different news channels: Top Stories, Politics, World, Business, Technology and Sports. When the user selects a news channel, the Applet connects to a news site on the Internet (*Yahoo!News*) and starts to download stories. Since the Applet is multi-threaded, download of stories continues in the background while the synthesizer is reading, which typically allows filling a queue of stories to be read without any waiting time. The user can interrupt the synthesizer at any point and provide feedback for the story being read. One of the design goals for our system was to provide a variety of feedback options that go beyond the commonly used *interesting/uninteresting* rating options. If we consider an intelligent information agent to be a personal assistant that gradually learns about our interests and retrieves interesting information, it would only be natural to have more informative ways to communicate our preferences. For example, we might want to tell the agent that we already know about a certain topic, or request information related to a certain story. In summary, the system supports the following feedback options: *interesting, not interesting, I already know this,* and *tell me more about this*. After an initial training phase, the user can ask the agent to compile a personal news program. The goal of this process is to compute a sequence of news stories ordered with respect to the user's interests.

Figure 1. News Dude user interface.

3 Learning a User Model

The specific design of our agent's user model is motivated by a number of observations and requirements. First, the model must be capable of representing a user's multiple interests in different topics. Second, the model must be flexible enough to adapt to a user's changing interests reasonably quickly, even after a long preceding training period. Third, the model should take into account that a user's information needs change as a direct result of interaction with information (Belkin, 1997). Surprisingly, this aspect has received virtually no attention in the IR community. For our application, we take into account the stories the user has recently heard, to avoid presenting the same information twice.

The above requirements led to the development of a multi-strategy learning approach that learns two separate user-models: one represents the user's short-term interests, the other represents the user's long-term interests. Distinguishing between short-term and long-term models has several desirable qualities in domains with temporal characteristics (Chiu and Webb, 1998). Learning a short-term model from only the most recent observations may lead to user models that can adjust more rapidly to the user's changing interests. Here, we restrict the short-term model to the *n* most recently rated stories (we set *n* to 100 in the experiments reported in Section 4). The need for two separate models can be further substantiated by the specific task at hand, i.e. classifying news stories. Users typically want to track different "threads" of ongoing recent events - a task that requires short-term information about recent events. For example, if a user has indicated interest in a story about a current Space Shuttle mission, the system should be able to identify follow-up stories and present them to the user during the following days. In addition, users have general news preferences, and modeling these general preferences may prove useful for deciding if a new story, which is not related to a recent rated event, would interest the user. With respect to the Space Shuttle example, we can identify some of the characteristic terminology used in the story and interpret it as evidence for the user's general interest in technology and science related stories. In the following sections we describe these two parts of the user model individually, motivate why they conform to the above requirements, and describe how they can work together to form a comprehensive user model appropriate for our purposes.

For the following description of the user model and its automated induction, we assume that a user provides the system with a set of rated news stories, using the feedback options described in Section 2. Each user rating is then internally converted to a score ranging from 0 to 1, according to the following rules. Let *pl* be the proportion of a story that the user has heard.

*If story was rated as uninteresting, score = 0.3 * pl*
*If story was rated as interesting, score = 0.7 + 0.3 * pl*
If user asked for more information, score = 1.0

This conversion scheme allows for differentiation between different levels of ratings without requiring any extra work by the user. Note that the constants *0.3* and *0.7* were chosen arbitrarily with the only constraint being that the resulting scores for stories rated as uninteresting always receive lower scores than stories rated as interesting. In future work we will determine improved values for these constants experimentally, using a tuning set of user data.

3.1 Modeling Short-Term Interests with the Nearest Neighbor Algorithm

The purpose of the short-term model is two-fold. First, it should contain information about recently rated events, so that stories which belong to the same threads of events can be identified. Second, it should allow for identification of stories that the user already knows. A natural choice to achieve the desired functionality is the nearest neighbor algorithm (NN). The NN algorithm simply stores all its training examples, in our case rated news stories, in memory. In order to classify a new, unlabeled instance, the algorithm compares it to all stored instances given some defined similarity measure, and determines the "nearest neighbor" or the *k* nearest neighbors. The class label assigned to the new instance can then be derived from the class labels of the nearest neighbors. The utility of the NN algorithm has previously been explored in other text classification applications (Cohen and Hirsh, 1998; Yang, 1998; Allan et al. 1998).

To apply the algorithm to natural language text, we must define a similarity measure that quantifies the similarity between two text documents. This is a well-studied problem in Information Retrieval, and we rely on a commonly used document representation and an associated similarity measure. We convert news stories to TF-IDF vectors (term-frequency / inverse-document-frequency), and use the cosine similarity measure to quantify the similarity of two vectors (Salton, 1989).

Each rated story is converted to its TF-IDF representation and then stored in the user model. A score prediction for a new story is then computed as follows. All stories that are closer than a threshold *t_min* to the story to be classified become voting stories. The predicted score is then computed as the weighted average over all the voting stories' scores, where the weight is the similarity between a voting story and the new story. If one of the voters is closer than threshold *t_max* to the new story, the story is labeled as *known*, and its computed score is multiplied by a factor $k << 1.0$, because the system assumes that the user has already heard about the event reported in the story. If a story does not have any voters, the story cannot be classified by the short-term model at all, and is passed on to the long-term model (see Section 3.2).

The nearest neighbor-based short-term model satisfies our requirements that a user model be able to represent a user's multiple interests, and it can quickly adapt to a user's novel interests. The main advantage of the nearest-neighbor approach is that only a single story of a new topic is needed to allow the algorithm to identify future follow-up stories from the same story thread. The "tracking" abilities of the nearest neighbor algorithm have also recently been explored by other researchers in a similar project (Allan et al., 1998). In contrast, most other learning algorithms would require a large number of training examples to identify a strong pattern.

3.2 Modeling Long-Term Interests with a Naïve Bayesian Classifier

The purpose of the long-term user model is to model a user's general preferences for news stories and compute predictions for stories that could not be classified by the short-term model. To achieve this we selected a probabilistic learning algorithm, the naïve Bayesian classifier (Duda and Hart, 73). Naïve Bayes has been shown to perform competitively with more complex algorithms and has become an increasingly popular algorithm in text classification applications (McCallum and Nigam, 1998; Pazzani and Billsus, 1997).

We represent news stories as Boolean feature vectors, where each feature indicates the presence or absence of a word. Not all the words that appear in news stories are used as features. Since it is our explicit goal to model a user's general preferences, we provide the algorithm with background knowledge by hand-selecting a set of domain specific features, i. e. words that are likely to be indicators for commonly recurring themes in daily news stories. Approximately 200 words were selected, ranging from countries to crime, disaster, politics, technology, business and sport related terms. Making the "naïve" assumption that features, here words, are independent given the class label (interesting vs. not interesting), the probability of a story belonging to class *j* given its feature values, $p(class_j | f_1, f_2, \ldots f_n)$ is proportional to:

$$p(class_j)\prod_{i}^{n} p(f_i \mid class_j)$$

where $p(class_j)$ and $p(f_i \mid class_j)$ can be easily estimated from training data. Specifically, we use the multi-variate Bernoulli event model formulation of naïve Bayes (McCallum and Nigam, 1998), and compute Bayes-optimal estimates of $p(class_j)$ and $p(f_i \mid class_j)$ by straightforward counting of word and class occurrences in the training data. We use Laplace smoothing to prevent zero probabilities for infrequently occurring words. A news story to be classified can thus be labeled with its probability of belonging to the interesting class.

Most applications of the multi-variate Bernoulli formulation of naïve Bayes consider both the presence and absence of words in text documents as evidence in the probability computation. We restrict the evidence used to the presence of words, similar to a naïve Bayes model proposed by Maron (1961). This results in a more conservative classifier that requires examples classified as class *c* to be similar to other examples in class *c*.

Finally, we would like to prevent the long-term model from classifying stories that do not contain a sufficient number of features that are indicators for class membership. More formally, we require the story to contain at least *n* features for which $p(f \mid interesting) > p(f \mid \neg\, interesting)$ in order to allow a classification as interesting, and likewise, at least *n* features for which $p(f \mid interesting) < p(f \mid not\ interesting)$ in order to allow a classification as not interesting. In our current implementation we set *n* to 3, which means a story must contain at least 3 terms that are all indicators for the same class.

3.3 User-Modeling with a Multi-Strategy Learning Approach

Using a hybrid user model consisting of both a short-term and long-term model of the user's interests, a previously unseen news story, *u*, is classified as follows:

If *∃d: d ∈ {short-term-stories} ∧ cosine-similarity(d, u) > t_min*
{
    score = nearest-neighbor-prediction(u, {short-term-stories})
    If *∃n: n ∈ {short-term-stories} ∧ cosine-similarity(d, n) > t_max*
        *score = score * k, where k << 1.0*
}
Else
    If $\exists\{f_1, f_2,\ldots, f_n\}\colon \forall f \in \{f_1, f_2,\ldots, f_n\}\ p(f \mid c) > p(f \mid \neg c)$
        score = naïve-Bayes-prediction(u, {all stories})
    Else
        score = default

In summary, the approach tries to use the short-term model first, because it is based on the most recent observations only, allows the user to track news threads that have previously been rated, and can label stories as already known. If a story cannot be classified with the short-term model, the long-term model is used. If the long-term model decides that the story does not contain sufficient evidence to be classified, a default score is assigned. In the current implementation we set the default score to *0.3*, so that stories that cannot be reliably classified do not appear too high in the recommendation queue, but still receive a higher score than stories that are classified as not interesting.

4 Evaluation

In order to evaluate the recommendation performance of our agent and to assess the performance contribution of its hybrid user model, we used the web-based agent prototype to collect user data. Ten users trained the system on a daily basis over a period ranging from 4 to 8 days, resulting in about 3,000 total rated news stories, i.e. on average 300 stories per user. While this amount of data might not lead to overall performance estimates that generalize to other users or different collection dates, it allows us to analyze the relative performance contributions of the short-term and long-term user models, as well as the benefit of explicitly modeling known stories.

Evaluating the agent's performance is difficult for several reasons. First, standard evaluation methodologies commonly used in the machine learning literature, for example n-fold cross-validation, are not applicable to this scenario. This is mainly due to the chronological order of the training examples, which cannot be presented to the learning algorithm in random order, without skewing results. Second, if we measure the agent's performance on a daily basis, we not only measure the effects of the agent's updated user model, but also of the changing distribution of news stories. Finally, we are trying to approximate a model of user interests that can be assumed to be neither static nor consistent. A user going through the same list of stories at a later time might assign different labels.

We chose to evaluate the agent's performance as follows. We divided each user's data into separate training sessions, corresponding to the user's use of the system, i.e. typically one training session per day. We started to train the algorithm with all rated examples from the first training session, and compared its predictions for class labels of stories from the second training session to the user's ratings. We then incremented the training set session by session and measured the agent's performance on the following session. Finally, we averaged the results over all users. This methodology models the way the system is used realistically, because all training data available up to a certain day was used to classify stories.

In addition to the system's classification accuracy, i.e. the proportion of correctly classified news stories, we used common Information Retrieval performance measures, *precision*, *recall* and F_1 to evaluate the system. In our domain, precision is the percentage of stories classified as interesting that are interesting, and recall is the percentage of interesting stories that were classified as interesting. It is important to evaluate precision and recall in conjunction, because it is easy to optimize either one separately. For a classifier to be useful for our purposes we demand that it be precise as well as have high recall. In order to quantify this with a single measure, Lewis and Gale (1994) proposed the *F-measure,* a weighted combination of precision and recall that produces scores ranging from 0 to 1. Here we assign equal importance to precision and recall, resulting in the following definition for F_1:

$$F_1 = \frac{2 \cdot precision \cdot recall}{precision + recall}$$

Figure 2 summarizes the system's performance averaged over all users, showing a rapid increase of classification performance during the first three training sessions. Results for more training sessions are only available for a subset of the users and therefore cannot be presented in one plot. However, results for users that collected data of up to 8 training sessions revealed that performance increases rapidly during the first few training sessions and then starts to fluctuate as a result of changing distributions of daily news stories. Figure 2 also shows the relative performance of the two user model components. As expected, the hybrid approach combining a short-term and long-term user model performs better than each individual approach with respect to both classification accuracy and the F_1 measure. Further inspection of the achieved performance revealed that the short-term model tends to have high precision, but low recall. In contrast, the long-term model has higher recall than the short-term model, but lower precision. Combining both models allows taking advantage of both models' strengths, resulting in substantially higher F_1 values as well as overall classification accuracy.

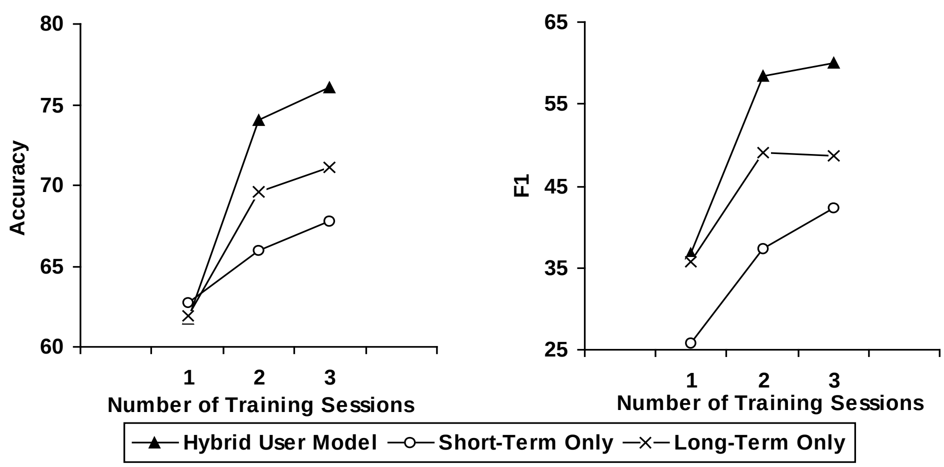

Figure 2. Overall system performance.

In an additional experiment we quantified the utility of explicitly representing stories previously presented to the user. Figure 3 compares the performance of two approaches, here labeled as knowledge-dependent and knowledge-independent classification. The knowledge-dependent approach is the full multi-strategy user modeling algorithm as described in Section 3.3. This approach can classify stories as uninteresting if they are assumed to be known by the user. In contrast, the knowledge-independent approach does not take into account that the user might already know about certain events. Algorithmically, this simply amounts to setting *t_max* to a value > 1 (see Section 3.1), so that predicted scores are never reduced due to the presence of similar stories in the short-term model. Figure 3 shows that knowledge-dependent classification leads to an overall increase of classification accuracy. Knowledge-dependent classification reduces the relevance scores for stories that are assumed to be known, and therefore leads to a substantial increase in precision, but to a decrease in recall (since fewer stories are classified as interesting). Since the overall increase in precision is larger than the decrease in recall, we observed both higher classification accuracy as well as increased retrieval performance as measured by the F_1 statistic.

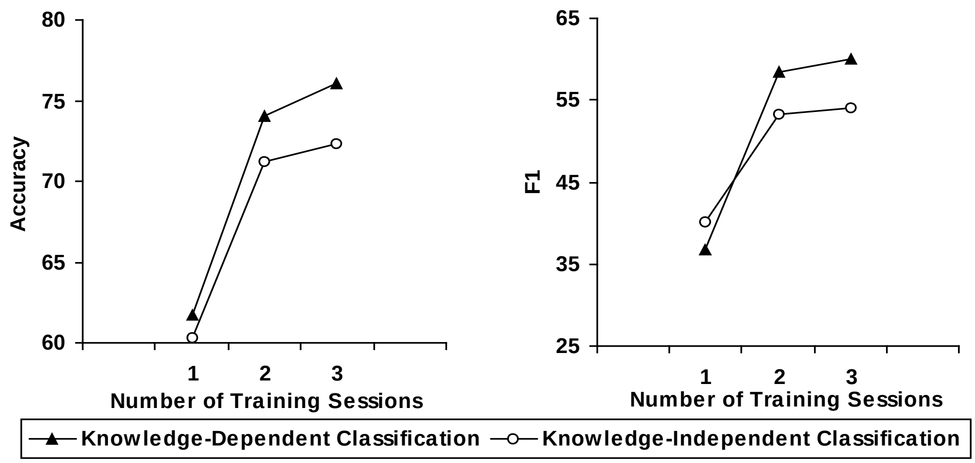

Figure 3. Effect of knowledge-dependent classification

5 Related Work

Chiu and Webb (1998) have previously studied the utility of the induction of a dual user model in the context of student modeling. While the studied domain, data representation and learning algorithms differ significantly from the text classification approach presented here, the underlying motivation for the use of a dual model is similar. In general, user modeling is a task with inherent temporal characteristics. We can assume recently collected user data to reflect the current knowledge, preferences or abilities of a user more accurately than data from previous time periods. However, restricting models to recent data can lead to overly specific models, i.e. models that classify instances that are similar to recently collected data with high precision, but perform poorly on instances that deviate from data used to induce the model. To overcome this problem, Chiu and Webb use a dual model that classifies instances by first consulting a model trained on recent data, and delegating classification to a model trained over a longer time period if the recent model is unable to make an accurate prediction.

6 Summary and Conclusions

We have presented the functionality, design and underlying algorithms of a news agent that learns about a user's interests in daily news stories. The agent uses a multi-strategy machine learning approach to induce user models that model a user's short-term and long-term interests separately. The two models were experimentally shown to complement each other. Furthermore, we have investigated the utility of explicitly representing stories a user has heard before. This allows us to take into account that a user's information need changes as a direct result of interaction with information. Similar news stories can be prevented from being presented multiple times, which was experimentally shown to lead to increased predictive performance.

References

Allan, J., Carbonell, J., Doddington, G., Yamron, J. and Yang Y. (1998). Topic detection and tracking pilot study final report. *Proceedings of the DARPA Broadcast News Transcription and Understanding Workshop*, Lansdowne, Virginia.

Balabanovic, M. (1998). *Learning to Surf: Multiagent Systems for Adaptive Web Page Recommendation*. Ph.D. Thesis, Stanford University.

Belkin, N. (1997). User modeling in Information Retrieval. Tutorial Overheads, available at http://www.scils.rutgers.edu/~belkin/um97oh/, *Sixth International Conference on User Modeling*, Chia Laguna, Sardinia.

Belkin, N., Kay, J., Tasso, C. (eds) (1997). Special Issue on User Modeling and Information Filtering. *User Modeling and User Adapted Interaction*, 7(3).

Chiu, B. and Webb, G. (1998). Using decision trees for agent modeling: improving prediction performance. *User Modeling and User-Adapted Interaction* 8:131–152.

Cohen, W. and Hirsh, H. (1998). Joins that generalize: text classification using WHIRL. In *Proceedings of the Fourth International Conference on Knowledge Discovery & Data Mining*, New York, New York, 169-173.

Duda, R., and Hart, P. (1973). *Pattern Classification and Scene Analysis*. John Wiley & Sons, New York.

Lang, K. (1995). NewsWeeder: learning to filter news. *Proceedings of the Twelfth International Conference on Machine Learning*. Lake Tahoe, CA, 331–339.

Lewis, D. and Gale, W. A. (1994). A sequential algorithm for training text classifiers. *Proceedings of the Seventeenth Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval*. London, Springer Verlag, 3-12.

Maron, M. (1961). Automatic indexing: an experimental inquiry. *Journal of the Association for Computing Machinery*, 8:404-417.

McCallum, A. and Nigam, K. (1998). A comparison of event models for naïve Bayes text classification. *American Association for Artificial Intelligence (AAAI) Workshop on Learning for Text Categorization*. Available as Technical Report WS-98-05, AAAI Press.

Pazzani M., and Billsus, D. (1997). Learning and revising user profiles: the identification of interesting web sites. *Machine Learning 27*, 313-331.

Salton, G. (1989). *Automatic Text Processing*. Addison-Wesley.

Webb, G. (ed) (1998). Special Issue on Machine Learning for User Modeling. *User Modeling and User-Adapted Interaction*, 8(1-2).

Yang, Y. (1999). An evaluation of statistical approaches to text categorization. Manuscript submitted for publication.