Using Content-Based Filtering for Recommendation[1]

Robin van Meteren[1] and Maarten van Someren[2]

[1]NetlinQ Group, Gerard Brandtstraat 26-28, 1054 JK, Amsterdam, The Netherlands, robin@netlinq.nl
[2]University of Amsterdam, Roeterstraat 18, The Netherlands, marten@swi.psy.uva.nl

Abstract

Finding information on a large web site can be a difficult and time-consuming process. Recommender systems can help users find information by providing them with personalized suggestions. In this paper the recommender system PRES is described that uses content-based filtering techniques to suggest small articles about home improvements. A domain such as this implicates that the user model has to be very dynamic and learned from positive feedback only. The relevance feedback method seems to be a good candidate for learning such a user model, as it is both efficient and dynamic.

1 Introduction

As the World Wide Web continues to grow at an exponential rate, the size and complexity of many web sites grow along with it. For the users of these web sites it becomes increasingly difficult and time consuming to find the information they are looking for. To help users find the information that is in accordance with their interests a web site can be personalized. Recommender systems can improve a web site for individual users by dynamically adding hyperlinks. In this paper the recommender system PRES (acronym for *P*ersonalized *Re*commender *S*ystem) is introduced. PRES creates dynamic hyperlinks for a web site that contains a collection of advises about do it yourself home improvement. The purpose of these dynamic hyperlinks is to make it easier for a user to find interesting items and thus improving the interaction between the system and the user.

motivation

When users browse through a web site they are usually looking for items they find interesting. Interest items can consist of a number of things. For example, textual information can be considered as interest items or an index on a certain topic could be the item a user is looking for. Another example, applicable for a web vendor, is to consider purchased products as interest items. Whatever the items consist of, a web site can be seen as a collection of these interest items.

açklanabilir

Every large collection needs a certain structure to make it easy for visitors to find what they are looking for. A web site can be structured by dividing its web pages into content pages and navigation pages. The content pages provide the user with the interest items while the navigation pages help the user to search for the interest items. This is not a strict classification however. Pages can also be hybrid in the sense that they both provide content as well as navigation facilities. Furthermore, what is a navigation page for one user may be a content page to another and visa versa. In general however, this classification provides a way of describing the structure of a web site and how this structure can be improved for individual users by dynamically adding hyperlinks.

[1] This research has been supported by NetlinQ

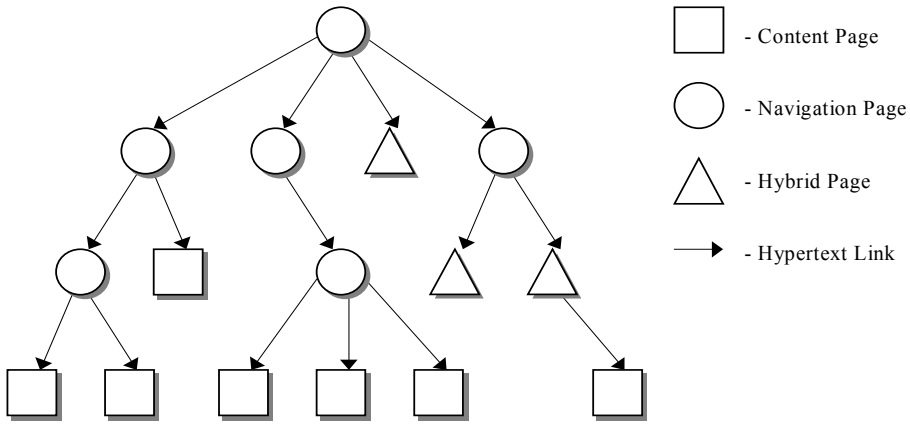

Figure 1 Structure of sample web site

Figure 1 shows an example of a web site with a typical tree structure. The content pages are found at the bottom of the tree while the navigation pages are found at the top. A recommender system can display its recommendations by dynamically creating hypertext links to content pages that contain the items a user might be interested in. Several factors determine whether or not a recommended page should be linked to the page that is shown to the user. Sometimes content pages are only recommended if they contain items that are similar to the item(s) shown on the current page. Another consideration for dynamic linking is the proximity to the recommended page. The distance between two pages is determined by the minimal number of links it takes to navigate from one page to another. There is not much use of linking the current page to a recommended page if the distance between the two pages is 1. The further the distance, the more useful a dynamically created link becomes.

2 Content-based filtering

Recommender systems are a special type of information filtering systems. Information filtering deals with the delivery of items selected from a large collection that the user is likely to find interesting or useful and can be seen as a classification task. Based on training data a user model is induced that enables the filtering system to classify unseen items into a positive class c (relevant to the user) or a negative class $\bar{c}$ (irrelevant to the user). The training set consists of the items that the user found interesting. These items form training instances that all have an attribute. This attribute specifies the class of the item based on either the rating of the user or on implicit evidence. Formally, an item is described as a vector $X = (x_1, x_2, ..., x_n)$ of n components. The components can have binary, nominal or numerical attributes and are derived from either the content of the items or from information about the users' preferences. The task of the learning method is to select a function based on a training set of m input vectors that can classify any item in the collection. The function $h(X)$ will either be able to classify an unseen item as positive or negative at once by returning a binary value or return a numerical value. In that case a threshold can be used to determine if the item is relevant or irrelevant to the user.

A content-based filtering system selects items based on the correlation between the content of the items and the user's preferences as opposed to a collaborative filtering system that chooses items based on the correlation between people with similar preferences. PRES is a content-based filtering system. It makes recommendations by comparing a user profile with the content of each document in the collection. The content of a document can be represented with a set of terms. Terms are extracted from documents by running through a number of parsing steps. First all HTML tags and stop words (words that occur very often and cannot be used as discriminators) are removed. The remaining words are reduced to their stem by removing prefixes and suffixes [Porter 1980]. For instance the words "computer", "computers" and "computing" could all be reduced to "comput". The user profile is represented with the same terms and built up by analyzing the content of documents that the user found interesting. Which documents the user found interesting can be determined by using either explicit or implicit feedback. Explicit feedback requires the user to evaluate examined documents on a scale. In implicit feedback the user's interests are inferred by observing the user's actions, which is more convenient for the user but more difficult to implement.

There are several ways in which terms can be represented in order to be used as a basis for the learning component. A representation method that is often used is the vector space model. In the vector space model a document D is represented as an m-dimensional vector, where each dimension corresponds to a distinct term and m is the total number of terms used in the collection of documents. The document vector is written as, where w_i is the weight of term t_i that indicates its importance. If document D does not contain term t_i then weight w_i is zero. Term weights can be determined by using the *tf-idf* scheme. In this approach the terms are assigned a weight that is based on how often a term appears in a particular document and how frequently it occurs in the entire document collection:

$$w_i = tf_i \cdot \log\left(\frac{n}{df_i}\right)$$

where tf_i is the number of occurrences of term t_i in document D, n the total number of documents in the collection and df_i the number of documents in which term t_i appears at least once. The assumptions behind *tf-idf* are based on two characteristics of text documents. First, the more times a term appears in a document, the more relevant it is to the topic of the document. Second, the more times a term occurs in all documents in the collection, the more poorly it discriminates between documents.

In the vector space model user profiles can be represented just like documents by one or more profile vectors. The degree of similarity between a profile vector P, where $P = (u_1,...,u_k)$ can be determined by using the cosine measure:

$$sim(D,P) = \frac{D \cdot P}{\|D\| \cdot \|P\|} = \frac{\sum_k u_k \cdot w_k}{\sqrt{\sum_k u_k^2 \cdot \sum_k w_k^2}}$$

dot product

magnitude or norm (Euclidean norm)

A.B = ||A||.||B||.cos(theta)

3 PRES

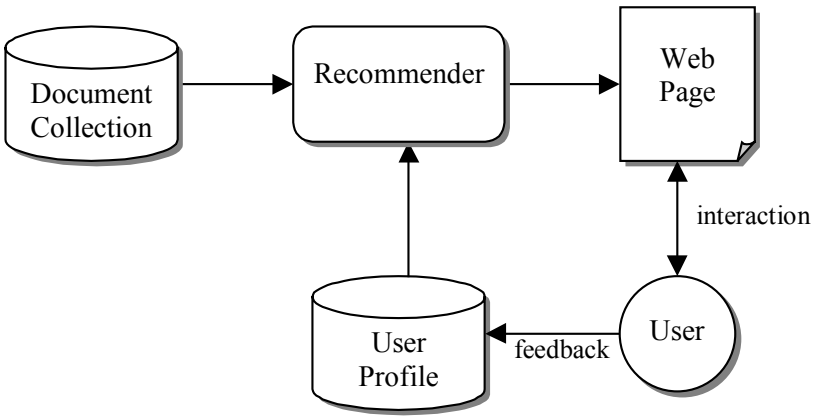

Figure 2 The PRES architecture

Figure 2 shows the PRES architecture. A user profile is learned from feedback provided by the user. The recommender system compares the user profile with the documents in the collection. The documents are then ranked on the basis of certain criteria such as similarity, novelty, proximity and relevancy and the best ranked documents appear as hyperlinks on the current web page.

3.1 Domain characteristics

As stated earlier, PRES recommends documents that consist of textual information about do it yourself home improvements. The most important aspect of this type of information is the fact that a certain topic is only interesting to a user for a short period of time. Once an improvement has been carried out or enough information has been provided the user will lose interest in that topic. As a consequence the user model that PRES learns has to be very dynamic. A domain in which users' interests change quickly also has implications for how the feedback is acquired. Explicit feedback is not a good choice for two reasons. Users will have to rate items frequently and will be very reluctant in doing so as their ratings expire quickly. Acquiring feedback by observing the users' actions is therefore favorable.

A user that selects and reads a document for a certain amount of time provides a strong indication that the document contains information a user is interested in. After such action the documents is therefore classified as a positive example. Finding negative examples is much more problematic. Users ignoring links to documents could be seen as a clue. This action does not provide strong evidence however as users might not have noticed the link or visit the document later. Another possible clue is a document that is being read for a very short time but this could be caused by the fact that the document is similar to what the user has already seen although the topic of the document is still interesting to the user. Classifying documents as negative based on weak assumptions leads to much noise in the training data and may result in inaccurate predictions. Negative examples will therefore not be used. In short, the user model has to be dynamic and learned from positive examples only.

only the positive examples used, because negative examples has problems such as:

3.2 Learning algorithm

There exist a variety of machine learning methods that can be used for learning a user model. PRES employs the relevance feedback method because it is both efficient and dynamic. Relevance feedback was introduced by Rocchio as an information retrieval utility that implements retrieval in several passes but it can also be applied to information filtering. Documents and profiles are represented as vectors in the vector space model. The term weights of a document are calculated using the standard *tf-idf* scheme. The profile consists of one vector that represents a topic of interest. In several information filtering systems that use relevance feedback the profile consist of more than one vector. For example, WebMate [Chen & Sycara 1997] a personal agent that helps users browse the web, uses clustering to maintain several profile vectors that each represent a different topic. The problem with using more than one vector is that it takes a while before a vector will represent a topic accurately enough. A document about a certain topic will therefore not always be assigned to a profile vector about the same topic. Because PRES operates in an environment in which the topic interest of users changes constantly only one vector is used. This vector is initially empty and is adjusted as the user navigates through the web site. When a user has spent a certain amount of time reading a document D the user profile P is updated with the following equation:

$$P' = \alpha P + \beta D$$

Note that if the document or the profile does not contain a term the corresponding weight is zero. Terms in the profile that have very low weights are removed after the profile vector has been updated. Weight β determines the relative importance of a document to the user. In information filtering systems that rely on explicit feedback β usually corresponds to the rating a user has given to a document and can either be a positive or a negative value. Systems that use implicit feedback may use different values for β for different kinds of feedback. In the recommender system Slider [Balabanovic 1998] for example, β is set to –3 when a user has deleted an article and to 0.5 when a users has read an article. Because PRES only determines whether or not a document is relevant, weight β is always set to 1. The profile vector is adjusted to a diminishing of the user's interests by α, a weight between 0 and 1 that reduces the term weights in the profile. This weight is determined via experimentation.

Besides the profile vector, a user profile also contains information about which pages the user has visited and the current page. All this information is necessary to generate recommendations that appear as hyperlinks on web pages. Several factors can be considered in determining which documents should be suggested to the user:

- The *similarity* between a document vector and the profile vector can be calculated using several similarity measurements.
- The *novelty* of a document is determined by the existence of information in a document that is new to the user.
- The *proximity* of a document is determined by the minimal number of links it takes to navigate from the current page to a page that presents the document. Mobasher et al. [Mobasher et al. 1999b] for example take the log of the number of links as a measurement of distance.
- Some recommender systems also check if a document is *relevant* to the information shown on the current page.

In PRES the similarity between the profile vector and a document is determined by using the cosine measurement. Documents that have already been visited by the user or already appear as hyperlinks on the current page are filtered out. The remaining documents are then sorted by rank. Two different strategies for determining which of these documents appear as hyperlinks were examined. In the first approach only a fixed number of top ranked documents are suggested. In the second approach documents are suggested if their score is above a relevance threshold [Yan & Garcia-Molina 1994].

3.3 Implementation

PRES has been largely written in the object-oriented programming language Java. Java is often used to make applets that run on client machines. Client-side Java has several limitations however. In order to run applets the client machine has to support Java. Although this is a standard feature in most browsers, many of them also have the option to disable applets. It also takes a considerable amount of time to load the applet on the client machine and for the applet to request or receive data from the server machine. PRES does not make use of applets but runs entirely on the server-side by using Java servlets and Java server pages (JSP). A servlet is a Java class that expands the functionality of a server. Because servlets operate solely within the domain of a server they do not have the limitations that applets impose. Java server pages are files that contain both HTML and embedded Java code. JSP files are compiled to servlets by a servlet engine that also runs the servlets. PRES uses the servlet engine of JRun that functions as a plug-in to an existing server. This means that the servlets run independently from the web server. The advantage of an add-on servlet engine is that if the server changes, the same servlet engine can still be used and hence no alterations have to made to the servlet code.

In many servers that support servlets a request for a JSP file is handled by a special servlet. Usually this servlet can be replaced by another servlet or by a sequence of servlets. The request is then sent to the first servlet in the chain while the response from the last servlet is sent back to the browser. In between the output of each servlet is passed to the next servlet. This technique is called servlet chaining and can be used to place a special servlet behind the servlet that normally handles JSP files so that it is called upon every time a user request a JSP file. This servlet carries out the relevance feedback method by updating the user profile with the terms of the JSP file. It also replaces a predefined tag that could be included in the JSP file with personalized recommendations.

The architecture of the PRES implementation is shown in figure 3. The database contains two different types of data: Information about users and information about web pages. The information about users is obtained online either from users themselves or by observing their behavior. The information about web pages is gathered offline, largely by the *parser* component which is run every time new pages are added to the web site. It analysis the collection of web pages and extracts terms and calculates their *tf-idf* weights which are then stored in the database. The *profiler* and *membership* component both react to user requests. The *membership* component enables the user to become a member by creating an account. Information that the users provide such as a unique user name and a password is stored in the database. A user needs to be a member in order to receive personalized recommendations. The

profiler component keeps track of the pages that members visit. It also carries out the relevance feedback method for every member. All this information is stored in the user profile which is initially empty for new members. The *recommender* component tries to find relevant web pages by comparing the user

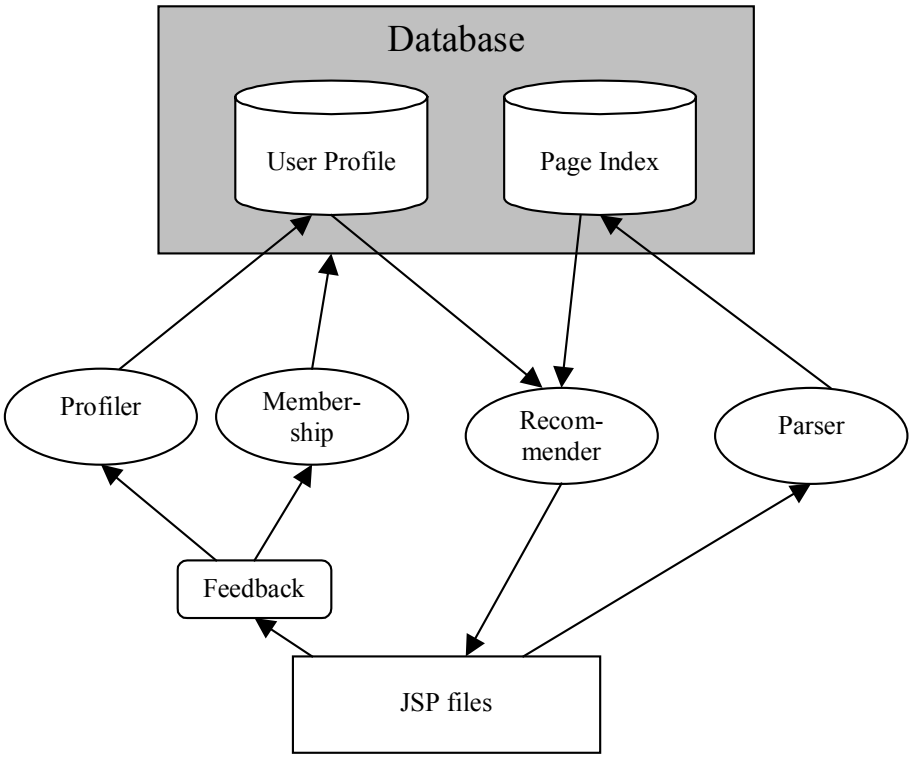

Figure 3 Architecture of the PRES implementation

profile with the pages that were indexed. Recommendations are presented to the user by inserting hyperlinks into the next JSP file that the user requests.

4 Performance

To evaluate PRES three fictitious users were created which all had different interests. Two topics about home improvement were assigned to the first user, three to the second user and four to the third user. For each topic all the relevant documents in the collection were selected and classified as relevant if it contained information that was associated with the topic. The percentage of documents that the users selected per topic also differed. The first user selected 80% of all the relevant documents about the two topics, the second user 50% and the third user 30%. The training sessions were then processed just like a normal user session would have been.

Figure 4 shows the results of this experiment. From left to right the precision (relevant retrieved/retrieved) and recall (relevant retrieved/relevant) at different points in time are shown of the first, second and third training session respectively. The precision ratios differ significantly for different topics of interest. This is probably caused by the fact that documents about several topics contain many different terms which makes it difficult for the learning method to find a relation between these documents. Precision ratios also differ at different points in time for the same topics of interest. As more documents about a certain topic are selected it becomes easier for a learning method to make

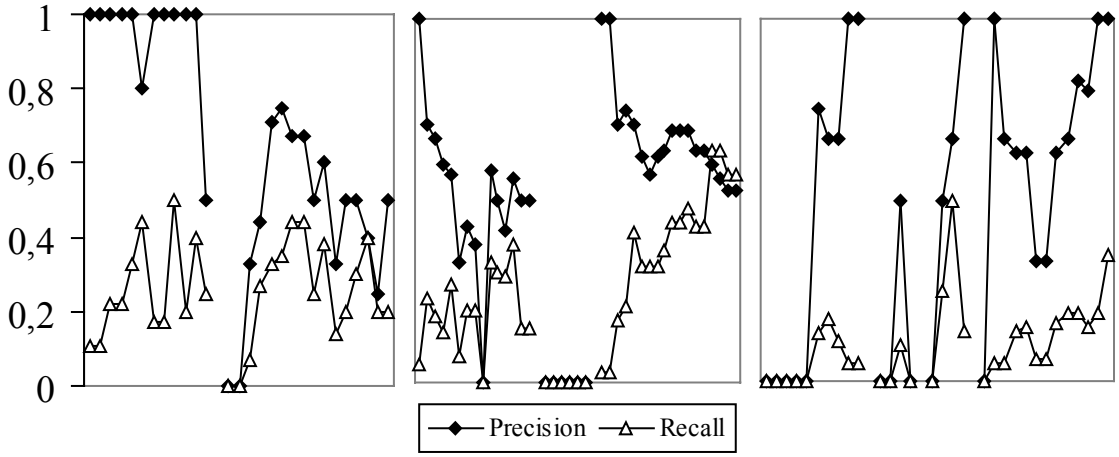

Figure 4 Precision and recall for three

recommendations as it is provided with more similar terms. On the other hand, as more documents are selected the number of remaining documents that are relevant decreases and it therefore also becomes more difficult for the learning method to find other relevant documents. This last factor will have little effect in the case of the third user session as only 30% of the total amount of documents about a topic are selected. In this case, precision usually increases overtime. The exact effects however remain difficult to measure as other factors, such as a page change, also have an influence on these ratios.

5 Related work

Many of the content-based filtering techniques that were described are borrowed from the field of information retrieval. Content-based filtering however, differs from information retrieval in the manner in which the interests of a user are represented. Instead of using a query an information filtering system tries to model the user's long term interests. There are several other systems that use content-based filtering to help users find information on the World Wide Web. Letizia [Lieberman 1995] is a user interface that assists users browsing the web. The system tracks the browsing behavior of a user and tries to anticipate what pages may be of interest to the user. Syskill & Webert [Pazzani et al. 1996] is a software agent that tries to determine which web pages might interest a user by using a naive Bayesian classifier. A user provides training instances by rating explored pages as either hot or cold. Jennings and Higuchi [Jennings & Higuchi 1992] describe a neural network that models the interests of a user in a Usenet news environment. The neural network is formed and modified as a result of the articles a user has read or rejected.

The work presented in this paper differs from these systems by focusing on single web sites. Information filtering systems that personalize web sites often use a collaborative approach to filtering. Amazon.com for example uses the GroupLens system [Resnick et al. 1994] to make recommendations about books and videos. Mobasher et al. [Mobasher et al. 1999b] describe several mining techniques for personalization. User access logs are examined to discover clusters of users that exhibit similar browsing behavior. These clusters can be used to predict the browsing behavior of individual users.

6 Conclusions

The test results seem to indicate that on average, slightly more than one out of two of the suggestions that PRES makes is relevant. The results are negatively influenced by the fact that the same concept can usually be described with several terms and many terms have more than one meaning. This makes the user profile less accurately, especially because the documents in the collection are relatively short and normally only a few documents about the same topic are selected by the user. Better results might be obtained by improving the vector space model. What content-based filtering systems cannot do however, is make predictions about the future interests of users. Collaborative filtering systems can make suggestions to a user that are outside the scope of previous selected items. The effectiveness of PRES can therefore be further improved if content-based and collaborative filtering would be combined.

The menu structure also has a great influence on the effectiveness of PRES. Because it is not useful to recommend items that already appear in the current menu, it becomes more difficult to make recommendations if the user has selected a menu that already contains most of the relevant items. The average precision of the recommendations that PRES makes drops about 15% if the menu is taken into account. So in general, the better the web site is organized, the harder it will be to make recommendations. Most web sites, especially larger ones, however can never be perfectly optimized for all users. Users have different interests and personalizing a web site could help them find information faster as they otherwise would have or wouldn't have found at all.

References

[Armstrong et al. 1995] Armstrong, R., Freitag, D., Joachims, T. and Mitchell, T. (1995). WebWatcher: A learning apprentice for the World Wide Web. In *Proceedings of AAAI Spring Symposium on Information Gathering*, Standford, CA.

[Atsumi 1997] Atsumi, M. (1997). Extraction of User's Interests from Web Pages based on Genetic Algorithm. *IPSJ SIG* Vol. 97 No. 51, pp. 13-18.

[Balabanovic 1998] Balabanovic, M. (1998). An Interface for Learning Multi-topic User Profiles from Implicit Feedback. *AAAI-98 Workshop on Recommender Systems*, Madison, Wisconsin.

[Balabanovic & Shoham 1995] Balabanovic, M. and Shoham, Y. (1995). Learning Information Retrieval Agents: Experiments with Automated Web Browsing. In *Working Notes of the AAAI Spring Symposium Series*, pp. 13-18. Stanford University.

[Chen & Sycara 1998] Chen, L. and Sycara, K. (1998). WebMate: A Personal Agent for Browsing and Searching. In *Proceedings of the 2nd International Conference on Autonomous Agents and Multi Agent Systems*, Minneapolis, MN.

[Jennings & Higuchi 1992] Jennings, A. and Higuchi, H. (1992). A Personal News Service based on a User Model Neural Network. *IEICE Transactions on Information and Systems*, Vol. E75-D, No. 2, pp. 198-209.

[Lieberman 1995] Lieberman, H. (1995). Letizia: An Agent that Assists Web Browsing. In *Proceedings of the 1995 International Joint Conference on Artificial Intelligence*. Montreal, Canada.

[Mobasher et al. 1999a] Mobasher, B., Cooley, R. and Srivastava, J. (1999). Data Preparation for Mining World Wide Web Browsing Patterns. *Journal of Knowledge and Information Systems*, Vol. 1, No. 1.

[Mobasher et al. 1999b] Mobasher, B., Cooley, R. and Srivastava, J. (1999). Automatic Personalization Based on Web Usage Mining. Technical Report TR99-010, Department of Computer Science, Depaul University.

[Nichols 1997] Nichols, D. M. (1997). Implicit Rating and Filtering. In *Proceedings of the Fifth DELOS Workshop on Filtering and Collaborative Filtering*, pp. 31-36. Budapest, Hungary.

[Nilsson 1996] Nilsson, N. J. (1996). *Introduction to Machine Learning*. Unpublished draft, Department of Computer Science, Stanford University.

[Pazzani et al. 1996] Pazzani, M., Muramatsu, J. and Billsus, D. (1996). Syskill & Webert: Identifying interesting web sites. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence AAAI 96*, pp. 54-61.

[Porter 1980] Porter, M. (1980). An Algorithm for Suffix Stripping. *Program*, 14(3), pp. 130-137.

[Resnick et al. 1994] Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P. and Riedl, J. (1994). GroupLens: An Open Architecture for Collaborative Filtering of Netnews. In *Proceedings of ACM 1994 Conference on Computer Supported Cooperative Work*, Chapel Hill, NC, pp. 175-186.

[Schwab & Pohl 1999] Schwab, I. and Pohl, W. (1999). Learning User Profiles from Positive Examples. In *Proceedings of the International Conference on Machine Learning & Applications*, pp. 15-20. Chania, Greece.

[Yan & Garcia-Molina 1994] Yan, T. W. and Garcia-Molina, H. (1994) Index Structures for Information Filtering Under the Vector Space Model. In *Proceedings of the International Conference on Data Engineering*.