RC22004 (98836) 19 March 2001 Computer Science

IBM Research Report

Personalization with Dynamic Profiler

Kun-Lung Wu, Charu C. Aggarwal, Philip S. Yu
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 704
Yorktown Heights, NY 10598

Personalization with Dynamic Profiler

Kun-Lung Wu, Charu C. Aggarwal and Philip S. Yu
IBM T.J. Watson Research Center
30 Saw Mill River Road
Hawthorne, NY 10532
{*klwu, charu, psyu*}*@us.ibm.com*

Abstract

Personalization of Web contents has been widely adopted. It provides users with a more customized experience of a Web site. In this paper, we describe a prototype system, called *Dynamic Profiler*, that generates dynamic user profiles for personalization. The system can be used in many personalized applications, including targeted advertising, product or content recommendations, and user community services. It uses content-based collaborative filtering techniques to create dynamic user profiles, form user communities and make recommendations. The system analyzes user logs, fetches the documents accessed and categorizes them. Each user is then described by a vector of document categories. Such user characterizations are then used to find user communities based on a projected clustering scheme. The log processing and content categorization are run periodically off-line to capture dynamic user profiles, which are then used online for personalized applications.

Keywords: Personalization, Projected Clustering, Supervised Clustering, Content-Based Collaborative Filtering, Dynamic Profiling.

1 Introduction

Personalization of Web contents has been widely popular. In particular, personalized contents are often provided to individual users on the portal pages, the first pages users normally see right after logging on to a Web site. For example, popular consumer Web sites, such as Yahoo! and Amazon.com, have various personalization features. Yahoo! allows a user to customize the contents as well as the presentation of the My Yahoo! page [20]. Amazon.com recommends product items based on a user's purchasing history. Most corporations and organizations also have personalized features on their corporate portal pages.

Personalization builds customer loyalty by establishing a one-to-one relationship between customers and a content provider. It can be deployed in the form of targeted advertisements, product or content recommendations, customized information delivery, or user community services. Personalization helps satisfy the goal of presenting the right information to the right people. In the face of vast amount of information and vast number of individual users on the Web, personalization is clearly an interesting and difficult problem.

In order to provide personalization, a content provider must understand the desires and behaviors of individual customers. This typically involves collecting information about the customers in some direct or indirect way and uses this information to develop personalization systems. Most Web sites may explicitly ask their customers to specify certain interests or preferences, which often reflect their desires of buying certain products or services. For example, customers may be asked to specify ratings (likes or dislikes) for certain products. These ratings are then used to make product recommendations. On the other hand, a Web site may implicitly collect the buying or browsing behaviors of its customers. However, there are privacy concerns in this type of implicit collection of user data, and these concerns must be treated with great care [28].

There are generally three types of personalization systems for the Web: rule-based systems, collaborative filtering systems and content-based systems [21, 5]. Rule-based systems, such as Broadvision (www.broadvision.com), allow Web site administrators to specify rules based on user demographics or static profiles (collected during registration), or other statistics. The rules are used to determine the contents served to individual users. Collaborative filtering systems, such as Firefly [26] and Net Perceptions (www.netperceptions.com), typically take explicit user information in the form of ratings or preferences, and, via the collaboration of like-minded users in a peer group, return information that is predicted to closely match the customers' preferences. Content-based systems usually rely on content similarity of Web documents to personal profiles obtained explicitly or implicitly from users.

In general, rule-based systems can become too complex to manage as the number of rules grows large. Content-based systems have the advantages of directness and simplicity, but they lack the sophistication of collaborative filtering systems which analyze the behaviors of peer groups for making recommendations. Collaborative filtering is generally effective, but it is usually difficult to explicitly collect a sufficient amount of user ratings because most users do not like to provide ratings to a Web site.

In this paper, we describe a prototype personalization system, called *Dynamic Profiler*, that uses content-based collaborative filtering techniques [5]. It does not require explicit user ratings; yet it uses the concept of peer groups for making recommendations. It combines the advantages of both content-based systems and collaborative filtering systems. The basic idea is to use the contents of pages accessed by the users of a Web site as a form of implicit ratings of document categories. The system analyzes user log, fetches documents accessed by the users, categorizes these documents based on a supervised clustering scheme [2]. Each user is then described by a vector of document categories. Such user characterizations are then used to find user communities (peer groups) based on a projected clustering scheme [1]. Each user community represents a peer group whose members share common interests based on the contents they have accessed on the Web site. The log processing and content categorization are run periodically off-line to capture the dynamic changes in user behaviors. These dynamic user profiles are then used online for personalized applications.

Note that traditional user profiles, such as demographics and preferences, are generally collected via a registration process. These profiles tend to be static, unless the users make specific changes. In contrast, the user characterizations as well as user communities derived by the Dynamic Profiler presented in this paper are dynamic. They change automatically if the behaviors of the users change.

There has been other related work on personalization [21, 3, 4, 18, 22, 11]. For example, an entire issue of *Communications of the ACM*, August 2000, was devoted to various issues of personalization, including the business of personalization, the human element, various issues and enabling technologies [23]. Many commercial systems, such as WebTrend (www.webtrend.com), and prototype tools have been developed for Web log analysis [29, 12, 24]. Similar to Dynamic Profiler, the WebPersonalizer system [21] proposed to analyze user log and create peer groups for personalization. However, WebPersonalizer characterizes user behaviors by the URIs from the log. In contrast, Dynamic Profiler fetches the documents, categorizes them and uses the content categories to describe each user. In general, content categories provide a better description of user behaviors than URIs because similar type of contents may exist in different pages.

The rest of the paper is organized as follows. Section 2 presents the architecture of Dynamic Profiler. Section 3 describes the details of a supervised clustering approach to content categorization and user characterization. Section 4 presents the implementation of identifying user peer groups based on a projected clustering approach. Section 5 then shows some online personalization applications using dynamic profiles produced by the off-line component. Finally, Section 6 summarizes the paper.

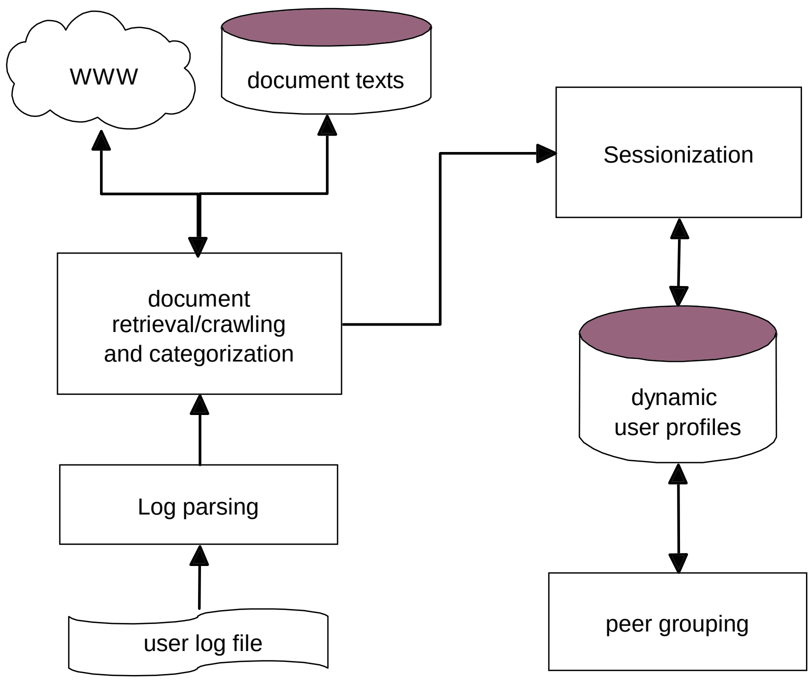

Figure 1: System architecture for off-line processing of user log for generating dynamic user profiles.

2 System architecture

Dynamic Profiler contains two major components (Java classes): an off-line component and an online component. The off-line component is a batch processing unit that runs periodically. It generates dynamic user profiles from user access log and stores them on database tables. The online component provides a few Java APIs for the personalization applications to access the dynamic user profiles. Fig. 1 shows a flow chart diagram containing the major modules in the off-line component, including log parsing, document retrieval (or crawling) and categorization, sessionization and peer grouping.

Log Parsing: This off-line component takes a log file as its input. Each log entry is parsed to extract important information, including user ID, request type, and document ID or URL. In order to personalize, the user ID must be available in the log to collect individual activities. The most important request type is "GET", although we also consider other requests, such as searches. We developed Dynamic Profiler for two different types of applications: a corporate portal and an ISP portal. For the corporate portal, document IDs are recorded as users retrieve documents from an internal database. For the ISP portal, URLs are recorded.

Document Retrieval and Categorization: After a document ID is identified, it is used to retrieve the document text from a database. In the case of a URL, a crawler is used to fetch the document page from the Web. This document text is then categorized into a few category names among a list of category clusters. The creation of the category clusters is based on a supervised clustering technique with training documents and an initial taxonomy. Each category cluster is defined by a vector of a few words (about 100 to 200 words) from a dictionary of about 76,000 words. Note that the construction of category clusters is done only once. These vectors are used for fast categorization of document texts during log processing. More details will be discussed in Section 3.

Sessionization: The category names for each document are then accumulated for each user session. Their occurrences are counted. At the end, each user session is represented as a list of (*category name, count*) pairs. These user categories from the current session are then combined with the previous ones from the database. The resulting top N (15 in our prototype) categories are then used to replace the old user categories in the database. In addition to user categories, other user statistics can be collected as dynamic user profiles. For example, we also collect the top M most frequently accessed documents for each user session. In order to capture the changes in behaviors, an aging mechanism is needed to allow a new set of categories to emerge. Periodically, the category occurrence counts for a user are reduced by a similar amount. This is equivalent to exponential averaging smoothing.

Peer Grouping: Once all the log entries are processed and user categories are written into the database, the peer grouping module is activated. It partitions all users into a number of peer groups based on a projected clustering technique (more details in Section 4). Here each user is described by a vector of categories and their associated weights. Members of a group represent a user community where they share certain common interests according to their categories. These user communities are stored in the database as part of the dynamic user profiles. Note that a user may change from one community to another, but he/she can only be in one group.

The online component of Dynamic Profiler contains a set of Java APIs that allow personalization applications to access the dynamic profiles generated by the off-line component. For example, our prototype includes three interesting Java methods. The first one returns a list of user IDs who are the community members of a given user (`getCommunityMembers(String user)`). The second one returns a list of category names for a given user (`getUserCategories(String user)`). The third one returns a list of top documents accessed by a given user (`getTopDocuments(String user)`).

3 Categorizing contents

In this section, we will discuss the techniques which we used for the categorization of the individual text documents. Several text classifiers have recently been proposed, such as those discussed in [7, 9, 10, 17, 19]. These classifiers have shown excellent results on document collections such as the *Reuters* dataset or the US patent database [9] and to a somewhat lesser extent on the web with the Yahoo! taxonomy. However, categorization of web documents has proven to be especially difficult because of the widely varying style, authorship and vocabulary in different documents.

Most of the above-mentioned categorizations are created manually by subject experts. There is apparent inaccuracy of classification methods on large document collections. It is a result of the fact that a large heterogeneous collection of manually categorized documents is usually a poor fit for any given classification model.

Our system differs significantly. It uses a semi-supervised model which can avoid many of the difficulties caused by manually-defined categories. It is generally useful to construct categorization systems which relax the restriction imposed by pre-defined sets of classes (categories). Here, we apply clustering on a set of initially-available documents and the associated document taxonomy in order to create the desired categories in a new taxonomy. The initially-available document taxonomy can provide sufficient supervision in creating a set of categories which can handle similar subjects as the original, but with some freedom in choosing exactly how to define and create the desired categories. Once such a set of categories has been obtained, it is easy to perform the categorization of a given text document by using the same distance measures as were used to perform the supervised clustering. As long as the final application of the categorization system does not restrict us to the use of a fixed set of class labels, this supervised approach may provide considerable advantage because of the tight integration of the measures which are used for clustering and classification.

The fact that we actually know the model used to construct each partition in the supervised clustering ensures that we can theoretically obtain a perfect accuracy on this categorization. Therefore the quality of categorization depends completely on the quality and coherence of each cluster in the new taxonomy, rather than the accuracy of a training procedure on the original taxonomy. Thus, if the supervised clustering procedure can create a new set of classes which are qualitatively comparable to the original taxonomy (in terms of human perception and judgment), the accuracy of the overall categorization system is substantially improved.

The use of clustering for providing browsing capabilities has been espoused in earlier work by Cutting et al. [14, 13]. Other work on clustering algorithms for text data may be found in [6, 8, 15, 25, 27, 30]. These methods do not use any kind of supervision from a pre-existing set of classes, and are attractive for creation of a small number of clusters such as fifty or so, though the clustering rapidly degrades in quality when there is a need to find more fine-grained partitions. Typically, when categories are related sufficiently such that some documents can be considered to be related to both, unsupervised clustering methods are unable to create distinct sets of classes for such categories. The use of a pre-existing manual categorization helps in the creation of a new set of clusters, so that we have some control over the range of subjects that we would like the categorization system to address. The resulting set of clusters may contain additional, new or similar, classes to the original taxonomy, and may be quite different in terms of the distribution of the documents among the different classes.

3.1 Document representation

In order to represent the documents, we used the vector space model. In the vector space model, it is assumed that each document can be represented as as *term vector* of the form $\overline{a} = (a_1, a_2, \ldots a_n)$. Each of the terms a_i has a weight w_i associated with it, where w_i denotes the normalized frequency of the word in the vector space. A well-known normalization technique is the cosine normalization. In cosine normalization, the weight w_i of the term i is computed as follows:

$$w_i = \frac{tf_i \cdot idf_i}{\sqrt{\sum_{i=1}^{n} (tf_i \cdot idf_i)^2}} \tag{1}$$

Here the value of tf_i denotes the *term frequency* of a_i, whereas the value of idf_i denotes the *inverse document frequency*. The inverse document frequency is the inverse of the number of documents in which a word is present in the training data set. Thus, less weight is given to words which occur in larger number of documents, ensuring that the commonly-occurring words are not given undue importance.

The similarity between two documents may be measured by calculating the *cosine* similarity between the documents. The cosine similarity between two documents with weight vectors $U = (u_1 \ldots u_n)$ and $V = (v_1 \ldots v_n)$ is given by:

$$cosine(U, V) = \frac{\sum_{i=1}^{n} f(u_i) \cdot f(v_i)}{\sqrt{\sum_{i=1}^{n} f(u_i)^2} \cdot \sqrt{\sum_{i=1}^{n} f(v_i)^2}} \tag{2}$$

Here $f(\cdot)$ is a *damping function* such as the square root or the logarithmic function.

A *centroid* of a set of documents is defined by a concatenation of the documents in the set. Thus a centroid of a set of documents is a meta document which contains all the terms in that set with the appropriate term frequencies added. A *damped centroid* (or pseudo-centroid) of a set of documents is defined in the same way as the centroid, except that in this case a damping function is applied to the frequencies of the terms in each document before adding them together. The damping function ensures that the repeated presence of a word in a single document does not affect the pseudo-centroid of the entire cluster excessively. Thus, the pseudo-centroid is often a much more stable representation of a central point in a cluster of documents as compared to the centroid.

A projection of a document is defined by setting the term frequencies (or weights) of some of the terms in the vector representation of the document to zero. These are the terms which are said to be *projected out*. We will use the process of projection frequently in the course of the supervised clustering algorithm. Each cluster is represented by a seed vector containing only a certain maximum number of projected words. The aim in projection is to isolate a relatively small vocabulary which describes the subject matter of a cluster well, while filtering out the non-relevant features for that class. We use an incremental process of gradually finding the best set of projected words, while simultaneously refining the clusters, so as to gradually converge to an optimum feature set for each cluster.

Our first phase was to perform the feature selection in such a way so that only the more differentiating words are used in order to perform the clustering. Note that in unsupervised clustering methods, where a pre-existing taxonomy is not used, the feature selection is somewhat rudimentary in which only *stop words* (very commonly-occurring words in the English language) are removed. In this case, since more information is available, we use it in order to prune the feature set further, and bias the clustering process to use words which are discriminatory with respect to the original class labels. We use a number called the *normalized gini index* of a word in order to calculate its importance in the clustering process.

Let there be K classes $C_1, C_2, \ldots, C_K$ at the lowest level in the original taxonomy. Let $f_1, f_2, \ldots, f_K$ be the number of occurrences of a particular word in each of the K classes, and let $n_1, n_2, \ldots, n_K$ be total word count for the documents in each of the K classes. Thus, the *fractional presence* of a word in a particular class is given by f_i/n_i. We define the *skew fraction* of a word for class i by $\frac{f_i/n_i}{\sum_{i=1}^{K} f_i/n_i}$. We shall denote this skew fraction by p_i. Note that if the word is very noisy, and is very evenly distributed among the different classes, then the skew fraction for the word is likely to be approximately $1/K$ for many classes.

The normalized gini index of a word with skew fractions $p_1 \ldots p_K$ is given by $1 - \sqrt{\sum_{i=1}^{K} p_i^2}$. If the word is distributed evenly across the different classes, then the gini index is $1 - 1/\sqrt{K}$. This is the maximum possible value of the gini index. On the other hand, when the word is highly correlated with particular categories and is very skewed in its distribution, then the normalized gini index is much lower.

For our feature selection phase, we calculated the normalized gini index of each word in the lexicon in order to calculate its significance to the lexicon. All those words whose gini index was higher than a predefined value were removed from contention. Thus, the removal of these words ensures the use of a much better set of features than the simple stop-word removal of unsupervised clustering techniques. In subsequent phases of clustering and categorization, only the reduced feature set was used for all analysis.

3.2 Supervised clustering

The clustering algorithm uses a seed-based technique in order to create the clusters. Traditional clustering methods have often used seed-based algorithms in order to serve as an anchor point for the creation of the clusters. In other words, seeds form an implicit representation of the cluster partitioning in which each item to be categorized is assigned to its closest seed based on some distance (or similarity) measure. In the context of information retrieval, a seed is a meta-document which can be considered as a pseudo-representation of a central point in a given cluster. Most of the current clustering algorithms discussed in [6, 14, 13, 15] are based on finding a set of seeds in order to define the implicit partitions.

Since the focus of the algorithm is on *supervised clustering*, we started off with a set of seeds which are representative of the classes in the original taxonomy. These representative seeds are constructed by finding the damped centroids (or pseudo-centroids) of the corresponding classes. This choice of starting point (and features picked) ensures the inclusion of supervision information from the old taxonomy, but the subsequent clustering process is independent of any further supervision. Providing this level of independence is critical in the construction of a much more refined set of classes, which are based purely upon content. One of the aspects of the algorithm is that it projects out some of the words in order to represent the seeds. Thus, each seed consists of a vector in which the number of words with a non-zero weight is restricted to a predefined maximum. This vector of words is indicative of the subject material which is most relevant to that cluster. The algorithm starts with a projected dimensionality of about 500 words, and gradually reduces it in each iteration as the clusters get more refined, and a smaller number of words are required in order to isolate the subject of the documents in that cluster. This technique of representing clusters by using both the documents and the projected dimensions in order to represent a cluster is referred to as *projected clustering*, and is an effective technique for the creation of clusters for very high dimensional data. Thus, the projected clustering technique merges the problem of finding the best set of documents and features for a cluster into one framework. More details on the advantages of using projected clustering for very high dimensional data may be found in [1].

The definition of each cluster ensures that it is possible to categorize any test document very easily by assigning it to the class for which the corresponding seed is the closest in the projected feature space. As in the case of the clustering, the cosine measure is used in order to perform the classification.

3.3 Distinguishing closely-related subjects

An important feature which we added to our categorization process was a method for distinguishing between very closely related subjects. This is required because even a supervised clustering technique may not provide perfect subject isolation, and a small percentage of the documents do get clustered with documents from a closely related (though slightly inaccurate) category. Even though a theoretical accuracy of 100% can be obtained by reporting the cluster label for the most similar seed, it may sometimes be desirable to correct for the errors in the clustering process by using a context-sensitive comparison method.

We build a *domination matrix* on a subset of the universe of categories, such that we know that all of these categories are good candidates for being the best match. As we will see, the simplicity of this process ensures that speed is not compromised by the use of the flat organization of clusters.

The first step in the algorithm is to find the k closest cluster seeds to the test document. The similarity of each cluster to the test document is calculated by using the cosine measure of the test document to the seed corresponding to each cluster. The value of k is a user-chosen parameter, and is typically a small number compared to the total number of nodes in the taxonomy. These k categories are the candidates for the best match, and may often contain a set of closely related subjects. This ranking process is designed to re-rank these categories more appropriately.

In order to understand the importance of distinguishing among closely related subjects, let us consider the seeds for two nodes in the taxonomy: **Business Schools** and **Law Schools**. Recall that our process of projection limits the number of words in each seed to only words which are

```
Algorithm Classify(TestDocument: T, DominationThreshold: DomThresh)
begin
  Use cosine measure to find the $k$ closest seeds $\{S_1 \ldots S_k\}$
                    to the test document $T$;
  for $i = 1$ to $k$ $domination[i] = 0$;
  for $i = 1$ to $k$ do
    for $j = (i + 1)$ to $k$ do
    begin
      if $(cosine(S_i, T) > cosine(S_j, T) + DomThresh)$ then
      $domination[i] = domination[i] + 1$;
      else if $(cosine(S_j, T) > cosine(S_i, T) + DomThresh)$ then
      $domination[j] = domination[j] + 1$;
      else if $(cosine(S_i - S_j, T) > cosine(S_j - S_i, T))$ then
      $domination[i] = domination[i] + 1$;
      else
      $domination[j] = domination[j] + 1$;
      end
    end
  Rank order the $k$ categorizations in decreasing order of $domination[i]$;
end
```

Figure 2: The classification algorithm

relevant to the corresponding categories. Some examples of words (with non-zero weights) which could be represented in the seed vector of each of these categories are as follows:

(1) **Business Schools:** business (35) , management (31), school (22), university (11), campus (15), presentation(12), student(17), market(11), operations(10)....

(2) **Law Schools:** law(22), university (11), school (13), examination (15), justice (17), campus (10), courts (15), prosecutor (22), student (15) ...

A document in the generic category of schools is likely to contain all of the words such as university and school. Thus both these categories may be among the k closest seeds for the document. In order to establish the relative closeness of two categories to a given document more accurately, we need to ignore the contributions of the words common to both categories to the cosine measure. In other words, we need to compare the closeness based on the words which are *not* common in the seed vector of both categories. This is done by performing a *relative seed subtraction* operation on the seed vectors of each of the categories. The seed subtraction operation is defined as follows: Let S_1 and S_2 be two seed vectors. Then, the seed $S_1 - S_2$ is obtained by taking the seed S_1 and setting the weight of all those words which are common to S_1 and S_2 to 0.

We say that the seed S_1 dominates the seed S_2 under the following conditions:

• The (cosine) similarity of S_1 to the test document T is larger than the similarity of S_2 to T by at least a predefined threshold referred to as the *domination threshold.*

• The (cosine) similarity of S_1 to T is not larger than the similarity of S_2 to T by the predefined threshold, but the similarity of $(S_1 - S_2)$ to T is larger than the similarity of $(S_2 - S_1)$ to T.

The use of a domination threshold ensures that it is only possible to reorder seeds whose similarity to the test document are very close together. This is because it is primarily in these cases that the differences in the contributions of the common words tends to be a result of noise, rather than any actual pattern of difference in the frequencies of the (common) words in the seeds for the two categories. For each pair of the closest k seeds to the test document, we compute the domination matrix, which is the pairwise domination of each seed over the other. In order to rank order the k candidate seeds, we compute the *domination number* of each seed. The *domination number* of a seed is equal to the number of seeds (among the remaining $(k-1)$ seeds) that it dominates. The k seeds are ranked in closeness based on their domination number; ties are broken in favor of the original ordering based on cosine measure. The algorithm for returning the ranked set of k categorizations is illustrated in Figure 2.

It is obvious that the best matching category is more likely to be contained among the top k categories based on cosine measure, than only the closest category based on this measure. (If the clustering is perfect then it suffices to use $k = 1$.) The re-ranking process is then expected to rank this category highly among the k choices. If there are a total of K classes created by the clustering algorithm, then the categorization algorithm needs to perform $O(K+k^2)$ cosine similarity calculations. Further, since the projected dimensionality of each seed is restricted to a few hundred words, each similarity calculation can be implemented efficiently. Thus, the categorization system is extremely fast because of its simplicity, and scales almost linearly with the number of classes. This feature is critical for its use in performing automated categorization of large libraries of documents.

4 Identifying user communities

The categorization process helps in the identification of the categories of the documents which the user has browsed. These can be utilized further in order to create the user communities. These are communities containing users that are very similar to one another in terms of the categories of documents that they access. Each user is described by a list of about 15 (*category name, count*) pairs. Here the count represents the frequency with which a user identifies a web page of a particular category.

Since the description of a user can be directly represented by a vector-space in the real domain, it is possible to cluster them together in order to create groups of users that are very similar to one another. The clustering technique discussed in the previous section applies only to the case of text documents with supervision and cannot really be used for this case because there is no supervision. Furthermore, since high dimensional data is sparse, classical clustering techniques do not work very effectively because of the effects of the dimensionality curse. In order to cluster these users into groups of closely related users, we use projected clustering techniques [1].

The idea behind projected clustering techniques is to identify each user-community with a "projection" of the kind of categories that they are most interested in. To this effect, each user community is identified by a vector of the categories that they are most interested in together with a weight which is indicative of their level of interest. For example, a community of users that is very interested in sporting events may have a vector which appears as follows:

sports (50), NHL (25), NBA (42), Basketball (31)....

Note that this vector is created as a result of the fact that the projected clustering algorithm is able to mine the dimensions which are most closely related to the interests of the users which belong to the cluster. This is also quite useful in many ways, since it provides an intuitive idea of what a user-community is most interested in. More details of a projected clustering algorithm may be found in [1]. This method uses a variation of K-medoid methods [16], which requires many passes over the database. The version used for this system is somewhat different since it uses K-means methods which are significantly more efficient in terms of the number of passes over the database.

5 Applications

The results of user categorization as well as peer grouping are stored in database tables. As an example, Table 1 shows 15 category names and their associated counts for user Tom. Based on the log, Tom has been accessing documents most closely related to "Current Events News and Media". This is an example from a sample run we have conducted. In this sample run, we created a general taxonomy containing 1,167 categories. Note that there are very closely related category names in Table 1, indicating our categorization algorithm can make distinction among very closely related documents. For instance, "Current Events News and Media", "U.S. Newspapers", "News (Commercial)", and "Daily News and Media" are closely related categories. "Computer Workstation Products and Services" and "Computer Workstations – IBM" are another example of closely related categories.

Table 2 shows a peer group to which Tom belongs. The group has eight members. These eight community members share certain common interests (see Table 3), i.e., common category names in their respective user category tables. The size of each peer group varies. In our implementation, the total number of peer groups is pre-determined. However, one user can belong to only one peer group. After each batch processing, the peer group table is completely replaced. Thus, a user may change from one group to another as his/her behavior changes.

Various personalized applications can use the tables created by the off-line component of Dynamic Profiler. Here, we describe three examples: targeted advertising, content recommendations based on community members, and personalized search.

Targeted Advertising: Table 4 shows a list of advertisements that need to be placed. With the user categorization shown in Table 1, we can find the targeted advertisements for user Tom. This can be done by first categorizing each of the advertisements in Table 4 and then simply find the ones that most closely match Tom's categories. As a result, the top 5 advertisements to be targeted for

Table 1: A user category table.

User ID	Category	Count
Tom	Current Events News and Media	63
Tom	Misc. Online Shopping Malls (Commercial)	37
Tom	Indices to World Wide Web	37
Tom	World Wide Web Search Engines	29
Tom	U.S. Newspapers	28
Tom	News (Commercial)	27
Tom	Daily News and Media	26
Tom	Computer Workstation Products and Services	25
Tom	Computer Workstations – IBM	21
Tom	National Elections	18
Tom	Misc. Baseball	17
Tom	Sports News and Media	16
Tom	Misc. Meteorology	16
Tom	U.S. States Meteorology	14
Tom	Misc. Aviation (Commercial)	7

Table 2: A peer group table.

User ID	Group ID
Tom	3
Eric	3
John	3
Mary	3
Peter	3
Allice	3
Vicky	3
Andrew	3

Table 3: Top categories common among Peer Group 3.

Category
Current Events News and Media
Misc. Online Shopping Malls (Commercial)
Sports News and Media
World Wide Web Search Engines
Misc. Baseball

Table 4: A list of advertisements to be placed.

Advertisement Banner
www.ibm.com
www.cnn.com
www.cnnfn.com
www.latimes.com
cbs.marketwatch.com
www.zdnet.com
www.nba.com
realguide.real.com
www.yahoo.com
www.drkoop.com
www.weather.com

Tom are www.yahoo.com, www.cnn.com, www.latimes.com, www.ibm.com, and www.weather.com.

Content Recommendations: The above targeted advertisements use only Tom's interests without the help of his peers. However, Tom's peers can help him in content recommendations. For example, we can recommend documents that are commonly accessed by Tom's peers but not Tom himself. This can be accomplished by (1) identifying all of Tom's peers by calling `getCommunityMembers(Tom)`; (2) for each of Tom's peers, finding his/her top documents by calling `getTopDocuments()`; (3) aggregating all top documents from all of Tom's peers; (4) removing Tom's own top documents from the aggregated top documents.

Personalized Search: Search is provided in almost every corporate portal. The categories stored in the user category table can be used to either narrow the search space or filter the search results. With personalized search, the search engine can compose a more specific search query, rank search results and present a more relevant search results to the user.

6 Summary

In this paper, we have described the design of a personalization system, called Dynamic Profiler. It provides personalization based on a content-based collaborative filtering approach. It analyzes user log, fetch document texts, categorizes documents and describes each user with a vector of document categories. The content categorization system is based on a supervised clustering approach. It can distinguish closely related subjects. User categories are then used to do peer grouping using a projected clustering method. The Dynamic Profiler prototype has been completed. It is currently being further developed for personalized applications for a corporate portal.

Acknowledgment

We would like to thank Tao Zhao of New York University for helping develop part of the prototype system when he was a summer student at T. J. Watson Research Center.

References

[1] C. C. Aggarwal et al. Fast algorithms for projected clustering. In *Proc. of the ACM SIGMOD Conference*, pages 61–72, 1999.

[2] C. C. Aggarwal, S. C. Gates, and P. S. Yu. On the merits of building categorization systems by supervised clustering. In *Proc. of the ACM SIGKDD Conference*, pages 352–356, 1999.

[3] C. C. Aggarwal, J. L. Wolf, K.-L. Wu, and P. S. Yu. Horting hatches an egg: Fast algorithms for collaborative filtering. In *Proc. of the ACM SIGKDD Conference*, 1999.

[4] C. C. Aggarwal, J. L. Wolf, K.-L. Wu, and P. S. Yu. The intelligent recommendation analyzer. In *Proc. of Workshop on Knowledge Discovery and Data Mining on the WWW*, 2000.

[5] C. C. Aggarwal and P. S. Yu. Data mining techniques for personalization. *Data Engineering Bulletin*, 23(1):4–9, Mar. 2000.

[6] P. Anick and S. Vaithyanathan. Exploiting clustering and phrases for context-based information retrieval. In *Proc. of the ACM SIGIR Conference*, pages 314–322, 1997.

[7] C. Apte, F. Damerau, and S. M. Weiss. Automated learning of decision rules for text categorization. Technical Report RC 18879, IBM T. J. Watson Research Center, 1993.

[8] L. D. Baker and A. K. McCallum. Distributional clustering of words for text classification. In *Proc. of the ACM SIGIR Conference*, pages 96–103, 1998.

[9] S. Chakrabarti, B. Dom, R. Agrawal, and P. Raghavan. Scalable feature selection, classification and signature generation for organizing text databases into hierarchical topic taxonomies. *VLDB Journal*, 7:163–178, 1998.

[10] S. Chakrabarti, B. Dom, and P. Indyk. Enhanced hypertext categorization using hyperlinks. In *Proc. of the ACM SIGMOD Conference*, pages 307–318, 1998.

[11] I. Cingil, A. Dogac, and A. Azgin. A broader approach to personalization. *Communications of the ACM*, 43(8):136–141, Aug. 2000.

[12] R. Cooley, B. Mobasher, and J. Srivastava. Data preparation for mining World Wide Web browsing patterns. *Journal of Knowledge and Information Systems*, 1(1), 1999.

[13] D. R. Cutting, D. R. Karger, and J. O. Pedersen. Constant interaction-time scatter/gather browsing of very large document collections. In *Proc. of the ACM SIGIR Conference*, pages 126–134, 1993.

[14] D. R. Cutting, D. R. Karger, J. O. Pedersen, and J. W. Tukey. Scatter/gather: A cluster-based approach to browsing large document collections. In *Pro. of the ACM SIGIR Conference*, pages 318–329, 1992.

[15] M. A. Hearst and J. O. Pedersen. Re-examining the cluster hypothesis: Scatter/gather on retrieval results. In *Proc. of the ACM SIGIR Conference*, pages 76–84, 1996.

[16] A. Jain and R. Dubes. *Algorithms for Clustering Data.* Prentice Hall, 1998.

[17] D. Koller and M. Sahami. Hierarchically classifying documents using very few words. In *Int. Conference on Machine Learning.* Morgan-Kaufmann, 1997.

[18] J. A. Konstan et al. Applying collaborative filtering to USENET news. *Communications of the ACM*, 40(3):77–87, Mar. 1997.

[19] W. Lam and C. Y. Ho. Using a generalized instance set for automatic text categorization. In *Proc. of the ACM SIGIR Conference*, pages 81–88, 1998.

[20] U. Manber, A. Patel, and J. Robinson. Experience with personalization on Yahoo! *Communications of the ACM*, 43(8):35–39, Aug. 2000.

[21] B. Mobasher, R. Cooley, and J. Srivastava. Automatic personalization based on Web usage mining. *Communications of the ACM*, 43(8):142–151, Aug. 2000.

[22] N. Ramakrishnan. PIPE: Web personalization by partial evaluation. *IEEE Internet Computing*, pages 21–31, Nov.-Dec. 2000.

[23] D. Riecken. Personalized views of personalization. *Communications of the ACM*, 43(8):27–28, Aug. 2000.

[24] S. Schechter, M. Krishnan, and M. D. Smith. Using path profiles to predict HTTP requests. In *Proc. of the 7th World Wide Web Conference*, 1998.

[25] H. Schutze and C. Silverstein. Projections for efficient document clustering. In *Proc. of the ACM SIGIR Conference*, pages 74–81, 1997.

[26] U. Shardanand and P. Maes. Social information filtering: Algorithms for automating "word of mouth". In *Proc. of the ACM CHI Conference*, pages 210–217, 1995.

[27] C. Silverstein and J. O. Pedersen. Almost-constant time clustering of arbitrary corpus sets. In *Proc. of the ACM SIGIR Conference*, pages 60–66, 1997.

[28] E. Volokh. Personalization and privacy. *Communications of the ACM*, 43(8):84–88, Aug. 2000.

[29] K.-L. Wu, P. S. Yu, and A. Ballman. SpeedTracer: A Web usage mining and analysis tool. *IBM Systems Journal*, 37(1):89–105, 1998.

[30] O. Zamir and O. Etzioni. Web document clustering: A feasibility demonstration. In *Proc. of the ACM SIGIR Conference*, pages 46–53, 1998.