6-1-2001

A System for Audio Personalization with Applications on Wireless Devices

David Marmaros
Dartmouth College

Dartmouth College Computer Science Technical Report TR2001-398

A System for Audio Personalization with Applications on Wireless Devices

David Marmaros

Senior Honors Thesis

Department of Computer Science

Dartmouth College

Advisor: Professor Daniela Rus

Co-Advisor: Professor David Kotz[1]

June 1, 2001

Abstract

We present and analyze a system for dynamically tailoring discrete audio content for numerous users based on aggregate data and intuitive feedback mechanisms. The framework for this system utilizes a flexible client-server architecture to facilitate audio dissemination, with particular attention to distribution over wireless networks. We discuss the requirements and specifications of such a system. We further analyze the algorithms and protocols required for its operation. Finally, we outline and provide data from a demonstration of this application.

[1] Email: {marmaros,rus,dfk}@cs.dartmouth.edu

Table of Contents

1. Introduction

The Internet revolution of the late 1990s brought with it a glut of media. As of May 2001, Google,[2] a popular text search engine, reports to have indexed 1,346,966,000 pages. With more web pages than humans on this continent, it is no wonder that it is difficult to find relevant information. Many companies have devised search engines to help find textual content [Google, Yahoo,[3] AltaVista,[4] and so forth]. Unfortunately, there are few comparable tools to aid in finding relevant audio information. Even more importantly, traditional audio-based media such as news, weather, and sports-scores are extremely time and location dependent.

It would be a boon for society to allow each person to have information that they need when they need it. Ideally, this information would be provided before the person realized that they needed it and without significant input. It is our goal to lay the groundwork for such a system, and discuss its components, uses, and extensions.

We envision a system for both work and play. Imagine a busy couple waking up in the morning. Within half an hour, they are awake, dressed, and headed off to their respective workplaces. Before heading out, each grabs hold of a small device that has been resting overnight in small stands. The wife is fortunate to live close to work, but on the way she plugs her device into the car stereo, and can listen to up to date news, weather, and relevant traffic reports immediately. The device has tailored the information to her tastes and downloaded the audio during the night.

The husband has a longer commute. He also listens to news, but as he travels, a GPS unit in his device finds pertinent location based information such as traffic jams and detours, and informs him in a timely manner. If he is running low on gas, then he can ask the device to search for the closest station with the lowest price, and it will tell him where to go. Having listened to his desired news at the beginning of the trip, the device chooses music to play. Based on his feedback, the device will learn and improve its choices.

---

[2] http://www.google.com

[3] http://www.yahoo.com

[4] http://www.altavista.com

Upon arriving at work, each can attach the device to his computer using a small stand. The device will download a new batch of content. Using predictive technology, a constant connection to the Internet is not required. The device can store a days worth of audio with extra space to spare. As such, it can predict what audio might be required and store content accordingly. The device proceeds to upload rating, listening, and GPS data to aid in improving the choices in the future.

The system designed within this project takes the first steps in realizing this vision. The system functions as a client-server model. The server is capable of storing extensive details about the users and content. It is able to quickly process this data to formulate recommendations for any particular user. It passes these recommendations to the client. The framework deliberately makes as few assumptions about the client as possible. The system functions correctly and efficiently whether the client is a mobile wireless device, or if it has a fixed device with a permanent real-time Internet access.

The remainder of this paper is structured as follows. Section 2 outlines existing research within academic forum and applications in the burgeoning Internet industry. Section 3 discusses the framework of our audio recommendation system. Section 4 describes and analyzes algorithms and protocols used within this project. It proceeds to discuss extensions to the core algorithms, including wireless considerations. Section 5 describes the technical and logistical details of the demonstration. It also discusses results, metrics, and conclusions related thereto. Section 6 proposes future extension and applications of this technology. This section discusses resolutions to problems proposed in the earlier sections, and a brief analysis of the financial viability of this technology. Finally, Section 7 concludes the paper.

2. Related Work

The academic and commercial forums have been working extensively to develop the underlying technologies in these devices. This section explores the related research found in institutional studies, and discusses the similar systems that have been brought to market.

2.1 Papers and Studies

This paper binds together research from three different fields:
1) Personalization: zero-knowledge and metadata based algorithms
2) Audio: distribution, content management, and deriving metadata
3) Wireless technologies

The literature is rich with papers that span several of these fields. For example, Alghoniemy and Tewtik [AT00] propose algorithms that personalize and devise an optimal order for content. These algorithms operate under the assumption that metadata describing the content is available in advance. Khan and McLeod [KM00] review personalization relating to audio data specifically, and focus on the process of deriving content metadata using speech recognition. Furthermore, the metadata is stored ontologically for ease of conducting specific queries. This representation facilitates accurate searching of the corpus.

Our work differs from these previous works in three ways. First, our proposed system is not a search engine. It does not query the corpus in response to a specific question. On the contrary, its sole task is to choose content rapidly and accurately from the corpus that the user will enjoy. Second, the proposed system does not assume that there is any metadata describing the content. It is capable of classifying a content item and determining which users to recommend it to, relying solely on user feedback. Despite this ability, it is still be able to use any available content metadata to form optimal recommendations. Third, we examine the personalization algorithm in the context of a complete content distribution and management system.

The academic forum has conducted considerable research in the above three fields in isolation. This paper builds on several of these.

Clustering Algorithms form the basis of the personalization algorithm presented in this paper. Extensive research demonstrates that clustering has applications that span numerous unrelated fields [HS86, AB84]. Jain and Dubes [JD88] provides a comprehensive general overview, while a description of Information Retrieval (IR) related applications are can be found in [Wil88]. These uses of the technique are a result of the Clustering Hypothesis [Rij79]. This insight establishes that relevant content items are more closely related to each other than non-relevant content items.

2.2 Commercial Applications

Outside of the academic forum, the software industry has conducted extensive research and development into technologies involving Personalization and Distribution of Audio. Much of the research is ad-hoc and the exact methodology is not public, but nonetheless it has provided insight into the feasible and interesting applications of this technology.

The necessity for personalization occurred during the Internet revolution of the late 1990s. At this time, a myriad of companies 'realized' the potential of the Internet and chose to place a vast supply of audio, visual, and textual content onto the web. The prevalent logic was that if something is **accessible** online, then consumers will buy it. In their haste to establish a presence online, these corporations failed to create a long-term plan to allow end users to search for their content. As such, everything may be online, but there was no way for the end user to ever find it.

Search engines such as Yahoo, Excite, and Google sprung up to remedy this problem. Some such as Excite offered help in finding audio. These companies succeeded admirably in solving to problem of allowing users to find relevant content online. However, with the ability to find specific content came the realization that users do not know what content they desire. Any of the above search engines yield thousands of matches to nearly any query, but this is useless to such a user. The new goal has become personalization: to give the end user what he wants before he knows that he wants it.

Examples of this application are varied and widespread. The goal of top Internet retailer Amazon.com[5] is to aid customers in "Finding and Discovering" their merchandise. While the first verb is a standard searching algorithm, the latter is an entirely different class of operations based on personalization, 'finding things that the customer wants before he knows that he wants them.' Amazon.com created numerous personalization and clustering features for its web-store. Intricate knowledge of its customers' habits both improved sales and allowed it to expand into new markets. Amazon.com discovered that their fast store of user book purchasing habits allowed them to predict music tastes. This insight allowed the company to dominate the online CD marketplace despite being a later entrant into this field.

Similarly, advertising leader DoubleClick[6] set up a network that spanned an estimated 10,000 websites. Through this network, the company could track the actions of a specific user and determine his or her preferences. By doing so, DoubleClick could tailor ads accordingly and produce a higher rate of sales.

[5] http://www.amazon.com

[6] Site: http://www.doubleclick.com, Logo: http://www.doubleclick.net/us/images/logo.dc.gif

While the above descriptions focus solely on the process of marketing and retail, corporations have conducted additional development to personalize content presentation. TiVo[7] created a hybrid Video Cassette Recorder (VCR) and content classification system. This device hooks up to a standard Television, and uses personalization algorithms to determine what kind of content a user enjoys. It proceeds to recommend new content for the user, and automatically records it for future playback. Numerous companies have devised audio personalization systems for the Internet, but none operate on the individual level. These sites involve segmenting by genre[8] or catering to a specific group of people. These methodologies do not make use of complex selection algorithms or feedback mechanisms. As such, there is significant room for improvement within this field.

Moving away from the topic of personalization, many firms have also been using the Internet for broadcasting radio content. Within the past 5 years, thousands of traditional radio stations have been simply broadcasting their content over the Internet. While technically feasible, legal community currently hinders this process. The Recording Industry Association of America [RIAA] recently chose to charge stations content fees three times higher than normal for webcasting. This is in addition to a host of rules concerning limitation on the number of songs that a station can play from a particular album or artist within a specified period[9]. Numerous web sites such as MP3.com and Launchcast.com have attempted to create online radio stations, with legal repercussions.

Digital radio technologies exist on platforms other than to desktop computers. For example, Ericsson Telephone Co. and IBM are designing hardware systems to play audio content in

---

[7] Site: http://www.tivo.com, Image Locations:
http://a423.g.akamai.net/7/423/1788/4800507a0e5da6/www.tivo.com/images/home_logo.gif
http://a423.g.akamai.net/7/423/1788/a11cf2100b64e1/www.tivo.com/images/home_act1.gif

[8] http://www.mp3.com, http://www.emusic.com

[9] Wired. Webcasters in License Limbo. March 27, 2000.
http://www.wired.com/news/technology/0,1282,34115,00.html

automobiles. Microsoft has designed the AutoPC operating system to run on these devices. GM has created OnStar, an in-dash computer to both aid the driver and entertain passengers.[10]

3. System Architecture

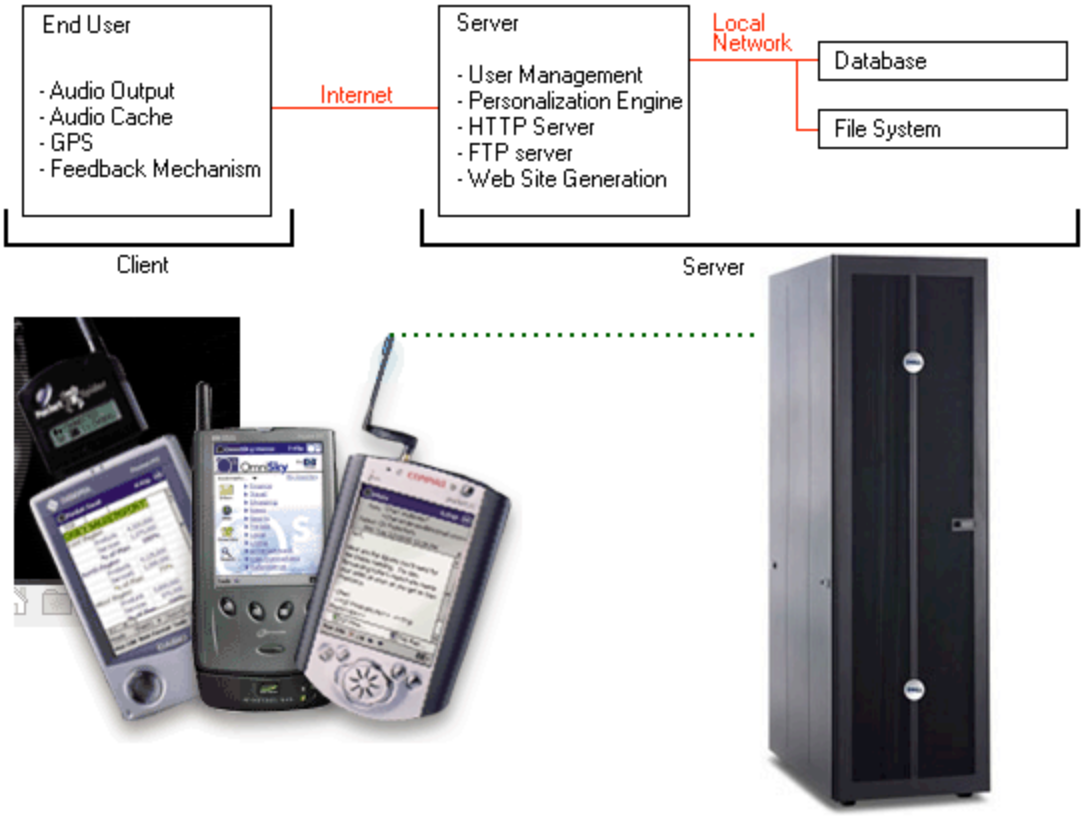
We have created a system that is capable of recommending and delivering audio content. Using stored user information, the framework can formulate recommendations for each user, and facilitate playback for the user in a platform and bandwidth agnostic manner. The system can match thousands of audio content items with hundreds of users, using algorithms that are based on proven clustering, stochastic, and Information Retrieval methodologies. We also attempt to simulate a marketable functionality within this framework by placing and testing advertisements. Furthermore, we formulate and conduct a demonstration to test these algorithms. The purpose of this exercise is three-fold:

1) To test the accuracy of these algorithms
2) To test the limiting hardware, software, and network load of these algorithms
3) To provide a positive and intuitive user experience

To demonstrate the process of tailoring discrete audio content on a per user basis, we created a novel system. This client-server system is pictured below, and Figure 1 below provides a high-level snapshot of the components, together with a visual representation of the devices involved.

[10] New York Times on the web. Microsoft Announces Windows in Cars. By AP. http://www.nytimes.com/aponline/technology/AP-Microsoft-Autos.html

Figure 1: The Components of Our System[11]



From left to right in the above diagram, we explain several critical components in detail below:

1) The Client [Section 3.1]

2) The Internet connection between the client and server [Section 4.2]

3) The Personalization Engine [Section 4.1]

4) The database [Section 5.2]

We formulated this model to make as few assumptions about the client as possible. Under the current system, a minimal client needs only the ability to connect to the server via HTTP, be able to download audio content, and be able to play it back to the user.

---

[11] Portable Devices Image:
http://www.microsoft.com/mobile/pocketpc/wireless/images/wire_hmbuzz_1.gif
Server Image: http://www.dell.com/images/global/products/pedge/rack_4210.jpg

3.1 The Client

Under the current model, many devices can function as a client. The client can be a Pocket PC that the user carries around during the day. The client can be an autoPC, which is permanently located in the user's automobile and only has sporadic access to the Internet when it happens to be in range. The client can be a personal computer that is located on a stationary desktop and has a permanent connection to the Internet.

The client has three requirements:

1. A method of playing audio to the user
2. A method of communicating with the server
3. A method of downloading content to play immediately, or storing it until the correct time.

The client has several optional requirements:

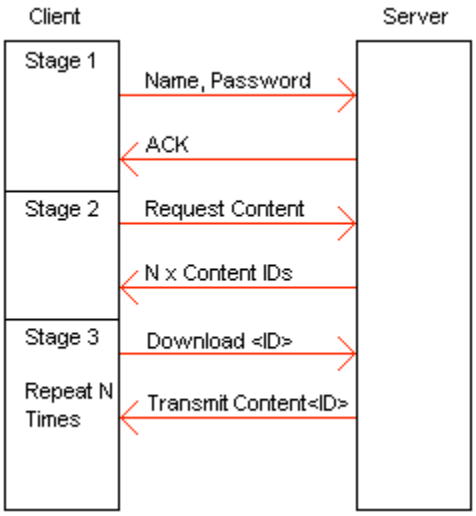
4. A Feedback Mechanism, with the ability to store results until the device comes back into contact with the server
5. A GPS device [if mobile]
6. An onboard method of selecting from stored content based on GPS information

The second requirement is that the client be able to communicate with the server. At present, the accepted protocol is HTTP due to its ubiquitous nature and compatibility. However, there is no unique aspect of HTTP that makes it the sole choice. The system might integrate other protocols in the future.

The client uses a three-stage protocol to obtain content. First, it must authenticate its identity to the server. The server responds by acknowledging the client. Second, the client must request new content. The server responds with the identification codes of the recommended content. Third, the client submits the content identification codes to the server. The server responds with the complete content file.

Note that the client is then free to play the content for the user immediately, or cache the content for future playback. The client is also free to cache the recommended codes and download the content in the future. This flexibility proves useful in adapting the protocol to the wireless environment presented in Section 4.2.

Figure 2 – A visual representation of the 3 stage protocol between the client and the server.



The third requirement is that the client possesses a method to download content for immediate playback, or the facility to store it until the correct time. In short, the client must have a real-time connection to the Internet so that it can stream content directly, or it must have a cache so that it is capable of downloading content in advance and storing it for play later.

Within the above parameters, it is possible to build many forms of clients. We have experimented with four types. First, it is possible for the client to be a web page, which plays content within the Internet browser. Second, it is possible for the client to be an external plugin to a web site. Thirdly, the client can be a completely separate application able to download content and play it. Finally, the client can be a standard MP3 playing device that can interface with software that fulfills the three aforementioned requirements.

3.2 The Server

The server is the more complicated entity in the system. It must perform 5 core tasks:

1) Communicate with the client over a network,
2) Create, authenticate, and manager user accounts,
3) Track all data and metadata about each user and content item,
4) Formulate recommendations based on this data, and
5) Selectively hand out content.

Client-Server communications use the HTTP protocol because it is ubiquitous and flexible.

We use CGI to create, authenticate, and manage user accounts. We store the results in a SQL based database. Similarly, we store all available user information in numerous database tables. We describe these scenarios in detail in Section 5.2.

The recommendation engine is our most complex task algorithmically. It uses all available information to choose content items for users within a given set of constraints. We discuss this algorithm in depth in Section 4, and optimized in Section 5.

4. Recommendation Model and Algorithms

There are two principle algorithmic areas in this paper. First, we present the algorithms responsible for the personalization and ordering of the recommended content. Second, we discuss the protocols and algorithms required to resolve issues encountered within a wireless framework.

4.1 Personalization Algorithm

Before describing a Personalization Algorithm (PA), it is necessary to define its functionality in detail:

> A Personalization Algorithm uses **available information (1)** to choose a **subset (2)** of the content in a **specific order (3)** within **time and space constraints (4)** to **optimize utility (5)**.

This definition requires elaboration. First, we assume that the PA has available all of the information that the current user as well as all other users has provided to it. These consist of both past recommendations to the users, their feedback, existing or derived metadata describing the content, and metadata describing the users.

Second, we define the output of such an algorithm as an ordered subset of the audio corpus.

Third, the content may provide different utility if played in a certain order. For example, the user may derive utility from listened to content which is similar in genre, or possibly with alternating genres [AT00]. There may also be an advantage to playing content at a certain time. The order of play dictates the exact time at which the client presents a content item to the user, and so it is

significant. Finally, it may be desirable to provide 'instant gratification' to the user to improve their initial impression of the system and improve retention. Thus is may be beneficial to present the content item with the highest estimated utility first. As such, an algorithm can have different priorities when determining an ordering.

Fourth, constraints may limit the type, size, or length of content that may be recommended. For example, the client device may have a fixed cache size that can only hold 10 megabytes of data. Alternately, the data may have an associated cost, for example, if the user has to pay 10 cents per content item and she has limited her spending to 2 dollars per day, the utility per unit price metric must be optimized and its constraint followed.

Fifth, utility is defined as 'the amount of benefit that the user receives' from the recommended content. It is an extremely difficult and hard to quantify problem to perfectly assess utility for a human. As such, an estimation function is the best that the system can do to predict how much utility a content item will provide for the end user.[12]

Consider the algorithmic implications. As defined above, the PA problem is NP-Complete.[13] Since the academic community considers this category of algorithm to be intractable using present computing methodologies, we must either pare down the definition or begin examining approximation functions. We investigate two methods to simplify the computability of the problem. First, we stochastically filter the set of content items. This has several implications. The most important us that we generate a smaller input set, which can reduce an intractable problem to a problem that can be computed in real-time. The implication of this operation on the quality of the algorithm is minimal, as is shown in the following section.

---

[12] Though we politely glossed over this detail, note that the fifth point could be defined alternately. Currently the algorithm will optimize the utility of the user. It is equally possible to optimize the utility of other parties related to the system. For example, the creator of the content, who will perhaps be selling each item at a specific cost, would optimize her utility by a maximum profit. It is similarly possible to jointly optimize the sum of the users' utilities. While this is feasible, we will commence by focusing solely on optimizing the one user's utility.

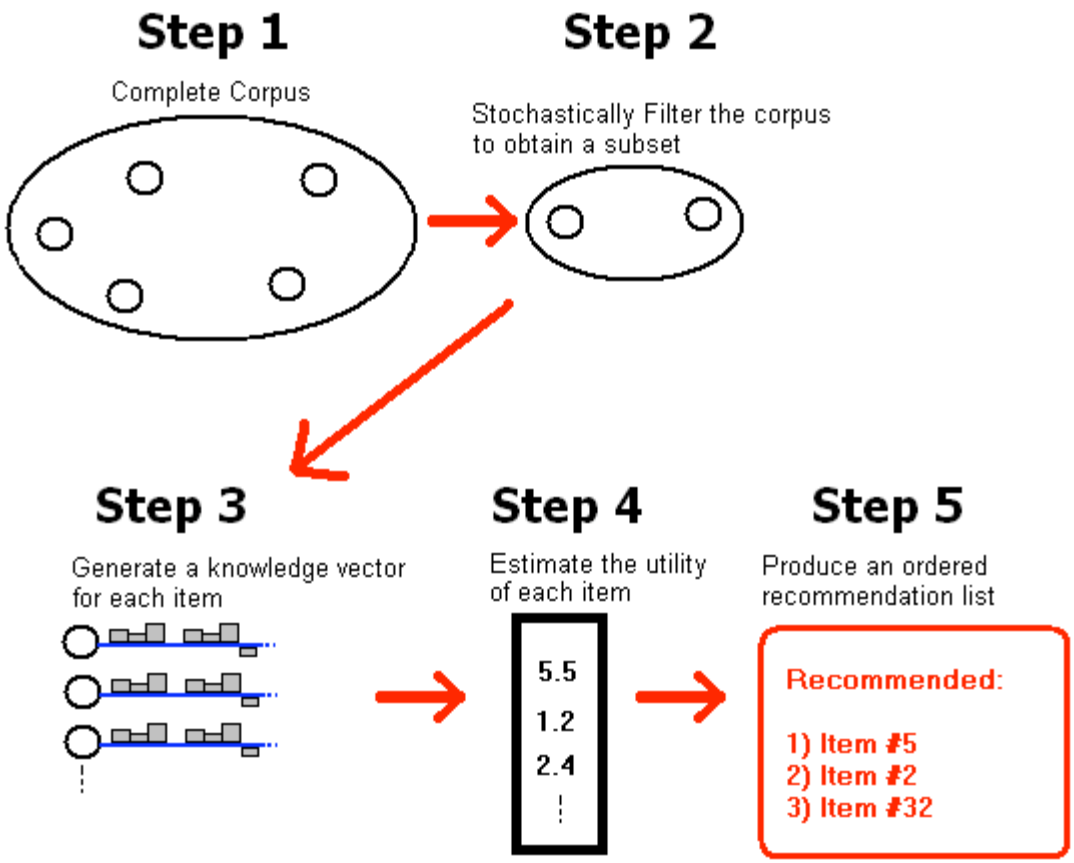
[13] The 0-1 Knapsack problem reduces to the PA algorithm.

The second option for simplifying the Personalization Algorithm is to fill the recommendation list with items using a greedy algorithm instead of the optional 0-1 knapsack methodology. While this step would reduce the running time to a computable level, it may influence quality. Note, however, that if we limit the scope of the possible constraints imposed by the client device, a greedy algorithm may still yield an optimal solution. If the only constraint the number of songs that could be given to the client, then the greedy algorithm would function optimally. On many streaming clients, this is the only constraint imposed, and thus it is worthy of consideration.

The Personalization Algorithm:

1) Begin with the full corpus
2) Stochastically Filter Content to create a workable sample size
3) Create a knowledge vector for each item in the sample
4) Estimate Utility of each item in the sample using clustering methodologies
5) Use 0-1 knapsack problem to fill up the recommendation list

Figure 3: The process of generating recommendations



Utility Estimation

The Utility Estimation in this project is based on prior work on Clustering Algorithms in the field of information retrieval [APR99], and in particular its use in filtering [APR00]. The core to the Utility Estimation process is based on a offline clustering algorithm.

Building on the definition of a PA, it is necessary that we define a method to represent all of the available information within the system in a manner that allows convenient estimation of utility. Using Clustering intuition as a guide, we formulate our data into a *similarity graph*, which is defined as an undirected, weighted graph G = (V,E,w) where each vertex corresponds to a content item. Furthermore, we add one vertex to the graph G for each user. Each edge weight corresponds to the similarity between two items. We measure similarity between two documents by using a metric that is standard within the Information Retrieval field. The cosine metric in the vector space model of the Smart information retrieval system [Sal91, Sal89] has been used extensively.

The vector space model within our Utility Estimation function contains seven different categories of dimensions. Each of these represents a type of data used to estimate utility. There is no fundamental difference between these categories, and they are all formulated such that the cosine metric will function without modification. We use these divisions to emphasize the different sources of the data, since user and content vertices are filled using different mechanisms.

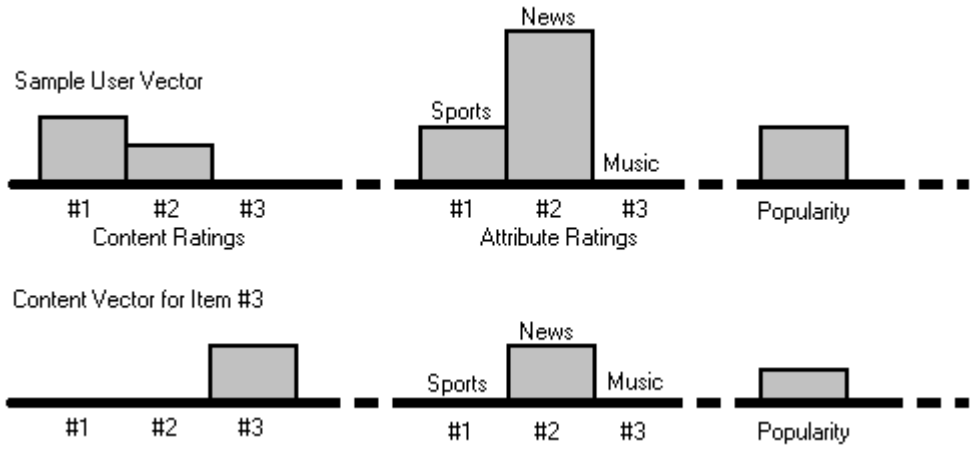
The seven categories of dimensions are as follows:

Category	**Content Data**	**User Data**
1) Content Ratings. n dimensions are allocated for content, where n equals the number of content items.	Content j receives a 1 in dimension j, and a 0 in all other content dimensions	0 for all unrated content. The rating, positive or negative, for each rated content.
2) **Temporal Rating.** 24 dimensions for hours. 365 dimensions for days. Or as many as applicable.	0 if no applicable temporal rating. Above zero if it is tuned for a specific time or date.	Set the current time and date with a rating of 1. Create a bell curve ahead and behind.
3) **Spatial Rating.** 3 dimensions, x-y-z. These are applied to longitude, latitude, and altitude	0 if no applicable spatial rating. Above zero if it is tuned for a specific location.	Set the dimensions equal to the current location.

and aid in geographic relevance.		
4) **Attribute Rating.** m dimensions are allocated for attributes, where m equals the number of attributes.	0 if the content does not possess that attribute. 1 if it does. [or higher as applicable]	0 if the user is neutral on that attribute. 5 if the user is in favor. 50 if the user would like to listen to this attribute exclusively.
5) **Frequency Rating.** n dimensions are allocated for content, where n equals the number of content items.	Content j receives 0 if it does not diminish in frequency. Above 0 if it does.	Negative if the user has heard the content j recently. Positive if the user has not heard the content recently.
6) **Popularity Rating.** 1 dimension	Sum of all of the user vectors assigned to the song in the content rating	1
7) **User Rating.** w dimensions were w is equal to the number of users	Dimension k contains the rating of this content by user k.	Dimension k contains Similarity rating with user k as described in the User to User Comparison section below

The result of this representation is a high-dimensional vector space. Once formulated in this manner, the model can compare any user with any content item to determine the similarity between them. By construction, similar users and content items map to nearby vectors. In the vector space model, similarity equals the angle between the corresponding document vectors. The standard in the information retrieval community is to map the angles to the interval [0,1] by taking the cosine of the vector angles.

Figure 4: The following are sample dimensions within a User Vector and a Content Vector. The vectors below depict the Content, Attribute, and Popularity Dimension. In this particular instance, the user has not rated this content item before. The user has also exclusively requested News [as shown by the high bar in the News dimension]. Since the content item is a news attribute, when these vectors are multiplied, they produce a high value that denotes similarity.



Given the nature of the algorithm, three applications are presently feasible. User-to-User Comparison and User to Content Comparisons are vital in order to perform the operation. Content-to-Content comparison is of interest within the clustering field, but is not used within the context of this paper.

User to User comparison

This process is an intermediate step in formulating the user Information Vectors, specifically filling in the User Rating category of dimensions. The critical insight is that we can determine which users possess similar tastes to the current user, and then factor in their content ratings accordingly to aid in formulating future recommendations. Only two categories differ in User-User comparisons: Attribute Rating and Content Rating. As such, we can construct vectors for each user leaving all but these two dimensions at zero, or simply leaving out the extraneous dimensions. We then determine the cosine of the vector angles to determine the similarity between the users. The algorithm then places this result in the User Rating dimensions, and it aids in User to Content Comparison.

The justification for this critical insight stems from Dense Star-Shaped covers as presented by Aslam, Pelekov, and Rus [APR99].

The intuition of this result is as follows:
Assume that the similarity between person 1 and person 2 is 0.9. Furthermore, we know that person B heard a content item and rated it highly. Person A has not heard this content item yet. However, we can infer that person A is likely to enjoy this content item.

Since we have defined users and content items within the same vector space, we can compare them directly. The mathematically rigorous proof of this statement is as follows:

Consider three Information Vectors U_1, U_2, and C. U_1 and U_2 represent the two users described above, while C represents the content item. These entities are vertices of a star-shaped subgraph of graph G, which has a threshold of σ. We can assume that $\sigma < 0.9$ and that the similarity between U_2 and C is greater than σ [lets say 0.7]. U_1 and C are satellite vertices and U_2 is the star center. We obtain the similarity coefficients in the vector space model by s calculating the cosine of the angle between the information vectors of each document.

Fact 1 *Let $G\sigma$ be a similarity graph and let S1 and S2 be two satellites in the same cluster in $G\sigma$. Then the similarity between S1 and S2 must be at least [APR99]*

Equation 1: $\cos(\alpha_1 + \alpha_2) = \cos \alpha_1 \cos \alpha_2 - \sin \alpha_1 \sin \alpha_2$

Using the numbers stated above, we conclude that the similarity between User 1 and the content item must be a minimum of:[14]

$$(0.7) \bullet (0.9) - \sqrt{1-(0.7)^2}\sqrt{1-(0.9)^2} \approx 0.32$$

While the above result does not imply a strong match, it mathematically proves that User 1 is likely to enjoy the content item C. Note further that this result is a worst-case scenario, and it is possible that a stronger match is present.

---

[14] Note that $\sin \phi = \sqrt{1-\cos^2 \phi}$

Having established that our intuition is mathematically grounded, we show that it is computable within the vector model. Recall that we constructed the User Rating set of dimension so that the user vector contains the similarity between the current user and user j [cos α1], while the content vector contains the similarity between user j and the content item [cos α2]. Multiplying these two items in a vector multiplication yields the first term in the equation 1. We will proceed to show that this is sufficient to provide useful results.

Equation 2: $\cos(\alpha_1 + \alpha_2) \approx \cos \alpha_1 \cos \alpha_2$ as $\cos \alpha_1$ approaches 1

Since the construction of the user vector only includes the users that are similar, we conclude that ($\cos \alpha_1$) approaches 1. If this is the case, then the second term will approach 0 regardless of the value of $\cos \alpha_2$. Thus, the remaining value is the first term, and our approximation is correct.

User to Content Comparison

The previous section describes the process for constructing the vector for each user vertex within the similarity graph G. At this point, we are ready to discuss the process of determining m recommendations.

User to Content Personalization Algorithm, choose m items:

1) Pick the user in question V_0
2) Stochastically pick n content items, V_1-V_n, where n > m
3) Create edges between (V_0, V_i) for all i between 1 and n
4) Calculate the weights of the edges according to the cosine of the vector angles
5) Perform the 0-1 knapsack algorithm to choose the optimal items within constraints. If no constraints or n is too large, then use a threshold to greedily take the top m items.
6) Sort the items according to the optimal time ordering if applicable

The recommendations are currently optimal within the Stochastically selected sample set. We sample to reduce the running time of the algorithm. For example, assume that we draw k items for each item needed. We assume that the similarity of documents to the user is distributed uniformly about the range [0,1]. The average quality of the resulting documents can be estimated. According to statistics, if we draw k items from a uniformly distributed range of [0,1], then the expected maximum is (1-1/k). Assuming that (the size of the corpus) >> (the number of

documents sampled), then we can assume that each drawing is independent. Thus if we draw k samples m times and take the maximum each time, then the average quality of the results will be E[1-1/k]. As k grows, the marginal benefit of this extra sampling shrinks asymptotically.

Within the model using Stochastic sampling, we state a desired expected quality. In the case of the demonstration, we used 90%, which resulted in a k of 10. In comparison to using the full corpus, the running time of the algorithm is significantly reduced since we are using a fixed constant as opposed to a variable input equal to the size of the corpus.

Stochastic filtering prevents an optimal outcome. This is helpful a helpful feature. Without some randomness injected into the system, it is likely that the recommendations for a user would settle at a local maxima. The filtering allows the system to attempt slightly sub-optimal content items to check user feedback and change accordingly.

Ordering Algorithms

There are numerous methods for choosing the ordering of the content. Alghoniemy and Tewtik [AT00] describe situations that allow the algorithmic optimization of content ordering in response to the users' preferences. The algorithm could perform this process at step 6 above to determine the proper order to place the items.

We presently choose to optimize for 'instant gratification,' by sorting the content in decreasing order of similarity number. Exploring in depth other efficient manners to extract ordering preferences from the data remains an important research topic.

Exclusive Categories

One of the requirements within our model is that the user be able to choose to listen to a particular category exclusively. For example, a busy person on the way to work may decide to listen to the news, weather, and nothing else during the short commute. The algorithm can account for this by assigning an extremely high value on the user's Attribute Rating segment for those particular items. Within the algorithm, vector dimensions generally have a magnitude between [0,1]. By placing a magnitude of 100 on an attribute causes it to dominate all other factors, and ensures that the personalization algorithm chooses items in this category exclusively. The other dimensions are still be counted accordingly for personalization and ordering purposes. Thus, the concept of an exclusive category can exist within the vector similarity model.

Advertisements

In some applications, the algorithm must interlace advertisements with the content. To fulfill this requirement, it is simple to grant a particular category a bonus by predisposing each user towards it. The PA fulfills this goal by assigning each user to have a value of 1 [or higher] in the advertising Attribute Rating. This ensures that advertisements are presented within the model and do not swamp the regular content. Advertisements can also possess additional attributes, allowing the PA to select advertisements based on their other traits accordingly. The algorithm will target users with advertisements that they are likely to enjoy. See Appendix 1 for a discussion of the marketable features of this approach.

4.2 Wireless Considerations

Wireless devices can fit within the previously established framework, but they do require significant variations. First, wireless devices are often capable of movement. Thus, user location becomes a particularly valid attribute when formulating recommendations. Second, wireless devices often have a tenuous connection to the network, or do not have sufficient bandwidth to connect at speeds which are capable of downloading content in real-time. Third, wireless devices might connect to networks that rely on multicasting. This aspect is inherent to satellite and various wireless broadband technologies. While these three facets are not confined to wireless devices, connectivity variations are especially extreme in the context of wireless devices. As such, we discuss these connectivity issues together in this section.

Real-time Connectivity

We define Real-time Connectivity as the ability to download and play content in real-time. The category of devices that possess this attribute includes most desktops and any device on a wireless network of sufficient speed. The advantages of this category of device are multifold. First, the device can query the personalization often to yield more accurately recommendations based on up to date information. The PA can account for variables such as time and location that are subject to frequent fluctuations. Second, these devices can play audio directly off the network, and do not need to cache content. This aspect results in a cheaper device since they does not need caching functionality.

Slow Connectivity

A device with slow connectivity is capable of communicating with the server at anytime, but it is not able to stream content fast enough to play it immediately. Two primary mechanisms can compensate for this deficiency. First, the device must be able to cache content. This ability allows the device to spend its idle resources downloading content for use later. Caching also allows the device to reuse existing content, which both occupies the end user and frees up bandwidth for downloading new content. A second compensation mechanism is to transfer content in a more concise format. The client device can then use increased processing power to present it to the user. An example is to transfer content in textual form and require that the client use Text-To-Speech software to convert it before presenting it to the user.

Slow connectivity still assumes that the client can query the server at any time. The client can still receive new content selections in adjustment to factors with high variability such as location. A device which can download textual information in real-time, coupled with a Text-To-Speech module reduces this problem to the aforementioned Real-time Connectivity case.

The format of the content warrants discussion. Content is stored and transmitted in a concise format for two primary reasons. First, the client could store more content in less space. Thus, the client could have less storage and thus be cheaper to manufacture, or that the client could store more content in available space. Second, if the audio file is smaller, then the bandwidth required to fulfill the definition of Real-time connectivity it reduced. For these two reasons, we chose the MP3 format for the demonstration. It is approximately an order of magnitude smaller than a WAV file of equivalent quality, and is nearly as ubiquitously accepted. As such, within the context of this paper, we define Real-time connectivity as 17kBps. If we had chosen an equivalent WAV file, real-time connectivity would have been approximately 100kBps.

Sporadic Connectivity

A device with sporadic connectivity is capable of communicating with the server only during certain intervals, while the rest of the time it has no connectivity whatsoever. To compensate, the device requires a cache. However, it also requires sufficient predictive abilities to compensate for the lack of any direct communication with the server. One possibility is to download content in advance and use an onboard predictive engine can pick from the available media on the fly based on current data. Such a device also needs to cache any user feedback for uploading later.

The sporadic connectivity case requires a larger cache even more than in the slow connectivity case. The former has a significantly longer lag between updates from the server and requires more onboard content to avoid being repetitive.

Multicast Downloading

A current technology is fast-multicast download coupled with slow or non-existent upload. These qualities are present in various satellite networks, cable, and digital radio stations. The advantage of this technology is that it is already in place, and that it requires significantly less processing power and bandwidth on the part of the server since it does not need to connect to each device individually.

Three mechanisms allow the model to function using this technology:

1) Client intelligence
2) Server Prefixing
3) Server Confirmation

Client Intelligence implies that the client possesses a minimized version of the Personalization Algorithm. It also implies that the server must broadcasts each content item prefixed with its attributes. This process allows the client to create an information vector for the content item as it arrives, compare it to the user to determine its similarity, and then compare the similarity rating to other items in the cache. If the quality of the item in question exceeds the items in the cache, then it will be stored and played for the user. If it is not likely to exceed the utility of the items in the cache, then it will simply not be stored. A client using this model needs a cache and enough processing power to run a version of the PA.

Server Prefixing assigns the server the task of deciding which content the client should store. The server prefixes each content item with a header that lists all of the users that should download it and store it in their cache. The client can thus be simpler, all it needs to do is read the header, check if it is one of the listed users, and then choose whether to store the content item for later playback.

Both of the above options assumed that the client could not communicate with the server directly. Server Confirmation makes the assumption that the client has Slow Connectivity available. In this model, no prefixes are required for each content item. The client must occasionally send a

message to the server asking about content items. For example, the client may cache 10 broadcasted content items. It then communicates with the server and checks if it should play any of these items for the user. If the server responds with a message stating that they are not likely to generate utility, then they will be not be stored, and the process is repeated. This method has the advantage that it will continually update the selection with recent items, but has the disadvantage that the cache is constantly churning.

5. Experiment

In order to test the functionality of this system, we have created a demonstration. For these reasons, a demo was created. It was small scale. The specifications were as follows:

Goals for the Demonstration	
Users:	50-100 Dartmouth Undergraduates, Graduates, and Faculty
Dates:	May 11, 2001 – May 26, 2001
Content:	A mixture of News [The Daily Dartmouth, online news sources], Weather, Music [local bands, free music, classical, user's personal collection], Stock quotes, Local Events, ads, User Generated content [i.e. – Radio shows]
Content amount:	1 hour of new content per day
Ads:	10+ local businesses

The purpose of the demo was to act as proof of concept for the algorithms and system. As such, numerous questions need to be answered:

First, can it be done? Is it possible to implement these algorithms to distribute content to users on present day architecture at a reasonable cost? This small-scale demo simultaneously identifies difficult to implement software modules and costly hardware units.

Second, can it be used? In order to make this project market friendly, the desired user has minimal computer knowledge. Can they use the system correctly and find it to be functional?

Third, what processes are required in a large-scale application of this concept? What are the bottlenecks within the software that would be difficult to scale?

5.1 Client Details

We designed our system for four types of client software.

Stand Alone Client

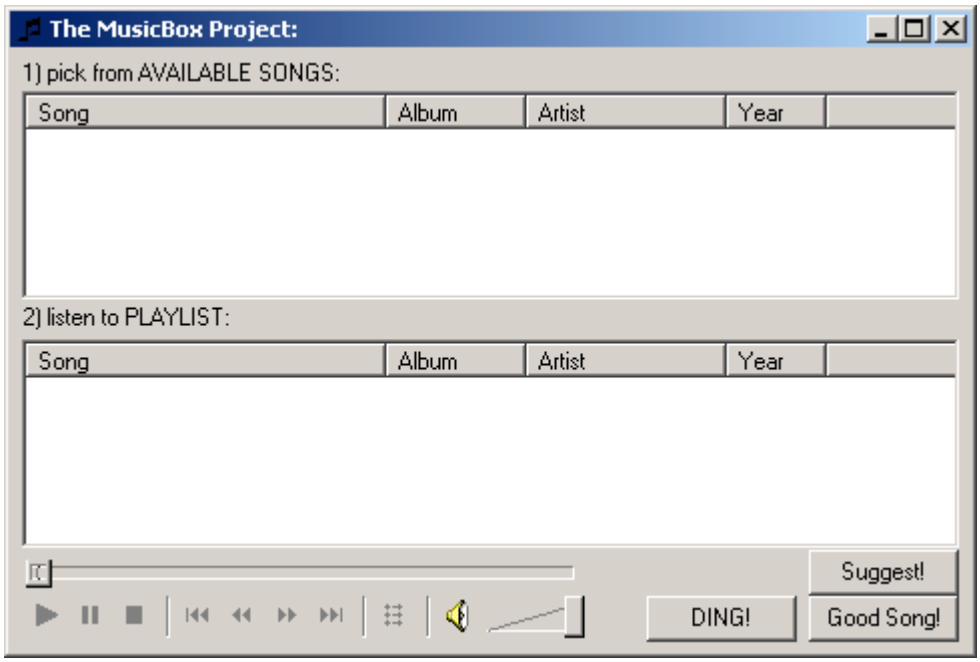
We wrote the earliest client for Windows, in C++ using the MFC libraries. The name of this program was 'The MusicBox Project' and Figure 5 contains a picture of it. It existed as a 250k executable that required no installation. It conducted authentication of the user's identity through the Dartmouth Name Directory. This client used an embedded copy of Microsoft Media Player to stream content. It used HTTP to communicate with the demonstration server, and assumed real-time communication. As can be seen on the image below, the interface of this program consisted of three primary sections.

In the top section contained 5 content items as a subset of the corpus. The user could select which content item would begin playing. This limited sampling allowed the recommendation engine to acquire preliminary user preferences without giving the user direct control over all of the content available.

The middle section contained the current playlist of five content items, and allowed the user to view the upcoming selections.

The bottom section allowed the user to rate and skip content items. The rating simple rating system included one negative button ["Ding"], which recorded a negative rating and caused interface to skip to the next content item. There was also one positive rating button ["Good Song"], which merely recorded this fact and kept playing. The "Suggest" button prompted the client to load 5 new suggestions from the server. Finally, the user could skip through the song, control the volume, or pause it using the standard Media Player interface on the bottom left.

Figure 5 – The layout of a Windows Stand-Alone Client



We distributed this program to 20 Dartmouth undergraduates and received positive results. Feedback included the desire to add one's own music to the selection, and the desire to rate items more directly so that the client would learn tastes more rapidly.

The server was set up to communicate with this client via CGI scripts. The client would query the server using the protocol shown in Figure 2.

Web Browser Based External Client

While the above client functioned very well, it was desirable to create a client which A) was system agnostic so that it could run anywhere B) had no installation or binaries. The solution was to create a client that ran entirely within a Web Browser.

The technical aspects of the solution require more explanation. In this model, the client does not exist as a standalone program. Instead, the server generates the HTML code that forms the client. The protocols still work in the same fashion in that they transfer the same core information. The server now must embed that data in human readable HTML, which contains the link to the next step in the protocol. For example, the user first goes to the starting page, and the server presents a web page with a login form [stage 1 of the protocol in figure 2]. The user fills in her name and

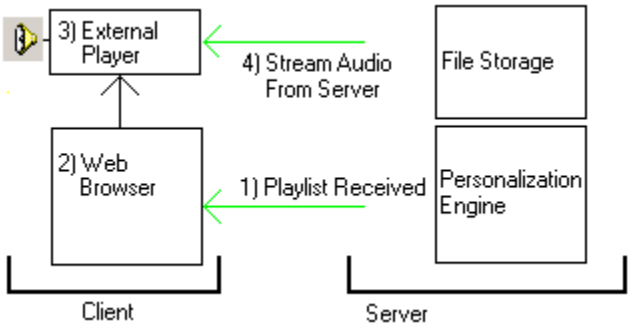
submits it to the server, at which point the server authenticates. If it is successful, then it replies with a web page linking the user to request content from the recommendation engine [stage 2 of the protocol in figure 2].

The main difficulty was that not all web browsers are created equal. Specifically, we cannot assume that a web browser has the ability to play audio content. Even the browser could play the sound file; there is no ubiquitous manner to direct a browser to play multiple files in order. As such, the remaining solutions were to use external programs to accomplish this goal.

Fortunately, the HTTP technology has a near universal method of executing a specific application using a downloaded file. All that is required is that one sends the file with the correct MIME type. The browser spawns a copy of the desired program and directs it to open the file. The desired file type was a playlist M3U file. The MIME type is "audio/x-mpeg". This activates an audio player and sends it a playlist with one URL per line. Similar applications exist for all platforms.

The client architecture is thus consistent with the original protocol. The playlist is now the response to the request for content. The external audio player conducts the third stage of the protocol by downloading the required content items and playing them directly. An additional feature is that one can setup the playlist with embedded titles for each content item. Many applications display the titles correctly, while the rest politely ignore it.

Figure 6 – The data flow in a Playlist based external system. The browser receives the playlist, and forwards it to an audio player. The player then streams the content.



Advantages

1) This methodology is completely platform independent. The only assumptions are that the end user has an audio player, and that your web browser's MIME types are set up correctly.

Disadvantages

1) There is no method within this model to add new content once the playlist is exhausted.
2) Digital Rights Management is currently impossible in a system neutral manner
3) It is extremely browser dependent on how to set mime types, and this should not be a task for those unfamiliar with computers.
4) There is no mechanism custom rating mechanisms within the external audio player. In fact, the server never knows what application it is.

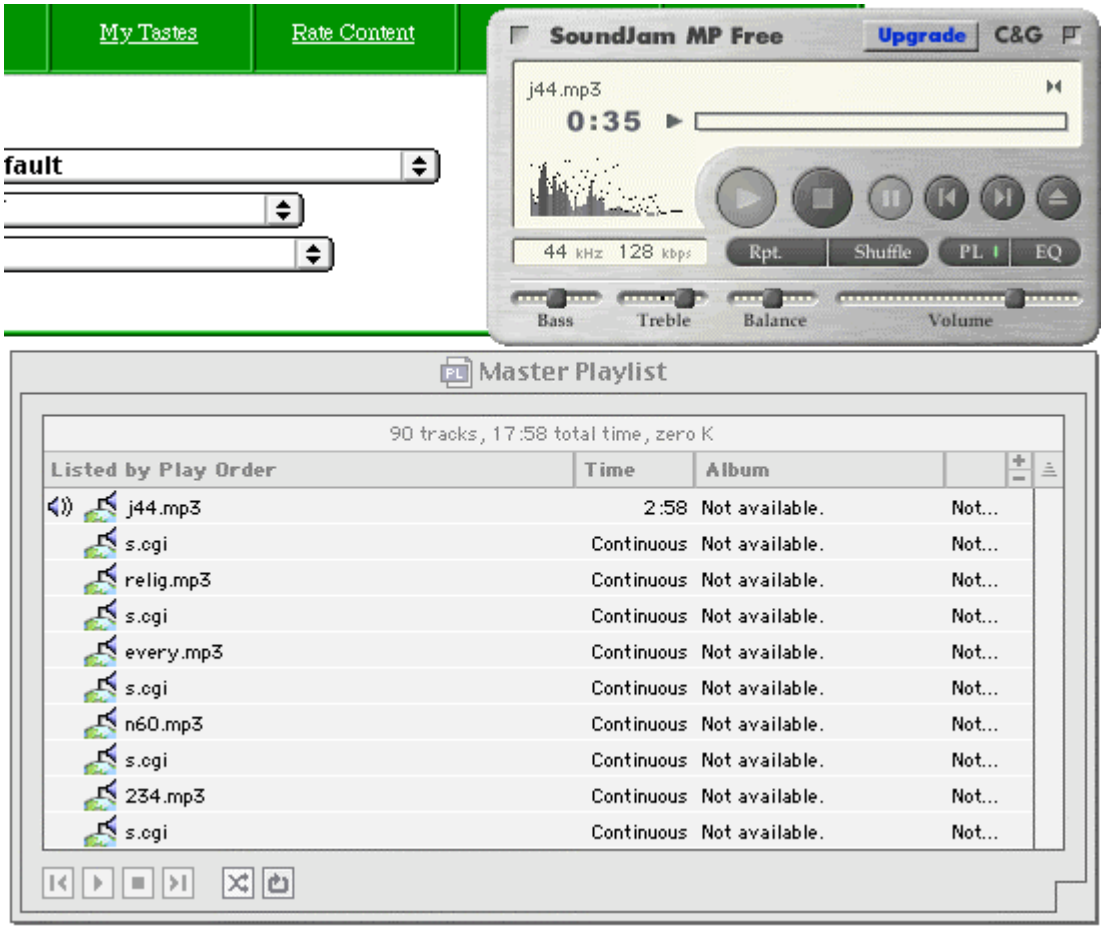
Unfortunately, there are no remedies for the first two disadvantages. The third can be resolved with extensive and user-friendly documentation on how to set up the programs correctly. However, we can minimize the fourth disadvantage by carefully constructing the playlist. While it is not possible to directly determine what content the user listens to, we know that the external player attempts to download every item on the playlist as it encounters them. Thus, it is possible to interlace URLs to scripts between every item on the playlist. After finishing content item #1, the external player tries to access the script that immediately follows. This script has its parameters specified so that it alerts the server that content item #1 just finished and content item #2 will begin next. The server notes this information and the time that it arrived. The script then returns a sound file containing a fraction of a second of silence, which prevents the external player from recording an error.

With the recorded information, it is possible to make numerous inferences as to the actions of the user in relation to the external player. For example, suppose that 9 seconds after the script in the above paragraph was called, the server is alerted that the user has finished item #2 and will start listening to item #3. The server checks and realizes that item #2 is a three minute long song. As such, the server can infer that the user must have skipped item #2. At this point, it infers that the user did not like that song, and rates it accordingly. The server can make two additional inferences in other circumstances. First, if the user jumped to a non-contiguous item on the playlist and listens to it in full, the server can not only infer that she did not like the original song, but also infer that she must have liked the second one. Secondly, if the user skips from one

content item to a new item and does not listen to either in full, then the user liked neither, so the server rates these items accordingly.

This solution works extremely well in practice. We tested this mechanism extensively in the final demonstration, and no users had any negative comments on this system. A visual example of this system running in MacOS using the SoundJam audio player is below in Figure 7. As is visible in the background, the client is running in Netscape, and it has spawned a copy of SoundJam to play the audio. The "s.cgi" files represent the interlacing scripts. While it is possible to embed the title of content items within a playlist file, many applications politely ignore them. While the protocol works correctly, the URLs are not helpful in identifying the content item.

Figure 7 – An external player [SoundJam] used by the system for a Macintosh



Internal Web Browser Client

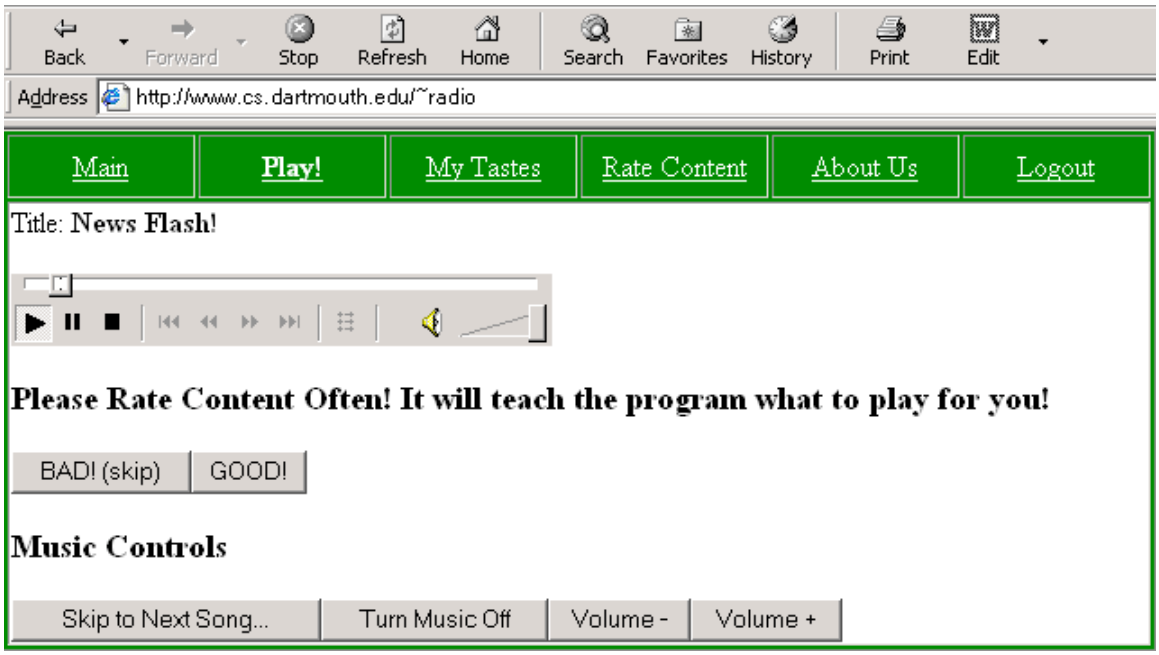
While the Sound Jam browser-neutral technology is useful in theory, in practice its disadvantages weigh heavily on this method. Foremost, it is quite difficult for novices to set up the MIME types correctly. In particular, many popular Apple audio players do not process playlists. In the consumer sector as a whole, Windows encompasses approximately 95% of Internet traffic. However, within the test population at Dartmouth College, Apple computers account for approximately 66% while Windows computers make up the remainder[15]. Internet Explorer encompasses approximately 90% of the Windows based traffic.

Windows is a significantly easier platform to work on for Internet based clients. The primary reason is that Windows is bundled with standard audio and Web Browsing software. As such, it is extremely likely that all the software that the user requires to run a web-based client is already present on his or her home system. Given the significant attrition during a lengthy software installation and the desire to obtain a very high participation rate, we used a specific web site for the Windows platform. Aside from the use of a different but compatible plugin, the Netscape for Windows platform functions in the exact same manner as Internet Explorer for Windows.

The Windows version of the web Client functions in nearly the same manner as the aforementioned system agnostic-version. The primary difference is that instead of relegating the audio playback to an external program, it is played within the browser as seen in Figure 8.

[15] Stephen Campbell, Head of Dartmouth Network Services

Figure 8 – The layout of the In Browser Player for IE for Windows



Note that the controls are extremely similar to the standalone client presented earlier. This client requires no setup whatsoever. It also allows content to be rated in a manner consistent with the other clients.

Within the demonstration, the server automatically determined which Operating System and browser the client was using, and presented the content using the appropriate method. The user was also capable of overriding these default settings if that proved necessary, though none of the users within the study found it necessary to do so.

Portable devices

We tested portable devices for compatibility with the demonstration. The server is capable of generating a playlist of songs and allowing the user to download them to a directory on their personal computer. At this point, it can be transferred to any the portable device for future playback. The devices tested include a Cassiopeia 115 and a Rio MP3 player. However, there is no theoretical reason why this methodology would not function with any current MP3 playing device.

5.2 Server Details

The server side of the demo is by far more complex than the client. It has four primary modules that are each be discussed in detail:

1. CGI Scripts
2. MySQL database
3. HTTP server
4. FTP server

CGI Scripts

We developed the entire personalization algorithm in a Common Gateway Interface. This format rendered it accessible to the client through the HTTP server. The program was implemented in ANSI C++ using MySQL libraries to communicate with the database. The executables were capable of identifying the variety of client though the HTTP protocol and tailoring its communication mechanisms accordingly. For example, the CGI scripts were capable of generating HTML websites for Browser Based clients, or stripped down Comma Separated Files (CSV) data for the Stand Alone client. The scripts are capable of receiving commands via GET or POST methods within HTTP.

The server contains only one CGI file. It would run all of the functionality of the program depending on the CGI parameters that used when calling it. The primary parameter is the "cx" flag, which states the function. The chart below described the seven primary system tasks. They operate for all of the client platforms.

CX =	Action	Auxiliary CGI Parameters
Login	Login to the system	ID, Password
Recommend	Produce a list of recommended content items	Number of songs
Create	Create a new account	ID, Password, Name, Class
Rate	Rate a content item	ID, Content ID, Rate
Listen	Inform the server that client listened to an item	ID, Content ID
Addcontent	Allows the user to add or upload content	URL, Content, Name, Source
NULL	Login page in the web client	

Recommend is the most processor intensive function. It runs the algorithm as described in Section 3.1. We implemented numerous optimizations to improve the running time, which are described in Section 5.4.

HTTP server

To be accessible to the client, we assumed that all forms of the client were able to communicate though the HTTP protocol. As such, the server was equipped with an HTTP server. This component allows the client to download recommendations and streaming content.

MySQL database

We used MySQL to handle all data storage and retrieval aspects within this demonstration. This software is free, robust, powerful, and quick enough for the standard and stress tested user load. The system made use of nine tables. Six are for user information, two are for content, and the last is for setting up a metadata hierarchy.

1) Users
2) User Attributes
3) User Preferences by category [meta-data]
4) User Ratings [preferences by individual content item]
5) User Listening History
6) User Listening Temporary
7) Content
8) Content Attributes
9) Preference Hierarchy

Table schemas:

Users

user_id	id	password	first_name	last_name	creation
int(11)	varchar(50)	varchar(50)	varchar(50)	varchar(50)	timestamp(14)

The user table stores all data that is unique to the user and is required for identification and authentication. The *id* and *password* are used for logging in. The *first_name* and *last_name* allow us to identify the user and contact her if necessary. *User_id* is the unique key for this table.

User Attributes

user_id	attribute_name	attribute_data
int(11)	varchar(100)	varchar(100)

All optional non-preference related user data is stored in the User Attributes table. This open architecture allows us to define new attributes later without altering the tables. Client settings and email addresses are examples of attributes that are stored in this table.

User Preferences

user_id	pref_id	pref_data	Time
int(11)	varchar(100)	varchar(100)	timestamp(14)

User preferences stores category ratings. For example, if the user states that he likes music [category 4], then the *pref_id* will be 4 and the *pref_data* will be a positive number designating how much the user approves of this category. Items in this table are used as part of the preference engine.

User Rating

content_id	user_id	rate	Time
int(11)	int(11)	float	timestamp(14)

When a user rates a specific content item, the result is stored in this table. *Content_id* is a foreign key into the Content table and *user_id* is a foreign key into the User table. Rate is a value symbolizing the rating. It is a float to allow it to be modified according to future research.

User Listening History

content_id	user_id	Time
int(11)	int(11)	timestamp(14)

This table stores the approximate time when a user listens to a content item. Different schemes of the system place it either at the beginning, the end, or when the server confirms that the user has played the song. The exact time is not required. This table is used in the recommendation process to ensure that a content item is not repeated too often.

User Listening Temporary

content_id_next	user_id	time
int(11)	int(11)	timestamp(14)

This table is used in the Web Browser Based External Client in order to implement the inference engine. It stores the next song that a user will listen to.

Content

content_id	name	type_id	user_id	creator	creation
int(11)	varchar(100)	int(11)	int(11)	varchar(100)	timestamp(14)

All identifying information about a content item is stored in the content table. *Content_id* is the unique key within this table. The field *name* is the human readable name of this item. The field *type_id* is a binary type identification of the item. The field *user_id* determines if a single user is allowed to play the item. The field *creator* identifies the artist in the case of music, or author in the case of a text article.

Content Attributes

content_id	attribute_name	attribute_data
int(11)	varchar(100)	varchar(100)

All additional metadata on a content item is stored in the Content Attributes section. This data is used to aid in the recommendation process.

FTP Server

Though this section has no direct impact on the user experience, the FTP server is required in order to place content onto the system. After files are added, they must further be registered with the personalization. This process adds the content to the SQL content tables, and inserts any known metadata. After this process is complete, the Personalization algorithms can recommend the new content to clients accordingly.

5.3 Content and Ratings

The demonstration contained an extensive collection of content. The corpus contained a mixture of local and international News, such as The Daily Dartmouth and readings of AP reports coupled with online news sources. Local up to date Weather was also included. The corpus contained a large amount of music, particularly local bands, free music, and classical works. We also devised the facility for the user to insert items of her personal collection into the model. The PA would process this content in the normal fashion with the exception that it would not recommend this item to anyone but the individual who uploaded it. We recorded and inserted local events on a daily basis. The server included an insertion page that facilitated this action, and allowed the person who was adding content to choose relevant attributes for it. A representative of our group gathered ads from local businesses. These ads possessed with the category bonuses described in Section 4.1.

One of the comments on the original system was that rating in a piecewise manner was too slow, and the system required a long time to learn the user's tastes. As such, we devised a quicker 'batch' learning technique. This method presented the user with the names of 10 content items. Each was accompanies by a dropbox containing rankings from 2 [Great!] to 0 [Neutral] to –2 [Horrible!]. Users could rank these items and receive yet another page of random items to rate. We found that many users enjoyed this activity and would continue rating items for numerous screens. By the end of this quick exercise, the system had accumulated a significant amount of information on each user and produced significantly more accurate recommendations [according to a closing poll].

5.4 Optimizations

After making the personal radio system publicly available to users, few problems appeared in response to load. We believe this is due Dartmouth College's extensive bandwidth. Usage of the system was steady throughout the demonstration period. The main problem occurred in the middle as the ratings table increased in size. At this point, the recommendation algorithm experience slowdown. We fixed this problem through the user of an index. Future solutions to this problem include the possibility of precalculating results or omitting old data.

Whenever possible, formulas were carried out directly within SQL to optimize access times and minimize the number of required queries [for example, the cosine metric can be computed in 2 SQL queries].

Few of the tables required indices to improve access times. Notably, the User Rating table was used extensively within the Personalization Algorithm. The *content_id* and *user_id* fields were indexed, and this process improved the running time of this algorithm by 2 orders of magnitude. No other tables were of sufficient size or use within this model to warrant extensive optimization.

5.5 Conclusions and Metrics

The demonstration ran from noon on Friday, May 11 to Midnight on Saturday, May 26th. At the end of the demonstration, we collected approximately 40 hours of audio content within the corpus. A total of 210 users had subscribed to the system and listened to a total of 24438 content items. These users rated 15118 contents items. The average user listened to 116 content items. Note however that the median user listened to 40 content items. This disparity between the mean and median indicates that a skew exists among the data were those who liked the system used it extensively while most users exhibited moderate use of the program. Figure 9 below outlines the usage of the system by day. The Internet Explorer client for the Windows Operating System encompasses 86.6% of the system's use. The Other category was primarily composed of RealPlayer, which exists on both Windows and Macintosh platforms.

Figure 9 – Number of content items listened to segmented by Operating System, Browser, and Date.

Date	Windows IE	Windows Other	Mac SoundJam	Mac Netscape	Mac Other	Other	Total	Note
11/May/2001	225	0	5	14	41	4	289	Mailings 100 Initial Users
12/May/2001	140	9	0	0	0	21	170	
13/May/2001	190	0	36	0	0	25	251	
14/May/2001	571	0	106	0	0	22	699	
15/May/2001	330	1	55	17	0	14	417	
16/May/2001	487	0	90	0	0	5	582	
17/May/2001	439	0	0	0	0	19	458	
18/May/2001	639	1	0	1	0	20	661	
19/May/2001	422	0	0	18	0	15	455	
20/May/2001	307	3	21	1	0	56	388	
21/May/2001	1,771	74	106	4	74	198	2,227	Mailings 150 New Users
22/May/2001	3,005	106	359	7	119	107	3,703	Mailings 150 New Users
23/May/2001	1,900	129	128	1	32	68	2,258	Mailings 150 New Users
24/May/2001	2,709	54	44	0	31	74	2,912	
25/May/2001	1,848	20	14	1	47	63	1,993	Mailings 150 New Users
26/May/2001	2,319	12	78	6	6	83	2,504	
Total	17,302	409	1,042	70	350	794	19,967	

We structured the demonstration to test the load on the system and functionality of the algorithms under various usage levels. During the first week, the system was confined to 100 test users. These users were seniors, with a significant number of knowledgeable Computer Science majors represented. During the second week, we emailed 600 people to use the system. These people were a random sampling of freshmen with no pre-assumed knowledge. The Notes column describes the exact dates of the mailings.

As expected, the usage level was significantly higher during the second week. The server performed well under the increased bandwidth. The SQL database functioned correctly, with no significant slowdown due to the increased amount of data that the PA needed to process.

Figure 10 – The number of days that a user accessed the system

# Days	# Users
15	1
10	2
7	1
6	3
5	7
4	8
3	17
2	42
1	129
Total:	210

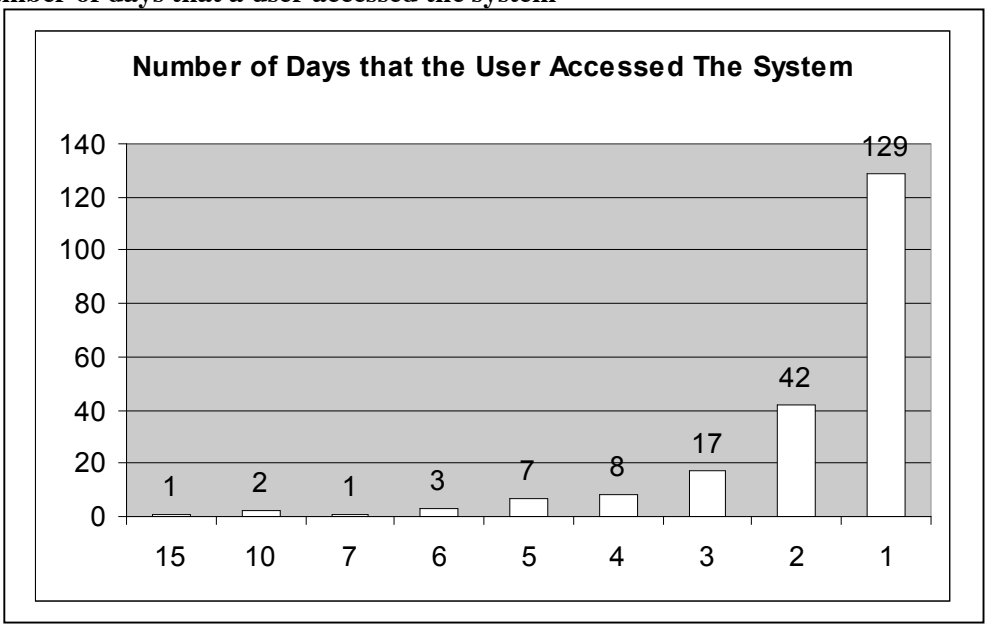
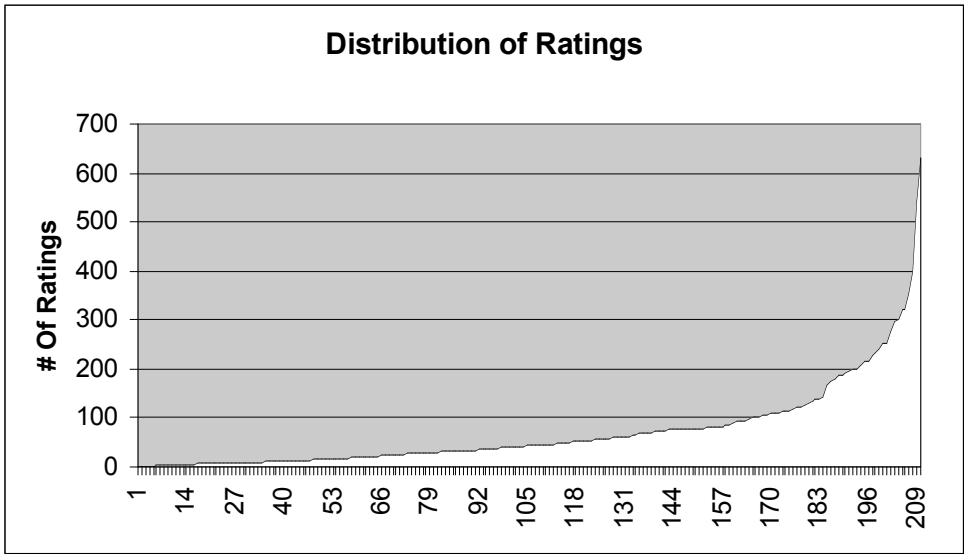


Figure 10 describes the average level of use of the system. The heaviest user used the system on 15 of the days of the demonstration. The usage peaks at the one-day use. One of the difficulties with a system such as this is that it requires significant branding and advertising in order to remind the user to user the system daily. Without frequent reminders, the user may simply forget to come back.

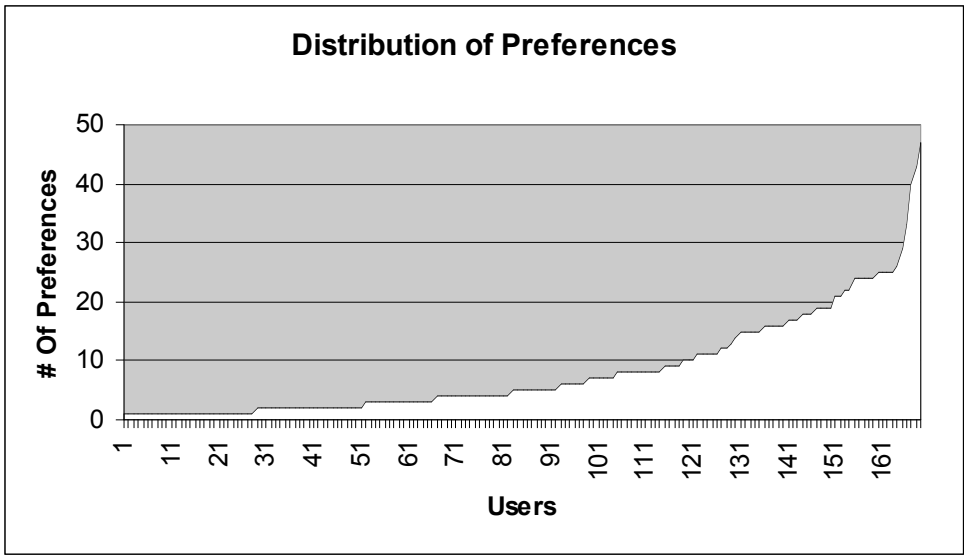
Users were pleased with the rating facility within the system. Users first discovered the batch rating system when they began using the program since it is introduced in the walkthrough for new users. Many users rated dozens of content at the onset. Branding within the web client requested the users continue to rate content as they heard them. Finally, the program could infer ratings as described above. When the demonstration had ended, a large amount of rating information had been accumulated as is evident in figure 11 below. This distribution is consistent with expectations.

Figure 11 – The Number of Ratings by User



Users were also invited to set category preferences when they began using the system. The option to set exclusive preferences was also given. Most users set a few preferences, but by design, the amount was an order of magnitude less than the number of ratings, as is evident in Figure 12 below. This distribution is consistent with expectations.

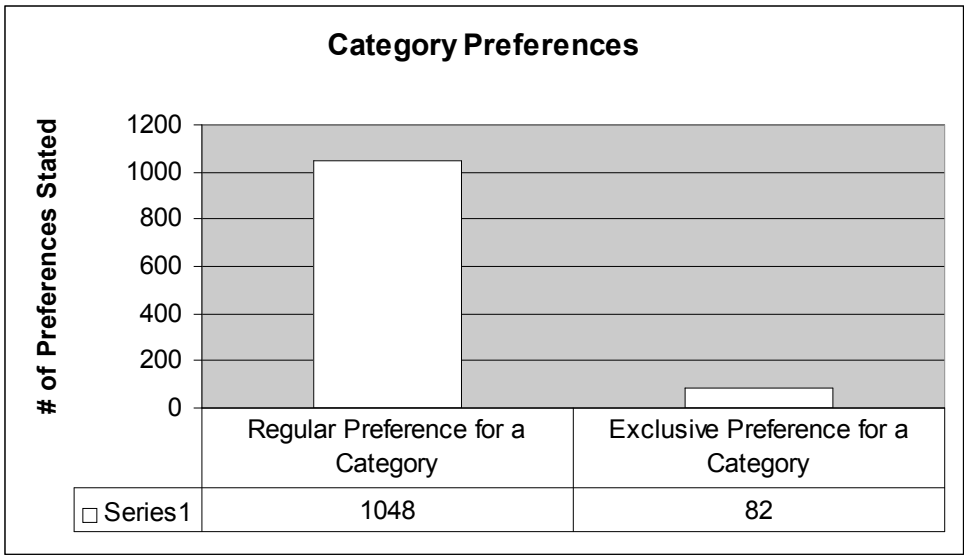
Figure 12 – The Number of Preferences by User



Another issue presented in the algorithmic framework is the concept Exclusive Categories. These allow the user to listen to a specific category or categories of content items exclusively. The users

were presented with the option of doing so within the demonstration. The metrics show that the number of regular preferences exceeded the number of exclusive preferences by nearly 13 times. This finding may be an artifact of the user base. Dartmouth undergraduates formed the bulk of the test users and most of them were using the program as a source for background music while performing another task, rather than as a source for specific news. The distribution of regular versus exclusive preferences is shown in Figure 13.

Figure 13 - The Number of Exclusive Categories by User



A Survey of User Opinions

We conducted a brief poll the day after the demonstration was completed. We emailed an invitation to participate in this poll to all users who had subscribed in the demonstration and listened to at least one content item. Each person was to log into a polling web site that we designed.[16] This site would authenticate their identity using the Dartmouth Name Directory, present them with the poll questions, and record their responses. We extracted the results 24 hours later. There were 44 responses, representing 21% of the user base.

The aggregate responses for the survey are as follows. First, the users were asked whether they approved of the Online Radio. On a scale of 1-5, with five being the highest, the users reported a 4.3 approval rating of the program. Second, in response to the statement "The Online Radio

[16] NetPoll. This program is available at http://www.cs.dartmouth.edu/~marmaros/p

adjusted quickly to my tastes?" the users were presented with a scale of 1-5, with five being "Agree" and one being "Disagree". The users reported a 3.91 agreement with this statement. Thirdly, 17% of the users reported being willing to pay for such a system. The average monthly subscription that these users were willing to spend was $7.87.

The final two questions were freeform responses, and asked what aspects the users liked most and least about the system. The top three aspects that the users enjoyed were, in order of popularity:

1) The diverse selection of content
2) The ability of the system to quickly adjust to the user's tastes
3) The ability to skip content.

The three aspects of the system that the users enjoyed least about the system were, in order of popularity:

1) Repetition of content items that the user approved of
2) Limited selection of content
3) Missing a genre that the user wanted

We will now discuss the negative comments and propose solutions. The first reflects misweighting within the system. The Personalization Algorithm notices if a user has listened to a content item lately and compensates accordingly. This process is described in the Frequency Rating set of dimensions in Section 4.1. However, this critique demonstrates that the coefficient attached to a recently played song was not sufficiently negative in comparison to the coefficients on the other sets of dimensions. Altering this ratio will cause content items to repeat less often.

The second negative comment was that the program had a "Limited selection of content." The trivial solution would be to add more audio data to the corpus. However, it is also possible that this complaint is correlated with the first negative aspect. It is possible that the PA adjusted too quickly to the user's tastes and then would overplay a specific segment of the corpus. The corollary is that it would underplay the rest of the corpus, leaving the user with the impression that the selection of content in the corpus was smaller than it truly is. The solution to this problem would also be a reexamining the ratios of coefficients within the dimensions of the Personalization Algorithm.

The third negative aspect was that the corpus was "missing a genre that the user wanted." Once again, the naïve aspect is to simply add this genre. However, given that a significant portion of the popular music today is copyrighted, it was not possible for this demonstration to distribute several genres of music that do not have free versions. Once the copyright issues have been resolved, then it will be possible to add missing genres to the corpus, and remedy this problem.

6. Future Extensions

There is much room for future research in this field. The three main areas are Algorithmic Research, Device Research, and Security Research.

Algorithmic Research describes the process of updating the Personalization Algorithm. There are numerous aspects of the algorithm that prompt further study. For example, it was necessary to choose coefficients for the dimension categories within the information vectors within algorithm. Whenever possible, we attempted to keep the values roughly equal to one. However, it is likely that altering the ratios of the coefficients between the various categories can optimize the algorithm. This area of research would be fruitful in resolving the problems of over-focusing and over-repeating, which were the top problems mentioned in the user survey.

Furthermore, it is likely that the coefficients within the Personalization Algorithm adjust depending on usage levels within the system. For example, as more users join the system and the body of ratings on content items expands, the popularity dimension must normalize itself to prevent it from dwarfing the other dimension. Furthermore, as the tenure of a particular user within the system increases, that user may benefit from having her dimensional coefficients altered accordingly.

Finally, Algorithmic Research could encompass methods of optimizing the above Personalization Algorithm. It may be determined that certain dimensions do not contribute to significant utility estimation and can be omitted. A more efficient algorithm can be designed with the same properties.

Device Research encompasses research for extensions to this project on different devices. The system can be implemented on devices with bandwidth limitations as described in Section 4.2. Numerous wireless and automotive-based devices are presently in design for release in the near

future, and major automobile manufacturers have endorsed the premise. There will shortly be a new generation of devices to test this system on.

Furthermore, additional Device Research can be done into the exact caching algorithms to optimize benefit / memory constraints. This is particularly relevant on the extensive assortment of wireless connections that are possible in the current environment. Finally, it is important to determine the optimally priced sub-PC hardware device to implement this system.

Security Research encompasses the devising and implementing Digital Rights Management. Numerous advances and legislature will undoubtedly appear in the near future to control the dissemination of copyrighted material. The algorithms that must be used will thus be dictated by legal and market forces.

7. Conclusion

We devised these algorithms and created this two-week demonstration to accomplish three goals:

1. To test the accuracy of a Personalization Algorithm
2. To test the limiting hardware, software, and network load of this system
3. To provide a positive and intuitive user experience

We believe that we have succeeded on all accounts. First, users report that the Personalization Algorithms were able to accurately zero in on the user's preferences. Indeed, the algorithms may have worked too accurately, thereby causing repeats once it located an optimal set of items.

Second, the network and hardware loads of the system were well within operating parameters and barely dented the campus bandwidth. Software limitations in the recommendation engine were the key bottleneck to the system, but this can be remedied through optimizations. An additional software dilemma was the difficulties of setting up the system in a true platform independent manner. While feasible, this process is not trivial for the novice user.

Third, users report high satisfaction with the system and beyond setting up the program on various non-standard program, we received no questions on how to operate the system. The user interface was intuitive and trouble-free to a novice user.

In summary, this project was a success in recommending discrete audio content to a large number of users. Both skilled and novice users reported high levels of use and satisfaction with the system, and it received higher levels of use than was originally intended. The servers, database, and network received the extra traffic without any difficulties. The Personalization Algorithms functioned as intended, and users were pleased with the results. We hope that this work leads to more accurate and scalable approaches to information customization.

Testimonials:

"hi visited your site today - GREAT STUFF!! keep it up" '04

"wow.... great idea! :)" '04

"this is the coolest thing i've ever seen!" '04

"this program is pretty awesome." '04

"I really like your radio project. nice work =)) thanks!!" '04

8. Bibliography

[AB84] M. Aldenderfer and R. Blashfield. “Cluster Analysis.” Sage, Beverly Hills, 1984.

[APR99] Javed Aslam, Katya Pelekhov, Daniela Rus. “A Practical Algoritm for Static and Dynamic Information Organization.” In Proceedings of the 1999 Symposium on Discrete Algorithms, Baltimore, MD.

[APR00] Javed Aslam, Katya Pelekhov, Daniela Rus. “Using Star Clusters in Filtering.” Dartmouth College, 2000.

[ARR00] Javed Aslam, Fred Reiss, Daniela Rus, “Scalable Information Organization,” Dartmouth College Senior Thesis, 2000.

[AT00] M. Alghoniemy, A.H. Tewtik. “Personalized Music Distribution.” Acoustics, Speech, and Signal Processing, 2000. ICASSP ’00 Proceedings. 2000 IEEE International Conference on. pp 2433-2436.

[BMO94] J. Brassil, S. Low, N. Maxemchuk, and L. O’Gorman. “Electronic marking and identification techniques to discourage document copying.” in Proc. Infocom ’94, pp. 1278-1287.

[CKLS97] Ingemar J. Cox, Joe Killian, F. Tomson Leighton, and Talal Samoon. “Secure Spread Spectrum Watermarking for Multimedia. ” IEEE Trans. on image Processing, 6(12): 1673-1687, 1997.

[CMB00] Ingemar J. Cox, Matt L. Miller, and Jeffrey A. Bloom. “Watermarking applications and their properties.” Information Technology: Coding and Computing, 2000. Proceedings. International Conference on , 2000. Pages 6-10.

[HR00] F. Hartung, F. Ramme. “Digital Rights Management and Watermarking of Multimedia Content for M-Commerce Applications.” IEEE Communications Magazine, Volume: 38, Issue: 11, Nov. 2000.

[HS86] D. Hochbaum and D. Shmoys. “A unified approach to approximation algorithms for bottleneck problems.” Journal of the ACM, no. 33, pp 533-550, 1986.

[JD88] A. Jain and R. Dubes. “Algorithms for Clustering Data.” Prentice Hall 1988.

[KM00] L.Khan and D. McLeod. “Audio Structuring and Personalized Retrieval Using Ontologies.” Advances in Digital Libraries, 2000. Proceedings. IEEE. Pp 116-126.

[Rij79] C.J. van Rijsbergen. “Information Retrieval.” Butterworths, London, 1979.

[Wil88] P. Willett. “Recent Trends in Hierarchical Document Clustering: A Critical Review.” Information Processing and Management, 24 (5) 577-597, 1988.

9. Acknowledgements

We thank Frank Yoshida who acted as advertising manager in the demo. Robert Strong who aided in newscasting, Wayne Cripps and Arne Grimstrup for generously setting up the required software and accounts, Rodney Jacobson for his efforts on the Windows client, Professors Farid and Aslam for their help in proving the correctness of the algorithm, Professor Kotz for his time and involvement, and especially Professor Daniela Rus for her time, advice, and encouragement.

Appendix 1 A Brief Profitability Analysis and Business Plans

There are four principle sources of revenue that are possible within this model:

1) Commissions and Cobranding

Pros: Potentially extremely profitable

Cons: Overt commercialism may alienate the users

Most successful commercial websites have experimented in cobranding their product. They allow other sites to become affiliates or associates which are licensed to sell their products in exchange for a commission. The PA system is no different. While the traditional radio model allow the content to be played at random, it is perfectly reasonable to sell a CD containing the music alongside its radio counterpart. This process allows the user to buy the content if they enjoy it enough, and allow them to play it at anytime instead of at random under the current model.

Commissions would be given to the RIAA, the artist, or whoever owns the copyright to the content item.

2) Selling aggregate data

Pros: Few ethical issues, not hard to implement

Cons: Low margin

There are two distinct groups that purchase aggregate data from a content intermediary: advertisement firms and manufacturing firms.

Traditionally, advertisement companies have thrown their campaign into the waiting jaws of the public. Few metrics were available to determining if the campaign was successful. The primary method was simply to analyze sales patterns and attempt to find a correlation. This method was prone to error, and that fact was known to both the ad companies and their clients. The former could always argue that the ad had a long-term impact in building brand awareness and the lack of an immediate spike in demand was a direct result. On the flipside, they were quick to argue that any surge in demand was due to a campaign as opposed to random chance.

The web revolutionized this model with the advent of real-time feedback of ads. Since customers now had the opportunity to act on the ad immediately, the makers of the ad could similarly determine the effect of the ad immediately. This mechanism for immediate feedback allowed for improved accountability within the realm of advertising, and could facilitate campaigns that are more profitable since ineffective ones would be detected quickly and discontinued.

On the other side to this discussion, manufacturers are willing to pay for aggregate product purchasing information as a form of forecasting. These corporations are extremely dependent on the whims of the market. For example, if 128 megabyte RAM chips happen to be extremely popular this week, a semiconductor manufacturer would like to know about it immediately. This will allow them to retool their production line to produce more (or less) chips accordingly. This will allow them to avoid having excess inventory and having to liquidate their supply, or not having sufficient stock product in a boom time. In the past, these companies have needed to poll the market directly or process huge amounts of information from varied distributors in order to glean this forecasting knowledge.

Aggregate data is easily gathered based on listening and purchasing habits. As an added bonus, the public is relatively unperturbed by having their information collated and anonymously aggregated. As such, this business practice does not raise ethical qualms or promote customer backlash. On the other side of the equation, aggregate data fetches a relatively lower price. The primary method of adding value is to provide a continual supply of data in an up to date and constant manner, for example, a day by day or hour by hour dependent on the particular product and industry. A second method of adding value to this information commodity is to provide tools to aid in the analysis of the data.

3) Selling personalized data

Pros: High margin

Cons: Ethical minefield

Even more valuable than aggregate data is personalized data. Companies are currently attempting to learn everything about their customers in the hope of better tailoring products and increasing sales. As such, many corporations would pay generously for data about individuals. As part of the functionality of the PA, it collects and categorizes a significant amount of user information. This information could be packages and sold to whichever parties would like to purchase it.

The problem with this is ethical. Users wish to maintain their privacy and seek to avoid dissemination of their personal information. If this practice is followed, users may deliberately shun the system or avoid entering their tastes to avoid disseminating them. This area is legally challenging since numerous legislature limit the distribution of personal information without prior notice.

4) Targeted advertising

Pros: High Margin

Cons: Overt commercialism may alienate the users

The current incarnation of the program includes ads from local businesses. It would be possible to ad many more such ads in the future from a host of sources. Thus would provide a proven revenue stream, since this mechanism has long been used by the traditional radio industry. However, the new model of personalization would allow for better targeted advertisements than

traditional radio which would increase the revenue yield. However, users generally are not in favor of advertising and this aspect may cause users to shun the system.

Appendix 2 The Post Mortem Survey

We distributed a survey the day after the demonstration was completed. We emailed it to all of the users of the system. The eight questions are listed below.

```
1) Did you like the Online Radio?
      5 (I liked it a lot)
      4
      3 (I am Neutral)
      2
      1 (I did not like it)

2) The Online Radio adjusted quickly to my tastes?
      5 (Agree)
      4
      3 (Neutral)
      2
      1 (Disagree)

3) How often did you listen to this program?
      Several hours each day
      Less than one hour per day
      2-3 times per week
      Once per week
      Less than once per week
      Never

4) Would you pay for a product like this?
      Yes
      No

5) If you answered yes to the above question, how much per month?

6) What was your favorite aspect about the Online Radio?
```