Flycasting: On the Fly Broadcasting

James C. French and David B. Hauver
Department of Computer Science, University of Virginia

Abstract

In recent years, the popularity of online radio has exploded. This new entertainment medium affords an opportunity not available to conventional broadcast radio: the instantaneous listening audience can be known, or what is more important, the musical tastes of the current listening audience can be known. Thus, it is possible in the new medium to tailor the playlist in real-time to the musical tastes of the listening audience. We summarize our method, termed *flycasting*, for using collaborative filtering techniques to generate a playlist in real-time based on the request histories of the current listening audience.

Introduction

In recent years, online radio has exploded from being an interesting technological experiment to an entertainment medium considered important enough to be measured by Arbitron, the company primarily responsible for compiling traditional broadcast radio ratings [Marx]. In fact, a recent study by Arbitron shows that 20% of Americans have listened to an online radio station – a percentage that has more than tripled in the last two years [Rose]. Furthermore, online radio has brought a new definition to the term "competitive market." In a traditional radio market, a limited number of radio stations compete for the same local audience. This relationship does not hold for the online radio market. Listeners in Tokyo have no more difficulty listening to a station broadcasting from New York than a local station. If they don't like what they hear there are literally thousands[1] of other stations to choose from. Consequently, it is vital that an online radio station find a way to connect to its audience and retain its listeners. This paper summarizes our approach, flycasting, that strives to meet this goal by creating a playlist that matches the musical preferences of an online radio station's current listeners. To the best of our knowledge, no online radio station has implemented any comparable method of playlist generation. A complete description of the flycasting technique can be found in [Hauver].

[1]As reported in January 2001 (http://www.arbitron.com/nycu_archive/1_9_01_1337.htm), Arbitron Webcast Ratings covered 2,233 stations and channels.

The underlying assumption behind our approach is that radio station listeners tend to request the songs that they enjoy hearing the most. Steven Snyder, the CEO of Net Perceptions, summed up the situation best when he stated that "behavior is very often a more accurate predictor than … explicitly stated preferences [Sullivan]." By providing a web page from which listeners can request songs, an online radio station can gather data concerning the individual musical preferences of its listeners. Combining this knowledge with that of who is currently listening, an online station can create a playlist that is directly targeted toward who is listening at any given time. Such a targeted playlist is an extremely worthwhile goal – hopefully, listeners will stay tuned in longer since they are hearing music that they enjoy. By pleasing its listeners, an online radio station enters into a mutually beneficial relationship.

Online Radio Station Background

Before we begin discussing the details of our method to create an adaptive playlist, we first need to lay out our model of an online radio station and its associated entities. To begin with, there are three primary entities associated with an online radio station: *artists*, *songs* and *requesters*. *Songs* are music files capable of being broadcast by the station. Songs can also be requested by people visiting the station's web page People who request songs become known as *requesters*. We assume that all requests can be identified as stemming from their requester. Finally, in our simplified model, each song is performed by a single *artist*. A more refined model would associate a set of performers with each song. We make no attempt to decompose groups of artists into individual performers, preferring instead to regard a group as an artist for simplicity in this initial investigation.

Every time that a song is successfully requested, the request is stored in a *request log*. The information associated with a request includes the requester making the request, the song being requested, the artist that performs the song and the time that the request is made. All of the requests made by a single requester compose his or her request history. It is possible for this request history to contain multiple requests for the same song.

A request can be either successful or unsuccessful. Due to broadcasting license

restrictions on the number of times artists and songs can be repeated within a time frame, a certain number of songs may be ineligible to play at any given time. If the song requested is eligible to be played and the *playlist* is not too long, the request is successful and the song is added to the end of the playlist. A request is stored in the request log only if it is successful. The request entries in the station log provide the data that the flycasting algorithm uses to generate songs for the playlist.

Finally, a *listening audience* consists of all the individuals that are listening to the station's broadcast at a given time. We assume that all requesters present in a listening audience can be identified. This allows us to gather information concerning the musical preferences of some members of the listening audience. Because we can only obtain information about the requesters, when we refer to a listening audience, we are only referring to the members that are requesters. It should also be noted that the listening audience is a dynamic set with individuals tuning in and out in an unpredictable way. The current listening audience is defined to be the set of requesters who are listening to the station at the instant that the playlist generation process begins. The eventual goal of our analysis will be to pick an appropriate song to play based on the consensus musical taste of this audience.

Overview of Flycasting

Our overall goal is to lay out a method that generates a playlist based upon the musical tastes of the people currently listening to an online radio station. As the audience changes, the type and style of songs being played should also change to match the audience's consensus tastes. We call such a method of playlist generation *flycasting*. This term reflects that broadcasting decisions are made on-the-fly and also notes that an attempt is being made to lure requesters into staying members of the audience.

Our flycasting algorithm involves five phases:

1. Translate the request histories of all requesters into ratings for artists.
2. Predict ratings for each artist that a requester has never requested.
3. Determine what artists are the most popular among the listening audience.
4. Determine what artists are similar to the final artist on the playlist.
5. Select a song to play that is performed by an artist that is both popular among the listening requesters and similar to the artist that precedes it.

The first phase is fairly straightforward and can be computed ahead of time from entries in the request log. The second phase, ratings prediction, is performed only for the members of the current listening audience. *Collaborative filtering* [Goldberg] is used to make educated guesses about artists that a requester has never requested. The third phase uses the predicted ratings of requesters and the concept of *popularity* to determine what artists will be best received by the listening audience. The fourth phase introduces the notion of artist *similarity* and ensures that the generated playlist is *coherent*. We define coherency to simply mean that consecutive songs in the playlist are performed by artists that are sufficiently similar. Much of this phase can also be done ahead of time. Finally, a song is selected to play during the fifth phase. The general procedure is to select a song that is performed by an artist that is both popular and similar to the artist that performs the song preceding it in the playlist.

It should also be noted that artists are being used as the basis for musical preferences instead of songs. By using artists as the basis of musical preferences, relationships can be discovered that may have been missed by comparing songs alone. The fact that songs performed by the same artist tend to be similar is the second reason for using artists instead of songs for the basis of requester preferences.

Flycasting Phase 1: Translating Request Histories Into Ratings

The first step involved in the flycasting process is to translate request histories into ratings for artists. These ratings are used later on when attempting to quantify which artists are the most popular among the current listening audience.

Two factors are used to calculate a requester's rating for an artist. First, we consider the total number of requests made for songs performed by the artist. However, an exponential decay is used for each subsequent request's influence on the overall rating. Each request for a song by the artist increases the requester's overall rating for the artist, but each subsequent request increases the rating less than the previous request. Consequently, the "most important" request is the first time a request is made for a particular artist.

The second factor used is the number of *repeated requests* made for the artist. A repeated request should increase a requester's rating for an artist, but not as much as a *unique request* for a song not previously requested. Given two requesters with an identical number of requests for songs by a particular artist, it seems intuitive that the requester who

requested a wider range of songs should have a higher rating for that artist. Consequently, repeated requests for the same song by an artist are considered to be worth only a fraction of what each unique request is worth.

Flycasting Phase 2: Predicting Ratings

At this stage, the request histories of all requesters have been translated into ratings for artists. Although one could conceivably use the artist ratings given by the current listening audience to choose an artist to play, there is still more information that can be gathered concerning the preferences of the listening requesters. The application of collaborative filtering techniques lets us make predictions about the artists that have not been directly rated by a requester. By making predictions concerning the artists not rated by each member of the current listening audience, we can gain even more information to guide the song selection process.

Collaborative filtering refers to techniques that match items to users on the basis of whether similar users found the items useful. Although it was pioneered in the Tapestry system at the Xerox Palo Alto Research Center to filter news articles [Goldberg], it has been used for a wide range of purposes, from search engines [Direct Hit] to movie recommendation systems [Movie Critic]. Nearly any collaborative filtering technique can be used to make ratings predictions for artists not already rated. We have chosen to use the Mean Squared Difference (MSD) Algorithm as presented by Shardanand [Shardanand] as the basis of our collaborative filtering approach. It is an instance of a *neighborhood-based* collaborative filtering algorithm [Herlocker].

The predicted ratings generated from the collaborative filtering process are stored in the Requester-Artist prediction matrix. In practice, this entire matrix will not need to be calculated. The only values that are needed are those that correspond to the actual and predicted ratings of the current listening audience.

Flycasting Phase 3: Determining Popular Artists

The popular artists are extracted from the Requester-Artist prediction matrix. Based on the predicted ratings of the current listening audience, a set of artists is chosen that is considered to be *popular* among the current listening audience. These artists are simply those that have an average rating among the current listening audience above some threshold.

Flycasting Phase 4: Determining Similar Artists

At this stage, it may seem that the best thing to do is to select an artist from the popular artists and play a song by that artist. However, this could lead to a playlist that is not *coherent*. For our purposes a coherent playlist is comprised of similar artists. Imagine a case where the listening audience is split between rap fans and bluegrass fans. The set of popular artists will surely include artists from both genres. Mixing the two into a playlist will create a disjointed playlist that is unlikely to satisfy either segment of the audience for very long. A better choice would be to play songs from only one of the genres and lose the other portion of the audience.[2]

For this reason, a set of artists is generated that is considered to be *similar* to the artist that is at the end of the playlist. Playing songs by these artists will lead to a more coherent playlist. The similarity is determined relative to the final song on the playlist. If an entire playlist is being generated, the set of similar artists will need to be recalculated every time a song is placed at the end of the playlist.

The concept of a coherent playlist necessitates a method for determining how similar two artists are to each other. We use two factors to determine this. First, we examine the differences in the actual ratings given to the two artists by each requester that rated both of them. It should be noted that we care only about the differences in the ratings and not what the requesters actually rated the two artists. Having a passing curiosity in the two artists or having a deep love for the two artists will provide the same information. As the average difference between requester's ratings for two artists decreases, the measure of similarity between the two increases.The second factor that we use as a measure of artist similarity is the degree of overlap between the number of requesters who actually rate the two artists. If we are trying to determine how similar artist *a2* is to artist *a1*, it seems intuitive that as the percentage of requesters who rated both *a1* and *a2* increases, the measure of similarity should also increase. These two factors are combined to provide a measure of the dissimilarity between two artists. This measure is used to determine the set of similar artists.

[2] Eventually we intend to consider ways to split the audience into separate segments by genre and flycast to those segments independently. This approach is clearly superior to losing an audience segment.

Flycasting Phase 5: Selecting a Song

It is now relatively trivial to choose a song to add to the playlist. We have a set of artists PA that are popular among the currently listening requesters and we have a set of artists SA that, if added to the playlist, would lead to a coherent playlist. With a few exceptions, we can simply choose an artist from the intersection of the two sets and play a song by that artist. However, broadcasting licenses suddenly matter at this point. For example, online stations in the United States are required by law not to repeat an artist or a song more than a certain number of times within a given time period [Digital, RIAA]. In effect, a few artists and songs are ineligible to play at any given time due to these restrictions. When selecting a song, one must also ensure that it is actually eligible for playing.

All that remains is the actual selecting of a song from the eligible set *E* to add to the playlist. Several approaches could be taken. First, a song could be picked at random. However, this would give the artist with the most eligible songs the greatest chance of being played next. A second approach would be to first select an artist from the pool of artists that actually have songs in *E* and then choose a song by that artist. This prevents an artist from becoming favored simply because more songs by that artist are in the system. A third approach would be to select the artist with the highest average rating among the listening audience from the pool of artists that have songs in *E*. A song by this artist could then be chosen from *E*. The problem with this approach is that playlist generation becomes deterministic on the artist scale if the listening audience is static. The highest rated eligible artist always gets added to the playlist. Then the next highest rated artist that is similar to the highest rated one will get added. When the highest rated artist becomes eligible to play again, a cycle will begin, with the playlist continually cycling through the same artists in the exact same order. Our current approach for selecting a song is to randomly select an artist first and then randomly select a song by the artist.

One important case that must be considered is when the intersection between the popular artists *PA* and the similar artists *SA* is empty. In this situation, choosing a coherent playlist is contradictory to selecting a song that will be well received by the current listening audience. Relaxing the thresholds for similarity or popularity does not seem to help – the new overlap will consist of artists that are not very popular or similar. Our solution is to simply choose an artist from the set of popular artists and then choose an eligible song by that artist. This treats an empty intersection as a "reality check" that shows that the requirement of coherency is slowly driving a playlist away from an audience's taste. It needs to be re-centered by briefly ignoring coherency and catering only to musical preferences.

Conclusions

We described a novel scheme for ranking artists based on listener request histories and showed how this scheme could be integrated into an online radio. Our approach uses a collaborative filtering strategy for the purpose of generating a playlist that is adapted to the musical tastes of the "current" listening audience. We call this strategy for targeted broadcast flycasting. A complete description has been omitted due to space limitations. A full account can be found in [Hauver]. We are currently evaluating the quality of the algorithm by means of listener feedback.

The flycasting technique proposed here can be smoothly integrated into an operational online radio station. In fact, the technique could be adapted to partition the current listening audience by genre for the purpose of selective multicast of songs to groups with more similar tastes than would be found in a large diversified listening audience. In the limit, it would be possible to cater to individuals. This provides the opportunity to simultaneously play different songs for different interest groups.

Privacy concerns can be addressed by anonymizing request histories. It is not necessary to know the identities of the listeners in the current audience, only their musical tastes. On the other hand, it would be entirely possible to enable ad placement via this mechanism provided that listeners agreed and authorized such placement.

Acknowledgement

This work supported in part by DARPA contract N66001-97-C-8542.

References

1. "About Direct Hit." 13 Feb. 2001 <http://www.directhit.com/about/>.
2. "Digital Millennium Copyright Act." 105 Congress, 2nd Session. Report 105- 796, Section 405, Oct. 1998: 34. 13 Feb. 2001 <http://www.loc.gov/copyright/legislation/hr2281.pdf>.
3. Goldberg, D., D. Nichols, B. M. Oki, and D. Terry. "Using Collaborative Filtering to Weave an Information Tapestry." Communications of the ACM, Dec. 1992: 61-70.
4. Hauver, D. B. and J. C. French. "Flycasting: Using Collaborative Filtering to Generate a

Playlist for Online Radio." Submitted to the *International Conference on Web Delivering of Music* (WEDELMUSIC'2001).

5. Herlocker, J. L., J. A. Konstan, A. Borchers and J. Riedl. "An Algorithmic Framework for Performing Collaborative Filtering." *Proc. of the 22nd International Conference on Research and Development in Information Retrieval (SIGIR'99),* Berkeley, CA, August 1999, pp. 230-237.
6. Marx, A. "Measure Streaming? Keep on Dreaming." Internet World 1 Nov. 1999: 43-44.
7. "Movie Critic FAQ." 13 Feb. 2001 <http://www.moviecritic.com/faq.html>.
8. Resnick, P., N. Iacovou, M.Suchak, P. Bergstrom and J. Reidl. "GroupLens: An Open Architecture for Collaborative Filtering of Netnews." Proceedings of the Conference on Computer Supported Cooperative Work, 22-26 Oct. 1994, Chapel Hill, NC: 175-186.
9. "RIAA/Webcasting FAQ." 13 Feb. 2001 <http://www.riaa.com/ Licensing-Licen3a.cfm>.
10. Rose, B. and L. Rossin. "Internet Study V: Startling New Insights About the Internet and Streaming." Arbitron/Edison Media Research, 21 Sep. 2000. 13 Feb. 2001 <http://www.arbitron.com/studies/internet_study_v.pdf>.
11. Shardanand, U. "Social Information Filtering for Music Recommendation." Master's Thesis, Massachusetts Institute of Technology, Sep. 1994.
12. Sullivan, E. "The Collective Mind is a Terrible Thing To Waste." PC Week, 21 Apr. 1997: 29.