MIXTAPE: DIRECTION-BASED NAVIGATION IN LARGE MEDIA COLLECTIONS

João Paulo V. Cardoso **Luciana Fujii Pontello** **Pedro H. F. Holanda**
Bruno Guilherme **Olga Goussevskaia** **Ana Paula C. da Silva**
Computer Science Department, Universidade Federal de Minas Gerais (UFMG), Brazil
jpcardoso@ufmg.br, lucianafujii@dcc.ufmg.br, holanda@dcc.ufmg.br,
brunoguilherme@dcc.ufmg.br, olga@dcc.ufmg.br, ana.coutosilva@dcc.ufmg.br

ABSTRACT

In this work we explore the increasing demand for novel user interfaces to navigate large media collections. We implement a scalable data structure to store and retrieve similarity information and propose a novel navigation framework that uses geometric vector operations and real-time user feedback to direct the outcome. In particular, we implement this framework in the domain of music. To evaluate the effectiveness of the navigation process, we propose an automatic evaluation framework, based on synthetic user profiles, which allows to quickly simulate and compare navigation paths using different algorithms and datasets. Moreover, we perform a real user study. To do that, we developed and launched Mixtape [1] , a simple web application that allows users to create playlists by providing real-time feedback through liking and skipping patterns.

1. INTRODUCTION

Internet cloud and streaming services have become the state-of-the-art in terms of storage and access to media collections. Even though the storage problem of media collections seems to have been practically solved with cloud-based applications, a challenge still remains in conceptualizing and developing novel interfaces to explore them. User interfaces are expected to be intuitive and easy, yet flexible and powerful in understanding and delivering what users expect to see.

In this work we propose a framework that uses real-time user feedback to provide direction-based navigation in large media collections. The navigation framework is comprised of a data structure to store and retrieve similarity information and a novel navigation interface that allows users to explore the content of the collection in a personalized way. We begin by focusing on the music domain, because the intrinsic usage pattern behind listening to music is favorable to the design and verification of a dynamic real-time feedback based system.

We define media item-to-item similarity based on user-generated data, assuming that two items are similar if they frequently co-occur in a user's profile history. Media co-occurrence information is increasingly available through many online social networks. For example, in the domain of music, such usage information can be collected from Last.fm, a social music site. Collected co-occurrence data is usually sparse (not all pairs of items will have co-occurred at least once in the collected dataset) and nevertheless might occupy a lot of memory space ($\Omega(n^2)$, where n is the size of the collection). To guarantee $O(n)$ space complexity and $O(1)$ query complexity of all-pairs similarity information, we transform the collected pairwise co-occurrence values into a multi-dimensional Euclidean space, by using nonlinear dimensionality reduction [21].

Our main contribution is a novel randomized navigation algorithm, based on the geometry of the constructed similarity space. Each navigation session is modeled as a Monte Carlo simulation: given a starting item and a set of close neighbors in the similarity space, each neighbor is assigned a probability of being the *next* current item. If the returned next item is not quite what the user wants to see, they can skip it, so the previous item is used as the seed again. To define these probabilities, we propose a geometric vector-based approach, which explores the notion of direction, using user feedback and the Euclidean distances between items to establish a concept of "direction inertia", which creates a tendency for users to "keep going" in the direction of the items they enjoy and "turn away" from items, or regions, they don't like.

The evaluation of the resulting system is twofold. First, we propose an automatic evaluation framework, based on synthetic user profiles, which allows to quickly simulate and compare navigation paths using different algorithms and datasets. We also propose two basic metrics: number of skips per like ratio and smoothness of consecutively accepted items in a navigation session. Second, we evaluate real-user interaction with the system. To do that, we developed and launched Mixtape, a simple web application that allows users to create playlists by providing real-

[1] www.projectmixtape.org

time feedback through liking and skipping patterns. Overall, we analyzed over $2,000$ simulated and $2,000$ real-user navigation sessions in a map comprised of more than $62,000$ songs. Besides analyzing quantitative parameters, such as the proportion of skipped to accepted songs and the smoothness of the generated trajectories, we gathered feedback left by users and analyzed what they expect and appreciate in a media navigation system.

2. RELATED WORK

A closely related line of research to this work is automatic playlist generation. There are techniques that use statistical analysis of radio streams [4, 5, 15, 22], are based on multidimensional metric spaces [2,4,9,13,16,17], explore audio content [3,8,14,23], and user skipping behavior [18]. In particular, Chen et al [4] model playlists as Markov chains, which are generated through the Latent Markov Embedding (LME) machine learning algorithm, using online radio streams as a training set. We use this algorithm as a baseline in our experiments. The idea to embed co-occurrence information into a multi-dimensional space has been explored before, e.g., in [1,2,9,13], where the authors focus mostly on visual exploration of a collection. The idea to use skipping behavior to generate playlists has been explored in [18], however, the presented algorithms do not scale to large collections. Our work goes beyond playlist generation, providing a real-time flexible navigation interface that receives immediate user feedback through skipping behavior to guide the user within the music collection towards directions chosen on-the-fly.

3. NAVIGATION FRAMEWORK

Our goal is to design a media navigation framework comprised of two main components: (1) A scalable data structure to store and retrieve item-to-item similarity information; (2) Directed-based navigation functions, that take the current item and user feedback in real time and return the next item; moreover, we want the navigation output to be computationally efficient and nondeterministic, so the user can be surprised with new items in each navigation sequence.

3.1 Item-to-item similarity representation

In this work, we use the assumption that similarity between two items can be deduced by analyzing usage habits of a large number of media users. More specifically, we assume that the more often two items co-occur in the same user's profile, the more similar they are. So we define pairwise similarity between two items by using cosine similarity: $cos(i,j) = coocc(i,j)/\sqrt{occ(i)occ(j)}$, where $coocc(i,j)$ is the number of co-occurrences between two items and $occ(i)$ the individual occurrences in the users' profiles.

Since co-occurrence data is typically sparse, i.e., only a few pairwise similarities are known, we applied the Isomap method [21], which extends classical multidimensional scaling (MDS) [6] by incorporating the geodesic distances imposed by an (intermediate) weighted graph. We defined the weight of an edge as the complement of the cosine similarity, $(w(i,j) = 1 - cos(i,j))$ and built a graph G with these weights.

To generate the map we calculated the complete nXn distance matrix from G and then applied the classical MDS algorithm in this matrix. Building a new d-dimensional Euclidean space such that $d << n$. The final space is a nXd matrix. Note that, for larger datasets one can use approximate algorithms, such as LMDS or LINE [7, 20].

3.2 Navigation functions

In order to guide the navigation process, a navigation session is treated as a run of a Monte Carlo simulation, in which the choice of the next item depends on the current item and a probability function that assigns different probabilities to each of its neighboring nodes. Given a starting item, the navigation system retrieves the set K of its nearest neighbors in the Euclidean space, and uses them as candidates to be the next item. Once an item $k_i \in K$ is chosen to be next, users can provide immediate feedback to the system by accepting or skipping it explicitly, through user interface feedback, or implicitly by *skipping* it. In case the new item is accepted, it becomes the current item, and the process starts again. The probability function should have a strong influence on the overall outcome of the navigation.

Parameter $|K|$ is used to vary the size of each "step" of the navigation process. It can be configured as a constant, or to be variable. In our experiments, good results were achieved using exponentially growing step size:

$$|K| = \begin{cases} 2|K|, & \text{if the previous item was skipped} \\ |K_0|, & \text{otherwise,} \end{cases}$$

where $|K_0|$ is configurable minimum neighborhood size. In our experiments we used $|K_0| = 10$ and $|K| \leq 640$.

Map navigation: We start with the following basic approach, to which we refer as Map, that explores the idea that users prefer to navigate through items that are close to each other in the Euclidean space. We define the probability of node $k_i \in K$ to be *next* as:

$$P_i^{nextMap} = \begin{cases} 1/|K|, & \text{if } k_i \in K; \\ 0, & \text{otherwise,} \end{cases}$$

Vector navigation: Vector navigation explores the notion of direction of navigation, assuming users would like to travel through different regions in the space. To do so, it treats the possible steps in the space as vectors. The *hop vector* $\vec{ab}$ of any given hop from item A to item B can be derived from the straight line between them (see Figure 1.1).

As the navigation progresses, the system keeps a *direction vector* $\vec{V}$, which is recalculated after every hop. This vector represents the directions in which the system has recently moved. As a simplified example, consider the following sequence from item A to item D (see Figure 1.2). $\vec{V_0}$ was derived from the first hop, $\vec{ab}$. $\vec{V_1}$ is half the sum of $\vec{V_0}$ and $\vec{bc}$, which was the second hop. $\vec{V_2}$ is half the sum of $\vec{V_1}$ and $\vec{cd}$, and the process goes on.

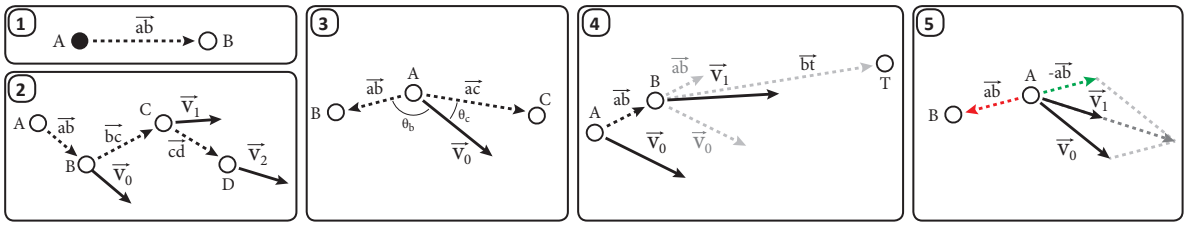

Figure 1. Direction-based navigation (Vector algorithm)

In each step, the system calculates the probabilities of suggesting each neighbor by comparing the current direction vector $\vec{V}$ to each of the vectors towards the $|K|$ considered candidates (i.e., to vectors from the current to neighbor nodes). For example, consider the decision to move from item A with current *direction vector* $\vec{V_0}$ to two neighbors, B and C (see Figure 1.3).

The more aligned the direction and the candidate vectors are, the smaller the angle θ will be, and the higher the probability of suggesting that neighbor. In Figure 1.3, since $\theta_c < \theta_b$, node C will have a higher probability to be the next item.

It is also possible to set a target destination T, which is an arbitrary point on the map, where the navigation should try to go to. This adds a third element to the direction vector update procedure, by creating vector $\vec{bt}$ in each update and also adding it to $\vec{V}$. Let's consider the situation in which a hop was accepted from item A to item B, and the target destination T was configured (see Figure 1.4). Note that $\vec{V_1}$ is made by adding vectors $\vec{V_0}$, $\vec{ab}$ and $\vec{bt}$ and dividing the module of the resulting vector by three.

Last but not least, feedback is incorporated by considering that, when a user skips a suggestion, it would be interesting to increase the probability of suggesting something different from the skipped item. So, when an item is skipped, the system does not change the current node, and the opposite vector is added to $\vec{V}$. Consider the example in Figure 1.5, where item B was skipped, and so $\vec{V_1}$ was calculated by adding $\vec{V_0}$ to $-\vec{ab}$ to reflect the user's preference.

Defining the method formally: Consider a set K of closest neighbors of *current node* A and the current *direction vector* $\vec{V}$ with the respective angles θ_i between $\vec{V}$ and each hop vector $a\vec{k_i}, k_i \in K$. Also, consider the optional parameter with the location of a target destination T. We define the direction-based navigation function using the following weight variables:

$$w_i = 1 + cos(\theta_i) = 1 + \frac{a\vec{k_i} \bullet \vec{V}}{|a\vec{k_i}||\vec{V}|}.$$

Note that the weight is proportional to the cosine of the angle between the current direction vector $\vec{V}$ and the direction of each neighbor k_i relative to the current node A. We add 1 to avoid negative values. Finally, we define the probability of node $k_i \in K$ to be *next* as:

$$P_i^{nextVec} = \frac{w_i}{\sum_{j=1}^{|K|} w_j}.$$

Note that we have a proper probability distribution, since the sum over probabilities $P_i^{nextVec}, i \in K$ is 1.

After the next item has been returned, say k_i, and the user has provided feedback by accepting or skipping it, we update the direction vector $\vec{V}$ of node A as follows:

$$\vec{V} = \begin{cases} (a\vec{k_i} + \vec{V})/2, & \text{if } k_i \text{ was accepted} \\ (-a\vec{k_i} + \vec{V})/2, & \text{if } k_i \text{ was skipped.} \end{cases}$$

If target destination T has been defined, then the calculation also includes the new target vector $\vec{at}$ between the chosen item and the target destination:

$$\vec{V} = \begin{cases} (a\vec{k_i} + \vec{V} + \vec{at})/3, & \text{if } k_i \text{ was accepted} \\ (-a\vec{k_i} + \vec{V} + \vec{at})/3, & \text{if } k_i \text{ was skipped.} \end{cases}$$

If the item was accepted, node k_i becomes the next current node. Otherwise, the current node does not change, and only the direction vector is updated. Note that this approach is domain-independent and uses nothing but the coordinates of the embedding itself. It also carries an explicit dependency on user feedback, since $\vec{V}$ is determined by the user's skipping behavior.

4. MUSIC DOMAIN

The navigation framework described in Section 3 can be applied to different media domains. In this work, we focus on the domain of music.

4.1 Last.fm Dataset

In order to define music similarity, we assume that the more frequently two songs co-occur in a user's listening history, the more similar they are. We collected co-occurrence data from `Last.fm`, a social music site that tracks user musical tastes, from November, 2014 to March, 2015. More specifically, we collected the top-25 most listened songs of each user, reaching a total of 372,899 users, 2,060,173 tracks, and 374,402 artists. Moreover, we also collected a total of 1,006,236 user-generated tags, associated with songs. In particular, 75% of songs have had at least one associated tag in our dataset. We considered a subset of 983,010 tracks in our dataset with a known MBID [2], from which we selected another subset of 83,180 tracks that co-occurred 5 or more times, forming a connected component of 62,352 songs. A detailed characterization of the dataset can be found in [19].

[2] MusicBrainz Identifier (MBID) is a reliable and unambiguous form of music identification (`musicbrainz.org`).

The connected graph with 62,352 vertices enabled us to run IsoMap [21] and Multidimensional Scaling (MDS) [6] without any approximations. By parallelizing parts of the algorithm, we computed the all-pairs shortest path matrix of size $62,352 \times 62,352$, in 7 minutes on a server with 50 GB of RAM and 16 CPU cores, and computed the embedding into 100 dimensions in approximately 2 hours on the same server. Note that a larger collection could have been embedded using a less computationally intensive approximate algorithm, such as LMDS or LINE [7, 20]. An evaluation of the embedding process can be found in [12].

4.2 Mixtape

We wanted to collect real user feedback in order to evaluate the navigation framework in the music domain. For this purpose, we developed Mixtape, a web-based application with a simple user interface (see Figure 2). On the server side, a k-d tree was loaded with a 100-dimensional space of $62,352$ tracks. On the client side, the design goal was to provide a minimalist user interface that fully explored the navigation functions defined in Section 3.2. Each user would choose the starting song and then be presented with one suggestion at a time, with explicit feedback-generating actions. The user interface is comprised of a playlist, on the left, which shows the songs the user has accepted or skipped, and a Youtube video window, which finds and plays the current suggested item. Users can then decide whether they like the song or not, using the *like* and *dislike* buttons, one of which the user must press in order to receive the next suggestion. In case the user does not press anything and listens to the entire song, we assume they liked it, and consider the song as accepted. There is also a settings button, which allows the user to switch between different navigation functions.

5. EXPERIMENTS

The evaluation of the proposed navigation framework in the domain of music is twofold: Firstly, we propose an automatic evaluation framework and perform an extensive analysis based on simulated user profiles. Furthermore, since real users might behave differently, and the perception of a song is subjective, we observed how real users interacted with our Mixtape application. As a result, we were able to evaluate not only how effective and engaging the proposed navigation system is, but also how well the simulated user profiles approximated real user behavior.

5.1 Simulated user profiles

To test the navigation framework, we simulated synthetic user profiles, in which hypothetical users intend to listen to 20 songs (about one hour of music), and count the number of skips (songs that are skipped by the simulated user profile following the algorithm described below) until 20 songs are accepted. A similar evaluation approach was used in [18]. We simulated two types of users:

Tag-based user profile: This user profile is based on tag information and the notion of transition between two regions on the map. Recall from Section 4.1 that we collected over 1,006,236 user-generated tags, associated with songs. We assume this user wishes to listen to a sequence of songs that transitions from initial tag T_i to final tag T_f.

To do that, the simulated user accepts all songs associated with tag T_i in the first $1/3$ of the navigation path (skips otherwise), accepts songs with tags T_i or T_f in the second $1/3$, and accepts only songs with tag T_f in the last $1/3$ of the path, comprised of a total of 20 items. Note that real users do not necessarily know what tags are associated to particular tracks. Since these users are hypothetical, we can use the collected tag information for simulation purposes.

We manually selected tag transitions among the top 200 most popular tags in our dataset. We noticed these tags could be divided in three categories: Mood tags, such as *Chill, Upbeat, Relaxing*, Genre tags, such as such as *Rock, Hip Hop, Folk*, and Age tags, such as *60's, 90's, 2000's*. We then paired them up manually, selecting 14 transitions to experiment with. For each tag transition ($T_i \Rightarrow T_f$), we considered a navigation path starting at the most popular song associated to tag T_i, and applying the skipping rule until a path of 20 accepted songs was achieved.

Artist-based user profile: This user profile is based on artist information and the notion that certain users wish to listen to songs by artists they already know. Since this user wishes to listen to preferred artists, whenever the suggested song is by an artist contained in the user's history, it is accepted. Otherwise, it is skipped. We collected the complete listening histories of 20 users to simulate this user profile, and started the playlist at a random song within each user's profile. Moreover, for this experiment only users whose profiles were *not* used to construct the embedding were simulated.

5.2 Baselines

As baselines, we tested the following approaches:

LME: Logistic Markov Embedding [4, 16], a probabilistic approach that models sequences in a Euclidean space using radio streams as a training set. We used the implementation available at the authors' homepage, with all parameters set to default values, except for $\alpha = 5$ (this value resulted in superior overall performance), as our dataset did not have music sequences, only music occurrences in a user profile, we used the "yes-complete" dataset (also made available by the authors) in a combination with our dataset, since LME needs a sequence of items as input, which resulted in an intersection set of 31,544 items with our dataset. We made one modification to the LME algorithm by incorporating user feedback when computing the next item. More specifically, whenever an item n_j has been skipped after a previously accepted item n_i, we recompute the probabilities at n_i setting $Pr(n_j|n_i) = 0$, and maintaining n_i as the current item.

Random: A random song is returned, considering all songs in the dataset.

Random Tag: A random song with tag T is returned. This baseline was used for the tag-based navigation evaluation.

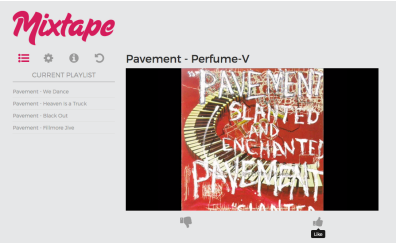

Figure 2. Mixtape screenshot

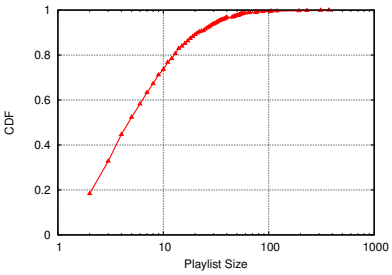

Figure 3. Mixtape user study: playlist length distribution

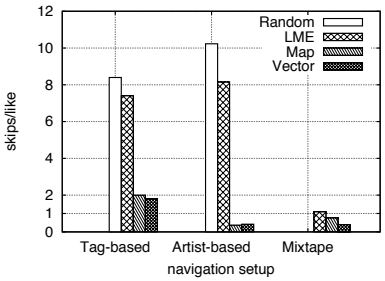

Figure 4. Total number of skips per like ratio: simulated versus real-user profiles

5.3 Mixtape user study setup

We collected all user actions on Mixtape over a course of 2 weeks, resulting in the participation of over 800 users, generating a total of over 2000 navigation sessions. In order to compare the performance of different navigation algorithms, each navigation session was randomly assigned either Map, Vector or LME algorithms (with undefined tag parameters), but the user could explicitly change the algorithm in the settings menu as well. Users could also choose the Random approach, however, since the engagement in this setting was very low, we did not include it in the plots.

5.4 Results

In our experiments, we measure the effectiveness of the two navigation approaches proposed in this work (Map and Vector) and the two baselines (Random and LME) in three navigation setups: simulated tag-based user profiles, simulated artist-based user profiles, and real users on Mixtape. We use two main metrics: skipping behavior and playlist smoothness, defined below. Each scenario was executed 20 times, and all figures show the 95% confidence interval.

Skipping behavior: Figure 3 shows the CDF of playlist length generated on Mixtape. It can be seen that almost 30% of the playlists [3] contain 10 tracks or more, and almost 15% have size 20 or longer, which shows that many people really engaged with the application.

In Figure 4 we compare the ratio of the total number of skips (dislikes) and the total number of accepted songs (likes) in playlists generated by all navigation algorithms for simulated and real user profiles. Analyzing the simulated user profiles, we can see that the baseline algorithms present several times more skips per like (LME: $skips/like > 7.5$, Random: $skips/like > 8$) than Map and Vector ($skips/like < 2$). Map and Vector have similar results and perform especially well in the artist-based simulated setup ($skips/like < 0.5$), which shows they are more effective not only in directing the user between different regions in the space, but also in presenting the user with music by preferred artists.

Looking at Mixtape results on Figure 4, we can see that all three approaches perform well on average ($skips/like < 2$). LME has still more skips than likes ($skips/like > 1$), whereas Map and Vector have significantly more likes than skips (Map: $skips/like < 0.8$, Vector: $skips/like < 0.4$), indicating that users enjoyed the vast majority of the suggested songs, especially by the Vector algorithm. Note that Vector outperforms Map for real users, indicating that the direction in the map, provided by the real-time feedback, does matter for real users.

Comparing real and synthetic user profiles, we note that LME performed much better with real rather than simulated users. That might be because real users are more open-minded and accept more diversity in their playlists. Nevertheless, the number of skips per like for Vector and Map on Mixtape was similar to the simulated artist-based user profiles, indicating that in some aspects the simulation was accurate in portraying a real user.

In Figures 5 through 7 we analyze the number of skips along each step of the navigation process. In Figure 5 the number of skips per step decreases in the second third and then reaches a maximum in the beginning of the third part of the playlist for all algorithms. This illustrates how the algorithms react to the simulated tag-based navigation setup. Afterwards, however, we can see that Map and Vector quickly decrease the number of skips, as opposed to LME and Random, showing that the former algorithms succeed in adjusting the direction of the navigation towards the destination tag.

Playlist smoothness: In Figures 8 through 10 we analyze how similar consecutive songs are on a navigation path, by measuring the cosine similarity of the artists of consecutive (accepted) songs. [4] Note that in Figure 8 we plot the RandomTag baseline instead of Random, to shed light on the following question: if the objective of tag-based simulations is to recommend songs with a given tag, why not simply choose songs from the database that have that tag? That method might work when we ignore the relationship between songs in a playlist. However, we argue that a playlist should be more than a group of songs with a given tag–it should present a relationship between the songs. We can see that RandomTag and LME baselines provide almost zero similarity along the navigation path, even though RandomTag only returns songs with accepted tags, i.e.,

[3] We refer to the sequence of tracks accepted, or liked, by a user in one navigation session as a *playlist*.

[4] We define artist similarity as the cosine similarity computed from artist co-occurrence in our dataset.

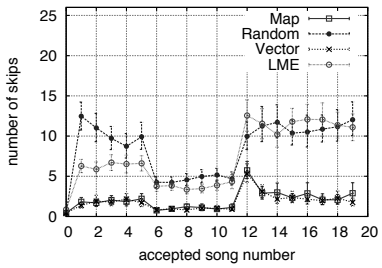

Figure 5. Skips along playlists: simulated tag-based navigation

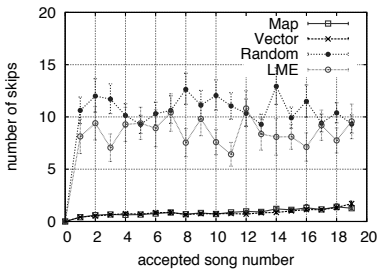

Figure 6. Skips along playlists: simulated artist-based navigation

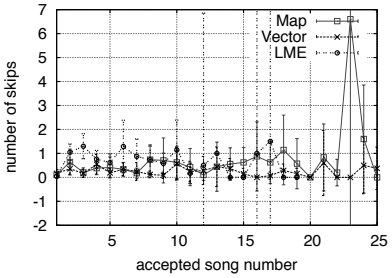

Figure 7. Skips along playlists: real-user navigation (Mixtape)

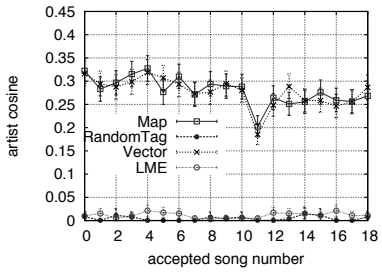

Figure 8. Playlist smoothness: simulated tag-based navigation

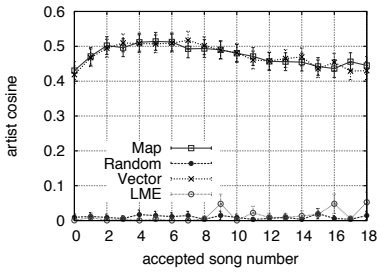

Figure 9. Playlist smoothness: simulated artist-based navigation

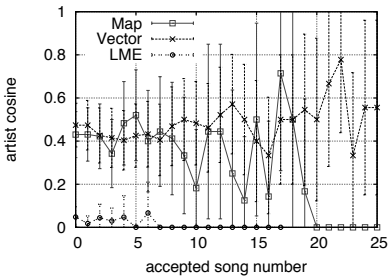

Figure 10. Playlist smoothness: real-user navigation (Mixtape)

makes zero skips in the tag-based simulation setup. Map and Vector, on the other hand, trace highly smooth navigation paths, offering the user songs with high similarity to the previously chosen songs, especially in the artist-based simulation setup (Figure 9, artist cosine > 0.4). Figure 10 [5] shows that Map and, especially, Vector playlists on Mixtape also present high similarity between consecutive items, indicating that people prefer smooth, rather than abrupt, transitions in their navigation paths.

User feedback: To enhance our perception about how users perceive our Mixtape application, we created a short online survey, linked from the Mixtape application, which was answered by 44 unidentified subjects. The users were not provided with any information about the navigation algorithms. They were asked to provide feedback on the experience of using Mixtape, leading to the following numbers: 95% enjoyed the songs suggested by Mixtape, and only 11% of the users were not able to find most of the songs they were searching for (recall from Section 4.1 that we used a reduced sample of the map in our experiments: 62,352 songs with co-occurrence at least 5.).

Interestingly, 70% of the participants said they discovered new artists or songs. Most people said they didn't change the navigation policy and they didn't know which policy they used during their navigation. From those who did experiment with different policies, they equally enjoyed Map and Vector approaches (even though, on average, users skipped less songs when using direction-based navigation).

[5] The CIs in Figures 7 and 10 are high, due to insufficient data for certain song numbers.

To sum up this first user study, we can conclude that users enjoyed navigating music collections by giving their real-time feedback and that the navigation allowed them to discover previously unknown songs they enjoyed.

6. CONCLUSION

In this work we proposed a navigation framework for large media collections and evaluated an implementation of the framework in the domain of music. Potentially, the same ideas could be applied to other kinds of media, e.g. movies or TV shows [10, 11]. Rather than creating fixed playlists, our approach allows users to provide feedback through skipping behavior and direct the navigation process in real-time. We evaluated the framework through simulation of more that 2,000 synthetic navigation paths and performed a real user study by launching Mixtape, a web application with a minimalist user interface that allows people to navigate a collection of over 60,000 music tracks. We analyzed over 2,000 playlists generated by over 800 real users and received positive feedback about the application. When comparing playlists generated by Mixtape and simulated hypothetical users, we could observe several similarities, indicating that in some aspects the simulation was accurate in portraying a real user. Moreover, not only did this user study serve as validation of the proposed framework, but it also provided insights into what users look for and appreciate in a media navigation system. [6]

[6] Acknowledgments: This work is supported in part by CNPq, FAPEMIG and LG Electronics in cooperation with Brazilian Federal Government through Brazilian Informatics Law (n. 8.2.48/1991).

7. REFERENCES

[1] E. Bernhardsson. Systems and methods of selecting content items using latent vectors, August 18 2015. US Patent 9,110,955.

[2] Lukas Bossard, Michael Kuhn, and Roger Wattenhofer. Visually and Acoustically Exploring the High-Dimensional Space of Music. In *IEEE International Conference on Social Computing (SocialCom), Vancouver, Canada*, August 2009.

[3] Pedro Cano, Markus Koppenberger, and Nicolas Wack. Content-based music audio recommendation. In *Proceedings of the 13th annual ACM international conference on Multimedia*, pages 211–212. ACM, 2005.

[4] Shuo Chen, Josh L Moore, Douglas Turnbull, and Thorsten Joachims. Playlist prediction via metric embedding. In *18th ACM SIGKDD*, 2012.

[5] Shuo Chen, Jiexun Xu, and Thorsten Joachims. Multi-space probabilistic sequence modeling. In *19th ACM SIGKDD*, 2013.

[6] Trevor F. Cox and M.A.A. Cox. *Multidimensional Scaling*. Chapman and Hall/CRC, 2000.

[7] Vin De Silva and Joshua B Tenenbaum. Sparse multidimensional scaling using landmark points. Technical report, Technical report, Stanford University, 2004.

[8] Arthur Flexer, Dominik Schnitzer, Martin Gasser, and Gerhard Widmer. Playlist generation using start and end songs. In Juan Pablo Bello, Elaine Chew, and Douglas Turnbull, editors, *ISMIR*, pages 173–178, 2008.

[9] Olga Goussevskaia, Michael Kuhn, Michael Lorenzi, and Roger Wattenhofer. From web to map: Exploring the world of music. In *Web Intelligence and Intelligent Agent Technology, 2008. WI-IAT'08*, volume 1, 2008.

[10] Pedro Holanda, Bruno Guilherme, Joao Paulo V. Cardoso, Ana Paula Couto da Silva, and Olga Goussevskaia. Mapeando o universo da midia usando dados gerados por usuarios em redes sociais online. In *The 33rd Brazilian Symposium on Computer Networks and Distributed Systems (SBRC)*, 2015.

[11] Pedro Holanda, Bruno Guilherme, Ana Paula Couto da Silva, and Olga Goussevskaia. TV goes social: Characterizing user interaction in an online social network for TV fans. In *Engineering the Web in the Big Data Era - 15th International Conference, ICWE 2015, Rotterdam, The Netherlands, June 23-26, 2015, Proceedings*, pages 182–199, 2015.

[12] Pedro Holanda, Bruno Guilherme, Luciana Fujii Pontello, Ana Paula Couto da Silva, and Olga Goussevskaia. Mixtape application: Music map methodology and evaluation. Technical report, Department of Computer Science, Universidade Federal de Minas Gerais, Belo Horizonte, MG, Brazil, May 2016.

[13] Michael Kuhn, Roger Wattenhofer, and Samuel Welten. Social audio features for advanced music retrieval interfaces. In *Proceedings of the international conference on Multimedia*, pages 411–420. ACM, 2010.

[14] Beth Logan. Content-based playlist generation: Exploratory experiments. In *ISMIR*, 2002.

[15] François Maillet, Douglas Eck, Guillaume Desjardins, and Paul Lamere. Steerable playlist generation by learning song similarity from radio station playlists. In *ISMIR*, 2009.

[16] Joshua L Moore, Shuo Chen, Thorsten Joachims, and Douglas Turnbull. Learning to embed songs and tags for playlist prediction. In *ISMIR*, pages 349–354, 2012.

[17] Joshua L Moore, Shuo Chen, Douglas Turnbull, and Thorsten Joachims. Taste over time: The temporal dynamics of user preferences. In *ISMIR*, 2013.

[18] Elias Pampalk, Tim Pohle, and Gerhard Widmer. Dynamic playlist generation based on skipping behavior. In *ISMIR*, 2005.

[19] Luciana Fujii Pontello, Pedro Holanda, Bruno Guilherme, Joao Paulo V. Cardoso, Olga Goussevskaia, and Ana Paula Couto da Silva. Mixtape application: Last.fm data characterization. Technical report, Department of Computer Science, Universidade Federal de Minas Gerais, Belo Horizonte, MG, Brazil, May 2016.

[20] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. Line: Large-scale information network embedding. In *24th International Conference on World Wide Web*, pages 1067–1077, 2015.

[21] J. B. Tenenbaum, V. Silva, and J. C. Langford. A Global Geometric Framework for Nonlinear Dimensionality Reduction. *Science*, 290(5500):2319–2323, 2000.

[22] Douglas R Turnbull, Justin A Zupnick, Kristofer B Stensland, Andrew R Horwitz, Alexander J Wolf, Alexander E Spirgel, Stephen P Meyerhofer, and Thorsten Joachims. Using personalized radio to enhance local music discovery. In *CHI'14*, 2014.

[23] Aaron Van den Oord, Sander Dieleman, and Benjamin Schrauwen. Deep content-based music recommendation. In *Advances in Neural Information Processing Systems*, 2013.