UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE ENGENHARIA DE COMPUTAÇÃO

ALEXANDRE TADEU SALLE

Recommendation-assisted Playlist Creation

Monografia apresentada como requisito parcial para a obtenção do grau de Bacharel em Engenharia de Computação.

Orientador: Prof. Dr. Érika Cota

Porto Alegre
2014

ABSTRACT

This paper proposes a new mechanism for the creation of music playlists called recommendation-assisted playlist creation (RAPC). It uses a recommender system to assist the user in the manual creation of a playlist. Based on a seed, tracks are proposed to the user. The user can select one or more of these tracks or request a new set of recommendations. The recommender system uses the user's track selections to improve each subsequent round of recommendations. This novel mechanism is implemented using The Echo Nest as its recommender system. This implementation of RAPC is evaluated in a user study which compares it to two other mechanisms: (1) manual creation, where users can search and browse a catalog and (2) automated creation using The Echo Nest's artist radio. Participants were asked to create 10 track playlists using all three mechanisms. The results show that playlists created using RAPC: (1) are better than the playlists created using the other mechanisms (p=0.03271) (2) contain a greater number of artists than those created using the manual mechanism (p=0.001953) (3) contain more tracks that users have not listened to recently compared to playlists created using the manual mechanism (p=0.003906) (4) contain less dissatisfactory tracks compared to playlists created using the automated mechanism (p=0.0009766).

Keywords: music playlists, playlist creation, recommendations, user study

FIGURES

TABLES

ABBREVIATIONS

RAPC	Recommendation-assisted playlist creation
IQR	Inter-quartile range

TABLE OF CONTENTS

1 INTRODUCTION

Creating playlists from scratch is a tiresome endeavor. The repetitive cycle is: (1) Think of a track. (2) Find the track. (3) Add it to the playlist. Steps (2) and (3) are tiring because they are mechanical in nature, especially on small form-factor devices such as mobile phones, where typing artist names and track titles is inherently more difficult than on a larger device. Step (1), however, is the one that causes the most frustration. It forces the user to answer the open question: what do I want to listen to next? Open questions take longer to answer than closed questions. Consider the following open question: where would you like to go on your next vacation? Now consider its closed form: on your next vacation, would you rather go to (a) Paris (b) Moscow (c) Rio de Janeiro? Granted, the closed question might not lead to an optimal answer, but it yields a much faster response.

There are of course ways to build playlists with little or no effort using automated systems. Assume the system has access to a music library with thousands of songs from different artists and genres. The user has listened at least one time to a small fraction of the library. The simplest playlist generation system is the random shuffle of the entire library. This system generates poor playlists, as it does not take into account context, which consists of the user's past music preferences and his immediate preference. To remedy this problem, intelligent recommender systems were developed.

The downside of these systems is that they invariably make recommendations that displease the user, forcing him to skip certain tracks. In scenarios where the user wants to avoid interacting with the music player, reducing or completely eliminating skipped tracks (skip-free) is desirable. This is only possible if: (a) The recommendation system consistently makes perfect predictions or (b) The user selects each of the tracks he will listen to immediately before the listening session. As will be shown, we are still a ways from reaching (a). Alternative (b) imposes the aforementioned burden of manual playlist creation upon the user.

This paper will proceed as follows: (1) Manual and automated playlist creation mechanisms will be characterized. (2) A novel mechanism called recommendation-assisted playlist creation (RAPC) will be presented. (3) RAPC, manual, and automated mechanisms will be evaluated in a user study. (4) Results will be presented and discussed, followed by closing remarks.

2 MANUAL PLAYLIST CREATION

This mechanism is pretty simple: users search/browse a music catalog and manually select tracks that they would like added to their playlist. For the purposes of this study, to search and browse the catalog will be limited to: (1) Searching for an artist or track. (2) Browsing an artist's track collection. This basic functionality is available in all online music services, and so serves as a baseline for generalizing the results of this study.

3 AUTOMATED PLAYLIST CREATION

Rather than having the user create playlists manually, automated mechanisms allow the user to supply context, then generate playlists using heuristics. The heuristic can be as simple as a random shuffle of the catalog's tracks, or as complex as predicting tracks that are likely to please the user. This latter approach of prediction is now ubiquitous in online music services. Predictions are made by tapping recommender systems: systems that provide suggestions for items (in this case, tracks) to be of use to the user (RICCI, ROKACH and SHAPIRA, 2011). These systems can follow two approaches: content-based filtering or collaborative filtering (or a mix of both).

3.1 Content-based Filtering

Content-based filtering works by finding items whose description is similar to items a user likes. In the context of music, an item is a track, and the track's description includes its artist, genre, tempo, etc. A content-based filtering approach could identify that most of the tracks a user likes are from the Jazz genre, and accordingly recommend Jazz tracks. It is even possible to analyze actual track content and perform automated spectral analysis to identify similar tracks (LOGAN and SALOMON, 2001). Others use a combination of laborious human analysis along with automated methods to build track descriptions for use in content-based filtering (WESTERGREN, 2007).

3.2 Collaborative Filtering

This approach does not use item descriptions, instead relying on users' behavior towards items. The key idea is that users with similar tastes will like the same items. Suppose a user A likes 1000 items. User B likes 999 of these items and has not yet rated the 1000th. Collaborative filtering relies on the high probability that user B will like this 1000th item as well. Using this principle, it's possible to calculate item-to-item similarity (SARWAR, KARYPIS, *et al.*, 2001).

Applying this latter approach to music, it is possible to treat a set of playlists as users and the tracks within them as items to construct a track-to-track similarity matrix (AIZENBERG, KOREN and SOMEKH, 2012).

3.3 Imperfect Playlist Generation

A human evaluation of playlist generation by Apple's state-of-the-art collaborative filtering recommender system, Genius, shows that in generating a 5 track playlist given a track as a seed, on average 1.3 songs did not fit well with the seed track (BARRINGTON, ODA and LANCKRIET, 2009). By way of comparison, in the same study, a random algorithm selected on average 2.56 bad tracks per 5 tracks playlist.

3.4 Skips

No matter how good the recommender system, it will make recommendations that displease the user. A bad recommendation will result in a skipped track, from hereon referred to as a skip. Table 3.1 makes an intuitive comparison of the number of skips resultant from different playlist generation mechanisms.

Table 3.1: The frequency with which users will skip tracks in playlists generated via different mechanisms.

Mechanism	*Skip Frequency*	*Intuition*
Random Shuffle	High	Pays no attention to user preference
Recommender System	Low	Considers user preference
Manually Created, Stale	Medium	User has probably listened to it many times and has grown tired of some tracks
Manually Created, Fresh	Very Low	Given that the user created the playlist and is listening to it moments later, he is unlikely to skip any tracks

Fonte: Salle (2014)

Granted, the chances of a user discovering new tracks in a manually created playlist are very low. Recommender systems are great for discovery because they suggest tracks that the user will probably enjoy, even if he has never heard them before. Discovery is one of the factors that lead to skipping: either the user is not interested in discovering new music or he is unhappy with the song he's been suggested. In either case, he is dissatisfied and will probably skip.

So even though recommender systems allow for nearly effortless playlist generation, they does not offer the main benefit of manual creation: selection, and by direct result, low skip frequency.

This intuitive analysis that automated systems lead to more skips than a manual approach will be validated in Section 5.

4 RECOMMENDATION-ASSISTED PLAYLIST CREATION

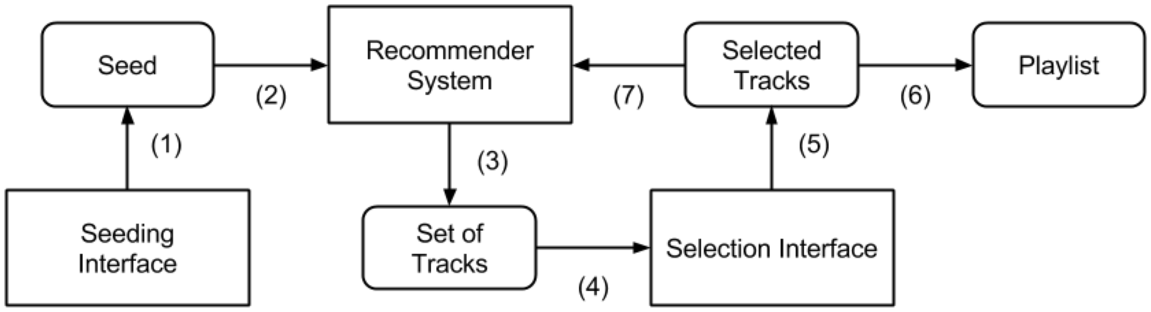
Sections 3.4 hints very strongly at a way to lower the effort required for manual playlist creation. As Section 3.3 indicates, recommender systems do a pretty good job at predicting what tracks a user will like. What if rather than simply playing the recommended tracks, the system presented the user with a list of recommendations, and let him choose the ones he wants to listen to?

As described in Section 1, the frustration in manual playlist creation comes from answering open questions repeatedly. Using a recommender system to guess tracks that a user is likely to enjoy and asking him to select one, the open question of what track to listen to next is transformed into a closed question. Additionally, the user's choice serves as relevance feedback to the recommender system, allowing it to make better suggestions in the next round of track selection. The author calls this mechanisms recommendation-assisted playlist creation (RAPC).

4.1 Specification

Figure 4.1 illustrates the entire flow of RAPC. The steps are as follows: (1) Through a seeding interface, the user selects a seed for the recommender system. The choice of seed is left to the implementation. (2) The seed is fed into the recommender system. (3) The recommender system predicts a set of tracks that are likely to please the user. The choice of recommender system is left to the implementation. (4) The set of tracks is displayed in a selection interface. The interface is implementation specific. (5) The user selects a subset of the set of tracks. The number of tracks and the selection mechanism is interface specific. The set can be empty, meaning none of the selected tracks were to the user's liking. (6) The set of selected tracks is added to the playlist. (7) The selected tracks are fed into the recommender system as positive implicit feedback. If the set of selected tracks is empty, the system should use the previous seed to suggest a new set of tracks that does not overlap with those previously suggested. Step (3) is then repeated until the user is satisfied with the size of the playlist.

Figure 4.1: Flow of recommendation-assisted playlist creation (RAPC).



Fonte: Salle (2014)

4.2 Ideal Recommender System

In implementing RAPC, 4 questions must be answered regarding the recommender system: (1) What should be the size of the set of recommended tracks? (2) How should these tracks be selected? (3) What's a good seed? (4) How to handle relevance feedback?

Questions (1) and (2) are not specific to recommender systems used in RAPC; they are common to all recommender systems. Fortunately, there is a rich literature on the subject to help answer them. In regards to (1): In the literature, this recommendation set is known as a recommendation list. Although the size of a recommendation list matters, it is more important that users be easily able to generate new lists (SWEARINGEN and SINHA, 2002). Naturally, the list should be large enough to minimize the number of refreshes required for the user to get a satisfactory recommendation, but not so big that it overwhelms the user with information.

Question (2) asks how the n tracks that make up this list should be selected. Suppose the system uses tracks as seeds. In an item-to-item collaborative filtering recommender system, a natural approach would be to select the n tracks most similar to the seed track. A study shows that even though this yields a recommendation list with the highest average accuracy (where accuracy is measured as the similarity between each track and the seed track), a selection process that aims for higher diversity leads to greater user satisfaction (ZIEGLER, MCNEE, *et al.*, 2005). To illustrate, suppose a user is looking for recommendations for tracks similar to "Smells Like Teen Spirit" by Nirvana. Assuming the user is also interested in other artists (not just Nirvana), suggesting only tracks by Nirvana would lead to a poor experience, even though these tracks would be highly accurate recommendations. In summary, the makeup of the recommendation list is more important than the accuracy of its individual items.

Another study shows that users like systems that make familiar suggestions, but that too much familiarity can breed contempt (SWEARINGEN and SINHA, 2002). In the case of RAPC, recommender systems need to aim for familiarity, for users are highly unlikely to add tracks they do not recognize to their playlists. Unlike a radio experience where discovery is desired, the argument of contempt if recommendations are too familiar does not apply. RAPC recommender systems should aim for diverse yet familiar recommendations.

In regards to question (3) (seeding): the goal of seeding is describing the user's current preference as accurately as possible. Prompting him for an artist name or track title yields data that is more current than, for example, seeding based on his entire music collection. The seed should be fresh. Ideally, the system should support artist, track, genre and mood as seeds, as well as combinations of these (e.g. multiple artists).

Question (4) asks how the recommender system should respond to relevance feedback. It should consider the set of selected tracks implicit positive feedback. A trivial way to do this is to treat these tracks as additional seeds, either exclusively, or in combination with previous seeds. This feedback can also be used to gage the level of diversity the user wants in the recommendation lists.

4.3 Autopilot

One possible extension of RAPC is autopilot mode, where the system automatically appends a track to the currently playing playlist when the last track finishes playing. This brings the benefit of endless playback offered by automatic playlist generation (radio) methods, with two significant advantages: (1) The user has until the last second to choose a different track. (2) The user can extend the playlist at any time by using RAPC, and autopilot will only kick in once playback of the playlist is complete. This extension will not be evaluated in this study.

5 EVALUATION

This study aims to compare two existing playlist creation mechanisms, manual and automated, to the newly proposed RAPC mechanism. A system that simulates an online music service possessing all three mechanisms is constructed and used in a user study. The idea is that these results can be replicated if RAPC is implemented in an online music service such as Spotify which already possesses manual and automated playlist creation mechanisms (SPOTIFY).The Echo Nest, the recommender system used as the basis of both automated and RAPC mechanisms, is in fact the same one used by two popular online music services: Spotify and Rdio (THE ECHO NEST).

5.1 Experiment

A user is asked to think of a context for which he might create a playlist. It can be a party, a run, a study session, anything he wishes. He is asked to think of a single track that will inspire a playlist. Using this track as inspiration, he proceeds to create 10 track playlists (excluding the inspiration track) using three different mechanisms: RAPC, manual, automated. The evaluation's supervisor demonstrates how each mechanism works with a quick run-through of the evaluation system. Control is then turned over to the user. Using each mechanism, he creates 10 track playlists. The order of mechanisms is randomized per user to eliminate any bias that might stem from proceeding in a specific order. Once the playlist is complete, the user listens to each track long enough to decide if he is pleased or displeased with the track. The track should fit in with the context for which the playlist is being created. If he is pleased, he "thumbs up" the track; otherwise he gives it a "thumbs down". Once all tracks in the playlist have been evaluated, the user reports how many of these tracks he has not listened to in the 2 months. After creating all three playlists, he decides which one he likes best. He also indicates whether he found the manual or RAPC mechanism more tiring. The time taken to create each playlist is tracked. The number of searches conducted by the user is also tracked.

5.2 Music Catalog

To simulate an experience similar to that of online music services such as Spotify, a music catalog that is as expansive is desirable. The Echo Nest's music metadata database provided a catalog so sufficiently large that, during this experiment, not a single user search failed to turn the desired result. The Echo Nest only provides metadata, and part of the experiment requires actual audio content. To that end, YouTube's Data API was used to search for videos, which were then played hidden using YouTube's IFRAME API to simulate an experience similar to that of online music services (GOOGLE).

5.3 Mechanism Implementations

5.3.1 Manual

The implementation of this mechanism follows the description given in Section 2: users can search for artists and tracks and browse artists' track lists. As is common in online music services, an artist's most popular tracks precede the listing of all his tracks, as the user is more likely to want one of these tracks in his playlist. Both search and browse operations use The Echo Nest's Web API.

5.3.2 Automated

Given the user's seed track, the track's artist is used to seed The Echo Nest's artist radio through its Web API. The API returns a playlist that is used as-is.

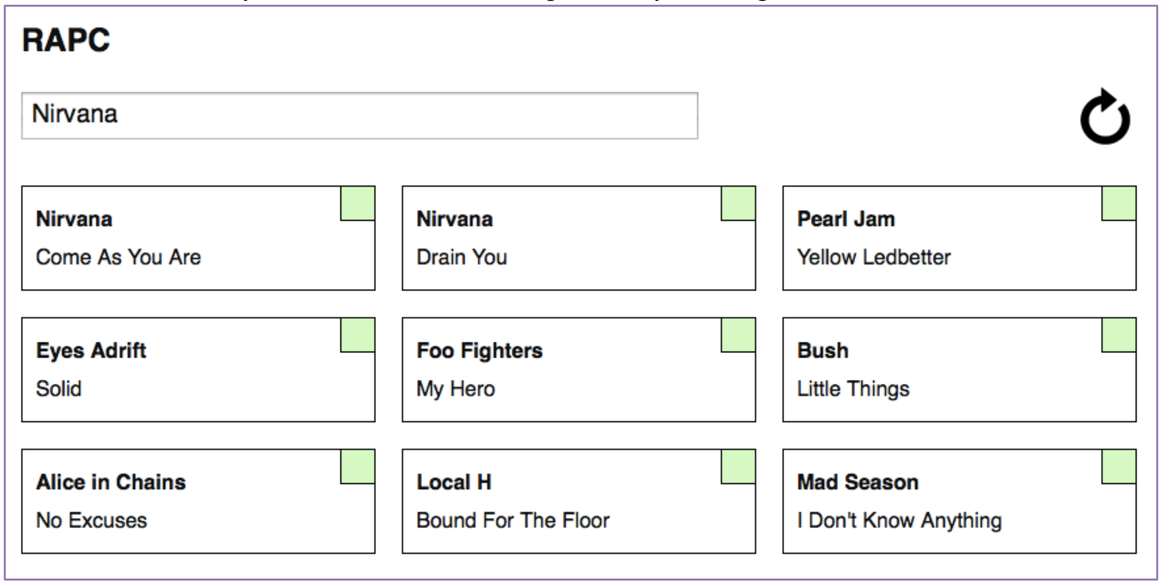
5.3.3 RAPC

An RAPC implementation needs a seeding interface, a selection interface, and a recommender system. The interface shown in Figure 5.1 serves as both the seeding and selection interface. Users seed the system through the input box. The circular arrow button lets users request a new set of recommended tracks. The small squares in the corners of each track box enable users to reseed the system with the artist of that box's track and receive track recommendations exclusively from that artist. Clicking anywhere else in the box adds the track to the playlist and sends feedback to the recommender system. The recommender system used, The Echo Nest's artist radio, is the same one used in the automated mechanism implementation described in Section 5.3.2. Hence the recommender system itself is a constant when comparing RAPC and automated, eliminating the chance that one mechanism is superior to the other because of the superiority of its recommender system.

When the user starts the experiment, the recommender system is seeded with the artist of the playlist's inspiration track. As tracks are selected, the artists of these tracks are also used as seeds. A sliding window of 4 artists is used to seed the system. Suppose the inspiration track for the playlist is T1 by artist A1. The window is [A1]. Now suppose the user selects tracks from artists A2, A3, and A4. The window is now [A1, A2, A3, A4]. If he selects another track from artist A5, A1 is removed from the window, yielding [A2, A3, A4, A5].

It should be noted that the interface and recommender system decisions above were made to adhere to the RAPC specification. The author does not, however, claim that they are optimal. They simply serve to enable the evaluation of RAPC. For a commercial implementation, following human-computer interaction principles would yield a better implementation of RAPC.

Figure 5.1: Screenshot of the seeding and selection interface. The input box and small corner squares seed the system. Selection is accomplished by clicking on the track box.



Fonte: Salle (2014)

5.4 Evaluation System

All three mechanisms were deployed on a web application whose component diagram is shown in Figure 5.2. A screenshot of the running application is show in Figure 5.3. Table 5.1 displays the number of lines of code written by the author to create the entire system.

The technologies used to create the application were:

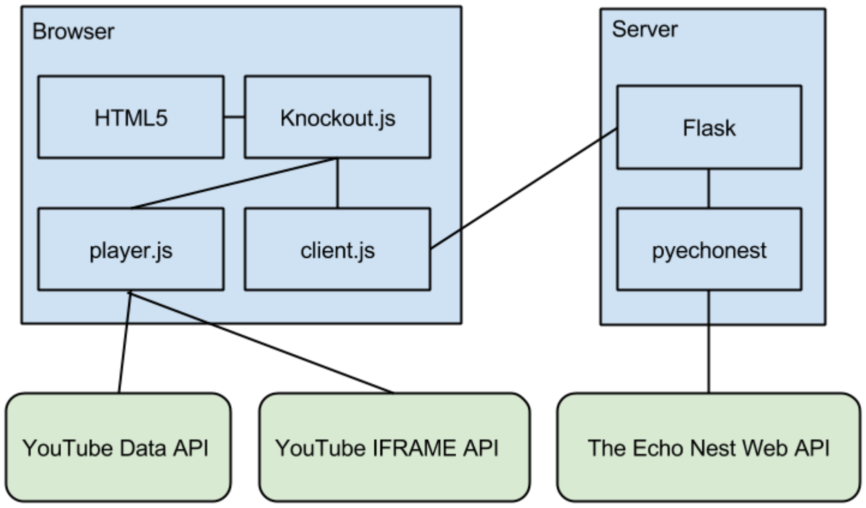
- Backend
  - Python version 2.7.5 (ROSSUM)
  - CherryPy (web framework) version 3.3.0 (CHERRYPY TEAM)
  - pyechonest (Python wrapper of The Echo Nest Web API) version 8.0.2 (DAUBMAN)
- Frontend
  - HTML/CSS
  - JavaScript
  - Knockout.js, MVVM, version 3.1.0 (SANDERSON)
  - YouTube Data API version 3 (GOOGLE)
  - YouTube IFRAME API (GOOGLE)

Table 5.1: Lines of code written to create the evaluation system.

Language	*Lines of Code*
Python	212
JavaScript	530
HTML	83
CSS	267
TOTAL	**1092**

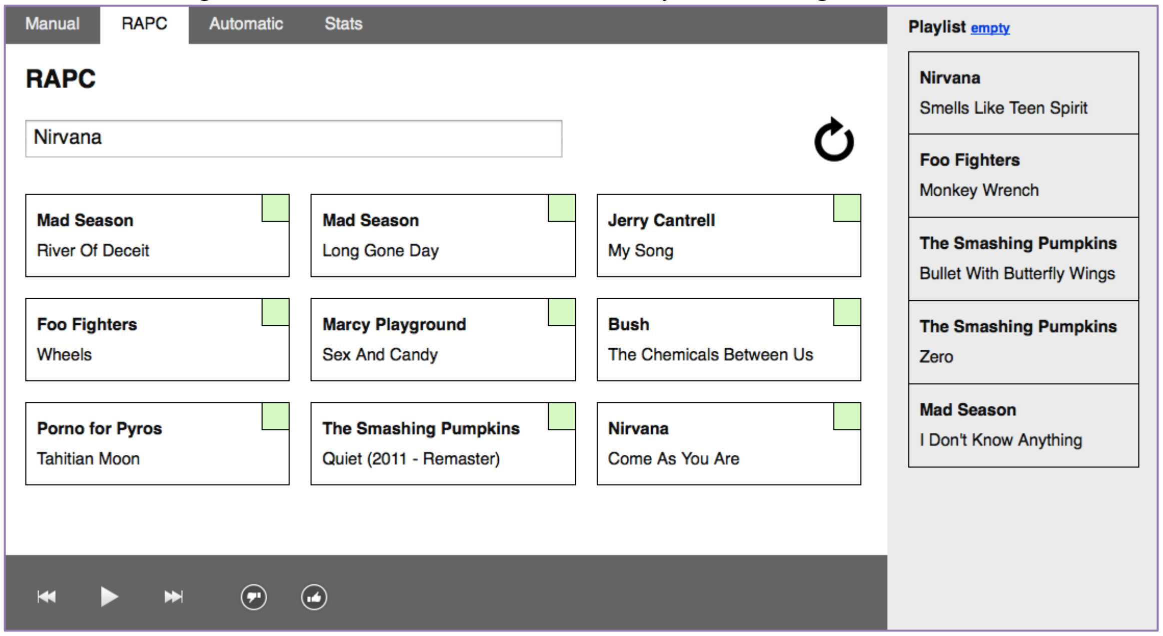
Fonte: Salle (2014)

Figure 5.2: Component diagram of the evaluation system.



Fonte: Salle (2014)

Figure 5.3: Screenshot of the evaluation system running in a browser.



RAPC

Nirvana

Mad Season
River Of Deceit

Mad Season
Long Gone Day

Jerry Cantrell
My Song

Foo Fighters
Wheels

Marcy Playground
Sex And Candy

Bush
The Chemicals Between Us

Porno for Pyros
Tahitian Moon

The Smashing Pumpkins
Quiet (2011 - Remaster)

Nirvana
Come As You Are

Playlist

Nirvana
Smells Like Teen Spirit

Foo Fighters
Monkey Wrench

The Smashing Pumpkins
Bullet With Butterfly Wings

The Smashing Pumpkins
Zero

Mad Season
I Don't Know Anything

Fonte: Salle (2014)

6 RESULTS

6.1 Participants

The target population was: users between the age of 20 and 35 who regularly listen to music online. The author asked his acquaintances who fit the population profile to take part in this study. In all, eleven people participated. Users were not told what hypotheses were being tested. They were simply told the experiment's protocol. None had ever used RAPC. The evaluations were conducted in-person by the author.

6.2 Descriptive Statistics

All statistics and graphs were generated using R version 3.1.0 (THE R FOUNDATION).

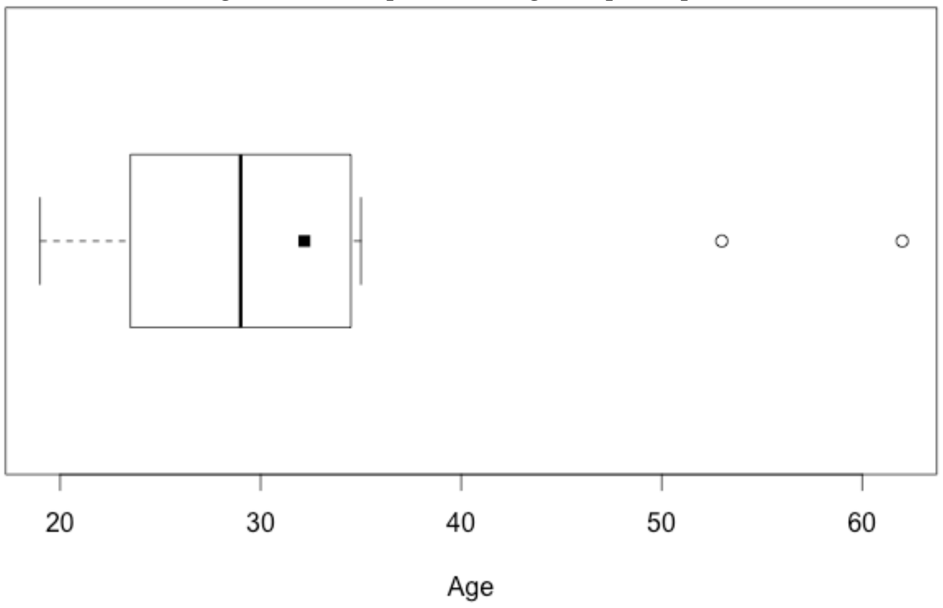
Box plots should be interpreted as follows:

- The box starts at the first quartile and ends at the third quartile.
- The band in the middle of the box marks the second quartile (the median).
- Data to the right of the left whisker is within 1.5 IQR (interquartile range) of the first quartile. Data to the left of the right whisker is within 1.5 IQR of the third quartile.
- Data which is more than 1.5 IQR from the first and third quartile is drawn as outlined dots.
- The mean is drawn as a solid black square.

6.2.1 Participants

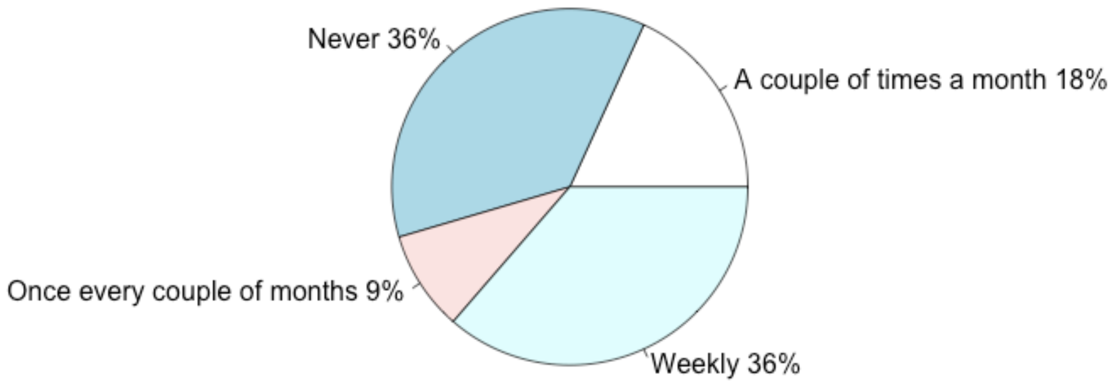
Participants were asked to report their age. The results are illustrated in Figure 6.1. With the exception of 2 outliers, participants were within the target population of 20-to-35-year-olds. Participants were also asked how often they create playlists, be it in online music services or in traditional desktop music players. Figure 6.2 shows the results.

Figure 6.1: Box plot of the ages of participants.



Fonte: Salle (2014)

Figure 6.2: Self-reported frequency with which participants create playlists.

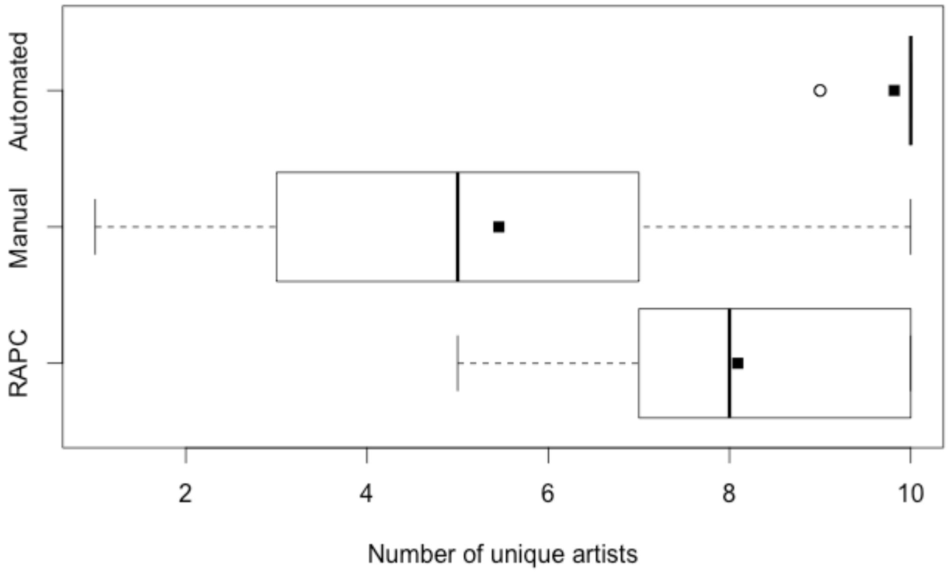


Fonte: Salle (2014)

6.2.2 Playlist Data

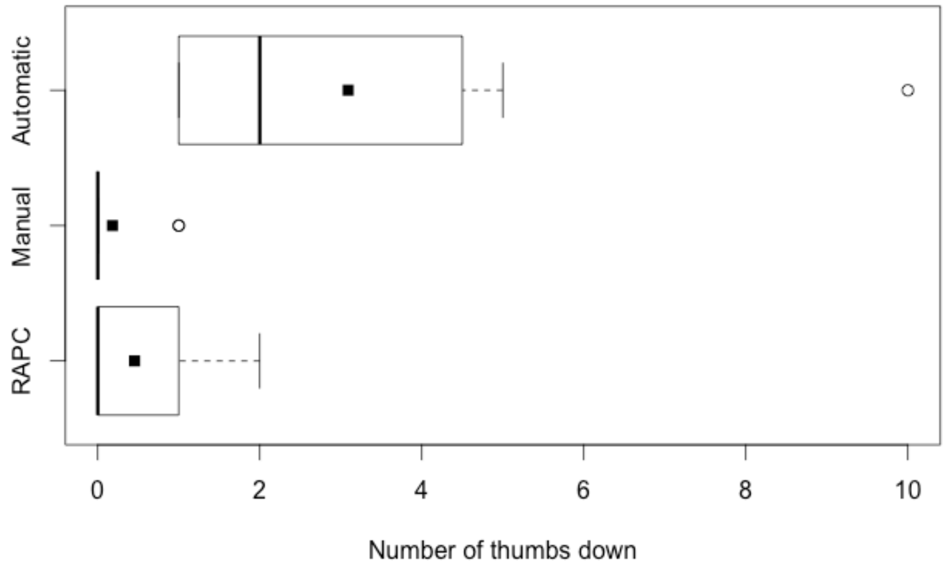
The number of unique artists and of tracks thumbed down was tracked for each playlist created using all 3 creation mechanisms. Figure 6.3 and Figure 6.4 show the results. For each playlist, participants were asked how many of the 10 tracks they had not listened to in the 2 months prior to the experiment. Figure 6.5 shows the results.

Figure 6.3: Number of unique artists in playlists created using different mechanisms.



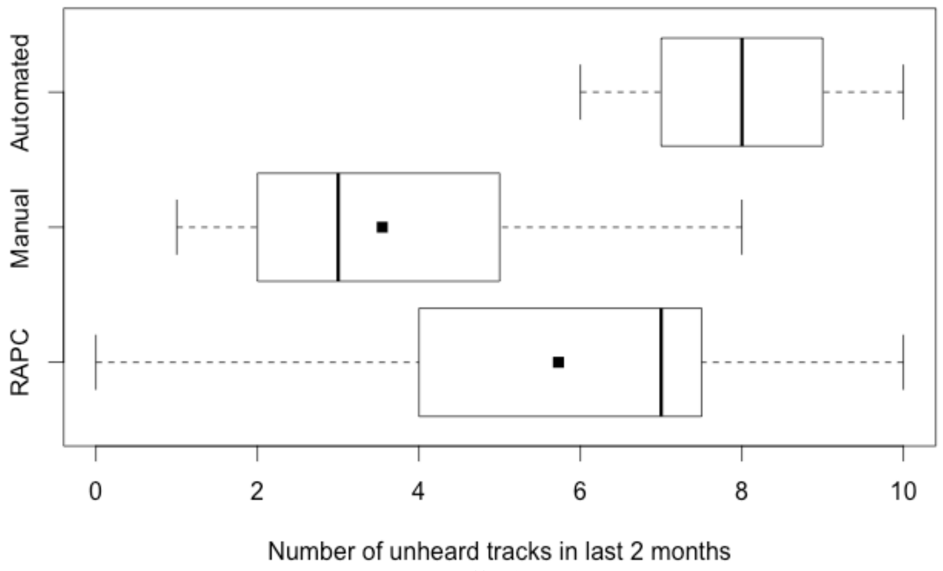
Fonte: Salle (2014)

Figure 6.4: The number of tracks thumbed down by users in playlists created using different mechanisms.



Fonte: Salle (2014)

Figure 6.5: The number of tracks in each playlist that users have not listened to ever or in the last 2 months.



Fonte: Salle (2014)

6.2.3 Speed-up

The time taken to create each playlist was tracked. The time taken to create automated playlists is very close to zero, since all it takes is a response from The Echo Nest's Web API. It is therefore not considered in this section.

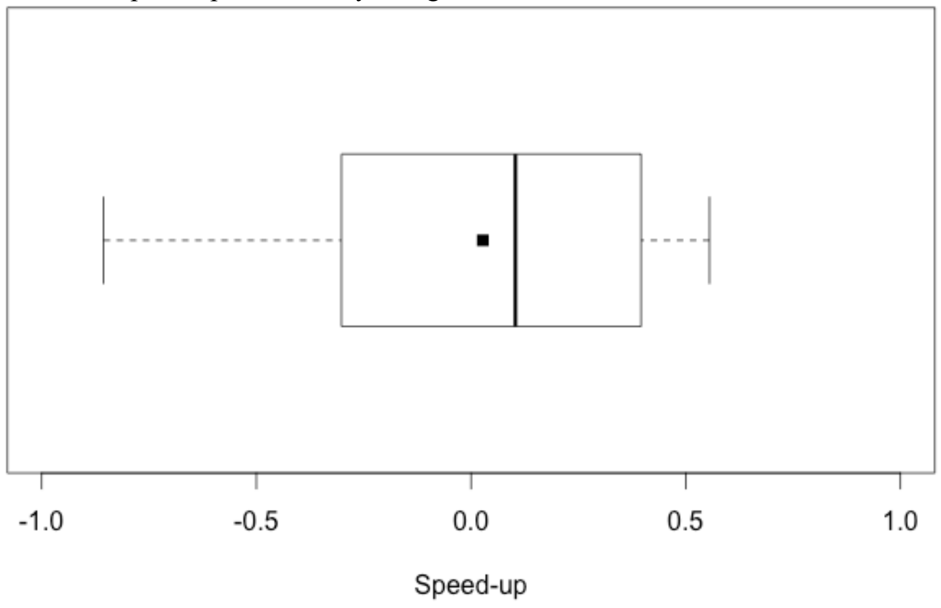
The time difference between RAPC and manual was calculated. This difference was used to calculate a speed-up metric using Equation 1. Figure 6.6 plots the speed-up of all participants.

The number of searches user performed when creating the playlists was tracked and is shown in Figure 6.7.

Equation 1: Formula used to calculate the speed-up obtained by using RAPC over manual.

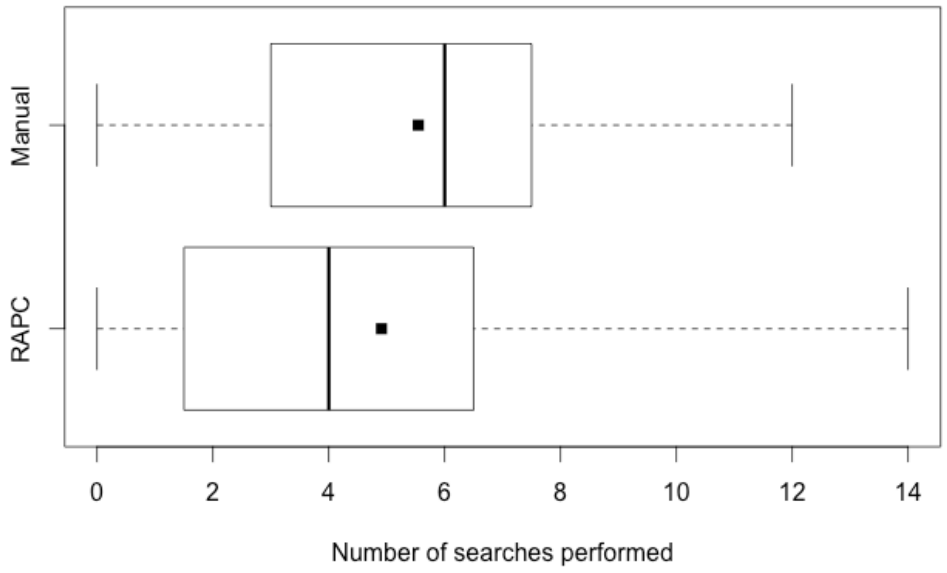
$$speedup = (manual\_time - rapc\_time) / manual\_time$$

Figure 6.6: The speed-up obtained by using RAPC mechanism over the manual mechanism.



Fonte: Salle (2014)

Figure 6.7: The number of searches users performed when creating playlists. Automated is omitted because the number is always zero.

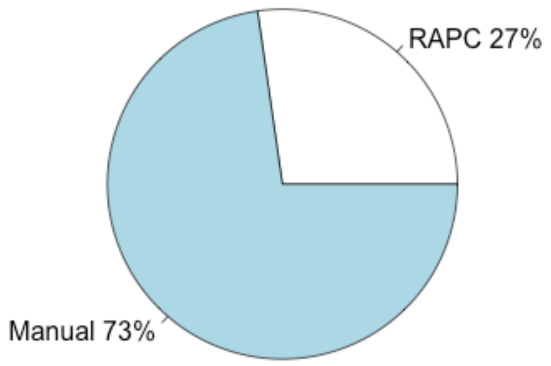


Fonte: Salle (2014)

6.2.4 User Opinion

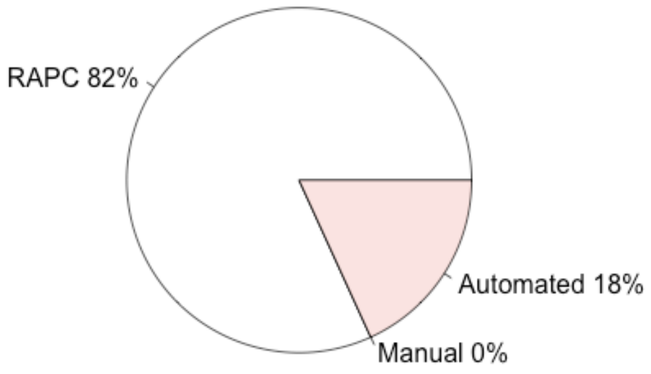
Once participants had created all three playlists, they were asked whether they found the RAPC or manual mechanism more tiring. Results are shown in Figure 6.8. They were also asked which of the 3 playlists they preferred: the one created using RAPC, manual, or automated mechanism. Figure 6.9 shows the results.

Figure 6.8: 73% of participants reported that creating a playlist using the manual mechanism was more tiring than using RAPC.



Fonte: Salle (2014)

Figure 6.9: 82% of participants reported that their favorite playlist was the one created using RAPC. 18% preferred the playlist generated by the automated mechanism. None of the users preferred the manual playlists.



Fonte: Salle (2014)

6.3 Hypothesis Testing

Since the sample size is small (11 cases), it is not safe to make assumptions about the distribution of data. Tests such as the Student's t-test, which assume normal distributions, are therefore not adequate. According to (MOORE, MCCABE and CRAIG, 2007), nonparametric tests should be used. Proportions are tested using the R's binomial.test function, and locations are tested using the wilcox.exact function found in the exactRankTests package. This study uses a α level of 0.05. Given the experimental results, hypothesis whose p-value is less than 0.05 are rejected. The results of all tests are shown in Table 6.1.

Table 6.1: The results of all hypothesis tests. Null hypotheses with p-values < 0.05 were rejected.

	Null Hypothesis	*Alternate Hypothesis*	*Test*	*p-value*
1	**Users prefer RAPC playlists over manual and automated playlists at most 50% of the time**	**Users prefer RAPC over manual and automated playlists more than 50% of the time**	**Binomial**	**0.03271**
2	Users find RAPC more tiring than manual at least 50% of the time	Users find RAPC less tiring than manual more than 50% of the time	Binomial	0.1133
3	**RAPC playlists contain less or the same number of artists as manual playlists**	**RAPC playlists contain more artists than manual playlists**	**Wilcoxon Signed-rank**	**0.001953**
4	**RAPC playlists contain less or the same number tracks the user has not heard in the last 2 months as manual playlists**	**RAPC playlists contain more tracks the user has not heard in the last 2 months than manual playlists**	**Wilcoxon Signed-rank**	**0.003906**
5	Users perform more or the same number of searches using RAPC as manual	Users perform less searches using RAPC than manual	Wilcoxon Signed-rank	0.2671
6	**RAPC playlists contain more or the same number of tracks thumbed down by users as automated playlists**	**RAPC playlists contain less tracks thumbed down by users than automated playlists**	**Wilcoxon Signed-rank**	**0.0009766**
7	**Manual playlists contain more or the same number of tracks thumbed down by users as automated playlists**	**Manual playlists contain less tracks thumbed down by users than automated playlists**	**Wilcoxon Signed-rank**	**0.0004883**

8	The speed-up of using RAPC over manual is zero or negative	The speed-up of using RAPC over manual is positive	Wilcoxon Signed-rank	0.3501

Fonte: Salle (2014)

7 DISCUSSION

The author hypothesized that playlist creation using RAPC would be faster than manual, but that was not the case. Figure 6.6 shows that speed-up is almost centered on zero. Consistent with this plot analysis, test #8 fails to reject the null that RAPC is slower or as slow as manual. The author hypothesized that users would find RAPC less tiring than manual. Though Figure 6.8 seems to indicate that, it could not be proven, as shown in test #2, which fails to reject the null. A study with a larger sample may be able to prove that RAPC is less tiring. Figure 6.7 shows the number of searches users performed when creating playlists. There is no clear difference between RAPC and manual. As expected, test #5 fails to prove that users perform less searches using RAPC over manual. This could explain why there's no apparent speed-up in using RAPC and why RAPC is not provably less tiring than manual.

The author also hypothesized that RAPC playlists would lead to less skips than automated playlists. The assumption is made that a thumbed down track is equivalent to a skipped track. This hypothesis can thus be rephrased as: RAPC playlists lead to less thumbs down than automated playlists. Figure 6.4 makes a strong case for this hypothesis, and it is proven to be true by test #6.

The evaluation revealed an unexpected result: users prefer RAPC playlists over both manual and automated playlists more than 50% of the time, as was strongly indicated by the results shown in Figure 6.9 and proven in test #1. The author gathers this is the case because of 3 other results: (1) RAPC leads to lower thumbs down counts as stated above. (2) As proven by test #3, RAPC playlists possess more artists than manual. (3) As proven by test #4, RAPC playlists posses more tracks that the user has not listened to recently.

8 CONCLUSION

This study provides evidence that recommendation-assisted playlist creation leads to better playlists than manual and automated approaches. The evaluation's weak point is the sampling method, which relied on the author's acquaintances. A more rigorous sampling method would make the results here presented more representative of the target population of 20-to-35-year-old online music listeners. As was already noted, however, the participants of the study were not aware of the hypotheses being tested, and so could not attempt to favor the author in validating one or another, as shown by the failure to prove the author's main hypothesis that RAPC playlist creation is faster than manual creation.

Future work will focus on more representative population sampling and using iterative human-computer interaction processes to develop a better implementation of RAPC.

9 REFERENCES

AIZENBERG, N.; KOREN, Y.; SOMEKH, O. **Build Your Own Music Recommender by Modeling Internet Radio Streams**. WWW. Lyon: [s.n.]. 2012.

BARRINGTON, L.; ODA, R.; LANCKRIET, G. **SMARTER THAN GENIUS? HUMAN EVALUATION OF MUSIC RECOMMENDER SYSTEMS.** International Society for Music Information Retrieval. [S.l.]: [s.n.]. 2009.

CHERRYPY TEAM. CherryPy. Avalaible at: <http://www.cherrypy.org/>.

DAUBMAN, A. pyechonest. Avalaible at: <https://github.com/echonest/pyechonest>.

GOOGLE. YouTube Data API. Avalaible at: <https://developers.google.com/youtube/iframe_api_reference>.

GOOGLE. YouTube IFRAME API. Avalaible at: <https://developers.google.com/youtube/iframe_api_reference>.

LOGAN, B.; SALOMON, A. **A Content-Based Music Similarity Function**. [S.l.]. 2001.

MOORE, D.; MCCABE, G.; CRAIG, B. **Nonparametric Tests**. [S.l.]: W. H. Freeman, 2007. 15-1 p.

RICCI, F.; ROKACH, L.; SHAPIRA, B. **Introduction to Recommender Systems Handbook**. [S.l.]: Springer, 2011. 1-35 p.

ROSSUM, G. V. Python. Avalaible at: <https://www.python.org/>.

SANDERSON, S. Knockout.js. Avalaible at: <http://knockoutjs.com/>.

SARWAR, B. et al. **Item-Based Collaborative Filtering Recommendation Algorithms**. WWW10. Hong Kong: [s.n.]. 2001.

SPOTIFY. Spotify. **http:** //www.spotify.com/.

SWEARINGEN, K.; SINHA, R. **Interaction Design for Recommender Systems**. DIS2002. London: [s.n.]. 2002.

THE ECHO NEST. The Echo Nest. Avalaible at: <http://the.echonest.com/>. Accessed on: 12 jul. 2014.

THE R FOUNDATION. The R Project for Statistical Computing. Avalaible at: <http://www.r-project.org/>.

WESTERGREN, T. The music genome project, 2007. Avalaible at: <http://www.pandora.com/about/mgp>. Accessed on: 8 may 2014.

ZIEGLER, C.-N. et al. **Improving Recommendation Lists Through Topic Diversification**. WWW 2005. Chiba: [s.n.]. 2005.