ROPE and STRAW: Automatic Music Playlist Generators

Marcos A. de Almeida
Universidade Federal de Minas Gerais
Belo Horizonte, Brazil
marcos.almeida@dcc.ufmg.br

Carolina C. Vieira
Universidade Federal de Minas Gerais
Belo Horizonte, Brazil
carolcoimbra@dcc.ufmg.br

Pedro O. S. Vaz de Melo
Universidade Federal de Minas Gerais
Belo Horizonte, Brazil
olmo@dcc.ufmg.br

Renato M. Assunção
Universidade Federal de Minas Gerais
Belo Horizonte, Brazil
assuncao@dcc.ufmg.br

ABSTRACT

With the growth of multimedia content available on internet to users, music playlist generators is gaining importance nowadays. People search for ways to generate enjoyable playlists satisfying their tastes, and that suit the situation in which they are. Although many authors have proposed methods to generate music playlists satisfying certain restrictions, defining the qualities a good playlist should have is a difficult task. This paper revise and presents the implementation of a general method presented on literature that generates heterogeneous music playlists based on a simple input given by the user. The method was constructed to satisfy five quality constraints: heterogeneity, smooth transitions, novelty, usability and scalability. The tool is available on a website, where is posible to choose between two algorithms named ROPE and STRAW, and listen on YouTube to the playlist generated.

KEYWORDS

Playlist generator, heterogeneous playlists, web application

1 INTRODUÇÃO

Ouvir música é uma das formas de entretenimento mais utilizada pelas pessoas no dia a dia. Com o crescimento de serviços de *streaming* de música, como Spotify, Microsoft Groove e Google play music, usuários possuem acesso a milhões de música para ouvirem quando quiser. Uma consequência desse fácil acesso a músicas é o crescente número de playlists criadas por usuários para serem escutadas em diversas ocasiões, como em festas, durante sessões de exercícios na academia, ou até mesmo em uma viagem. Porém, para um usuário, criar uma playlist que seja agradável e se adeque bem à situação em que ela será tocada pode se tornar uma tarefa complicada e que demanda bastante tempo [6, 8]. Muitos usuários possuem gosto musical restrito, isto é, apreciam somente um pequeno grupo de estilos musicais. Por outro lado, alguns usuários possuem um gosto musical mais amplo, desejando ouvir uma variedade de gêneros diferentes. Até mesmo para um grupo de pessoas heterogêneas, isto é, um grupo de pessoas com gostos variados reunidos por algum motivo específico, como uma festa por exemplo, criar uma playlist que satisfaça o gosto musical de todos é uma tarefa difícil.

Vários autores propõem formas e métricas para dizer se uma playlist é considerada boa ou não. Alguns autores retratam que para uma playlist ser considerada boa ela deve ser homogênea [6] enquanto outros autores conceituam que uma playlist não precisa necessariamente ser homogênea, desde que as músicas estejam de acordo com a situação em que irão ser tocadas. De fato, definir uma métrica para dizer o quão boa é uma playlist é algo complexo, já que depende não somente de quais músicas estão na playlist, mas também da ordem em que são tocadas. Em [4] foram definidas cinco critérios para avaliar a qualidade de um gerador de playlists, que são *heterogeneidade*, *transições suaves*, *novidade*, *escalabilidade* e *usabilidade*. Baseado nestes critérios, os autores propuseram um método geral para se gerar playlists de músicas e criaram dois algoritmos baseados no método, denominados ROPE e STRAW.

Neste trabalho descrevemos a implementação de um sistema web que executa os algoritmos propostos em [4]. A partir de uma entrada definida por um usuário, o sistema gera automaticamente uma playlist de músicas que satisfaça as qualidades mencionadas acima e possibilita que o usuário ouça a playlist gerada no YouTube. O restante deste artigo está organizado da seguinte forma. Na seção 2 apresentamos os trabalhos relacionados. Na seção 3 descrevemos os dados utilizados no trabalho, e na seção 4 realizamos uma breve explicação dos algoritmos utilizados. Na seção 5 apresentamos os detalhes da implementação. Finalmente, na seção 6 apresentamos as conclusões do trabalho.

2 TRABALHOS RELACIONADOS

Vários autores já propuseram algoritmos para o problema da geração automática de playlists de músicas [3, 5, 9, 12, 13]. Um dos primeiros a propor uma solução para o problema foi [3] que modelou o problema como programação linear com dois parâmetros: a música inicial e a música final e duas restrições. Da mesma forma, [5] propuseram uma técnica de otimização restrita para gerar playlists em grandes coleções de músicas. Porém ambos os métodos não são escaláveis (violando a *escalabilidade*) e sempre geram a mesma playlist (não satisfazendo a *novidade*). Já [9] propôs um método simples e escalável para se gerar playlists, porém o método é determinístico, sempre gerando a mesma playlist e não satisfazendo a restrição de *novidade*.

Outra abordagem é o caminhamento em grafos, como abordado por [12], que propuseram utilizar o algoritmo do Caixeiro Viajante

In: XVI Workshop de Ferramentas e Aplicações (WFA 2017), Gramado, Brasil. Anais do XXIII Simpósio Brasileiro de Sistemas Multimídia e Web: Workshops e Pôsteres. Porto Alegre: Sociedade Brasileira de Computação, 2017.

ISBN 978-85-7669-380-2.

em um grafo de similaridade de música, formando uma ordem sequencial das músicas da base de dados. Porém este método se torna inviável para um grande conjunto de músicas, violando a *escalabilidade*. Ragno et al. [13] construíram um grafo ponderado a partir de dados de estações de rádio, onde o peso da aresta é o número de vezes que as músicas são tocadas em sequência. Porém tal método tende a permanecer em um grupo de músicas similares, não criando playlists heterogêneas.

Quando se trata de aplicações web, existem várias ferramentas online com o propósito de gerar playlists de músicas baseadas em uma música semente escolhida pelo usuário, como por exemplo o *MagicPlaylist* [1] e o *Spotibot* [2]. Porém estas aplicações tendem a criar playlists homogêneas, como o MagicPlaylist, que seleciona várias músicas do mesmo artista semente.

3 DADOS UTILIZADOS

Os dados utilizados neste trabalho são os mesmo utilizados por [11]. Cada música é caracterizada por quatro grupos de características: harmonia, timbre, ano de lançamento e gênero musical. A fim de colocar todos os grupos na mesma escala, transformamos cada grupo em uma distribuição de probabilidade, isto é, todas as características do mesmo grupo somam 1. Como tanto as características de timbre quanto as características de harmonia já somavam 1, modificamos somente as características do ano de lançamento e gênero musical. Para o grupo do gênero musical, para cada música dividimos o vetor de gêneros pela quantidade de gêneros que ela possuía. Já para o grupo do ano de lançamento, transformamos o ano de lançamento y_i da música em um vetor de duas posições onde a primeira é o valor $\hat{y}_i$ normalizado entre 0 e 1, isto é, $\hat{y}_i = \frac{y_i - min(y_i)}{max(y_i) - min(y_i)}$ e a segunda é o complemento do ano normalizado. Com isso, cada música é caracterizada por um vetor de características $x_i = (h_i, t_i, g_i, y_i)$. Utilizando a técnica de redução de dimensionalidade t-SNE [10], reduzimos a dimensionalidade dos dados para um espaço euclideano 2D, que foi definido como espaço de músicas, e pode ser visualizado na figura 1. Cada ponto representa uma música, que foi colorido de acordo com o gênero da mesma. Caso uma música possua mais de um gênero, o ponto que representa a música foi colorido com o gênero menos popular. Para colorir os pontos foi utilizado somente os 11 gêneros que mais ocorreram nos dados, cobrindo aproximadamente 85% das músicas. As demais músicas foram coloridas com cinza. Na figura também podemos ver duas playlists geradas pelos métodos implementados, passando como parâmetro a mesma música inicial e final e a mesma quantidade de músicas na playlist.

Para complementar os dados, utilizamos uma API do YouTube [1] para procurar por IDs de vídeos das músicas que temos no nosso banco de dados, para que possamos criar no YouTube a playlist gerada para que o usuário possa escutá-la, conforme mostrado na figura 2.

4 ALGORITMOS UTILIZADOS

Os algoritmos que podem ser utilizados na ferramenta de geração de playlists são denominados de ROPE (***Br**ownian **P**ath **g**enerator*) e STRAW (***St**eered **Ra**ndom **W**alker*), os mesmos descritos em [4].

[1] https://developers.google.com/youtube/1.0/developers_guide_python

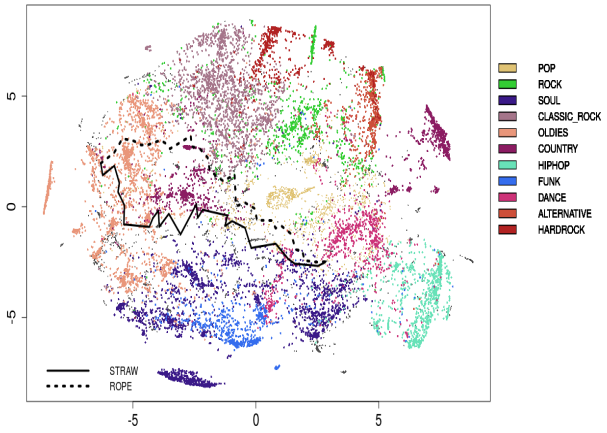


Figura 1: O espaço de músicas gerado e duas playlists geradas pelos métodos ROPE e STRAW. Cada ponto representa uma música que é colorida de acordo com seu gênero. Deve ser visualizada em cores.

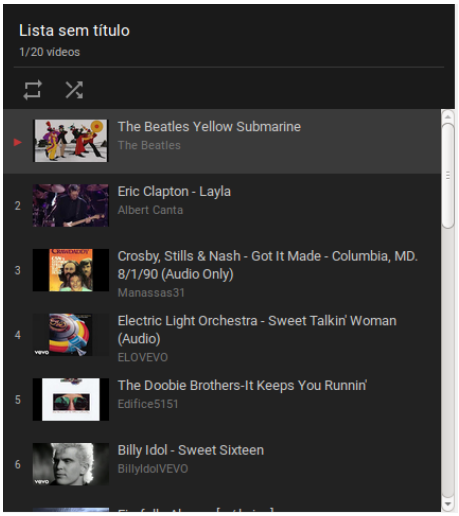


Figura 2: Exemplo de playlist sendo executada no YouTube após ter sido gerada pela ferramenta. Deve ser visualizada em cores.

O método geral para gerar uma playlist possui como entrada três parâmetros, sendo uma música inicial s_0, um vetor direção x_d [2] e um inteiro k que é a quantidade de músicas que se deseja ter na playlist. Com essa entrada, o algoritmo seleciona iterativamente uma música para ser adicionada na playlist baseada na última música que foi adicionada, de forma a satisfazer as métricas desejadas. No algoritmo 1 está descrito o método geral para se gerar as playlists.

[2] O gerador de playlists implementado recebe uma música do espaço de músicas como vetor direção

Algorithm 1 Gerando playlists heterogêneas.

```
1: procedure GERA(S, s_0, x_d, k)          ▷ uma playlist heterogênea com k músicas
2:     n ← 1                               ▷ número de músicas na playlist
3:     while n < k do
4:         P ← F(s_{n-1})                  ▷ músicas elegíveis para adicionar
5:         s_n ← c(P, x_d)                 ▷ adiciona a nova música à playlist
6:         n ← n + 1
       return s
```

A função $F(s_{n-1})$ seleciona um conjunto de músicas elegíveis a serem adicionadas na n-ésima posição da playlist dada a música s_{n-1} da playlist. Já a função $c(\mathcal{P}, \vec{x_d})$ seleciona aleatoriamente uma música dentre o conjunto de músicas retornadas pela função $F(s_{n-1})$.

Ambos os algoritmos seguem o modelo geral descrito. A diferença entre o ROPE e o STRAW está na forma de caminhar no espaço de músicas. No ROPE é gerado um caminho a partir de um movimento browniano com passos discretos na reta unidimensional, fazendo com que haja exatamente $k-2$ pontos entre a música inicial e final. Após gerar o caminho, é realizado uma transformação linear das coordenadas dos pontos de forma que o caminho ligue a música inicial e a música final no espaço de músicas. Já no STRAW, é criado um grafo ponderado de forma que o peso das arestas é dado pela distância euclidiana entre os pontos que representam as músicas no espaço de músicas e, a partir da música inicial, caminha-se no grafo até encontrar a música final. Caso a música final seja alcançada com menos do que $k-1$ passos, é realizado um movimento aleatório no grafo a partir da música final até que se tenha exatamente k músicas na playlist. Caso seja alcançada com mais do que $k-1$ passos, isto é, com k' passos, são removidas $k'-k$ músicas da playlist preservando ao máximo a qualidade de *transições suaves*.

5 IMPLEMENTAÇÃO

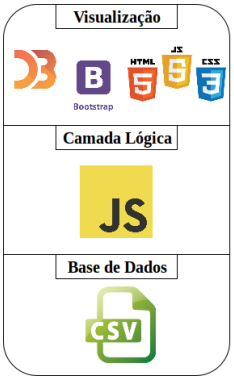


Figura 3: Diagrama representando a arquitetura da ferramenta. Deve ser visualizada em cores.

Após a extração e tratamento dos dados, bem como o desenvolvimento do modelo geral e dos algoritmos ROPE e STRAW, implementamos um site através do qual é possível gerar playlists a partir de três entradas definida por usuários. Basicamente, o usuário escolhe o algoritmo que deseja utilizar na geração de sua playlist e, em seguida escolhe duas músicas dentre as que estão na base de dados (uma música inicial e uma música final) e a quantidade de músicas que deseja ouvir. Na figura 3 apresentamos um diagrama da arquitetura do site que foi construído, em que podemos visualizar as tecnologias utilizadas em cada uma das etapas de desenvolvimento.

Na camada de visualização há um formulário para que o usuário escolha as músicas inicial e final e a quantidade de músicas que se deseja ter na playlist. Após definir essas entradas, a listagem das músicas contidas na playlist criada é mostrada ao usuário, como mostra a figura 4. Ao mostrar a playlist gerada, se o usuário quiser ouví-la, há um botão que o redireciona para uma página do YouTube com a playlist gerada, conforme representado na figura 2 que mostra no Youtube, a mesma playlist da figura 4. O link para o YouTube é criado concatenando os IDs dos vídeos separados por uma vírgula, e adicionando a string ao final do link "*https://www.youtube.com/watch_videos?video_ids=*". Devido a uma limitação do YouTube, a playlist no mesmo possui somente as primeiras 50 músicas da playlist gerada.

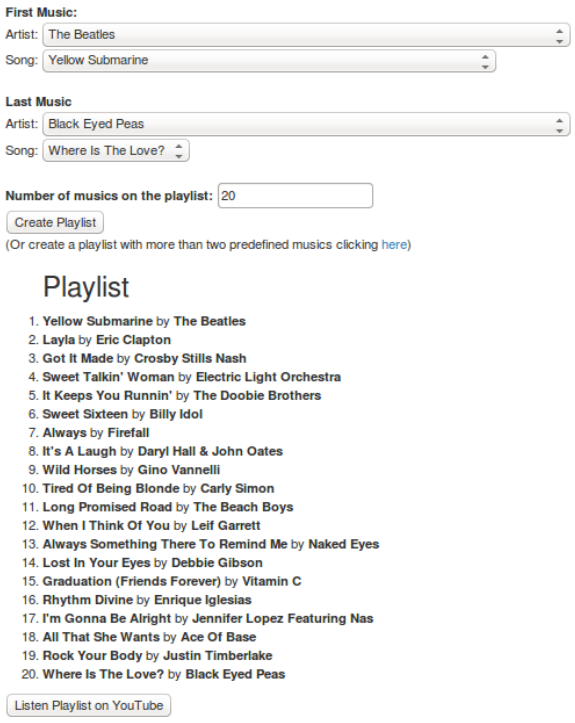
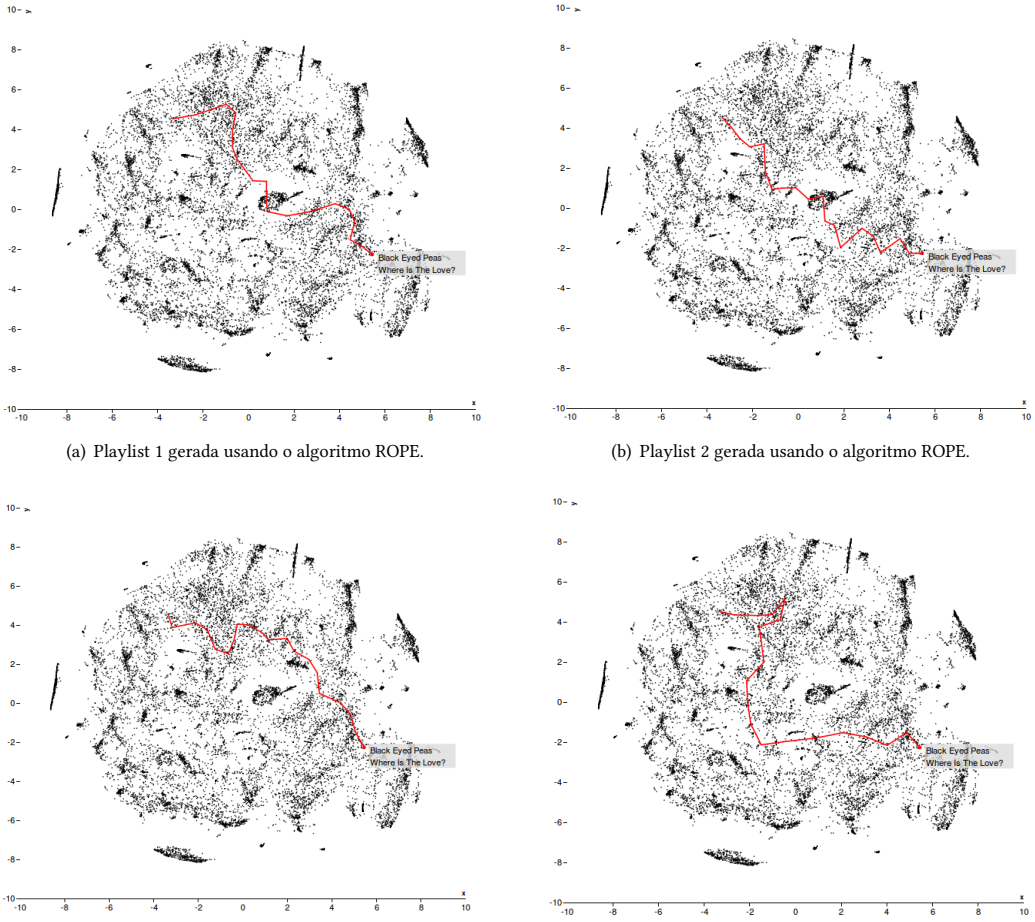


Figura 4: Exemplo de playlist gerada com o algoritmo ROPE.

Após a geração da playlist também é criada uma visualização interativa que mostra no espaço de músicas a playlist gerada a partir da entrada do usuário. A visualização é construída utilizando uma biblioteca de Javascript, D3, através da qual é possível criar visualizações interativas em websites. Além disso, essa implementação permite que, ao utilizar o mouse, por meio dos *tooltips*, o usuário veja o nome de cada uma das músicas que compõem a playlist, bem como a sua posição no espaço de músicas, conforme mostrado na figura 5.

A camada lógica é responsável por realizar a comunicação entre as camadas de visualização e de dados. Dessa forma, além de carregar os dados necessários para o funcionamento da ferramenta, ela



(a) Playlist 1 gerada usando o algoritmo ROPE.

(b) Playlist 2 gerada usando o algoritmo ROPE.

(c) Playlist 1 gerada usando o algoritmo STRAW.

(d) Playlist 2 gerada usando o algoritmo STRAW.

Figura 5: Caminho de duas playlists geradas com os algoritmos ROPE e STRAW. As playlists possuem 20 músicas e começam com a música Yellow Submarine dos The Beatles e terminam com a música Where Is The Love? do Black Eyed Peas. Deve ser visualizado em cores.

é também responsável por executar os algoritmos ROPE e STRAW, que por sua vez, estão diretamente ligados à camada de visualização. Toda essa camada foi implementada utilizando o Javascript.

No algoritmo ROPE, o Javascript carrega as coordenadas das músicas no espaço de músicas e, a partir das coordenadas da música inicial e final e o número de músicas k que se deseja ter na playlist, gera um movimento browniano com $k - 1$ passos. A transformação de Box-Muller [7] foi utilizada para gerar os ruídos gaussianos do movimento browniano. Após a geração do movimento browniano foi realizada uma transformação linear nas coordenadas de forma que o caminho gerado conecte a primeira e a última música escolhida.

Já no algoritmo STRAW, além das coordenadas das músicas no espaço de músicas, o Javascript carrega o arquivo que contém a lista de adjacência de cada um dos nós do grafo modelado a partir do espaço de músicas. Após carregar os dados, o algoritmo STRAW funciona como um caminho aleatório dirigido que parte da música inicial até a música final, determinada pelo usuário. Para isso, a partir da última música adicionada à playlist, o algoritmo busca entre seus vizinhos aquelas músicas que minimizam a distância entre a próxima música a ser adicionada e a música final. Entretanto, como desejamos ter um algoritmo não-determinístico, é feita uma distribuição de probabilidade entre os vizinhos de forma a atribuir maior probabilidade aos que mais se aproximam da música final, conforme explicado em [4].

Uma outra funcionalidade presente no sistema quando se utiliza o método ROPE é a possibilidade de se escolher mais de duas músicas para compor a playlist, isto é, o usuário pode escolher músicas intermediárias na playlist. Neste caso, é possível definir a quantidade de músicas que se deseja ter entre as músicas escolhidas. Como exemplo temos a figura 6, onde escolhemos como música inicial Dear Dad de Chunck Berry e como música final Miami de Will Smith, mas passando pela música Highway to Hell do grupo AC/DC. O mapa com a playlist gerada pode ser visto na figura 7.

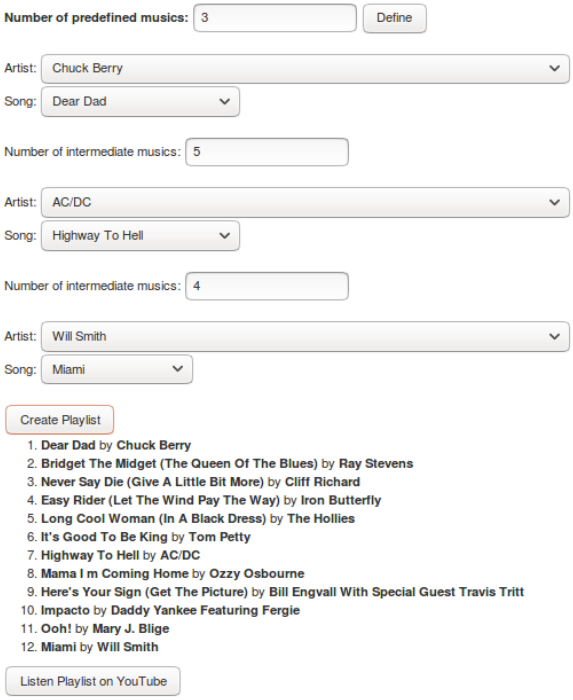


Figura 6: Exemplo de playlist gerada com o algoritmo ROPE definindo músicas intermediárias.

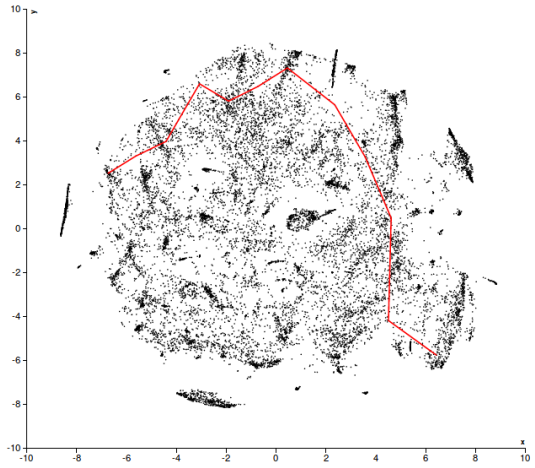


Figura 7: Exemplo de caminho da playlist gerada com o algoritmo ROPE definindo músicas intermediárias. Deve ser visualizada em cores.

Por fim, todos os dados utilizados pela ferramenta, o que inclui as coordenadas das músicas no espaço de músicas e as informações do grafo de similaridade, foram armazenados em arquivos no formato comma-separated values (CSV). Optamos por utilizar esta forma de armazenamento, pois todos os dados foram pré-processados, não sendo necessário carregar uma grande quantidade de informação para executar os algoritmos implementados.

6 CONCLUSÃO

Neste trabalho apresentamos uma ferramenta para geração automática de playlists de músicas que satisfazem cinco características desejáveis, que são *heterogeneidade*, *transições suaves*, *novidade*, *escalabilidade* e *usabilidade*. A partir de uma música inicial e final e a quantidade de músicas que se deseja ter na playlist, que são entradas definidas pelo usuário, é gerado uma playlist que parte da música inicial, e vai até a música final. Com a aplicação também é possível visualizar a playlist gerada no espaço de músicas.

As playlists personalizadas geradas por outros sistemas, como o Spotify, são baseadas no gosto musical dos usuários e, portanto, são criadas a partir de um conjunto de músicas que possívelmente agradará o mesmo, não se preocupando com a ordem que as músicas serão reproduzidas. Já o foco do sistema implementado é criar uma playlist que possua uma sequência de músicas que ligue de forma suave as músicas de um subconjunto definido por um usuário, idealmente de gêneros diferentes, mantendo transições suaves.

Como trabalhos futuros, pretendemos extrair e analisar dados de músicas do Spotify, e adaptar os algoritmos implementados para que as playlists possam ser geradas e ouvidas no próprio sistema do Spotify, que é um serviço próprio para se escutar músicas, diferente do YouTube, cujo objetivo são clipes de músicas. Também pretendemos realizar testes com usuários finais para se testar a usabilidade do sistema implementado e a qualidade das playlists geradas. Além disso, uma funcionalidade interessante que pretendemos incorporar à ferramenta seria o compartilhamento das playlists geradas entre os usuários do nosso sistema. Por fim, ressaltamos que o sistema implementado e as futuras melhorias possuem um objetivo puramente acadêmico e sem nenhum fim lucrativo.

REFERÊNCIAS

[1] Magic Playlist. https://magicplaylist.co. (????). Accessed: 2017-08-07.

[2] Spotibot. https://www.spotibot.com/. (????). Accessed: 2017-08-07.

[3] Masoud Alghoniemy and Ahmed H Tewfik. 2000. User-defined music sequence retrieval. In *Proceedings of the eighth ACM international conference on Multimedia*. ACM, 356–358.

[4] Marcos Almeida, Carolina Vieira, Pedro Olmo Vaz de Melo, and Renato Assuncao. 2017. The Fast and Winding Roads that Lead to The Doors: Generating Heterogeneous Music Playlists. In *WebMedia 2017 - Full and Short papers ()*. Gramado, RS.

[5] J-J Aucouturier and François Pachet. 2002. Scaling up music playlist generation. In *Multimedia and Expo, 2002. ICME'02. Proceedings. 2002 IEEE International Conference on*, Vol. 1. IEEE, 105–108.

[6] Geoffray Bonnin and Dietmar Jannach. 2015. Automated generation of music playlists: Survey and experiments. *ACM Computing Surveys (CSUR)* 47, 2 (2015), 26.

[7] George EP Box, Mervin E Muller, et al. 1958. A note on the generation of random normal deviates. *The annals of mathematical statistics* 29, 2 (1958), 610–611.

[8] Ricardo Dias, Daniel Gonçalves, and Manuel J Fonseca. 2017. From manual to assisted playlist creation: a survey. *Multimedia Tools and Applications* 76, 12 (2017), 14375–14403.

[9] Arthur Flexer, Dominik Schnitzer, Martin Gasser, and Gerhard Widmer. 2008. Playlist Generation using Start and End Songs.. In *ISMIR*, Vol. 8. 173–178.

[10] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of Machine Learning Research* 9, Nov (2008), 2579–2605.

[11] Matthias Mauch, Robert M MacCallum, Mark Levy, and Armand M Leroi. 2015. The evolution of popular music: USA 1960–2010. *Royal Society open science* 2, 5 (2015), 150081.

[12] Tim Pohle, Elias Pampalk, and Gerhard Widmer. 2005. Generating similarity-based playlists using traveling salesman algorithms. In *Proceedings of the 8th International Conference on Digital Audio Effects (DAFx-05)*. 220–225.

[13] Robert Ragno, Christopher JC Burges, and Cormac Herley. 2005. Inferring similarity between music objects with application to playlist generation. In *Proceedings of the 7th ACM SIGMM international workshop on Multimedia information retrieval*. ACM, 73–80.