Repetitividade em músicas brasileiras

Athos B. Lagemann
Universidade Federal do Rio
Grande do Sul
Porto Alegre, RS, Brazil
athos.lagemann@inf.ufrgs.br

ABSTRACT
This article has as main objective to present, succintly, a variation of Colin Morris' work on Music repetition, where the author uses some techniques presented by Colin to try and show the same effect in Brazilian music. Another technique is used to reassure Morris' opinion that it does not generates reliable results.

Este artigo tem como principal objetivo apresentar, de forma sucinta, uma variação do trabalho feito por Colin Morris sobre repetição em músicas, onde o autor utiliza-se de uma das técnicas apresentadas por Colin para mostrar o mesmo efeito em músicas brasileiras. A outra técnica utilizada pelo autor foi para reforçar o ponto de Morris de que a mesma não traz resultados corretos ou consistentes com o observado.

Palavras-chave do autor
Lyrics; Repetitiveness; Brazil; Billboard; Vagalume; Lyric visualization; Letras; Repetição; Brasil; Visualização de letras.

Palavras-chave de Classificação da ACM
H.5.m. Data analysis and presentation: Miscellaneous.

INTRODUÇÃO

Este trabalho foi baseado no trabalho desenvolvido por Colin Morris [6] [7] [8], que trata sobre o fator da repetição nas músicas mais populares da *Billboard*.

Morris analisou as 100 músicas mais populares de diversas décadas para demonstrar o quão repetitivas as músicas ficaram ao longo dos anos. Para tal, ele utilizou duas técnicas distintas:

1. Medir a taxa de compressão de cada música: a diferença no tamanho do arquivo antes e depois da compressão apresenta fortes indícios do quão repetitiva a música é. Em [2], há uma explicação mais detalhada de como estes algoritmos funcionam, mas este não é o foco deste trabalho.
2. Gerar uma matriz de adjacências: montando a matriz de adjacências a partir da letra da música, é possível ter uma boa noção visual de como a música repete certos trechos em determinados momentos, apresentando padrões de repetição interessantes. Este é o principal foco deste trabalho.

Como será explicado mais adiante, não foi possível implementar o método de compressão de músicas. Alternativamente, optei por utilizar um método que Morris disse não ter eficaz para determinar a repetição das músicas, que leva em conta apenas a repetição da letra.

Um conjunto de dez músicas analisadas por Colin foi separado para fazer a análise de repetição com o meu método, e o resultado de ambos foi posto em comparação. De fato, como afirmado por Morris, este método é ineficiente.

Tanto o método das matrizes de adjacências quanto este método de contagem de palavras serão analisados mais detalhadamente nas próximas seções.

Este artigo é divido nas seguintes seções: a Introdução trata de um resumo geral do trabalho feito e o que é coberto no artigo; a Caracterização do Problema trata do problema em si, como apresentado por Colin Morris e da minha proposta para este trabalho; as Técnicas e Ferramentas utilizadas falam sobre o que foi utilizado para a implementação, bem como os métodos usados para gerar cada um dos artefatos finais de visualização; o Resultados trata da execução do projeto e do que foi obtido a partir dela, bem como uma comparação com o que foi demonstrado por Colin Morris em sua palestra [6]; e finalmente, a Conclusão trata de forma mais aprofundada sobre a comparação dos meus resultados com os do Colin, os impedimentos técnicos encontrados durante o trabalho e os trabalhos futuros a respeito deste problema.

CARACTERIZAÇÃO DO PROBLEMA

Em sua excelente palestra na TEDx [6], Colin Morris mostra duas formas de como visualizar a repetição em músicas. Em seus exemplos, ele utilizou como *dataset* as 100 músicas pop mais ouvidas da *Billboard*. Sua conclusão foi de que as músicas mais ouvidas estão ficando cada vez mais repetitivas, ao longo das décadas.

O primeiro método utilizado por Colin se baseia em formar uma matriz de adjacência das palavras, a partir da letra da música. Este método é comumente utilizado em bioinformática para o sequenciamento do genoma humano, e foi reproduzido por mim, de forma semelhante.

O segundo método apresentado por Colin foi o cálculo baseado na taxa de compressão do arquivo mp3 da música. Como eu não tenho acesso legal e gratuito aos arquivos das músicas, não pude reproduzir este método. Contudo, tentei contorná-lo utilizando outra técnica, que será explicada mais adiante.

Como a questão da repetição em músicas foi demonstrado por Colin apenas considerando as top 100 músicas da *Billboard* ao longo das décadas, optei por fazer algo parecido, porém considerando o top 100 geral das rádios brasileiras. Além disso, analisei também os top 100 de dois dos estilos musicais mais populares no Brasil: sertanejo e funk.

Matriz de Adjacência

A matriz é formada a partir da letra da música, onde cada elemento em ambos os eixos são as palavras, e cada ponto da matriz em que a palavra encontra ela mesma é preenchida. Isso constrói uma matriz simétrica, onde cada hemisfério da diagonal é exatamente igual ao hemisfério oposto.

Colin, utilizou esta mesma técnica, mas a partir do contexto utilizado na bioinformática. Nesta técnica, o eixo das linhas começa no topo da matriz e vai descendo, e as colunas começam na esquerda e vão para a direita. A versão que eu reproduzi utiliza as mesmas regras, para manter a verossimilhança, mas utiliza outro espaço de cores.

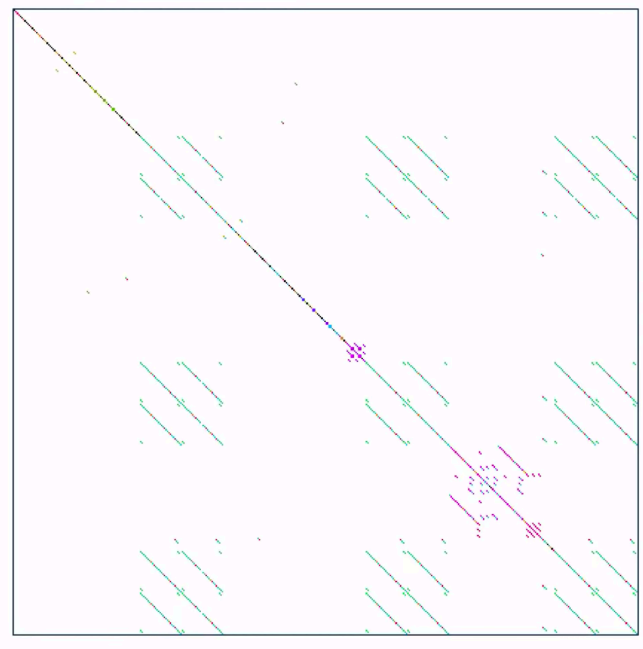

Figura 1. Ke$ha - Tik Tok, como apresentado por Colin Morris em [1].

Neste espaço de cores, as primeiras palavras a aparecer têm um tom mais escuro, e à medida que novas palavras vão aparecendo, sua tonalidade vai ficando mais clara.

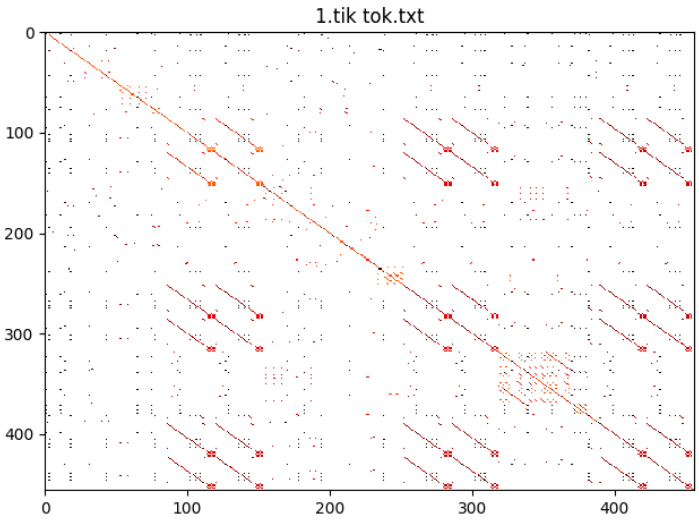

Figura 2. Ke$ha - Tik Tok, como apresentado pelo autor, a partir do método de grafos de adjacência

Este espaço de cores nos permite ter uma dimensão a mais sobre a matriz de adjacência, podendo perceber quando cada palavra ou trecho da música aparece e é repetido em detrimento a outras palavras, e sobre a ordem de aparição das palavras dentro da letra.

Com isto, é possível visualizar padrões inicialmente imperceptíveis apenas analisando-se a letra da música ou as ocorrências das palavras.

Compressão de arquivos

Outro método utilizado por Colin consiste em comprimir o arquivo *mp3* de cada música e comparar a variação do tamanho. O algoritmo de compressão atua em cima de padrões de repetição do conteúdo do arquivo, de modo que músicas mais repetitivas acabam sendo mais comprimidas.

Por exemplo, a música *Around the World* do *Daft Punk* apresentou uma diferença de 98% em seu tamanho, após a compressão.

Colin utilizou este como o principal método para calcular a repetição das músicas, e encontrou padrões interessantes, como a repetitividade média das 100 músicas mais ouvidas ao longo das décadas.

Contudo, este método me é inviável, pois eu desconheço uma maneira eficiente e legal de obter os arquivos *mp3* necessários para efetuar o cálculo. Ao invés disso, optei por utilizar um método alternativo, que Colins disse não ser muito preciso.

Contagem de palavras

Como Colin afirma no início da sua palestra [6], apenas contar a repetição das palavras não é nenhuma garantia de que a música resultante será repetitiva.

O argumento utilizado por ele é de que a repetição de uma palavra específica não necessariamente indica que a música será repetitiva, pois isto também depende de outros fatores externos à letra, como melodia e entonação das palavras.

TÉCNICAS E FERRAMENTAS UTILIZADAS

O ambiente de desenvolvimento predominantemente utilizado foi o Mac OS X, mas teoricamente o sistema em si pode rodar em qualquer ambiente UNIX sem grandes problemas.

O sistema utiliza uma combinação de *scripts* e códigos em *Python 3*. Para executá-lo, é necessário possuir alguns Frameworks instalados e conectividade com a internet, como será explicado mais adiante.

O código final está disponível na plataforma *Github*, seguindo o link [1]. Na documentação do projeto consta o embasamento da ideia, os objetivos, como instalar os Frameworks e APIs, e como executar e visualizar os resultados. O trabalho ainda será melhorado e atualizado futuramente.

APIs e Frameworks

Para obter as letras das músicas, foi utilizada a API do Vagalume. É uma API gratuita, com suporte a *Python 3*, e pode ser utilizada muito facilmente. Contudo, dois fatores complicam a obtenção das letras: o primeiro é que, como ela não é tão completa quanto seria desejável, ela não possui todas as músicas dos top 100, tanto gerais quanto dos gêneros específicos. O segundo fator é a origem dos conjuntos analisados: [9], [10] e [11]. Como os sites de origem não têm como objetivo servirem de base de dados, os mesmos não são práticos para a obtenção dos rankings em si. Na prática, os conjuntos analisados contaram com, em média, entre 60 e 67% das músicas buscadas.

Para plotar os grafos, foi utilizado o Framework de *Python 3* chamado *Matplotlib*. Ele permite a geração de gráficos programaticamente, e por ser bastante documentado, a sua utilização não é muito complicada.

Para melhorar a visualização das requisições do script, foi utilizado o Framework *Colorama*. Este Framework basicamente troca a cor do texto no terminal, apenas com o intuito de facilitar o acompanhamento do script.

Para a geração dos gráficos finais, foi utilizado o Google Planilhas. Apenas por ser prático para a geração dos gráficos.

Técnicas utilizadas

O projeto é dividido em três scripts em *Python 3*, cada qual com sua funcionalidade:

1. *main.py*: tem como único objetivo ler o arquivo de entrada contendo os pares <*artista*, *música*>, fazer as requisições das letras correspondentes, e salvá-las em duas formas: a letra pura no diretório <lyrics>, e a letra dividida por palavras no diretório <songs>.
2. *graph.py*: lê cada um dos conjuntos de palavras de cada música do diretório <songs> e monta a estrutura da matriz correspondente, que é colocada em um *plot* usando-se o *matplotlib.* Após este processo, o arquivo do *plot* contendo a matriz de adjacências é salvo no diretório <graphs>.
3. *repCount.py*: acessa cada um dos conjuntos de palavras de cada música do diretório <songs>, salva cada palavra como a chave de um dicionário que contém o número de aparições da palavra na letra. À medida que as palavras vão se repetindo, o valor da chave vai sendo incrementado. Após preencher o dicionário, ele calcula a porcentagem de palavras que estão acima do limiar médio de aparições e salva esse dado em um arquivo chamado *repetitiveness.txt*.

Esses três *scripts* em Python são executados em sequência por um *script* em bash, que também gerencia os diretórios nos quais os arquivos serão salvos.

Adicionalmente, há um *script* em Python para desfazer as mudanças de cores de letra no terminal, e um script bash chamado *reset,* que limpa os diretórios e cache do projeto.

Para executar os scripts, é necessário possuir a linguagem Python 3 instalada no ambiente, bem como o gerenciador de pacotes *pip3*, que pode ser obtido através do seguinte comando:

```
$ sudo apt-get Install python3
```

Para instalar a API e os Frameworks necessários, basta executar o gerenciador de pacotes recém obtido com:

```
$ pip3 install numpy matplotlib vagalume colorama
```

Uma vez que tudo estiver corretamente instalado, é necessário obter os dados desejados para a análise propriamente dita. Para tal, deve-se fornecer um arquivo de texto contendo os artistas e as músicas no seguinte formato:

```
artista -> música
```

Todos os pares artista-música devem estar separados por uma quebra de linha. O formato do arquivo de entrada deve possuir a extensão *.txt*, pois é o formato que será lido pelo script.

Para executar o arquivo salvo, basta chamar o script com o seguinte comando, considerando o que o nome do arquivo é *músicas.txt*:

```
./lyrics músicas
```

O script então começará a sua execução lendo o arquivo *músicas.txt* e criando os diretórios necessários dentro do novo diretório chamado *músicas_dir*.

Para limpar as execuções já feitas, basta executar o script *clean*:

```
./clean
```

Esse script irá limpar todos os diretórios terminados por *_dir* a partir do diretório atual, e irá reiniciar o *colorspace* do terminal atual, que por ventura pode ter sido indevidamente alterado pelo script *main.py*.

RESULTADOS

O script foi rodado para os seguintes três casos: o top 100 das rádios brasileiras [9], os top 100 sertanejos mais ouvidos [10], os top 100 funks mais ouvidos [11], e um pequeno conjunto de músicas [7], para comparar com o conjunto analisado por Morris.

Devido a limitações da API do Vagalume e ao método usado para obter as top 100 músicas de cada gênero, não foi possível obter todas as 100 músicas intencionadas para cada conjunto. Ao invés, foram obtidas 60 dos top 100 geral, 64 dos top 100 funks e 67 dos top 100 sertanejos. O conjunto do Colin, por ser menor, pode ser completamente obtido e analisado, totalizando 10 itens. Este último conjunto será doravante chamado "conjunto de prova", para facilitar a nomenclatura.

Quanto aos resultados em si, serão abordados os três casos analisados, em ordem: análise por matriz de adjacência, análise por contagem de palavras e análise por compressão. Apenas os dois primeiros casos foram abordados por mim, pois o último foi meramente analisado.

Matriz de Adjacência

O método da matriz foi reproduzido de acordo com o demonstrado por Colin Morris, sendo ela construída de cima para baixo, da esquerda para a direita. O espaço de cores utilizado foi outro, pois Colin não disponibilizou a ferramenta que ele utilizou, e o Matplotlib não apresentou nenhuma opção parecida.

A seguir, é possível comparar a demonstração de Morris (*figura 1*) com a minha implementação (*figura 2*). Ambas ficaram bem semelhantes em forma, comprovando que o método foi suficientemente similar. As pequenas diferenças provavelmente se dão devido ao *parsing* da letra e à API através da qual ela foi obtida.

Quanto aos conjuntos de músicas analisados, surgiram outras matrizes com padrões interessantes de se observar, que podem ser conferidos a seguir. O caso de repetição mais icônico é o da música *Around the World*, do *Daft Punk* (figura 3). Notavelmente esta música não possui refrão, verso, ponte, nem outros elementos comumente encontrados na maioria das músicas.

As músicas do conjunto de funk também apresentaram padrões repetitivos bem interessantes. A seguir, foram separados três exemplos interessantes de cada um dos conjuntos:

1. O conjunto dos top 100 possui as três seguintes matrizes: *Espaçosa Demais*, do Felipe Araújo (figura 4), *Ouvi Dizer* de Melim (figura 5) e *Adrenalizou* do Vitor Kley (figura 6).
2. O conjunto de funk possui *Só quer Vrau* do MC MM (figura 7), *Kikadinha*, do Jerry Smith (figura 8) e *Menina Braba*, também do Jerry Smith (figura 9).
3. Por fim, o conjunto de sertanejo apresenta *Aperte o Play* da Simone e Simaria (figura 10), *Disk Recaída* do PH e Michel (figura 11) e *Ainda sou tão seu* do Felipe Araújo (figura 12).

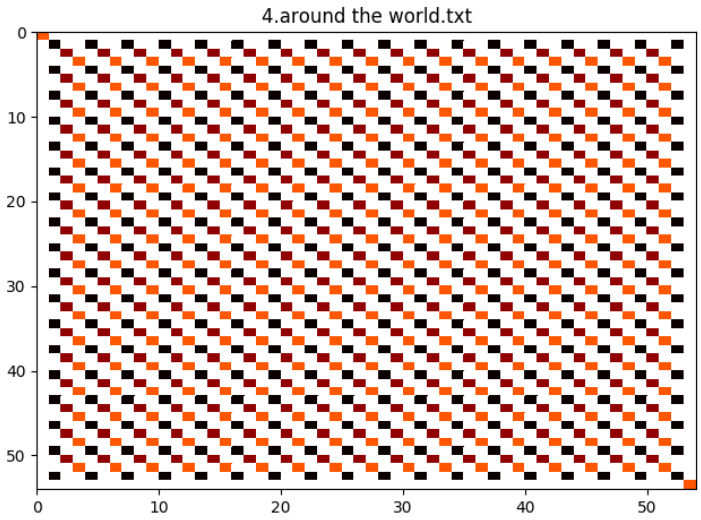

Figura 3. Daft Punk - Around the World, como apresentado pelo autor, a partir do método de grafos de adjacência

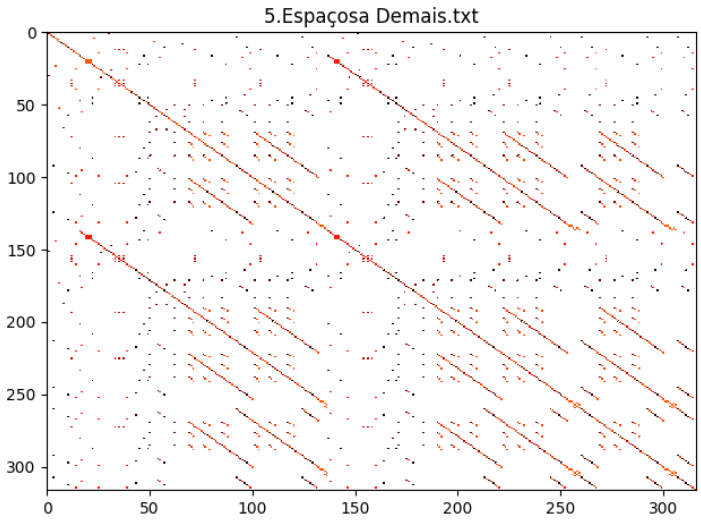

Figura 4. Espaçosa Demais, de Felipe Araújo, como apresentado pelo autor, a partir do método de grafos de adjacência

Contagem de palavras

O cálculo de repetição a partir da contagem de palavras foi feito para cada um dos conjuntos analisados. Contudo, comparando o conjunto de prova da contagem de palavras com o de compressão, pode-se verificar na tabela de

comparação dos métodos de contagem (tabela 1 e figura 13) que ambos não condizem o resultado obtido.

A partir disso, pode-se afirmar que os outros conjuntos não têm serventia prática, e portanto optei por manter apenas as matrizes de adjacências como objeto de análise.

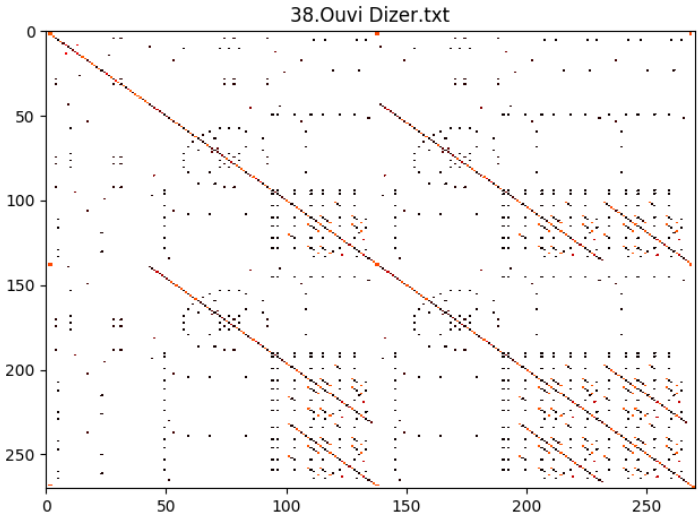

Figura 5. Ouvi Dizer, de Melim, como apresentado pelo autor, a partir do método de grafos de adjacência

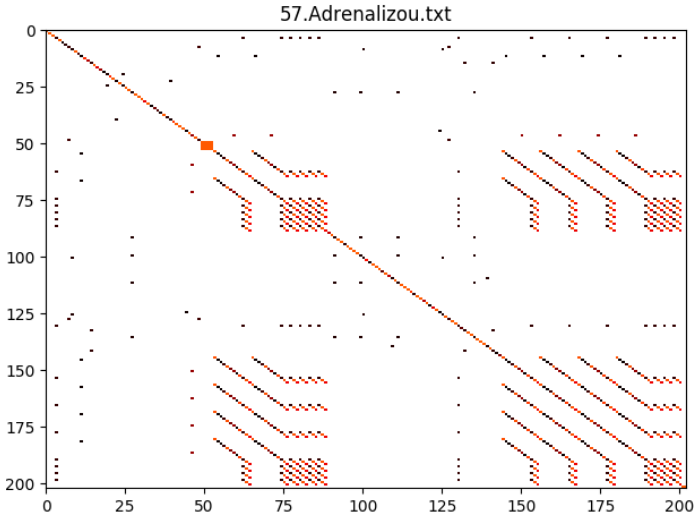

Figura 6. Adrenalizou, de Vitor Kley, como apresentado pelo autor, a partir do método de grafos de adjacência

CONCLUSÕES

Como pode ser verificado na tabela 1, o resultado das duas métricas de repetição foi consideravelmente discrepante, demonstrando que Colin estava correto quando afirmou a ineficácia do cálculo por palavras.

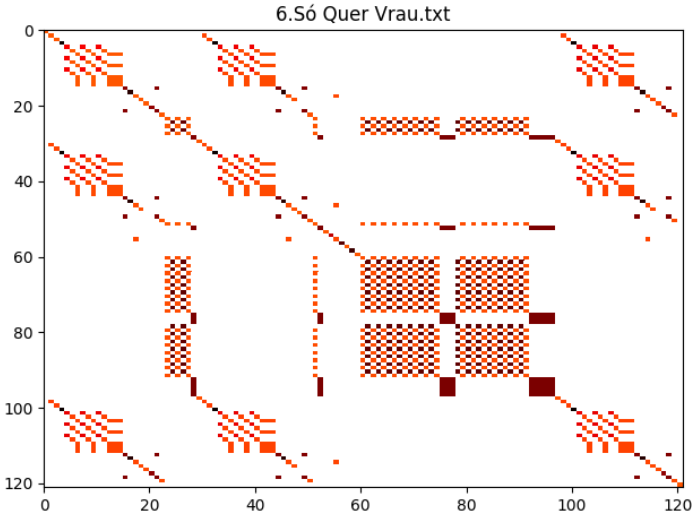

Figura 7. Só Quer Vrau, de MC MM, como apresentado pelo autor, a partir do método de grafos de adjacência

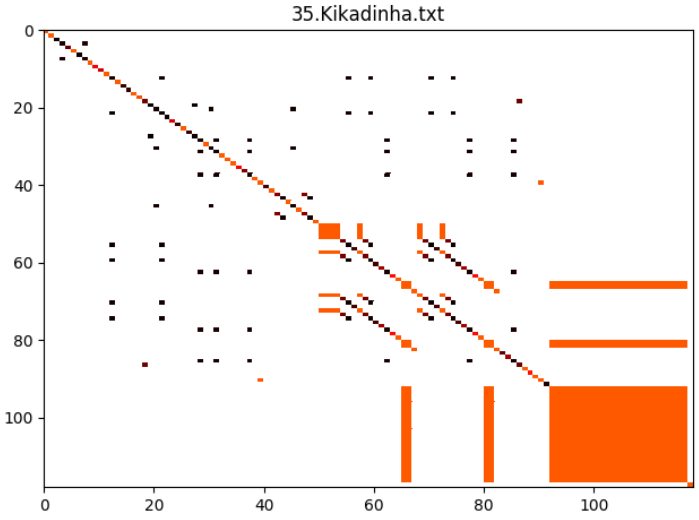

Figura 8. Kikadinha, de Jerry Smith, como apresentado pelo autor, a partir do método de grafos de adjacência

Contudo, obtive resultados interessantes através do método das matrizes de adjacência, sendo alguns exemplos demonstrados ao longo do artigo.

Limitações técnicas

Sobre as limitações técnicas encontradas: a primeira, como foi ressaltado ao longo deste artigo, se fundamenta na impossibilidade de obter os arquivos mp3 das músicas analisadas. Este método é comprovadamente eficaz e muito provavelmente traria resultados interessantes sobre os conjuntos de músicas no contexto do Brasil.

Outro aspecto limitante foi a API do Vagalume, utilizada para obter as letras respectivas dos conjuntos analisados. Pelo que pude observar, a plataforma do Vagalume não é tão completa quanto seria desejável.

Além disso, os rankings de músicas foram obtidos de sites em que não é exatamente prático de efetuar o *parse* dos artistas e títulos de músicas. A saber, a principal fonte das músicas foi o site *Mais Músicas*, que disponibiliza um ranking das músicas mais tocadas, tanto geral quanto separadas por gênero musical.

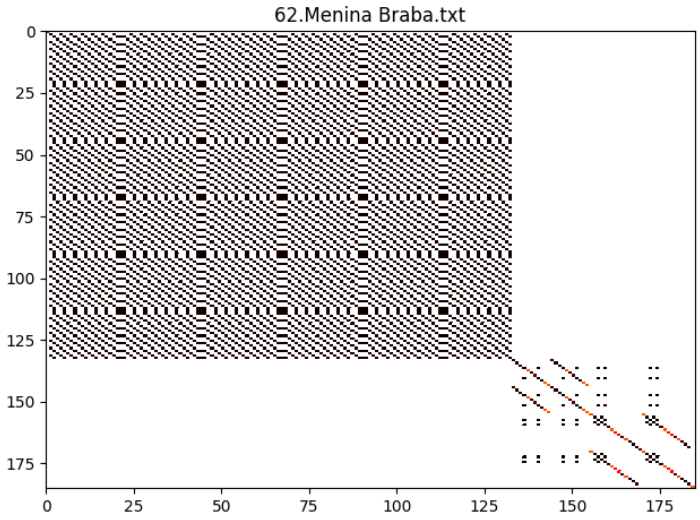

Figura 9. Menina Braba, de Jerry Smith, como apresentado pelo autor, a partir do método de grafos de adjacência

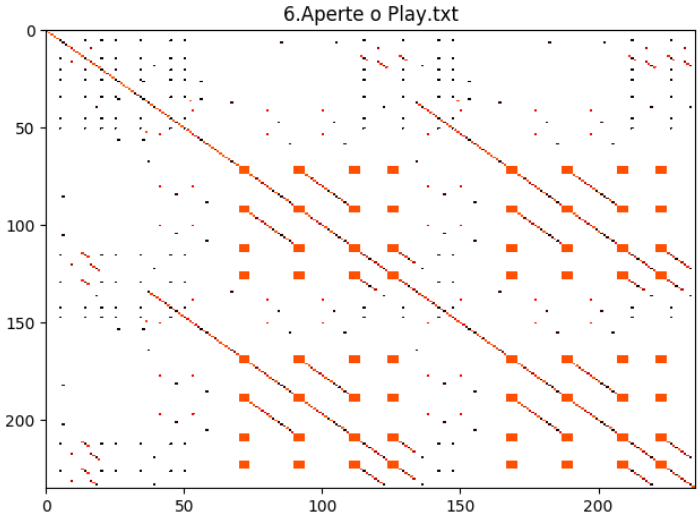

Figura 10. Aperte o Play, de Simone e Simaria, como apresentado pelo autor, a partir do método de grafos de adjacência

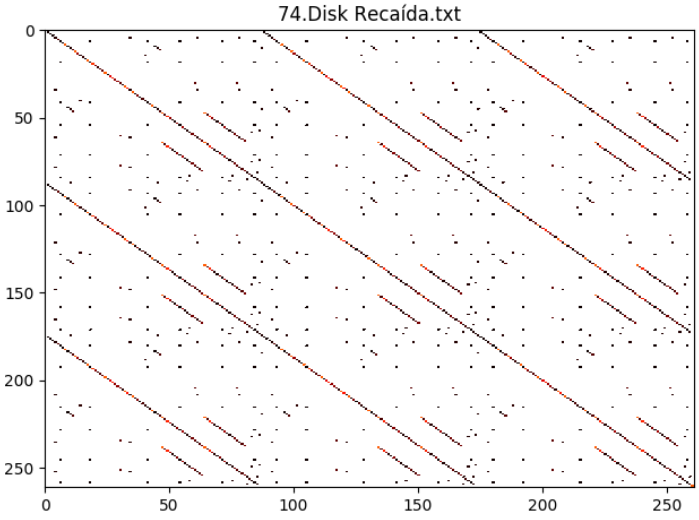

Figura 11. Disk Recaída, de PH e Michel, como apresentado pelo autor, a partir do método de grafos de adjacência

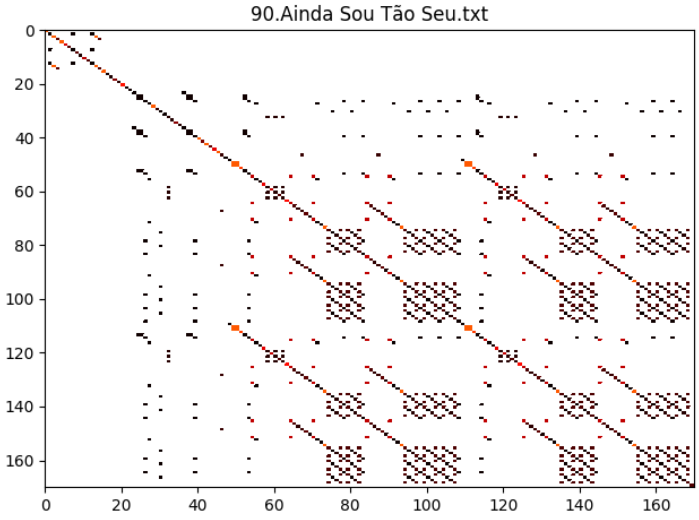

Figura 12. Ainda Sou Tão Seu, de Felipe Araújo, como apresentado pelo autor, a partir do método de grafos de adjacência

Trabalhos futuros

A minha intenção é ainda trabalhar em cada uma das limitações. Prioritariamente, obtendo uma fonte mais prática e conveniente para efetuar o *parse* dos rankings musicais, de modo a aumentar a taxa de músicas obtidas para cada conjunto.

Em seguida, seria interessante descobrir uma forma prática de obter uma cópia de todos os arquivos musicais necessários, de forma que possibilitasse a implementação do método de compressão dos arquivos.

Este passo talvez fosse o mais importante do projeto, pois possibilitaria uma análise confiável da repetição de cada música, bem como métricas precisas.

Vale lembrar que o projeto está disponível publicamente, sob uma licença livre. Revisões, comentários, implementações, *forks* e *Pull Requests* são bem-vindos. Além disso, você pode clonar o repositório na sua máquina e testar à vontade, seja com os conjuntos padrão ou com seus próprios.

Música	Compressão (Colin Morris)	Contagem de palavras (autor)
Tik Tok	60%	59%
Cheap Thrills	76%	35%
Can't get you out of my head	80%	47%
After Midnight	81%	71%
Macarena	82%	59%
You Make Me	82%	77%
Send me on my way	85%	61%
Turn Down for What	87%	73%
Get Low	90%	100%
Around The World	98%	100%

Tabela 1. Comparação das músicas do conjunto de prova, evidenciando a discrepância dos métodos utilizados

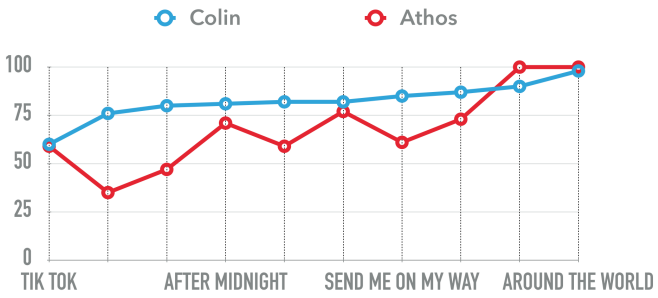

Figura 13. Comparação das músicas do conjunto de prova, evidenciando a discrepância dos métodos utilizados

REFERÊNCIAS

As referências estão distribuídas em duas seções: as que tratam do aspecto técnico do projeto, e as que tratam do aspecto teórico.

Aspecto Técnico:

1. Link do projeto no *Github*: https://github.com/athoslag/lyrics.
2. Documentação da linguagem *Python*, disponível em https://www.python.org/doc/.
3. Documentação do framework Matplotlib, disponível em https://matplotlib.org/.
4. Documentação da API do Vagalume, disponível em https://api.vagalume.com.br/docs/.
5. Referência em *Python* da API do Vagalume, desenvolvida por Diego Teixeira, disponível em https://github.com/diegoteixeir4/python-vagalume.

Aspecto Teórico:

6. Pop Music is Stuck on Repeat | Colin Morris | TEDx Penn, 2018, disponível em https://youtu.be/_tjFwcmHy5M. Acesso em Jul/2019.
7. Are Pop Lyrics Getting More Repetitive?, Colin Morris, 2017, https://pudding.cool/2017/05/song-repetition/. Acesso em Jul/2019.
8. Six Weird Pop Songs, Visualized; Colin Morris, 2017, https://colinmorris.github.io/blog/weird-pop-songs. Acesso em Jul/2019.
9. (*Dataset*) Top 100 músicas mais tocadas, Mais Músicas, https://maismusicas.mus.br/musicas-mais-tocadas/. Acesso em Jul/2019.
10. (*Dataset*) Top 100 músicas sertanejo mais tocadas, Mais Músicas, https://maismusicas.mus.br/musicas-sertanejas/. Acesso em Jul/2019.
11. (*Dataset*) Top 100 músicas funk mais tocadas, Músicas Mais Tocadas, https://www.musicasmaistocadas.mus.br/musicas-de-funk-mais-tocadas/. Acesso em Jul/2019.