Universidade Federal de Pernambuco
Centro de Informática

Graduação em Ciência da Computação

Recomendação musical para grupos baseada nas preferências individuais

Bruno Vitorino Cortez de Lira

Orientador: Sergio Ricardo de Melo Queiroz

Recife
Setembro, 2020

Dedico esse trabalho a todos que me deram apoio e acreditaram em mim.

Agradecimentos

Primeiramente, gostaria de agradecer aos meus pais por todo o apoio e ensinamentos ao longo da minha vida que me tornaram quem sou.

Agradeço também aos meus amigos que, sem dúvida alguma, foram pessoas importantíssimas para meu crescimento em diversos âmbitos da vida.

Um agradecimento especial aos meus professores de colégio e da UFPE que tornaram possível essa jornada até aqui.

Gostaria de agradecer também a todos que se disponibilizaram em contribuir para o desenvolvimento desse trabalho, em especial ao meu orientador e às pessoas que se ofereceram para participar do experimento. Por fim, gostaria de agradecer a cada um que de alguma forma contribuiu para que eu chegasse até esse momento.

A menos que modifiquemos a nossa maneira de pensar, não seremos capazes de resolver os problemas causados pela forma como nos acostumamos a ver o mundo

—ALBERT EINSTEIN

Contexto

A recomendação de items para grupos, apesar de ser um tema ainda pouco abordado, apresenta diversos casos de aplicação tais como filmes para um casal ou uma família, um local de viagem para um grupo de amigos ou, como será melhor abordado nesse trabalho, uma lista de músicas para um grupo de pessoas. Esse trabalho busca comparar algumas técnicas já utilizadas para recomendar items para grupos e analisar seus resultados quando são aplicadas ao contexto musical. Além disso, busca contribuir de alguma forma com uma nova abordagem que consiga obter resultados melhores dos que as técnicas previamente desenvolvidas. O conjunto de dados foi obtido usando um crawler que, através da API do Spotify, coleta dados sobre os usuários que permitiram tal coleta. Diversos processamentos foram feitos com esses conjuntos de dados para chegar aos experimentos e conclusões aqui presentes.

Palavras-chave: recomendação para grupos, recomendação musical, sistema de recomendação, vetorização

Abstract

The recommendation of items for groups, despite being a topic still little addressed, presents several cases of application such as films for a couple or a family, a travel place for a group of friends or, as will be better discussed in this work, a list of songs for a group of people. This work seeks to compare some techniques already used to recommend items to groups and analyze their results when applied to the musical context. In addition, it tries to contribute in some way with a new approach that can achieve better results than previously developed techniques. The data set was obtained using a crawler that, using Spotify API, collects data about the users who allowed such collection. Several processing was done with these data sets to achieve the experiments and conclusions presented here.

Keywords: recommendation for groups, musical recommendation, recommendation system, vectorization

Sumário

Lista de Figuras

Lista de Tabelas

CAPÍTULO 1

Introdução

1.1 Contexto

Os sistemas de recomendação estão amplamente presentes na vida das pessoas atualmente. Diversas plataformas como Spotify, Netflix e Amazom utilizam esses sistemas para obter melhor engajamento e melhorar suas receitas. Esses sistemas trazem diversos benefícios às empresas que os utilizam e também são bastante úteis para os usuários dos produtos dessas empresas. Como exemplo, podemos citar: entrega de conteúdo relevante, engajamento de usuários, aumento na utilização do serviço (por exemplo: tempo de uso, valor médio de uma compra), entre outros [ben20].

Mostrando alguns números, segundo a própria Netflix, cerca de 75 por cento de tudo que seus usuários assistem vem de uma recomendação. A importância desses sistemas é tanta que a empresa cria concursos com boa premiação para que pessoas consigam melhorar o desempenho do algoritmo da empresa. No caso da Amazom, cerca de 35 por cento das vendas foram geradas por meio de recomendações. Diante desses números, nota-se o quão relevante é ter um bom sistema de recomendação e o quão impactante ele pode ser para uma empresa [Und20].

Apesar de o foco da maioria desses sistema estar voltado para recomendação para um único usuário, ou seja, uma recomendação individual, recentemente houve um aumento na demanda de recomendações para grupos. Essa demanda se deve à percepção de que muitos dos serviços de recomendação para uma pessoas também podem ser utilizados por mais de uma pessoa. Como exemplo podemos citar a recomendação de um restaurante ou de um filme, que podem ser tanto recomendados para uma pessoa como para um grupo de pessoas. Essa nova percepção trouxe consigo a necessidade de novos sistemas e técnicas, pois apesar de estarem ligados, são consideravelmente diferentes. Quando tratamos de grupos, devemos levar em consideração diversos aspectos que não são analisados nas recomendações individuais. Conseguir atingir as necessidades dos membros do grupo e diminuir os conflitos dentro do grupo são apenas alguns dos desafios que sistemas de recomendação para grupos têm de enfrentar [Jam04].

Como citado anteriormente, o contexto musical não difere tanto dos outros contextos. Nele também é possível realizar tanto a recomendação individual quanto a recomendação para um grupo de pessoas. Também estão presentes vários desafios específicos da área musical. No entanto, diversos casos de aplicação podem ser visualizados, tais como realização de festas, músicas para ambientes de trabalho ou até mesmo músicas para agradar clientes presentes em uma loja. Com isso em mente, nota-se a utilidade de se desenvolver ainda mais esses sistemas para que sua utilização se torne realmente eficaz e traga benefícios para quem os utilize.

1.2 Objetivo

O objetivo deste trabalho é analisar métodos e técnicas de recomendação para grupos já existentes, compará-las e tentar criar uma nova abordagem ou melhorar alguma já existente de modo a melhorar o desempenho final das recomendações.

Para isso, inicialmente a proposta é construir uma base de usuários do Spotify de modo a obter informações sobre músicas, playlist e artistas que esses usuários ouvem e também obter características das músicas ouvidas. Vale lembrar que todas as pessoas que se dispuserem a participar do estudo o farão sabendo de todos os detalhes sobre o que será buscado de suas contas no Spotify, podendo a pessoa também se retirar do estudo a qualquer momento. Também é importante lembrar que todos os dados serão buscados utilizando o próprio serviço do Spotify.

Com essa base inicial, pretendo mapear as músicas em vetores, assim como os usuários, que serão baseados nas músicas que ouvem. Dessa forma, poderei analisar semelhanças, seja por uma característica específica ou uma mais genérica, tanto de músicas como de usuários. Após esse processo de transformação em vetores, serão aplicadas técnicas já conhecidas como as citadas no livro Group Recommender Systems, por exemplo [FBST].

Após analisar os resultados obtidos seguindo algumas técnicas de avaliação já conhecidas, tais como as encontradas no livro citado acima, e métricas próprias, tentarei melhorar o desempenho final do sistema modificando ou criando uma nova abordagem com base no que foi utilizado anteriormente.

Por fim, será feita uma avaliação da nova abordagem juntamente com os resultados obtidos das técnicas já existentes, levando em conta a opinião dos usuários que se disponibilizarem a fazer parte dessa avaliação.

Como um adicional, pretendo disponibilizar a ferramenta em sua versão final para uso geral afim de que mais pessoas possam testá-la e consequentemente mais dados possam ser gerados para refinar ainda mais os resultados.

CAPÍTULO 2

Fundamentação

Esse capítulo tem como objetivo abordar alguns conceitos e métodos que serão utilizados ao longo do trabalho e explicá-los ao leitor de forma a tornar mais fácil a compreensão do que será discutido posteriormente.

2.1 Conceitos Básicos

2.1.1 Crawler

Um crawler é um programa de computador que busca informações de maneira automática

2.1.2 Vetorização

Vetorizar é pegar um conjunto de valores e transformar em um objeto que se assemelha a uma linha de uma matriz.

2.1.3 Normalização

Em estatística e aplicações que envolvem estatística, normalizar um conjunto de dados, normalmente na forma de vetor, se refere a ajustar os valores medidos em diferentes escalas para uma escala comum. Técnicas de normalização são muito utilizada para evitar problemas que podem ser causados devido a unidades de medida ou ordem de grandeza dos dados.

2.1.3.1 Desvio padrão

É uma medida que expressa o grau de dispersão de um conjunto de dados. Ou seja, o desvio padrão indica o quanto um conjunto de dados é uniforme.

2.1.3.2 Z-Score

É um tipo de normalização na qual o resultado final mostra quantos desvios padrões acima ou abaixo da media um valor está. Em outras palavras, representa o quão distante da média um valor está. Para calcular o valor do Z-Score se utiliza a formula $\frac{Z-\mu}{\sigma}$ na qual Z é o valor original, μ é a média e σ é o desvio padrão. Ao realizar essa normalização, teremos nosso conjunto de dados distruibuído de maneira semelhante à da figura 2.1 abaixo.

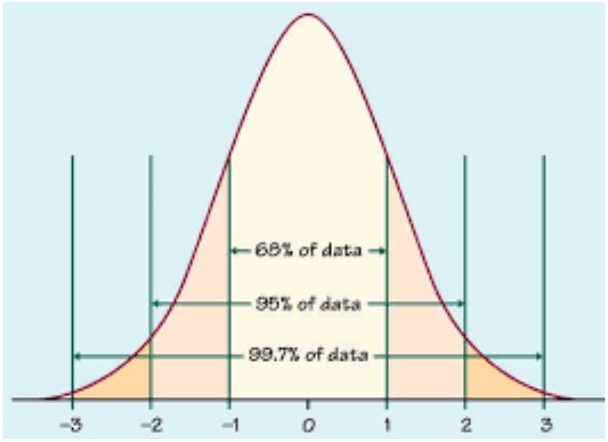

Figura 2.1 Exemplo da distruição dos dados após normalização z-score.

2.1.3.3 Escala logarítmica

É um tipo de normalização na qual o resultado final usa o logaritmo de uma grandeza em vez da grandeza propriamente dita. Se torna útil quando os dados cobrem uma gama muito alta e/ou esparsa de valores.

2.1.4 Similaridade entre vetores

A similaridade entre vetores define o quão parecidos são dois vetores baseado em seus atributos. Existem diversas maneiras de realizar essa medida, tais como usando distância euclidiana, cosseno, entre outros.

2.1.4.1 Similaridade de cosseno

A similaridade de cosseno é uma medida que leva em consideração o ângulo formado entre os vetores, sendo útil para evitar problemas com a grandeza de algum atributo. Como se utiliza do cosseno desse ângulo, os valores de similaridade ficam sempre entre -1 e 1, sendo -1 para vetores que apontam para sentidos contrários e 1 para vetores que apontam para o mesmo lugar.

2.1.5 Clusterização

Clusterização é o processo de separar um grupo em grupos grupos menores de acordo com características em comum apresentadas pelos elementos do grupo.

2.1.5.1 Algoritmo KMeans

O KMeans é um algoritmo de clusterização que avalia e agrupa os dados de acordo com suas características. Resumindo o que é descrito por Mahmoud Parsian em seu livro Data Algorithms[PSMD] , o funcionamento do KMeans ocorre da seguinte forma: primeiro se escolhe um valor para K, que será o número de grupos, depois, deve-se definir um centroide

(representa o centro de cada grupo) para cada cluster. Após isso, para cada valor do conjunto de dados deve-se achar o centroide mais próximo e colocar o valor no grupo daquele centroide, então se recalcula a posição do centroide com base nos valores pertencentes ao grupo daquele centroide. Os últimos dois passos são repetidos até chegar a uma estabilidade.

CAPÍTULO 3

Metodologia

3.1 Coleta de dados

A coleta dos dados foi feita utilizando um crawler que faz uso da API do Spotify para recuperar algumas informações sobre músicas, playlists e artistas que os usuários escutam. Vale lembrar que todos os usuários foram devidamente informados sobre o que seria coletado e estavam cientes dos objetivos do trabalho. O crawler estava sendo executado uma vez ao dia e recuperava as músicas e informações relacionadas que cada usuário ouvira no dia. Na figura abaixo pode-se ver o modelo das informações que o crawler obtinha a cada execução.

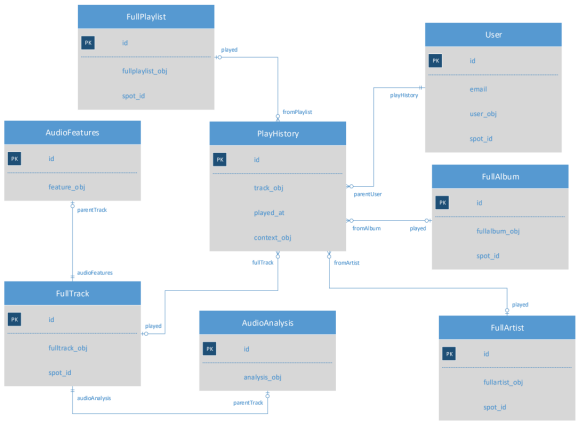

Figura 3.1 Modelo dos dados coletados.

Detalhando um pouco melhor o modelo, PlayHistory corresponde a uma stream e possui informações tais como qual música foi reproduzida, quando foi reproduzida, qual o usuário a escutou, entre outras. Boa parte dessas informações se encontram no FullTrack, que por sua vez possui AudioFeatures, parte extremamente relevante para o estudo aqui presente. As AudioFeatures correspondem aos seguintes atributos: tempo, energy, valence, liveness, loudness, speechiness, acousticness, danceability, instrumentalness. Cada um desses atributos corresponde a uma característica da música sendo analisada.

Para realização dos experimentos foram utilizados os dados de 30 usuários. Uma observação importante é que cerca de metade desses usuários passaram pelo processo diário de coleta durante aproximadamente 6 meses, enquanto a outra metade passou pelo processo durante aproximadamente 2 meses.

Para armazenar esses dados, foi utilizado um banco de dados Postgres e ao fim da coleta,

puderam ser obtidos dados de pouco mais de 55 mil streams, assim como as informações relacionadas a essas streams, tais como artista da música, identificador da música, dia em que foi escutada, entre outros.

3.2 Análise

O objetivo dessa parte é discutir todo o processo de análise e tratamento dos dados obtidos ao fim da coleta. Aqui serão debatidos pontos positivos e negativos do uso de certas abordagens e o motivo da escolha das técnicas. Vale lembrar que todo o desenvolvimento foi feito utilizando Python.

3.2.1 Extração e Pré-processamento

O primeiro passo foi extrair o conjunto de dados do Postgres e converter para documento no formato csv. Após isso, foram sendo construídas tabelas para reunir as informações mais relevantes da coleta de modo a facilitar a manipulação e visualização das informações.

Após juntar informações sobre as músicas, os atributos delas e as streams foram removidas algumas linhas que não possuíam os atributos musicais. Esse problema ocorreu devido à API do Spotify não retornar os valores dos atributos para algumas músicas. No entanto, foram apenas 30 das mais de 55 mil streams que precisaram ser retiradas.

Com a tabela toda sem problemas de dados faltantes, foi aplicada uma normalização utilizando a técnica de Z-Score para deixar os valores dos atributos em uma mesma escala. Agora, com os dados devidamente normalizados e sem valores não preenchidos, como pode-se ver na tabela abaixo, foi possível passar para a próxima etapa.

fulltrack_id	id	instrumentalness	liveness	loudness	playlistPropria	speechiness	tempo	track_id	user_id	valence
7777.0	18297.0	-0.417201	-0.599041	0.668432	0.0	-0.610196	-0.167998	4rPkN1FMzQyFNP9cLUGIIB	8.0	-0.122606
1593.0	18296.0	-0.404329	-0.801679	0.354318	0.0	-0.227610	-0.101414	2dXsILW8gzkosqIeHAwl0v	14.0	-0.996422
2033.0	21887.0	-0.417307	0.041135	0.648893	1.0	1.631322	0.733242	6MC0anRfGPzSWrvybgsY7O	3.0	0.699340
14788.0	36413.0	-0.417307	-0.180658	0.475164	1.0	0.218166	-1.070387	6oW166unB6EiFjZ2KCdgj3	9.0	1.253954
536.0	18298.0	-0.417236	-0.838476	0.530171	0.0	0.953467	1.057672	1DmSaQjZkuwFi57KPv20oB	8.0	-1.299664
7785.0	18299.0	-0.417307	0.711555	0.586027	0.0	-0.487263	0.667493	7vpGKEUPrA4UEsS4o4W1tP	8.0	1.617046
4500.0	21888.0	-0.417307	-0.100006	0.717704	1.0	-0.453944	1.121751	6kR8TFJ4BowMMBW2tokFWr	3.0	1.297844
2266.0	27772.0	-0.417307	-0.155455	0.176342	0.0	-0.043784	-1.063391	7eYEQnY6Lzh3Y8jSOt3xEF	6.0	-1.515126
6362.0	48597.0	-0.417307	3.695680	0.250675	1.0	-0.421775	1.076294	0KkZFQSvZrTL1zU3cX8Z7e	19.0	1.142233
8785.0	18300.0	-0.417307	-0.801679	0.261719	0.0	1.413030	-0.750134	0TwBtDAWpkpM3srywFVOV5	8.0	0.807070

Figura 3.2 Tabela após pré-processamento.

3.2.2 Processamento

Por ser um sistema de recomendação, é de suma importância conseguir achar a melhor forma de relacionar usuários e músicas. Em relacionar, estamos falando de encontrar a melhor forma de dizer quão similares dois elementos são.

3.2.2.1 Vetorização e similaridade

A técnica escolhida para obter essa similaridade foi a de similaridade de cosseno. Essa técnica foi escolhida pois consegue lidar bem com problemas de valores absolutos diferentes,

já que se importa com o ângulo entre os vetores, além de ser bastante rápida de ser executada.

Assim, foram implementadas funções para vetorizar os usuários e as músicas de forma que pudessem ser comparados entre si e pudessem ser relacionados usando a similaridade de cosseno. O processo de vetorização dos usuários consiste em buscar todas as streams de um certo usuário e obter a média dos atributos das músicas dessas streams. Já o processo de vetorização das músicas é mais direto, usando apenas os próprios atributos que são retornados pela API do Spotify.

3.2.2.2 Recomendação

Para realizar a recomendação, foram utilizadas duas abordagens. A primeira delas, desenvolvida com ideias próprias, constrói um vetor único baseado nos vetores de cada usuários do grupo, ou seja, um vetor grupal. A segunda abordagem utiliza as avaliações de cada usuário do grupo para cada música de forma a chegar em uma lista final.

3.2.2.2.1 Recomendação utilizando vetor grupal

Essa abordagem se baseia em tentar construir um único vetor que consiga representar o grupo de usuários e, então, buscar as músicas que tenham maior similaridade com esse vetor.

Para construir esse vetor, algumas informações foram levadas em consideração além do vetor de cada usuário do grupo. A principal delas foi o número de streams que o usuário possuía na base. Essa informação foi julgada como relevante pois em usuários com muitas streams, é possível criar um vetor que o represente de forma mais confiável. No entanto, não estava sendo levado em consideração o número bruto de streams, já que alguns usuários possuíam milhares de streams enquanto outros possuíam poucas dezenas, o que levaria o grupo a ser majoritariamente dominado pelo usuário com milhares de streams e, consequentemente, a uma playlist final que só levaria em consideração o gosto do usuário dominante. Para resolver esse problema da utilização dos dados brutos, foi aplicada uma escala logarítmica, de forma que os valores ficassem muito mais próximos, mesmo quando a diferença original fosse de milhares de streams.

A partir disso, o vetor grupal foi formado aplicando uma média ponderada, onde o peso de cada usuário é o valor do seu numero de streams na escala logarítmica. A figura 3.4 mais abaixo ilustra o processo de escala logarítmica aplicado.

Sendo esse vetor grupal vector1 e sendo lista de músicas tracks, aplicou-se o seguinte pseudocódigo:

Algorithm 1 Algoritmo de geração de playlist

```
1: for track in tracks do
2:     vector2 = criarVetorMusica(track)
3:     similarity = similaridadeCosseno(vector1,vector2)
4:     resultado.inserir(track,similarity)
5: end for
6: return resultado.ordernarPorSimilaridade()
```

O resultado desse algoritmo nos retorna uma tabela com as músicas e sua similaridade com o vetor grupal. Para obter nossa playlist final basta selecionar as primeras X linhas, sendo X o número de músicas desejadas. A tabela abaixo mostra um exemplo de resultado para o processo descrito acima.

track_id	track_similarity	trackName
0AVDN1LiDn0abh39xqDRtQ	0.937181	Rave Jogando o L
1wHNSciCOxqs4dft2EWrIC	0.924851	SOFRENDO FEITO UM LOUCO - SUMMER VERSION
1Q9w2a5dvoIpE8fkP8mUK6	0.924801	C.L.A.T. (feat. DJ Mitch Ferrino)
2nCKTSg3tWkfth4TSYBVzy	0.924555	Ao Som do 150
3tAsQTsf0Z54ug6iCgGsLS	0.924553	Party Band
WBOWPmga0BZF0OKUmky4r	0.922008	Arapuca
20fQRD2UabmuHdfI8JxVoo	0.914018	Rave da Quarentena (feat. MC 2D, MC Vigary & Mc Mr. Bim) - Remix
2sknbace6e6g3fzVcsXh1e	0.913217	Break
2OFO6Zws6Pxo5p0J7SmkOp	0.910549	Empurra Empurra
5579yRF96PYOFoh69ghsRZ	0.909582	Fica à Vontade
1oQppEGTHYN53xfEaTz4ua	0.908714	Vai menina

Figura 3.3 Exemplo de resultado final da geração de playlist usando o vetor grupal.

10	13	12.065753	4287
2	3	11.929998	3902
13	16	11.861862	3722
8	11	11.730046	3397
4	5	11.562242	3024
9	12	11.358102	2625
11	14	11.136350	2251
15	18	10.421013	1371
6	8	10.250298	1218
21	24	10.131857	1122
18	21	9.927778	974
19	22	9.481799	715

Figura 3.4 Tabela de escala logarítmica.

Essa abordagem no entanto apresentava alguns problemas segundo a avaliação de alguns usuários. Os dois problemas mais relevantes apontados foram: playlists com músicas desconhecidas pelos integrantes do grupo e um quase descarte da influência de usuários que possuem gosto musical diferente do restante do grupo.

Para tentar resolver o primeiro dos problemas citados acima, foi desenvolvida uma abordagem de utilizar não só a similaridade da música com o vetor grupal, mas também um atributo que indica quantas pessoas do grupo já ouviram aquela música. Assim sendo, a técnica conseguia aliar músicas relevantes que já eram de conhecimento prévio do grupo. Como apenas músicas relevantes ficam entre as candidatas para a lista final, o principal critério de ranqueamento foi esse novo atributo.

As diferenças nos resultados após a aplicação dessa nova etapa foram significativas e tiveram bons retornos por parte dos usuários.

A tabela a seguir mostra o resultado após a aplicação do novo atributo, indicando na coluna countTrue quantas pessoas do grupo ouviram e ranqueando primeiramente por essa coluna e

depois pela similaridade. Para efeito de comparação, segue na tabela mais abaixo o resultado sem levar em conta esse novo atributo para o mesmo grupo de usuários.

	track_id	track_similarity	trackName	23	25	29	countTrue
11	4xWzZmX4K1yyrdtRfbUvjt	0.904925	Oh Juliana	True	True	True	3
177	7Ac3BmqTQoLd4i7HtZyfgN	0.801245	Te Prometo	True	True	True	3
50	0uaVr4TJNoD7x3lQ0Z1cRP	0.841250	Pede Chamando de Amor	False	True	True	2
62	0AVDN1LiDn0abh39xqDRtQ	0.937181	Rave Jogando o L	False	True	False	1
114	20fQRD2UabmuHdfI8JxVoo	0.914018	Rave da Quarentena (feat. MC 2D, MC Vigary & Mc Mr. Bim) - Remix	False	True	False	1
13	0h1mVA8gOPR71lxSeY67uT	0.902626	Fica à vontade	False	False	True	1
107	6a2ge6gH01iO5vHl6194Al	0.892595	Dança Sensual	False	False	True	1
75	1dKYlw6v9wofuqb7EXSvPn	0.878117	Isso Que é Vida	False	False	True	1

Figura 3.5 Exemplo de tabela com novo atributo.

	track_id	track_similarity	trackName
62	0AVDN1LiDn0abh39xqDRtQ	0.937181	Rave Jogando o L
175	1wHNSciCOxqs4dft2EWrIC	0.924851	SOFRENDO FEITO UM LOUCO - SUMMER VERSION
14	1Q9w2a5dvoIpE8fkP8mUK6	0.924801	C.L.A.T. (feat. DJ Mitch Ferrino)
122	2nCKTSg3tWkfth4TSYBVzy	0.924555	Ao Som do 150
29	3tAsQTsf0Z54ug6iCgGsLS	0.924553	Party Band
163	5WBOWPmga0BZF0OKUmky4r	0.922008	Arapuca
114	20fQRD2UabmuHdfI8JxVoo	0.914018	Rave da Quarentena (feat. MC 2D, MC Vigary & Mc Mr. Bim) - Remix

Figura 3.6 Exemplo de tabela antes de aplicar o novo atributo.

Um outro problema que vale ser mencionado e que está relacionado com o que foi descrito acima é o fato de que, principalmente quando o valor do atributo de contagem era igual a 1 (apenas uma pessoa do grupo havia escutado a música), às vezes o resultado ficava com muito viés para a pessoa mais relevante do grupo, já que em casos de contagem igual a similaridade é o critério de desempate. Para contornar esse problema, usou-se o princípio do algoritmo de fairness descrito na tabela 3.1.

Em resumo, quando há empate na contagem, escolhe-se de forma alternada a música de maior similaridade que o usuário da vez tenha escutado, deixando a lista final mais democrática.

Para resolver o segundo dos problemas citados mais acima, foi necessário realizar uma análise um pouco mais detalhada do problema. O principal ponto aqui seria corrigir a playlist de modo que usuários com gostos musicais diferentes fossem representados na playlist. No entanto, o problema se tornou mais complexo à medida que um grupo maior de pessoas era utilizado para estudo, pois havia casos de o grupo possuir três ou mais gostos musicais bem distintos entre si.

Para resolver essa situação foi pensada em uma abordagem envolvendo clusters.

Primeiramente se media a distância de cada um dos vetores dos integrantes do grupo para o vetor grupal, gerando assim um conjunto de distâncias. Nesse conjunto, media-se o desvio padrão de modo a entender quão dispersos se encontravam os dados. É importante lembrar que as distâncias sempre variam de -1 a 1, implicando que o valor do desvio padrão sempre estará entre 0 e 1.

Com isso, criou-se uma escala a partir desse valor de desvio padrão para definir quantos clusters serão gerados. A escala era subdividida em três partes: valores menores que 0.3 (grupo coeso), menores que 0.6 (grupo razoavelmente incoeso) e menores ou iguais a 1 (grupo bastante incoeso). Como o tamanho máximo dos grupos para os experimentos nesse trabalho é de 9 integrantes, três clusters se mostraram suficientes para conseguir bons resultados, por isso a escala foi dividida em três partes.

Sabendo agora o número de clusters e também já possuindo o vetor de distâncias para o vetor grupal, cada elemento do grupo recebe uma classificação de forma a gerar subgrupos. Essa classificação foi feita utilizando o algoritmo KMeans. Com isso, foi possível dividir o grupo inicial em subgrupos e aplicar as técnicas já previamente descritas em cada um dos subgrupos.

Após aplicar as técnicas acima em cada um dos subgrupos, obtemos várias playlists, uma para cada subgrupo, de modo que aqui se faz necessário realizar uma junção parcial delas para obter a playlist final do grupo. Para saber quantas músicas de cada subgrupo serão selecionadas, utilizou-se mais uma vez a relevância de cada usuário, que é o logaritmo do número de streams. Mais detalhadamente, cada subgrupo teria uma representação de X músicas na playlist final, sendo X o valor arredondado para o inteiro mais próximo dado pela seguinte fórmula $\frac{N * p}{P}$, na qual N é o número de músicas desejado, p é o peso do subgrupo e P é a soma dos pesos de todos os subgrupos. O peso de cada grupo é definido pela seguinte fórmula $\frac{\mu * t}{T}$, na qual μ é a média da relevância dos integrantes do grupo, t é o tamanho do subgrupo e T é o tamanho total do grupo. Assim, a soma do valor de X para cada grupo totaliza N (o número desejado de músicas).

Sabendo agora quantas músicas cada subgrupo terá na playlist final, basta pegar esse número de músicas em ordem da playlist de cada subgrupo e juntá-las em uma playlist única.

21	23	25	27	29	countTrue	track_id
NaN	True	True	NaN	True	3	4xWzZmX4K1yyrdtRfbUvjt
NaN	True	True	NaN	True	3	7Ac3BmqTQoLdAt7HtZyfgN
NaN	False	True	NaN	True	2	0uaVr4TJNoD7x3lQ0Z1cRP
NaN	True	False	NaN	False	1	2wc8HF4fqX4oPBuZnadO8t
NaN	False	True	NaN	False	1	0AVDN1LiDn0abh39xqDRtQ
NaN	False	False	NaN	True	1	0h1mVA8gOPR71lxSeY67uT
NaN	False	True	NaN	False	1	20fQRD2UabmuHdfl8JxVoo
NaN	False	False	NaN	True	1	6a2ge6gH01iO5vHi6194AI
NaN	False	True	NaN	False	1	2dCbTWrHg6cbamZYe2CI2z
NaN	False	False	NaN	True	1	1dKYlw6v9wofuqb7EXSvPn
NaN	False	True	NaN	False	1	4XtSd73tYUBHVAENEdvg6q
NaN	False	False	NaN	True	1	3H7ihDc1dqLriiWXwsc2po
False	NaN	NaN	True	NaN	1	2kJl5AVoftb3FGXZEEHumi
True	NaN	NaN	False	NaN	1	6FlQ8o2hqlDmHQFoBKmKgW
False	NaN	NaN	True	NaN	1	2lHZAm1jHuN74nbguKtORv
True	NaN	NaN	False	NaN	1	6llm1TWzvn3UaYtlyJajln
False	NaN	NaN	True	NaN	1	094zj17xhvUrneycU2teq5
True	NaN	NaN	False	NaN	1	3Ty7OTBNSigGEpeW2PqcsC
False	NaN	NaN	True	NaN	1	7lFuyRfwa42sXopPHUBHIx
True	NaN	NaN	False	NaN	1	6ZFbXlJkul1dVNWvzJzown

Figura 3.7 Exemplo playlist final. Atentar para o countTrue e para a alternância de valores true nas colunas dos usuário. Os subgrupos nesse exemplo são [21,27] e [23,25,29]

3.2.2.2.2 Recomendação utilizando similaridade entre músicas e usuários

Nessa abordagem, utilizaremos uma matriz como fonte primária de informação para executar os algoritmos que serão descritos mais abaixo. Essa matriz é M x N sendo M o número de tracks e N o número de usuários. Nessa matriz, o cruzamento de uma linha com uma coluna corresponde à similaridade entre a música e o usuário.

Ainda sobre essa matriz, foram criadas duas variações dela. A primeira variação apresenta os valores de similaridade em sua forma original, enquanto a segunda variação apresenta os valores de similaridade mapeados para o intervalo de 0 a 10. As tabelas 3.8 e 3.9 abaixo exemplificam essa matriz. A matriz mapeada para valores de 0 a 10 seria algo mais próximo de uma avaliação manual de cada item por um usuário, mas a matriz em sua forma original nos da mais precisão para distinguir e ranquear as músicas de maneira mais eficaz.

track_id	1	2	3	4	5	6	8	9	11	...	22	23	24	25	26	27	28	29	3
2H4PHyrS6GiLHRVST7i38S	10	4	2	3	10	8	2	2	9	...	6	4	3	4	5	6	2	4	
5TzeYf9rUCAtT3o6DloTti	9	5	4	2	8	7	3	3	10	...	4	6	3	6	7	3	1	6	
1jf3wYtynxiPOdhqAp3cUN	10	4	2	4	9	7	2	1	9	...	6	3	4	4	5	7	4	3	
2YUnvS5bWNjZrzoMDRnvYS	10	5	2	3	10	7	2	2	9	...	7	3	3	4	5	7	3	3	
24tanGYdTNn3D3RoCXdr3W	10	5	2	4	10	8	2	1	9	...	7	3	3	4	5	7	3	3	
6hwZy9tvq8fB0bM7yxdZLr	10	5	2	4	10	7	2	1	9	...	7	3	3	3	5	7	4	3	
6ZpEs42RMOO7x9WpXgvlOl	10	4	2	3	10	8	2	1	10	...	6	4	3	4	6	6	3	4	
0nCqnajNbrUqYnud3MzAja	7	6	5	3	6	5	5	5	7	...	4	7	4	5	7	3	4	7	
1ARYYJ8H3LiCLvv8yCAhzc	8	5	4	3	8	7	2	3	9	...	4	6	5	6	8	4	3	6	
6TMoKdGcSIHjLfyzINFB5X	10	5	2	4	10	6	2	2	8	...	8	2	3	3	4	8	5	2	
2sas1gLt9n02iHVKKIXj2s	7	7	6	4	7	5	5	5	7	...	6	5	5	5	4	4	3	6	

Figura 3.8 Matriz com mapeamento da similariade.

track_id	1	2	3	4	5	6	8	9	11	...	22	23	
2H4PHyrS6GiLHRVST7i38S	0.921898	-0.217918	-0.651676	-0.549736	0.855250	0.447690	-0.702064	-0.783408	0.788962	...	0.145304	-0.386095	-
5TzeYf9rUCAtT3o6DloTti	0.613057	-0.124615	-0.212555	-0.625046	0.504823	0.378597	-0.476009	-0.467989	0.800476	...	-0.265344	0.142989	-
1jf3wYtynxiPOdhqAp3cUN	0.807510	-0.231872	-0.784392	-0.329784	0.748744	0.363093	-0.738965	-0.844700	0.616599	...	0.104102	-0.507692	-
2YUnvS5bWNjZrzoMDRnvYS	0.926573	-0.060133	-0.615005	-0.452105	0.859481	0.331445	-0.753839	-0.774387	0.761957	...	0.355129	-0.476490	-
24tanGYdTNn3D3RoCXdr3W	0.939601	-0.179149	-0.723038	-0.393728	0.860393	0.411655	-0.759222	-0.832438	0.768812	...	0.243783	-0.532930	-
6hwZy9tvq8fB0bM7yxdZLr	0.911643	-0.111359	-0.737750	-0.373622	0.833590	0.298341	-0.769577	-0.850184	0.723644	...	0.267191	-0.528221	-
6ZpEs42RMOO7x9WpXgvlOl	0.903565	-0.223394	-0.674598	-0.540718	0.820346	0.435670	-0.769939	-0.830161	0.804906	...	0.062940	-0.334243	-
0nCqnajNbrUqYnud3MzAja	0.248832	0.003110	-0.045646	-0.485940	0.138412	-0.148161	-0.101145	-0.194124	0.368010	...	-0.288642	0.390259	-
1ARYYJ8H3LiCLvv8yCAhzc	0.545847	-0.165828	-0.356827	-0.492498	0.442830	0.339709	-0.622867	-0.597317	0.670130	...	-0.377218	0.104940	-
6TMoKdGcSIHjLfyzINFB5X	0.897401	-0.016758	-0.684979	-0.341887	0.854525	0.162217	-0.694596	-0.762800	0.571091	...	0.555326	-0.613573	-

Figura 3.9 Matriz sem mapeamento da similariade.

Os algoritmos que serão discutidos agora, usam essa matriz citada acima e operam sobre ela para encontrar as músicas com melhor avaliação segundo o critério do algoritmo. Os algoritmos utilizados já são consolidados e utilizados em outros casos de aplicações. [Ric11] Os algoritmos escolhidos foram: Average, Multiplicative, Approval Voting, Least Misery, Most Pleasure, Average without Misery e Fairness.

De forma semelhante à abordagem anterior, essa também usa esses algoritmos aplicados à matriz descrita acima para obter a lista de músicas com melhor avaliação.

A figura abaixo mostra um exemplo e tabela a seguir descreve brevemente o funcionamento dos algoritmos citados acima:

	A	B	C	D	E	F	G	H	I	J
Peter	10	4	3	6	10	9	6	8	10	8
Jane	1	9	8	9	7	9	6	9	3	8
Mary	10	5	2	7	9	8	5	6	7	6

Figura 3.10 Exemplo de matriz com avaliações individuais.

	Como funciona	Exemplo
Average	Média das avaliações individuais	O item B terá uma avaliação final de 6 (4+9+5)/3
Multiplicative	Multiplicação das avaliações individuais	O item B terá uma avaliação final de 180 (4*9*5)
Approval Voting	Número de avaliações acima de um determinado limiar	Para um limiar de 6, o item B terá uma avaliação final de 1 e o item F terá uma avaliação final de 3
Least Misery	Menor valor dentre as avaliações individuais	O item B terá uma avaliação final de 4 (o menor entre 4,9,5)
Most Pleasure	Maior valor dentre as avaliações individuais	O item B terá uma avaliação final de 9 (o maior entre 4,9,5)
Average without Misery	Média das avaliações individuais após excluir itens com avaliações menores que um certo limiar	Para um limiar de 4, o item J terá uma avaliação final de 7.3 (média entre 8,8 e 6). Já o item A será excluído, pois possui uma avaliação com valor 1.
Fairness	Seleciona o item melhor ranqueado para cada usuario individualmente em turnos	Os itens escolhidos poderiam ser E.F e A (mais bem avaliados por Peter, Mary e Jane, respectivamente)

Tabela 3.1 Funcionamento dos algoritmos

3.3 Testes

O objetivo dessa parte é explicitar como foram desenvolvidos os testes para cada um dos algoritmos citados acima de forma a deixar clara a metodologia de testagem para que os resultados possam ser interpretados de maneira correta.

A testagem ocorreu gerando grupos de tamanhos 3,5,7 e 9 pessoas dentre um conjunto de 29 usuários. Os grupos foram gerados de maneira aleatória de modo a tentar gerar grupos variados. Cada um dos grupos gerados foi utilizado para gerar playlists segundo as metodologias descritas acima, tendo cada grupo gerado oito playlists, uma usando a abordagem que utiliza o vetor grupal e sete usando as abordagens que utilizam a matriz de similaridade.

Após a geração das playlists, os usuários que estavam presentes nelas retornaram um feedback com uma nota de 0 a 10 representando o quão satisfeitos eles estavam com aquela playlist. Além disso, também houve uma conversa com os usuários para entender em que pontos a playlist estava falhando para que se pudesse tentar corrigir as falhas alterando o algoritmo.

CAPÍTULO 4

Resultados

Nesse capítulo será discutido o resultado dos testes feitos a fim de comparar cada uma das possibilidades elencadas durante o capítulo de metodologia. Temos como objetivo avaliar o quão bem foram os algoritmos utilizados na geração das playlists e identificar os pontos nos quais eles foram efetivos e onde deixaram a desejar.

4.1 Análise dos resultados

Aqui iremos analisar algumas das playlists geradas e o retorno dado pelos usuários. Será analisado também o contexto no qual aquela playlist foi gerada, passando pela similaridade do grupo, pelo tamanho do grupo, entre outros.

Nesta análise, iremos usar de exemplos de grupos gerados como descrito no capítulo anterior usando um subconjunto de usuários que se dispuseram a dar um retorno sobre as playlists. Dessa forma, foram gerados os seguintes grupos:

	Usuários	Desvio padrão
1	27, 28, 31	0.117
2	24, 26, 28	0.228
3	21, 26, 28	0.414
4	20, 21, 22, 28, 31	0.405
5	22, 23, 24, 29, 31	0.478
6	22, 26, 27, 29, 31	0.308
7	21, 22, 23, 24, 26, 27, 31	0.364
8	21, 23, 24, 25, 26, 28, 29	0.395
9	20, 21, 23, 24, 25, 28, 31	0.404
10	20, 21, 22, 24, 25, 26, 27, 29, 31	0.303
11	20, 21, 22, 24, 25, 26, 27, 28, 31	0.397
12	20, 21, 23, 24, 25, 26, 27, 28, 29	0.328

Tabela 4.1 Grupos para análise

Usando o algoritmo próprio e os pré-existentes, foram geradas playlists para cada um dos grupos da tabela acima. Abaixo temos duas tabelas, sendo a primeira delas a avaliação da playlist gerada com algoritmo próprio por cada usuário. Na segunda tabela, a primeira coluna representa o número do grupo e as demais colunas a média do grupo para o determinado algoritmo.

	1	2	3	4	5	6	7	8	9	10	11	12
20	X	X	X	X	X	10	X	X	7	9	8	6
21	X	X	8	X	X	9	6	7	6	5	7	7
22	X	X	X	5	7	7	6	X	X	5	6	X
23	X	X	X	X	8	X	6	5	7	X	X	8
24	X	7	X	X	8	X	6	7	7	8	7	9
25	X	X	X	X	X	X	X	8	8	7	X	9
26	X	7	6	8	X	X	8	8	X	7	10	9
27	10	X	X	6	X	X	8	X	X	8	8	6
28	10	10	9	X	X	8	X	5	6	X	8	6
29	X	X	X	8	8	X	X	7	X	6	9	8
31	8	X	X	6	4	9	5	X	7	6	6	8

Tabela 4.2 Avaliações das playlists pelos usuários

	Avg	LeastMisery	Approval	MostPleasure	AvgNoMisery	Fairness
1	8	6	6	9	7	8
2	7	7	6	7	7	6
3	5	5	3	5	4	6
4	5	6	3	5	4	5
5	4	8	4	4	3	6
6	7	6	3	5	4	6
7	5	6	5	6	5	5
8	6	6	5	7	6	6
9	5	5	6	3	6	6
10	7	4	6	6	5	4
11	4	5	6	4	4	5
12	4	4	5	4	5	5

Tabela 4.3 Média das avaliações das playlists pelos grupos

Analisando os resultados foi possível concluir que o algoritmo desenvolvido nesse trabalho apresentou um desempenho muito bom, já que a maioria dos usuário avaliou a playlist com notas maiores ou iguais a 7. Alguns casos apresentaram notas abaixo de 7, o que também era previsível dada a complexidade do problema.

Em relação aos algoritmos pré-existentes, dois problemas foram os principais relatados. O primeiro deles foi a falta de representatividade na playlist quando o grupo apresentava algum integrante com gosto musical diferente dos demais. O segundo foi a presença de muitas músicas desconhecidas pelos integrantes do grupo. Aqui vale ser feita uma observação que, embora o objetivo seja recomendar uma playlist não necessariamente de músicas conhecidas para o grupo, os usuários se mostraram mais favoráveis a receber uma playlist de músicas conhecidas quando se trata de uma situação de grupo de pessoas, principalmente nos casos de aplicação abordados no início do trabalho.

Esses dois problemas ainda foram agravados pela presença de músicas que não foram muito aprovadas pelos usuários, como pode ser observado pela grande presença de notas 5 e 6. E em alguns casos, algoritmos que não conseguiam encontrar músicas suficientes para uma playlist, ou tinham que ser nivelados muito pot baixo (por exemplo o Approval voting). Assim, os algoritmos pré-existentes, embora apresentem bastante velocidade de execução e ideias que podem ser aproveitadas, não conseguiram atingir o objetivo de gerar uma playlist que agrade o grupo.

CAPÍTULO 5

Conclusão

Durante o estudo e com a ajuda dos usuários, os resultados obtidos de forma objetiva (nota dos usuários) e também de uma conversa mais subjetiva com os mesmos para tentar descobrir onde as playlists estavam falhando, a metodologia foi avançando e melhorando cada vez mais para tentar suprir as lacunas antes não observadas. Dessa forma, o algoritmo desenvolvido para esse estudo específico se mostrou muito superior aos pré-existentes, ainda que apresente alguns resultados ruins para alguns usuários.

A principal dificuldade encontrada foi conseguir criar uma playlist que agrade usuários muito distintos entre si de modo a deixar todos satisfeitos. Um outro ponto a ser levado em consideração é que o número de integrantes no grupo não se mostrou um fator tão decisivo quanto a coesão entre os integrantes, visto que grupo maiores, porém coesos, apresentaram resultados melhores do que grupos menores e incoesos.

Para grupos coesos, os resultados obtidos foram excelentes, como pode ser observado no capítulo de resultados. Já os resultados para grupo incoesos foram muito variados, pois alguns usuários (os mais ecléticos) ficaram satisfeitos, enquanto outros usuários não tão ecléticos acabaram não ficando tão satisfeitos. Quando questionados sobre uma avaliação para a playlist levando em consideração o fator de que a playlist seria para um grupo, os resultados tenderam a ficarem melhores, principalmente os que na avaliação individual se mostraram ruins. Isso me levou a concluir que o fato de estar em grupo também faz com que as pessoas acabem relevando um pouco suas preferências individuais.

Um outro ponto relatado pelos usuários durante a coleta dos resultados e que se mostrou bastante frequente foi que algumas playlists se mostraram muito ecléticas, ou seja, havia nelas umas mistura grande de gêneros musicais. Para alguns usuários, isso foi um ponto positivo, dado que a playlist deveria tentar agradar um grupo de pessoas, já para outros foi um ponto negativo, principalmente quando a playlist incluía gêneros que o usuário não gostava.

Como já foi citado na parte de análise de resultados, os algoritmos pré-existentes apresentaram diversos problemas, principalmente para grupos maiores. Esses problemas envolviam desde playlists com músicas ruins de acordo com a avaliação dos usuários até problemas em conseguir qualquer música que atendesse ao critério do algoritmo. De modo geral, para grupo pequenos e coesos, conseguiram um desempenho que pode ser considerado bom, mas dada a restrição de casos de aplicação, se mostraram um tanto quanto ineficientes no caso geral.

De modo geral, os experimentos mostraram a complexidade do problema de gerar uma lista de músicas para um grupo de pessoas e me permitiu entender muito mais sobre o tema, me revelando detalhes que eu não havia pensado previamente e que são de suma importância

para o usuário final.

Por fim, acredito que ainda há muitos pontos de melhora na área de recomendação para grupos como um todo e muitos outros detalhes e ideias que devem ser debatidas quando se aplica essa recomendação à área musical.

Referências Bibliográficas

[ben20] Benefits of a recommendation engine, 2020. https://www.certona.com/article/benefit-of-recommendation-engines/ .Acessado por último em 02 de Setembro 2020.

[FBST] Alexander Felfernig, Ludovico Boratto, Martin Stettinger, and Marko Tkalčič. *Group recommender systems*.

[Jam04] Anthony Jameson. More than the sum of its members. *Proceedings of the working conference on Advanced visual interfaces - AVI '04*, 2004.

[PSMD] Mahmoud Parsian, Ann Spencer, Judy McConville, and Rebecca Demarest. *Data algorithms*.

[Ric11] Francesco Ricci. *Recommender systems handbook*. Springer, 2011.

[Und20] Corinna Underwood. Use cases of recommendation systems in business - current applications and methods | emerj, 2020. https://emerj.com/ai-sector-overviews/use-cases-recommendation-systems/ .Acessado por último em 02 de Setembro 2020.