SISTEMA DE RECOMENDAÇÃO DE MÚSICAS USANDO LDA E ATRIBUTOS DE ÁUDIO

Pedro Henrique Ribeiro Jordão

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia de Sistemas e Computação.

Orientadores: Gerson Zaverucha
Luiz Wagner Pereira Biscainho

Rio de Janeiro
Março de 2016

SISTEMA DE RECOMENDAÇÃO DE MÚSICAS USANDO LDA E ATRIBUTOS DE ÁUDIO

Pedro Henrique Ribeiro Jordão

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Examinada por:

Prof. Gerson Zaverucha, Ph.D.

Prof. Luiz Wagner Pereira Biscainho, D.Sc.

Prof. José Gabriel Rodriguez Carneiro Gomes, Ph.D.

Prof. Marley Maria Bernardes Rebuzzi Vallesco, Ph.D.

RIO DE JANEIRO, RJ – BRASIL
MARÇO DE 2016

Jordão, Pedro Henrique Ribeiro

Sistema de recomendação de músicas usando LDA e atributos de áudio/Pedro Henrique Ribeiro Jordão. – Rio de Janeiro: UFRJ/COPPE, 2016.

XI, 71 p.: il.; 29, 7cm.

Orientadores: Gerson Zaverucha

Luiz Wagner Pereira Biscainho

Dissertação (mestrado) – UFRJ/COPPE/Programa de Engenharia de Sistemas e Computação, 2016.

Referências Bibliográficas: p. 64 – 68.

1. Recomendação. 2. LDA. 3. Música. 4. Sistemas inteligentes. 5. Processamento de texto. 6. Processamento de Sinais. 7. Geração de tags. I. Zaverucha, Gerson *et al.* II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

Aos meus professores que me
guiaram nesse caminho

Agradecimentos

Agradeço à minha familia e aos amigos que me acompanharam em cada passo desse processo, em especial à Lilian, com quem compartilhei os meus melhores momentos. A Leandro Justino, Amanda Portela, Luiz Guilherme, Luiza Sarmento, Natalia Avila, Guilherme Guimarâes, Érico Henrique, Thabata Maciel, Paulo Guerra, Daniel Tenan, Ingrid, Gabriela Ribeiro, Lucas Guzzo, e Gustavo Lessa, obrigado pelos grandes momentos. Aos amigos do IPqM Raáma Costa, Fernando Leone, Paulo Bruno, Felipe Lázaro, Rubens Pailo, Margareth Braz e Rodrigo Ferreira, agradeço o apoio diário. Também agradeço aos meus orientadores, Gerson Zaverucha e Luiz Wagner Biscainho, pelas orientações e dedicação demonstradas.

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

SISTEMA DE RECOMENDAÇÃO DE MÚSICAS USANDO LDA E ATRIBUTOS DE ÁUDIO

Pedro Henrique Ribeiro Jordão

Março/2016

Orientadores: Gerson Zaverucha
Luiz Wagner Pereira Biscainho

Programa: Engenharia de Sistemas e Computação

Com a popularização de serviços de música pela internet, como *Spotify*, *Last.fm*, e *Tidal*, faz-se necessário um sistema que possa recomendar músicas com base no gosto do usuário. Diversos métodos para recomendação existem na literatura, como sistemas baseados em *tags*, sistemas baseados em análise de sinal e sistemas de contribuição social. Este trabalho apresenta um novo sistema de recomendação de músicas baseado em *tags* que utiliza o modelo LDA como ferramenta de redução de dimensionalidade mantendo uma alta capacidade de representação. O sistema proposto é capaz de gerar *playlists* aleatórias e determinísticas, utilizando apenas medidas de distância comuns, com métodos de baixa complexidade.

A avaliação da qualidade dos sistemas é feita de forma indireta, uma vez que testes do tipo *A/B* são custosos e demorados para avaliação de sistemas de recomendação. Então utilizamos uma medida da homogeneidade das *playlists* geradas pelo sistema através da classificação de gêneros.

Também são propostos novos métodos para tratar músicas que não possuem *tags*, permitindo aproveitamento máximo de músicas recêm-inseridas na base de dados e reduzindo os problemas de *cold-start* e esparsidade, comuns em outros sistemas de recomendação.

Os métodos de tratamento de músicas sem tags ainda precisam de trabalho adicional para atingirem resultados satisfatórios. Já os resultados indicados pelos escores de homogeneidade para os métodos de recomendação de música propostos são muito bons. A medida de qualidade proposta permitiu lançar uma nova luz no estudo de como são as *playlists* resultantes do sistema de recomendação.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

MUSIC RECOMMENDATION SYSTEM USING LDA AND AUDIO FEATURES

Pedro Henrique Ribeiro Jordão

March/2016

Advisors: Gerson Zaverucha
Luiz Wagner Pereira Biscainho

Department: Systems Engineering and Computer Science

With the popularitzation of music services on the internet, such as *Spotify*, *Last.fm*, and *Tidal*, systems which are able to recommmend songs based on user tastes have become necessary. Many such systems can be found in the literature, like *tag* based systems, *signal* based systems and social contribution systems. This paper presents a new *tag* based system that uses an LDA model as a dimensionality reduction tool, while mantaining quality of representation. The system is able to generate both random and deterministic *playlists* using simple distance metrics, with low complexity.

The recommender quality is indirectly assessed, since *A/B* tests are costly and time consuming to evaluate such systems. For that reason we adopt a score based on the homogeneity of the *playlists* created by the system related to genre classification.

We also propose new methods to deal with songs without *tags*, so that we can take advantage of the newly added songs in the dataset and deal with the sparsity and *cold-start* problems, inherent to with other recommendation systems.

While the treatment of songs without tags requires further investigation to achieve acceptable results, the results indicated for the generated playlists by the homogeneity scores were very good. The proposed quality measure was able to shed a new light on the observation of resulting *playlists*.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

1.1 O problema de recomendação

Com a expansão de tecnologias digitais e a popularização da internet nas últimas décadas, a distribuição de mídia mudou radicalmente. Passados são os dias em que a única alternativa para o acesso a músicas era a coleção pessoal e as únicas maneiras de conhecer novas músicas eram recomendações de conhecidos, a televisão e o rádio. No antigo formato de distribuição, os usuários se viam à mercê da programação de um canal ou estação de rádio, com pouca ou nenhuma ideia do que veriam ou ouviriam em seguida. No novo meio interativo possibilitado pela internet, surgiram serviços como Spotfy, iTunes, Youtube, Rdio e Pandora, entre outros, que facilitam o acesso imediato a coleções digitais contendo um vasto catálogo de programação. Com essa sobrecarga de conteúdo prontamente acessível, usuários agora se veem com um novo problema: o que consumir em seguida? Paradoxalmente, a facilidade de acesso a novos conteúdos dificultou a *descoberta* de novas músicas [1]. Essa dificuldade é a motivação para a criação de sistemas de recomendação para música.

Os sistemas de recomendação utilizados na indústria e sendo pesquisados academicamente no momento podem ser divididos em três grandes tipos: baseados em anotações semânticas e metadados, baseados em conteúdo e híbridos [2] [3]. O primeiro depende de pessoas para anotar cada item com palavras que descrevam bem aquela música com informações relevantes (metadados explícitos), ou informações como co-ocorrência de compra e número de vezes que uma faixa foi escutada ou pulada (metadados implícitos). Sistemas de metadados podem ser considerados colaborativos, uma vez que é impossível que uma única pessoa crie anotações para todas as músicas em grande escala. Essas anotações podem ser utilizadas para indexação de músicas, busca e recomendações. Métodos baseados em conteúdo tentam criar recomendações a partir do sinal de uma faixa de música diretamente. Esses sistemas estimam similaridades matemáticas entre itens, comparando atributos extraídos do

sinal digital de cada objeto. Já métodos híbridos buscam criar uma combinação entre métodos baseados em metadados e baseados em conteúdo, visando a compensar as deficiências que cada método tem quando utilizado sozinho.

Note que enquanto sistemas que utilizam anotações semânticas se mostram muito precisos, também são altamente custosos, já que para criar anotações relevantes dependemos de profissionais treinados ouvindo cada música da coleção. Uma alternativa é aproveitar as facilidades colaborativas da internet e permitir que usuários criem suas próprias anotações, gerando dessa forma uma base de dados a partir de diversas fontes. É verdade que a precisão desse sistema é reduzida, mas esse é o custo da criação das anotações. Sites como *Last.fm* utilizam essa alternativa, permitindo que seus clientes criem anotações (*tags*) para as músicas em sua coleção, além de disponibilizar uma API (*Application Program Interface*) para o acesso a essas *tags* por interessados.

Um bom sistema de recomendação deve ser capaz de prever o que o usuário irá ouvir em seguida e gerar *playlists* automaticamente. Porém, música é subjetiva, e os gostos e preferências variam de pessoa para pessoa, sendo importante entender como pessoas classificam músicas antes de criar um sistema de recomendação. A maneira mais frequente de se classificar o gosto musical de uma pessoa é através de gêneros musicais. Logo, não é de espantar que a maioria das *tags* observadas em sites como o *Last.fm* tentam classificar músicas com informações de gênero. Por outro lado, assim como gosto musical é subjetivo, a maneira como uma pessoa percebe uma música é subjetiva; por isso, temos usuários que avaliam o mesmo item com gêneros similares, mas não idênticos, ou até mesmo avaliações que parecem contraditórias.

O objetivo desse trabalho é desenvolver um sistema de recomendação que concilie as maneiras distintas com que diversos usuários classificam uma música, i.e. que se mostre robusto à subjetividade da avaliação humana, além de ser o mais simples possível do ponto de vista computacional. Pretendemos também gerar recomendações que sejam facilmente explicáveis a partir dos dados, de forma que usuários possam entender o porquê delas.

1.2 Recomendação de músicas por similaridade entre gêneros

O problema da recomendação pode ser visto como a definição de uma medida de similaridade entre itens (sejam eles os próprios usuários, utilizando dados de co-ocorrência, ou músicas, utilizando metadados de anotações semânticas e informações de conteúdo). O desafio surge quando tentamos definir o que são músicas similares. Uma opção imediata é utilizar atributos do sinal de uma música e concluir que, a

partir de alguma medida de distância, se os vetores de atributos de duas músicas são próximos, então elas são similares. Esse método é conhecido como recomendação baseada em conteúdo. Como será visto mais à frente, métodos baseados puramente em conteúdo não parecem ser uma boa solução para o problema de recomendação, uma vez que informações contextuais e culturais influenciam a qualidade de uma recomendação. Logo, parece desejável que um bom método de recomendação leve em consideração a maneira como pessoas entendem músicas similares.

Como já foi discutido, a maneira como seres humanos avaliam similaridade em um contexto musical se dá principalmente através da classificação de gêneros. À medida que novas vertentes culturais, instrumentos e tecnologias surgem, também novos gêneros surgem. À medida que estes amadurecem e novos artistas criam novas composições, subgêneros também começam a surgir. Um exemplo canônico desse fenômeno pode ser encontrado no desenvolvimento da música eletrônica, um gênero que surgiu recentemente com a criação de sintetizadores e popularizado pelo uso de computadores para composição. Com a rápida popularização da música eletrônica, novos subgêneros se formaram, além da mistura da música eletrônica com outros estilos.

Com essas informações em mente, acreditamos que uma base de dados que inclua informações de como as pessoas classificam músicas por gênero parece uma boa solução para o problema de recomendação. Esse tipo de base de dados pode ser encontrado em bancos de *tags*, onde pessoas criam anotações semânticas sobre como definem determinada música, banda ou álbum. Essas informações incluem gênero, sentimentos, informações de ano de composição e nome do artista, entre outros.

De posse dessa base de dados, podemos agora definir uma função de distância a partir de *tags*. Algumas soluções parecem surgir imediatamente, como recomendar músicas com a mesma *tag* ou músicas que possuam a maior quantidade de *tags* em comum. Se por um lado esse tipo de algoritmo possui grande poder de explicação, por outro lado ele apresenta um grande problema de desempenho: uma vez que novas nomeações para gêneros são criadas constantemente, a dimensionalidade de uma base de *tags* tende a crescer rapidamente, além de gerar recomendações pouco flexíveis. Assim, seria interessante se pudéssemos encontrar uma solução que diminuísse a dimensionalidade da base de *tags*, que agilizasse a comparação entre itens da base de dados e, se possível, não perdesse o poder de explicação. Com esse objetivo, nesse trabalho aplicamos o modelo LDA (Latent Dirichlet Allocation) [4], uma técnica de modelagem para itens discretos (nesse caso o texto das *tags*), para transformar as anotações de cada música em um vetor de distribuição sobre tópicos de dimensão predeterminada.

A escolha do modelo LDA como forma de redução de dimensionalidade se mostra uma solução interessante porque, além de cumprir a sua função principal ao simpli-

ficar a comparação entre itens, ainda mantém o seu poder de explicação. Em vez de cada música ter uma série de anotações semânticas que definem como pessoas a avaliam (com um número não definido de anotações para cada música), agora os itens são definidos como um vetor de distribuição com um tamanho predefinido sobre uma série de tópicos. Cada tópico agrega *tags* que o modelo vê como similares, assim formando o que podemos considerar *supergêneros*. O vetor de distribuição que representa cada item pode ser interpretado como a mistura de *supergêneros* que define aquela música. Logo, músicas similares são aquelas que possuem *misturas de supergêneros* similares. Essa similaridade pode ser medida a partir de distâncias vetoriais já conhecidas (como euclidiana ou cosseno).

1.3 Trabalhos similares

Sistemas de recomendação são um tópico de pesquisa de grande interesse na literatura de recuperação de informação e inteligência artificial. Métodos específicos para o campo musical são apresentados em diversas publicações. A tese em [5] faz uma grande revisão sobre métodos de recomendação de música existentes utilizando informações de conteúdo, como análise de sinal, além de uma discussão sobre métodos de avaliação de qualidade de recomendação. As técnicas de recomendação envolvem similaridade entre sinais de áudio e podem ser utilizadas não apenas para gerar recomendações como para a geração de *tags* automaticamente.

Para sistemas de recomendação baseados em *tagging* social, [6] apresenta uma revisão de metodologias propostas para lidar com as diversas maneiras com que *tags* podem ser utilizadas para gerar recomendações a partir de um perfil de *tags* para o usuário.

Em [7] são apresentados métodos de avaliação de *playlists* e a utilização de tópicos para avaliar similaridade a partir de *tags* utilizando o modelo LDA. O trabalho considera *playlists* inteiras como itens na base de dados, nunca vizando gerar recomendações de música. Nesse trabalho também é apresentada uma nova forma de medir a qualidade das recomendações geradas utilizando informações disponibilizadas por serviços como *yes.com* e *Radio Paradise*.

A base de músicas utilizada é apresentada em [8]. Essa base contém um agregado de 1 milhão de músicas, com análises de *features* de áudio executadas pelo serviço *The Echo Nest*, além de metadados para cada faixa. Essa base é complementada por *tags* retiradas do serviço *Last.fm*, que agrega anotações disponibilizadas por usuários. Quando necessário, novas *tags* foram recuperadas utilizando-se a API disponibilizada pelo serviço.

1.4 Contribuições

Esse trabalho propõe a possibilidade de utilização do modelo LDA para redução de dimensionalidade de bancos de *tags* com o intuito de gerar recomendações. Em uma investigação similar à apresentada em [7], o modelo é utilizado para transformar uma nuvem de *tags* em um objeto numérico (o vetor de tópicos) que possa ser manipulado matematicamente para a geração de recomendações que possuam explicações intuitivas.

O trabalho citado busca encontrar uma forma de comparar *playlists* já existentes entre si, de forma a dizer se *playlists* distintas são similares. Diferentemente, neste trabalho propomos diversas formas de geração de recomendações. Primeiramente, em 4.4 investigamos a modelagem de músicas a partir de suas *tags*, com o objetivo de fazer recomendações a partir de uma *query*, de forma que músicas similares à utilizada para pesquisa sejam apresentadas para o usuário. Em seguida, em 4.5 investigamos a modelagem de *playlists* objetivando a seleção automática de músicas a serem adicionadas em uma *playlist* com base em suas distribuições de palavras sobre os tópicos. Por fim, em 4.6 modelamos usuários utilizando as *tags* de suas músicas ouvidas em um perfil que será utilizado para modelagem e permitirá a recomendação de músicas para esse usuário. Em 4.7.3 também fazemos uma exploração das diferentes funções de distância que podem ser utilizadas para gerar recomendações.

Na Seção 4.7.4 são discutidas formas de se avaliar um sistema de recomendação que gera *playlists* sem a utilização de testes custoso. Essa avaliação busca estudar a homogeneidade de gêneros nas músicas da *playlist* gerada para uma *query* qualquer.

Além disso, também propomos formas de lidar com músicas sem *tags* na base de dados utilizando *features* de áudio das músicas da base de dados na Seção 5.

1.5 Organização do trabalho

O Capítulo 2 apresenta uma revisão dos componentes de um sistema de recomendação, além de apresentar conceitos relacionados ao problema. Também são discutidos os tipos de sistemas de recomendação existentes na literatura e quais são as suas deficências e vantagens.

O Capítulo 3 discute o modelo de modelagem de textos *Latent Dirichlet Allocation* (LDA) proposto em [4] como uma forma de modelagem probabilística para geração de textos. Também apresentamos um exemplo de aplicação do modelo em um *corpus* de textos como forma de demonstrar a capacidade de representação que o modelo LDA é capaz de alcançar.

No Capítulo 4 propomos como a utilização do modelo LDA pode ser feita para

um banco de dados de *tags* de músicas, analisamos o modelo obtido e apresentamos formas diversas de gerar *playlists* para usuários utilizando o sistema. Concluimos o capítulo com a realização de experimentos, onde utilizamos métodos indiretos para inferir a qualidade das recomendações geradas sem a utilização de testes A/B, que são custosos e demorados.

Em seguida, no Capítulo 5, discutimos formas de evitar problemas de esparsidade do banco de dados e como aproveitar músicas recém inseridas na base de dados que ainda não possuem anotações, um problema conhecido como *cold-start*.

Finalmente, no Capítulo 6 concluimos o trabalho através de uma avaliação geral dos resultados obtidos no decorrer do desenvolvimento, e apresentamos potenciais melhorias que o sistema pode receber.

Capítulo 2

Componentes em um sistema de recomendação

Em geral, sistemas de recomendação envolvem 3 componentes: usuários, itens e um algoritmo de recomendação. Neste capítulo, iremos explorar como se podem modelar itens e usuários (Seções 2.1 e 2.2, respectivamente). Em seguida faremos uma análise de problemas comumente encontrados em sistemas de recomendação (Seção 2.3). Concluiremos com uma discussão sobre formas de avaliar a qualidade de recomendações (Seção 2.4).

2.1 Modelagem de itens

Neste trabalho, o principal componente do sistema de recomendação é o item a ser recomendado: a música. Um cenário de recomendação [9] [3] pode ser descrito como a predição de um subconjunto I_{u_a} do total de itens $I = \{i_1, ..., i_m\}$ que são considerados *interessantes* para um usuário específico u_a, pertencente ao conjunto de usuários $U = \{u_1, ..., u_n\}$. Além disso, também possuímos alguma informação sobre as preferências do usuário, como avaliação de itens, histórico de compras ou simplesmente a música utilizada para *query* no sistema. Nesse trabalho iremos focar no cenário em que o usuário busca recomendações baseadas em um item específico, que será utilizado como *seed* para a recomendação. Naturalmente, o conjunto I_{u_a} deve ter um número de elementos muito menor do que o total de itens, $|I|$. Por questões práticas, recomendações são restritas a K recomendações (chamadas de *top-K* recomendações).

Existem três principais vertentes para recomendações musicais: sistemas baseados em metadados, sistemas baseados em conteúdo e sistemas híbridos. Sistemas baseados em metadados, por sua vez, podem ser subdivididos em 4 categorias, de acordo com o tipo de informação que utilizam. A Figura 2.1 é uma visualização de

como esses sistemas podem ser classificados [5].

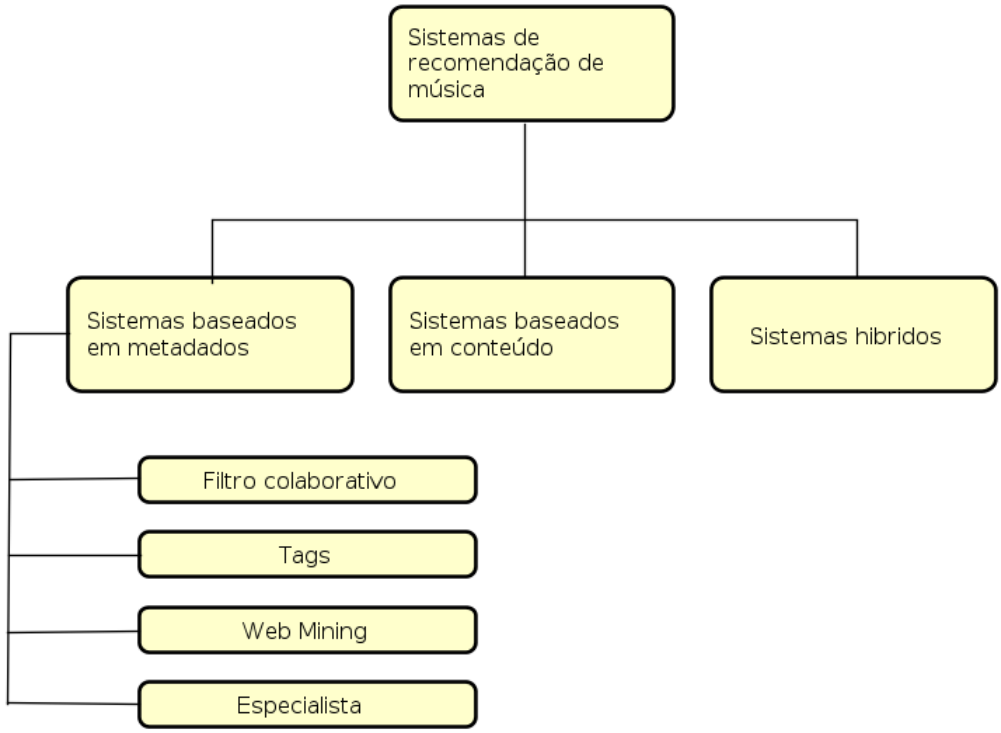

Figura 2.1: Classificação dos sistemas de recomendação (baseada em [5]).

2.1.1 Sistemas baseados em metadados

São chamados de sistemas baseados em metadados aqueles que utilizam informações sobre uma música que não são derivadas diretamente de seu sinal, mas são atribuídas por um humano. Esses dados podem ser anotações explícitas como gênero, *tags*, nome do artista etc., ou metadados implícitos como informações de compra, número de pulos e quantas vezes foi tocada. Embora metadados implícitos pareçam similares a dados de experiência do usuário, como será discutido na próxima seção, aqui eles estão relacionados a cada item, sendo as informações coletivas de vários usuários. Todos os métodos com base em metadados são considerados colaborativos, uma vez que uma única pessoa não pode criar anotações para todas as músicas existentes.

Em seguida analisaremos as diferentes formas que metadados podem tomar.

Filtro colaborativo: Filtros colaborativos utilizam informações de avaliação de usuários sobre um item (em uma escala de 5 estrelas, por exemplo). Muitos *players* modernos, como iTunes da Apple e Windows Media Player da Microsoft, permitem que usuários avaliem suas músicas, tipicamente em uma escala de 5 pontos.

Além de avaliações explícitas, o sistema também pode coletar avaliações implícitas, como número de vezes que uma música foi pulada, tocada inteiramente ou até certo ponto. Nesse ponto podemos distinguir dois métodos colaborativos, o baseado em usuários e o baseado em itens. O primeiro utiliza informações ao nível de usuários para estimar similaridades entre eles e com isso realizar recomendações, enquanto o segundo as realiza com base em similaridade entre itens, de forma que itens similares ao que o usuário ouve são recomendados.

***Tags*:** Métodos baseados em *tags* também são conhecidos como *tagging* colaborativo ou classificação social. Em sistemas baseados em *tags*, usuários criam anotações descritivas, chamadas de *tags*, para cada item. Uma música é descrita por todas as *tags* a ela atribuídas. Assim, músicas, álbuns e artistas com descrições parecidas são considerados similares. Uma grande vantagem de métodos baseados puramente em *tags* é o fato de que similaridades são facilmente explicadas, uma vez que basta verificar as anotações similares entre os itens. Por outro lado, existe uma grande esparsidade nas bases de dados disponíveis. Outra desvantagem é a necessidade de pré-processamento para *tags* sinônimas, como por exemplo "*Hip-Hop*"e "*HipHop*", além da remoção de *tags* pessoais como "Minha favorita". Um exemplo de base de dados desse tipo é a disponibilizada pelo *Last.fm*, a qual será utilizada neste trabalho.

***Web-mining*:** Técnicas baseadas em *web-mining* são similares às baseadas em *tags*. Mas diferentemente destas, que utilizam pessoas para criar anotações, *web-mining* busca retirar informações sobre uma música extraindo termos descritivos de páginas na internet. Outra diferença em relação aos sistemas baseados em *tags* é que, enquanto usuários podem criar qualquer anotação, em *web-mining* o sistema só pode buscar termos pré-definidos.

Especialistas: Diferentemente dos métodos anteriores, sistemas baseados em especialistas não coletam informações a partir de conjuntos de usuários, mas se baseiam em anotações criadas por especialistas. Uma grande desvantagem desses métodos é que a avaliação por especialista é custosa e não escala bem para grandes bases de músicas. Além disso, devido à subjetividade da interpretação de conceitos musicais, ainda podemos ter inconsistências nos dados. Um exemplo de sistema que utiliza anotações por especialista é o Pandora, onde anotações relacionadas a melodia, harmonia, ritmo, instrumentação, orquestração, arranjo etc. são criadas manualmente por especialistas. Mais de 400 atributos são classificados para cada item. A partir dessas anotações o sistema gera rádios interativas, voltadas para as preferências do usuário.

Com isso concluímos os métodos baseados em metadados. Agora discutiremos métodos baseados em conteúdo, que utilizam informações obtidas a partir do sinal de cada música, com o intuito de utilizar similaridades matemáticas para geração de recomendações.

2.1.2 Sistemas baseados em conteúdo

Sistemas baseados em conteúdo, em contraste com os baseados em metadados, não utilizam anotações criadas por usuários, mas tentam analisar o conteúdo do sinal de cada item na base de dados diretamente. A ideia é selecionar atributos significantes que descrevam bem uma música, gerando assim uma representação matemática de cada objeto. De posse dessa representação, podemos utilizar conceitos de similaridade matemática para avaliar como cada elemento da base de dados se relaciona com os outros. Uma questão importante para sistemas baseados em conteúdo é quais atributos devem ser selecionados, representando bem as características de cada canção.

Uma importante vantagem desse tipo de sistema é que, enquanto sistemas baseados em metadados necessitam que seres humanos criem as anotações para cada novo item (o que pode não acontecer imediatamente para itens novos e pouco conhecidos), os sistemas baseados em conteúdo necessitam apenas do sinal digital, podendo gerar recomendações para músicas novas e desconhecidas. Por outro lado, métodos baseados em conteúdo possuem a desvantagem de não incluírem informações culturais ou contextuais em suas recomendações. Além disso, não é sempre fácil selecionar atributos que descrevam bem os objetos e gerem recomendações significativas.

2.1.3 Sistemas híbridos

O princípio de sistemas híbridos de recomendação é utilizar as vantagens de dois ou mais sistemas distintos, ao mesmo tempo eliminando fraquezas específicas que cada um possa ter. Na literatura, muitas formas de combinação foram propostas para sistemas de recomendação de forma geral [10]. Para sistemas de recomendação de música, esse método ainda é pouco pesquisado. No caso mais simples, podemos combinar sistemas baseados em conteúdo e baseados em metadados para contornar problemas como *cold-start*, esparsidade e polarização de popularidade (discutidos na Seção 2.3).

2.2 Modelagem de usuários

O foco deste trabalho é em recomendações a partir de modelagem de itens, porém apresentamos aqui uma rápida discussão de como modelagem de usuários pode ser

utilizada para refinar recomendações a partir de informações no nível de usuário.

Informações como região geográfica, idade e sexo são chaves para modelagem de usuário. Além disso, pesquisas recentes indicam que inteligência, personalidade e preferências musicais estão correlacionados [11]. De acordo com Rentfrow e Gosling [12] [13], que estudaram a relação entre gosto musical e testes de personalidade, pessoas extrovertidas tendem a preferir músicas energéticas, enquanto pessoas que são extrovertidas e altamente sociáveis não só preferem músicas energéticas, mas que também possuam ritmo forte. Assim, modelagem de usuários é um componente importante para realização de boas recomendações. Existem dois aspectos nesse tipo de análise: modelagem de perfil e modelagem de experiência.

2.2.1 Modelagem de perfil

Celma [3] sugere que o perfil de um usuário pode ser dividido em 3 partes: demográfico, geográfico e psicológico. Informações demográficas incluem sexo, idade etc. Informações geográficas (país, estado etc.) são utilizadas para recomendações baseadas na cultura do usuário. Por fim, informações psicológicas são divididas em estáveis e fluidas. Informações consideradas estáveis incluem estilo de vida, personalidade etc. Já atributos fluidos são humor, atitude etc.

2.2.2 Modelagem de experiência

Dependendo do nível de conhecimento musical de um usuário, suas expectativas são variadas. Jenkins [14] analisou diferentes tipos de usuários e os caracterizou da seguinte forma:

Tipo	**Porcentagem**	**Atributos**
Sábio	7	Possui conhecimento musical vasto. Relaciona-se com música no dia a dia.
Entusiasta	21	Música é uma parte importante de sua vida, mas balanceada por outras atividades.
Casual	32	Música faz parte de sua vida, mas existem coisas mais importantes.
Indiferente	40	A parte predominante dos usuários: música não é uma parte importante de sua vida.

Tabela 2.1: Tipos de usuários.

Esses dados mostram que a experiência do usuário deve ser levada em conta. Por exemplo, dependendo de suas expectativas e experiências, um sistema pode ajustar o número de músicas a serem recomendadas e filtradas [15]. Outras informações de

experiência incluem vezes que um usuário ouviu ou pulou uma determinada música e detalhes de compras anteriores.

2.3 Problemas conhecidos em sistemas de recomendação

Esta seção busca discutir problemas existentes em sistemas de recomendação, e como eles se relacionam com os métodos de recomendação discutidos.

- **Problema de esparsidade (SP):** Quando itens possuem poucas informações (por exemplo, poucas anotações para sistemas que utilizam *tags*), sistemas de recomendação podem fazer sugestões de baixa qualidade. Esse tipo de problema comumente ocorre com itens desconhecidos. Esse problema também aparece se, mesmo que tenhamos uma boa quantidade de informação, ela é apenas no nível de artista, não no nível de músicas individuais.

- **Problema de *cold-start* (CSP):** O problema de *cold-start* é similar ao de esparsidade. Itens novos na base de dados ainda não possuem informações suficientes para receber boas recomendações.

- **Problema da ovelha cinza (GSP):** Para usuários que possuem opiniões muito diferentes das de usuários comuns, sistemas de recomendação não podem gerar boas sugestões, pois eles não concordam com ou discordam de nenhum grupo de usuários consistentemente.

- **Problema de polarização de popularidade (PBP):** Itens muito populares na base de dados costumam ser recomendados frequentemente. Esse problema ocorre especialmente em sistemas baseados em filtro colaborativo. Já itens pouco populares tendem a ser pouco recomendados e ficam escondidos no que é chamado de *long-tail* de recomendação [16][17][18].

- **Problema de falta de explicação (NEP):** À medida que algoritmos de recomendação aumentam de complexidade, a capacidade de explicação para cada sugestão resultante do algoritmo tende a se reduzir. Idealmente um sistema de recomendação deve ser transparente, de forma que o usuário possa entender o porquê dos itens recomendados.

- **Efeito de portfólio (PF):** Efeito de portfólio ocorre quando itens idênticos são recomendados. Em termos de recomendação de música vemos esse efeito ocorrer quando a maioria dos itens recomendados são do mesmo artista utilizado como *query*. No campo de recomendação de música esse problema também é chamado de *efeito de artista* por esse motivo.

- **Escalabilidade (SA):** Escalabilidade é a capacidade de o sistema lidar com o crescimento na quantidade de dados que devem ser processados. Um sistema que cria recomendações de alta qualidade mas possui pouca escalabilidade irá demorar para gerar novas recomendações, reduzindo a qualidade da experiência do usuário.

A Tabela 2.2 resume como esses defeitos costumam acontecer nos diversos métodos de recomendação discutidos [5].

Método	**SP**	**CSP**	**GSP**	**PBP**	**NEP**	**PE**	**SA**
Filtro Colaborativo	A	A	A	A	A	A	3
Tags	A	A	A	A	NA	NA	3
Web-Mining	A	A	A	A	NA	A	3
Especialistas	NA	NA	A	NA	NA	A	1
Conteúdo	NA	NA	A	NA	A	A	3

Tabela 2.2: Métodos de recomendação e seus problemas, onde NA significa *"Não afetado"* e A, *"Afetado"*. A coluna **SA** está dividida em uma escala onde 1 = *"Fraco"*, 2 = *"Médio"* e 3 = *"Bom"*.

A Tabela 2.2 parece indicar que métodos baseados em especialistas são uma boa solução, porém sua baixa escalabilidade os torna difíceis de serem utilizados na prática. Além disso, podemos ver que o problema da ovelha cinza é difícil de ser contornado, de forma que todos os métodos são afetados. Dos problemas restantes, muitos podem ser reduzidos utilizando-se a combinação de múltiplos métodos de recomendação (sistemas híbridos).

Nosso objetivo neste trabalho é desenvolver um método que combine as vantagens que métodos baseados em anotações semânticas possuem (ou seja, um sistema que possa ser utilizado com *tags*, *Web-Mining* ou anotações por especialistas) e que possua uma representação mais simples, visando a permitir o uso de cálculos simplificados de distância entre itens. Além disso, utilizaremos métodos baseados em conteúdo para contornar problemas como esparsidade do banco de dados e *cold-start* em itens novos.

2.4 Medindo qualidade de recomendações

O método mais comum para medir a qualidade de recomendações é a avaliação humana, onde ouvintes avaliam a similaridade entre uma música usada como *query* e as recomendações que o sistema gera, utilizando a satisfação do usuário como critério de qualidade de uma recomendação [5]. No entanto, esse método é muito custoso e demorado, pois depende de usuários que participem do teste ouvindo uma

playlist completa e ao término de cada item o avaliem através de algum tipo de questionário.

Um típico problema encontrado na literatura é o tamanho do experimento. Muitos estudos são baseados em apenas 10 a 20 participantes [19] que avaliam *playlists* através de questionários ou aplicativos que foram desenvolvidas para o estudo. Aplicativos costumam ser mais efetivos em grande escala pois permitem que o usuário avalie o sistema em seu próprio tempo e sem necessidade de haver instalações físicas para reunir os usuários que realizarão o experimento. Por exemplo, [2] apresenta resultados para um estudo envolvendo 185 usuários utilizando esse método.

Um método diferente é apresentado em [20], onde o autor observa o número de vezes que uma música é pulada em uma *playlist* gerada. Todas as músicas puladas são tratadas como falsos positivos e músicas ouvidas completamente são tratadas como verdadeiros positivos. Um método como esse pode ser utilizado em um sistema de testes *A/B*. Nesse sistema, usuários são divididos em diferentes grupos e cada grupo recebe *playlists* geradas com técnicas diferentes. Após o periodo de testes, os resultados podem ser utilizados para analisar qual gerador de *playlists* funcionou melhor (por exemplo, gerou menos falsos positivos). Assim como a avaliação anterior, esse método ainda é muito custoso, uma vez que é necessário que usuários passem por toda a *playlist* e escutem músicas completas. No entanto, algumas plataformas como o *Last.fm* permitem o acesso a informações sobre padrões de aprovação de seus usuários através de *logs* de uso. Porém, poucos trabalhos que utilizam esses recursos foram desenvolvidos.

Outro método de avaliação busca aproximar a forma como um usuário avaliaria a qualidade de uma recomendação [21]. Uma forma típica de atingir esse objetivo é medir a homogeneidade ou diversidade de uma *playlist*. Uma forma simples de calcular a diversidade em uma *playlist* é contar o número de artistas ou gêneros distintos que foram recomendados. Alguns trabalhos que utilizam essa forma de avaliação são [22], [23], [20] e [24]. No entanto, consideramos essa forma de avaliação muito simplista, uma vez que alguns artistas podem ser avaliados como pertencendo a uma gama de gêneros distintos, ou gêneros como jazz, que são heterogêneos internamente. Outro ponto negativo de tais métodos é que alguns estudos indicam que diversidade pode ser tão importante quanto homogeneidade para usuários que irão avaliar os sistemas [25][26][27][28].

Em contraste, muitos sistemas de geração de recomendações não utilizam qualquer método formal de avaliação de qualidade.

Neste trabalho utilizaremos um método de avaliação similar ao apresentado em [7], onde *playlists* são retiradas do site *yes.com*, que reúne listas de músicas tocadas em estações de rádio nos Estados Unidos. Estações que buscam tocar músicas de um determinado gênero são escolhidas, e tratamos como verdadeiros positivos músicas

que foram recomendadas pelo sistema e também já foram tocadas em uma das estações de determinado gênero. Este método busca simular uma comparação com *playlists* geradas por profissionais.

Capítulo 3

Modelo LDA

O problema de modelar coleções de dados discretos (especificamente corpos de texto) deu origem ao estudo de *modelagem de tópicos*. Neste trabalho iremos utilizar o método de *Latent Dirichlet Allocation* (LDA) para modelar conjuntos de *tags* associadas às músicas. Este capítulo apresentará o modelo LDA, proposto em [4].

Na Seção 3.1 apresentaremos a notação e terminologia utilizada em modelagem de tópicos. Em seguida discutiremos o problema de modelagem de tópicos, apresentando diferentes soluções conhecidas na literatura, finalmente chegando ao modelo LDA (Seção 3.2). Seguimos com a Seção 3.3, onde revisamos o LDA, um modelo generativo probabilístico para corpos de textos e dados discretos. Apresentamos métodos de estimação de parâmetros e inferência através do método do estimador de Gibbs na Seção 3.4 e realizamos comentários sobre os hiperparâmetros do modelo em 3.5. Na Seção 3.6 demonstramos uma aplicação do modelo a fim de exemplificar seu uso em recomendação de música. Por fim, na Seção 3.7 definimos uma medida de similaridade entre documentos baseada no vetor de tópicos resultante do modelo LDA.

3.1 Notação e terminologia

Devido à sua aplicação principal em modelagem de textos, utilizaremos termos usados nessa área, com expressões como "palavra", "documentos" e "corpus". No entanto, é importante observar que esses modelos não se restringem à modelagem de textos e podem ser utilizados para qualquer conjunto de dados discretos.

Definimos os seguintes termos:

Palavra: A unidade básica de dados discretos, definida como um item contido em um vocabulário indexado por $\{1, ..., V\}$.

Documento: Uma sequência de N palavras, denotada por $\vec{w} = (w_1, w_2, ..., w_N)$, onde w_n é a n-ésima palavra na sequência.

Corpus: Uma coleção de M documentos, denotada por $D = \{\vec{w}_1, \vec{w}_2, ..., \vec{w}_M\}$. Desejamos encontrar um modelo probabilístico para corpora de texto.

3.2 Modelagem de tópicos

O problema de modelagem de tópicos se resume a encontrar uma descrição dos membros de uma coleção de dados discretos que permita o processamento eficiente de grandes conjuntos de dados, preservando relações estatísticas significantes. Dessa forma, é possível utilizar essa descrição para desenvolver métodos de classificações, estatísticas de resumo, estudo de similaridade e relevância.

Antes de introduzirmos o modelo LDA realizaremos uma discussão dos modelos *tf-idf* (do inglês *Term Frequency - Inverse Document Frequency*), unigrama e mistura de unigramas, pois cada um introduz conceitos necessários para a melhor compreensão do modelo final.

***tf-idf*:** A metodologia básica proposta no campo de Recuperação de Informação (RI) reduz cada documento no corpus a um vetor de números reais, cada qual representa frações de contagens. No popular modelo *tf-idf* [29], um vocabulário de "palavras" é escolhido e:

1. Para cada documento do corpus, contam-se as ocorrências de cada palavra; a partir da contagem, define-se alguma forma de frequência normalizada dessas palavras (a *tf*).

2. Para cada palavra, contam-se os documentos no corpus que a contêm; a partir da contagem, define-se alguma forma de frequência inversa normalizada (a *idf*).

3. Calcula-se o produto desses dois fatores (o *tf-idf*).

O resultado é uma matriz X em que cada linha corresponde a uma palavra, cada coluna corresponde a um documento e seus elementos são os valores de *tf-idf* de cada palavra para cada documento do corpus.

Enquanto o modelo *tf-idf* possui vantagens (sendo uma delas a simplicidade de se calcular), ele faz pouco para a redução da dimensionalidade dos dados, além de revelar pouco sobre a relação estatística inter e intradocumentos. Pesquisadores do campo de Recuperação de Informação (RI) propuseram diversas outras técnicas de redução de dimensionalidade em que esses efeitos possam ser capturados e estudados

através do uso de métodos bayesianos, que permitem criar um modelo generativo para corpus de texto. Dessa forma, cada documento é reduzido a uma distribuição de probabilidades sobre uma série de tópicos, sendo esse um modelo mais compacto que o *tf-idf*. Agora veremos alguns desses modelos, que utilizam conceitos similares ao LDA e facilitarão o entendimento do modelo final.

Modelo Unigrama: Nesse modelo palavras de cada documento são selecionadas independentemente de uma única distribuição multinomial (discutida no Apêndice A). Assim, temos que a probabilidade de um documento a ser gerado é:

$$p(\vec{w}_m) = \prod_{n=1}^{N} p(w_n). \tag{3.1}$$

Quando estimada de um corpus, a distribuição de palavras pode ser interpretada como uma representação de tópicos, assumindo que todos os documentos são gerados a partir de exatamente um tópico. A rede bayesiana geradora desse modelo é dada na Figura 3.1.

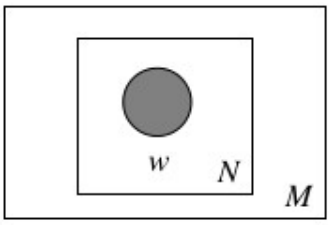

Figura 3.1: Modelo unigrama.

Embora extremamente simples, esse modelo serve de base para os próximos modelos que serão estudados.

Mistura de unigramas: Ao estendermos o modelo de unigramas adicionando uma variável de mistura z, representando tópicos, obtemos o modelo de mistura de unigramas [30]. Nesse modelo cada documento é gerado primeiramente selecionando-se aleatóriamente um tópico z e, então, selecionando-se N palavras independentemente a partir da distribuição multinomial $p(w|z)$, a distribuição de probabilidades para as palavras no tópico z. Assim, a distribuição de probabilidade de um documento é a seguinte combinação convexa:

$$p(\vec{w}_m) = \prod_{i=1}^{N} \sum_{k} p(w_i = w|z = k)p(z = k), \quad \sum_{k} p(z = k) = 1, \tag{3.2}$$

onde cada componente $p(w_i = w|z = k)$ segue uma distribuição multinomial sobre os termos que correspondem a um dos tópicos do corpus. As proporções de mistura

consistem da probabilidade dos tópicos $p(z = k)$. Dessa forma chegamos em um modelo em que a distribuição de palavras pode ser interpretada como uma representação de tópicos se assumirmos que cada documento é gerado a partir de apenas um tópico, de onde são retiradas as palavras que formam o documento. A rede bayesiana geradora do modelo é dada na Figura 3.2:

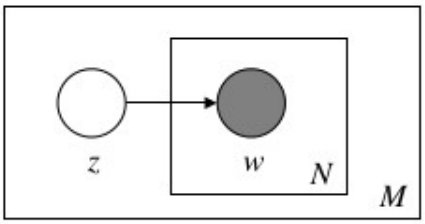

Figura 3.2: Modelo mistura de unigramas.

Seguindo essa linha de raciocínio, chegamos ao modelo LDA, que permite que cada documento seja criado por um conjunto de tópicos em vez de apenas um. Na próxima seção iremos descrever esse modelo.

3.3 Modelo LDA

O modelo Latent Dirichlet Allocation (LDA) [4] é um modelo generativo probabilístico que é capaz de estimar propriedades de observações de uma multinomial utilizando treinamento não supervisionado. Ele é obtido ao relaxarmos o pressuposto de que cada documento é gerado a partir de apenas um tópico. Ao permitirmos que um documento seja gerado a partir de um conjunto de tópicos, criamos uma descrição mais robusta para análises estatísticas posteriores.

Partindo de um modelo de mistura de unigramas, o modelo LDA vai ainda mais longe e, em vez de utilizar uma proporção de tópicos global (uma mesma proporção de tópicos para todos os documentos), ele condiciona as probabilidades dos tópicos ao documento a que uma palavra pertence. O modelo generativo para um corpus

de texto é descrito no algoritmo 1.

Algoritmo 1: Modelo generativo para LDA.

/* Nível de tópicos: */;
para cada *tópico* $k \in [1, K]$ **faça**
  selecione os componentes da mistura $\vec{\varphi}_k \sim Dir(\vec{\beta})$;
fim
/* Nível de Documentos: */ ;
para cada *documento* $m \in [1, M]$ **faça**
  selecione as proporções de mistura $\vec{\theta}_m \sim Dir(\vec{\alpha})$;
  selecione o tamanho do documento $N_m \sim Poisson(\eta)$;
  /* Nível de palavras: */ ;
  para cada *palavra* $n \in [1, N_m]$ **faça**
    selecione o tópico $z_{m,n} \sim Multi(\vec{\theta}_m)$;
    selecione a palavra $w_{m,n} \sim Multi(\vec{\varphi}_{z_{m,n}})$;
  fim
fim

Cada documento possui um vetor de proporções de mistura $\vec{\theta}$ que indica a probabilidade de seleção de cada tópico que será utilizado para a amostragem na distribuição multinomial. Uma vez que $\vec{\theta}$ é amostrado a partir de uma distribuição de Dirichlet (veja o Apêndice B para mais detalhes) podemos garantir que seus componentes somarão 1. A escolha da distribuição de Dirichlet também se justifica porque esta é a conjugada priori da distribuição multinomial, fato que é utilizado para simplificação dos resultados. Assim, o vetor $\vec{\theta}$ pode ser interpretado como o peso que cada tópico possui para um documento. De forma similar, $\vec{\varphi}$ representa a distribuição de palavras em um tópico, de forma que cada componente pode ser interpretado como a importância da palavra em um tópico.

Assim, como os vetores $\vec{\varphi}_k$ foram selecionados a priori para todos os tópicos e a proporções de tópicos $\vec{\theta}$ foi selecionada, a n-ésima palavra no m-ésimo documento (notada como $w_{m,n}$) é gerada primeiramente amostrando-se um tópico $z_{m,n}$, com as probabilidades descritas em $\vec{\theta}$ e em seguida selecionando-se uma palavra com probabilidade descrita em $\vec{\varphi}_{z_{m,n}}$. Esse processo é repetido para todas as palavras do documento. Uma vez que cada documento possui um vetor $\vec{\theta}$ distinto, sua seleção deve ser feita antes da geração das palavras do documento.

As quantidades necessárias para completa descrição do modelo generativo são:

M, o número de documentos a serem gerados (constante);

K, o número de tópicos (constante);

V, número de termos no vocabulário (constante);

$\vec{\alpha}$, um hiperparâmetro da mistura de proporções (um K-vetor ou escalar, caso

simétrico);

$\vec{\beta}$, um hiperparâmetro na mistura de componentes dos tópicos (V-vetor ou escalar, caso simétrico);

$\vec{\theta}_m$, notação paramétrica para $p(z|d = m)$, a proporção de mistura de tópicos para o documento m;

Θ, a matriz composta por M proporções de mistura de tópicos $\vec{\theta}$ (matriz KxM);

$\vec{\varphi}_k$, notação paramérica para $p(t|z = k)$, a mistura de componentes no tópico k;

Φ, a matriz composta por K misturas de componentes em tópicos (matriz KxV);

N_m, tamanho do documento, aqui modelado por uma distribuição de Poisson com parâmetro constante η;

$z_{m,n}$, uma variável indicadora que seleciona a n-ésima palavra no documento m;

$w_{m,n}$, indicador da n-ésima palavra no documento m.

A rede bayesiana que descreve o modelo generativo do LDA se encontra na Figura 3.3.

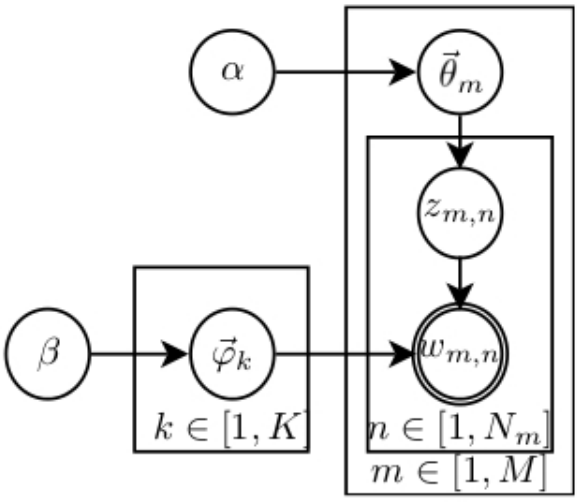

Figura 3.3: Modelo LDA.

De posse do modelo generativo, desejamos descrever a distribuição de probabilidade do corpus de documentos (ou seja, $p(D)$). Primeiramente vamos descrever a probabilidade de seleção da palavra $w_{m,n}$. De acordo com o modelo apresentado na Figura 3.3, temos que a probabilidade de seleção de uma palavra é:

$$p(w_{m,n} = w|\vec{\theta}_m, \Phi) = \sum_{k=1}^{K} p(w_{m,n} = w|\vec{\varphi}_k)p(z_{m,n} = k|\vec{\theta}_m). \tag{3.3}$$

Note que essa fórmula nada mais é que uma instanciação da forma do modelo de mistura de unigramas na equação 3.2, condicionando a seleção do tópico a uma proporção de mistura $\vec{\theta}_m$. Além disso, a partir da rede bayesiana, podemos definir

a distribuição conjunta de todas as variáveis escondidas dados os hiperparâmetros para um documento.

$$p(\vec{w}_m, \vec{z}_m, \vec{\theta}_m, \Phi|\vec{\alpha}, \vec{\beta}) = \prod_{n=1}^{N_m} p(w_{m,n}|\vec{\varphi}_{m,n})p(z_{m,n}|\vec{\theta}_m)p(\vec{\theta}_m|\vec{\alpha})p(\Phi|\vec{\beta}). \tag{3.4}$$

Por fim, podemos obter a probabilidade de $\vec{w}_m$, do documento, marginalizando sobre $\vec{\theta}_m$, Φ e $z_{m,n}$:

$$p(\vec{w}_m|\vec{\alpha}, \vec{\beta}) = \int\int p(\vec{\theta}_m|\vec{\alpha})p(\Phi|\vec{\beta}) \prod_{n=1}^{N_m} \sum_{z_{m,n}} p(w_{m,n}|\varphi_{z_{m,n}})p(z_{m,n}|\vec{\theta}_m)\mathrm{d}\Phi\mathrm{d}\vec{\theta}_m, \tag{3.5}$$

podendo ser reescrita como

$$p(\vec{w}_m|\vec{\alpha}, \vec{\beta}) = \int\int p(\vec{\theta}_m|\vec{\alpha})p(\Phi|\vec{\beta}) \prod_{n=1}^{N_m} p(w_{m,n}|\vec{\theta}_m, |\Phi)\mathrm{d}\Phi\mathrm{d}\vec{\theta}_m. \tag{3.6}$$

Assim, a verossimilhança completa para o corpus D é dada pelo produto das probabilidades de cada documento:

$$p(D|\vec{\alpha}, \vec{\beta}) = \prod_{m=1}^{M} p(\vec{w}_m|\vec{\alpha}, \vec{\beta}). \tag{3.7}$$

É importante observar que esse modelo possui a propriedade de permutabilidade como definida por De Finetti [4]. Isso significa que a ordem das palavras no documento não é considerada no modelo. Para nosso estudo, isso resulta em a ordem das *tags* de cada item da base de dados ser irrelevante, como é de se esperar em tal sistema.

3.4 Estimação e inferência de parâmetros

De posse da descrição completa da distribuição do corpus, podemos finalmente nos voltar ao problema de estimação dos parâmetros principais do modelo, sendo estes $\vec{\theta}_m$ para cada documento e $\vec{\varphi}_k$ para cada tópico ($\vec{\alpha}$ e $\vec{\beta}$ serão tratados como hiperparâmetros do modelo—mais detalhes na Seção 3.5). No entanto, a distribuição encontrada na equação 3.7 é intratável para os métodos convencionais de estimação devido à sua complexidade. Assim, devemos buscar outra forma de realizar essa estimação.

Para estimação dos parâmetros do modelo, utilizaremos os cálculos do amostrador de Gibbs, proposto em [31]. O método consiste na utilização de uma distribuição

de Dirichlet simétrica, de forma que pode haver pequenas diferenças em relação ao LDA original, mas o método é computacionalmente mais ágil.

O amostrador de Gibbs (GS) é um caso especial de simulação de Markov-Chain Monte Carlo (MCMC). Métodos MCMC são capazes de emular distribuições de probabilidade $p(\vec{x})$ em grandes dimensões [32] aproveitando-se da propriedade estacionária de uma cadeia de Markov. Especificamente, o GS retira uma amostra de cada dimensão x_i condicionada ao valor das demais dimensões, denotado por $\vec{x}_{-i}$.

Como demonstrado em [33], o procedimento completo para o GS na estimação do modelo LDA é dado no Algoritmo 2, onde as seguintes notações são definidas:

$n_m^{(k)}$ é a contagem das vezes que o tópico k foi observado com uma palavra no documento m;

n_m é a soma de $n_m^{(k)}$ para todos os termos;

$n_k^{(t)}$ é a contagem das vezes que a palavra t foi observada no tópico k;

n_k é a soma de $n_k^{(t)}$ para todos os termos.

As seguintes equações são utilizadas:

$$p(z_i = k|\vec{z}_{-i}, \vec{w}) \propto \frac{n_{k,-i}^{(t)} + \beta_t}{\sum_{t=1}^{V}(n_{k,-i}^{(t)} + \beta_t)} \cdot \frac{n_{m,-i}^{(t)} + \alpha_k}{(\sum_{k=1}^{K}(n_{m,-i}^{(k)} + \alpha_k)) - 1}, \tag{3.8}$$

$$\varphi_{k,t} = \frac{n_k^{(t)} + \beta_t}{\sum_{t=1}^{V}(n_k^{(t)} + \beta_t)}, \tag{3.9}$$

$$\theta_{m,k} = \frac{n_m^{(t)} + \alpha_k}{\sum_{k=1}^{K}(n_m^{(k)} + \alpha_k)}. \tag{3.10}$$

A notação $n_{\cdot,-i}^{(\cdot)}$ se refere à contagem retirando-se a i-ésima posição do tópico ou documento.

Uma das desvantagens de métodos MCMC é a necessidade de se determinar a quantidade de passos iterativos para retirada do efeito da inicialização das variáveis, sendo essa etapa chamada de *burn-in*. Uma das técnicas para avaliar a convergência da cadeia de Markov é a checagem manual da similaridade semântica das palavras em cada tópico.

3.5 Seleção de hiperparâmentros

Os hiperparâmetros da distribução de Dirichlet criam um efeito de suavização nos parâmetros da distribuição multinomial. A redução dos valor de α e β resultará em associações de tópicos mais restritas, logo Θ e Φ serão mais esparsos. A esparsidade em Φ, controlada por β, significa que o modelo irá preferir atribuir menos palavras para cada tópico. De forma similar, a esparsidade de Θ, controlada por α, implica

Algoritmo 2: Algoritmo GS para LDA.

/* Inicialização: */ ;
$n_m^{(k)}, n_m, n_k^{(t)}, n_k = 0$;
para cada *documento* $m \in [1, M]$ **faça**
    para cada *palavra* $n \in [1, N_m]$ **faça**
        selecione o índice de de tópico $z_{m,n} = k \sim Multi(\frac{1}{K})$;
        $n_m^{(k)} = n_m^{(k)} + 1$;
        $n_m = n_m + 1$;
        $n_k^{(t)} = n_k^{(t)} + 1$;
        $n_k = n_k + 1$;
    fim
fim
/* Algoritmo GS: */ ;
enquanto *Não terminar* **faça**
    para cada *documento* $m \in [1, M]$ **faça**
        para cada *palavra* $n \in [1, N_m]$ **faça**
            $n_m^{(k)} = n_m^{(k)} - 1$;
            $n_m = n_m - 1$;
            $n_k^{(t)} = n_k^{(t)} - 1$;
            $n_k = n_k - 1$;
            selecione indice de tópico $\hat{k} \sim p(z_i|\vec{z}_{-i}, \vec{w})$, de acordo com equação 3.8 (utilizando os decrementos calculados anteriormente) ;
            $n_m^{(\hat{k})} = n_m^{(\hat{k})} + 1$;
            $n_m = n_m + 1$;
            $n_{\hat{k}}^{(t)} = n_{\hat{k}}^{(t)} + 1$;
            $n_{\hat{k}} = n_{\hat{k}} + 1$;
        fim
    fim
    /* Checagem de convergência e leitura de parâmetros: */ ;
    se *convergiu e L passos desde ultima leitura* **então**
        calcule Φ de acordo com equação 3.9 ;
        calcule Θ de acordo com equação 3.10 ;
    fim
fim

que o modelo prefere definir documentos com menor número de tópicos.

Bons resultados heurísticos foram encontrados utilizando-se $\alpha = 50/K$ e $\beta = 0{,}1$ [34], que serão os valores utilizados no restante do texto, a menos que se explicite algo diferente.

3.6 Aplicação do LDA em um corpus

A fim de demonstrar os resultados obtidos pelo modelo LDA, apresentamos nesta seção os resultados empíricos obtidos ao modelarmos o banco de dados TREC AP, que consiste de 2247 artigos de notícia. A base de dados foi tratada para a remoção de 173 *stop words*, palavras que são selecionadas para serem removidas a priori, pontuações e palavras que aparecem apenas uma vez. Após o tratamento, o corpus de texto possui 21110 termos distintos. Um modelo LDA foi estimado utilizando-se 100 tópicos, $\beta = 0{,}1$ e $\alpha = 1$. O número de tópicos foi selecionado a partir do estudo cuidadoso da base de dados, com o objetivo de criar o exemplo a seguir.

O gráfico apresentado na Figura 3.4 mostra a distribuição de $\vec{\theta}$ para o artigo selecionado. Os dois maiores picos de probabilidade se encontram nos tópicos 4 e 84. Na Tabela 3.1, apresentamos os dois tópicos de maior probabilidade para o artigo, ordenando as palavras de forma decrescente em suas probabilidades ($\vec{\varphi} = p(w|z = k)$), para as primeiras 15 palavras. O nome representativo do tópico é derivado das palavras contidas neste e atribuído por uma pessoa. Um exemplo de artigo modelado se encontra na Figura 3.5, onde as cores das palavras representam a qual tópico estas possuem maior probabilidade de pertencer. Os tópicos selecionados para a Tabela 3.1 são os 2 de maior probabilidade para o texto.

Presidência	Sindicato
Bush	workers
Dukakis	union
campaign	labor
president	strike
democratic	contract
Jackson	jobs
republican	employees
presidential	work
state	pay
convention	plant
George	unions
Reagan	service
primary	job
candidate	wage

Tabela 3.1: Palavras nos tópicos mais representativos para o texto na Figura 3.5.

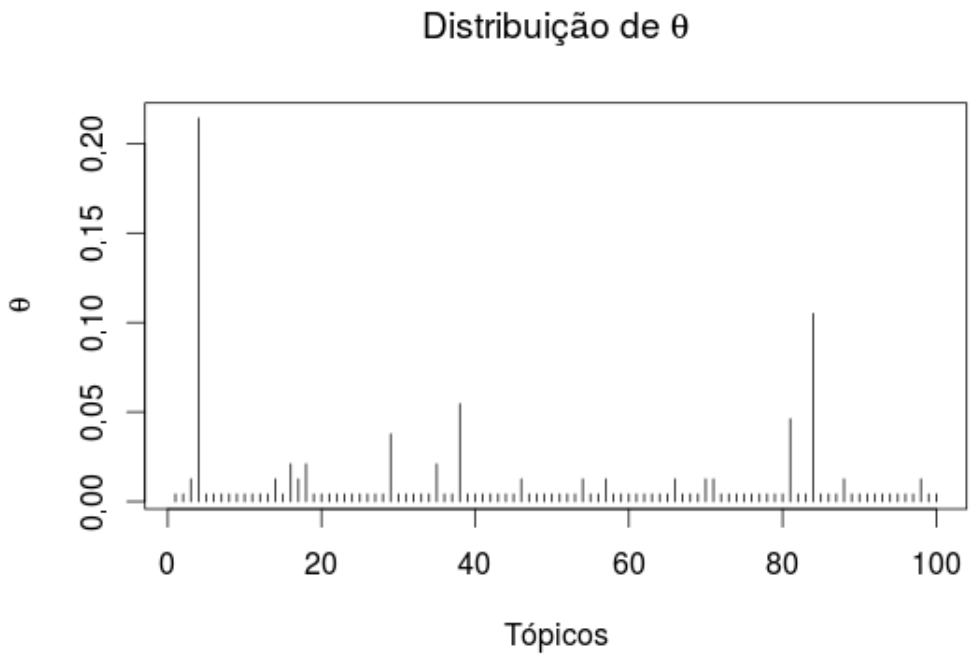

Figura 3.4: Distribuição de $\vec{\theta}$. Picos se encontram em 4 e 84.

There will be no organized union boost behind a single candidate in Saturday's Democratic caucuses in Michigan, a state where union members can wield more clout than almost anywhere else. While national labor leaders are assuming Michael Dukakis will be the eventual nominee, they are prevented from endorsing him by what appears to be growing rank-and-file support for Jesse Jackson, who has gotten more union votes than any of the other candidates in primaries so far. Richard Gephardt also has considerable union support. None of the Democratic candidates appears to have won the hearts _ or votes _ of a majority of the state's 750,000 rank-and-file union workers, nearly half of them members of the United Auto Workers.

Figura 3.5: Um exemplo de artigo do corpus. Cada cor representa de qual dos tópicos da Tabela 3.1 a palavra possui maior probabilidade de ter sido selecionada.

Assim, é intuitivo que um texto pode ser representado pelo seu vetor de mistura de tópicos $\vec{\theta}$. Essa representação é mais compacta do que utilizar todas as palavras do documento.

Utilizando esse exemplo, definiremos na seção seguinte o conceito de similaridade entre documentos.

3.7 Definindo similaridades entre documentos

Desejamos definir similaridades entre itens distintos da base de dados para realizar recomendações. Esse objetivo é alcançado através dos vetores de mistura, $\vec{\theta}_m$, que definem as proporções de cada tópico em determinado documento da base de dados.

Dessa forma, podemos definir que documentos similares são aqueles que possuem vetores $\vec{\theta}_m$ similares. A similaridade pode ser calculada por qualquer fórmula matemática que defina distância entre vetores. Assim, o documento i que é mais similar ao documento j será:

$$i = \text{argmin}\{D(\vec{\theta}_i, \vec{\theta}_j)\}, \forall i, i \neq j, \tag{3.11}$$

onde $D(\cdot, \cdot)$ é uma função de distância definida para um espaço de k dimensões, sendo k o número de tópicos utilizados no modelo. Para gerar uma lista de recomendação, iremos listar as n menores distâncias encontradas na base de dados.

Nas Figuras 3.6, 3.7 e 3.8 demonstramos esse conceito utilizando a distância euclidiana para encontrar os 3 textos mais similares ao apresentado na Figura 3.5 no modelo calculado na Seção 3.6.

Presidential rivals George Bush and Michael Dukakis encountered loud, hostile demonstrators Tuesday as Oregon shipyard workers shouted ``union buster'' at the Republican candidate and anti-abortion activists called the Democratic nominee ``baby killer.'' ``Do not gamble on another liberal Democrat coming out of nowhere,'' Bush said, shouting to be heard over the boos of workers at Northwest Marine Iron Works in Portland, Ore. In the Chicago suburb of Niles, Ill., Dukakis was interrupted for several minutes by the anti-abortion protesters early in a speech on economic policy. ``I just hope as the campaign goes on we can address the real issues that face the campaign and that face the country and do so in a way that's respectful of each other,'' he said afterward.

Figura 3.6: Texto com distância 0,135.

Democratic Sen. Paul Simon captured a second term Tuesday, fending off a challenge by Republican Rep. Lynn Martin, whose anti-tax message and late call for limiting the terms of Washington lawmakers failed to dent the popularity of the liberal incumbent.

Figura 3.7: Texto com distância 0,14.

Lowell P. Weicker Jr., the maverick three-term Republican senator whose liberal positions often enraged his party's more conservative wing, was ousted Tuesday by moderate Democrat Joseph I. Lieberman, the state's attorney general.

Figura 3.8: Texto com distância 0,155.

Observe que todos os textos estão relacionados ao sistema político americano. Em particular, o texto mais próximo, apresentado na Figura 3.6, também está relacionado a sindicatos, assim como texto original. Esse exemplo ilustra o poder do modelo LDA em reduzir a dimensão da base de dados ao reduzir sua representação anterior de um vetor de palavras de tamanho variável para um vetor numérico de

tamanho fixo sem perder a capacidade de representar similaridade entre itens na base de dados.

Assim, dizemos que dois documentos i e j são similares se a distância entre seus vetores de distribuição de tópicos $D(\vec{\theta}_i, \vec{\theta}_j)$ for pequena. Em um contexto de recomendação, ao ordenarmos de forma crescente as distâncias obtidas entre um dos documentos da base de dados e todos os demais, podemos interpretar os K documentos de menor distância como as *top-K* recomendações a partir de uma *query* de interesse.

Capítulo 4

Utilização do modelo LDA em um conjunto de *tags*

Neste capítulo apresentamos uma forma de representar músicas, usuários e *playlists* a partir de um modelo LDA. Essa representação nos permite reduzir um vetor de anotações sobre as músicas a um vetor númerico contínuo que irá ser utilizado para gerar recomendações de *playlists* para usuários.

Na Seção 4.1 apresentamos a base de dados utilizada no trabalho, baseada na coleção *Million Song Dataset*. Na Seção 4.7.2 são discutidas as particularidades da criação do modelo LDA para um conjunto de *tags*. Na Seção 4.3 discutimos os resultados e apresentamos uma maneira de interpretá-los. Em seguida, nas Seções 4.4, 4.5, e 4.6 apresentamos formas de modelar músicas, *playlists* e usuários, respectivamente, utilizando o modelo apresentado anteriormente. Concluímos o capítulo realizando experimentos para os métodos de recomendação de música na Seção 4.7.4.

4.1 As bases de dados

A seleção de músicas foi baseada na coleção disponibilizada no *Million Song Dataset* (MSD), uma base utilizada para trabalhos de recuperação de informação em um contexto de música. Essa base inclui dados como o conjunto de *tags* de músicas (essas *tags*, por sua vez, retiradas do serviço *last.fm*), álbum, artista, informações referentes ao áudio disponibilizadas pelo sistema *The Echo Nest* e outras geradas pela comunidade que utiliza essa base [8]. Também utilizadas para a análise foram as bases apresentadas em [35], uma extensão pública da base *MSD*, contendo informações de gênero musical para as músicas, bem como informações de áudio.

A fim de tornar os testes necessários mais ágeis, apenas um subconjunto arbitrário de 10000 músicas foi considerado para os exemplos das Seções 4.7.2, 4.3, 4.4, 4.5 e 4.6. Destas, apenas as 4833 que possuíam *tags* foram, afinal, selecionadas.

No entanto, para os experimentos finais, uma base mais robusta, apresentada na Subseção 4.7.1, foi utilizada.

4.2 Modelagem

Com o objetivo de gerar recomendações baseadas em *tags*, iremos realizar um processo similar ao apresentado em [7]. Cada documento na base de dados consistirá de um conjunto de *tags*, pré-processadas para retirada de termos únicos e não informativos. Um modelo LDA será gerado, de forma que cada música possua um vetor $\vec{\theta}_m$ para representá-la. Essa nova representação é claramente menor que todo o conjunto de *tags* na base de dados, mesmo que estas sejam substituídas por *tokens* númericos, uma vez que cada documento passa a ser estritamente descrito por um vetor de k dimensões, enquanto o número de *tags* pode variar entre músicas.

O tratamento das *tags* é realizado em 4 etapas:

1. Todas as anotações são convertidas para maiúsculo.
2. São removidas as *tags* repetidas que se diferenciavam apenas na caixa de algumas letras, como *Rock*, *rock* e *ROCK*.
3. Realizamos um tratamento das *tags* para a remoção de dados não informativos (anotações como *FAVOURITE*, *FAVOURITE_ARTISTS* e *I_LIKE*) através de uma lista de *stop words* escolhidas manualmente.
4. Por fim, *tags* que aparecem apenas uma única vez na base também são removidas.

Esse tratamento busca refinar as informações contidas na base de dados, removendo informações de baixa relevância que afetam a qualidade do modelo e a interpretação dos tópicos. Para demonstrar os efeitos desse tratamento, modelamos a base de dados com 13 tópicos sem realizar o refinamento prévio de *tags*. Os tópicos encontrados se encontram na Tabela 4.1.

As *tags* em destaque são as anotações problemáticas. Em especial, o tópico 7 possui 4 anotações que não agregam nenhuma informação sobre o gênero musical e que não ajudariam na recomendação. Outro problema, que pode ser observado no tópico 4, se refere à repetição de anotações escritas de maneiras distintas (nesse caso, *Hip-hop* e *hiphop*). O tratamento busca remover essas anotações não informativas da base de dados para gerar uma base que ao ser modelada ofereça mais informações de gênero e características musicais que poderão ser utilizadas para geração de recomendações e também sua interpretação. A tabela 4.2 mostra os tópicos após o processamento.

Tópico 1	Tópico 2	Tópico 3	Tópico 4
rock	blues	jazz	**Hip-Hop**
cover	guitar	instrumental	**hip_hop**
german	classic_rock	reggae	rap
covers	rock	world	funk
christian	blues_rock	new_age	**heard_on_pandora**
experimental	70s	piano	hiphop
live	guitar_virtuoso	french	funky
industrial	Psychedelic_Rock	Smooth_Jazz	old_school
best	hard_rock	Soundtrack	soul
Tópico 5	Tópico 6	Tópico 7	Tópico 8
beautiful	oldies	alternative	alternative_rock
Mellow	country	indie	rock
chill	folk	indie_rock	alternative
singer-songwriter	pop	alternative_rock	punk
acoustic	male_volcalists	00s	hard_rock
sad	60s	**Awesome**	**favorites**
easy_listening	Soundtrack	**Favorite**	emo
pop	classic_country	**Favourites**	Progressive_rock
textbffavorites	fun	**loved**	punk_rock
Tópico 9	Tópico 10	Tópico 11	Tópico 12
female_vocalists	rock	metal	eletronic
pop	80s	heavy_metal	eletronica
female_vocalist	classic_rock	hard_rock	dance
sexy	british	death_metal	ambient
rnb	new_wave	black_metal	chillout
00s	punk_rock	**Awesome**	electro
soul	punk	**seen_live**	downtempo
dance	70s	glam_rock	House
female	post_punk	Progressive_metal	techno
Tópico 13			
90s			
pop			
rock			
latin			
singer-songwriter			
spanish			
female_vocalists			
latin_pop			
legend			

Tabela 4.1: Palavras mais significativas para os tópicos gerados na modelagem das músicas.

Tópico 1	Tópico 2	Tópico 3
INDIE	JAZZ	INDIE
INDIE_ROCK	FUNK	INDIE_ROCK
ALTERNATIVE	SOUL	INDIE_POP
BRITISH	INSTRUMENTAL	FOLK
ALTERNATIVE_ROCK	FUNKY	ALTERNATIVE
Tópico 4	Tópico 5	Tópico 6
ALTERNATIVE_ROCK	COUNTRY	PROGRESSIVE_ROCK
ALTERNATIVE	MALE_VOCALISTS	METAL
ROCK	POP	PROGRESSIVE
AMERICAN	OLDIES	PROGRESSIVE_META
HARD_ROCK	SINGER_SONGWRITER	EPIC
Tópico 7	Tópico 8	Tópico 9
METAL	ROCK	NEW_WAVE
DEATH_METAL	HARDCORE	ROCK
THRASH_METAL	CHRISTIAN	BRITISH
HEAVY_METAL	EMO	POST_PUNK
MELODIC_DEATH_METAL	POST_HARDCORE	ALTERNATIVE
Tópico 10	Tópico 11	Tópico 12
EXPERIMENTAL	HIP_HOP	REGGAE
AMBIENT	RAP	INSTRUMENTAL
ELECTRONIC	DOWNTEMPO	WORLD
INSTRUMENTAL	CHILL	CHILLOUT
POST_ROCK	TRIP_HOP	NEW_AGE
Tópico 13		
CLASSIC_ROCK		
BLUES		
GUITAR		
HARD_ROCK		
ROCK_N_ROLL		

Tabela 4.2: Palavras mais significativas para os tópicos gerados na modelagem das músicas após o processamento.

4.3 Interpretação dos tópicos em uma base de *tags*

Este trabalho propõe uma representação que resume de forma compacta a informação contida nas anotações referentes às músicas em uma base de recomendação. Ela permite estudar como os usuários classificam músicas, e busca ser robusta à grande variabilidade com que estes atribuem informações relacionadas a uma música. Entendemos que um gênero musical, sendo uma classificação subjetiva e pessoal para cada usuário, é um conjunto de ideias e sentimentos que usuários possuem sobre músicas que se assemelham. Assim vemos o conjunto de *tags* como uma descrição destas ideias e sentimentos. Portanto, um conjunto de *tags*, agregado de diversos usuários, deve estar próximo de descrever o conceito de gênero musical.

Do ponto de vista de modelagem, avaliamos a distribuição sobre os tópicos como uma distribuição entre distintos gêneros musicais. Assim, interpretamos cada tópico como uma descrição de um gênero real, que é criado a partir da maneira como as *tags* são distribuídas e co-ocorrem entre as músicas na base de dados.

Utilizando essa interpretação, o número de tópicos deve ser escolhido com a consciência de que pode existir um número limitado de gêneros em uma base de dados. Assim, embora a escolha seja feita de forma heurística (como discutido na Subseção 4.7.2), bases de dados com músicas de um número maior de gêneros musicais idealmente se adequarão a modelos com mais tópicos, enquanto bases mais estritas irão se adequar a modelos mais simples.

Com o objetivo de criar um algoritmo de recomendação simples, gostaríamos de aplicar o mesmo método de recomendação em diferentes níveis de representação, como a recomendação de música por *query* (Seção 4.4), a recomendação de *playlists* (Seção 4.5) e a recomendação por *profiling* (Seção 4.6). Esses distintos escopos de modelagem proporcionam a possibilidade de criar um sistema de recomendações completamente direcionado à preferência de como o usuário deseja interagir com o recomendador. Músicas podem ser recomendadas sem uma *query* para o usuário, ou o usuário pode realizar uma *query* por músicas similares e por último deixar que o sistema continue gerando uma *playlist* automaticamente. As próximas subseções descreverão como podemos efetuar a modelagem nesses distintos níveis.

4.4 Modelagem de músicas

A modelagem no nível da música busca gerar uma representação númerica de cada música para utilização no método de recomendação baseado no cálculo de similaridade descrito na Seção 3.7. Essa representação consiste no vetor de mistura de tópicos $\vec{\theta}$ gerado pelo modelo LDA. A modelagem no nível da música nos permite

gerar recomendações a partir de *queries*, que podem ser realizadas da seguinte forma:

1. Um usuário seleciona uma música para a qual ele deseja obter músicas semelhantes ou uma lista de anotações que representam o que ele busca;

2. O sistema verifica se a música selecionada já faz parte do modelo através de algum indentificador, como seu nome;

   (a) Caso positivo: O vetor $\vec{\theta}$ de mistura de tópicos já foi calculado e o utilizamos;

   (b) Caso negativo: O vetor $\vec{\theta}$ ainda não foi calculado. Neste caso as *tags* são obtidas de um banco de *tags* (utilizando a API do serviço *Last.fm*, por exemplo) e a música é adicionada ao modelo, gerando assim o vetor necessário;

3. As distâncias entre a música de *query* e as demais músicas da base de dados são calculadas e a música de menor distância é apresentada ao usuário.

Como exemplo, apresentamos a música *Overdrive* da banda *Foo Fighters*. As *tags* na base de dados para a música são apresentadas na Tabela 4.3.

ROCK ALTERNATIVE_ROCK GRUNGE FOO_FIGHTERS ALTERNATIVE POST-GRUNGE HARD_ROCK AMERICAN COOL DAVE_GROHL INDIE_ROCK ALTERNATIVE_PUNK CATCHY FAST FOOS BLACKBERLIN MOJE_FOO_FIGHTERS FAVORITTER GOOD_MOOD_MUSIC MAARTS

Tabela 4.3: Tags da música Foo Fighters - Overdrive.

Ao modelarmos a base de dados utilizando o modelo LDA tal como apresentado na Seção 3.3, utilizando os hiperparâmetros com seus valores padrões como descrito na Seção 3.5, e modelando 13 tópicos com 2000 passos de estimação, obtivemos a distribuição apresentada na Figura 4.1. O número de tópicos foi definido através do estudo cuidadoso da base de dados e dos modelos gerados com diferentes valores, enquanto o número de passos foi definido de forma a agilizar o cálculo do modelo.

É possível ver que a distribuição é altamente concentrada nos tópicos 4, 7 e 12. A Tabela 4.4 mostra as 10 *tags* de maior probabilidade para esses tópicos. Os tópicos obtidos são consistentes com as *tags* da música, além de se ter conseguido inferir novas informações, como a *tag* de *MALE_VOCALIST* no tópico 12.

Ao analisarmos as 10 músicas mais similares segundo o critério discutido na Seção 3.7, obtivemos os resultados da Tabela 4.5. O resultado é consistente com a base de dados, uma vez que todas as músicas mais similares pertencem ao mesmo

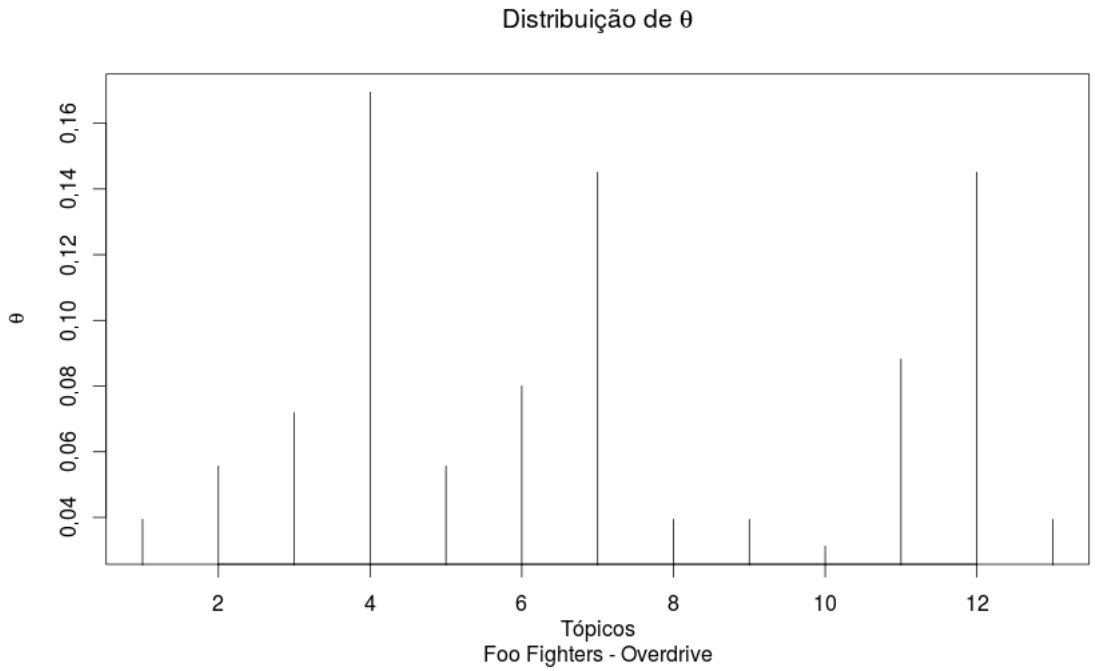

Figura 4.1: Distribuição da música *Foo Fighters - Overdrive*.

Tópico 4	Tópico 7	Tópico 12
HARD_ROCK	INDIE	POP
ROCK	ALTERNATIVE	MALE_VOCALISTS
CLASSIC_ROCK	ALTERNATIVE_ROCK	CATCHY
HEAVY_METAL	ROCK	LOVE
AMERICAN	INDIE_ROCK	POP_ROCK
METAL	BRITISH	PARTY
ROCK_N_ROLL	NEW_WAVE	LOVED
S_ROCK	GOOD	UPBEAT
USA	UK	HAPPY
GRUNGE	POST-PUNK	ENERGETIC

Tabela 4.4: Palavras nos tópicos mais representativos para a música *Foo Fighters - Overdrive*.

Música	Distância
Silverchair - The Door	0,0605251
The White Stripes - Blue Orchid	0,0665086
Weezer - Dope Nose	0,0722095
Candlebox - Breathe Me In	0,0889291
Juliette & The Licks - Shelter Your Needs	0,0933802
Kings Of Leon - Ragoo	0,0953399
Eagles Of Death Metal - Tight Pants	0,103348
The Rolling Stones - Start Me Up	0,103795
The Pharcyde - The Hustle	0,105938
Nirvana - Aneurysm	0,10615

Tabela 4.5: Top-10 recomendações para *Foo Fighters - Overdrive*.

grupo da música utilizada como *query*. Para melhor entender como as distribuições de θ para cada música se comparam com a música utilizada como *query*, podemos observar a Figura 4.2. Observe que as músicas com menor distância para a música original possuem distribuição de θ próxima à da música original. À medida que a distância aumenta, as distribuições começam a se diferenciar.

Fica claro pelo gráfico que as top-4 músicas recomendadas possuem picos em suas distribuições nos mesmos tópicos da música original. Para uma compreensão ainda maior de como o sistema gerou essas recomendações, podemos avaliar as *tags* de cada música recomendada. Estas se encontram na Tabela 4.6.

Observamos que 3 das 4 músicas possuem a anotação de *GRUNGE*, como a música original. Além disso, todas possuem anotações de *ROCK* e derivadas. O modelo LDA se concentra nessas co-ocorrências de palavras para gerar a distribuição θ, resultando não apenas em distribuições similares para músicas com *tags* similares, mas também para as que possuem *tags* apenas relacionadas, ainda que distintas.

No contexto de recomendação, definimos que a música recomendada pelo sistema será aquela cuja distribuição de θ apresenta a menor distância à da *query*. Escolhemos recomendar apenas a música mais similar como uma forma de tentar garantir uma boa recomendação para o usuário. Assim, no exemplo dado onde o usuário utiliza a música *Foo Fighters - Overdrive* como *query*, o sistema retornaria a música *Silverchair - The Door* como resposta.

Pode-se pensar em criar uma série de recomendações a partir de uma *query*, em que o sistema se auto-alimente de forma a criar uma série de recomendações que poderiam ser tocadas continuamente em forma de *playlist*. A próxima seção investiga tal estratégia para recomendações e busca refinamentos que melhorem a qualidade de *playlists* geradas pelo sistema.

Silverchair - The Door
GRUNGE ROCK ALTERNATIVE_ROCK ALTERNATIVE AUSTRALIAN POST-GRUNGE SILVERCHAIR THE_DOOR POST_GRUNGE FUN JUST_COOL ALLTIME_FAVE_SONGS COMPLETELY_IN_LOVE_WITH DAGS_FRIENDLY EMPERORCRYN V3DD3R
The White Stripes - Blue Orchid
LOVE_AT_FIRST_LISTEN *******_AWESOME POP LOVED LOVE_THIS_ONE SEXY COLORS METAL PROGRESSIVE_ROCK MALE_VOCALISTS PARTY PUNK_ROCK GRUNGE HEAVY MODERN_ROCK BLUE GARAGE_ROCK_REVIVAL LOUD RETRO RAW AWESOME_SONGS DARK
Weezer - Dope Nose
NOSTALGIA-INDUCING_MIDDLE_SCHOOL SONGS_I_NEVER_GET_TIRED_OF_LISTENING_TO LOVED_AT_FIRST_LISTEN OF_SPECIAL_NOTE THE_SHROOMS_AND_PURPLE_HAZE MELHOR_MUSICA_DO_MUNDO WHISKEY_RAMBLE FLYA_ALTERNATIVE TH_GRADE HIGH_SCHOOL ALTERNATIVE_PUNK ******** SING-ALONG *******_AWESOME POP_PUNK CALIFORNIA ENERGETIC POP_ROCK FAST S_ROCK SUMMER AMERICAN USA ELECTRONICA ELECTRONIC CANADIAN PERSONAL_FAVOURITES JAMMIN ED WEED
Candlebox - Breathe Me In
GRUNGE ROCK ALTERNATIVE_ROCK HARD_ROCK ALTERNATIVE CANDLEBOX MALE_VOCALISTS AITCH VIAJE RAINY_DAY ROCK_N_ROLL RELAX POP_ROCK INDIE_ROCK ALTERNATIVE_POP INDIE_POP POP INDIE

Tabela 4.6: *Tags* das top-4 recomendações, onde asterísticos são palavras de baixo calão omitidas.

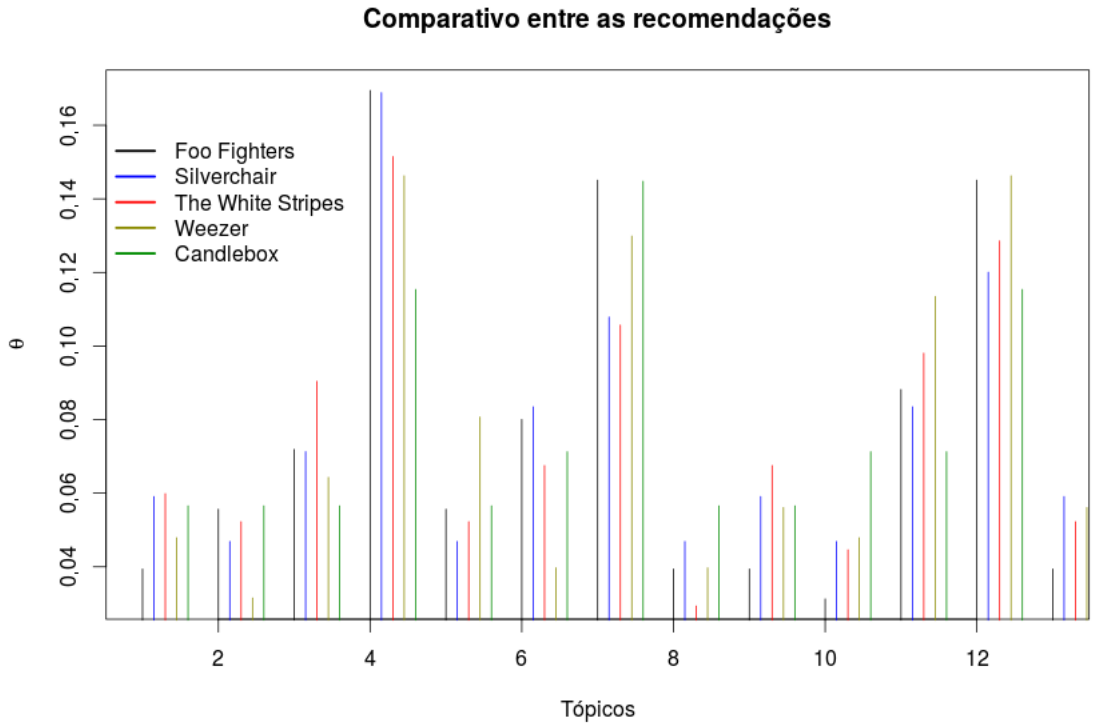

Figura 4.2: Distribuições de θ das top-4 recomendações para a música *Foo Fighters - Overdrive*.

4.5 Modelagem e criação de *playlists*

A primeira ideia intuitiva para geração de *playlists* é fazer o sistema recomendar M músicas a partir de uma música utilizada como alimentação. Dessa forma, a Tabela 4.5 mostra uma *playlist* de 10 músicas gerada a partir da música *Overdrive* da banda *Foo Fighters*. No entanto, esse método se mostra pouco flexível, reduzindo a possibilidade de o usuário descobrir novas músicas, já que o sistema irá investigar apenas músicas muito similares à música de *query*, e não proverá ao usuário uma forma simples de moldar a *playlist* à medida que é gerada.

Uma alternativa simples é permitir que o sistema se autoalimente de músicas, isto é, que após o usuário adicionar uma música como *query*, o sistema utilize sua recomendação da forma descrita na Seção 4.4 para gerar a próxima recomendação. Caso a recomendação já tenha sido tocada, o sistema escolhe a próxima possível música a ser recomendada. Esse método permite a implementação simples de uma interface para o usuário moldar sua *playlist*, de forma que caso o sistema detecte a insatisfação do usuário (por que o usuário avaliou negativamente ou não escutou completamente a música) ele possa se adaptar e continuar a série de recomendações facilmente buscando músicas mais próximas à original. O diagrama na Figura 4.3 mostra como o sistema se autoalimenta.

Para mostrar esse sistema em funcionamento, foi criada uma *playlist* de 10

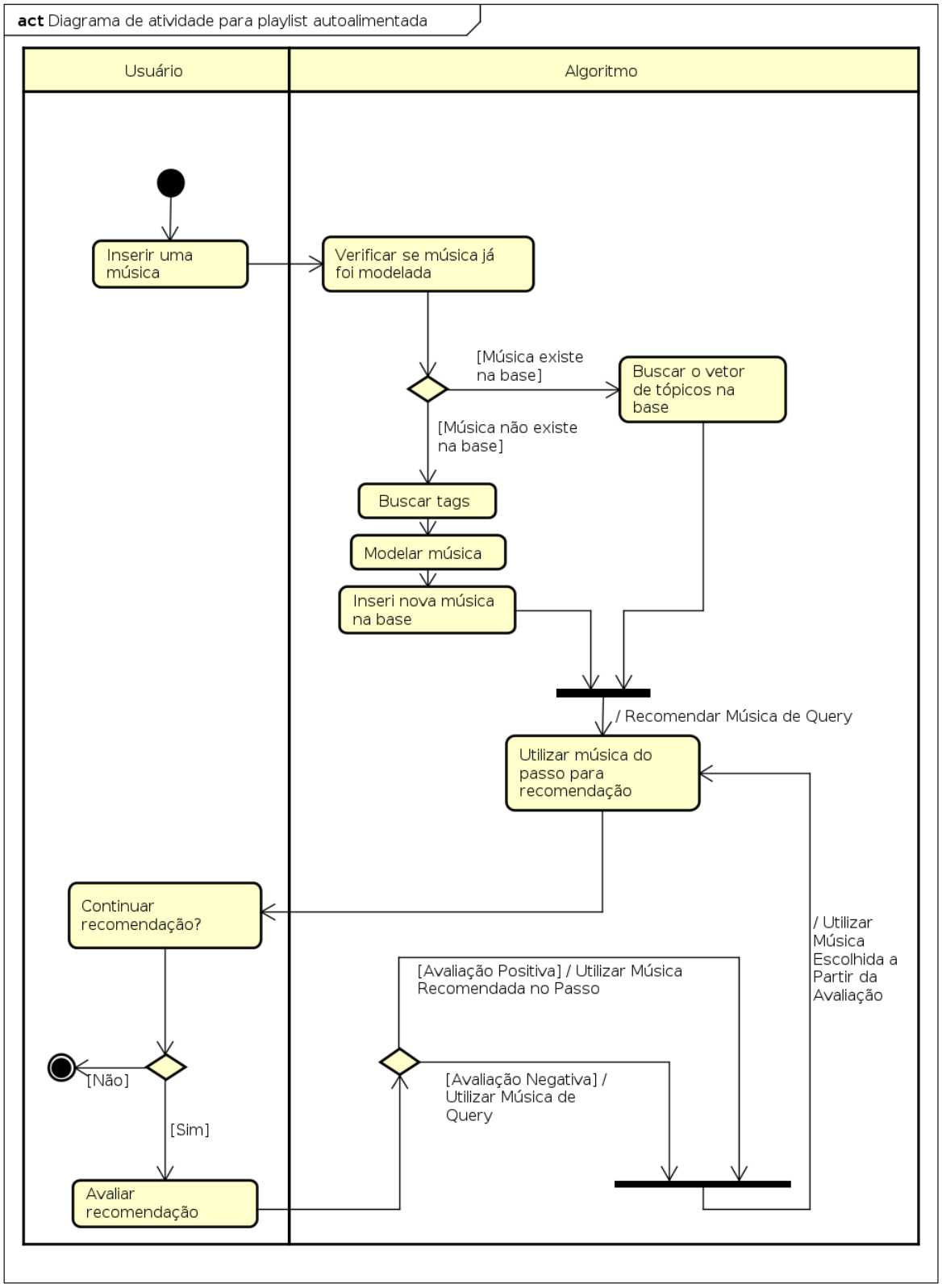

Figura 4.3: Fluxo de dados para recomendação de músicas por autoalimentação.

músicas a partir da música de *query Big Ten-Inch Record - Aerosmith*, apresentada na Tabela 4.7.

Música	Distância para o anterior	Distância para *query*
Bait A Hook - Justin Moore	0.02805	0.02805
Snowblind - Styx	0.0480	0.3898
Mean To Me - Brett Eldredge	0.0264	0.3285
A Jukebox With A Country Song - Douge Stone	0.0490	0.2267
Came To The Rescue - Hillsong United	0.0378	0.3431
Rhythm From A Red Car - Hardline	0.0369	0.4397
On Sight - Kanye West	0.0426	0.4420
Body Talk - Ratt	0.0541	0.4805
You Give Love A Bad Name - Bon Jovi	0.0640	0.379698

Tabela 4.7: Top-10 recomendações por autoalimentação para a música *Big Ten-Inch Record - Aerosmith*.

Os valores na Tabela 4.7 mostram o comportamento da distância entre as músicas no decorrer do processo de recomendação. Uma desvantagem de se utilizar autoalimentação para geração de *playlist* é o *drift* que a distribuição θ sofre a cada nova música. Embora o sistema recomende músicas com distribuições semelhantes, o uso da música anterior como base para a recomendação seguinte pode introduzir discrepâncias em relação a recomendações que seriam feitas utilizando-se apenas a música original. No exemplo apresentado na Tabela 4.7, observa-se que existe uma discrêpancia ao se recomendar *Kanye West*, do gênero *hip-hop*, quando a música de alimentação foi *Aerosmith*, do gênero *rock*. No entanto, é importante notar que neste caso o sistema foi capaz de retornar a músicas mais relacionadas rapidamente, recomendando em seguida músicas do mesmo gênero da música original. Uma forma de reduzir o *drift* gerado por esse método de geração de *playlists* é reintroduzir recomendações a partir da música de *query* de tempos em tempos no sistema, em uma tentativa de retornar músicas com alta similaridade em relação à original.

Um ponto importante é que os métodos apresentados acima irão gerar a mesma *playlist* todas as vezes em que a mesma música de *query* for utilizada, se a base de dados não tiver sofrido alteração. Para diversificar as músicas apresentadas pelo usuário, apresentamos uma terceira solução para geração das *playlists*. Basta tratar as distâncias entre músicas como distâncias entre vértices de um grafo completo, no qual a probabilidade de transição entre vértices é proporcional à distância entre as músicas. Sendo assim, a *playlist* pode ser gerada a partir de um passeio aleatório por esse grafo. Assim, definimos uma matriz de adjacência M composta de elementos $m_{i,j}$ definidos como na fórmula 4.1. A Tabela 4.8 apresenta uma *playlist* gerada por esse método e a Figura 4.4 apresenta o fluxograma do método.

$$m_{i,j} = \frac{1}{D(i,j)}, \tag{4.1}$$

onde D é uma função de distância.

Uma vez que essa abordagem permite que qualquer música seja recomendada com alguma probabilidade, também incorporamos um fator de correção que dê preferencia a músicas mais semelhantes à música original. Desta forma qualquer, *drift* que possa ocorrer durante a execução da *playlist* deve ser corrigido rapidamente, retornando a recomendação de músicas mais semelhantes à música utilizada como entrada. A fórmula com o fator de correção é

$$m_{i,j} = \frac{1}{D(i,j)D(j,q)}, \tag{4.2}$$

onde q é a música utilizada como *query*. A Tabela 4.8 apresenta uma *playlist* gerada pelo método *random walk* com a medida de distância 4.2.

Música	Distância para o anterior	Distância para *query*
Everclear - Like a California King	-	0,2055
Weezer - Tired of Sex	0,0910	0,2100
A-Ha - Shadow Side	0,2089	0,1530
De/Vision - Nine Lifes	0,06960	0,1450
Japandroids - Crazy/Forever	0,2457	0,2461
Black Dice - Smilling Off	0,1004	0,2762
Real Estate - Fake Blues	0,0712	0,2659
Ronee Blakley - Attachment	0,0544	0,2784
Beach House - Take Care	0,1011	0,2743
Final Fantasy - The Arctical Circle	0,0713	0,2446

Tabela 4.8: Top-10 recomendações por *random walk* utilizando distância 4.2 para música *Buckethead - Lone Sal Bug*.

A comparação entre os métodos de geração de *playlists* será aprofundada na Seção 4.7.4.

4.6 Modelagem de usuários

Assim como para *playlists*, a criação de perfis de usuários nos permite algumas diferentes abordagens. Aqui apresentaremos duas formas de encarar o problema de forma simples e funcional, que adicionalmente apresenta baixo custo computacional.

O perfil de um usuário deve consistir de uma série de anotações que definem o seu gosto musical. Essas anotações devem em seguida ser transformadas em um

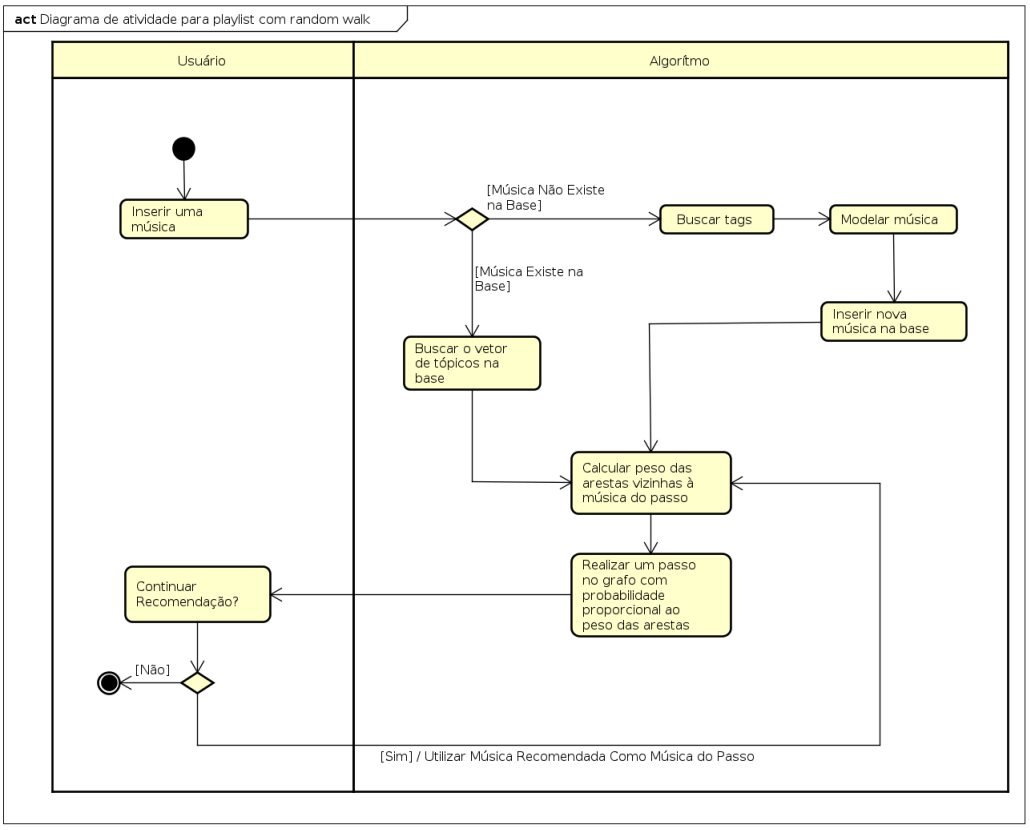

Figura 4.4: Fluxo de dados para recomendação de músicas por *random walk*.

vetor sobre tópicos no modelo em que as músicas existentes no sistemas foram modeladas. Desta forma, o vetor sobre tópicos do perfil do usuário representa uma descrição matemática de seu gosto musical, perimitindo realizar recomendações de forma semelhante à que realizamos na seção anterior, cumprindo o objetivo de utilizar o mesmo algoritmo de recomendação para diferentes níveis de representação. Neste sistema, o perfil de usuário seria utilizado como entrada para o algoritmo de recomendação e o sistema deve responder com uma lista de recomendações baseadas no perfil do usuário. Sendo assim, o problema de recomendação para perfis de usuários passa a ser como adquirir anotações para definir o seu perfil.

Uma abordagem manual para a geração de perfis é requisitar ao usuário informações do que ele deseja ouvir. Essas informações podem incluir gêneros, sentimentos, tempo e palavras relacionadas. Essa abordagem é bem simples para o sistema, uma vez que ele não precisa inferir em nada sobre o que o usuário deseja ouvir. Porém não é ideal do ponto de vista do usuário, que terá o trabalho de construir o próprio perfil. Um sistema ideal seria aquele capaz de entender o gosto do usuário e construir um perfil sem nunca explicitamente pedir informações.

Um usuário deve ser definido por anotações relacionadas às músicas de que ele gosta. Assim sendo, é natural buscar um método que seja capaz de entender o gosto do usuário a partir das músicas que ele ouve. Uma forma simplista e de baixo custo

para realizar essa operação é aproveitar a maneira com que o usuário interage com o sistema, capturando informações de como ele avalia músicas, quais músicas são ouvidas completamente e quais são ouvidas parcialmente. Assim, o perfil pode ser criado simplesmente a partir de anotações existentes nessas músicas, procurando *tags* frequentes entre elas.

No entanto, diferentemente de músicas, o perfil de um usuário pode variar com o tempo. Portanto, é necessário adicionar ao sistema uma forma de se alterar com o tempo, podendo assim seguir as variações das preferências do usuário. Tendo em vista o objetivo de simplicidade do sistema, propomos o seguinte algoritmo de geração e manutenção de perfis de usuários:

Algoritmo 3: Algoritmo de geração de perfis.

enquanto *usuário está no sistema* **faça**
  //Etapa de adição de *tags*.
  Seja m a música que o usuário está ouvindo no momento ;
  se m *foi avaliada negativamente* **então**
    Ignore m ;
  fim
  caso contrário
    Adicione as *tags* de m ao perfil do usuário ;
  fim
  //Etapa de remoção de *tags*.
  Remova as *tags* da k-ésima última música ouvida ;
fim

Observe que as recomendações iniciais irão depender de um perfil de usuário pré-existente. Uma forma de contornar esta limitação é utilizar um perfil inicial padrão para o usuário. Propomos as seguintes opções para geração do perfil inicial:

- Utilizar um perfil contendo as *tags* mais frequentes no sistema.
- Utilizar um perfil composto das *tags* mais comuns encontradas em perfis do sistema.
- Questionar o usuário sobre o tipo de música por que ele se interessa.

A geração de perfis proposta aqui é apenas uma sugestão de implementação de baixo custo, no entanto qualquer alternativa de criação de perfil pode ser utilizada. O ponto importante a ser observado é que ser capaz de gerar uma série de anotações para representar o perfil do usuário coloca-o no mesmo nível de representação de músicas e *playlists*, de forma que somos capazes de recomendar músicas e avaliar estas recomendações utilizando as mesmas estratégias. Embora uma avaliação ideal de um algoritmo de geração de perfis envolvesse um teste A/B de longo prazo, o

alto custo destes dificulta sua realização. Portanto, iremos supor que os resultados obtidos para geração de *playlists* na Subseção 4.7.4 são também representativos para recomendações baseadas em perfis.

4.7 Experimentos de recomendação

4.7.1 Dados

Os dados para os testes de qualidade de recomendação foram retirados do banco de dados *Million Song Dataset*. As músicas foram filtradas para remoção de músicas sem *tags*. Em seguida, as anotações das músicas restantes foram processadas como descrito na Subseção 4.7.2. Após o procedimento, o banco utilizado totaliza 173555 músicas únicas.

4.7.2 Escolha do número de tópicos e modelagem

A escolha de tópicos em um modelo LDA pode ser feita sob duas abordagens:

1. Maximização da função de verossimilhança: Essa abordagem consiste em maximizar a fórmula 3.7 (em [4] essa maximização é realizada através da função de *perplexity*, que é comumente utilizada em problemas de recuperação de informação em texto).

2. Priorização da consistência lógica dos tópicos e heurística: Essa abordagem também se mostra importante, como discutido em [36] (no entanto, o mesmo trabalho também propõe um método de encontrar o número natural de tópicos através de fatoração matricial).

Com o intuito de criar um método de redução de dimensão para a representação em *tags* de uma música que o presente trabalho se propõe a encontrar, utilizamos a segunda abordagem, de forma a manter também a capacidade de explicação dos tópicos criados.

Para os testes realizados, utilizamos um modelo LDA com 20 tópicos. Esse número foi encontrado após testes objetivando criar tópicos com informações consistentes, evitando gerar tópicos *mistos* (aqueles que agregam informações de gêneros distintos). Outros valores testados para o número de tópico foram: 5, 10, 12, 15, 35 e 50. Os hiperparâmetros foram escolhidos como descrito na Seção 3.5. A lista das 5 palavras mais significantes nos tópicos modelados se encontra na Tabela 4.9.

Tópico 1	**Tópico 2**	**Tópico 3**
INDIE	JAZZ	INDIE
INDIE_ROCK	FUNK	INDIE_ROCK
ALTERNATIVE	SOUL	INDIE_POP
BRITISH	DRJAZZMRFUNKMUSIC	FOLK
ALTERNATIVE _ROCK	INSTRUMENTAL	ALTERNATIVE
Tópico 4	**Tópico 5**	**Tópico 6**
ALTERNATIVE _ROCK	COUNTRY	PROGRESSIVE _ROCK
ALTERNATIVE	MALE_VOCALISTS	METAL
ROCK	POP	PROGRESSIVE
AMERICAN	OLDIES	PROGRESSIVE _METAL
HARD_ROCK	SINGER_SONGWRITER	HEAVY_METAL
Tópico 7	**Tópico 8**	**Tópico 9**
METAL	ROCK	NEW_WAVE
DEATH_METAL	HARDCORE	ROCK
THRASH_METAL	CHRISTIAN	BRITISH
HEAVY_METAL	ALTERNATIVE	POST_PUNK
MELODIC_DEATH _METAL	EMO	ALTERNATIVE
Tópico 10	**Tópico 11**	**Tópico 12**
EXPERIMENTAL	HIP_HOP	REGGAE
AMBIENT	RAP	INSTRUMENTAL
ELETRONIC	DOWNTEMPO	WORLD
PSYCHEDELIC	CHILLOUT	CHILLOUT
INSTRUMENTAL	CHILL	NEW_AGE
Tópico 13	**Tópico 14**	**Tópico 15**
CLASSIC_ROCK	ROCK	PUNK
BLUES	POP	ROCK
GUITAR	GERMAN	PUNK_ROCK
HARD_ROCK	LATIN	ALTERNATIVE
ROCK_N_ROLL	SPANISH	COVER
Tópico 16	**Tópico 17**	**Tópico 18**
FUN	FEMALE_VOCALISTS	BEAUTIFUL
CATCHY	SINGER_SONGWRITER	MELLOW
HAPPY	POP	SAD
UPBEAT	FEMALE	LOVE
POP	FOLK	MELANCHOLY
Tópico 19	**Tópico 20**	
RNB	ELETRONIC	
SOUL	DANCE	
POP	ELETRONICA	
SEXY	ELECTRO	
FEMALE_VOCALISTS	TECHNO	

Tabela 4.9: Top-5 palavras em cada tópico.

4.7.3 Escolha de função de distância

Até este ponto utilizamos a distância euclidiana para a geração de recomendações, como por exemplo nas Tabelas 4.8 e 4.7, no entanto qualquer função definida em $\mathbb{R}^n$ pode ser utilizada. Para os testes realizados na Subseção 4.7.4 consideramos duas medidas de distância, descritas a seguir:

- Distância euclidiana: A distância mais comumente utilizada para comparar dois pontos em um espaço euclidiano. Representa o comprimento do segmento de reta ligando estes pontos. Sua fórmula pode ser encontrada em 4.3.

$$D(\vec{p}, \vec{q}) = \sqrt{\sum_{i=1}^{n} (p_i - q_i)^2} \tag{4.3}$$

- Divergência de Kullback-Leiber (KLD): Trata-se de uma medida da diferença entre as distribuições de probabilidade P e Q. Sua fórmula pode ser encontrada em 4.4.

$$D(P||Q) = \sum_{i=1}^{n} P(i) \log \frac{P(i)}{Q(i)} \tag{4.4}$$

  Note que a medida não é simétrica, portanto é importante interpretar corretamente o resultado obtido pela medida. A análise do ponto de vista de teoria da informação interpreta o resultado como a quantidade de informação perdida ao se utilizar a distribuição Q para estimar P. Essa medida parece se mostrar ideal para o intuito do presente trabalho, uma vez que podemos considerar a distribuição de P como a distribuição das proporções de tópico $\vec{\theta}$ para a música utilizada como *query*.

Na seção seguinte demonstraremos os resultados obtidos na geração de *playlists* utilizando as medidas apresentadas.

4.7.4 Qualidade das recomendações

Descrevemos a qualidade da recomendação como a proporção de músicas do mesmo gênero que foram recomendadas ao usuário, como descrito na fórmula 4.5. Essa métrica foi escolhida devido à dificuldade de se descrever boa métrica para qualidade de recomendação sem realizar experimentos do tipo *A/B*, como descrito em 2.4. Optamos, então, por utilizar uma interpretação de como seria uma *playlist* ideal e comparar os resultados obtidos contra essa idealização. Este método está relacionado à homogeneidade das *playlists* geradas de acordo com o gênero das recomendações.

Trabalhos que utilizam métodos similares utilizando consistência de gêneros em uma *playlist* podem ser encontrados em [22], [23], [20] e [24].

$$S(F) = \frac{\sum_{i=1}^{N} S(F(i))}{N}, \tag{4.5}$$

onde F é um sistema de recomendação qualquer, N é o total de músicas na base de dados, $S(F(i))$ representa o escore da recomendação gerada para a i-ésima música. $S(F(i))$ pode ser encontrado na fórmula 4.6.

$$S(F(i)) = \frac{\sum_{j=1}^{|F(i)|} \omega(i,j)}{|F(i)|}, \tag{4.6}$$

onde $|F(i)|$ representa o tamanho da *playlist* gerada por F e $\omega(i,j)$ é uma função que resulta em 1 caso o gênero da música i seja o mesmo gênero da música j e 0 em caso contrário. Fica claro que o valor de $S(F(i))$ está definido entre 0 e 1, simbolizando a proporção de músicas do mesmo gênero existentes na lista de recomendação.

Entendemos que uma *playlist* ideal não deve ser completamente homogênea no gênero das músicas recomendadas. Para surpreender o usuário desejamos uma parcela de músicas que sejam de gêneros diferentes e possam apresentar novidades ao usuário, de forma que a recomendação de músicas exclusivamente do mesmo gênero pode ser considerado um resultado negativo. Para a análise de qualidade da recomendação, isso implica dizer que a medida apresentada em 4.5 não pode ser muito próxima a 1. Ao mesmo tempo, caso o sistema recomende músicas de gêneros muito diversos o usuário pode sentir-se insatisfeito com a inconsistência entre o que ele deseja ouvir e os resultados obtidos, de forma que não desejamos que a medida esteja abaixo de $0,5$, que indicaria que menos da metade das músicas inseridas na *playlist* são do mesmo gênero da música de *query*.

Para calcular a qualidade de uma recomendação i definimos a função de escore a seguir, onde $\omega_{(i)}$ representa a quantidade de músicas do mesmo gênero encontradas na recomendação i e M representa o número de recomendações. Note, no entanto, que definir o que são músicas do mesmo gênero é extremamente subjetivo, e ainda se complica mais devido à existência de correlação entre gêneros que são considerados diferentes mas possuem muitas características em comum. O desenvolvimento de uma medida de qualidade de recomendação a partir de gêneros que considere essas correlações requer um trabalho de pesquisa muito extenso, envolvendo pesquisas sociais e de opinião, e se mostra fora do escopo deste trabalho. Portanto, optou-se por tratar os gêneros como não correlacionados.

A seguir, na Tabela 4.10 apresentamos os resultados obtidos utilizando a medida apresentada na equação 4.5 para diferentes medidas de distância e distintos métodos de geração de *playlists* com 10 recomendações.

	playlist simples	*playlist* auto alimentada	*random walk* sem ponderação	*random walk* com ponderação
Euclidiana	0,6478	0,6508	0,5775	0,6329
KLD	0,6538	0,6520	0,597	0,6384

Tabela 4.10: Escore das diferentes estratégias de geração de *playlists*.

Os resultados obtidos estão próximos de uma divisão de 2/3 de músicas do mesmo gênero da música de *query* e 1/3 de músicas de outros gêneros, indicando uma têndencia do sistema a recomendar músicas do mesmo gênero, mas mantendo um fator de músicas de gêneros distintos, podendo ainda surpreender o usuário. Note também que a geração de *playlists* por *random walk* utilizando a ponderação exibida na fórmula 4.2 causa uma baixa perda no escore, enquanto é capaz de gerar *playlists* aleatórias, diferentemente dos outros métodos. Em particular, a utilização da ponderação causou um ganho de precisão de 0,0554 em relação à *random walk* comum, indicando que a ponderação está cumprindo o papel proposto de redução de *drift*. A medida de distância KLD indicou valores similares aos da distância euclidiana; no entanto, devido à sua relação direta com distribuições de probabilidade (sendo uma medida da perda de informação ao utilizarmos a distribuição de uma música a ser recomendada comparada à música de *query*), acreditamos ser a medida mais adequada para ser utilizada na prática, facilitando a interpretação das distâncias encontradas.

Como forma de gerar uma análise mais profunda da capacidade de geração de recomendações do sistema, realizaremos os testes em condições mais restritas. Alimentaremos o sistema com músicas de apenas um gênero músical, tentando avaliar se a qualidade da recomendação se mantém consistente com a da base total. Em caso positivo, tal resultado indica que o sistema é capaz de realizar recomendações mais refinadas, que exijam maior capacidade de diferenciação entre músicas. Para realizar esse experimento, modelamos apenas as músicas classificadas como *pop* pela base de dados e repetimos o experimento acima, utilizando a medida de qualidade da equação 4.5 e utilizando a medida de distância KLD.

Para o cálculo do escore, iremos avaliar a consistência do estilo músical das músicas incluidas na *playlist*. O estilo músical é outra classificação contida na extensão da base de dados MSD [35]. Essa classificação é mais específica do que a classificação de gêneros, e contém misturas de gêneros, como *pop-rock*, *post-punk*, etc. A classificação por estilos não foi utilizada anteriormente porque cada estilo, sendo mais complexo que um gênero, muitas vezes indica uma correlação existente entre gêneros distintos; e como foi discutido anteriormente, o estudo de tal correlação se encontra fora do escopo deste trabalho. No entanto, ao restringirmos as músicas utilizadas para avaliação a itens de apenas um gênero, acreditamos que a

classificação por estilos seja um indicador confiável para medir a consistência das recomendações.

Para a execução dos testes, outro modelo LDA foi calculado utilizando-se apenas as músicas do gênero *pop*. A redução da base de dados causa a necessidade da reavaliação dos parâmetros do modelo. Desta vez, o modelo foi calculado utilizando 8 tópicos, vizando manter a consistência lógica dos tópicos, uma vez que a base de dados foi reduzida. Os hiperparâmetros α e β foram calculados como descrito na Seção 3.5.

A Tabela 4.11 mostra os resultados obtidos sob tais condições, utilizando *playlists* de 10 itens.

	playlist simples	*playlist* auto alimentada	*random walk* sem ponderação	*random walk* com ponderação
Euclidiana	0,6738	0,6811	0,5519	0,6377
KLD	0,6734	0,6817	0,6068	0,6629

Tabela 4.11: Escore de homogeneidade de estilos das diferentes estratégias de geração de *playlist* com restrição de gêneros.

Os resultados encontrados na base de dados restrita são coerentes com os resultados da base de dados completa, indicando que o sistema é capaz de se manter consistente mesmo quando utilizado em condições que requerem maior especialização para diferenciação entre músicas. Concluímos, portanto, que o sistema cumpre os objetivos propostos de gerar recomendações consistentes com a música utilizada como *query* (ou perfil de usuário, como apresentado na Seção 4.6) e se mantém dentro da faixa entre 0,5 e 1 de músicas do mesmo gênero em uma *playlist* gerada.

Na próxima seção realizaremos um estudo mais detalhado de como o sistema se compara a sistemas reais, utilizando as classificações apresentadas na Seção 2.3.

4.7.5 Análise de dificuldades do sistema

A Seção 2.3 apresenta problemas comuns em sistemas de recomendação, e a Tabela 2.2 mostra como diferentes sistemas de recomendação apresentam estes problemas. A Tabela 4.12 busca caracterizar o sistema de recomendação proposto neste trabalho segundo os mesmos critérios.

- **Problema de Esparsidade (SA) :** Músicas que possuem mais anotações tendem a possuir um vetor de distribuição θ mais especializado para descrevê-la, tendendo a gerar melhores recomendações. As propostas apresentadas do Capítulo 5.1 podem ser utilizadas também para aumentar o número de *tags* em músicas que possuem poucas.

Método	SP	CSP	GSP	PBP	NEP	PE	SA
Recomendação por LDA	A	A	A	NA	NA	NA	3

Tabela 4.12: Dificuldades do método de recomendação por LDA, onde NA significa *"Não afetado"* e A, *"Afetado"*. A coluna **SA** está dividida em uma escala onde 1 = *"Fraco"*, 2 = *"Médio"* e 3 = *"Bom"*.

- **Problema de *Cold-Start* (CSP):** Para o funcionamento do algoritmo de recomendação é necessário que as músicas possuam anotações. Músicas sem anotações não podem ser utilizadas diretamente pelo sistema. No entanto, no Capítulo 5 propomos formas de contornar esse problema.

- **Problema de Ovelha Cinza (GSP):** A dificuldade em criar boas recomendações para usuários ovelha cinza está na inconsistência em seu gosto musical. O sistema de recomendação baseado em LDA não é capaz de contornar essa dificuldade.

- **Problema de Polarização de Popularidade (PBP):** A mudança de representação de *tags* para vetores de distribuição resultante do modelo LDA não é afetada pela popularidade de itens, tratando-os igualmente.

- **Problema de Falta de Explicação (NEP):** A fácil interpretação do vetor de tópicos e da distribuição de termos em tópicos resulta em um grande poder de explicação para o sistema de recomendação baseado em LDA.

- **Efeito de Portfólio (PF):** Uma vez que o modelo LDA não considera a origem das informações, o efeito de portfólio não aparece no sistema de recomendações.

- **Escalabilidade (SA):** Após o custo computacional inicial de treino do modelo, cada inserção de novas músicas possui baixo custo, uma vez que novas músicas podem ser modeladas facilmente. O custo da recomendação é diretamente relacionado ao tamanho da base de dados, no entanto estratégias de *caching* das distâncias já pré-computadas pode reduzir o custo computacional consideravelmente. Da mesma forma, a redução de dimensionalidade que o modelo LDA oferece reduz o espaço necessário para armazenar cada item da base de dados.

O sistema de recomendação apresentado no presente trabalho possui características similares às de sistemas baseados puramente em *tags*, com a vantagem da redução de custo relacionado com o menor espaço de armazenamento que a redução de dimensionalidade oferece e a simplificação da criação de algoritmos de

recomendação, uma vez que é mais simples trabalhar com vetores numéricos de dimensão definida do que vetores de palavras sem dimensão fixa.

Capítulo 5

Recomendação com itens sem *tags*

Para o bom funcionamento do sistema de recomendação, novas músicas devem ser continuamente adicionadas à base para mantê-lo atualizado. No entanto, novas músicas ainda não possuem informações de *tag*, e portanto não podem ser modeladas pelos métodos já discutidos (o problema de *cold start*, como discutido na Seção 2.3). Em casos em que uma música na base de dados não possui *tags* (como ocorre com as músicas recém adicionadas à base de dados que não foram anotadas por usuários) devemos encontrar uma forma de adicioná-la ao sistema, para que ela possa ser recomendada. Uma vez que músicas sem *tags* começam a ser recomendadas, espera-se que usuários comecem a anotá-las, e com isso elas passam a ser tratadas pelo processo usual de recomendação.

Outra alternativa para lidar com o caso de músicas com poucas *tags* é encontrada em [37], onde o próprio modelo é utilizado para recomendar novas *tags*. No entanto, neste trabalho escolhemos utilizar informações dos sinais de áudio para integrar os casos sem ou com poucas *tags* ao sistema.

Três abordagens são exploradas. A primeira envolve utilização de características de áudio para estimação de *tags* a partir de músicas semelhantes (Seção 5.1). Em seguida, estudamos uma forma de inserir músicas sem informação de *tags* nas *playlists* geradas utilizando apenas informações de sinal (Seção 5.2). Por fim propõe-se o uso de um classificador de gênero externo para inserir apenas *tags* de gênero, de forma a possibilitar início da utilização das músicas no sistema (Seção 5.3).

5.1 Estimação de *tags* a partir de *features* de áudio

O desenvolvimento de um algoritmo para estimação de *tags* para músicas sem anotações é um trabalho relacionado a classificadores *multi-label*. Um trabalho neste campo utilizando *features* de áudio pode ser encontrado em [38], onde o autor as

utiliza para estimação de *tags*. No entanto, o objetivo do presente trabalho é o desenvolvimento de um sistema de recomendação baseado em *tags* geradas por usuários. Portanto, o interesse na geração de *tags* deixa de ser o objetivo principal e se torna apenas uma ferramenta para que músicas que não possuem anotações possam ser inseridas no sistema até que usuários complementem as suas *tags*. No decorrer desta seção, propomos algoritmos para estimação de *tags* e apresentamos parametrizações e os resultados de precisão obtidos.

O algoritmo apresentado a seguir para esitmação de *tags*, embora simples, cumpre o objetivo proposto de inserir músicas sem informações no sistema. O Algoritmo 4 é uma proposta de implementação, onde:

Algoritmo 4: Algoritmo de estimação de *tags*.

para cada *música* $k \in K_-$ **faça**
selecione de K_+ as m músicas mais próximas de k utilizando uma função de distância F. Adicione no conjunto C ;
selecione as n *tags* que mais se repetem em C e adicone-as em k;
fim

K_- é o conjunto de músicas sem *tags*;

K_+ é o conjunto de músicas com *tags*;

F é uma função de distância que utiliza *features* de áudio;

n e m são parâmetros do algoritmo.

A seleção das *features* de áudio e da função F a serem utilizadas é crucial para que o algoritmo apresente bons resultados. A base de dados disponibiliza *features* de áudio calculadas pela plataforma *echoNest*. Essas *features*, de acordo com a documentação da base de dados, são similares aos *Mel-frequency cepstrum coefficients* (MFCCs); no entanto, devido à falta de documentação sobre como tais *features* foram calculadas exatamente e por conseguinte como interpretá-las, escolheu-se utilizar os próprios MFCCs como *features*, já que se encontram disponibilizados para um subconjunto de aproximadamente um décimo da base de dados [35]: os 13 primeiros coeficientes foram calculados a partir de cortes contendo entre 30 e 60 segundos de música; e as médias e desvios padrão dos coeficientes foram calculados e disponibilizados, totalizando 26 *features* para cada música.

Para a medição de qualidade dos testes escolhemos a medida de *one-error*, utilizada em problemas *multi-label*, [39], que avalia quantas vezes o *label* (nesse caso, a *tag*) de maior *rank* estimado não se encontra no conjunto de *labels* corretos. A qualidade das recomendações é considerada perfeita caso a medida seja 0. Quanto menor o valor da medida, melhor a qualidade das recomendações. A medida se encontra na fórmula 5.1.

$$\text{one-error(f)} = \frac{1}{p}\sum_{i=1}^{p}\langle[\arg\max_{y\in Y} f(x_i, y)] \notin Y_{x_i}\rangle, \tag{5.1}$$

onde:

f é o método de estimação em uso;

p é o tamanho do conjunto de testes;

x_i é um item do conjunto de testes;

Y é o conjunto de *labels*;

Y_{x_i} é o conjunto de *labels* reais para o item x_i;

$\langle\pi\rangle$ é tal que, para qualquer predicado π , $\langle\pi\rangle$ valha 1 caso π seja verdadeiro e 0 em caso contrário. Note que para casos em que os itens possuem apenas um *label*, a medida é idêntica ao erro de classificação comum.

A função de distância utilizada é uma simples distância euclidiana, que já mostrou fornecer bons resultados para o cálculo de similaridade entre músicas utilizando MFCC em [40]. Uma vez que a medida de precisão *one-error* avalia apenas a *tag* de maior *rank*, o parâmetro n não irá alterar o valor da precisão. A Tabela 5.1 mostra o *one-error* da recomendação de tags com diferentes valores de m.

	$m = 1$	$m = 5$	$m = 15$	$m = 30$
one-error	0,9605	0,7899	0,7169	0,6985

Tabela 5.1: *One-error* do algoritmo 4 com diferentes parâmetros.

O uso de todas as *features* disponíveis pode acarretar em perda na qualidade do algoritmo. Portanto, visando a melhorar os resultados, desejamos encontrar a combinação de *features* que aumente a precisão do algoritmo. No entanto, uma busca exaustiva de todas as $2^{26} - 1$ combinações seria muito custosa, e por esse motivo aplicamos um algoritmo de otimização natural baseado em fenômenos físicos conhecido como *Simulated Anealling* (SA) [41]. Este método é um método probabilístico ideal para casos em que temos um campo de busca muito extenso e complexo de ser avaliado, e desejamos encontrar algum máximo local, mesmo que este não seja o máximo global, em um tempo limitado. O pseudocódigo para o presente trabalho

pode ser encontrado no Algoritmo 5, onde:

Algoritmo 5: Algoritmo de estimação de *tags*.

selecione um vetor $v \in V$ aleatoriamente ;
$v_b := v$;
$k := 0$;
$b := 0$;
$i := 0$;
enquanto $k < K_{max}$ **faça**
  $m := M(v)$ /*aplique uma mutação em v */ ;
  $i := i + 1$;
  se $S(v) < S(m)$ **então**
    $v := m$;
  fim
  caso contrário
    selecione um número aleatório uniformemente entre 0 e 1 e atribua a u;
    $t := T(S(v) - S(m), i)$;
    se $t > u$ **então**
      $v := m$;
    fim
  fim
  se $S(V) > b$ **então**
    $b := S(v)$;
    $v_b := v$;
  fim
fim
retorne v_b ;

V é o conjunto de todas as possíveis combinações de *features*;

$M(\cdot)$ é uma função de mutação aleatória;

$S(\cdot)$ é uma função de precisão;

$T(\cdot, \cdot)$ é uma função que busca simular o resfriamento que é utilizado no processo natural de *simulated annealing*;

$K_{\max}$ é o número máximo de iterações previamente selecionado;

b e v_b são variáveis auxiliares para acompanhar o melhor resultado encontrado;

V é um vetor tal que $v[i] = 1$ se a *feature* 1 está sendo utilizada, e 0 em caso contrário.

Ao término da execução, o vetor v_b nos apresenta um valor de máximo no campo de busca V. A função de mutação M utilizada neste trabalho altera um valor do vetor v de forma uniformemente aleatória. Caso o valor antes da mutação seja 1,

seu valor após a mutação será 0, e caso o valor seja 0 seu resultado será 1. A função de avaliação S utilizada é a apresentada na equação 5.2, e a função de resfriamento é a função apresentada na equação 5.3.

$$S(v) = 1 - E(f), \tag{5.2}$$

onde E é a função *one-error*, apresentada na equação 5.1

$$T(\Delta v, i) = \exp \frac{\Delta v}{10 * 0.9^{\lfloor i/1000 \rfloor}}, \tag{5.3}$$

onde $\lfloor x/y \rfloor$ representa o piso da divisão de x por y.

Após 100000 iterações, utilizando-se $m = 30$, a melhor combinação de *features* encontradas pode ser encontrada na Tabela 5.2; ela obteve um erro de $54,84\%$.

μ_{13}, μ_{12}, μ_{11}, μ_{10}, μ_9, μ_7, μ_6, μ_4, μ_3, μ_2, σ_{13}, σ_{12}, σ_{11}, σ_{10}

Tabela 5.2: *Features* encontradas pela otimização.

Os parâmetros μ_i e σ_i são a média e o desvio padrão do i-ésimo coeficiente do MFCC, respectivamente.

Outras melhorias podem ser feitas também no algoritmo original. Para um refinamento das músicas que são utilizadas para estimar *tags*, visando a diminuir o número de músicas com poucas informações, podemos filtrar o conjunto K_+ removendo músicas que possuam menos que uma quantidade mínima de *tags*. Portanto, propomos subsituir o critério de seleção de músicas para estimação de *tags* e utilizar o Algoritmo 6.

Algoritmo 6: Algoritmo revisado de estimação de *tags*.

para cada *música* $k \in K_-$ **faça**
  selecione de K_+ as m músicas mais próximas de k com no mínimo t *tags* utizando uma função de distância F. Adicione no conjunto C ;
  se C *é um conjunto vazio* **então**
    adicione k em K_*
  fim
  caso contrário
    selecione as n *tags* que mais se repetem em C e adicone-as em k;
  fim
fim

Após as alterações propostas, obtivemos os resultados apresentados na Tabela 5.3 com diferentes parâmetros, onde t é o mínimo de *tags* que uma música deve ter para ser utilizada na estimação, e utilizando as *features* da Tabela 5.2.

Nota-se que à medida que restringimos a quantidade mínima de *tags* a serem

	$t = 1$	$t = 5$	$t = 10$	$t = 20$
$m = 1$	0,9758	0,97655	0,9804	0,9858
$m = 5$	0,8441	0,7087	0,6228	0,5309
$m = 10$	0,7752	0,6285	0,5304	0,4601
$m = 20$	0,7356	0,5922	0,525	**0,4424**
$m = 30$	0,7221	0,5915	0,5119	0,4477

Tabela 5.3: *One-error* do Algoritmo 6 com diferentes parâmetros.

utilizadas na estimação, a medida de erro se reduz consideravelmente. Uma última adaptação visando à melhoria do algoritmo é apresentada do Algoritmo 7. A limitação do número de vizinhos a serem utilizados para estimação de *tags* utilizando um valor fixo pode ser uma medida muito forte. Items que possuam muitos vizinhos próximos potencialmente estão deixando de utilizar informações relevantes, caso o número de vizinhos selecionado seja pequeno. Ao mesmo tempo, músicas que se encontram distantes das demais estariam recebendo informações pouco relevantes para estimar suas *tags*, uma vez que o algoritmo é obrigado a selecionar m músicas vizinhas. O Algoritmo 7 utiliza uma restrição para a distância máxima para seleção de vizinhos, em vez de uma quantidade fixa. A Tabela 5.4 mostra os resultados obtidos com diferentes parâmetros.

Algoritmo 7: Algoritmo revisado de estimação de *tags*.

para cada *música* $k \in K_-$ **faça**
  selecione de K_+ as músicas mais próximas de k com no mínimo t *tags* e uma distância menor do que d, utilizando uma função de distância F. Adicione no conjunto C ;
  se C *é um conjunto vazio* **então**
    adicione k em K_*
  fim
  caso contrário
    selecione as n *tags* que mais se repetem em C e adicone-as em k;
  fim
fim

	$t = 1$	$t = 5$	$t = 10$	$t = 20$
$d = 1,0$	0,9995	0,9992	0,9989	1,0
$d = 5,0$	0,7422	0,6263	0,5456	0,4707
$d = 8,0$	0,7482	0,6301	0,5445	**0,4690**
$d = 10,0$	0,7496	0,6316	0,5510	0,4707
$d = 15,0$	0,7519	0,6331	0,5543	0,4761

Tabela 5.4: *One-error* do Algoritmo 7 com diferentes parâmetros.

A melhor combinação de parâmetros encontrada, em destaque na Tabela 5.4, é ligeiramente menor do que a melhor precisão encontrada pelo Algoritmo 6. No entanto, este algoritmo faz com que músicas sem vizinhos próximos permaneçam sem *tags*, exigindo que outras técnicas sejam utilizadas para elas. As músicas que não puderem receber *tags* por terem uma distância maior do que d para todas as outras devem ser tratadas separadamente. As Seções 5.2 e 5.3 apresentam alternativas que podem ser utilizadas nesses casos.

Um ponto importante a ser discutido é a medida de precisão utilizada. Ela mede o acerto preciso da estimação das *tags*, como é feito em problemas de classificação *multi-label*. No entanto, para fins de modelagem de músicas a partir de suas anotações desejamos apenas que as *tags* estimadas tenham alta probabilidade nos mesmos tópicos em que a música possui alta probabilidade. O desenvolvimento de uma medida de precisão que leve em consideração tais fatos deve ser abordado em trabalhos futuros.

5.1.1 Exemplo de estimação de *tags*

Para demonstrar a qualidade da recomendação de *tags*, iremos mostrar um exemplo do processo de estimação. A música escolhida para ter suas *tags* estimadas foi *Blonde Redhead - U.F.O.*, cujas *tags* originais são mostradas na Tabela 5.5. Utilizaremos os parâmetros que resultaram no menor *one-error* da Tabela 5.4, ou seja, utilizaremos músicas com no mínimo 20 *tags* e com distância menor do que 8. Finalmente, a Tabela 5.6 mostra as top-5 *tags* mais frequentes entre as músicas mais próximas, ordenadas por frequência. Nota-se que 3 destas são anotações corretas da música original. O algoritmo de recomendação de *tags* tem capacidade de indicar anotações pertinentes à música original, e é ideal para utilização em músicas recém adicionadas na base de dados, reduzindo o problema de *cold-start* existente no sistema de recomendação proposto no Capítulo 4.

5.2 Adição de músicas em *playlists* a partir de *features* de áudio

Outra alternativa para o aproveitamento de músicas sem *tags* na geração de recomendações envolve pular completamente a etapa de recomendação baseada em *tags* e inserir músicas na lista de recomendação utilizando apenas *features* de áudio. Assim, uma vez que músicas são inseridas nas recomendações, é esperado que usuários comecem a inserir as anotações necessárias para que a modelagem de *tags* possa ser realizada normalmente. O esquema de recomendação se torna o apresentado no Algoritmo 8.

INDIE_ROCK BLONDE_REDHEAD INDIE ALTERNATIVE EXPERIMENTAL_ROCK ALTERNATIVE_ROCK ELOVIBELOVED PUNK NOISE DREAM_POP NEW_WAVE FEMALE_VOCALIST ROCK CHILL PAINFULLY_COLOURED UNIVERSITA STONESOUP STONESPIRALS PIN_IN_MY_HEART BREAKING_EVERYBODYS_HEART ALTERNATIVE_PUNK SAHILAS_LOVED ERISEE_ROOM LUV_IT SLORDIG KAPPE_SILENCE DESCOPERIRI MA_MAKESJOY_PERFECT ALVARODELICO THINGS_TO_LOOK_AT ABSOLUTTE_FAVORITTER KAPPE_MELA JSINDIE Q_JSLFM SONGS_FROM_FRIENDS_OF_STRETCHEAD GUITAR JAPANESE PSYCHEDELIC SINGER_SONGWRITER USA AMERICAN NEW_YORK ATMOSPHERIC ART_ROCK STONER AVANTGARDE OTHER AMERICAIN SURREALISM AMAZING_DRUMS TAG_THIS FAV_TRACKS FAB SONIC_YOUTH DOWNLOAD PERFECTION FEMALEVOCALISTSGDCHILL SONGS_BY_BANDS_WITH_THE_SUFFIX_HEAD QUIET_HOURS_OF_THE_NIGHT YESSS INTROSPECTIVE

Tabela 5.5: *Tags* reais da música *Blonde Redhead - U.F.O.*.

ALTERNATIVE
ROCK
POP
INDIE
CHILLOUT

Tabela 5.6: *Tags* estimadas para *Blonde Redhead - U.F.O.* com *tags* estimadas corretamente em destaque.

Algoritmo 8: Algoritmo de adição de músicas sem *tags* à *playlist*.

para cada *música utilizada como* query **faça**
  recomende n músicas utilizando um dos métodos discutido no Capítulo 4;
  calcule o vetor de *features* v médio para a lista de recomendação ;
  selecione as m músicas de K_{-} cujos vetores de *features* sejam os mais próximos de v;
fim

Um exemplo de resultado encontrado para $m = 1$ e $n = 10$, e utilizando as *features* na Tabela 5.2, a base de dados de MFCC discutida na seção anterior e geração de *playlist* por *random walk*, pode ser visto na Tabela 5.7. A música em destaque foi adicionada utilizando-se o Algoritmo 8.

Recomendações para *Emmylou Harris - I Will Dream*
Keller Williams - Keep It Simple
Billi Dean - You Don't Count The Cost
Mississippi Fred McDowell - Brooks Run into the Ocean
Taj Mahal - Keep Your Hands Off Her
Bugs Henderson - Los On Austin
Alabama - (There's A) Fire In The Night
The Greencards - All The WayFrom Italy
Julie Roberts - I Can't Get Over You
Shelby Lynne - If I Were Smart
Mary Gauthier - Just Say She's A Rhymer
D.R.I. - Money Stinks

Tabela 5.7: Recomendações para música *Emmylou Harris - I Will Dream* por *random walk* com música recomendada por MFCC.

Este método se propõe a ser uma alternativa de recomendação temporária para o aproveitamento de músicas sem informação de *tags* pelo sistema. Após a adição de um número mínimo de anotações a uma música por usuários do sistema, esta deve ser removida do conjunto K_- e modelada da forma usual.

5.2.1 Experimentos

Para avaliar o método proposto na Seção 5.2, iremos verificar a proporção de músicas recomendadas que se encontram no top-k% das músicas recomendadas utilizando os métodos baseados em LDA. Os experimentos foram realizados utilizando-se *playlists* de 10 itens, a distância KLD e gerando uma *playlist* simples, por razões de eficiência. Para os experimentos foram utilizadas apenas músicas que possuíam ambas as informações, de *tag* e *features* de áudio, reduzindo a base de dados para um total de 18580 itens. Os resultados obtidos se encontram na Tabela 5.8 para diferentes valores de k.

	$k = 10$	$k = 25$	$k = 40$	$k = 50$
Proporção de músicas	0,0768	0,1363	0,2417	0,89

Tabela 5.8: Proporção de músicas no top-k%.

Nota-se um grande salto nos valores em torno de $k = 40$. Os resultados encontrados indicam que ainda há necessidade de grandes melhorias no algoritmo. No

entanto, note que a probabilidade de uma música escolhida aleatoriamente estar nos top-10% seria de aproximadamente 0,005, o que indica que o algoritmo possui um desempenho melhor do que a escolha completamente aleatória.

Outra medida que poderia ser utilizada na a avaliação do algoritmo é o *Mean Reciprocal Rank* (MRR) [42], que é uma medida similar à precisão média em casos em que existe exatamente um valor estimado de interesse e desejamos avaliar a precisão da estimação em uma lista ranqueada. Caso o valor estimado seja o de ranque n, o seu ranque recíproco será $1/n$. Desta forma, caso as estimações sejam perfeitas (ou seja, todas possuam ranque 1 na lista) o MRR será igual a 1. A fórmula 5.4 apresenta como calcular esse valor.

$$\mathrm{MRR}_F = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{1}{\mathrm{R}_i(F(i))}, \tag{5.4}$$

onde $|Q|$ é a dimensão da base de dados, F é o algoritmo sendo avaliado, $F(i)$ a música selecionada através de *features* para ser adicionada à *playlist* gerada para música i e $R_i(x)$ representa o ranque da música x na *playlist* gerada para música i.

O resultado obtido foi 7×10^{-4}, mais uma vez indicando a necessidade de um maior estudo sobre o algoritmo usado para inserir músicas diretamente na *playlist* utilizando apenas *features* de áudio. No entanto, note que para grandes bases de dados é necessário interpretar o resultado do *MRR* com mais cuidado. Um item recomendado possuir alto ranque de semelhança não significa, necessariamente, que ele não seja similar ao item de *query*. No entanto, o cálculo do *MRR* não considera essa informação e irá atribuir um ranque recíproco baixo para o ítem recomendado.

5.3 Adição de *tag* de gênero em músicas sem anotações utilizando um classificador externo

O último método apresentado neste trabalho para o aproveitamento de músicas sem *tags* pelo sistema envolve o uso de um classificador de gêneros externo e preciso. Este classificador pode ser um sistema baseado em análise temporal de sinais, análise de padrões rítimicos ou até mesmo um ser humano especialista. Após a classificação de gênero, este é inserido como *tag* e o sistema de recomendação pode modelá-la da forma usual. Uma vez que *tags* relacionadas a gênero musical são muito frequentes, é de se esperar que o sistema resulte em boas recomendações.

Capítulo 6

Conclusão e trabalhos futuros

A presente dissertação apresentou um método de recomendação de músicas baseado na utilização do modelo LDA sobre *tags*. A redução de dimensionalidade sem redução de representatividade proporcionada pelo modelo LDA se mostrou altamente vantajoso para representar músicas em lugar de um conjunto de *tags*, permitindo a criação de um sistema de recomendação simples e que gera bons resultados, de acordo com os testes realizados na Seção 4.7.4.

Os sistemas de recomendação propostos na Seção 4.5 foram capazes de gerar *playlists* que mantêm consistência de gêneros em sua maior parte, sendo também capazes de adicionar músicas de gêneros distintos, possibilitando a geração de *playlists* diversificadas capazes de surpreender o usuário. Também foram propostos métodos de geração de perfis de usuário para recomendação sem a necessidade de *query* pelo usuário. Os métodos propostos representam o perfil de usuário de forma similar à representação de músicas, permitindo que os mesmos métodos para geração de *playlists* sejam utilizados, simplificando o sistema.

Também foi proposta uma nova forma de avaliação de sistemas de recomendação de música sem a utilização de testes *A/B*, reduzindo o custo da medição da qualidade do sistema. Tal medida é baseada na homogeneidade do gênero das músicas nas *playlists*. Ela é capaz de auxiliar no entendimento de como um sistema de recomendação se comporta, uma vez que é interessante para o usuário que as recomendações recebidas não sejam completamente homogêneas, i.e, que o sistema tenha alguma capacidade de surpreender o usuário. Embora não seja uma forma de avaliação que possa prescindir completamente da realização de testes com usuários reais, a medida enriquece o conhecimento que temos sobre as recomendações geradas.

Os métodos utilizados para tratar músicas sem *tags* no Capítulo 5 carecem de estudos mais aprofundados, uma vez que é extremamente importante para um sistema de recomendação a adição contínua de músicas novas. Também são necessários experimentos utilizando-se classificadores de gênero externos, como descrito na Seção 5.3.

Ainda se faz necessário um estudo mais profundo de métodos de avaliação de recomendações de música que considerem correlação entre gêneros. Para tal, um estudo social precisa ser desenvolvido para criar as fundações de como diferentes gêneros musicais são percebidos por ouvintes e como eles se relacionam. Além disso também se faz necessário o estudo de qual é a prporção ideal entre homogeneidade e heterogeneidade em *playlists* para satisfazer um usuário comum.

Por fim, para avaliação própria da qualidade das recomendações geradas pelo sistema, ainda se faz necessária a realização de testes do tipo A/B, colocando o sistema contra outros recomendadores existentes na literatura. Tal teste mostraria a real viabilidade, vantagens e desvantagens que o sistema possui.

Referências Bibliográficas

[1] SHALEV-SHWARTZ, S., SINGER, Y., NG, A. Y. "Online and batch learning of pseudo-metrics". In: *Proceedings of the 21st International Conference on Machine Learning (ICML'04)*, pp. 94–101, Banff, Canada, Jul 2004.

[2] BARRINGTON, L., ODA, R., LANCKRIET, G. "Smarter than genius? Human evaluation of music recommender systems". In: *Proc. of the 10th International Society for Music Information Retrieval Conference (ISMIR 2009)*, pp. 357–362, Kobe, Japan, Oct 2009. ISMIR.

[3] CELMA, O. *Music Recommendation and Discovery in the Long Tail*. Phd thesis, Department of Information and Communication Technologies, Universitat Pompeu Fabra, Barcelona, Spain, 2008.

[4] BLEI, D. M., NG, A. Y., JORDAN, M. L. "Latent Dirichlet allocation", *Journal of Machine Learning Research*, v. 3, pp. 993—-1022, Jan 2003.

[5] SEYERLEHNER, K. *Content-Based Music Recommender Systems: Beyond simple Frame-Level Audio Similarity*. Phd thesis, Faculty of Engineering and Natural Sciences, Johannes Kepler University Linz, Linz, Austria, 2010.

[6] LOPS, P., DE GEMMIS, M., SEMERARO, G. "Content-based recommender systems: State of the art and trends". In: Ricci, F., Rokach, L., Shapira, B., et al. (Eds.), *Recommender Systems Handbook*, Springer, cap. 3, pp. 73–105, Boston, USA, 2011.

[7] FIELDS, B., RHODES, C., D'INVERNO, M. "Using song social tags and topic models to describe and compare playlists". In: *Proc. of the Workshop on Music Recommendation and Discovery (WOMRAD 2010)*, Barcelona, Spain, Sep 2010. ACM RecSys.

[8] BERTIN-MAHIEUX, T., ELLIS, D. P. W., B., W., et al. "The Million Song Dataset". In: *Proc. of the 12th International Society for Music Information Retrieval Conference (ISMIR 2011)*, pp. 591–596, Miami, USA, Oct 2011. ISMIR.

[9] SARWAR, B.; KARYPIS, G. K. J., REIDL, J. "Item-based collaborative filtering recommendation algorithms". In: *Proc. of the 10th International Conference on World Wide Web (WWW'01)*.

[10] BURKE, R. "Hybrid recommender systems: Survey and experiments", *Journal on User Modeling and User-Adapted Interaction*, v. 12, n. 4, pp. 331—370, Nov 2002.

[11] RENTFROW, P. J., GOSLING, S. D. "The do re mi's of everyday life: The structure and personality correlates of music preferences", *Journal of Personality and Social Psychology*, v. 84, n. 6, pp. 1236—1256, Jun 2003.

[12] RENTFROW, P. J., GOSLING, S. D. "Message in a ballad: The role of music preferences in interpersonal perception", *Psychological science*, v. 17, n. 3, pp. 236—242, Mar 2006.

[13] GOSLING, S. D., R. P. J., SWANN, W. B. "A very brief measure of the big-five personality domains", *Journal of Research in Personality*, v. 37, n. 6, pp. 504—-528, Dec 2003.

[14] JENNINGS, D. *Net, Blogs and Rock 'n' Rolls: How Digital Discovery Works and What It Means for Consumers*. London, UK, Nicholas Brealey Publishing, 2007.

[15] SONG, Y., DIXON, S., PEARCE, M. "A survey of music recommendation systems and future perspectives". In: *Proc. of the 9th International Symposium on Computer Music Modelling and Retrieval (CMMR 2012)*, pp. 395–410, London, UK, Jun 2012.

[16] BRYNJOLFSSON, E., HU, Y. J., SMITH, M. D. "Consumer surplus in the digital economy: Estimating the value of increased product variety at online booksellers", *Management Science*, v. 49, n. 11, pp. 1580–1596, Nov 2003.

[17] BRYNJOLFSSON, E., HU, Y. J., SMITH, M. D. "From niches to riches: Anatomy of the long tail", *Sloan Management Review*, v. 47, n. 4, pp. 67–71, Summer 2006.

[18] ANDERSON, C. *The Long Tail: Why the Future of Business Is Selling Less of More*. London, UK, Random House, 2006.

[19] BONNIN, G., JANNACH, D. "Automated generation of music playlists: Survey and experiments", *ACM Computing Surveys*, v. 47, n. 2, pp. 26:1–26:35, Jan 2015.

[20] KNEES, P., POHLE, T., SCHEDL, M., et al. "Combining audio-based similarity with web-based data to accelerate automatic music playlist generation". In: *Proceedings of the 8th ACM International Workshop on Multimedia Information Retrieval (MIR'06)*, pp. 147–154, Santa Barbara, USA, Oct 2006. ACM.

[21] CREMONESI, P., GARZOTTO, F., NEGRO, S., et al. "Looking for "good"recommendations: A comparative evaluation of recommender systems". In: *Proceedings of the 13th IFIP TC 13 International Conference on Human-Computer Interaction (INTERACT 2011)*, v. III, pp. 152–168, Lisbon, Portugal, Sep 2011. IFIP.

[22] DOPLER, M., SCHEDL, M., POHLE, T., et al. "Accessing music collections via representative cluster prototypes in a hierarchical organization scheme". In: *Proc. of the 9th International Conference on Music Information Retrieval (ISMIR 2008)*, pp. 179–184, Philadelphia, USA, Sep 2008. ISMIR.

[23] FLEXER, A., SCHNITZER, D., GASSER, M., et al. "Playlist generation using start and end songs". In: *Proc. of the 9th International Conference on Music Information Retrieval (ISMIR 2008)*, pp. 173–178, Philadelphia, USA, Sep 2008. ISMIR.

[24] PAMPALK, E., POHLE, T., WIDMER, G. "Dynamic playlist generation based on skipping behavior". In: *Proc. of the 6th International Conference on Music Information Retrieval (ISMIR 2005)*, pp. 634–637, London, UK, Sep 2005. ISMIR.

[25] KAMALZADEH, M., BAUR, D., MÖLLER, T. "A Survey on Music Listening and Management Behaviours". In: *Proc. of the 13th International Society for Music Information Retrieval Conference (ISMIR 2012)*, pp. 373–378, Porto, Portugal, Oct 2012. ISMIR.

[26] LEE, J. H. "How similar is too similar?: Exploring users' perceptions of similarity in playlist evaluation". In: *Proc. of the 12th International Society for Music Information Retrieval Conference (ISMIR 2011)*, pp. 109–114, Miami, USA, Oct 2011. ISMIR.

[27] SHI, Y., LARSON, M., HANJALIC, A. "Exploiting user similarity based on rated-item pools for improved user-based collaborative filtering". In: *Proc. of the 3rd ACM Conference on Recommender Systems (RecSys'09)*, pp. 125–132, New York, USA, Oct 2009. ACM.

[28] SLANEY, M., WHITE, W. "Measuring Playlist Diversity for Recommendation Systems". In: *Proc. of the 1st ACM Workshop on Audio and Music Computing Multimedia (AMCMM'06)*, pp. 77–82, Santa Barbara, USA, Oct 2006. ACM.

[29] SALTON, G., MCGILL, M. *Introduction to Modern Information Retrieval*. New York, USA, McGraw-Hill, 1983.

[30] NIGAM, K., MCCALLUM, A. K., THRUN, S., et al. "Text classification from labeled and unlabeled documents using EM", *Machine Learning*, v. 39, n. 2-3, pp. 103–134, May-Jun 2000.

[31] GRIFFITHS, T. *Gibbs sampling in the generative model of Latent Dirichlet Allocation*. Tecnical report, Stanford University, 2002.

[32] ANDRIEU, C., DE FREITAS, N., DOUCET, A., et al. "An introduction to MCMC for machine learning". In: *Machine Learning*, v. 50, pp. 5–43, Jan 2003.

[33] HEINRICH, G. *Parameter Estimation for Text Analysis - v. 2.9*. Technical report, Fraunhofer IGD, Darmstadt, Germany, 2009.

[34] GRIFFITHS, T. L., STEYVERS, M. "Finding scientific topics", *Proceedings of the National Academy of Sciences of the USA*, v. 101, n. Suppl. 1, pp. 5228–5235, Apr 2004.

[35] SCHINDLER, E., RAUBER, A. "Capturing the temporal domain in echonest features for improved classification effectiveness". In: *Proc. of the 10th International Workshop on Adaptive Multimedia Retrieval (AMR 2012)*, pp. 214–227, Copenhagen, Denmark, Oct 2012.

[36] ARUN, R., SURESH, V., VENI MADHAVAN, C. E., et al. "On finding the natural number of topics with Latent Dirichlet Allocation: Some observations". In: *Proc. of the 14th Pacific-Asia Conference (PAKDD 2010)*, v. I, pp. 391–402, Hyderabad, India, Jun 2010.

[37] SI, X., SUN, M. "Tag-LDA for scalable real-time tag recommendation", *Journal of Information & Computational Science*, v. 6, n. 1, pp. 23–31, Jan 2009.

[38] ECK, D., LAMERE, P., BERTIN-MAHIEUX, T., et al. "Automatic Generation of Social Tags for Music Recommendation". In: *Proc. of the 21st Annual Conference on Neural Information Processing Systems (NIPS 2007)*, pp. 385–392, Vancouver, Canada, Dec 2008.

[39] ZHANG, M.-L., ZHOU, Z.-H. "ML-KNN: A lazy learning approach to multi-label learning", *Pattern Recognition*, v. 40, n. 7, pp. 2038–2048, Jul 2007.

[40] KIM, H.-G., MOREAU, N., SIKORA, T. *MPEG-7 Audio and Beyond. Audio Content Indexing and Retrieval*. Chichester, UK, Wiley, 2005.

[41] KIRKPATRICK, S., GELATT, C. D., VECCHI, M. P. "Optimization by simulated annealing", *SCIENCE*, v. 220, n. 4598, pp. 671–680, May 1983.

[42] CRASWELL, N. "Encyclopedia of Database Systems". cap. Mean Reciprocal Rank, p. 1703, Boston, USA, Springer, 2009.

Apêndice A

Distribuição Multinomial

A distribuição multinomial é uma generalização da distribuição binomial. Em teoria da probabilidade, a distribuição binomial define a probabilidade de um número de sucessos x em n realizações de experimentos independentes de Bernoulli. Se o evento X segue uma distribuição binomial de parâmetros p e n, sua distribuição de probabilidade é dada por:

$$p(X = x) = \binom{n}{k} p^x (1-p)^{n-x}, \tag{A.1}$$

onde p representa a probabilidade de um sucesso do evento e n é o número de repetições do experimento. De forma similar, se temos k categorias distintas, a distribuição multinomial nos dá a probabilidade de qualquer combinação de sucessos nas diversas categorias. Sua distribuição de probabilidade é dada por:

$$p(\vec{x}|\vec{p}) = \begin{cases} \dfrac{n!}{x_1! \cdots x_k!} p_1^{x_1} \cdots p_k^{x_k}, & \text{quando } \sum_{i=1}^{k} x_i = n; \\ 0 & \text{caso contrário,} \end{cases} \tag{A.2}$$

onde $\vec{x}$ e $\vec{p}$ são vetores de k dimensões. No caso em que $k = 1$, é fácil verificar que a equação A.2 resulta na equação A.1. Além disso, a equação A.2 pode ser reescrita como

$$p(\vec{x}|\vec{p}) = \frac{\Gamma(\sum_i x_i + 1)}{\prod_i \Gamma(x_i + 1)} \prod_{i=1}^{k} p_i^{x_i}, \tag{A.3}$$

onde Γ representa a função Gamma.

Apêndice B

Distribuição de Dirichlet

A distribuição de Dirichlet é uma familia de distribuições com valores contínuos e reais. Sua densidade é dada por:

$$f(\vec{x};\vec{\alpha}) = \frac{1}{\mathrm{B}(\alpha)} \prod_{i=1}^{K} x_i^{\alpha_i - 1}, \tag{B.1}$$

onde $\vec{\alpha}$ é um parâmetro de dimensão K e composto de valores reais positivos. O vetor $\vec{x}$ se encontra em um $(K-1)$-simplex, de forma que:

$$x_1, ..., x_K > 0$$
$$\sum_{i=1}^{K} x_i = 1.$$

A constante de normalização $\mathrm{B}(\vec{\alpha})$ é a distribuição Beta multinomial, que possui forma

$$\mathrm{B}(\vec{\alpha}) = \frac{\prod_{i=1}^{K} \Gamma(\alpha_i)}{\Gamma\left(\sum_{i=1}^{K} \alpha_i\right)}. \tag{B.2}$$

A distribuição de Dirichlet é a priori conjugada para as distribuições categórica e multinomial, o que significa que se o vetor de probabilidades destas seguir uma distribuição de Dirichlet a priori, a probabilidade a posteriori também será uma distribuição de Dirichlet. Em redes bayesianas hierárquicas, esse resultado é desejável e comumente utilizado.

Um caso especial da distribuição de Dirichlet ocorre quando todos os componentes de $\vec{\alpha}$ possuem o mesmo valor, chamando-se distribuição de Dirichlet simétrica. Essa forma é comumente utilizada quando a distribuição de Dirichlet serve como priori dos parâmetros de uma distribuição multinomial ou categórica, pois indica que não temos nenhuma informação a mais sobre uma categoria ou outra. Nesse caso a equação B.1 pode ser simplificada para

$$f(\vec{x};\alpha) = \frac{\Gamma(\alpha K)}{\Gamma(\alpha)^K} \prod_{i=1}^{K} x_i^{\alpha-1}. \tag{B.3}$$