Universidade Federal
do Rio de Janeiro
Escola Politécnica

AVALIAÇÃO DE SISTEMAS DE RECOMENDAÇÃO COM UMA PROPOSTA DE UM ALGORITMO HÍBRIDO

Rafael Gonsalves Rozendo

Projeto de Graduação apresentado ao Curso de Engenharia de Computação e Informação da Escola Politécnica da Universidade Federal do Rio de Janeiro como parte dos requisitos necessários para a obtenção do título de Engenheiro.

Orientador: Alexandre Gonçalves Evsukoff

Rio de Janeiro
Fevereiro de 2017

AVALIAÇÃO DE SISTEMAS DE RECOMENDAÇÃO COM UMA PROPOSTA DE UM ALGORITMO HÍBRIDO

Rafael Gonsalves Rozendo

PROJETO SUBMETIDO AO CORPO DOCENTE DO CURSO DE ENGENHARIA DE COMPUTAÇÃO E INFORMAÇÃO DA ESCOLA POLITÉCNICA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE ENGENHEIRO DE COMPUTAÇÃO E INFORMAÇÃO.

Examinadores:

Prof. Alexandre Gonçalves Evsukoff, Dr.

Prof. Daniel Ratton Figueiredo, Ph. D.

Prof. Ricardo Guerra Marroquim, D. Sc.

RIO DE JANEIRO, RJ – BRASIL
FEVEREIRO DE 2017

Rozendo, Rafael Gonsalves

Avaliação de sistemas de recomendação com uma proposta de um algoritmo híbrido/Rafael Gonsalves Rozendo. – Rio de Janeiro: UFRJ/POLI – COPPE, 2017.

XII, 78 p.: il.; 29, 7cm.

Orientador: Alexandre Gonçalves Evsukoff

Projeto (graduação) – UFRJ/ Escola Politécnica/ Curso de Engenharia de Computação e Informação, 2017.

Referências Bibliográficas: p. 73 – 78.

1. Sistemas de Recomendação. 2. Avaliação. 3. Mahout. 4. Fatoração de matrizes. 5. Híbrido. I. Evsukoff, Alexandre Gonçalves. II. Universidade Federal do Rio de Janeiro, Escola Politécnica/ Curso de Engenharia de Computação e Informação. III. Título.

Agradecimentos

Agradeço à minha mãe, ao meu pai e ao meu avô por sempre estarem presentes, mesmo quando a distância era um empecilho.

Agradeço a todos os meus amigos, que de forma geral sempre trouxeram alegrias e ajudaram a aliviar o estresse do dia-a-dia. Em especial, agradeço aos amigos Gabriel Sab e Rafael Cardoso que estiveram presentes em todos os momentos ao longo desta trajetória universitária, sejam estes momentos bons ou difíceis.

Agradeço aos professores do curso de Engenharia de Computação e Informação da UFRJ. Em especial, aos professores Daniel Ratton, Ricardo Marroquim, Alexandre Evsukoff, Cláudio Esperança e Sergio Barbosa Villas-Boas.

Por fim, agradeço à Escola Politécnica da UFRJ por proporcionar uma educação de ótima qualidade e por todas as oportunidades.

Resumo do Projeto de Graduação apresentado à Escola Politécnica/COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Engenheiro de Computação e Informação.

AVALIAÇÃO DE SISTEMAS DE RECOMENDAÇÃO COM UMA PROPOSTA DE UM ALGORITMO HÍBRIDO

Rafael Gonsalves Rozendo

Fevereiro/2017

Orientador: Alexandre Gonçalves Evsukoff

Curso: Engenharia de Computação e Informação

Ao longo da última década a oferta de produtos e serviços na web aumentou drasticamente. Este fato, aliado à crescente disponibilidade de informações através da Internet, certamente trouxe inúmeros benefícios para o consumidor, porém também tem como consequência a dificuldade em filtrar os produtos realmente relevantes. Diversas lojas online aproveitaram esta oportunidade para rentabilizar em cima de sistemas de recomendação, que sugerem itens ao usuário dos quais ele provavelmente irá gostar. Desta forma, sistemas de recomendação diferentes podem produzir recomendações diferentes, impactando positiva ou negativamente o balanço da empresa. Este trabalho tem como objetivo avaliar de forma objetiva diferentes técnicas de recomendação, baseando-se em um conjunto de dados real disponibilizado por um serviço de streaming de músicas.

Palavras-Chave: Sistemas de Recomendação, Avaliação, Mahout, Fatoração de matrizes, Híbrido.

Abstract of the Undergraduate Project presented to Poli/COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Computer and Information Engineer.

EVALUATION OF RECOMMENDER SYSTEMS WITH A PROPOSAL OF A HYBRID ALGORITHM

Rafael Gonsalves Rozendo

February/2017

Advisor: Alexandre Gonçalves Evsukoff

Course: Computer and Information Engineering

During the last decade, the supply of products and services on the web has drastically increased. This fact, in addition to the increasing availability of information through the Internet, has certainly brought various benefits to the consumer. However, it also makes it difficult to filter indeed relevant products. Several online stores has seized this opportunity to profit on recommender systems, which suggest items that the user might enjoy. Thus, different recommender systems may produce different recommendations, affecting positively or negatively the revenue. This work aims to evaluate different recommendation techniques, basing on a real dataset made available by a music streaming service.

Keywords: Recommender Systems, Evaluation, Mahout, Matrix Factorization, Hybrid.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

O advento da Internet ao longo das últimas décadas tem causado grandes mudanças na vida das pessoas. A popularização de mídias digitais e de ferramentas de busca como a fornecida pela empresa Google facilitou o acesso à informação de forma talvez nunca antes vista. Além disso, a popularização de blogs, fóruns online e redes sociais como o Facebook proporcionou não só um espaço para interação, mas também uma grande plataforma para a troca de informações entre pessoas de diferentes partes do mundo. Em 2006, uma publicação do The Economist reportou que as pessoas leem por volta de 10MB (*megabytes*) de conteúdo por dia, ouvem 400MB por dia veem 1MB de informação por segundo [1]. Com esta grande quantidade de informação constantemente bombardeando os usuários da Internet, filtrar o conteúdo relevante passa a ser uma tarefa complicada.

O avanço da Internet também causou uma grande mudança no modelo de consumo. Hoje, presenciamos um rápido crescimento do comércio eletrônico em escala mundial. Nos EUA, o percentual de vendas através do comércio eletrônico em relação ao total de vendas vem crescendo constantemente, e no terceiro trimestre de 2016 registrou a maior alta até o momento - 8,4% do total [2]. Além disso, serviços de *streaming* de vídeo (e de conteúdo *on-demmand* em geral) como Netflix e Hulu se mostram cada vez mais populares. Apesar de prático, este modelo enfrenta também desafios inerentes ao fato de estar imerso na Internet. Ao utilizar este modelo de compras *online*, o consumidor se vê em meio a uma enorme quantidade de opções, tornando muitas vezes difícil a tarefa de encontrar o produto mais adequado.

Sendo assim, oferecer um sistema que possa recomendar produtos relevantes ao consumidor passa a ser um fator muito importante pois diminui o trabalho que ele teria de filtrar o conteúdo disponível, melhorando a experiência do usuário e garantindo uma maior satisfação. Este sistema recebe o nome de Sistema de Recomendação.

Sistema de recomendação é o nome dado ao conjunto de ferramentas, técnicas e algoritmos utilizados para sugerir itens a um usuário [3]. De forma geral, os

sistemas de recomendação ajudam os usuários a encontrar conteúdo, produtos ou serviços (como livros, produtos digitais, filmes, músicas, programas de TV e web sites) agregando e analisando sugestões de outros usuários e/ou utilizando o conhecimento prévio que se tem sobre as preferências do usuário e as características dos itens disponíveis [4] [5] [6]. A saída de um sistema de recomendação é normalmente uma lista de itens recomendados, ordenada pelo grau estimado de preferência do usuário.

1.1 Motivação

Sistemas de recomendação têm sido uma importante área de pesquisa desde meados da década de 1990 [6]. Dado o impacto positivo que o sistema pode causar na experiência do consumidor e portanto na capacidade de vendas, diversos sites de comércio eletrônico já utilizam algum tipo de sistema de recomendação. Alguns são simplesmente uma lista dos produtos mais populares do estoque (recomendação não-personalizada), enquanto que outros empregam técnicas mais complexas de mineração de dados produzindo resultados diferentes de acordo com o usuário em questão (recomendação personalizada) [6]. Inclusive, nomes importantes do comércio eletrônico como Amazon.com e Netflix empregam seus bem-sucedidos sistemas de recomendação como um de seus diferenciais [7]. A tabela 1.1 contém alguns exemplos de sites que utilizam sistemas de recomendação e o tipo de item que é recomendado, e a figura 1.1 mostra alguns exemplos de itens recomendados pela Amazon a um usuário que comprou recentemente materiais de acampamento.

Tabela 1.1: Sites populares que utilizam sistemas de recomendação [6]

Site	**Tipo de item**
Amazon	Livros/outros produtos
Facebook	Amigos
Netflix	DVDs
MovieLens	Filmes
eHarmony	Encontros
CareerBuilder	Empregos
Digg	Notícias

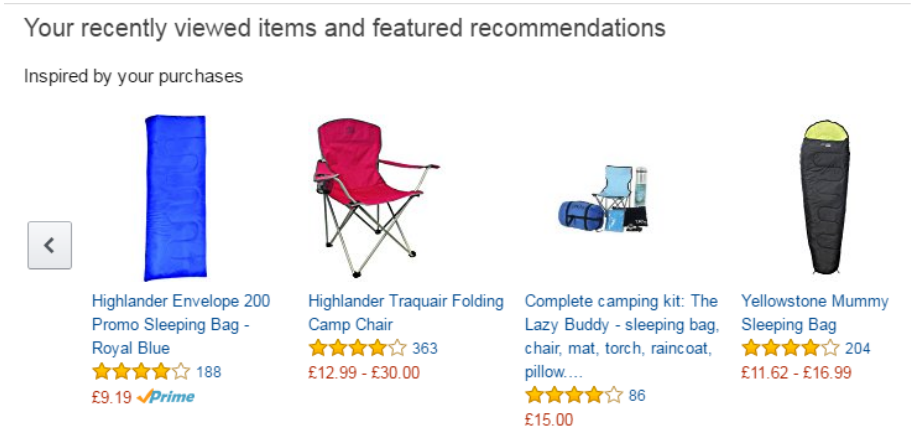

Figura 1.1: Exemplo de recomendações no loja eletrônica Amazon [8].

Os sistemas de recomendação são especialmente importantes em sites que querem se aproveitar do efeito da cauda longa [9], ou seja, os itens que individualmente não são muito vendidos mas que conseguem gerar lucros consideráveis devido à sua variedade. Na Amazon, por exemplo, entre 20% e 40% das vendas é devido a produtos que não estão entre os 100 mil produtos mais vendidos [10] [11]. A figura 1.2 ilustra o efeito da cauda longa. Sendo assim, é notável o valor que um sistema de recomendação consegue trazer para uma empresa.

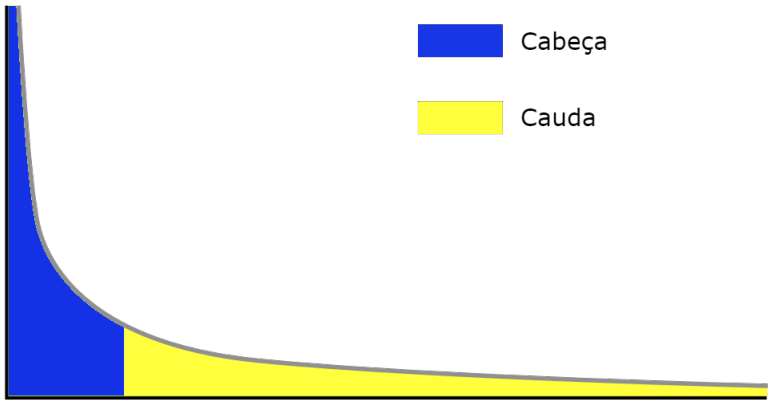

Figura 1.2: Cauda longa. Neste caso, a área total da cauda é igual à área total da cabeça. [12].

Em muitos casos, os usuários se mostram inclinados a fornecer seu nível de satisfação com os itens vistos, aumentando assim a quantidade de dados disponíveis e possibilitando que técnicas mais complexas sejam utilizadas para analisar as preferências dos consumidores e assim fornecer recomendações de maior qualidade [7].

Existem diversas formas de produzir uma recomendação, mas não existe necessariamente um método que é sempre o melhor. A escolha do método de recomendação

depende do contexto e inclui fatores como o tipo e a quantidade de informação disponível a respeito dos itens e dos usuários. Em alguns casos temos disponível o grau de satisfação que os usuários do sistema forneceram para os itens do estoque, tornando possível identificar o grau de semelhança entre eles. Por exemplo, na Amazon os usuários dão notas para os produtos que variam de 1 a 5 estrelas. Em outros casos, tudo o que temos disponível é um conjunto de palavras-chave que definem um item. Por exemplo, em documentos de texto em geral. Sendo assim, para traçar uma estratégia de recomendação adequada ao negócio, é necessário primeiro avaliar os resultados que diferentes métodos de recomendação iriam proporcionar.

1.2 Prêmio Netflix

Em outubro de 2006, a empresa Netflix anunciou um concurso. O objetivo dos participantes era desenvolver um sistema de recomendação que conseguisse uma precisão pelo menos 10% melhor que a do algoritmo da empresa, o Cinematch. Ao time vencedor seria concedido um prêmio de U$ 1 milhão. Para isso, a empresa disponibilizou um *dataset* com mais de 100 milhões de notas que os usuários deram para filmes. Cada nota era um número inteiro entre 1 e 5 (número de estrelas que o usuário deu ao filme), e acompanhava a data da avaliação. O *dataset* era composto por mais de 480 mil usuários anônimos e quase 18 mil filmes. Para cada filme, também era disponível o nome e o ano de lançamento. [13].

O prêmio Netflix foi um divisor de águas para a pesquisa em sistemas de recomendação, pois até o momento não havia nenhum outro *dataset* público com uma quantidade tão grande de dados. Antes disso, o único *dataset* público disponível era várias ordens de grandeza menor [7].

O concurso gerou um grande entusiasmo na comunidade. Em 2009, quando a meta foi atingida e o concurso foi concluído, haviam sido registrados 51 mil participantes de mais de 41 mil equipes de 186 países diferentes. Destes, foram registradas 44 mil entradas válidas de mais de 5 mil equipes [14].

O algoritmo de recomendação vencedor do concurso era uma combinação de 107 algoritmos diferentes. É interessante notar que a Netflix utilizou como base alguns destes algoritmos para melhorar o seu sistema de recomendação, porém nunca chegou a utilizar o algoritmo vencedor do concurso em seu ambiente de produção, pois o ganho de precisão não justificava os esforços de engenharia necessários principalmente devido ao enorme número de notas disponíveis (mais de 5 bilhões) [15].

1.3 Objetivo

O objetivo deste trabalho é analisar de forma objetiva a qualidade das recomendações que um sistema de recomendação consegue gerar para um dado conjunto de dados. Diferentes abordagens serão analisadas utilizando um *framework* já estabelecido. Novas alternativas também serão sugeridas para melhorar a precisão das recomendações de forma e geral e também para contornar um dos desafios que são encarados por sistemas de recomendação em aplicações reais: o *cold-start* ("arranque a frio"), ou seja, o momento inicial de um sistema de recomendação em que a quantidade de dados disponíveis pode não ser suficiente para produzir uma recomendação suficientemente boa.

1.4 Estruturação do documento

No capítulo 2, veremos conceitos básicos e uma revisão bibliográfica abrangente sobre sistemas de recomendação. Em seguida, o capítulo 3 apresenta as ferramentas tecnológicas e o conjunto de dados que foram utilizados. O capítulo 4 apresenta as metodologias utilizadas no processo de avaliação feito neste trabalho e detalha os algoritmos utilizados. O capítulo 5 apresenta os resultados de avaliação obtidos, mostrando também os efeitos que podem ser alcançados variando conjuntos de parâmetros. Por fim, o capítulo 6 conclui o trabalho e faz referência a trabalhos futuros.

Capítulo 2

Conceitos Básicos

Este capítulo irá apresentar alguns dos principais conceitos por trás de um sistema de recomendação de forma introdutória para que as metodologias que de fato foram usadas no trabalho possam ser compreendidas em um capítulo futuro.

Existem diversos tipos de sistemas de recomendação. Três das principais categorias são: filtragem colaborativa, filtragem baseada em conteúdo e filtragem baseada em conhecimento. Pode-se dizer que ainda existe uma quarta categoria: os sistemas híbridos. Porém, esta nada mais é do que uma combinação dos outros tipos (ou até mesmo de diferentes sistemas de um mesmo tipo).

2.1 Sistema de recomendação

Como já foi apresentado na seção 1, um sistema de recomendação analisa o histórico de preferências de todos os usuários do sistema e/ou o conjunto de características dos itens para assim determinar quais itens um dado usuário possivelmente irá gostar, recomendando então estes itens. Mais especificamente, um sistema de recomendação funciona da seguinte maneira:

Seja U o conjunto de todos os usuários, I o conjunto de todos os itens disponíveis e I' o conjunto dos itens que ainda não foram vistos/avaliados pelo usuário u. O sistema de recomendação tenta prever a nota que o usuário u daria para cada um dos itens em I'. Estes métodos de predição de nota serão vistos mais detalhadamente no capítulo 4, mas normalmente são baseados em uma matriz de utilidade R. A matriz de utilidade é de forma geral uma representação matricial das relações entre U e I. Esta representação é mais comum em sistemas de filtragem colaborativa e costuma conter explicitamente a avaliação dos usuários aos itens (por exemplo, o elemento $r_{i,j}$ é a avaliação que o usuário i deu ao item j), mas também pode conter informações como ocorrência de compra de itens (por exemplo, $r_{i,j} = 1$ se o usuário i comprou o item j e 0 caso contrário). Após as notas serem todas previstas, o sistema ordena os itens de I' por ordem decrescente de nota e recomenda os N primeiros itens ao

usuário u, onde N é normalmente um parâmetro do sistema.

(OBS: como será visto a seguir, nem todos os sistemas de recomendação utilizam explicitamente este conceito de avaliação do item, mas nestes casos a "nota"pode ser entendida como o grau de satisfação que o usuário tem ao utilizar o item)

2.2 Filtragem colaborativa

A ideia por trás da filtragem colaborativa (*collaborative filtering* - CF) é coletar e analisar as preferências dos usuários para determinar padrões de semelhança entre usuários diferentes. Estas preferências podem ser explícitas ou implícitas. Um exemplo de preferência explícita pode ser a quantidade de estrelas que um usuário do Netflix dá a um filme, enquanto que um exemplo de preferência implícita pode ser a lista de produtos que um usuário viu ou comprou na Amazon, indicando com o valor 1 os itens que foram comprados e com 0 os itens que não foram comprados. Na figura 1.1 também é possível ver um exemplo de preferência explícita, em que os itens recebem notas como número de estrelas.

A filtragem colaborativa também faz uma suposição fundamental: se dois usuários compartilharam os mesmos interesses no passado - por exemplo, se compraram ou viram os mesmos produtos - então eles também irão compartilhar os mesmos interesses no futuro [16]. Ou seja, se dois usuários X e Y no passado compraram muitos produtos em comum e o usuário X recentemente comprou um produto que Y ainda não comprou, então é válido pensar que o usuário Y também gostaria de comprar este mesmo produto.

Uma abordagem muito semelhante mas ainda assim sutilmente diferente - e com consequências consideráveis, como será explicado adiante - é a de considerar a semelhança entre itens em vez de usuários. Neste caso, a ideia é a de que se dois itens no passado foram constantemente avaliados de forma semelhante, então no futuro eles continuarão sendo avaliados de forma semelhante. Desta forma, se os itens A e B são semelhantes e o usuário X gostou do item A, então ele provavelmente também irá gostar do item B.

A filtragem colaborativa pode ser subdividida em 2 tipos:

- Filtragem colaborativa baseada em memória (*memory-based*) - Neste caso, é adotado o conceito básico explicado acima: os itens recomendados a um dado usuário são os que foram preferidos pelos usuários que compartilharam preferências semelhantes, ou então os itens semelhantes aos que o usuário preferiu no passado [11].

- Filtragem colaborativa baseada em modelos (*model-based*) - Neste caso, os itens recomendados são escolhidos a partir de um modelo que é treinado para

identificar padrões no conjunto de dados de entrada [11]. Estes modelos serão detalhados adiante.

Uma grande vantagem da filtragem colaborativa é que não é necessário saber nenhum tipo de informação a respeito dos usuários ou dos itens, como gêneros musicais ou autores dos livros. A única entrada necessária para o algoritmo é uma matriz contendo as notas que os usuários deram aos itens [16]. Porém, ao mesmo tempo isso pode ser considerado uma desvantagem, já que estas informações a respeito dos itens poderiam ser exploradas de forma a produzir recomendações de melhor qualidade.

2.2.1 Desafios

O método de filtragem colaborativa sofre com alguns problemas já bem estudados pela literatura [16]. Os principais são:

- Escalabilidade

  A quantidade de dados disponíveis, principalmente para grandes sites como Amazon e Netflix, é enorme. Isto implica em um custo computacional, tanto de processamento quanto de armazenamento, que deve ser levado em conta no caso de sistemas de recomendação que serão utilizados em aplicações reais. Em alguns casos pode ser interessante paralelizar o processamento [11] ou realizar parte do processamento *offline*.

- Esparsidade

  Devido ao grande número de itens disponíveis normalmente no catálogo, os usuários costumam dar notas para apenas uma fração bem pequena dos itens. Além disso, muitas vezes a sobreposição entre usuários é bem pequena ou até mesmo inexistente [11]. O sistema de recomendação deve ser capaz de recomendar itens mesmo nessa situação.

- *Cold-start*

  O método de filtragem colaborativa pura não possui informação suficiente para produzir recomendações a um novo usuário que acessa o sistema pela primeira vez, já que ele depende do histórico do usuário. Neste caso, normalmente o sistema pede por algumas informações adicionais ou então são utilizadas alternativas híbridas, que serão detalhadas adiante.

2.3 Filtragem baseada em conteúdo

Os sistemas de recomendação baseados em conteúdo (*content-based* - CB), diferente da filtragem colaborativa, utilizam as características dos itens para produzir recomendações. Por exemplo, algumas das características possíveis para descrever filmes são o gênero, diretor, ator principal, entre outros. Estas características (ou seja, o conteúdo) são comparadas com o perfil do usuário (ou seja, os seus gostos particulares) para que o sistema determine se o usuário irá se interessar ou não pelo item. Por exemplo, um usuário que tem o perfil de quem gosta de filmes de fantasia provavelmente iria se interessar pelos filmes da trilogia do *Senhor dos Anéis*.

As características dos itens podem ser obtidas de forma automática ou manual, mas, dependendo do tipo de item que é recomendado pelo sistema, nem sempre são fáceis de se obter. A filtragem baseada em conteúdo é muito utilizada para recomendar documentos de texto, já que o processo de mineração de texto já é bem consolidado e utiliza técnicas [17] que já são estudadas há décadas. Além disso, em muitos outros domínios de itens (como livros, filmes ou músicas) é possível extrair características de teor técnico facilmente, principalmente quando os itens estão disponíveis em um catálogo eletrônico, já que são características normalmente disponibilizadas pelo próprio fornecedor. Porém, características de teor subjetivo, como facilidade de uso ou qualidade do *design*, podem ser mais difíceis de extrair [16].

O perfil do usuário pode ser determinado automaticamente através de seu histórico de preferências (semelhante ao método de filtragem colaborativa) ou então a partir de informações adicionais perguntadas pelo sistema. Por exemplo, o sistema de recomendação de músicas da plataforma Deezer pergunta ao usuário em seu primeiro acesso quais são os seus gêneros musicais preferidos, para assim determinar um perfil. Esta situação pode ser vista na figura 2.1.

Figura 2.1: Informações adicionais perguntadas pelo Deezer para traçar um perfil de usuário [18].

A filtragem baseada em conteúdo, quando comparada com a filtragem colaborativa, tem a vantagem de não necessitar de grandes quantidades de usuários nem de informações prévias a respeito dos gostos dos usuários para produzir recomendações razoáveis - ainda que uma quantidade mínima de informações seja desejável para conseguir traçar automaticamente um perfil para o usuário. Além disso, uma vez que um item já está devidamente descrito através de suas características, ele já pode ser recomendado, diferentemente da filtragem colaborativa em que é necessário que se tenha alguma avaliação para o item. As desvantagens deste método estão relacionadas com a dificuldade de obter certas características de alguns tipos de itens (como já foi comentado acima) e com a possível introdução de erros no sistema ao tentar determinar manualmente o perfil de um usuário através de informações adicionais, caso elas não sejam utilizadas da forma mais adequada. Além disso, as semelhanças entre preferências de usuários diferentes não são exploradas [11].

2.4 Filtragem baseada em conhecimento

Não existe uma fronteira muito bem definida entre a filtragem baseada em conteúdo e a filtragem baseada em conhecimento [16] (*knowledge-based* - KB). Alguns autores consideram os dois como um único tipo de sistema de recomendação, e em muitos trabalhos (como [6], [11], [19]) sequer existe menção a sistemas de recomendação baseados em conhecimento, reforçando a hipótese de que este é visto por muitos

simplesmente como um sistema de recomendação baseado em conteúdo.

Entretanto, outros autores consideram que este se trata de um tipo diferente de sistema de recomendação. Em geral, considera-se que os sistemas de recomendação baseados em conhecimento costumam ser utilizados em domínios em que, por natureza, os usuários normalmente não possuem um histórico relevante de preferências. Por exemplo, itens como carros ou casas não costumam ser comprados frequentemente. Nestes cenários, o método de filtragem colaborativa não teria bons resultados dado o histórico limitado. [16] [20].

Além disso, em outros casos o usuário gostaria de informar requisitos adicionais que não necessariamente estão diretamente relacionados às características que descrevem o item. Por exemplo, ele gostaria de comprar um telefone celular na cor branca pagando no máximo R$1500. Neste caso, o sistema deveria tratar de formulações que normalmente não são abordadas nem pelos sistemas baseados em conteúdo [16]. Um ponto a se observar é que, para capturar este tipo de requisito, é necessário que o sistema seja altamente interativo, em contraste com a interação limitada que é exigida pelos demais métodos de recomendação, onde normalmente só é necessário informar uma nota [16].

A vantagem deste modelo é ser totalmente independente do histórico de preferências dos usuários. As desvantagens estão relacionadas à dificuldade de representação de conhecimento e à necessidade de compreender a fundo o domínio em questão. Traçando um paralelo entre a filtragem baseada em conhecimento e sistemas especialistas [21], a dificuldade de adaptação do sistema aos eventos de curto prazo também pode ser considerada uma desvantagem.

2.5 Sistemas híbridos

Cada um dos métodos de recomendação descritos até aqui exploram diferentes tipos de informação disponíveis, e por isso possuem suas próprias vantagens e desvantagens. Por exemplo, a filtragem colaborativa foca nas notas em itens fornecidas pela comunidade, enquanto que a filtragem baseada em conteúdo foca nas características dos itens e nos perfis dos usuários. Isto significa que nenhum deles pode ser considerado o ideal em todas as situações.

Pensando nisto, um sistema de recomendação híbrido (também chamado de método *ensemble*) utiliza diferentes métodos de recomendação e combina os resultados obtidos para produzir recomendações de melhor qualidade [11] [22]. Uma estratégia simples para combinar estes resultados poderia ser uma combinação linear dos resultados de cada método [23], em que cada método é ponderado por um fator que indica a confiança relativa dele em relação aos outros. A figura 2.2 ilustra a combinação de diferentes métodos em um sistema híbrido.

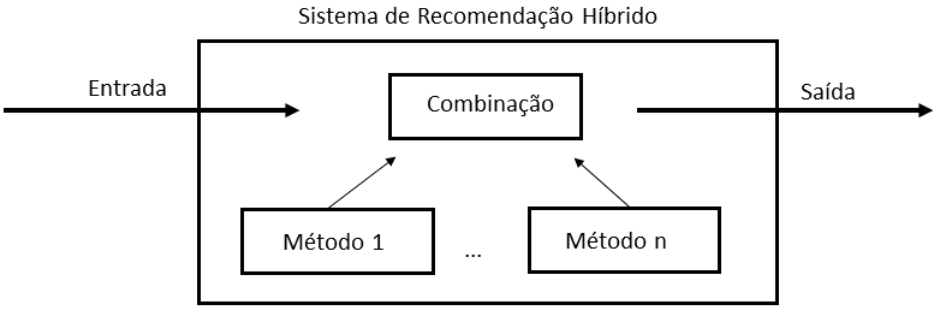

Figura 2.2: Representação de um sistema de recomendação híbrido [16].

Por utilizarem diferentes fontes de informação, os sistemas híbridos podem ser utilizados para resolver problemas enfrentados por estratégias individuais de recomendação. Em especial, os sistemas híbridos são utilizados para resolver o problema do *cold-start* da filtragem colaborativa, combinando os dados colaborativos e o conteúdo dos itens de forma que até mesmo usuários que nunca avaliaram nenhum item possam receber alguma recomendação [11] [24].

Além disso, os sistemas híbridos possuem a capacidade de produzir recomendações mais precisas. Como visto no Prêmio Netflix (ver seção 1.2), os algoritmos vencedores eram na verdade combinações complexas de diversos outros algoritmos.

2.6 Conclusão

Neste capítulo foram apresentados conceitos básicos por trás de um sistema de recomendação, como a ideia geral de seu funcionamento e como se comportam de forma geral os diferentes tipos de sistema de recomendação.

Os diferentes tipos de sistemas de recomendação ainda possuem suas subdivisões e peculiaridades, porém elas serão apresentadas nos próximos capítulos. Este capítulo é um pré-requisito para a compreensão dos conceitos mais específicos que serão apresentados nos próximos.

Capítulo 3

Ferramentas

Como foi descrito na seção 1.3, o objetivo deste trabalho é avaliar o resultado produzido por diferentes tipos de sistema de recomendação. Este capítulo irá começar a descrever o processo de avaliação que foi utilizado no trabalho, apresentando as ferramentas tecnológicas e o conjunto de dados utilizados. Ao final do capítulo estaremos prontos para detalhar as metodologias utilizadas, uma vez que os conceitos introdutórios já foram apresentados e as ferramentas terão sido devidamente apresentadas.

3.1 Conjunto de dados

Como já foi discutido anteriormente, o conjunto de dados (*dataset*) disponível está diretamente relacionado com a qualidade das recomendações e com a possibilidade de pesquisas mais conclusivas, principalmente em relação à filtragem colaborativa. Sendo assim, para este trabalho foi escolhido um *dataset* de tamanho considerável, mesmo com o nível de complexidade computacional exigido.

O *dataset* escolhido é disponibilizado publicamente pelo programa *Yahoo Webscope*, um programa da empresa *Yahoo* dedicado exclusivamente a oferecer *datasets* para fins acadêmicos [25]. Este dataset recebe o nome de **R1** e pode ser obtido em [26].

O *dataset* é um arquivo de texto que contém as notas que os usuários do serviço *Yahoo! Music* deram para vários artistas musicais ao longo de um mês. Ele é composto por 1.948.882 usuários anônimos e 98.211 artistas (itens). Ao total, foram registradas 11.557.943 notas. Cada nota pode ser um número inteiro entre 0 e 100 (note que 0 não significa "sem nota", e sim uma nota baixa) ou então o número 255, que significa "nunca tocar novamente". Cada linha do arquivo é uma tupla de 3 elementos, contendo o ID do usuário, o ID do artista e a nota que o usuário deu ao artista. Os artistas não são anônimos, pois existe um arquivo secundário contendo o nome do artista para cada ID. Porém, além do nome, não foi disponibilizada

nenhuma informação adicional a respeito dos artistas.

3.1.1 Tratamento

As notas 255 poderiam impactar o processo de avaliação de forma inesperada, uma vez que indicam um grande descontentamento com o item mas ainda assim são números maiores que a nota máxima. Isto implicaria em uma maior complexidade nos algoritmos para tratar estes casos especiais. Sendo assim, o *dataset* original foi modificado: as notas 255 foram transformadas em 0. Isto representa uma pequena mudança conceitual no valor das notas: antes, uma nota 0 significava uma nota ruim enquanto que uma nota 255 significava uma nota péssima. Agora, não existe esta diferença explícita. Em outras palavras, este tratamento penaliza mais as notas 0.

Porém, no ponto de vista do usuário, é razoável afirmar que ele não gostaria de receber como recomendação um item para o qual ele daria nota 255 e nem nota 0. Sendo assim, é razoável pensar que esta penalização em troca de um comportamento mais controlado e um sistema menos complexo é aceitável.

Além disso, existiam dois itens especiais no modelo: ”Não aplicável” e ”Artista desconhecido”. Todas as notas para estes dois itens foram descartadas, uma vez que poderiam representar na verdade inúmeros artistas diferentes ou então simplesmente notas inválidas.

3.1.2 Distribuição das notas

A figura 3.1 contém a distribuição das notas do *dataset* após ser feito o tratamento descrito acima.

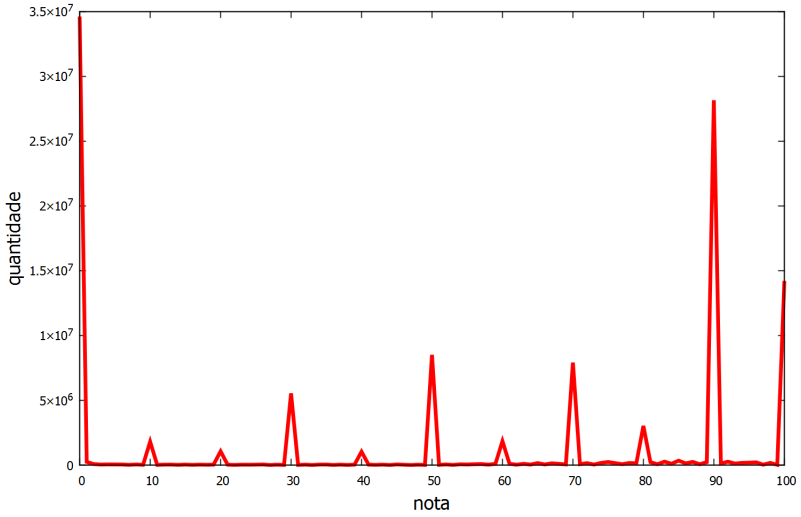

Figura 3.1: Distribuição das notas do *dataset*.

É possível perceber que a maioria das notas são números ”exatos”(múltiplos de

10). Além disso, a nota mais frequente é a nota 0 (até mesmo pelo fato de ter sido juntada com a nota 255).

As figuras 3.2 e 3.3 mostram respectivamente a distribuição das notas médias por usuário e por item no dataset. Elas são mostradas em faixas de notas, com exceção da nota 100 que não é agrupada em nenhuma faixa.

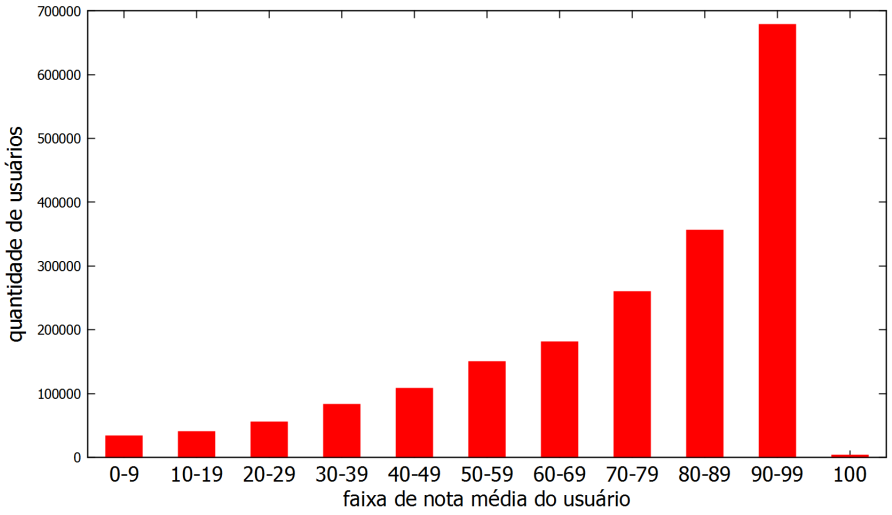

Figura 3.2: Distribuição das notas médias dos usuários.

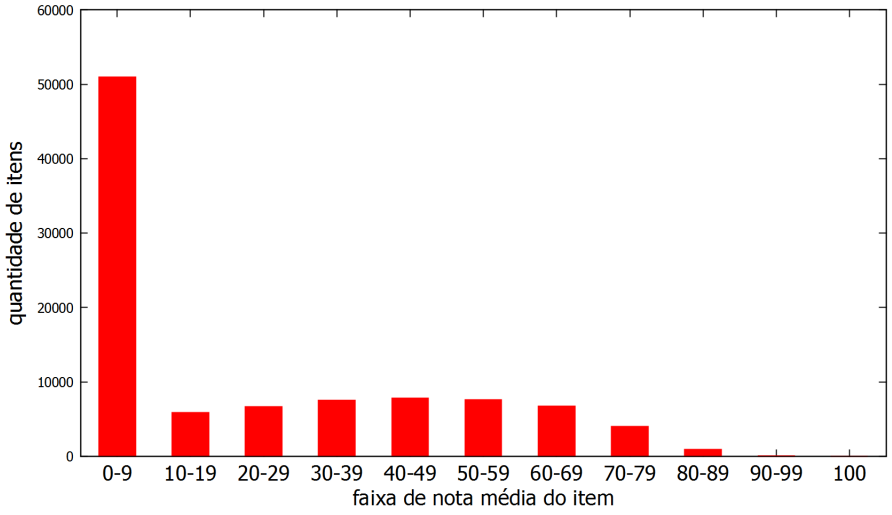

Figura 3.3: Distribuição das notas médias dos itens.

3.2 Computação em nuvem

A computação em nuvem é a entrega sob demanda de poder computacional, armazenamento de banco de dados, aplicações e outros recursos de TI por meio de uma plataforma de serviços de nuvem via Internet com uma definição de preço conforme o

uso [27]. A figura 3.4 ilustra os diferentes tipos de serviços que podem ser utilizados na nuvem.

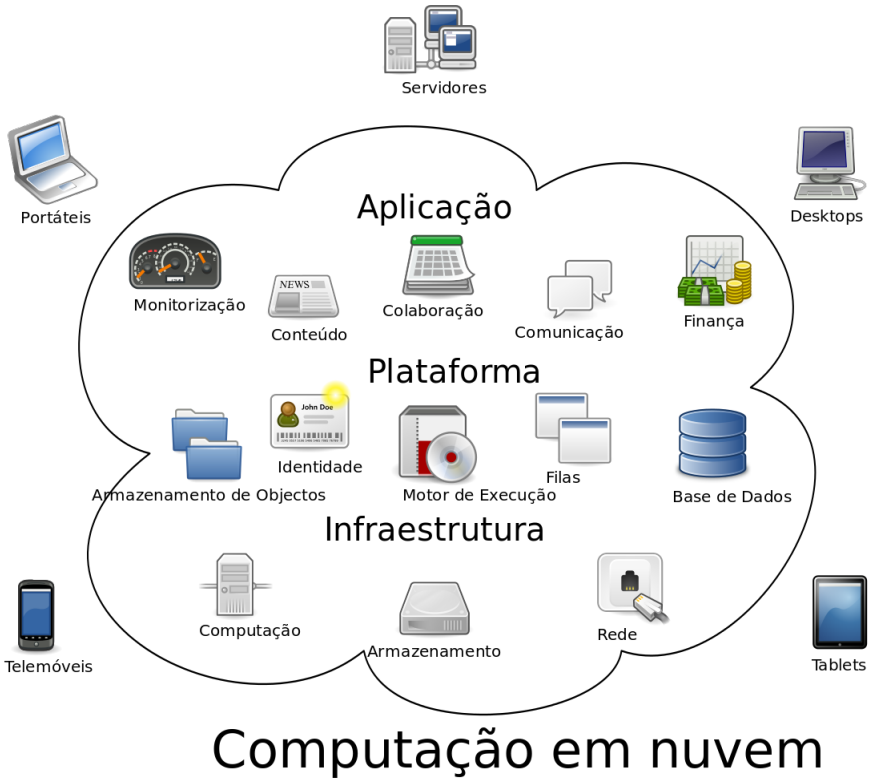

Figura 3.4: Representação dos diferentes tipos de serviços que podem ser utilizados com computação em nuvem [28].

Sendo assim, utilizar serviços de computação em nuvem é uma estratégia de baixo custo no curto prazo para tratar de problemas como, entre outros, limitações de infraestrutura. Ao utilizar a computação em nuvem, o contratante do serviço pode aumentar ou reduzir a capacidade computacional de acordo com a necessidade com um baixo tempo de reação - em alguns casos, em questão de minutos.

A computação em nuvem já se mostra uma boa alternativa para startups [29], onde o investimento inicial em infraestrutura poderia ser um empecilho em modelos tradicionais de negócios. Porém, esta estratégia não precisa ser utilizada apenas para fins comerciais: ela também pode ser utilizada para fins de pesquisa.

De fato, o tamanho do *dataset* apresentado na seção anterior junto com a grande complexidade inerente aos algoritmos de recomendação (que serão detalhados adiante) foram alguns desafios encontrados durante o trabalho. Estes são algoritmos intensivos tanto em memória quanto em processamento. Em alguns casos, os requisitos do computador local simplesmente não eram suficientes. Em outros, era interessante paralelizar os testes. Em ambos os casos, o serviço de computação em

nuvem da Amazon denominado *Amazon Web Services* (AWS) foi utilizado.

3.2.1 Amazon Web Services

AWS é a plataforma de serviços em nuvem da Amazon. Ela fornece serviços de computação redimensionável (Amazon EC2), armazenamento de objetos (Amazon S3), bancos de dados (Amazon RDS), distribuição de conteúdo (Amazon CloudFront), entre outros.

3.2.2 Amazon EC2

O Amazon Elastic Compute Cloud (Amazon EC2) é um serviço da web que fornece capacidade de computação redimensionável na nuvem [30]. Este serviço permite contratar em poucos minutos um ambiente computacional com componentes customizáveis, como sistema operacional, processador e memória RAM. A figura 3.5 mostra algumas das customizações que são feitas ao iniciar uma instância. Estas configurações podem ser alteradas a qualquer momento, de acordo com a necessidade do usuário.

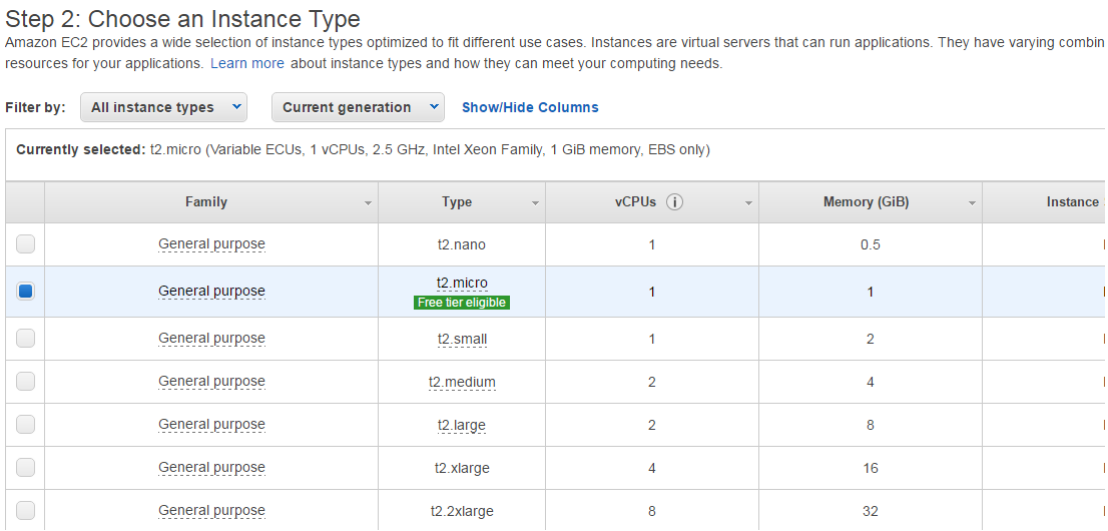

Figura 3.5: Exemplo de parte da configuração de instância no Amazon EC2.

O usuário possui controle pleno sobre a instância EC2, podendo instalar diversos componentes como bancos de dados (caso não queira utilizar o serviço fornecido pela Amazon) ou novas versões do ambiente Java. Todas as instâncias também possuem acesso ao armazenamento compartilhado de objetos denominado Amazon S3.

3.2.3 Amazon S3

O Amazon Simple Storage Service (Amazon S3) é o serviço de armazenamento de objetos na nuvem da Amazon. Este serviço pode ser acessado através de uma interface web ou através das próprias instâncias EC2 utilizando linhas de comando.

Cada nova versão do executável utilizado neste trabalho era enviada para o armazenamento do Amazon S3 através da interface web e posteriormente era consumida pela instância EC2 para realizar as novas baterias de testes. A interface web do Amazon S3 está ilustrada na figura 3.6, e a figura 3.7 mostra um exemplo de como copiar um arquivo hospedado no Amazon S3 através de linhas de comando para uma instância EC2.

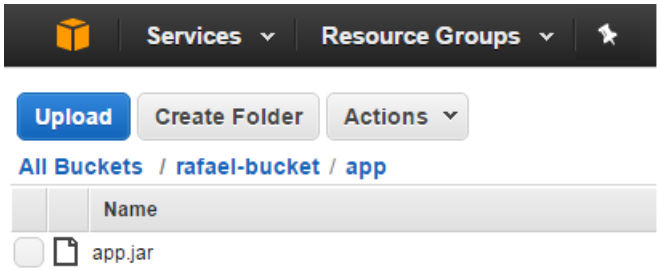

Figura 3.6: Interface web Amazon S3.

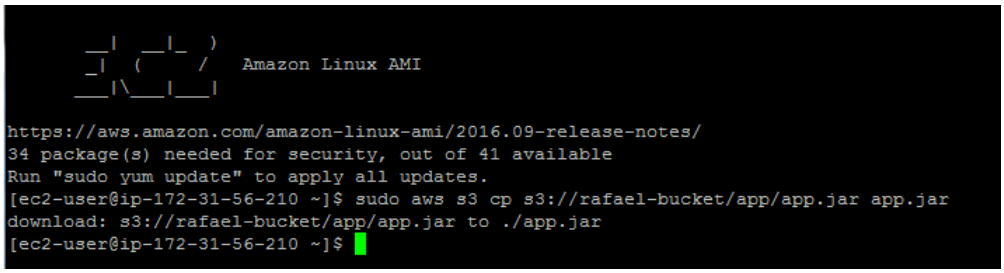

Figura 3.7: Acessando Amazon S3 através de uma instância EC2.

3.3 Apache Mahout

Apache Mahout (ou simplesmente Mahout) é uma biblioteca *open source* para a linguagem Java que fornece vários algoritmos de aprendizado de máquina (*machine learning*) com o objetivo de serem escaláveis e robustos [31].

O projeto Mahout nasceu em 2008 como um subprojeto do projeto Apache Lucene, que é um motor de buscas de texto. Como qualquer sistema de recuperação de informação, existia bastante sinergia com técnicas de aprendizado de máquina - em especial, técnicas de classificação e agrupamento (ou *clusterização*). Com o tempo, estas técnicas foram deslocadas para um projeto próprio que lidava apenas

com os algoritmos de aprendizado de máquina, e a este novo projeto (denominado Mahout) também foi incorporado o projeto Taste, que era um projeto *open source* de sistemas de recomendação de filtragem colaborativa [31].

A biblioteca Mahout, tendo como objetivo fornecer algoritmos escaláveis, desenvolveu algoritmos para serem executados com o Apache Hadoop, um projeto que fornece técnicas de computação e armazenamento distribuídos. Porém, uma vez que o objetivo deste trabalho era avaliar o desempenho de diversos sistemas de recomendação e que nem todos os algoritmos fornecidos pela biblioteca foram de fato desenvolvidos para um ambiente distribuído, a ferramenta Hadoop acabou não sendo utilizada.

Os algoritmos de recomendação implementados pela biblioteca Mahout são todos de filtragem colaborativa. Ou seja, alternativas baseadas em conteúdo ou híbridas não são suportadas nativamente, ficando por conta do desenvolvedor implementar estes tipos de sistema de recomendação.

O foco deste trabalho foi analisar os resultados produzidos por diferentes sistemas de recomendação utilizando a biblioteca Mahout. Muitos dos algoritmos que foram avaliados são disponibilizados nativamente pela biblioteca, enquanto que outros foram desenvolvidos como uma extensão das classes nativas do Mahout, em especial as técnicas de filtragem baseada em conteúdo que serão analisadas adiante.

3.4 Banco de dados MySQL

Um banco de dados é basicamente uma coleção de dados organizada de forma a ser facilmente indexada e recuperada por um computador [32] e armazenada de forma robusta. Um banco de dados está sempre associado a um sistema gerenciador de banco de dados, e, na prática, normalmente são utilizados como sinônimos, mesmo que o significado seja diferente. Um sistema gerenciador de banco de dados é o software responsável por se comunicar com o banco de dados para buscar, alterar e adicionar objetos, interagindo com o usuário e os sistemas que desejam utilizar as informações disponíveis no banco de dados.

A biblioteca Mahout possui uma classe específica para carregar o conjunto de dados a partir de um arquivo de texto. Além disso, quando é possível manter o conjunto de dados por inteiro em memória, existe um ganho computacional pois desta forma não existe o custo adicional de fazer constantemente acessos ao disco. Sendo assim, como a infraestrutura da nuvem (ver seção 3.2) é altamente escalável, não houve a necessidade de armazenar o conjunto de dados em si no banco de dados.

Porém, algumas informações derivadas do conjunto de dados - por exemplo: nota média por usuário, nota média por item, características extraídas dos itens (será explicado a seguir) etc - foram armazenadas em um banco de dados para que

não houvesse a necessidade de serem calculadas todas as vezes.

Para isto, foi utilizado o gerenciador de banco de dados MySQL. Na verdade, não existem motivos específicos para ter escolhido especificamente esta ferramenta e não outra semelhante. Por exemplo, comparando com o gerenciador PostgreSQL:

- ambos são *open-source*
- ambos utilizam modelos relacionais
- ambos utilizam a linguagem SQL, a linguagem padrão para consultas em bancos de dados relacionais
- ambos são facilmente integráveis com a linguagem Java
- ambos são facilmente integráveis com a biblioteca Mahout
- como as informações inseridas no banco de dados não foram tão numerosas, a possível diferença de performance não era um fator importante

Portanto, a escolha pelo MySQL foi meramente uma questão de familiaridade.

3.5 Spotify Web API

Uma API é um conjunto de funções que são disponibilizadas por um software e que podem ser utilizadas inclusive por aplicativos de terceiros.

Como foi comentado na seção 3.1, o *dataset* contém o nome dos artistas mas não contém nenhuma outra informação. Porém, seria interessante ter disponível uma lista de características dos artistas para que seja possível implementar um sistema de recomendação baseado em conteúdo, como foi discutido na seção 3.3 (por exemplo, gêneros musicais). Como esta lista de artistas foi retirada do serviço *Yahoo! Music*, o ideal seria utilizar alguma API deste mesmo serviço para extrair estas informações. Porém, nenhuma foi encontrada. Portanto, a Web API do Spotify - um famoso serviço de *streaming* de músicas - foi utilizada.

A Web API do Spotify é uma API acessível via Internet que fornece funções de busca musical, por exemplo: busca de música, de artista, de álbum etc. Esta API está disponível em [33]. O problema destas funções é que elas recebem como parâmetro o ID interno do item no Spotify, enquanto que só temos disponível o nome do artista. Existe ainda uma função mais genérica de busca de itens (que podem ser músicas, artistas, álbuns...) que recebe como parâmetro o nome do item e o tipo do item. No caso de busca de itens do tipo ”artista”, uma das informações retornadas é a lista de gêneros musicais. A figura 3.8 contém parte dos resultados retornados ao executar uma busca de itens do tipo ”artista” com o valor de consulta ”Pink Floyd”.

Esta consulta pode ser reproduzida acessando o seguinte endereço em qualquer navegador: "*https://api.spotify.com/v1/search?q=pink+floyd&type=artist*". Alguns atributos foram omitidos pois não são relevantes para este trabalho (como dimensões da foto do perfil do artista, por exemplo) e ocupariam muito espaço na figura.


```
{
  "artists" : {
    "items" : [ {
      "genres" : [ "album rock", "art rock", "classic rock", "hard rock", "mellow gold",
"progressive rock", "psychedelic rock", "rock", "space rock", "symphonic rock" ],
      "id" : "0k17h0D3J5VfsdmQ1iZtE9",
      "name" : "Pink Floyd",
    }, {
      "genres" : [ ],
      "id" : "0hioiM0MDloPdLjTE86zz8",
      "name" : "Pink Floyd Redux",
    }, {
      "genres" : [ ],
      "id" : "2tDvdcTW7H4vYW7S18MA2e",
      "name" : "The Pink Floyd Story",
    }, {
      "genres" : [ ],
      "id" : "0WhGOF7Zp2jKNwLkhnLQ2f",
      "name" : "Think Pink Floyd",
    }, {
```


Figura 3.8: Exemplo de informações retornadas pela Spotify Web API.

Utilizando esta API, foi possível capturar os gêneros musicais dos artistas. Na figura 3.8 podemos ver que o artista "Pink Floyd" está associado a diversos gêneros, como rock, rock progressivo, rock clássico, entre outros. Estas associações entre artista e gênero foram todas inseridas no banco de dados.

3.5.1 Escolha do artista

Na figura 3.8 também é possível notar que a consulta retornou mais de um resultado para o artista "Pink Floyd" (a figura foi editada para mostrar apenas 4 artistas diferentes, mas na verdade foram retornados vários outros). Na verdade, os itens "Pink Floyd Redux" e "Think Pink Floyd" são tributos à banda original que foram retornados devido à semelhança óbvia com o nome consultado.

Esta consulta ao artista "Pink Floyd" foi apenas um exemplo, mas serve para mostrar que, para extrair automaticamente características dos itens do *dataset*, é necessário tomar cuidado. Uma mesma consulta pode retornar diversos artistas diferentes, cada um com um gênero musical diferente (mesmo que isto seja improvável). Ainda existe a possibilidade de um mesmo artista possuir nomes diferentes no *Yahoo! Music* e no Spotify (por exemplo, devido a limites no tamanho do nome ou então na representação de números por algarismos ou por extenso). Extrair estas características manualmente é inviável, dado o grande número de artistas no *dataset*.

Para determinar qual dos itens é o que realmente está sendo pesquisado, o nome de cada artista retornado foi comparado com o termo pesquisado, e o item mais semelhante foi escolhido como o verdadeiro. Para esta métrica de semelhança, foi adotada a distância Levenshtein. No caso da figura 3.8, o artista escolhido seria o artista cujo nome é "Pink Floyd", pois dentre todas as opções este era o mais semelhante ao termo consultado - na verdade, era exatamente igual ao termo consultado.

A distância Levenshtein mede a diferença entre duas sequências de caracteres, e representa o menor número de operações (inserções, substituições ou exclusões) necessárias para transformar uma sequência na outra. Esta distância é normalmente calculada com técnicas de programação dinâmica. A distância Levenshtein entre duas cadeias de caracteres a e b de comprimentos respectivamente $|a|$ e $|b|$ é dada por $levenshtein_{a,b}(|a|, |b|)$, onde [34]

$$levenshtein_{a,b}(i,j) = \begin{cases} \max(i,j) & \text{se } \min(i,j) = 0 \\ \min \begin{cases} levenshtein_{a,b}(i-1,j) + 1 \\ levenshtein_{a,b}(i,j-1) + 1 \\ levenshtein_{a,b}(i-1,j-1) + 1_{(a_i \neq b_i)} \end{cases} & \text{caso contrário} \end{cases}$$

(3.1)

onde $levenshtein_{a,b}(i,j)$ é a distância entre os primeiros i caracteres de a e os primeiros j caracteres de b, e $1_{(a_i \neq b_i)}$ é a função indicadora que é igual a 0 se $a_i = b_j$ e igual a 1 caso contrário [34].

3.5.2 Pseudocódigo

Segue abaixo o pseudocódigo do algoritmo de captura automática de gêneros musicais para os artistas do *dataset*.

Um ponto interessante de se observar é que nem todos os artistas possuíam uma lista de gêneros musicais definida (em especial, os artistas com menos avaliações). Sendo assim, estes artistas não poderiam ser utilizados em um sistema de recomendação baseado em conteúdo, mas nada impede de serem utilizados em um sistema de recomendação de filtragem colaborativa.

Lista de artistas: a lista dos nomes de todos os artistas do *dataset*
Saída **:** estrutura de dados contendo a lista de gêneros por artista

```
for artista a em Lista de artistas do
    artista mais semelhante ← indefinido;
    menor distância ← infinito;
    artistas buscados ← BuscarArtistasApiSpotify(a);
    for ab em artistas buscados do
        distância ← DistanciaLevenshtein(a, ab);
        if distância < menor distância then
            artista mais semelhante ← ab;
            menor distância ← distância;
        end
    end
    gêneros ← PegarGeneros(artista mais semelhante);
    if gêneros não é vazio then
        gêneros(a)← gêneros
    end
end
```

Algoritmo 1: Captura de características dos itens

3.6 Conclusão

Este capítulo apresentou as ferramentas que foram utilizadas neste trabalho. Foram apresentadas as ferramentas de cunho tecnológico (como bibliotecas, infraestrutura, serviços etc) e também o conjunto de dados utilizados como base de avaliação. Também foi apresentada uma breve caracterização do conjunto de dados.

No caso específico da ferramenta Spotify Web API, a metodologia relacionada também foi apresentada por uma questão de sinergia com o conteúdo que foi apresentado.

As ferramentas descritas neste capítulo também são necessárias para a compreensão das metodologias e algoritmos utilizados no trabalho, que serão apresentados no próximo capítulo.

Capítulo 4

Metodologias e Algoritmos

Este capítulo irá detalhar as metodologias utilizadas no trabalho. Os algoritmos utilizados nos sistemas de recomendação (SR) e as técnicas de avaliação utilizadas serão apresentadas. Este capítulo estará fortemente relacionado com a biblioteca Mahout (ver seção 3.3), uma vez que ela foi a base de todos os sitemas de recomendação avaliados neste trabalho.

4.1 Divisão do conjunto de dados

Para este processo de avaliação de sistemas de recomendação, o *dataset* original foi dividido em 2:

1. *dataset* de treino - usado para aprender os parâmetros e montar o modelo que será utilizado para prever as notas. Os seus dados são exatamente iguais aos do *dataset* original. O *dataset* de treino é composto por 90% registros aleatórios do *dataset* original.

2. *dataset* de teste - é o conjunto de notas que não é utilizado para ajustar os parâmetros do sistema. Estas notas são utilizadas somente para fins de avaliação (o sistema tenta prever as notas contidas no *dataset* de teste). Composto pelos 10% restantes do *dataset* original.

Os mesmos conjuntos de treino e de teste foram utilizados em todas as baterias de testes.

É importante notar que no fundo as notas são conhecidas para o *dataset* de testes, porém elas não devem ser utilizadas no processo de previsão de nota. Elas são utilizadas para fins de avaliação: a nota prevista é comparada à nota original para assim determinar a precisão do algoritmo.

4.2 Métrica de avaliação

Existem diferentes métricas para avaliação da precisão de um sistema de recomendação. Estas métricas são normalmente divididas em dois grandes grupos: precisão das notas e precisão da classificação.

4.2.1 Precisão das notas

Como foi descrito na seção 2.1, um sistema de recomendação na verdade prevê um grau de satisfação que o usuário teria para cada item que ainda não foi visto por ele. No contexto do conjunto de dados utilizado neste trabalho, o grau de satisfação é a nota que o usuário explicitamente dá ao item. As métricas de precisão das notas avaliam a precisão destas notas previstas, ou seja, a nota prevista é comparada à nota original para que a proximidade entre elas seja avaliada.

Este tipo de métrica é de fato o mais utilizado em sistemas com preferências explícitas. As duas principais métricas de precisão de notas são o erro médio absoluto (*mean absolute error - MAE*) e a raiz do erro médio quadrático (*root mean square error - RMSE*), que são calculados da seguinte maneira:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|r_i - \tilde{r}_i| \tag{4.1}$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(r_i - \tilde{r}_i)^2} \tag{4.2}$$

onde n é o número total de pares (usuário,item) no *dataset* de teste, $\tilde{r}_i$ é a i-ésima nota prevista e r_i é a nota verdadeira correspondente. O valor de ambas as métricas é sempre não-negativo, e quanto menor o valor, mais preciso é o sistema.

A principal diferença entre estas duas métricas é que o RMSE penaliza fortemente as notas que são muito discrepantes pois as diferenças são elevadas ao quadrado, diferente do MAE que penaliza uniformemente as diferenças [16].

4.2.2 Precisão da classificação

A função das métricas de precisão da classificação é avaliar a relevância dos itens recomendados. Para isso, é necessário saber classificar um item em relevante ou não. Por este motivo, este tipo de métrica é mais comumente utilizado em sistemas com notas booleanas ou implícitas, mas também pode ser utilizado em sistemas com notas explícitas utilizando um *threshold* de nota (por exemplo, um item é considerado relevante para um usuário se sua nota for maior que 70). Neste caso, fica claro que a métrica passa a depender da escolha deste *threshold*.

Duas métricas muito utilizadas são a precisão e o *recall*. A precisão avalia qual é a fração dos itens recomendados que realmente é relevante, enquanto que o *recall* avalia qual é a fração de todos os itens relevantes que foi recomendada. A precisão $P(L)$ e o *recall* $R(L)$ referentes à lista de itens recomendados L para o usuário u são calculadas da seguinte maneira:

$$P(L) = \frac{relev_u(L)}{|L|} \tag{4.3}$$

$$R(L) = \frac{relev_u(L)}{|Relev_u|} \tag{4.4}$$

onde $relev_u(L)$ é o número de itens relevantes na lista de itens recomendados para o usuário u e $Relev_u$ é o conjunto de todos os itens relevantes para o usuário u [11]. O valor destas métricas está sempre entre 0 e 1, e quanto maior mais próximas de 1, melhor.

Existe ainda a métrica *F1-score* que combina os valores da precisão e do *recall* em uma única métrica, sendo calculada da seguinte forma:

$$F1 = \frac{2.P(L).R(L)}{P(L) + R(L)} \tag{4.5}$$

4.2.3 Escolha da métrica

Neste trabalho, a métrica utilizada foi o RMSE. Ela foi escolhida pelos seguintes motivos:

1. A maioria dos trabalhos da literatura que avaliam sistemas que utilizam preferências explícitas também utilizam uma métrica de precisão das notas [16] (em especial, normalmente é utilizada uma das duas métricas descritas aqui)

2. O peso maior dado a notas com muita discrepância quando comparado ao MAE foi considerado um fator positivo

4.3 Recomendação não-personalizada

Um sistema de recomendação não-personalizado produz recomendações sem levar em conta as preferências particulares do usuário. Normalmente, este tipo de sistema de recomendação sugere apenas os itens mais vendidos. É um método bem simples de produzir recomendações, porém nem sempre muito eficaz.

Neste trabalho, foi utilizado um método de recomendação não-personalizada para servir de *benchmark*: a nota média do item.

4.3.1 Nota média do item

Este método de recomendação é implementado pela biblioteca Mahout pela classe *ItemAverageRecommender*. A nota prevista para um usuário u para o item i é simplesmente a nota média global do item i.

4.3.2 Nota média do item ajustada pela nota média do usuário

Por definição, este método de recomendação na verdade é considerado personalizado. Porém, devido à simplicidade e à semelhança com o método descrito acima, ele foi colocado nesta seção.

Neste método, a nota prevista é a nota média global do item i ajustada pela nota média dada pelo usuário u. Por exemplo:

- a nota média do item i é 65
- a nota média do usuário u é 55
- a nota média global (ou seja, média de todas as notas de todos os usuários para todos os itens) é 50

Neste caso, a nota média do usuário u é 5 unidades maior que a nota média global. Sendo assim, a nota prevista para o item i seria igual a $65 + 5 = 70$.

Este método de recomendação é implementado pela biblioteca Mahout na classe *ItemUserAverageRecommender*.

4.3.3 Nota média do usuário

Seguindo o mesmo raciocínio, este método também produz recomendações personalizadas. Ele também foi colocado aqui devido à simplicidade e à semelhança com o método de nota média do item.

Neste caso, a nota prevista pelo sistema de recomendação para um usuário u para o item i é simplesmente a nota média do usuário u.

Na prática, este método de previsão de notas dificilmente é útil, pois a nota prevista para um dado usuário será sempre a mesma independente do item. Desta forma, não é possível gerar uma lista de itens recomendados ordenada por odem decrescente de preferência.

Este método de recomendação é implementado pela biblioteca Mahout na classe *UserAverageRecommender*.

4.4 Filtragem colaborativa baseada em memória

Como foi apresentado na seção 2.2, a filtragem colaborativa pode ser dividida em dois tipos: baseada em memória e baseada em modelos. O primeiro tipo também pode ser chamado de "baseado em similaridade", pois ele depende de encontrar usuários ou itens semelhantes entre si. Naturalmente, por se tratar de filtragem colaborativa, esta similaridade deve ser calculada tomando como base apenas as notas que os usuários deram para o conjunto de itens.

Os sistemas de recomendação de filtragem colaborativa baseados em similaridade podem ser divididos ainda em 2 grandes tipos, de acordo com a forma como calculam as similaridades: baseados em usuários ou baseados em itens.

4.4.1 Baseados em usuários

Os SR de filtragem colaborativa baseados em usuários (*user-based - UB*), ou simplesmente SR baseados em usuários, calculam as similaridades entre os usuários.

Para prever a nota do usuário u para o item i, o SR utiliza uma vizinhança de usuários em torno de u. Esta vizinhança pode ser simplesmente todos os outros usuários, somente os K usuários mais próximos ou então pode ser obtida através de algum outro método. Seja esta vizinhança denominada V_u. O SR então utiliza os usuários em V_u que também deram notas para o item i e combina estas notas através de uma média ponderada pela similaridade com o usuário u.

A nota do usuário u prevista para o item i $prev(u, i)$ pode ser escrita da seguinte maneira:

$$prev(u,i) = \frac{\sum_{v \in V_{u,i}} sim(u,v) \times r_{v,i}}{\sum_{v \in V_{u,i}} sim(u,v)} \tag{4.6}$$

onde $V_{u,i}$ corresponde aos usuários na vizinhança V_u que deram nota ao item i, $sim(u, v)$ é a similaridade entre o usuário u e seu vizinho v e $r_{v,i}$ é a nota que o vizinho v deu ao item i.

Este método de recomendação é implementado pela biblioteca Mahout na classe *GenericUserRecommender*.

Na verdade, a recomendação baseada em usuários não foi utilizada neste trabalho devido ao grande número de usuários (1.948.882). Seria necessário calcular a similaridade entre todos os pares de usuários, o que significaria calcular e armazenar mais de 1 trilhão de valores de similaridade.

4.4.2 Baseados em itens

O SR de filtragem colaborativa baseada em itens (*item-based - IB*) , ou simplesmente SR baseado em itens, funciona de forma muito semelhante ao SR baseado em usuários. A principal diferença é que este utiliza a similaridade entre itens ao invés de usuários.

O algoritmo de recomendação é análogo ao anterior. Para prever a nota do usuário u para o item i, o SR utiliza uma vizinhança de itens em torno do item i. Normalmente esta vizinhança é o conjunto de todos os itens, mas também pode ser empregada alguma técnica de filtragem como escolher somente os K itens mais próximos ou então os itens com valor de similaridade a partir de um limiar. Seja esta vizinhança denominada V_i. Desta vez, a nota prevista $prev(u, i)$ é dada por

$$prev(u,i) = \frac{\sum_{v \in V_{i,u}} sim(i,v) \times r_{u,v}}{\sum_{v \in V_{i,u}} sim(i,v)} \tag{4.7}$$

onde $V_{i,u}$ corresponde aos itens na vizinhança V_i para os quais o usuário u deu nota, $sim(i, v)$ é a similaridade entre o item i e seu vizinho v e $r_{u,v}$ é a nota que o usuário u deu ao item vizinho v.

Este método de recomendação é implementado pela biblioteca Mahout na classe *GenericItemRecommender*. Esta classe não é preparada para configurar uma vizinhança que não seja o conjunto de todos os itens. Caso o desenvolvedor queira filtrar a vizinhança (por exemplo, escolhendo os K itens mais próximos) ele deve estender esta classe com as alterações necessárias. Uma das variações que foram implementadas neste trabalho foi um sistema de recomendação baseado em itens que só considera na vizinhança os itens que possuem similaridade maior ou igual a um limiar configurável. Este sistema foi testado com valores diferentes para este limiar para avaliar como a nota prevista se comportava.

Os SR baseados em itens possuem duas grandes vantagens com relação aos baseados em usuários:

1. Normalmente o número de itens é menor que o número de usuários e, portanto, são necessários menos cálculos de similaridade

2. Os itens são em geral mais estáticos, ou seja, a similaridade entre dois itens diferentes costuma não variar muito. Isto significa que as similaridades podem ser calculadas *offline*, ou seja, elas podem ser calculadas previamente [16]

4.4.3 Cálculo da similaridade

Como foi visto até aqui, o cálculo da similaridade é um componente fundamental dos sistemas de recomendação baseados em itens ou em usuários. Porém, esta

similaridade pode ser calculada de diferentes maneiras. Portanto, é de se esperar que medidas de similaridade diferentes produzam previsões de notas diferentes.

Ainda assim, não existe uma medida que é necessariamente sempre melhor que as demais. Algumas delas são aplicáveis apenas em algumas situações específicas.

As medidas de similaridade utilizadas neste trabalho serão detalhadas a seguir.

Coeficiente de correlação Pearson

A correlação Pearson (ou similaridade Pearson) é um número entre -1 e +1 que mede a tendência de duas séries emparelhadas de números se moverem juntas, ou seja, a tendência de um número em uma série ser alto quando o número correspondente na outra série também for alto e vice-versa [31]. Quanto mais próximo de +1, mais correlacionados os números estão.

Esta similaridade é normalmente utilizada em SR baseados em usuários, mas também pode ser utilizada em SR baseados em itens.

A similaridade Pearson entre dois itens a e b é calculada da seguinte maneira:

$$sim(a,b) = \frac{\sum_{u \in U}(r_{u,a} - \bar{r_a})(r_{u,b} - \bar{r_b})}{\sqrt{\sum_{u \in U}(r_{u,a} - \bar{r_a})^2}\sqrt{\sum_{u \in U}(r_{u,b} - \bar{r_b})^2}} \tag{4.8}$$

onde U é o conjunto de usuários que deram nota ao mesmo tempo para os itens a e b, $r_{u,a}$ e $r_{u,b}$ são respectivamente as notas que o usuário u deu para os itens a e b, e $\bar{r_a}$ e $\bar{r_b}$ são respectivamente as notas médias dos itens a e b.

Analogamente, a similaridade Pearson entre dois usuários u e v é calculada da seguinte maneira:

$$sim(u,v) = \frac{\sum_{i \in I}(r_{u,i} - \bar{r_u})(r_{v,i} - \bar{r_v})}{\sqrt{\sum_{i \in I}(r_{u,i} - \bar{r_u})^2}\sqrt{\sum_{i \in I}(r_{v,i} - \bar{r_v})^2}} \tag{4.9}$$

onde I é o conjunto de itens que receberam nota ao mesmo tempo dos usuários u e v, $r_{u,i}$ e $r_{v,i}$ são respectivamente as notas que os usuários u e v deram para o item i, e $\bar{r_u}$ e $\bar{r_v}$ são respectivamente as notas médias dos usuários u e v.

A biblioteca Mahout implementa a similaridade Pearson na classe *PearsonCorrelationSimilarity*.

Similaridade cosseno

A similaridade cosseno também assume valores entre -1 e +1. Ela é utilizada para calcular o cosseno do ângulo entre dois vetores. No caso dos sistemas de recomendação, os vetores são compostos pelas notas dos usuários (ou, vendo pelo ponto de vista de um SR baseado em itens, pelas notas que os itens receberam). Quando dois vetores são colineares, ou seja, possuem exatamente a mesma direção, o cos-

seno do ângulo entre eles é igual a 1 (já que o cosseno de 0° é igual a 1). Portanto, quando os dois vetores de notas são muito semelhantes, a similaridade cosseno entre eles tende a +1.

A similaridade cosseno entre dois itens a e b é calculada da seguinte maneira:

$$sim(a,b) = \frac{\sum_{u \in U} r_{u,a} \times r_{u,b}}{\sqrt{\sum_{u \in U} r_{u,a}^2} \sqrt{\sum_{u \in U} r_{u,b}^2}} \tag{4.10}$$

onde U é o conjunto de usuários que deram nota ao mesmo tempo para os itens a e b, e $r_{u,a}$ e $r_{u,b}$ são respectivamente as notas que o usuário u deu para os itens a e b.

O cálculo de similaridade representado na equação 4.10 é implementado pela biblioteca Mahout na classe *UncenteredCosineSimilarity*.

Como todas as notas do conjunto de dados variam entre 0 e 100, na verdade a equação 4.10 produz valores de similaridade que variam entre 0 e 1. Se os dados fossem centralizados (ou seja, ajustados de forma que a média deles seja 0) isto não aconteceria. Porém, para um conjunto de dados centralizado, a similaridade cosseno é equivalente à similaridade Pearson. Além disso, a implementação da similaridade Pearson da biblioteca Mahout centraliza os dados [31]. Portanto, na prática ambas as classes calculam o ângulo entre os vetores de notas, e a possível diferença de resultados se deve à centralização ou não dos dados.

Similaridade *Log-Likelihood*

A similaridade *log-likelihood* é uma métrica de surpresa e coincidência que foi proposta em [35]. Esta medida de similaridade na verdade não utiliza os valores das notas, mas sim a ocorrência de nota para dois itens/usuários em comum.

A ideia por trás dessa medida de similaridade é determinar o quão improvável é o fato de dois usuários terem dado nota para um mesmo item apenas por sorte. Por exemplo, se os usuários U e V deram notas para 3 itens em comum e ao total deram nota para 5 itens cada, eles são considerados mais semelhantes do que se tivessem ao total dado nota para 100 itens cada [31].

Lembrando que, mesmo que as notas tenham sido completamente diferentes, o único fator que é levado em consideração é a ocorrência ou não de notas para itens em comum. Portanto, esta medida de similaridade pode ser mais indicada para preferências implícitas do tipo "usuário comprou/não comprou o item" ou preferências booleanas do tipo "usuário gostou/não gostou do item".

Para calcular a similaridade *log-likelihood* entre dois itens a e b, é necessário considerar a quantidade de vezes que os eventos ocorreram ao mesmo tempo - no caso, um evento significa um usuário ter dado nota para o item. Considere as variáveis representadas na tabela 4.1. Esta tabela pode ser vista como uma matriz

K.

Tabela 4.1: Variáveis necessárias para calcular a similaridade *log-likelihood* [36]

	Item a	**Outro Item**
Item b	Quantidade de usuários que deram notas para a e b ($k_{1,1}$)	Quantidade de usuários que deram nota somente para b ($k_{1,2}$)
Outro Item	Quantidade de usuários que deram nota somente para a ($k_{2,1}$)	Quantidade de usuários que não deram notas pra nenhum dos dois ($k_{2,2}$)

Note que a soma de todos os elementos de K é dada por $k_{1,1} + k_{1,2} + k_{2,1} + k_{2,2} = |U| = N_u$, ou seja, N_u é a quantidade total de usuários no *dataset*. Considere ainda as 3 seguintes matrizes (respectivamente, matrizes K, $rowSums(K)$ e $colSums(K)$):

$k_{1,1}$	$k_{1,2}$
$k_{2,1}$	$k_{2,2}$

$k_{1,1} + k_{1,2}$
$k_{2,1} + k_{2,2}$

$k_{1,1} + k_{2,1}$	$k_{1,2} + k_{2,2}$

Neste caso, a razão *log-likelihood*, denotada por RLL, é dada pela seguinte equação [36]:

$$RLL = 2N_u \times (H(K) - H(rowSums(K)) - H(colSums(K))) \tag{4.11}$$

onde a função H é a entropia de Shannon, que é dada pela seguinte equação [36]:

$$H(K) = \sum_i \sum_j \frac{k_{i,j}}{N_u} \times log(\frac{k_{i,j}}{N_u}) \tag{4.12}$$

Tendo agora o valor de RLL, a similaridade *log-likelihood* entre os itens a e b é dada por [37]

$$sim(a, b) = 1 - \frac{1}{1 + RLL} \tag{4.13}$$

Os conceitos foram descritos para uma similaridade baseada em itens, mas podem ser facilmente adaptados para calcular a similaridade entre usuários.

O cálculo da similaridade *log-likelihood* é implementado pela biblioteca Mahout na classe *LogLikelihoodSimilarity*.

4.5 Filtragem colaborativa baseada em modelos

O método de filtragem colaborativa baseada em modelos é comumente chamado de modelo de fatores latentes, pois procura utilizar o conjunto de notas para inferir

um modelo que representa os itens e os usuários por um conjunto de fatores [7]. Estes fatores podem representar características relativamente óbvias (por exemplo o gênero de um filme) ou então podem não possuir nenhuma interpretação clara, e por isso são chamados de latentes (ocultos) [16].

Esta abordagem se mostrou muito promissora principalmente a partir do Prêmio Netflix, já que os dois algoritmos vencedores do concurso tinham como um de seus principais componentes uma técnica de recomendação baseada em fatores latentes [15].

Um dos métodos de fatores latentes mais utilizados é a fatoração de matrizes [7]. A técnica de fatoração matricial mais empregada para o contexto de sistemas de recomendação é semelhante à decomposição em valores singulares (*singular value decomposition - SVD*), que é uma técnica bem-sucedida e já bem estabelecida para a captura de fatores latentes em sistemas de recuperação de informação[7][16].

A técnica SVD decompõe uma matriz R de dimensão $N_u \times N_i$ em um produto de matrizes, tal que

$$R = U\Sigma V \tag{4.14}$$

onde U é uma matriz de dimensão $N_u \times N_u$, Σ é uma matriz diagonal de dimensão $N_u \times N_i$ contendo os valores singulares de R, e V é uma matriz de dimensão $N_i \times N_i$.

No contexto de sistemas de recomendação, a matriz R seria a matriz de utilidades contendo as notas dos usuários aos itens. Sendo assim, N_u representa o número de usuários e N_i representa o número de itens do *dataset*.

A decomposição SVD também pode servir como base para a redução de dimensionalidade. Ao invés de utilizar a matriz Σ completa, pode-se utilizar uma matriz Σ' que contém apenas os f primeiros valores singulares (normalmente os maiores), utilizando somente as colunas correspondentes de U e V. Para simplificar a notação, podemos ainda incorporar a matriz Σ' em uma das outras duas, obtendo assim a matriz

$$R' = PQ \tag{4.15}$$

onde P é uma matriz de dimensões $N_u \times f$ e Q é uma matriz de dimensões $f \times N_i$. Note que f neste caso é arbitrário. Considere ainda que N_u e N_i podem ser arbitrariamente grandes (de fato, como descrito na seção 3.1, este trabalho lida com um conjunto de dados com um número grande de itens e um número ainda maior de usuários). Sendo assim, quando $f < \min(N_u, N_i)$, a matriz R' - que é uma matriz potencialmente bem grande - passa a ser representada por duas matrizes P e Q de dimensões arbitrariamente menores, reduzindo assim a complexidade computacional e o custo de armazenamento [11]. Por isso, este método também é conhecido como

redução de dimensionalidade.

Isto significa que a matriz P é uma matriz que modela os usuários através de f fatores (onde cada linha p_i da matriz modela o usuário i por um vetor de f elementos) e a matriz Q modela os itens através dos mesmos f fatores (onde cada coluna q_j da matriz modela o item j por um vetor de f elementos).

Porém, na prática a matriz de utilidade R é uma matriz esparsa, dado que os usuários normalmente não avaliam muitos itens. Portanto, existem vários elementos em R que estão faltando, e a técnica SVD na verdade não é definida para uma matriz incompleta. Sendo assim, o objetivo do sistema de recomendação passa a ser encontrar uma matriz de utilidade $\tilde{R} = PQ$ que aproxima R' (ou seja, mantendo a redução de dimensionalidade) e que seja totalmente preenchida. Cada elemento desta matriz é a nota aproximada $\tilde{r}_{i,j}$ do usuário i para o item j, que é calculada pelo produto interno de dois vetores

$$\tilde{r}_{i,j} = p_i^T q_j \tag{4.16}$$

onde p_i representa o usuário i através de f fatores e q_j representa o item j através de f fatores. Cada elemento de q_j (que pode ser um número positivo ou negativo) representa o quanto aquele item possui aquele fator específico, enquanto que cada elemento de p_i (que também pode ser positivo ou negativo) representa o quanto o usuário se interessa por aquele fator específico [7].

Para obter a aproximação $\tilde{R}$, utiliza-se como função objetivo a norma de Frobenius da matriz $E = R - \tilde{R}$ considerando apenas os elementos de R para os quais se sabe o valor. Cada elemento $e_{i,j}$ da matriz E é dado por

$$e_{i,j} = r_{i,j} - \tilde{r}_{i,j} \tag{4.17}$$

A função objetivo é minimizada, obtendo assim os parâmetros de $\tilde{R}$ (ou seja, parâmetros de P e Q) que melhor aproximam a matriz de utilidade original. Note também que minimizar a norma de uma matriz é equivalente a minimizar a sua norma quadrática, já que a norma por definição é não-negativa, e minimizar a norma quadrática resulta em expressões mais simples [11]. Portanto, deve-se minimizar a seguinte expressão:

$$\min ||E||^2 = \min ||R - \tilde{R}||^2 = \min \sum_{i,j} (r_{i,j} - \tilde{r}_{i,j})^2 = \min \sum_{i,j} (r_{i,j} - p_i^T q_j)^2 \tag{4.18}$$

Para evitar o sobreajuste do modelo (ou seja, modelo exageradamente especializado para o conjunto de dados disponível), TAKÁCS *et al.*[38] sugere que o erro regularizado seja minimizado, que é equivalente a minimizar a expressão

$||R - \tilde{R}||^2 + \lambda||P||^2 + \lambda||Q||^2$, onde λ é o parâmetro de regularização [11]. Logo, a expressão que deve ser minimizada passa a ser

$$\min \sum_{i,j} (r_{i,j} - \tilde{r}_{i,j})^2 + \lambda(||p_i||^2 + ||q_j||^2) \tag{4.19}$$

O valor de λ deve ser não-negativo, com $\lambda = 0$ significando uma expressão não-regularizada. Este parâmetro é normalmente determinado por técnicas como a validação cruzada [7], e este trabalho avaliou o impacto que a escolha do valor de λ possui na precisão do sistema.

Como os valores $r_{i,j}$ são dados e λ é um parâmetro definido, minimizar o erro significa encontrar P e Q que minimizam o erro. Para isto, são utilizados principalmente 2 métodos: o método do gradiente estocástico e o método dos mínimos quadrados alternados.

4.5.1 Método do gradiente estocástico

O método do gradiente estocástico (*stochastic gradient descent - SGD*) é uma técnica de otimização iterativa para uma função objetivo. No contexto de sistemas de recomendação, ela é empregada para minimizar a equação 4.19, encontrando assim os parâmetros de P e Q. Apesar de ser uma técnica relativamente antiga, ela foi popularizada para o contexto de sistemas de recomendação a partir do Prêmio Netflix, quando FUNK[39] descreveu como ela poderia ser utilizada para obter a matriz de utilidade considerando apenas o conjunto de notas conhecidas.

Este método primeiramente inicializa as matrizes P e Q. Esta inicialização pode ser feita por exemplo com valores aleatórios ou então com a nota média global em todas as posições. Feito isto, o algoritmo itera sobre cada uma das notas $r_{i,j}$ do *dataset* de treino. Para cada uma delas, a nota prevista associada $\tilde{r}_{i,j}$ é calculada através da equação 4.16 (lembre-se que P e Q já foram inicializadas, então é possível calcular a nota prevista). Tendo agora o erro de aproximação $e_{i,j}$, o vetor de fatores do usuário p_i e o vetor de fatores do item q_j são modificados na direção contrária do gradiente de $e_{i,j}$ através das expressões

$$p_i \leftarrow p_i + \gamma(e_{i,j} q_j - \lambda p_i) \tag{4.20}$$

$$q_j \leftarrow q_j + \gamma(e_{i,j} p_i - \lambda q_j) \tag{4.21}$$

onde λ é o parâmetro de regularização utilizado na equação 4.19 e γ é a taxa de aprendizado, que é um parâmetro que define o tamanho do passo que os vetores p_i e q_j irão percorrer [7].

Note que na segunda iteração o sistema de recomendação já calcula a nota prevista $\tilde{r}$ utilizando o valor atualizado dos vetores p e q, e assim por diante.

A cada *loop* completo sobre as notas do *dataset* de treino, a equação 4.19 é avaliada. Este processo pode então recomeçar e continuar até que o resultado da equação 4.19 não seja mais alterado, mas normalmente é definido um número máximo de iterações n_{it}. Este limite de iterações, além de agilizar o término do algoritmo, pode evitar problemas de *overfitting*. Porém, a inicialização das matrizes P e Q passa a ser um fator determinante, já que a forma como elas foram inicializadas irá interferir diretamente em como elas estarão depois de exatas n_{it} iterações [39].

O algoritmo 2 contém o pseudocódigo do método do gradiente estocástico.

Entrada: *Dataset* de treino
Saida : Matrizes P e Q utilizadas na fatoração SVD

$P \leftarrow$ inicialização;
$Q \leftarrow$ inicialização;
for *iter = 1 até* n_{it} **do**
  for *nota* $r_{i,j}$ *no dataset de treino* **do**
    $\tilde{r}_{i,j} \leftarrow$ `PreverNota(`i,j`)`;
    $e_{i,j} \leftarrow r_{i,j} - \tilde{r}_{i,j}$;
    $p_i \leftarrow$ `AtualizarVetorUsuario(`p_i,q_j,$e_{i,j}$,γ,λ`)`;
    $q_j \leftarrow$ `AtualizarVetorItem(`p_i,q_j,$e_{i,j}$,γ,λ`)`;
    $P \leftarrow$ `AtualizarMatriz(`P,p_i`)`;
    $Q \leftarrow$ `AtualizarMatriz(`Q,q_j`)`;
  end
  Avaliar equação 4.19 ;
end

Algoritmo 2: Pseudocódigo SGD

4.5.2 Mínimos quadrados alternados

A equação 4.19 possui duas variáveis: p_i e q_j. Por este motivo, ela não é convexa[7] (ou seja, ao minimizar a função não é garantido que seja obtido um mínimo global, mas sim um mínimo local).

A estratégia do método dos mínimos quadrados alternados é fixar uma destas duas variáveis de forma alternada. Dessa forma, o problema de otimização passa a ser quadrático e pode ser minimizado de forma ótima[7].

Em outras palavras, o algoritmo começa fixando os valores para o vetor p_i. Feito isto, o sistema calcula os valores para o vetor q_j que minimizam a equação 4.19. Após isto, o sistema fixa valores para q_j e calcula p_i. Sempre que um vetor é fixado, o outro é calculado por um problema de otimização de função quadrática. Sendo assim, também costuma-se definir um número máximo de iterações, mas que desta

vez possui um significado diferente: ele representa a quantidade de vezes que um vetor será fixado e o outro será calculado.

Este método costuma ser mais lento do que o método do gradiente estocástico, mas pode ser vantajoso no caso em que o *dataset* não é esparso, já que o SGD deve iterar sobre todo o conjunto de treino. Isto costuma acontecer em modelos de preferências implícitas[7].

4.5.3 Tratamentos adicionais na previsão

O método de fatoração de matrizes ainda permite que outras informações adicionais sejam incorporadas na nota prevista $\tilde{r}_{i,j}$. A principal é o viés (mais comumente chamado de *bias*, que é sua tradução para a língua inglesa).

O *bias* captura tendências particulares do item ou do usuário. A equação 4.16 tenta prever uma nota a partir das interações entre usuários e itens, mas as tendências dos usuários/itens também podem ter um impacto expressivo na nota. Por exemplo, um usuário pode se sentir mais inclinado a avaliar positivamente um filme que tenha sido aclamado pela mídia como um dos melhores da década, ou então um usuário u pode ser muito mais propenso a dar notas altas quando comparado com um usuário v.

O *bias* do usuário $b_u(i)$ indica o desvio da nota média do usuário i em relação à média global (ou seja, a média de todas as notas de todos os usuários para todos os itens, denotada por μ), e o *bias* do item $b_i(j)$ indica o desvio da nota média do item j em relação à média global [7].

De fato, muitos sistemas de recomendação que utilizam a técnica do SVD incorporam os *biases* do item e do usuário na previsão da nota[40][41][42], substituindo a equação 4.16 por

$$\tilde{r}_{i,j} = p_i^T q_j + b_u(i) + b_i(j) + \mu \tag{4.22}$$

Neste caso, o objetivo do sistema deixa de ser minimizar a função 4.19 e passa a ser minimizar a seguinte função [7]:

$$\min \sum_{i,j} (r_{i,j} - \tilde{r}_{i,j})^2 + \lambda(||p_i||^2 + ||q_j||^2 + b_u(i)^2 + b_i(j)^2) \tag{4.23}$$

Os métodos de otimização permanecem aplicáveis.

4.5.4 Modelo de fatores latentes na biblioteca Mahout

A biblioteca Mahout também utiliza a fatoração SVD para produzir recomendações no modelo de fatores latentes. Para isto, é necessário utilizar a classe *SVDRecommender*. Um dos argumentos do construtor desta classe é um objeto do tipo *Facto-*

rizer, que na verdade determina o algoritmo utilizado na minimização da função de custo. Ambos os algoritmos descritos aqui são implementados na biblioteca: método do gradiente estocástico e mínimos quadrados alternados.

O gradiente estocástico é implementado na biblioteca Mahout pela classe *RatingSGDFactorizer*. Esta classe minimiza a função de custo exatamente como foi descrita em 4.23, considerando os *biases* do usuário e do item na previsão da nota. Existe ainda uma implementação paralela deste mesmo algoritmo na classe *ParallelSGDFactorizer*, baseado no que foi descrito em [43].

Já o método dos mínimos quadrados alternados é implementado na classe *ALSWRFactorizer*. Esta, por sua vez, não considera os efeitos dos *biases* de item e de usuário na previsão da nota. Sendo assim, a função de custo minimizada é a equação 4.19. Na verdade, algumas outras implementações deste método que considerem o efeito do *bias* foram procuradas na literatura (como em [44], [45] ou na análise feita em [46]) mas não foram encontradas.

4.6 Filtragem baseada em conteúdo

Como já foi dito na seção 3.3, a biblioteca Mahout não implementa nenhum algoritmo de recomendação baseado em conteúdo. Portanto, todas as abordagens utilizadas aqui foram implementadas por este trabalho. Em especial, os métodos de recomendação utilizados aqui foram propostos por este trabalho.

Para que seja possível comparar de forma objetiva as recomendações produzidas pela filtragem colaborativa e a filtragem baseada em conteúdo através da métrica descrita na seção 4.2, esta última também deve produzir como resultado uma previsão de nota para os itens.

Os algoritmos de recomendação baseada em conteúdo utilizados neste trabalho foram desenvolvidos com a intenção de mitigar o problema de *cold-start* de usuário. Ou seja, eles foram desenvolvidos de forma que consigam recomendar itens para um novo usuário que nunca deu nota para nenhum item. Porém, eles não mitigam o problema de *cold-start* de item, pois todas as notas previstas dependem de alguma forma da nota média do item para o qual a nota está sendo prevista.

Como foi descrito na seção 2.3, a filtragem baseada em conteúdo depende de dois componentes principais: o conjunto de características dos itens e o perfil do usuário.

A extração automática de características de músicas para um sistema de recomendação baseado em conteúdo é um tema que já foi discutido em alguns artigos da literatura [47][48]. Entre as alternativas propostas, está a extração das MFCCs(*Mel-frequency Cepstrum Coefficient*)[49], que são características do espectro do sinal acústico.

Porém, o processo de extração de características dos itens adotado neste trabalho

foi mais simples. Ele utilizou a API do Spotify que foi apresentada na seção 3.5. As características escolhidas foram os gêneros musicais dos artistas, pois estas eram as características mais relevantes disponíveis através da API.

Já o perfil do usuário foi determinado através de um padrão de níveis de preferência inferidos a partir de suas notas, através de um procedimento que será detalhado a seguir.

4.6.1 Extração de características dos itens

O processo de extração de características dos itens já foi detalhado na seção 3.5. Isto resultou em um total de 1.443 gêneros musicais diferentes. Dos 98.211 artistas do *dataset*, somente 25.181 possuíam algum gênero listado. Isto significa que o sistema de recomendação baseado em conteúdo só é capaz de recomendar estes 25.181 artistas.

Os itens foram representados pelo modelo de espaço vetorial. Seja N_g o número de gêneros. Cada item i foi representado por um vetor $I_i \in \mathbb{R}^{N_g}$ tal que

$$I_i = [g_1, g_2, ..., g_{N_g}], \quad g_j = \begin{cases} 0 & \text{se o item } i \text{ não está associado ao gênero } j \\ 1 & \text{se o item } i \text{ está associado ao gênero } j \end{cases} \tag{4.24}$$

4.6.2 Extração de perfil do usuário

Os usuários também foram representados pelo modelo de espaço vetorial. As suas preferências foram classificadas em 3 níveis: "usuário gosta do gênero", "usuário é indiferente ao gênero"e "usuário não gosta do gênero".

Cada usuário i foi representado por um vetor $U_i \in \mathbb{R}^{N_g}$ tal que

$$U_i = [g_1, g_2, ..., g_{N_g}], \quad g_j = \begin{cases} -1 & \text{se o usuário } i \text{ não gosta do gênero } j \\ 0 & \text{se o usuário } i \text{ é indiferente ao gênero } j \\ 1 & \text{se o usuário } i \text{ gosta do gênero } j \end{cases} \tag{4.25}$$

O processo de obtenção destes níveis de preferência em uma aplicação real foi abstraído. Eles poderiam ser facilmente obtidos durante o primeiro acesso do usuário, tal como o exemplo da plataforma Deezer na seção 2.3, ou então poderiam ser inferidos sem muito esforço através do conjunto de notas já disponível para um usuário frequente do sistema.

O pseudocódigo descrito pelo algoritmo 3 ilustra o procedimento que foi seguido neste trabalho para determinar os perfis dos usuários para os quais era necessário

prever uma nota. A ideia deste procedimento é utilizar a nota que o usuário deu para um item para determinar se ele gosta ou não dos gêneros deste item. Se a nota for menor que um certo limite inferior (no caso, 20), adicionamos uma contribuição negativa para esses gêneros, e se a nota for maior ou igual a um limite superior (no caso, 80) adicionamos uma contribuição positiva para esses gêneros. Caso contrário, nenhuma contribuição é adicionada. Esta ponderação é feita para todos os itens do *dataset* de teste e, ao final, o perfil do usuário é a soma de todas as contribuições normalizada para o conjunto $\{-1, 0, 1\}$.

Porém, especialmente no caso em que o usuário está acessando o sistema pela primeira vez e está informando manualmente quais são os gêneros dos quais ele gosta ou não, é razoável pensar que ele não terá informado tantas preferências. Um usuário normal provavelmente não teria paciência para avaliar todos os 1.443 gêneros do *dataset*, dos quais muitos deles o usuário provavelmente nem teria conhecimento. Sendo assim, o procedimento de captura de perfil de usuário só levou em consideração uma parcela dos gêneros. Esta parcela foi o conjunto de gêneros associados a mais itens. No caso, foram os gêneros que pertenciam a pelo menos 200 itens, pois este valor representava exatamente 10% do conjunto de gêneros (os 10% principais). A figura 4.1 contém a distribuição do número de itens por gênero, e a figura 4.2 mostra que, seguindo este procedimento, a maior parte dos usuários informou se gostava ou não de um número bem pequeno de artistas (no eixo x está a quantidade de itens que o usuário disse se gostou ou não, e no eixo y está a quantidade de usuários que informou aquela quantidade de itens).

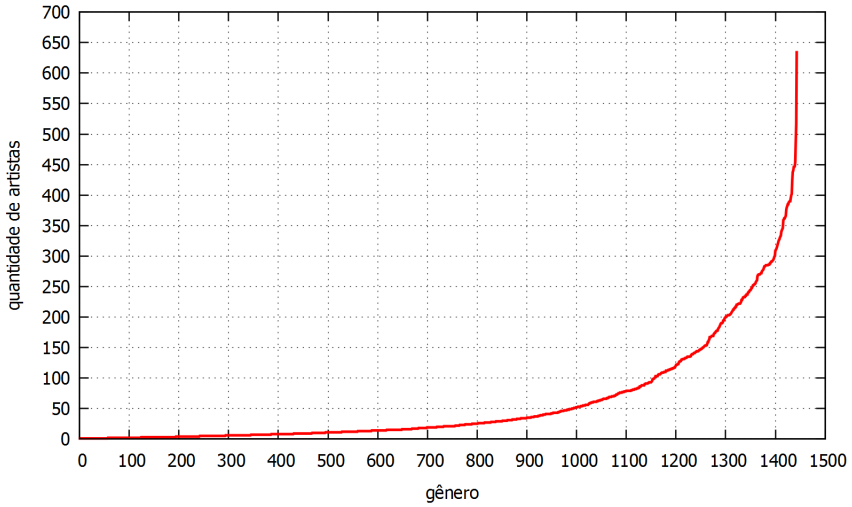

Figura 4.1: Número de artistas por gênero.

Entrada: *Dataset* de teste
Saida **:** Matriz U com os vetores de perfis de usuários

```
Minimoltens ← 200;
U ← inicializada com tudo 0;
for par de usuário/item (u, i) no dataset de teste do
    G_i ← BuscarGeneros(i);
    PontuacaoGeneros_i ← inicializado com tudo 0;
    for gênero g em G_i do
        if g pertence a pelo menos Minimoltens itens then
            NotaVerdadeira_{u,i} ← nota verdadeira do usuário u para o item i
            if NotaVerdadeira_{u,i} ⩾ 80 then
                PontuacaoGeneros_i (g) ← PontuacaoGeneros_i (g) + 1
            end
            if NotaVerdadeira_{u,i} < 20 then
                PontuacaoGeneros_i (g) ← PontuacaoGeneros_i (g) - 1
            end
        end
    end
    for gênero g em PontuacaoGeneros_i do
        U_{u,g} ← U_{u,g} + PontuacaoGeneros_i(g)
    end
end
for elemento U_{u,g} em U do
    if U_{u,g} ⩾ 1 then
        U_{u,g} ← 1
    end
    if U_{u,g} ⩽ -1 then
        U_{u,g} ← -1
    end
    else
        U_{u,g} ← 0
    end
end
```

Algoritmo 3: Captura de perfis de usuários

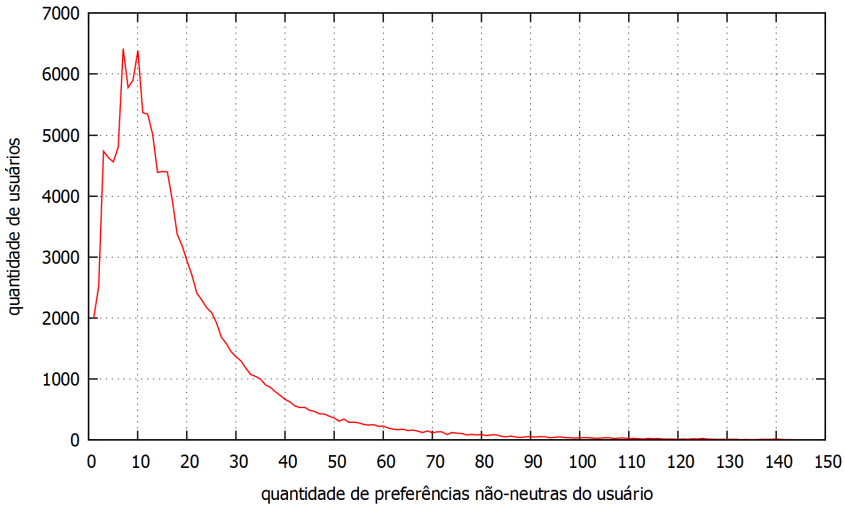

Figura 4.2: Número de preferências não-neutras de gêneros dos usuários.

4.6.3 Recomendação não-personalizada

Seguindo a mesma ideia do procedimento adotado para a filtragem colaborativa, outro método de recomendação não-personalizada foi utilizado, mas que desta vez está mais relacionado com sistemas de recomendação baseados em conteúdo. Este método foi utilizado também como *benchmark*.

Neste método, a nota prevista $\tilde{r}_{u,i}$ do usuário u para o item i é dada pela média das notas médias dos K itens mais similares ao item i. A similaridade desta vez não é calculada a partir de padrões de semelhança entre as notas dos itens, mas sim a partir da semelhança entre as características dos itens (ou seja, a correlação entre os gêneros musicais dos artistas). Ou seja, a nota prevista $\tilde{r}_{u,i}$ é dada por

$$\tilde{r}_{u,i} = \frac{1}{|I'|} \sum_{j \in I'} \bar{r_j} \tag{4.26}$$

onde I' é o conjunto dos K itens mais similares a i e $\bar{r_j}$ é a nota média do item j.

Para o cálculo da similaridade foi adotada a similaridade Dice. Dado o conjunto de gêneros G_i do item i e o conjunto de gêneros G_j do item j, a similaridade Dice entre os itens i e j é dada por

$$sim_{Dice}(i,j) = \frac{2 \times |G_i \cap G_j|}{|G_i| + |G_j|} \tag{4.27}$$

Portanto, artistas com muitos gêneros em comum tendem a ser mais semelhantes através desta métrica de similaridade.

4.6.4 Recomendação personalizada

Outro método de recomendação baseado em conteúdo foi adotado, e que leva em conta a relação entre o perfil do usuário e as características do item.

Até agora, todas as medidas de similaridade vistas até aqui calculam a similaridade entre dois usuários ou entre dois itens. O método proposto calcula a similaridade entre um usuário e um item. Isto é possível pois ambos são representados por um vetor de características de mesma dimensão. Na verdade, o nome "similaridade"é adotado para seguir o padrão em sistemas de recomendação, mas semanticamente está mais para uma medida de correlação. Esta medida de similaridade foi pensada de forma a satisfazer as interações descritas na tabela 4.2.

Tabela 4.2: Interações entre usuário, item e gênero na medida de similaridade do sistema de recomendação baseado em conteúdo.

Relação Usuário x Gênero	**Relação Item x Gênero**	**Resultado**
Gosta	É	Bom
Gosta	Não é	Neutro
Indiferente	É	Neutro
Indiferente	Não é	Neutro
Não Gosta	É	Ruim
Não Gosta	Não é	Neutro

Desta forma, a similaridade $sim(u, i)$ entre o usuário u e o item i representados respectivamente pelos vetores U e I é dada por

$$sim(u,i) = \frac{\sum_{j=1}^{N_g} U_j \times I_j}{\sum_{j=1}^{N_g} abs(U_j) \times abs(I_j)} \tag{4.28}$$

onde N_g é o número de gêneros musicais do *dataset*, e o operador *abs* é o módulo do número, que foi representado desta maneira para não ser confundido com a cardinalidade de um conjunto.

Note que essa medida de similaridade também é limitada pelo intervalo [-1,1]. Repare também que, como $I_j \in \{0, 1\}$ e $U_j \in \{-1, 0, 1\}$, só existem duas formas de adicionar uma contribuição não nula no numerador:

1. $I_j = 1$ e $U_j = 1$, que corresponde ao caso em que o usuário gosta do gênero do artista

2. $I_j = 1$ e $U_j = -1$, que corresponde ao caso em que o usuário não gosta do gênero do artista

Qualquer outra combinação de valores implica em nenhuma contribuição no numerador. Sendo assim, esta fórmula está alinhada com o comportamento esperado, representado na tabela 4.2. Ao final, o numerador representa um "saldo de contribuições"em relação ao artista. Ou seja, se um artista possui por exemplo 2 gêneros musicais diferentes, sendo um deles apreciado pelo usuário e o outro não, o efeito final é que um gênero acaba compensando o outro de forma que o saldo final (o numerador) seja nulo.

A ideia por trás do denominador é indicar o número de contribuições, sejam elas positivas ou negativas.

Repare que, quando o usuário gosta de todos os gêneros do artista, o maior valor de similaridade é obtido (+1). Já quando o usuário não gosta de todos os gêneros do artista, o menor valor de similaridade é obtido (-1). Gêneros para os quais o usuário é indiferente ou gêneros aos quais o artista não está associado simplesmente não adicionam informação nenhuma no ponto de vista desta medida de similaridade.

Desta forma, se, por exemplo, um artista possui 5 gêneros musicais e um usuário gosta de 3 deles e não gosta dos outros 2, a similaridade entre o usuário e o item seria de $\frac{1}{5}$. Já se o usuário gostasse de 3 gêneros e fosse indiferente aos outros 2, a similaridade seria de $\frac{3}{3} = 1$.

Uma consequência desta medida é que artistas que possuem poucos gêneros musicais tendem a produzir similaridades mais discretizadas, no limite produzindo similaridades "extremas"(ou mínimo, ou máximo ou nula) quando possuem somente um gênero musical. Da mesma forma, isto também acontece com os usuários que gostam ou desgostam de poucos gêneros musicais.

A partir deste valor de similaridade, foram definidos dois métodos diferentes de previsão de nota.

Método 1

A nota do usuário u prevista para o item i, $\tilde{r}_{u,i}$, é calculada da seguinte maneira:

$$\tilde{r}_{u,i} = \bar{r}_i + 100 \times \alpha \times sim(u, i) \tag{4.29}$$

onde $\bar{r}_i$ é a nota média do item i e α é o peso da similaridade, que é ajustado para determinar a influência da similaridade na nota final. O número 100 não é escolhido aleatoriamente: ele é a diferença entre a nota máxima (100) e a nota mínima (0).

Feito isto, a nota $\tilde{r}_{u,i}$ ainda recebe o seguinte tratamento:

$$\tilde{r}_{u,i} = \begin{cases} 0 & \text{se } \tilde{r}_{u,i} < 0 \\ 100 & \text{se } \tilde{r}_{u,i} > 100 \\ \tilde{r}_{u,i} & \text{caso contrário} \end{cases} \tag{4.30}$$

Sendo assim, a nota média do item serve como base para a previsão e tem papel semelhante ao *bias* descrito na seção 4.5.3. A similaridade, que mede o alinhamento entre os gêneros musicais do artista e os gostos musicais do usuário, indica o tamanho da fração de nota que deve ser adicionada à nota média do item de forma a obter a previsão de nota. O parâmetro α ainda pode ser livremente ajustado para obter previsões mais precisas.

Método 2

A nota do usuário u prevista para o item i, $\tilde{r}_{u,i}$, é calculada da seguinte maneira:

$$\tilde{r}_{u,i} = 50 \times (1 + sim(u, i)) \tag{4.31}$$

Neste outro método a nota média do item não é utilizada, e a similaridade indica o quanto a nota prevista irá se distanciar do centro do intervalo de notas. Uma similaridade de -1, que indica total descasamento entre interesses do usuário e características do item, resultaria em uma nota prevista igual a 0 (a menor nota possível), e uma similaridade de +1, que indica o alinhamento perfeito entre gostos do usuário e características do item, resultaria em uma nota prevista igual a 100 (a maior nota possível). Já uma similaridade de 0, que pode ser vista como o caso em que o usuário é totalmente indiferente aos gêneros musicais do artista, resultaria em uma nota prevista igual a 50.

4.7 Filtragem híbrida

Este trabalho também propõe uma técnica de filtragem híbrida. O objetivo deste sistema híbrido que foi implementado é especificamente combinar os resultados de um sistema de recomendação de filtragem colaborativa com os resultados de um sistema de recomendação baseado em conteúdo para produzir previsões de nota de forma mais assertiva sem sofrer com o problema de *cold-start* de usuário.

A nota do usuário u para o item i, $\tilde{r}_{u,i}$, prevista por este sistema híbrido é uma combinação linear entre as notas previstas individualmente pelos dois sistemas base (um de filtragem colaborativa e outro baseado em conteúdo). Cada uma dessas duas parcelas é ponderada de acordo com um parâmetro denominado confiança que varia

entre 0 e 1. A confiança do sistema de filtragem colaborativa (β_{cf}) deve aumentar quando o número de itens avaliados pelo usuário u aumenta, e a confiança do sistema baseado em conteúdo (β_{cb}) pode ser definida como $1-\beta_{cf}$. Note que isto não significa que a confiança do sistema baseado em conteúdo diminui quando um usuário avalia mais itens, mas sim que a confiança do sistema de filtragem colaborativa passa a ser relativamente maior até o ponto em que passa a ser tão dominante que a primeira pode ser desconsiderada.

Para alcançar este objetivo, foram definidos dois parâmetros: n_{inf} e n_{sup}. Seja n o número de itens que o usuário u avaliou e δ o intervalo $n_{sup} - n_{inf}$. As seguintes situações são desejadas:

- Se $n \leqslant n_{inf}$, então a nota prevista pelo sistema híbrido será somente a nota prevista pelo sistema baseado em conteúdo.
- Se $n \geqslant n_{sup}$, então a nota prevista pelo sistema híbrido será somente a nota prevista pelo sistema de filtragem colaborativa.
- Se $n_{inf} < n < n_{sup}$, então a nota prevista pelo sistema híbrido é uma combinação linear das previsões dos dois sistemas base, onde a confiança β_{cf} é diretamente proporcional a n

Sendo assim, os parâmetros de confiança β_{cf} e β_{cb} foram definidos através das seguintes equações:

$$\beta_{cf} = \frac{1}{\delta} \min(\delta, \max(0, n - n_{inf})) \tag{4.32}$$

$$\beta_{cb} = \frac{1}{\delta} \min(\delta, \max(0, n_{sup} - n)) \tag{4.33}$$

No caso especial em que $\delta = 0$ (ou seja, $n_{min} = n_{sup}$), os fatores de confiança são calculados pelas seguintes equações:

$$\beta_{cf} = \begin{cases} 1 & \text{se } n \geqslant n_{inf} \\ 0 & \text{caso contrário} \end{cases} \tag{4.34}$$

$$\beta_{cb} = 1 - \beta_{cf} \tag{4.35}$$

A nota do usuário u para o item i prevista pelo sistema de recomendação híbrido é dada por

$$\tilde{r}_{u,i} = \beta_{cf}.r^{cf}_{u,i} + \beta_{cb}.r^{cb}_{u,i} \tag{4.36}$$

onde $r^{cf}_{u,i}$ é a nota prevista pelo sistema de recomendação de filtragem colaborativa e $r^{cb}_{u,i}$ é a nota prevista pelo sistema de recomendação baseado em conteúdo.

4.8 Conclusão

Este capítulo iniciou descrevendo o processo de avaliação utilizado neste trabalho, apresentando uma breve discussão sobre a escolha da métrica de avaliação. Após isso, detalhou os métodos de recomendação (tecnicamente, os métodos de previsão de nota) utilizados durante o trabalho.

Foram apresentados métodos clássicos de recomendação personalizada e não-personalizada (especialmente para os sistemas de recomendação de filtragem colaborativa, que são muito bem documentados e padronizados na literatura) e também algoritmos propostos por este trabalho, como os métodos de recomendação baseada em conteúdo e os métodos híbridos.

Os algoritmos descritos aqui serão mencionados no próximo capítulo, onde serão de fato avaliados.

Capítulo 5

Resultados da Avaliação

Este capítulo irá apresentar os resultados de avaliação dos algoritmos descritos no capítulo 4, medidos através da raiz do erro médio quadrático. Para alguns sistemas de recomendação, a influência dos parâmetros também será abordada.

5.1 Filtragem colaborativa baseada em itens

O primeiro sistema de recomendação avaliado foi a filtragem colaborativa baseada em memória baseada em itens.

Inicialmente foi adotado o padrão de considerar na vizinhança dos itens todos os demais itens do *dataset*.

O *dataset* de treino era composto por 90% do *dataset* original e o *dataset* de teste era composto por 10% do *dataset* original.

A tabela 5.1 sumariza os resultados obtidos e a figura 5.1 ilustra a tabela. Os três primeiros algoritmos servem de *benchmark* por serem mais simples - é esperado que eles não performem tão bem quanto os outros. A diferença entre os outros três é a medida de similaridade utilizada.

Tabela 5.1: Resultados para filtragem colaborativa baseada em itens.

Algoritmo	**RMSE**
Nota média do item	36,5799561
Nota média do item ajustada pela média do usuário	30,4975990
Nota média do usuário	28,2208995
Baseado em itens - Coeficiente Pearson	27,0378967
Baseado em itens - Similaridade Cosseno	27,6971683
Baseado em itens - Log-Likelihood	28,1437640

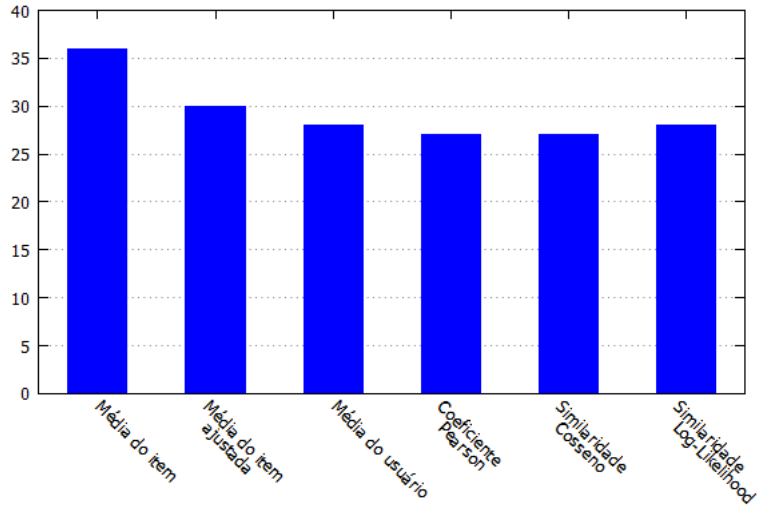

Figura 5.1: Resultados para filtra colaborativa baseada em itens.

Como era de se esperar, os dois primeiros métodos (nota média do item e nota média do item ajustada pela média do usuário) não performaram bem, porém o método de nota média do usuário performou surpreendentemente bem comparado aos outros. Isto pode indicar usuários que o *dataset* em questão contém usuários com notas altamente enviesadas. Porém, devemos lembrar que, mesmo que o resultado do RMSE tenha sido bom, o método de prever notas através da média do usuário dificilmente seria útil em uma aplicação real, já que não seria possível determinar os itens que de fato devem ser recomendados.

A similaridade *log-likelihood* não performou muito bem, o que reforça a afirmação feita na seção 4.4.3 de que esta medida de similaridade talvez seja mais indicada para preferências implícitas ou booleanas.

As similaridades Pearson e Cosseno produziram resultados bem parecidos, mas deve-se lembrar que neste caso a similaridade Pearson é exatamente igual à similaridade Cosseno calculada para um conjunto de dados centralizado. Portanto, conforme foi discutido na seção 4.4.3, o resultado ligeiramente melhor obtido pela similaridade Pearson pode ser explicado devido à ausência de valores negativos de similaridade para a similaridade Cosseno, limitando assim o intervalo de similaridades obtido.

Também foi avaliado o impacto de uma escolha mais seletiva de vizinhança de itens. Para isto, no cálculo da nota prevista foram considerados apenas os itens com valor de similaridade superior a um limiar τ. Foi adotada a similaridade Pearson, já que ela produziu os melhores resultados e variou-se o valor de τ. Os resultados podem ser vistos na tabela 5.2.

Tabela 5.2: Resultados para filtragem colaborativa baseada em itens usando similaridade Pearson filtrando a vizinhança.

τ	**RMSE**
-0,2	26,9684143
-0,1	26,9985132
0	27,0459785
0,1	27,0424766
0,2	27,0352896

Como podemos perceber, a variação no valor de τ não trouxe mudanças significativas.

5.2 Filtragem colaborativa baseada em modelos

A próxima categoria de sistemas de recomendação a ser avaliada foram os sistemas de recomendação que utilizam a fatoração SVD.

Foram avaliados o método do gradiente estocástico (que, conforme apresentado na seção 4.5.4, é implementado por duas classes distintas na biblioteca Mahout: *RatingSGDFactorizer* e *ParallelSGDFactorizer*) e o método dos mínimos quadrados alternados (implementado pela classe *ALSWRFactorizer*).

Ambos os métodos dependem de uma série de parâmetros: o parâmetro de regularização λ, o número de fatores latentes n_f e o limite de iterações n_{it}. No caso do método do gradiente estocástico ainda existe a taxa de aprendizado γ. Porém, como sugerido por FUNK[39], a taxa de aprendizado foi configurada para o valor 0,001 e não foi mais alterada.

5.2.1 Gradiente Estocástico

As tabelas 5.3 a 5.11 contêm os resultados do RMSE para o método do gradiente estocástico variando os diferentes parâmetros. Cada uma das tabelas representa um valor para n_f. Cada célula da tabela contém o RMSE para os parâmetros n_{it} e λ correspondentes. As tabelas contêm o resultado do RMSE tanto para a classe *RatingSGDFactorizer* quanto para a classe *ParallelSGDFactorizer*.

Observando os resultados dessas tabelas, é possível tirar algumas conclusões:

1. Para praticamente qualquer conjunto de parâmetros, o método do gradiente estocástico foi superior à filtragem colaborativa baseada em memória

Tabela 5.3: RMSE gradiente estocástico - $n_f = 10$

		RatingSGDFactorizer				***ParallelSGDFactorizer***			
		λ				λ			
		0,01	**0,05**	**0,1**	**0,5**	**0,01**	**0,05**	**0,1**	**0,5**
n_{it}	**5**	23,7371	23,7280	23,6417	23,5271	23,5725	23,5873	23,5801	23,5485
	6	23,8065	23,7272	23,6445	23,5297	23,6409	23,5389	23,5649	23,4501
	7	23,6419	23,6869	23,6458	23,5260	23,5753	23,5579	23,5212	23,4544
	8	23,7179	23,7051	23,6466	23,5682	23,6523	23,4776	23,5431	23,4491
	9	23,6995	23,6782	23,7652	23,5841	23,6016	23,5256	23,5543	23,4831
	10	23,8400	23,7163	23,7304	23,5717	23,5977	23,6391	23,5833	23,4806
	11	23,8671	23,7656	23,7562	23,5857	23,6266	23,5484	23,6675	23,4810
	12	23,8247	23,7466	23,7034	23,6022	23,6808	23,5648	23,4795	23,4192
	13	23,8625	23,7673	23,8305	23,5177	23,8109	23,6074	23,6372	23,5042
	14	23,8046	23,8424	23,7442	23,5681	23,5523	23,6090	23,5850	23,5391
	15	23,8696	23,8783	23,7556	23,7967	23,6847	23,6177	23,6164	23,4445

Tabela 5.4: RMSE gradiente estocástico - $n_f = 50$

		RatingSGDFactorizer				***ParallelSGDFactorizer***			
		λ				λ			
		0,01	**0,05**	**0,1**	**0,5**	**0,01**	**0,05**	**0,1**	**0,5**
n_{it}	**5**	24,1612	23,6339	23,2284	23,0243	23,6120	23,3301	22,9906	22,9267
	6	24,3870	24,0391	23,5801	22,8324	23,8348	23,4780	23,3165	22,7287
	7	24,6202	24,2711	23,8172	22,7777	24,0869	23,6568	23,3899	22,7460
	8	24,9648	24,3739	24,0535	22,8952	24,2186	23,9556	23,6913	22,6166
	9	25,1318	24,6937	24,1819	22,8455	24,4845	24,1426	23,7430	22,6589
	10	25,2712	24,7639	24,4391	22,9423	24,4661	24,2241	23,9469	22,7934
	11	25,4814	25,0073	24,5397	23,0916	24,7765	24,3389	24,0924	22,7795
	12	25,5328	25,2013	24,7309	23,2796	24,8690	24,5277	24,1558	22,8967
	13	25,7616	25,1770	24,9125	23,3277	24,9124	24,6220	24,2689	22,9811
	14	25,8780	25,4201	25,1061	23,3737	24,8880	24,6841	24,2788	23,0428
	15	25,9996	25,4923	25,0833	23,4407	25,0564	24,7740	24,4376	22,9880

Tabela 5.5: RMSE gradiente estocástico - $n_f = 100$

		RatingSGDFactorizer				***ParallelSGDFactorizer***			
		λ				λ			
		0,01	**0,05**	**0,1**	**0,5**	**0,01**	**0,05**	**0,1**	**0,5**
n_{it}	**5**	24,4529	23,8503	23,3385	22,9661	24,0984	23,6985	23,1765	22,8344
	6	25,1048	24,4593	23,8104	22,8186	24,5508	24,1322	23,5106	22,6265
	7	25,6106	24,8714	24,2699	22,8045	24,8140	24,5428	23,8257	22,5041
	8	26,0580	25,2183	24,6812	22,7733	25,2127	24,6630	24,2274	22,4771
	9	26,4431	25,7047	25,1083	22,7961	25,4593	24,9218	24,4256	22,5438
	10	26,6935	25,9300	25,3093	22,9428	25,7467	25,1964	24,5980	22,6548
	11	26,9749	26,1335	25,4564	23,1548	25,9735	25,3957	24,7955	22,6454
	12	27,2700	26,3238	25,6721	23,2416	26,1529	25,5691	24,9911	22,8989
	13	27,4706	26,5980	25,8853	23,3334	26,3935	25,7119	25,1981	23,0491
	14	27,7941	26,8163	25,9290	23,5881	26,5136	25,7904	25,3155	23,0609
	15	27,9232	26,8179	26,2277	23,6052	26,5810	25,9959	25,3795	23,2000

Tabela 5.6: RMSE gradiente estocástico - $n_f = 150$

		RatingSGDFactorizer				***ParallelSGDFactorizer***			
		λ				λ			
		0,01	**0,05**	**0,1**	**0,5**	**0,01**	**0,05**	**0,1**	**0,5**
n_{it}	**5**	24,4829	23,9199	23,3971	22,8877	24,1612	23,7587	23,1714	22,6782
	6	25,1535	24,4938	23,7158	22,7929	24,6748	24,1463	23,6274	22,6585
	7	25,7809	25,0365	24,3698	22,7221	25,0868	24,5617	23,9980	22,4775
	8	26,2722	25,5028	24,6716	22,7935	25,4656	24,8552	24,2470	22,4064
	9	26,6874	25,7470	25,0592	22,8287	25,7631	25,1117	24,6193	22,5188
	10	27,0251	26,0532	25,3047	22,9360	26,0512	25,2711	24,7417	22,6481
	11	27,2525	26,2985	25,5082	23,0183	26,2248	25,6812	24,9240	22,7452
	12	27,5935	26,6587	25,8184	23,0580	26,4110	25,8814	25,2152	22,8925
	13	27,7499	26,6419	26,0308	23,4262	26,7513	25,9160	25,2687	22,8693
	14	28,0902	26,9194	26,0211	23,5082	26,7159	26,0789	25,4241	23,0720
	15	28,1629	27,1215	26,3721	23,4861	27,0077	26,1096	25,5830	23,1612

Tabela 5.7: RMSE gradiente estocástico - $n_f = 200$

		RatingSGDFactorizer				***ParallelSGDFactorizer***			
		λ				λ			
		0,01	**0,05**	**0,1**	**0,5**	**0,01**	**0,05**	**0,1**	**0,5**
n_{it}	**5**	24,2927	23,7339	23,3394	22,9609	24,0497	23,6381	23,2152	22,7687
	6	25,1093	24,4303	23,7183	22,7180	24,4504	24,0897	23,5083	22,5572
	7	25,5736	24,8379	24,2102	22,7343	24,9402	24,3353	23,8747	22,4825
	8	26,1137	25,1832	24,4379	22,7747	25,3809	24,6250	24,1831	22,4878
	9	26,3177	25,5429	24,8494	22,7882	25,6218	25,0043	24,4547	22,5571
	10	26,7607	25,7279	25,0386	22,7618	26,0219	25,1499	24,6103	22,6621
	11	27,1505	26,0221	25,4120	23,0779	26,1154	25,4066	24,6648	22,7950
	12	27,4228	26,2685	25,4795	23,1557	26,4571	25,5330	24,8356	22,6651
	13	27,6535	26,4812	25,6673	23,3140	26,5779	25,6940	24,9814	22,8434
	14	27,7786	26,5650	25,8789	23,3911	26,6335	25,7838	25,0788	23,0088
	15	27,9512	26,7789	25,9201	23,4920	26,7055	25,8112	25,1804	22,9748

Tabela 5.8: RMSE gradiente estocástico - $n_f = 250$

		RatingSGDFactorizer				***ParallelSGDFactorizer***			
		λ				λ			
		0,01	**0,05**	**0,1**	**0,5**	**0,01**	**0,05**	**0,1**	**0,5**
n_{it}	**5**	24,1356	23,6282	23,1973	22,8705	23,9250	23,5751	23,0513	22,6009
	6	24,8235	24,1958	23,6255	22,7703	24,3607	23,8438	23,3976	22,6240
	7	25,3818	24,6128	24,0346	22,6917	24,7844	24,3318	23,6481	22,5276
	8	25,7741	24,9198	24,3705	22,6935	25,0773	24,5013	23,9603	22,4503
	9	26,1236	25,1913	24,4601	22,8517	25,4602	24,7129	24,2280	22,6162
	10	26,3794	25,5117	24,8615	22,9247	25,6188	24,8376	24,2973	22,6271
	11	26,7400	25,5066	25,1067	22,9241	25,7430	24,9665	24,5114	22,6686
	12	26,9648	25,7987	25,1277	23,1574	25,8805	25,1528	24,4877	22,7919
	13	26,9976	25,9609	25,0837	23,2383	26,0731	25,3816	24,7849	22,8180
	14	27,1581	26,0582	25,3693	23,2535	26,2922	25,4369	24,7295	22,8759
	15	27,3303	26,2550	25,3439	23,4129	26,5199	25,4182	24,8643	22,9456

Tabela 5.9: RMSE gradiente estocástico - $n_f = 300$

		RatingSGDFactorizer				***ParallelSGDFactorizer***			
		λ				λ			
		0,01	**0,05**	**0,1**	**0,5**	**0,01**	**0,05**	**0,1**	**0,5**
n_{it}	**5**	24,0294	23,5538	23,1276	22,8832	23,7349	23,4451	23,0426	22,6108
	6	24,5820	24,0063	23,5526	22,7105	24,1308	23,7931	23,2626	22,5758
	7	25,0431	24,3810	23,9063	22,6325	24,4073	24,0107	23,4991	22,5184
	8	25,4977	24,6661	24,1237	22,6456	24,7109	24,2620	23,7643	22,4783
	9	25,7417	24,9041	24,3398	22,7430	25,0349	24,4181	23,9451	22,5289
	10	25,9900	25,1711	24,3678	22,8941	25,2026	24,5949	23,9983	22,6134
	11	26,1192	25,2965	24,7163	22,9225	25,4307	24,6587	24,2273	22,6465
	12	26,3202	25,3554	24,6364	22,9204	25,6879	24,7412	24,2693	22,6438
	13	26,6368	25,5074	24,8577	23,0118	25,7943	24,8279	24,2656	22,8021
	14	26,6056	25,4856	25,0159	23,2367	25,7313	25,0093	24,3895	22,8214
	15	26,8065	25,5966	24,9211	23,2158	25,9840	25,0699	24,4843	22,7924

Tabela 5.10: RMSE gradiente estocástico - $n_f = 350$

		RatingSGDFactorizer				***ParallelSGDFactorizer***			
		λ				λ			
		0,01	**0,05**	**0,1**	**0,5**	**0,01**	**0,05**	**0,1**	**0,5**
n_{it}	**5**	23,8352	23,4593	23,0995	22,9309	23,5219	23,2661	22,9469	22,7289
	6	24,3312	23,7590	23,3531	22,7797	24,0031	23,5220	23,2466	22,4763
	7	24,8124	24,2941	23,7742	22,6777	24,2171	23,7355	23,4116	22,4434
	8	25,0421	24,3768	23,8670	22,7759	24,4666	23,9648	23,5432	22,4190
	9	25,4420	24,6856	24,0903	22,7128	24,7120	24,1772	23,6663	22,4899
	10	25,7920	24,6231	24,2544	22,8223	24,8795	24,1696	23,7923	22,5819
	11	25,8181	24,8233	24,2502	22,8411	25,0333	24,3478	23,9215	22,5502
	12	25,8808	24,9876	24,3785	23,0958	25,2194	24,5127	23,9192	22,6062
	13	26,0001	25,0719	24,4738	23,0622	25,2482	24,5326	23,9445	22,7317
	14	26,2224	25,0638	24,5203	23,0552	25,4397	24,6105	24,1516	22,7539
	15	26,2012	25,2038	24,5664	23,1346	25,4301	24,6372	24,0236	22,8566

Tabela 5.11: RMSE gradiente estocástico - $n_f = 400$

		RatingSGDFactorizer				***ParallelSGDFactorizer***			
		λ				λ			
		0,01	**0,05**	**0,1**	**0,5**	**0,01**	**0,05**	**0,1**	**0,5**
n_{it}	**5**	23,7697	23,3865	23,1706	22,8454	23,5002	23,2080	22,9141	22,5559
	6	24,1846	23,6334	23,2258	22,6337	23,8603	23,3916	23,0620	22,5425
	7	24,5351	23,8779	23,6243	22,6414	24,1814	23,5756	23,2895	22,4336
	8	24,9185	24,2621	23,7898	22,6203	24,2889	23,7895	23,3158	22,4441
	9	25,1482	24,4583	24,0077	22,6937	24,4344	23,8074	23,4996	22,4765
	10	25,2271	24,4225	24,0144	22,7504	24,5996	24,0437	23,6061	22,5221
	11	25,3996	24,4439	24,0730	22,9181	24,6925	24,1067	23,7043	22,5967
	12	25,4642	24,5728	24,1607	22,9613	24,8307	24,1472	23,7215	22,6363
	13	25,5934	24,7401	24,0668	23,0647	25,1000	24,1801	23,6761	22,6373
	14	25,5817	24,5689	24,1546	23,0448	25,0561	24,3011	23,8441	22,7140
	15	25,7606	24,6719	24,2067	23,2138	25,0347	24,3444	23,7284	22,7170

2. Independente do número de iterações ou do número de fatores latentes, o melhor resultado para ambas as classes era sempre obtido quando o parâmetro de regularização $\lambda = 0,5$. Isto nos leva a crer que ainda pode ser possível ajustar este parâmetro para conseguir resultados ainda melhores

3. Considerando apenas os casos em que $\lambda = 0,5$ (por ter resultado nos menores valores de RMSE), o parâmetro n_{it} também apresentou um padrão interessante: o valor do RMSE diminuía até que n_{it} atingia por volta de 7 ou 8, quando então começava a subir novamente, com os valores n_{it} próximos de 15 resultando constantemente em um RMSE maior. De fato, este efeito já havia sido observado por FUNK[39]. O número ideal de iterações depende dos demais parâmetros. Este comportamento pôde ser melhor observado para a classe não-paralela.

4. Não ficou muito claro como o número de fatores n_f influenciou no resultado. Não é possível afirmar se o valor ótimo não foi atingido por ser maior que os valores utilizados ou se os valores utilizados não foram granulares o suficiente. Mas, aparentemente, os bons resultados obtidos não tiveram nenhuma relação com os valores de n_f.

5.2.2 Mínimos Quadrados Alternados

As tabelas 5.12, 5.13 e 5.14 contêm os resultados do RMSE para o método dos mínimos quadrados alternados variando os diferentes parâmetros. Elas estão organizadas da mesma forma que as tabelas referentes ao gradiente estocástico, porém com a diferença de que só uma classe implementa o algoritmo.

Tabela 5.12: RMSE mínimos quadrados alternados - $n_f = 10$

		ALSWRFactorizer			
		λ			
		0,01	**0,05**	**0,1**	**0,5**
n_{it}	**5**	71,3645	33,8850	32,3518	30,7707
	6	62,0955	36,3208	32,3101	30,8593
	7	74,1811	34,5075	32,5216	30,8992
	8	79,6480	38,6263	32,1630	30,8021
	9	152,0617	36,5446	31,8172	30,8152
	10	105,3388	39,3879	32,3515	30,9883
	11	113,2721	39,0489	32,7381	31,0664
	12	121,0553	39,9597	34,5391	31,0925
	13	117,0804	39,5514	32,9814	31,2067
	14	143,7513	37,2128	32,9678	30,9055
	15	169,6235	43,0118	35,4800	30,9813

Tabela 5.13: RMSE mínimos quadrados alternados - $n_f = 50$

		ALSWRFactorizer			
		λ			
		0,01	**0,05**	**0,1**	**0,5**
n_{it}	**5**	70,5015	36,0950	33,7245	31,5491
	6	75,7356	36,5856	33,3146	31,5643
	7	67,5633	36,6146	33,1900	31,4384
	8	78,3103	36,8757	33,6967	31,5506
	9	63,2388	35,5231	33,9409	31,6279
	10	71,4817	36,9139	33,8334	31,4459
	11	71,3598	37,2327	33,8526	31,6187
	12	73,8657	37,7394	33,4731	31,5059
	13	72,1640	37,0648	33,4750	31,4397
	14	68,5501	36,6583	34,1623	31,6368
	15	86,0691	37,4797	34,3352	31,4787

Tabela 5.14: RMSE mínimos quadrados alternados - $n_f = 150$

		ALSWRFactorizer			
		λ			
		0,01	**0,05**	**0,1**	**0,5**
n_{it}	**5**	92,7699	38,6446	34,9864	32,1008
	6	79,2009	39,3414	35,1057	31,9098
	7	92,2576	38,9095	35,3219	31,8088
	8	89,3829	39,8833	35,0437	31,9226
	9	85,3676	39,5020	34,9622	31,9409
	10	85,2943	38,7298	35,1691	31,9259
	11	90,5554	40,5350	34,9183	31,9810
	12	79,2357	40,4765	34,8994	32,0039
	13	92,6549	39,3344	34,9188	32,0675
	14	89,5145	38,7314	35,0309	32,1517
	15	91,2172	40,6845	34,9399	31,9188

Um ponto que foi observado durante a execução dos testes é a enorme diferença de tempo entre os métodos do gradiente estocástico e dos mínimos quadrados alternados. A fatoração das matrizes não demorava mais do que alguns poucos minutos para o primeiro, enquanto que para o segundo demorava horas (em alguns casos, até mesmo mais de um dia). Em uma aplicação real isto poderia não ser muito problemático, já que a decomposição da matriz iria ocorrer uma única vez. Porém, como a escolha dos parâmetros influencia diretamente como a matriz vai ser decomposta e este trabalho analisou diversos conjuntos de parâmetros, a matriz teve que ser decomposta diversas vezes. Isto impossibilitou a decomposição da matriz pelo método dos mínimos quadrados alternados para o mesmo conjunto de parâmetros utilizado para o gradiente estocástico.

Dito isto, podemos tirar as seguintes conclusões:

1. Para qualquer conjunto de parâmetros, o método do gradiente estocástico foi superior ao método dos mínimos quadrados alternados. Até mesmo os algoritmos de filtragem colaborativa baseados em memória foram superiores, o que foi um pouco surpreendente.

2. Independente do número de iterações ou do número de fatores latentes, o melhor resultado também sempre obtido foi quando o parâmetro de regularização $\lambda = 0,5$. Além disso, o RMSE para valores pequenos de λ se mostrou incrivelmente ruim. Portanto, isto é mais um indicativo de que ainda pode ser possível ajustar este parâmetro para conseguir resultados melhores, e que para este algoritmo a regularização é especialmente importante.

3. O número de iterações não apresentou uma influência tão clara quanto no método do gradiente estocástico. Isto pode indicar também que seriam necessárias mais iterações para obter bons resultados.

4. De forma geral, os resultados ruins obtidos tornam a análise deste algoritmo de recomendação de certa forma inconclusiva. Porém, mesmo que não seja o escopo deste trabalho, a enorme diferença de tempo de execução entre este método e o do gradiente estocástico já é um indicativo de que este já não seria ideal de qualquer forma.

5.2.3 Aprofundando a análise dos parâmetros

Para tentar determinar de forma mais conclusiva os efeitos do parâmetros de regularização λ (que até aqui possui indicativos fortes de que ainda pode ser melhor ajustado) e do número de fatores latentes n_f (que não possui nenhum indicativo de se sequer chegou perto de um valor ótimo) individualmente, eles foram variados dentro de um intervalo maior de valores enquanto que todos os demais parâmetros foram mantidos fixos.

Para isto, foi utilizado o método do gradiente estocástico pois ele obteve os melhores resultados até agora. Foi utilizada a classe *RatingSGDFactorizer* por ela ter apresentado padrões um pouco mais bem definidos.

Regularização

Para testar o parâmetro λ, foram adotados os seguintes parâmetros:

- $n_f = 400$
- $n_{it} = 10$
- $\gamma = 0,001$

O valor de λ variou entre 0,1 e 2,2 em intervalos de tamanho 0,1. O desempenho do sistema pode ser visto na figura 5.2 para os diferentes valores de λ.

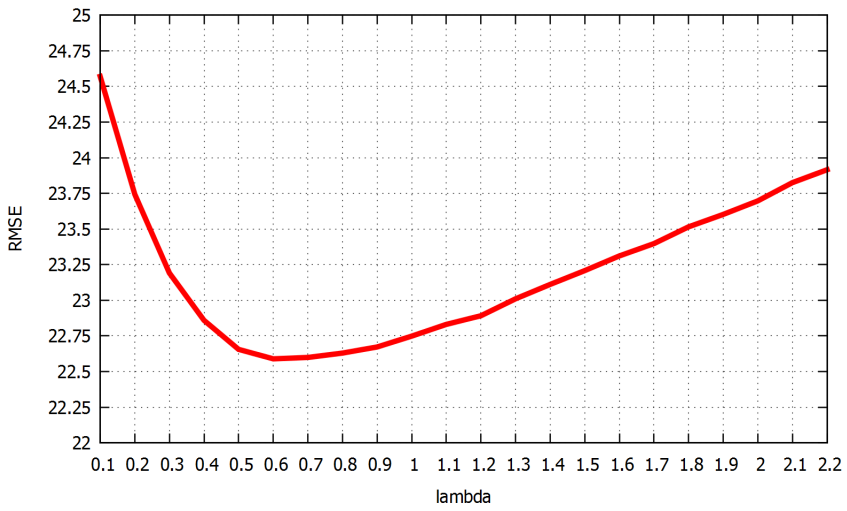

Figura 5.2: Efeito do parâmetro de regularização para o método do gradiente estocástico.

É possível perceber que o menor valor de RMSE ocorreu quando $\lambda = 0,6$. A partir deste valor, o desempenho do sistema começa a piorar.

É claro que este valor específico de λ só proporciona os melhores resultados para este conjunto de dados específico utilizado no trabalho. O que a figura 5.2 realmente ilustra é que o parâmetro de regularização tem influência direta na qualidade dos resultados do sistema de recomendação e pode ser ajustado até que atinja o seu valor ótimo.

Número de fatores latentes

Para avaliar o efeito do parâmetro n_f, foram adotados os seguintes parâmetros:

- $\lambda = 0,5$
- $n_{it} = 10$
- $\gamma = 0,001$

O valor de n_f variou entre 100 e 2600 em intervalos de tamanho 100. O desempenho do sistema pode ser visto na figura 5.3 para os diferentes valores de n_f.

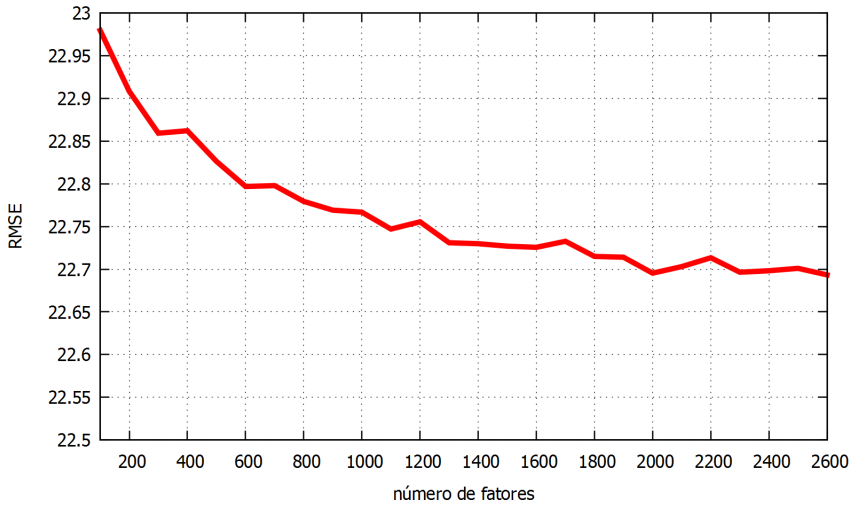

Figura 5.3: Efeito do número de fatores latentes para o método do gradiente estocástico.

Pode-se perceber que o desempenho do sistema melhorou a medida que n_f aumentou, até chegar a um ponto em que o RMSE estabilizou. O menor valor obtido foi para 2000 fatores, e a partir deste ponto o RMSE apresentou somente pequenas variações. A partir de 2600 fatores, a decomposição da matriz passou a ficar extremamente custosa em termos de tempo de execução e quantidade de memória.

Porém, também é possível perceber que, ao aumentar o número de fatores de 400 para 2000, o RMSE diminuiu em somente 0,7%. Portanto, é possível afirmar que o ganho de precisão do sistema não compensou muito o aumento na complexidade computacional, principalmente para valores de n_f superiores a 2000, quando o custo computacional passou a ficar muito elevado.

5.3 Filtragem baseada em conteúdo

A seguir, foram avaliados os sistemas de recomendação de filtragem baseada em conteúdo descritos no capítulo 4.

5.3.1 Recomendação não-personalizada

A primeira abordagem avaliada foi a de recomendação não-personalizada descrita na seção 4.6.3.

O único parâmetro do sistema é K, ou seja, a quantidade de itens similares utilizados. Para avaliar este parâmetro, o valor de K variou entre 5 e 50 em intervalos de tamanho 5. A tabela 5.15 contém o RMSE obtido para este sistema de recomendação para os diferentes valores de K, ilustrada na figura 5.4.

Tabela 5.15: Efeito do parâmetro K para a recomendação não-personalizada baseada em conteúdo.

K	**RMSE**
5	38,5075558
10	38,6137367
15	38,6387131
20	38,6988088
25	38,7642454
30	38,7920567
35	38,7958103
40	38,8371544
45	38,8648860
50	38,8860271

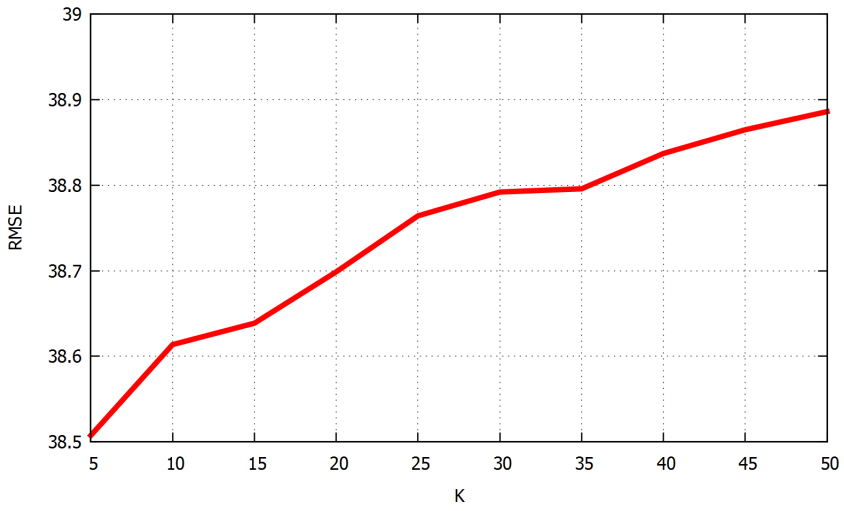

Figura 5.4: Efeito do parâmetro K para a recomendação não-personalizada baseada em conteúdo.

Como pode ser visto, o valor do RMSE aumentou à medida que K também aumentou. Isto era de certa forma esperado, pois à medida que K aumenta, são usados itens cada vez menos semelhantes (e, portanto, menos relevantes) para prever uma nota.

De qualquer forma, o desempenho deste sistema de recomendação foi pior que o dos demais vistos até aqui. Isto também não é nenhuma surpresa, já que este algoritmo foi feito com o intuito de servir como *benchmark*.

5.3.2 Recomendação personalizada - Método 1

Este é o método que prevê a nota do usuário através da equação 4.29. A tabela 5.16 contém o RMSE obtido para este sistema de recomendação utilizando diferentes valores para o parâmetro de peso da similaridade α. Esta tabela é ilustrada pela figura 5.5.

Tabela 5.16: Efeito do parâmetro α para a recomendação personalizada baseada em conteúdo - método 1.

α	**RMSE**
0,05	58,5851295
0,1	58,0891758
0,15	57,1840075
0,2	56,2185797
0,25	55,2782538
0,3	54,3684688
0,35	52,9896855
0,4	51,3715920
0,45	48,9253653
0,5	46,4861621
0,55	43,5832705
0,6	39,8158590
0,65	35,9588006
0,7	32,2797922
0,75	30,9593426
0,8	30,7605575
0,85	30,6211770
0,9	30,4958399
0,95	30,3810380
1	30,3103701

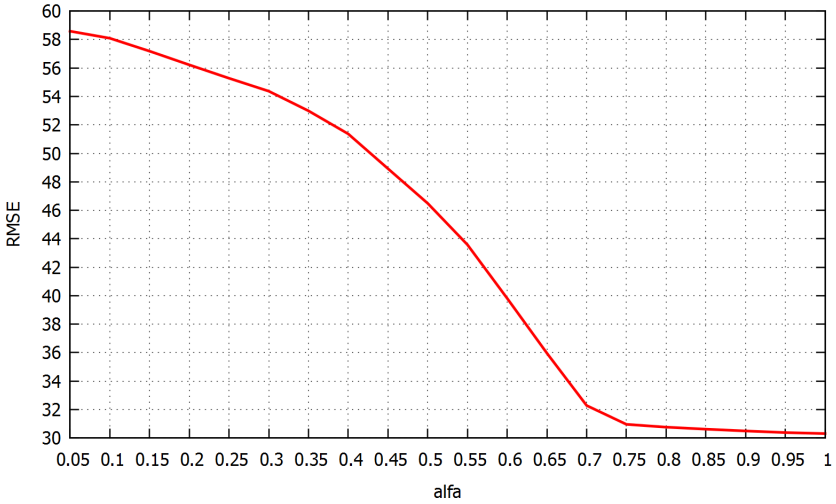

Figura 5.5: Efeito do parâmetro α para a recomendação personalizada baseada em conteúdo - método 1.

Não foram utilizados valores para α maiores do que 1 para evitar um ”efeito booleano”nas notas, ou seja, evitar que todas as notas previstas sejam sempre 0 ou 100.

Este método produziu resultados melhores que a recomendação não-personalizada baseada em conteúdo por levar em conta o perfil do usuário ao prever uma nota, mas mesmo para $\alpha = 1$ (que produziu o melhor resultado), este sistema de recomendação não foi capaz de gerar resultados tão bons quanto a filtragem colaborativa. Isto pode indicar que as características consideradas (no caso, somente os gêneros musicais dos artistas) não são suficientes para representar o sistema. Como já foi discutido anteriormente, isto é uma vantagem da filtragem colaborativa, que consegue gerar boas recomendações mesmo sem saber nenhuma informação adicional a respeito dos itens ou dos usuários.

5.3.3 Recomendação personalizada - Método 2

Este é o método que prevê a nota do usuário através da equação 4.31. Este sistema de recomendação não recebe nenhum parâmetro adicional. A tabela 5.17 contém o RMSE obtido.

Tabela 5.17: Resultado para filtragem baseada em conteúdo - recomendação personalizada método 2.

RMSE
31,5042978

Este método de recomendação produziu um resultado ligeiramente pior que o método anterior. Isto pode ser devido à ausência do termo de viés do item, que faz com que tendências específicas do item não sejam consideradas ao prever uma nota. Este efeito já foi discutido nas seções 4.5.3 e 4.6.4.

5.4 Filtragem híbrida

O algoritmo híbrido utilizado neste trabalho foi detalhado na seção 4.7. Porém, os sistemas de recomendação de filtragem colaborativa e de filtragem baseada em conteúdo utilizados como base deste método não foram especificados. Isto pois realmente o nosso sistema de recomendação híbrido é agnóstico aos sistemas base utilizados.

Para a avaliação deste trabalho, dois pares diferentes de sistemas de recomendação foram utilizados:

1. fatoração de matrizes com o método do gradiente estocástico + método 1 de recomendação personalizada baseada em conteúdo (equação 4.29)

2. fatoração de matrizes com o método do gradiente estocástico + método 2 de recomendação personalizada baseada em conteúdo (equação 4.31)

Em ambos os pares foi utilizado o sistema de recomendação de filtragem colaborativa de fatoração de matrizes utilizando o método do gradiente estocástico simplesmente pois este produziu os melhores resultados de RMSE. Variou-se apenas a escolha do sistema de recomendação baseado em conteúdo.

O método do gradiente utilizado foi o da classe *ParallelSGDFactorizer*, utilizando sempre o seguinte conjunto de parâmetros:

- $\lambda = 0,6$
- $n_f = 400$
- $n_{it} = 10$
- $\gamma = 0,001$

Em ambos os sistemas híbridos, existem os parâmetros de confiança β_{cf} e β_{cb} definidos nas equações 4.32 e 4.33. Estes parâmetros na verdade dependem de n_{inf} e n_{sup}. Sendo assim, para avaliar cada sistema híbrido,

foram utilizados diversos valores para estes dois últimos parâmetros. Foram utilizados todos os pares (n_{inf}, n_{sup}), onde $n_{inf} \leqslant n_{sup}$ e $(n_{inf}, n_{sup}) \in \{0, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70, 75, 80, 85, 90, 95, 100\}$.

O parâmetro n_{sup} merece atenção especial. Ele deve significar um número de itens avaliados pelo usuário tão alto que passa a ser razoável considerar que a filtragem colaborativa por si só é suficiente para produzir boas recomendações. Por este motivo, os testes utilizaram este parâmetro valendo no máximo 100. A figura 5.6 mostra a distribuição da quantidade de itens avaliados pelos usuários do *dataset*. No eixo x encontra-se a quantidade de itens avaliados, e no eixo y encontra-se a quantidade de usuários que avaliou aquela quantidade de itens.

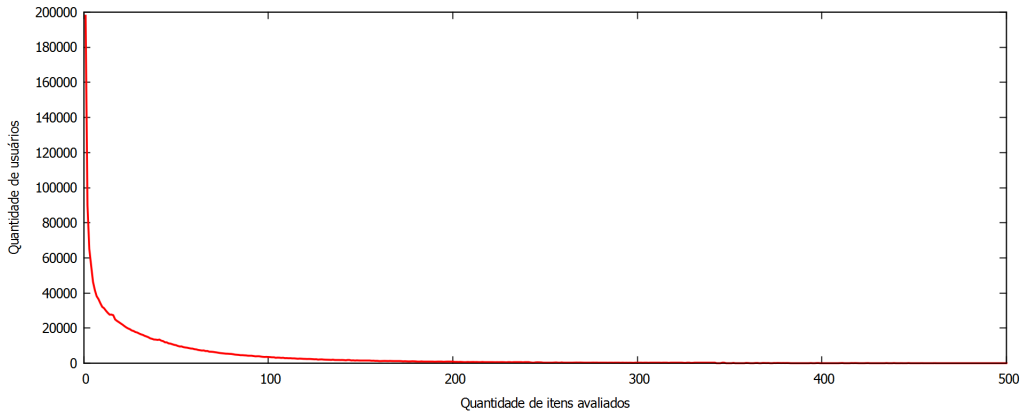

Figura 5.6: Quantidade de itens avaliados x quantidade de usuários.

Note que a figura mostra somente até 500 itens avaliados, mas na verdade existiam usuários com alguns milhares de itens avaliados. O gráfico foi limitado a somente 500 itens avaliados pois senão ficaria difícil de conseguir visualizar qualquer coisa.

É possível perceber que são poucos os usuários que deram notas a mais do que 100 itens. De fato, os usuários que deram notas a até no máximo 100 itens correspondem a um pouco mais de 85% dos usuários do *dataset*. Sendo assim, $n_{sup} = 100$ parece ser um valor no limite do que pode ser considerado razoável, e $n_{sup} > 100$ começa a ser um valor não realista.

5.4.1 Híbrido 1

Este método de filtragem híbrida utilizou o método 1 de recomendação baseada em conteúdo. O parâmetro α de peso da similaridade foi fixado em 1 por ter apresentado o melhor resultado. O RMSE obtido para todos os pares (n_{inf}, n_{sup}) pode ser visto na figura 5.7.

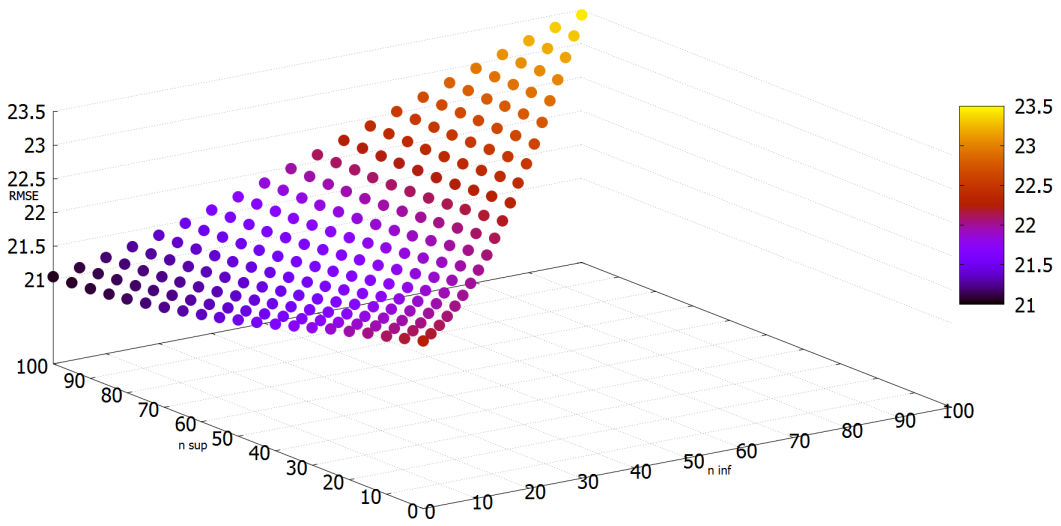

Figura 5.7: Efeito dos parâmetros n_{inf} e n_{sup} para a recomendação híbrida - método 1.

Como pode ser visto, o melhor resultado obtido para este sistema de recomendação ocorreu quando $n_{inf} = 0$ e $n_{sup} = 100$, para os quais $RMSE = 21,0453157$. Sendo assim, é possível afirmar que este sistema de recomendação híbrido superou o resultado da fatoração SVD pelo método do gradiente estocástico, tornando-se o melhor sistema de recomendação até o momento.

De forma geral, o RMSE seguiu dois padrões:

1. aumentou conforme n_{inf} aumentou
2. diminuiu conforme n_{sup} aumentou

Quanto maior n_{inf}, maior a quantidade de vezes que é utilizado somente o sistema de recomendação baseado em conteúdo, que apresentou um desempenho inferior ao da filtragem colaborativa. Desta forma, o resultado tende a piorar.

Os resultados mostraram que um valor alto para n_{sup} produz melhores recomendações, o que sugere que a contribuição do sistema de recomendação baseado em conteúdo foi positiva.

Mais do que isso: o padrão observado para o parâmetro n_{sup} sugere ainda que a contribuição do sistema baseado em conteúdo sempre será positiva, já que o desempenho do sistema melhorou constantemente à medida que o intervalo de hibridização aumentou. Sendo assim, podemos supor também que a escolha de intervalos discretos para o sistema híbrido (ou seja, um intervalo em que somente a filtragem colaborativa é usada, um em que somente a filtragem baseada em conteúdo é usada e outro em que os dois são usados) pode não ter sido a melhor alternativa, enquanto que duas funções contínuas talvez representassem mais fielmente o efeito da

confiança: uma para a filtragem colaborativa e outra para a filtragem baseada em conteúdo.

5.4.2 Híbrido 2

Este método de filtragem híbrida utilizou o método 2 de recomendação baseada em conteúdo. O RMSE obtido para todos os pares (n_{inf}, n_{sup}) pode ser visto na figura 5.8.

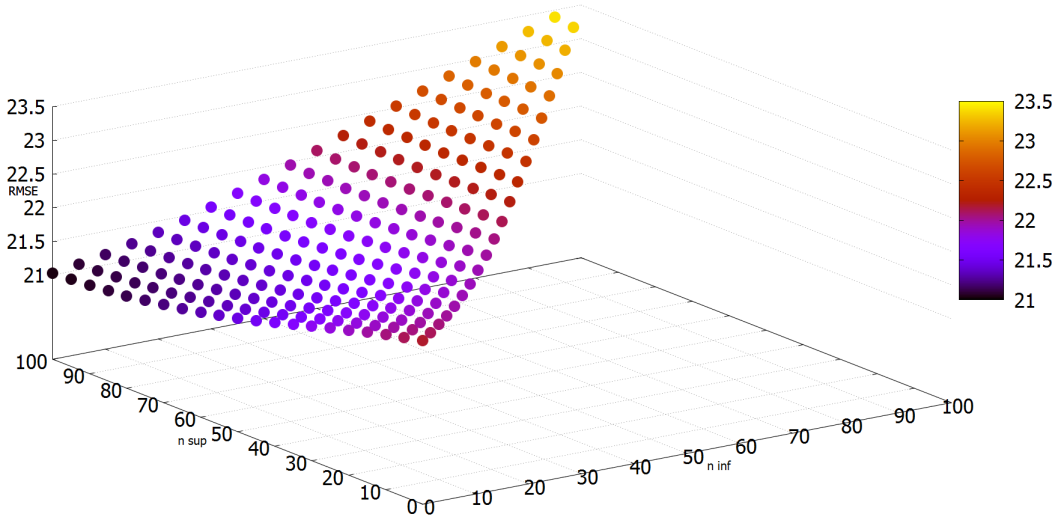

Figura 5.8: Efeito dos parâmetros n_{inf} e n_{sup} para a recomendação híbrida - método 2.

Novamente, o melhor resultado obtido foi para $n_{inf} = 0$ e $n_{sup} = 100$, para os quais $RMSE = 21,02432819$.

Este sistema híbrido seguiu exatamente o mesmo padrão que o anterior, e os resultados foram muito semelhantes. Não é possível afirmar se um é melhor que o outro através desta pequena diferença nos resultados. Porém, é possível afirmar que os dois sistemas híbridos foram igualmente bons e superaram os demais sistemas de recomendação avaliados por este trabalho.

5.5 Tamanho do *dataset*

Por último, o efeito do tamanho do *dataset* foi avaliado.

A filtragem colaborativa em especial depende de uma grande quantidade de dados disponíveis para produzir boas recomendações, e este teste avaliou justamente este ponto.

Para isto, foram realizadas 3 baterias de teste, cada uma delas utilizando um *dataset* de treino diferente. A fim de facilitar a identificação, será adotada a seguinte

nomenclatura:

1. **Dataset treino 0.1** - corresponde a 10% do *dataset* de treino original
2. **Dataset treino 0.5** - corresponde a 50% do *dataset* de treino original
3. **Dataset treino 1.0** - é o *dataset* de treino original

Para todos eles, foi utilizado o mesmo *dataset* de teste.
A tabela 5.18 contém algumas estatísticas referentes a estes conjuntos de dados.

Tabela 5.18: Estatísticas das 3 versões de *dataset* de treino.

Dataset	**Número de itens**	**Número de notas**	**Média de notas por item**
0.1	14604	1128174	77,3
0.5	31715	5734669	180,8
1.0	98210	114434161	1165,2

Para avaliar o efeito do tamanho do *dataset*, foram utilizadas duas métricas: o RMSE como de costume e a cobertura (*coverage*).

A cobertura mede quantas previsões o sistema foi capaz de gerar com relação a todas as tentativas. Um sistema de recomendação pode não ser capaz de prever uma nota para um item se este item simplesmente não existir no *dataset* de treino ou se não existirem informações suficientes a respeito do usuário (especialmente na filtragem colaborativa).

A cobertura pode ser calculada através da seguinte equação [50] [51]:

$$Cobertura = \frac{número\ de\ previsões\ bem\ sucedidas}{número\ total\ de\ tentativas\ de\ previsão} \tag{5.1}$$

As figuras 5.9 e 5.10 contêm respectivamente o RMSE e a cobertura para diferentes algoritmos de recomendação nas três versões de *dataset* de treino. Na figura 5.10 não é possível diferenciar muito bem os diferentes sistemas de recomendação pois eles apresentaram comportamentos extremamente semelhantes.

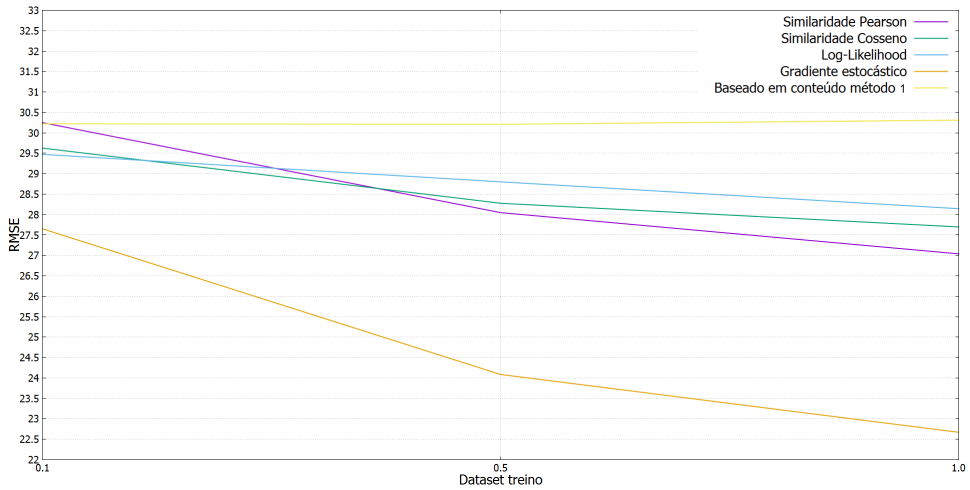

Figura 5.9: Efeito do tamanho do *dataset* para o RMSE.

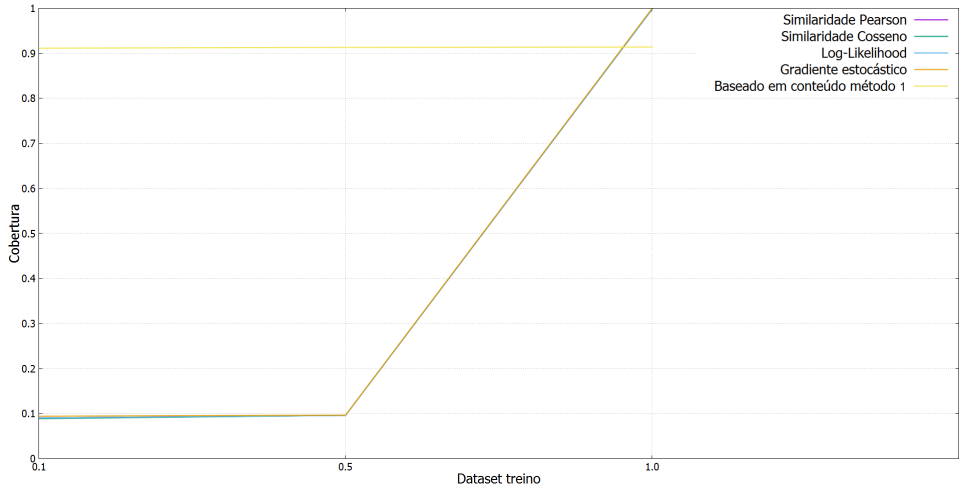

Figura 5.10: Efeito do tamanho do *dataset* para a cobertura.

Nas figuras acima, o método do de recomendação baseada em conteúdo foi o método 1 de recomendação personalizada com o peso de similaridade $\alpha = 1$, e o gradiente estocástico utilizou a classe *RatingSGDFactorizer* com o seguinte conjunto de parâmetros:

- $\lambda = 0,6$
- $n_f = 400$
- $n_{it} = 10$
- $\gamma = 0,001$

Em ambas as figuras é possível notar que o desempenho geral do sistema melhorou à medida que o tamanho do *dataset* de treino aumentou, exceto para a

recomendação baseada em conteúdo, que manteve um desempenho constante independente do tamanho do *dataset*. Isto significa que este último é mais robusto a problemas de *cold-start*, e que a qualidade da filtragem colaborativa de fato depende da qualidade do *dataset* de treino.

Na figura 5.9 é possível notar que a melhora no RMSE é mais ou menos linearmente proporcional ao aumento no tamanho do *dataset*.

Já na figura 5.10 o aumento na cobertura ocorre de forma muito mais expressiva entre os *datasets* 0.5 e 1.0 do que entre os *datasets* 0.1 e 0.5. Porém, analisando a tabela 5.18, vemos que o mesmo ocorre para o número médio de notas por item. Além disso, como os *datasets* 0.1 e 0.5 possuem muito menos itens, é natural que o *dataset* de teste contenha vários itens que não estavam presentes nesses *datasets* de treino.

5.6 Sumarizando resultados

A tabela 5.19 contém o melhor resultado de RMSE para cada um dos principais algoritmos avaliados utilizando os *datasets* de treino e de teste originais.

Tabela 5.19: Resumo dos resultados de RMSE.

Algoritmo	**RMSE**
Nota média do item	36,5799561
Nota média do item ajustada pelo usuário	30,4975990
Nota média do usuário	28,2208995
Similaridade Pearson	27,0378967
Similaridade Cosseno	27,6971683
Similaridade Log-Likelihood	28,1437640
Fatoração SVD - Gradiente estocástico [(1)]	22,5894508
Baseada em conteúdo - método 1 [(2)]	30,3103701
Baseada em conteúdo - método 2	31,5042978
Híbrido - método 1 [(3)]	21,0453157
Híbrido - método 2 [(4)]	21,02432819

[(1)]: classe *RatingSGDFactorizer* com parâmetros $\lambda = 0,6$; $n_{it} = 10$; $n_f = 400$; $\gamma = 0,001$

[(2)]: parâmetro $\alpha = 1$

[(3)]: classe *ParellelSGDFactorizer* com mesmos parâmetros de [(1)] + método 1 de

recomendação personalizada baseada em conteúdo com mesmos parâmetros de [2]; $n_{inf} = 0$; $n_{sup} = 100$

[4]: classe *ParellelSGDFactorizer* com mesmos parâmetros de [1] + método 2 de recomendação personalizada baseada em conteúdo; $n_{inf} = 0$; $n_{sup} = 100$

5.7 Conclusão

Este capítulo apresentou os resultados de avaliação utilizando o RMSE para os algoritmos descritos no capítulo anterior. Quando possível, os resultados também foram avaliados para diferentes conjuntos de parâmetros para que o efeito dos mesmos fosse analisado.

De forma geral, os resultados mostraram que as técnicas de filtragem híbrida são capazes de produzir melhores recomendações e de forma mais robusta ao problema de *cold-start*.

As técnicas de filtragem colaborativa (principalmente pelo método do gradiente estocástico) produziram resultados melhores que os da filtragem baseada em conteúdo, mas isto não necessariamente quer dizer que a filtragem baseada em conteúdo não é capaz de produzir bons resultados, mas sim que os métodos utilizados neste trabalho não são os mais eficientes possíveis.

Capítulo 6

Conclusão

A biblioteca Mahout se mostrou uma boa escolha para este trabalho, pois ela implementa diversas técnicas que foram avaliadas e ainda possibilita que suas classes nativas sejam facilmente estendidas para que o programador implemente novas técnicas, como as de filtragem baseada em conteúdo que foram avaliadas aqui.

Como pôde ser visto no capítulo anterior, as técnicas híbridas se mostraram as mais promissoras, resultando em menores valores de RMSE. Na verdade, esta conclusão comprova o efeito que foi visto no Prêmio Netflix, em que o algoritmo vencedor era na verdade composto por 107 algoritmos diferentes [15]. Vale lembrar ainda que as técnicas de hibridização utilizadas neste trabalho foram simples, e mesmo com este grau de simplicidade foram capazes de produzir melhores resultados. Além disso, não só reduziram o RMSE como também tornaram o sistema mais robusto ao problema de *cold-start* de usuário.

A qualidade das técnicas híbridas utilizadas neste trabalho depende diretamente da qualidade individual das técnicas base de filtragem colaborativa e de recomendação baseada em conteúdo. Dentre elas, ficou claro especificamente para a filtragem colaborativa que a qualidade dos resultados depende diretamente da qualidade do *dataset* de treino. Um dos principais fatores é o tamanho do *dataset*, pois quanto maior ele for, maior a disponibilidade de informações para produzir um modelo preditivo. Porém, o tipo de informação disponível também é um fator relevante. No Prêmio Netflix, a técnica de fatoração SVD de matrizes - que foi um dos pilares do algoritmo vencedor - utilizou também dinâmicas temporais, que avaliam como as notas dos itens e dos usuários se comportam ao longo do tempo [7]. Para isto, a data original de cada nota do *dataset* deveria estar disponível, e esta é uma informação que não estava presente do *dataset* utilizado neste trabalho.

De forma geral, a filtragem colaborativa simples também se mostrou uma boa alternativa para aplicações reais, principalmente o modelo de fatores latentes com fatoração SVD de matrizes através do método do gradiente estocástico. Esta técnica foi capaz de prever notas de forma muito mais assertiva que as demais utilizadas.

Além disso, esta é uma das técnicas implementadas pela biblioteca Mahout de forma distribuída, podendo assim ser utilizada em conjunto com a ferramenta Hadoop de armazenamento e processamento distribuídos. Desta forma, até mesmo um programador leigo em aprendizado de máquina poderia desenvolver um site com comércio eletrônico provido de um sistema de recomendação robusto e que produz boas recomendações. A maior dificuldade seria em determinar o conjunto ideal de parâmetros.

Infelizmente não foram encontrados outros trabalhos na literatura que tenham avaliado o desempenho de diversos tipos de sistemas de recomendação para o mesmo *dataset* utilizado neste trabalho, portanto fica difícil fazer uma comparação direta entre os resultados deste trabalho e de outros visto que eles dependem diretamente do *dataset* utilizado.

6.1 Trabalhos futuros

6.1.1 Tempo de processamento

Este trabalho focou em avaliar a qualidade das notas previstas por um sistema de recomendação, mas um ponto tão importante quanto é o tempo necessário para produzir estas recomendações. Este ponto chegou a ser mencionado nos casos em que foi realmente um empecilho para o andamento do projeto, porém em uma aplicação real ele seria muito mais crítico. Um sistema de recomendação real não pode demorar horas para produzir uma lista de 10 produtos recomendados para um cliente.

6.1.2 Processamento *offline*

Uma das principais estratégias para reduzir o tempo de processamento seria:

(a) Calcular as similaridades *offline* nos casos de filtragem colaborativa baseada em similaridades

(b) Fazer a decomposição da matriz *offline* nos casos de filtragem colaborativa baseada em modelos

Estas informações calculadas *offline* seriam armazenadas em um banco de dados. Desta forma, grande parte das informações necessárias para produzir uma recomendação já estariam disponíveis em tempo real, agilizando assim o sistema de recomendação.

Um dos desafios desta abordagem é determinar a frequência com que os cálculos *offline* devem ser realizados, pois é de se esperar que em algum momento as similaridades (ou as matrizes da decomposição) sejam impactadas pelas novas avaliações dos itens.

6.1.3 Avaliação subjetiva

Este trabalho avaliou os sistemas de recomendação através de uma métrica objetiva, mas a percepção do usuário do sistema também é muito importante. Um dos efeitos que pode ser avaliado através de pesquisas com usuários reais é a serendipidade, que é a capacidade de produzir recomendações inesperadas mas ainda assim relevantes [52].

Sistemas de recomendação que focam na serendipidade costumam resultar em métricas de precisão de nota (como o RMSE) piores, porém aos olhos do usuário isto não necessariamente implica em recomendações piores. Sistemas de recomendação tradicionais podem recomendar itens que o usuário talvez conseguisse achar por si próprio, enquanto que sistemas de recomendação com foco na serendipidade tentam recomendar itens que o usuário provavelmente não conseguiria achar [52].

Referências Bibliográficas

[1] THE ECONOMIST. "The phone of the future". . http://www.economist.com/node/8312260, nov. 2006. Acessado em dezembro de 2016.

[2] U.S. DEPARTMENT OF COMMERCE. "Quarterly Retail E-Commerce Sales". . https://goo.gl/pacujP, nov. 2016. Acessado em dezembro de 2016.

[3] RICCI, F., ROKACH, L., SHAPIRA, B., et al. *Introduction to Recommender Systems Handbook*. 1 ed. USA, Springer-Science+Business Media, 2010. ISBN: 9780387858197.

[4] FRIAS-MARTINEZ, E., CHEN, S. Y., LIU, X. "Evaluation of a personalized digital library based on cognitive styles: Adaptivity vs. adaptability", *International Journal of Information Management*, v. 29, pp. 48–56, 2009.

[5] FRIAS-MARTINEZ, E., MAGOULAS, G., CHEN, S. Y., et al. "Automated user modeling for personalized digital libraries", *International Journal of Information Management*, v. 26, pp. 234–248, 2006.

[6] PARK, D. H., KIM, H. K., CHOI, I. Y., et al. "A literature review and classification of recommender systems research", *Expert Systems with Applications*, v. 39, pp. 10059–10072, 2012.

[7] KOREN, Y., BELL, R., VOLINSKY, C. "Matrix Factorization Techniques for Recommender Systems", *Computer*, v. 42, n. 8, pp. 30–37, ago. 2009. ISSN: 0018-9162. doi: 10.1109/MC.2009.263. Disponível em: ¡http://dx.doi.org/10.1109/MC.2009.263¿.

[8] AMAZON. "Amazon.co.uk". . https://www.amazon.co.uk/, dec. 2016. Acessado em dezembro de 2016.

[9] ANDERSON, C. *The Long Tail: Why the Future of Business is Selling Less of More*. 1 ed. USA, Hyperion, 2008. ISBN: 9781401309664.

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

[11] LÜ, L., MEDO, M., YEUNG, C. H., et al. "Recommender systems", *Physics Reports*, v. 519, pp. 1 – 49, 2012. ISSN: 0370-1573. doi: http://dx.doi.org/10.1016/j.physrep.2012.02.006. Disponível em: ¡http://www.sciencedirect.com/science/article/pii/S0370157312000828¿.

[12] WIKIPEDIA. "Cauda longa". . https://pt.wikipedia.org/wiki/Cauda_longa. Acessado em dezembro de 2016.

[13] BENNETT, J., LANNING, S. "The Netflix prize", *Proceedings of KDD Cup and Workshop*, pp. 3–6, 2007.

[14] NETFLIX. "Netflix Prize". . http://www.netflixprize.com/, 2009. Acessado em dezembro de 2016.

[15] AMATRIAIN, X., BASILICO, J. "Netflix Recommendations: Beyond the 5 stars". . http://techblog.netflix.com/2012/04/netflix-recommendations-beyond-5-stars.html, apr. 2012. Acessado em dezembro de 2016.

[16] JANNACH, D., ZANKER, M., FELFERNIG, A., et al. *Recommender Systems: An Introduction*. USA, Cambridge University Press, 2010. ISBN: 9780521493369.

[17] SALTON, G., MCGILL, M. J. *Introduction to Modern Information Retrieval*. New York, NY, USA, McGraw-Hill, Inc., 1986. ISBN: 0070544840.

[18] DEEZER. "Deezer". . https://www.deezer.com/br/, dec. 2016. Captura de tela realizada em dezembro de 2016.

[19] YU, F., ZENG, A., GILLARD, S., et al. "Network-based recommendation algorithms: A review", *Physica A: Statistical Mechanics and its Applications*, v. 452, pp. 192 – 208, 2016. ISSN: 0378-4371. doi: http://dx.doi.org/10.1016/j.physa.2016.02.021. Disponível em: ¡http://www.sciencedirect.com/science/article/pii/S0378437116001874¿.

[20] BURKE, R. "Knowledge-based recommender systems", *Encyclopedia of Library and Information Science*, v. 69, n. 32, pp. 180–200, 2000.

[21] BUCHANAN, B. G., SHORTLIFFE, E. H. *Rule Based Expert Systems: The Mycin Experiments of the Stanford Heuristic Programming Project (The Addison-Wesley Series in Artificial Intelligence)*. Boston, MA, USA, Addison-Wesley Longman Publishing Co., Inc., 1984. ISBN: 0201101726.

[22] BURKE, R. “Hybrid Recommender Systems: Survey and Experiments”, *User Modeling and User-Adapted Interaction*, v. 12, n. 4, pp. 331–370, nov. 2002. ISSN: 0924-1868. doi: 10.1023/A:1021240730564. Disponível em: ¡http://dx.doi.org/10.1023/A:1021240730564¿.

[23] KIM, B. M., LI, Q., PARK, C. S., et al. “A New Approach for Combining Content-based and Collaborative Filters”, *J. Intell. Inf. Syst.*, v. 27, n. 1, pp. 79–91, jul. 2006. ISSN: 0925-9902. doi: 10.1007/s10844-006-8771-2. Disponível em: ¡http://dx.doi.org/10.1007/s10844-006-8771-2¿.

[24] SCHEIN, A. I., POPESCUL, A., UNGAR, L. H., et al. “Methods and Metrics for Cold-start Recommendations”. In: *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR ’02, pp. 253–260, New York, NY, USA, 2002. ACM. ISBN: 1-58113-561-0. doi: 10.1145/564376.564421. Disponível em: ¡http://doi.acm.org/10.1145/564376.564421¿.

[25] YAHOO! “Yahoo Webscope”. . https://webscope.sandbox.yahoo.com/, dec. 2016. Acessado em dezembro de 2016.

[26] YAHOO WEBSCOPE. “Yahoo! Music User Ratings of Musical Artists, version 1.0”. . https://webscope.sandbox.yahoo.com/catalog.php?datatype=r&did=1, dec. 2016. Acessado em dezembro de 2016.

[27] AMAZON WEB SERVICES. “O que é a computação em nuvem?” . https://aws.amazon.com/pt/what-is-cloud-computing/. Acessado em dezembro de 2016.

[28] WIKIPEDIA. “Computação em nuvem”. . https://pt.wikipedia.org/wiki/Computa\unhbox\voidb@x\setbox\z@\hbox{c}\accent24c~ao_em_nuvem. Acessado em dezembro de 2016.

[29] MCKENDRICK, J. “Cloud Computing Boosts Next Generation of Startups, Survey Shows”. . http://www.forbes.com/sites/joemckendrick/2013/02/20/cloud-computing-boosts-next-generation-of-startups-survey-shows/, feb. 2013. Acessado em dezembro de 2016.

[30] AMAZON. “Amazon EC2”. . https://aws.amazon.com/pt/ec2/. Acessado em dezembro de 2016.

[31] OWEN, S., ANIL, R., DUNNING, T., et al. *Mahout in Action*. Greenwich, CT, USA, Manning Publications Co., 2011. ISBN: 1935182684, 9781935182689.

[32] MERRIAM-WEBSTER. "Definition of Database". . https://www.merriam-webster.com/dictionary/database. Acessado em dezembro de 2016.

[33] SPOTIFY. "Spotify Web API". . https://developer.spotify.com/web-api/. Acessado em dezembro de 2016.

[34] WIKIPEDIA. "Levenshtein distance". . https://en.wikipedia.org/wiki/Levenshtein_distance. Acessado em dezembro de 2016.

[35] DUNNING, T. "Accurate Methods for the Statistics of Surprise and Coincidence", *COMPUTATIONAL LINGUISTICS*, v. 19, n. 1, pp. 61–74, 1993.

[36] DUNNING, T. "Surprise and Coincidence". . http://tdunning.blogspot.com.br/2008/03/surprise-and-coincidence.html, mar. 2008. Acessado em dezembro de 2016.

[37] APACHE MAHOUT. "GitHub Mahout LogLikelihoodSimilarity.java". . https://github.com/apache/mahout/blob/master/mr/src/main/java/org/apache/mahout/cf/taste/impl/similarity/LogLikelihoodSimilarity.java, . Acessado em dezembro de 2016.

[38] TAKÁCS, G., PILÁSZY, I., NÉMETH, B., et al. "On the Gravity Recommendation System". In: *Proc. of KDD Cup Workshop at SIGKDD'07, 13th ACM Int. Conf. on Knowledge Discovery and Data Mining*, pp. 22–30, San Jose, CA, USA, 2007.

[39] FUNK, S. "Netflix Update: Try This at Home". . http://sifter.org/~simon/journal/20061211.html, dec. 2006. Acessado em dezembro de 2016.

[40] KOREN, Y. "Factorization Meets the Neighborhood: A Multifaceted Collaborative Filtering Model". In: *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '08, pp. 426–434, New York, NY, USA, 2008. ACM. ISBN: 978-1-60558-193-4. doi: 10.1145/1401890.1401944. Disponível em: ¡http://doi.acm.org/10.1145/1401890.1401944¿.

[41] PATEREK, A. "Improving regularized singular value decomposition for collaborative filtering". In: *Proc. KDD Cup Workshop at SIGKDD'07, 13th ACM Int. Conf. on Knowledge Discovery and Data Mining*, pp. 39–42, 2007. Disponível em: ¡http://serv1.ist.psu.edu:8080/viewdoc/summary;jsessionid=CBC0A80E61E800DE518520F9469B2FD1?doi=10.1.1.96.7652¿.

[42] APACHE MAHOUT. "GitHub Mahout RatingSGDFactorizer.java". . https://github.com/apache/mahout/blob/master/mr/src/main/java/org/apache/mahout/cf/taste/impl/recommender/svd/RatingSGDFactorizer.java, . Acessado em dezembro de 2016.

[43] RECHT, B., RE, C., WRIGHT, S. J., et al. "Hogwild: A Lock-Free Approach to Parallelizing Stochastic Gradient Descent". In: *NIPS*, pp. 693–701, 2011.

[44] INSIGHT FELLOWS PROGRAM. "Explicit Matrix Factorization: ALS, SGD, and All That Jazz". . https://blog.insightdatascience.com/explicit-matrix-factorization-als-sgd-and-all-that-jazz-b00e4d9b21ea#.280ec2tuv, mar. 2016. Acessado em dezembro de 2016.

[45] AKYILDIZ, B. "Alternating Least Squares Method for Collaborative Filtering". . https://bugra.github.io/work/notes/2014-04-19/alternating-least-squares-method-for-collaborative-filtering/, apr. 2014. Acessado em dezembro de 2016.

[46] ABERGER, C. R. "Recommender: An Analysis of Collaborative Filtering Techniques". . http://stanford.io/28OR3XE. Acessado em dezembro de 2016.

[47] VAN DEN OORD, A., DIELEMAN, S., SCHRAUWEN, B. "Deep content-based music recommendation". In: *Advances in Neural Information Processing Systems 26 (2013)*, v. 26, p. 9. Neural Information Processing Systems Foundation (NIPS), 2013. ISBN: 9781632660244. Disponível em: ¡http://papers.nips.cc/paper/5004-deep-content-based-music-recommendation.pdf¿.

[48] SHAO, B., OGIHARA, M., WANG, D., et al. "Music Recommendation Based on Acoustic Features and User Access Patterns", *Trans. Audio, Speech and Lang. Proc.*, v. 17, n. 8, pp. 1602–1611, nov. 2009. ISSN: 1558-7916. doi: 10.1109/TASL.2009.2020893. Disponível em: ¡http://dx.doi.org/10.1109/TASL.2009.2020893¿.

[49] WIKIPEDIA. "Mel-frequency cepstrum". . https://en.wikipedia.org/wiki/Mel-frequency_cepstrum. Acessado em dezembro de 2016.

[50] GOOD, N., SCHAFER, J. B., KONSTAN, J. A., et al. "Combining Collaborative Filtering with Personal Agents for Better Recommendations". In: *Proceedings of the Sixteenth National Conference on Artificial Intelligence and the Eleventh Innovative Applications of Artificial Intelligence Conference Innovative Applications of Artificial Intelligence*, AAAI '99/IAAI

'99, pp. 439–446. American Association for Artificial Intelligence, 1999. ISBN: 0-262-51106-1. Disponível em: ¡http://dl.acm.org/citation.cfm?id=315149.315352¿.

[51] CASINELLI, P. "Evaluating and Implementing Recommender Systems As Web Services Using Apache Mahout". . http://cslab1.bc.edu/~csacademics/pdf/14Casinelli.pdf, 2014. Acessado em dezembro de 2016.

[52] KOTKOV, D., WANG, S., VEIJALAINEN, J. "A survey of serendipity in recommender systems", *Knowledge-Based Systems*, v. 111, pp. 180–192, 2016. ISSN: 0950-7051. doi: http://dx.doi.org/10.1016/j.knosys.2016.08.014. Disponível em: ¡http://www.sciencedirect.com/science/article/pii/S0950705116302763¿.