**RICARDO MITOLLO BERTANI**

# Combinando novidade e popularidade em recomendações personalizadas através do aprendizado do perfil do usuário

São Paulo
2020

**RICARDO MITOLLO BERTANI**

# Combinando novidade e popularidade em recomendações personalizadas através do aprendizado do perfil do usuário

**Versão Corrigida**
(Versão original encontra-se na unidade que aloja o Programa de Pós-graduação)

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do Título de Mestre em Ciências.

Orientadora:
Profa. Dra. Anna Helena Reali Costa

São Paulo
2020

Autorizo a reprodução e divulgação total ou parcial deste trabalho, por qualquer meio convencional ou eletrônico, para fins de estudo e pesquisa, desde que citada a fonte.

Este exemplar foi revisado e corrigido em relação à versão original, sob responsabilidade única do autor e com a anuência de seu orientador.

São Paulo, _____ de _____ de _____

Assinatura do autor: _____

Assinatura do orientador: _____

Catalogação-na-publicação

Bertani, Ricardo
Combinando novidade e popularidade em recomendações personalizadas através do aprendizado do perfil do usuário / R. Bertani, A. Costa -- versão corr. -- São Paulo, 2020.
62 p.

Dissertação (Mestrado) - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Computação e Sistemas Digitais.

1.INTELIGÊNCIA ARTIFICIAL 2.APRENDIZADO DE MÁQUINA 3.PERSONALIDADE I.Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Computação e Sistemas Digitais II.t. III.Costa, Anna

Dedido este trabalho à minha família. Em especial, à minha avó Angelina Mitollo (*in memoriam*) e ao meu pai Edenaldo Bertani (*in memoriam*), que tanto se alegraram pelo meu ingresso nesta jornada. Onde estiverem, sei que estão muito felizes com esta realização.

**AGRADECIMENTOS**

Primeiramente, gostaria de agradecer à minha orientadora, Anna Helena Reali Costa, por me receber em seu grupo de pesquisa, acreditar neste trabalho e sempre me encorajar. Todo seu conhecimento e compromisso com o rigor científico, foram fundamentais para atingirmos alta qualidade em minha pesquisa.

Agradeço também a Wei Zeng e An Zeng, autores do artigo Zeng et al. (2013), por todo apoio prestado durante a reprodução do algoritmo $MDHS$, utilizado como base para este trabalho.

Agradeço toda minha família, que sempre me apoiou e torceu pelo sucesso deste trabalho, em especial minha mãe Clarice Mitollo Bertani.

Por fim, agradeço também minha noiva Luiza Botti por todo amor e compreensão.

**RESUMO**

Sistemas de recomendação têm sido amplamente utilizados por grandes empresas do segmento de *e-commerce* como ferramenta de auxílio na busca de conteúdos relevantes de acordo com as preferências particulares dos usuários. Uma ampla variedade de algoritmos tem sido apresentada na literatura com o objetivo de aprimorar o processo de geração de recomendações; em particular, destacam-se aqueles baseados em filtragem colaborativa, os quais ainda falham em dados esparsos, afetando a qualidade das recomendações. Para mitigar essa lacuna, um algoritmo híbrido baseado em difusão foi proposto na literatura; no entanto, este algoritmo não diferencia os usuários de acordo com seus perfis. Nesta pesquisa, um novo algoritmo é apresentado para o aprendizado do perfil dos usuários e consequente geração de recomendações personalizadas através de difusão, combinando itens pouco conhecidos (novidade) a itens populares. Os experimentos realizados em três conjuntos de dados bem conhecidos na literatura mostram que os resultados superam aqueles obtidos pelo algoritmo original baseado em difusão, assim como os obtidos pelo algoritmo tradicional colaborativo, sob as mesmas configurações.

**Palavras-chave:** Sistemas de Recomendação, Esparsidade dos Dados, Aprendizado de Máquina, Algoritmos Baseados em Difusão, Perfil do Usuário

## ABSTRACT

Recommender systems have been widely used by large companies in the e-commerce segment as an aid tool in finding relevant content according to users' particular preferences. A wide variety of algorithms have been presented in the literature to improve the recommendation generation process; In particular, those based on collaborative filtering, which still fail with sparse data, affecting the quality of recommendations. In order to mitigate this gap, a diffusion-based hybrid algorithm has been proposed in the literature; However, this algorithm does not differentiate users according to their profiles. In this research, a new algorithm is presented for learning the users profile and consequent generation of personalized recommendations through diffusion, combining little known items (novelty) with popular items. Experiments performed on three well-known data sets in the literature show that the results outperform those obtained by the original diffusion-based algorithm, as well as those obtained by the traditional collaborative algorithm, under the same configurations.



**Keywords:** Recommender Systems, Data Sparsity, Machine Learning, Diffusion-based Algorithms, User Profile

## LISTA DE ILUSTRAÇÕES


1 Arquitetura de alto nível simplificada de um recomendador FBC. . . . . 8

2 Classificação de Algoritmos de Filtragem Colaborativa (FC). . . . . 11

3 Principais passos de algoritmos FC baseados em fatoração de matrizes. 19

4 Exemplo de um grafo bipartido de usuários e itens em FC. . . . . 22

5 Os três passos de propagação com o algoritmo $HS$ em um grafo bipartido usuário-item. . . . . 30

6 Os três passos de propagação com o algoritmo $MD$ em um grafo bipartido usuário-item. . . . . 30

7 As duas fases de UPOD. . . . . 34

8 Esquema macro da fase de treinamento de UPOD . . . . . 35

## LISTA DE TABELAS


1 Classificação dos algoritmos de filtragem de acordo com alguns critérios 13

2 Nível de esparsidade das bases de dados consideradas. . . . . 43

3 Tamanho das bases de treinamento e testes . . . . . 47

4 Resultados Experimentais para as três bases de dados. . . . . 51

## LISTA DE ABREVIAÇÕES

FBC Filtragem baseada em Conteúdo

FBCO Filtragem baseada em Conhecimento

FC Filtragem Colaborativa

FD Filtragem Demográfica

HB Algoritmos híbridos de filtragem

MD Mass Diffusion

HS Heat Spreading

MDHS Algoritmo híbrido baseado em difusão de massa

UPOD User Profile Oriented Diffusion

CCP Coeficiente de Correlação de Pearson

SC Semelhança de Cossenos

## LISTA DE SÍMBOLOS

| | |
|---|---|
| $U$ | Conjunto de usuários presentes no sistema |
| $I$ | Conjunto de itens presentes no sistema |
| $E$ | Conjunto de relacionamentos entre usuários e itens no sistema |
| $N$ | Número total de usuários em um Sistema de Recomendação |
| $M$ | Número total de itens em um Sistema de Recomendação |
| $K$ | Número total de interações entre usuários e itens |
| $Feat$ | Mapa entre o ID dos usuários e seu conjunto de atributos |
| $L$ | Lista de Recomendação |
| $G$ | Grafo bipartido usuário-item |
| $r_{u,i}$ | Valor de *rating* atribuído ao item $i$ pelo usuário $u$ |
| $\bar{r}_u$ | Valor médio de *rating* atribuído pelo usuário $u$ a todos itens com os quais ele teve alguma interação |
| $Vz$ | Tamanho da vizinhança entre usuários |
| $\lambda$ | Grau de mistura entre $MD$ e $HS$ em $MDHS$ |
| $\Lambda$ | Conjunto de valores possíveis de $\lambda$ |
| $k$ | Número de grupos de usuários em *UPOD* |
| $k_{min}$ | Número mínimo de grupos de usuários |
| $k_{max}$ | Número máximo de grupos de usuários |
| $C^k$ | Conjunto de k clusters (partição) |
| $Opt(C^k)$ | Critério de clusterização baseado em entropia |
| $Metric$ | Conjunto de métricas consideradas para o cálculo do melhor valor de $\lambda$ no algoritmo *UPOD* |

**SUMÁRIO**


**1 Introdução** **1**

1.1 Motivação . . . . . 1

1.2 Problema e Hipótese . . . . . 2

1.3 Objetivo . . . . . 2

1.4 Organização do texto . . . . . 3

**2 Fundamentos** **4**

2.1 Conceitos Básicos . . . . . 4

2.2 Algoritmos de Filtragem Baseada em Conteúdo . . . . . 7

2.3 Algoritmos de Filtragem Baseada em Conhecimento . . . . . 9

2.4 Algoritmos de Filtragem Demográfica . . . . . 9

2.5 Algoritmos de Filtragem Colaborativa . . . . . 10

2.6 Algoritmos Híbridos . . . . . 11

2.7 Análise comparativa entre os principais tipos de algoritmos de filtragem 13

2.8 Algoritmos colaborativos baseados em vizinhança . . . . . 13

2.8.1 Medidas de similaridade . . . . . 14

2.8.2 Identificação de vizinhança . . . . . 15

2.8.3 Predição de *ratings* . . . . . 15

2.8.4 Ranqueamento e seleção de itens . . . . . 16

2.8.5 O problema da partida a frio . . . . . 16

2.8.6 O problema da esparsidade dos dados . . . . . 17

2.9 Abordagens baseadas em Aprendizado Profundo (*Deep Learning*) . . . 23

2.10 Considerações finais . . . . . 25

**3 Algoritmos de Recomendação Baseados em Difusão 26**

**4 Proposta 33**

4.1 Reprodução do algoritmo $MDHS$ . . . . . 33

4.2 Algoritmo $UPOD$ . . . . . 34

4.3 Detalhes de Implementação . . . . . 41

**5 Resultados Experimentais e Discussão 42**

5.1 Bases de Dados . . . . . 42

5.2 Algoritmos utilizados para comparação . . . . . 43

5.3 Método de avaliação e métricas . . . . . 44

5.3.1 *Recall* . . . . . 44

5.3.2 *Precision* . . . . . 45

5.3.3 *Ranking Score* . . . . . 45

5.4 Resultados e Discussão . . . . . 47

**6 Conclusão e Trabalhos Futuros 52**

**Referências 53**

**Apêndice 59**

# 1 INTRODUÇÃO

Sistemas de Recomendação são softwares capazes de predizer o nível de interesse de um usuário por um conteúdo específico de acordo com suas preferências particulares. De acordo com Ricci et al. (2010) e Kotkov, Wang e Veijalainen (2016), o termo “item” é utilizado para referir-se a todo conteúdo que é sugerido ao usuário de um Sistema de Recomendação, podendo representar músicas, livros, notícias ou até mesmo amigos em redes sociais. Uma vez que o sistema prediz o nível de interesse de um usuário por um conjunto de itens, é gerada uma lista de recomendação contendo aqueles mais relevantes ao usuário.

## 1.1 Motivação

Os Sistemas de Recomendação são frequentemente utilizados por diversas empresas com o intuito de auxiliar seus clientes na busca dos conteúdos mais relevantes, frente ao grande volume de dados disponível hoje na Internet. Sugestões de filmes feita no site do Netflix, assim como as recomendações de produtos similares a aqueles adquiridos no site da Amazon ou até mesmo as sugestões de amizades em redes sociais tais como o Facebook são exemplos práticos de aplicações de Sistemas de Recomendação.

De acordo com Ricci et al. (2010), as aplicações mais comuns dos Sistemas de Recomendação são relacionadas a entretenimento (recomendações de músicas e vídeos); conteúdo (recomendações de notícias, páginas web, aplicações de e-learning, filtro de e-mail); e-commerce (recomendações de produtos feitas a consumidores); e serviços (recomendações de serviços de viagem e recomendações de imóveis para venda, por exemplo). Para Lu et al. (2015), ainda existem aplicações nos seguintes domínios: governo eletrônico (*e-government)*, negócios eletrônicos (*e-business)*, bibliotecas digitais (*e-library*) e educacionais (*e-learning*).

Devido à grande popularidade deste tipo de sistema e a necessidade de garantir

que os mesmos ofereçam recomendações com alta relevância e qualidade, torna-se cada vez mais necessário o aprimoramento de algoritmos para resolução de problemas nas principais abordagens utilizadas na área.

## 1.2 Problema e Hipótese

Na literatura dos Sistemas de Recomendação diversos autores ressaltam a existência do problema conhecido por "esparsidade dos dados" (ou *data sparsity*, em inglês) (BOBADILLA et al., 2013), (LU et al., 2015),(LU et al., ), (ZHANG; ZENG, 2015), (BEEL et al., 2016), o qual caracteriza-se pela existência de um alto número de usuários e itens (conteúdos passíveis de serem recomendados) no sistema e uma baixa quantia de avaliações de itens por parte de usuários. Tal situação afeta diretamente a qualidade das recomendações pois impede que métodos colaborativos tradicionais consigam determinar a similaridade entre usuários de forma eficiente.

Um algoritmo híbrido baseado em processos físicos de difusão de massa foi apresentado em Zhou et al. (2010) e demonstrou ser eficiente na amenização deste problema, porém o parâmetro utilizado para sintonizar as proporções de popularidade e novidade nas recomendações tem valor sempre fixo para todos usuários do sistema, sem distinção por perfil.

Dado este contexto, a hipótese apresentada nesta pesquisa é que é possível inferir o melhor parâmetro de sintonização a partir dos perfis dos usuários e com isto gerar recomendações personalizadas e com maior acurácia.

## 1.3 Objetivo

Esta pesquisa tem por objetivo propor um novo algoritmo, chamado $UPOD$ (*User Profile Oriented Diffusion)*, baseado em processos de difusão de massa, para a geração de recomendações personalizadas através do aprendizado do perfil do usuário. O algoritmo proposto baseia-se na solução proposta por Zhou et al. (2010) e a adapta para permitir combinar nas recomendações, de forma automática, itens populares com itens pouco conhecidos (novidade) através do aprendizado do perfil do usuário.

Tal aprendizado do perfil utiliza informações sobre as características individuais dos usuários, geralmente presentes em bases de dados voltadas a sistemas de recomendação. Isto significa que o algoritmo proposto pode ser facilmente integrado a outros dados, desde que algumas informações sobre os usuários estejam disponíveis nas bases de dados.

## 1.4 Organização do texto

No capítulo 2 são apresentados os conceitos fundamentais para a compreensão dos sistemas de recomendação, bem como seus principais tipos de algoritmos, as principais dificuldades e soluções conhecidas. Em seguida, no capítulo 3 são descritos os algoritmos baseados em processos de difusão de massa e sua contribuição para a amenização de um dos principais problemas da abordagem colaborativa tradicional. No capítulo 4 é apresentada a proposta desta pesquisa, bem como a arquitetura da solução proposta. Em seguida no capítulo 5 são descritos os experimentos realizados e é apresentada uma breve discussão. Finalmente, no capítulo 6 são apresentadas as conclusões e sugestões para próximos passos da pesquisa.

# 2 FUNDAMENTOS

Este capítulo apresenta a fundamentação teórica necessária para a compreensão dos Sistemas de Recomendação, o que inclui os principais algoritmos existentes, suas dificuldades e as respectivas soluções existentes na literatura.

## 2.1 Conceitos Básicos

Conforme apresentado em Bobadilla et al. (2013), o processo de geração de uma recomendação baseia-se na combinação das seguintes considerações: o **tipo de dados disponível** para a análise de preferência dos usuários; o **algoritmo de filtragem** considerado; a **abordagem considerada**, que pode ser baseada ou não no uso direto dos dados; a **técnica de recomendação** utilizada, que pode ser um algoritmo baseado em vizinhança mais próxima, modelos nebulosos, decomposição de valores singulares, algoritmos inspirados em biologia, etc; o nível de **esparsidade dos dados**; o **desempenho do sistema** também referido como velocidade de processamento; e a **qualidade dos resultados**.

Para Vargas (2014), os Sistemas de Recomendação podem ser vistos como um caso particular de recuperação de informação personalizada, na qual não são utilizadas consultas explícitas e sim informações implícitas dos interesses dos usuários para o processo de recomendação. Na prática, um Sistema de Recomendação procura inferir a relevância de um conjunto de itens até então desconhecidos para um determinado usuário no sistema e gerar uma lista de recomendação composta por tais itens em ordem de relevância, de forma a garantir que esta recomendação corresponda às preferências particulares do usuário.

Para que isto seja possível, primeiramente é necessário explorar as informações de usuários e itens disponíveis no sistema, as quais na maioria das vezes consistem de atributos (*features*) de usuários e itens e interações entre ambos. Tais interações podem ser representadas por ações como *download*, compra ou até mesmo a rotulação de um item por parte de um usuário (BEEL et al., 2016). São exemplos de

atributos de usuários: a idade, o gênero, a localização e a profissão; e exemplos de atributos de itens: o título, a descrição e a categoria. Ressalta-se, no entanto, que tais informações dependem exclusivamente da base de dados considerada.

A relevância de um item para um determinado usuário no sistema pode ser obtida de forma **explícita**, quando os usuários avaliam os itens de forma direta através de um valor de *rating* (JAVARI; JALILI, 2014), (LIU et al., 2014), (SHAMBOUR; LU, 2015); ou de forma **implícita**, quando o sistema infere a relevância de um item para um determinado usuário através de análise comportamental, como por exemplo através da contabilização de cliques em elementos da tela, *downloads* ou busca por determinados conteúdos (CHOI et al., 2012),(SANCHEZ-MORENO et al., 2016),(LACERDA, ). Os valores explícitos de *rating* podem ser representados de forma numérica (um número de estrelas, como por exemplo de 1 a 5), de forma ordinal ("concordo fortemente", "concordo", "discordo", "discordo fortemente"), de forma binária (“gostou” ou “não gostou”), ou de forma unária; a qual indica simplesmente que o usuário teve algum tipo de interação com o item (RICCI et al., 2010).

Pesquisas recentes apresentadas na literatura consideram determinadas notações matemáticas com o objetivo de representar formalmente os Sistemas de Recomendação (PATRA et al., 2015), (LU et al., ), (KOTKOV; WANG; VEIJALAINEN, 2016). Com base em um consenso das informações presentes em tais pesquisas, o presente trabalho considera que um Sistema de Recomendação pode ser representado por um conjunto de usuários $U = \{u_1, u_2, ..., u_N\}$, um conjunto de itens $I = \{i_1, i_2, ..., i_M\}$ e um conjunto de interações entre ambos $E = \{e_1, e_2, ..., e_K\}$. O número de usuários, o número de itens e o número de interações entre ambos são dados, respectivamente, por $N$, $M$, $K$.

Uma vez que usuários tendem a somente avaliar uma baixa quantia de itens no sistema em proporção ao alto número de itens disponíveis, a matriz de relações entre usuários e itens é frequentemente esparsa (LEVINAS, 2014). Desta forma, mensurar o nível de esparsidade nas bases de dados utilizadas para pesquisas em Sistemas de Recomendação – e em bases de dados de sistemas reais – é também muito importante. Tal nível de esparsidade, segundo (SHAMBOUR; LU, 2012), (LU et al., 2013) e

(SHAMBOUR; LU, 2015) pode ser calculado por

$$sparsity = 1 - \frac{K}{NM}. \tag{2.1}$$

onde K, N e M são, respectivamente, o número de interações entre usuários e itens, o número de usuários e o número de itens.

Quanto maior seu valor, mais esparsa é considerada a base de dados, o que significa que se torna mais difícil de identificar a similaridade entre usuários devido à baixa quantia de interações relacionadas a itens em comum. (LEVINAS, 2014).

De acordo com pesquisas recentes em Sistemas de Recomendação, tais como Vargas (2014),Kaminskas e Bridge (2016), Beel et al. (2016) e He, Parra e Verbert (2016), outro conceito de notável importância é a "serendipidade" (*serendipity*, em inglês), a qual pode ser definida como a qualidade de itens que não teriam sido descobertos pelo usuário sem o auxílio de um Sistema de Recomendação (VARGAS, 2014); ou ainda como a capacidade de um Sistema de Recomendação em recomendar itens novos, interessantes e imprevisíveis ao usuário (KOTKOV; WANG; VEIJALAINEN, 2016).

Normalmente, a maior parte das pesquisas envolvendo tais sistemas busca avaliar a qualidade das recomendações geradas através de métricas que mensuram a acurácia dos algoritmos utilizados, ou seja, procuram medir o quanto o conjunto de itens recomendados ao usuário atende, de fato, suas expectativas. No entanto, para a resolução de alguns dos principais problemas conhecidos na área dos Sistemas de Recomendação, faz-se necessário explorar também a presença de novidade e imprevisibilidade nas recomendações; ou seja, a presença de serendipidade.

Por fim, a seleção dos itens mais adequados à preferência dos usuários (filtragem de informação) é de responsabilidade dos chamados algoritmos de filtragem, os quais são classificados na literatura nos seguintes tipos: algoritmos de filtragem baseados em conteúdo (FBC), algoritmos de filtragem baseados em conhecimento (FBCO),algoritmos de filtragem demográfica (FD), algoritmos de filtragem colaborativa (FC) e algoritmos híbridos (HB) (BOBADILLA et al., 2013), (ZUO et al., 2015), (LU et

al., 2015), (KATARYA; VERMA, 2016), (BEEL et al., 2016), (SRIDEVI; RAO, 2017).

As seções a seguir apresentam em mais detalhes cada um destes tipos de algoritmos.

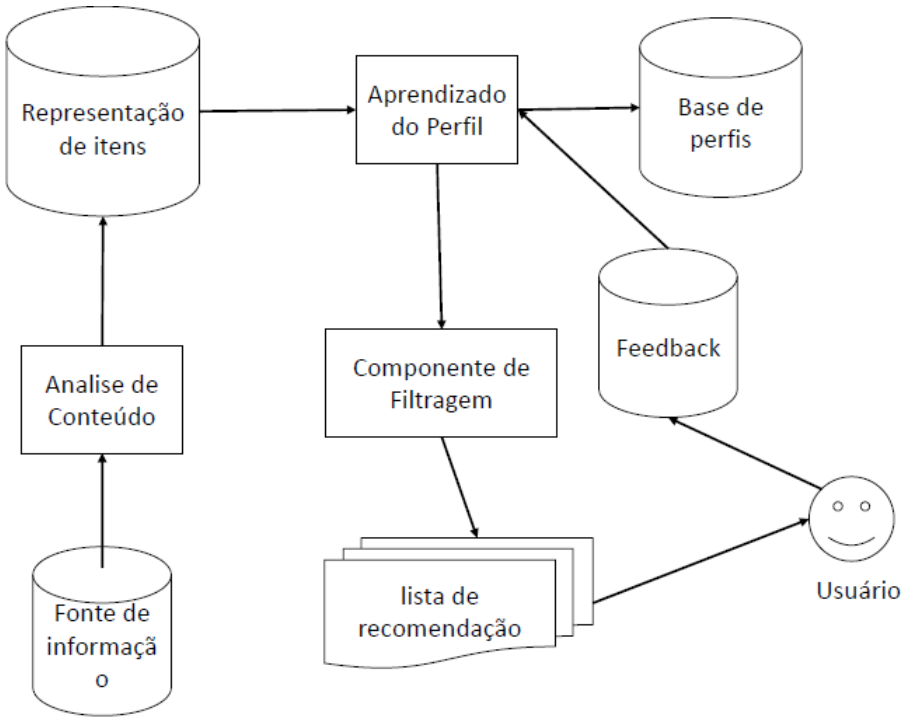
## 2.2 Algoritmos de Filtragem Baseada em Conteúdo

Algoritmos baseados em conteúdo (FBC) consideram os atributos de itens para a geração das recomendações, de forma que os itens recomendados ao usuário terão atributos semelhantes a itens com os quais ele interagiu anteriormente.(VARGAS, 2014), (KOTKOV; WANG; VEIJALAINEN, 2016).

De acordo com Ricci et al. (2010), este tipo de algoritmo busca analisar um conjunto de itens previamente conhecidos por um usuário para construir um modelo de representação de seus interesses, o qual consiste de uma representação estruturada dos interesses do usuário adaptada para recomendar itens que sejam interessantes ao mesmo. Desta forma, o processo de recomendação consiste em corresponder os atributos do perfil do usuário aos atributos de um determinado item e, a partir desta informação, obter uma conclusão sobre o nível de interesse do usuário por aquele determinado item. A figura 1 apresenta uma adaptação da arquitetura de alto nível apresentada por Ricci et al. (2010), na qual o processo de recomendação é executado em três etapas, cada qual gerenciada por um componente específico.

Uma vez que a fonte de informação pode não ser estruturada (ex. documentos em forma de texto), é necessário realizar um pré-processamento para extração de informações relavantes de forma estruturada. O componente de “Análise de Conteúdo” presente na ilustração é o responsável por tal extração e representação de informações dos itens. Na sequência, tais dados são utilizados como entrada para os componentes de “Aprendizado de Perfil” e “Filtragem”. O primeiro coleta dados representativos das preferências do usuário e procura generalizá-las para a construção de um perfil (são utilizadas técnicas de aprendizado de máquina para tal propósito); já o segundo procura corresponder o perfil do usuário gerado aos itens candidatos a recomendação, de forma a compor uma lista de recomendação com os itens de atributos

Figura 1: Arquitetura de alto nível simplificada de um recomendador FBC.




**Fonte:** Adaptação de (RICCI et al., 2010).

mais próximos aos itens previamente avaliados pelo usuário.

Após a recomendação ser feita ao usuário, suas reações são coletadas e armazenadas no repositório *Feedback*. Tais reações, juntamente com a informação dos itens recomendados, são utilizadas para aprimorar a geração do perfil do usuário através do componente de "Aprendizado de Perfil", armazenando-as na "Base de Perfis".

De acordo com Bobadilla et al. (2013), Lee et al. (2013) e Beel et al. (2016), os Sistemas de Recomendação baseados puramente em algoritmos FBC normalmente sofrem pela incapacidade de gerar recomendações compostas de itens que não sejam necessariamente similares a outros itens já conhecidos pelo usuário, o que é conhecido como "super especialização" (ou *overspecialization* em inglês). Na prática isto significa que o Sistema de Recomendação baseado neste tipo de algoritmo considerará como sendo igualmente relevantes itens que compartilham atributos em comum com o perfil construído do usuário. Uma das possíveis soluções para este problema se dá através do aumento da serendipidade.

## 2.3 Algoritmos de Filtragem Baseada em Conhecimento

Segundo Lu et al. (2013), os algoritmos de filtragem baseada em conhecimento (FBCO) são aqueles capazes de gerar recomendações com base em inferências sobre as necessidades e preferências do usuário; além disto, possui conhecimento funcional sobre como um item em particular corresponde a uma necessidade particular do usuário e, portanto, pode raciocinar sobre a relação entre uma necessidade e uma recomendação.

De acordo com Aggarwal (2016), diferentemente dos algoritmos filtragem colaborativa (FC, descritos a seguir) que dependem totalmente da informação de *ratings* atribuídos a itens por usuários e dos algoritmos FBC que apenas consideram informações sobre os itens, os algoritmos FBCO não necessitam reunir informações sobre um determinado usuário pois seu julgamento é independente de gostos individuais. Para estes autores, Sistemas de Recomendação baseados em algoritmos FBCO permitem que usuários interajam com o sistema, de forma que a cada interação, dependendo da escolha do usuário, o sistema realize alterações nas recomendações para que as necessidades do usuário sejam atendidas. Um exemplo apresentado em Aggarwal (2016) é uma sequência de interações de um usuário com uma interface de um Sistema de Recomendação de restaurante e, em função das escolhas do usuário (faixa de preço, nome e categoria de restaurante, por exemplo), o Sistema de Recomendação restringiu as opções de estabelecimentos recomendados.

## 2.4 Algoritmos de Filtragem Demográfica

Os algoritmos de filtragem demográfica (FD) realizam a recomendação de itens aos usuários levando em consideração as informações demográficas dos mesmos, sem considerar informações prévias dos itens e dos *ratings* atribuídos a eles. Por este motivo em particular este tipo de algoritmo consegue superar um conhecido problema chamado de partida a frio (*cold start* ) (SRIDEVI; RAO, 2017).

De acordo com os autores de (BOBADILLA et al., 2013) e (SRIDEVI; RAO, 2017), frequentemente este tipo de algoritmo usa técnicas para realizar o agrupamento de

usuários através de seus atributos demográficos e pode ser facilmente combinado com algoritmos de filtragem colaborativa ou filtragem por conteúdo.

Vale ressaltar que este tipo de algoritmo de filtragem foi pouco detalhado na literatura dos Sistemas de Recomendação frente à grande quantia de trabalhos relacionados aos demais tipos de algoritmos, sendo que na maioria das vezes os algoritmos FD são utilizados juntamente com as demais abordagens.

## 2.5 Algoritmos de Filtragem Colaborativa

Os algoritmos de filtragem colaborativa (FC), ou simplesmente algoritmos colaborativos, partem da intuição de que usuários que avaliaram itens de forma similar no passado tendem a ter preferências similares no futuro. Neste tipo de algoritmo a recomendação de itens a um determinado usuário é feita levando-se em conta o conjunto de usuários similares a ele. Os algoritmos FC são classificados ainda como sendo baseados em modelo (*model-based*) ou baseados em memória (*memory-based*) (RICCI et al., 2010),(MORADI; AHMADIAN, 2015), (YANG et al., 2016).

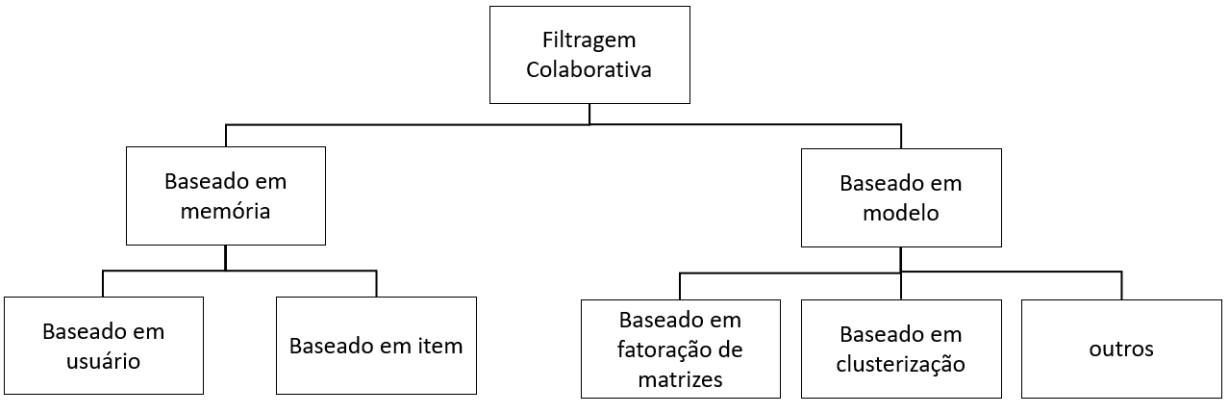
Os algoritmos FC baseados em memória são aqueles que fazem uso direto da informação dos *ratings* atribuídos a itens no sistema para a geração das recomendações (BOBADILLA et al., 2013), o que significa que é necessário que toda informação de *ratings* esteja disponível em memória para a predição do valor de relevância de itens para um determinado usuário do sistema. Como, na prática, tal predição é feita considerando a vizinhança de usuários similares ao usuário alvo da recomendação, este algoritmo também é conhecido por algoritmo de FC baseado em vizinhança mais próxima; ou *nearest neighborhood algorithms*, em inglês. (SANCHEZ-MORENO et al., 2016), (CHRISTAKOPOULOU; KARYPIS, 2016).

Por outro lado, os algoritmos baseados em modelo utilizam as informações de *ratings* para a construção de um modelo preditivo, que é então utilizado para a geração das recomendações (LEVINAS, 2014),(ZHANG; MIN, 2016). A ideia principal, neste caso, é a utilização de técnicas de aprendizado de máquina ou mineração de dados para a criação dos modelos de forma *offline*. Como exemplos deste tipo de algoritmo

há aqueles baseados em fatoração de matrizes e os baseados em técnicas de agrupamento (YANG et al., 2016).

A figura 2 ilustra esquematicamente a classificação de algoritmos FC.

Figura 2: Classificação de Algoritmos de Filtragem Colaborativa (FC).




**Fonte:** adaptação de (YANG et al., 2016).

Segundo a ilustração, os algoritmos FC baseados em memória podem ser classificados como sendo baseados em usuário ou baseados em item. Os algoritmos FC baseados em usuário são aqueles que fazem a recomendação de um determinado item $i$ a um usuário $u$, se outros usuários com preferências similares a $u$ tiveram afinidade por $i$. Por outro lado, algoritmos FC baseados em item são aqueles que fazem a recomendação de um item $i$ a um usuário $u$, se $i$ é similar a outros itens que o usuário $u$ teve afinidade anteriormente (BAO et al., 2015), (LACERDA, ). É importante frisar que, diferentemente dos algoritmos FBC, nesta abordagem as similaridades estão relacionadas a *ratings* e não a meta-dados de itens e/ou usuários. Da mesma forma, os algoritmos FC baseados em modelo podem ser classificados como sendo baseados em fatoração de matrizes, baseados em agrupamento (*clusterização*), dentre outros.

## 2.6 Algoritmos Híbridos

Na visão de Lu et al. (2013), algoritmos híbridos (HB) são aquelas que buscam combinar os pontos fortes das demais abordagens para alcançar melhorias nos resultados obtidos pelos Sistemas de Recomendação. Ainda de acordo com os autores, durante a construção de Sistemas de Recomendação híbridos podem ser considera-

dos até sete mecanismos de hibritização, os quais são descritos brevemente a seguir:

- **Mistura**: neste método as listas de recomendação são compostas pela combinação de itens recomendados em diferentes abordagens;
- **Ponderação**: é calculada uma pontuação para um determinado item recomendado de acordo com as pontuações obtidas por diferentes abordagens para este mesmo item; utiliza-se por exemplo combinação linear para tal ponderação;
- **Alternado**: através de um critério específico uma das abordagens de recomendação é selecionada para fazer a predição dos itens, quando o critério em questão é atendido;
- **Cascata**: trata-se de um processo em fases, onde uma abordagem de recomendação aprimora os resultados obtidos previamente por outra;
- **Combinação de atributos**: atributos de diferentes abordagens são integrados em um único atributo;
- **Aumento de atributos**: uma abordagem de recomendação é considerada como um atributo de entrada em outra abordagem;
- **Meta-Nível**: o modelo completo aprendido em uma abordagem de recomendação é utilizado como entrada para outra abordagem.

Pesquisas têm mostrado uma tendência pela adoção de algoritmos híbridos como solução para amenização de alguns problemas conhecidos. Lu et al. () explorou a combinação entre algoritmos FBC e FC para a recomendação de notícias, enquanto que Horsburgh, Craw e Massie (2015) integrou o aprendizado de pseudo-*tags* à representação de músicas em formato de *tags*, de forma a amenizar o problema de esparsidade presente nestas representações no contexto de Sistemas de Recomendação de músicas. Shambour e Lu (2015), por sua vez, utilizaram informações implícitas de usuários e itens para expandir as vizinhanças e amenizar os problemas de esparsidade de dados e partida a frio. Por outro lado, Zeng et al. (2013), Zhang e Zeng (2015), Chen, Zeng e Chen (2015) combinaram algoritmos baseados em processos de difusão de massa para a resolução de tais problemas.

## 2.7 Análise comparativa entre os principais tipos de algoritmos de filtragem

De forma geral, os algoritmos de filtragem são classificados de acordo com alguns critérios na literatura. Baseado nas pesquisas realizadas por Bobadilla et al. (2013) e Yang et al. (2016), a tabela1 ilustra tais critérios e a quais tipos de algoritmos de filtragem os mesmos são aplicáveis.

Tabela 1: Classificação dos algoritmos de filtragem de acordo com alguns critérios

| **Critério** | **Algoritmos de Filtragem** |
|---|---|
| Baseado em memória | FC |
| Baseado em modelo | FBC, FD, FC, HB |
| Utilizado com bases de dados comerciais (não públicas) | FBC e HB |
| Utilizado com bases de dados públicas | FC |
| Considera *ratings* nas recomendações | FC, FD e HB |
| Utiliza medidas de similaridade | FC e HB |
| Utiliza técnicas de fatoração de matrizes | FBC,FD, FC e HB |
| Considera técnicas como *Recall* e *Precision* | FC e HB |
| Utiliza técnica KNN (*k —Nearest Neighbors*) | FC |

**Fonte:** adaptação de Bobadilla et al. (2013) e Yang et al. (2016).

De forma geral, os algoritmos de FC são os mais amplamente utilizados em bases de dados públicas o que facilita para que pesquisadores possam reproduzir os trabalhos com mais facilitade, além disso são classificados dentre a maior parte dos critérios levantados pelos autores, o que mostra sua grande flexibilidade.

## 2.8 Algoritmos colaborativos baseados em vizinhança

Esta pesquisa inicialmente foca seus estudos nos algoritmos FC baseados em vizinhança, em especial a abordagem baseada em usuário. De acordo com Yang et al. (2016), o processo de geração de recomendação de itens através deste tipo de

algoritmo pode ser dividido em cinco principais etapas: cálculo de similaridade entre usuários; seleção da vizinhança; predição dos valores de *ratings* de itens desconhecidos de acordo com a vizinhança; ranqueamento dos itens de acordo com os *ratings* obtidos na predição; e seleção dos itens mais relevantes ao usuário.

Cada uma destas etapas é descrita a seguir.

### 2.8.1 Medidas de similaridade

A computação de similaridade entre pares de usuários e entre pares de itens desempenha um papel fundamental no processo de recomendação feito por algoritmos FC baseados em vizinhança. A duas principais medidas de similaridade consideradas na literatura dos Sistemas de Recomendação são o coeficiente de correlação de Pearson ($CCP$) (PEARSON, 1920) e a semelhança de cossenos ($SC$) (SARWAR et al., 2001).

Desta forma, de acordo com Ricci et al. (2010), Zhang et al. (2013) , Levinas (2014) e Yang et al. (2016), o cálculo de similaridade entre dois usuários $u$ e $v$ através do $CCP$, no contexto dos Sistemas de Recomendação, é dado por:

$$CCP(u,v) = \frac{\sum_{i \in I_{uv}} (r_{u,i} - \bar{r}_u)(r_{v,i} - \bar{r}_v)}{\sqrt{\sum_{i \in I_{uv}} (r_{u,i} - \bar{r}_u)^2 \sum_{i \in I_{uv}} (r_{v,i} - \bar{r}_v)^2}}, \tag{2.2}$$

onde $I_{uv}$ representa o conjunto de itens ranqueados por ambos usuários $u$ e $v$; $r_{u,i}$ e $r_{v,i}$ são os *ratings* atribuídos pelos usuários $u$ e $v$ ao mesmo item $i$; $\bar{r}_u$ e $\bar{r}_v$ são os valores médios de *ratings* atribuídos pelos usuários $u$ e $v$, respectivamente, a todos itens com os quais eles interagiram no sistema; e $CCP(u,v) \in \mathbb{R}$, com $-1 \leq CCP(u,v) \leq 1$.

Já o cálculo de similaridade entre dois itens $a$ e $b$ através de $SC$, no contexto dos Sistemas de Recomendação, é dado por:

$$SC(a,b) = \frac{\sum_{u \in U} (r_{u,a} - \bar{r}_u)(r_{u,b} - \bar{r}_u)}{\sqrt{\sum_{u \in U} (r_{u,a} - \bar{r}_u)^2 \sum_{u \in U} (r_{u,b} - \bar{r}_u)^2}}, \tag{2.3}$$

onde $U$ representa o conjunto de usuários que ranquearam ambos itens $a$ e $b$; $r_{u,a}$ e $r_{u,b}$ são os *ratings* atribuídos pelo usuário $u$ aos itens $a$ e $b$, respectivamente; $\bar{r}_u$ é a média do valor dos *ratings* atribuídos pelo usuário $u$ em todos itens com os quais ele

teve alguma interação; e $SC(a,b) \in \mathbb{R}$, com $-1 \leq SC(a,b) \leq 1$.

É importante ressaltar que o $CCP$ normalmente é utilizado quando se deseja obter a similaridade entre usuários; enquanto que a $SC$ é utilizada quando se deseja obter a similaridade entre itens.

### 2.8.2 Identificação de vizinhança

Após o cálculo de similaridade entre pares de usuários, aqueles mais similares ao usuário alvo da recomendação (usuários com maior grau de correlação) são considerados como sua vizinhança.

Normalmente, define-se um tamanho $Vz$ para a vizinhança de usuários, de forma que algumas pesquisas constataram experimentalmente que valores baixos de $Vz$ – normalmente entre 20 e 40 – são considerados aqueles que maximizam os resultados das recomendações, de acordo com algumas métricas (LU et al., 2013), (KAMINSKAS; BRIDGE, 2016).

### 2.8.3 Predição de *ratings*

Após a identificação da vizinhança de usuários similares, é necessário predizer o valor de *rating* para cada item desconhecido ao usuário alvo da recomendação. De acordo com a literatura (JANNACH et al., 2010), (ZHANG et al., 2013), (PATRA et al., 2015), (YANG et al., 2016), a versão mais simples e tradicional do algoritmo FC baseado em vizinhança faz a predição do valor de *rating* de um item $i$ para um usuário alvo $u$ através de:

$$pred(u,i) = \bar{r_u} + \frac{\sum_{v \in Vz} sim(u,v) * (r_{v,i} - \bar{r_v})}{\sum_{v \in Vz} sim(u,v)}, \tag{2.4}$$

onde $\bar{r_u}$ é a média de *ratings* atribuídos pelo usuário $u$ a todos itens com os quais ele interagiu no sistema; $Vz$ é o conjunto de usuários similares a $u$ (vizinhaça); $sim(u,v)$ é a similaridade calculada entre o usuário $u$ e cada usuário $v$; $r_{v,i}$ é o *rating* atribuído pelo usuário $v$ ao item $i$ e $\bar{r_v}$ é a média de *ratings* atribuído aos itens com os quais o usuário $v$ teve interação. Note que $sim(u,v)$ representa o valor calculado por uma medida de similaridade ($CCP$ ou $SC$).

### 2.8.4 Ranqueamento e seleção de itens

Após a predição de *ratings* a todo o conjunto de itens desconhecidos pelo usuário, os mesmos são ordenados de forma decrescente pelos valores de *rating* e recomendados ao usuário nesta ordem. A lista de itens que é enviada ao usuário é chamada lista de recomendação.

É nítido que para a computação da vizinhança de usuários ou itens existe uma forte dependência dos valores de *ratings* que são utilizados como base para os cálculos. Por conta disto, frequentemente os algoritmos FC baseados em vizinhança sofrem com as situações nas quais tal informação é escassa ou inexistente. Neste sentido, como principais problemas deste tipo de algoritmo podemos citar a partida a frio (*cold-start*) e a esparsidade dos dados (*data sparsity*) (BOBADILLA et al., 2013), (LU et al., 2015),(LU et al., ), (BEEL et al., 2016).

As subseções a seguir apresentam uma revisão literária de ambos os problemas, especialmente aquele relacionado à esparsidade dos dados, para o qual são apresentadas algumas soluções exploradas na literatura que visaram amenizá-lo.

### 2.8.5 O problema da partida a frio

De acordo com Bobadilla et al. (2013) e Levinas (2014), a partida a frio ocorre quando não é possível gerar recomendações devido a ausência de *ratings*. Sanchez-Moreno et al. (2016) ainda menciona que este problema afeta diretamente os novos usuários do Sistema de Recomendação, pois estes não podem receber recomendações devido a não terem ainda avaliado itens no sistema.

Segundo Nikolakopoulos, Kouneli e Garofalakis (2015) e Beel et al. (2016) existem três situações nas quais pode ocorrer o problema de partida a frio: (i) quando novos usuários ranqueiam poucos ou nenhum item e o sistema não consegue identificar usuários similares; (ii) quando um item é novo no sistema e não foi ranqueado ainda por ao menos um usuário e não pode ser recomendado; e (iii) em uma nova comunidade onde nenhum usuário ranqueou nenhum item e como consequência nenhuma recomendação pode ser gerada, além de ser baixo o incentivo para que usuários ran-

queiem itens.

Soluções exploradas na literatura para este problema vão desde sugestões para forçar novos usuários a ranquearem itens antes mesmo de poderem ter acesso ao Sistema de Recomendação (LEVINAS, 2014), até técnicas um pouco mais sofisticadas para inferir os valores de *ratings* de forma implícita em interações entre usuários e itens (BEEL et al., 2016). Tal estratégia foi considerada na solução proposta em (SANCHEZ-MORENO et al., 2016), na qual foi proposto um algoritmo capaz de obter as preferências dos usuários a partir de informações implícitas presentes na base de dados `last.fm`. A informação implícita considerada foi o número de vezes que usuários ouviam determinados artistas (frequência), informação esta que serviu para identificação de usuários com preferências mais incomuns e também artistas mais populares; tal algoritmo ainda é capaz de computar o valor de *rating* de usuários por artistas como uma função linear da frequência de reprodução dos artistas.

### 2.8.6 O problema da esparsidade dos dados

O problema da esparsidade dos dados é caracterizado pela situação na qual existe um elevado número de usuários e/ou itens no sistema, de forma que cada usuário ranqueia um conjunto pequeno de itens em proporção ao número total de itens disponíveis (ZHOU; ZENG; WANG, 2015). Em outras palavras, a matriz de *ratings* entre usuários e itens é muito esparsa, e como grande parte das métricas utilizadas para computar similaridade entre usuários necessita dos valores de *ratings*, existe uma dificuldade em identificar usuários similares (LEVINAS, 2014).

Diversas soluções foram propostas na literatura para a resolução deste problema, sendo as mais populares aquelas envolvendo *tagging social* (KIM; El Saddik, 2013), (HORSBURGH; CRAW; MASSIE, 2015),(ZHOU; ZENG; WANG, 2015), fatoração de matrizes (KOREN; BELL; VOLINSKY, 2009), (YANG et al., 2016) e algoritms baseados em grafos (LEE et al., 2013), (LEE; LEE, 2015). Particularmente, no caso das soluções através de algoritmos baseados em grafos, destacam-se pesquisas que exploram alguns processos físicos de difusão de massa em grafos bipartidos (ZHOU et al., 2010),(ZENG et al., 2013),(ZENG et al., 2014),(ZHANG; ZENG, 2015); este tema

será abordado mais profundamente no capítulo 4 deste documento, dado que foi a base para a pesquisa aqui desenvolvida.

As soluções via *tagging* social, conforme explicam Horsburgh, Craw e Massie (2015), permitem que usuários associem rótulos (*tags*) em forma de texto a itens no sistema. Segundo os autores, tais rótulos trazem benefícios da sabedoria social e do contexto em questão para os Sistemas de Recomendação. De acordo com Bellogín, Cantador e Castells (2013), o conjunto completo de rótulos é chamado de folksonomia (ou *folksonomy*, em inglês).

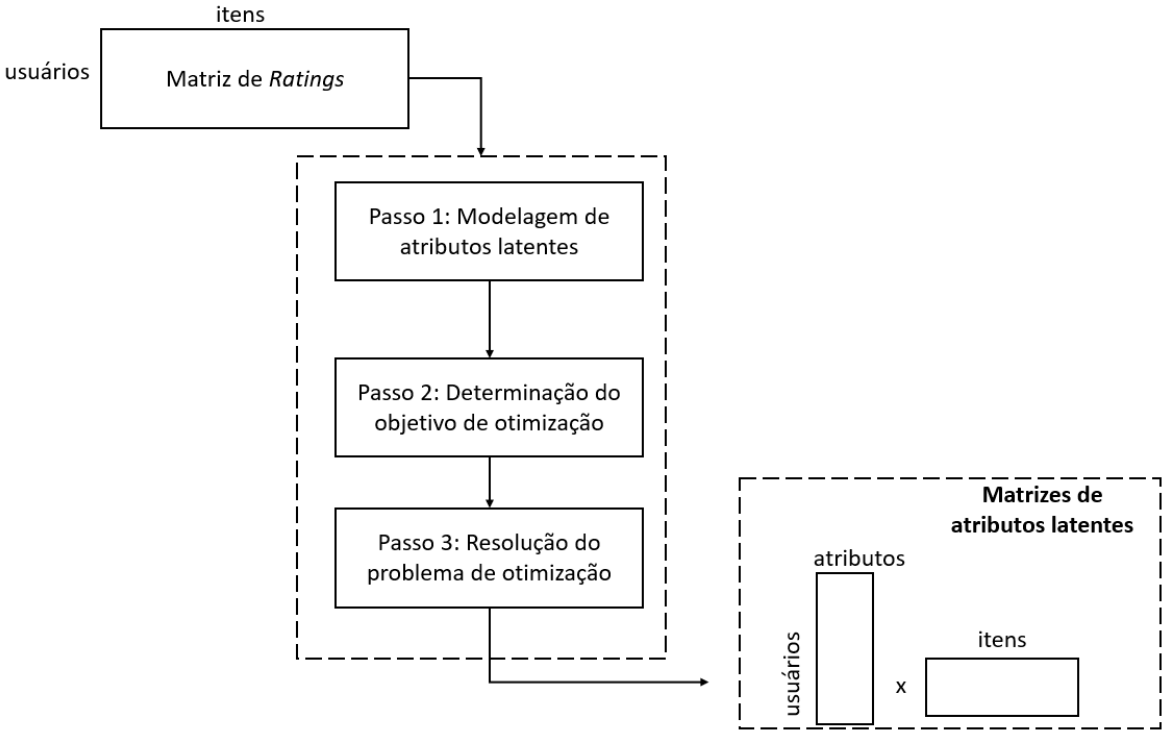
Zhou, Zeng e Wang (2015) apresentaram um método de integração de rótulos em algoritmos de FC baseadas em usuário. Tal método permitiu representar o perfil de um determinado usuário e o perfil de um determinado item no sistema como um vetor composto de pesos atribuídos a cada rótulo associado a ele no sistema. Este peso pode ser definido de acordo com diferentes esquemas, porém os autores utilizaram um esquema baseado no algoritmo TF-IDF (*frequency-inverse document frequency*). Na prática os pesos dos rótulos são calculados pela multiplicação de dois termos: o número de vezes que um determinado usuário tem anotado itens com um rótulo em específico e os fatores de frequência inversa que desfavorecem rótulos que aparecem frequentemente.

A grande diferença desta abordagem em relação às abordagens tradicionais FC está no processo de predição, uma vez que são considerados os vetores de rótulos de usuários e de itens para o cálculo de similaridade e a definição de preferência do usuário alvo da recomendação por um item, e por fim a lista de recomendação é composta pelos itens com os mais significativos valores de preferência. Se os itens obtidos com tal estratégia não forem suficientes para completar a lista em todas as suas posições, a lista é completada com itens selecionados pelo método tradicional de predição baseado apenas em *ratings*. (ZHOU; ZENG; WANG, 2015). Desta forma, devido à alta esparsidade dos rótulos em base de dados, a abordagem através de *tags* vem como uma alternativa para amenização do problema pois faz com que o cálculo do valor de relevância de itens seja feito sem considerar esta informação, ou utilizando-a com menor frequência.

Para Yang et al. (2016) a causa do problema da esparsidade dos dados é atribuída à alta dimensão dos vetores de usuários e de itens; desta forma, os algoritmos de fatoração de matrizes procuram, através de redução de dimensionalidade, aliviar tal esparsidade. De acordo com os autores, geralmente a matriz de *ratings* entre usuários e itens no sistema contém alguns atributos (*features*) latentes os quais podem ser utilizados para descrever o perfil dos usuários e itens. Como exemplo, os autores indicam que no caso de vídeos os atributos latentes poderiam ser o seu estilo – comédia, tragédia, drama, etc.

Ainda de acordo com Yang et al. (2016), os principais passos executados neste tipo de algoritmo são a modelagem de atributos latentes, a determinação do objetivo de otimização e a resolução do problema de otimização, ambos ilustrados na figura 3.

Figura 3: Principais passos de algoritmos FC baseados em fatoração de matrizes.




**Fonte:** adaptação de (YANG et al., 2016).

O primeiro passo (modelagem de atributos latentes) trata de associar os usuários e itens a vetores, onde o vetor de atributos de usuário, $\mathbf{p_u}$, indica o quanto o usuário $u$ possui interesse por cada atributo; enquanto que o vetor de atributos de item, $\mathbf{q_i}$, mede o nível de cada atributo para o item $i$. Com base nestes dois vetores, o valor de *rating*

do usuário $u$ para o item $i$ pode ser calculado através da seguinte equação:

$$r_{u,i} = \mathbf{p_u}^T \mathbf{q_i}. \tag{2.5}$$

Para Yang et al. (2016), após o estabelecimento das matrizes de atributos de usuários e itens, a matriz de *ratings* desconhecidos pode ser obtida facilmente através do produto dos vetores de atributos demonstrado na equação 2.5. No entanto, métodos tradicionais de fatoração de matrizes, tal como o SVD (*Single Value Decomposition*) e PCA (*Principal Component Analysis*), falham na decomposição devido ao alto número de valores ausentes.

Por este motivo é necessário realizar o segundo passo descrito pelos autores, a determinação do objetivo de otimização, que busca garantir que os vetores ($\mathbf{p_u}$ e $\mathbf{q_i}$) atendam à premissa de que seu produto seja próximo de $r_{u,i}$, o que pode ser modelado como um problema de otimização, cujo objetivo é definido por:

$$\min \mathbf{q}*, \mathbf{p}* \sum_{(u,i)\in K} (r_{u,i} - \mathbf{p_u}^T \mathbf{q_i})^2 + \gamma(\| \mathbf{q_i} \|^2 + \| \mathbf{p_u} \|^2), \tag{2.6}$$

onde $K$ é definido como o conjunto de pares de usuários e itens, $r_{u,i}$ é um *rating* conhecido que pode ser obtido a partir do conjunto de treinamento da base de dados utilizada, $\| \mathbf{q_i} \|$ e $\| \mathbf{p_u} \|$ são as magnitudes de $\mathbf{q_i}$ e $\mathbf{p_u}$; e $\gamma$ é introduzido para mitigar o problema de hiperespecialização (*overfitting*). Esta equação busca minimizar explicitamente o erro quadrático médio da predição dos *ratings* – repare que ela é minimizada quando a diferença quadrática entre o verdadeiro *rating*, $r_{u,i}$, e o *rating* estimado dado por $\mathbf{p_u}^T \mathbf{q_i}$ é mínima.

No entanto, eventualmente, esta equação pode sofrer com viés de usuários e itens; por exemplo, quando os *ratings* de um determinado usuário forem muito menores que de outros. Neste caso, portanto, algumas variáveis adicionais são necessárias para considerar tal viés (YANG et al., 2016).

Como último passo, tem-se a resolução do problema de otimização, que é feita geralmente por algoritmos bem conhecidos, tais como o gradiente descendente estocás-

tico (*stochastic gradient descent*) e o algoritmo de alternação de mínimos quadrados (*alternating least squares*).

Com relação às soluções via algoritmos baseados em grafos, uma pesquisa recente apresentada em Beel et al. (2016) revisou 200 artigos publicados sobre Sistemas de Recomendação nos últimos 16 anos e extraiu algumas estatísticas interessantes: a pesquisa mostrou que apesar da maioria dos artigos ser relacionada a algoritmos FBC e FC (abordagens mais tradicionais), com respectivamente 55% e 18% do total de artigos, aqueles baseados em grafos aparecem em 16% dos trabalhos. Já há alguns anos existem pesquisas que consideram a representação de Sistemas de Recomendação através de grafos, visando amenizar alguns dos principais problemas presentes em abordagens tradicionais, tal como o problema da esparsidade dos dados.

Uma abordagem muito comum é a representação de um Sistema de Recomendação como um grafo bipartido composto de conjuntos de usuários e itens, e a adoção de algumas técnicas para permitir a propagação de recursos entre os vértices do grafo, de forma a explorar os relacionamentos entre usuários e itens que não estão diretamente conectados (RICCI et al., 2010).

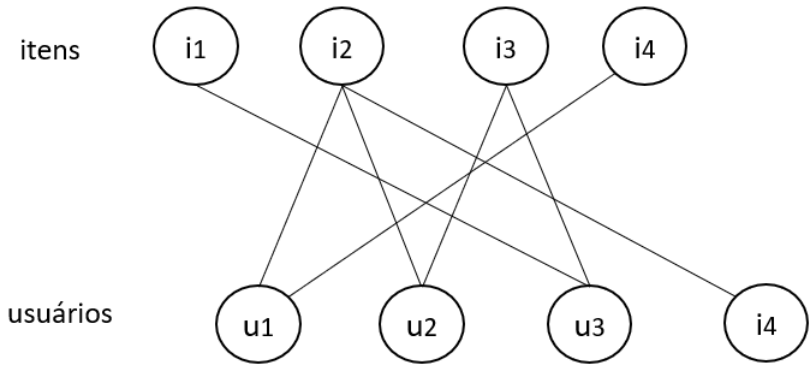
Os algoritmos de recomendação que seguem tal abordagem são conhecidos na literatura também como algoritmos de recomendação baseados em redes, de acordo com Yu et al. (2015), Chen, Zeng e Chen (2015), Ma et al. (2017); ou algoritmos de recomendação baseados em difusão, de acordo com Bobadilla et al. (2013), Lu et al. (2015), Zhang e Zeng (2015), os quais serão o foco do próximo capítulo deste documento.

De acordo com Huang, Chen e Zeng (2004), uma das pesquisas precursoras sobre algoritmos de recomendação baseados em grafos, a origem deste tipo de algoritmo vem da técnica de recuperação associativa (ou *associative retrieval technique*). A ideia básica é a construção de um grafo a partir de documentos, termos e *queries*, e a exploração das associações transitivas entre termos e documentos para aprimorar a qualidade da recuperação de informação.

Segundo os autores, no contexto dos Sistemas de Recomendação, as preferências

dos usuários podem ser representadas pelas interações (arestas) e os usuários e itens por vértices do grafo. A ideia é que o sistema seja representado como um grafo bipartido de usuários e itens, conforme ilustra a figura 4.

Figura 4: Exemplo de um grafo bipartido de usuários e itens em FC.




**Fonte:** adaptação de (HUANG; CHEN; ZENG, 2004).

Com base na figura 4, se um Sistema de Recomendação pretende fazer recomendações ao usuário representado pelo vértice $u_1$, e o algoritmo selecionado for o FC tradicional, o processo tomaria como base o cálculo de similaridade entre este usuário e os demais ($u_2$, $u_3$ e $u_4$), de forma que $i_3$ provavelmente seria recomendado a $u_1$. Isto ocorre porque $u_2$ teve interação com um item em comum ao usuário $u_1$ ($i_2$) e $i_3$ não havia ainda tido interações com $u_1$. Como os algoritmos de FC tradicionais não identificam similaridade entre $u_1$ e $u_3$ pelo fato de não terem interagido com itens em comum, $i_1$ que teve interação com $u_3$ não seria recomendado ao usuário $u_1$. Isto ocorre pois os algoritmos FC consideram apenas caminhos de tamanho 3 no grafo, ou seja, 3 arestas. No exemplo indicado acima, o item $i_3$ está distante de $u_1$ exatamente por 3 arestas.

Desta forma, os algoritmos baseados em grafos permitem explorar associações transitivas considerando caminhos maiores que 3 no grafo. Partindo deste raciocínio, algumas pesquisas consideraram diferentes formas de realizar as associações transitivas durante o processo de recomendação com o objetivo de amenizar o problema da esparsidade dos dados. (HUANG; CHEN; ZENG, 2004),(CHEN et al., 2011).

Segundo Huang, Chen e Zeng (2004) e Ricci et al. (2010) a computação de tais

associações transitivas em um Sistema de Recomendação com grande número de usuários e itens pode ser bastante custosa, de forma que para superar tais limitações a técnica conhecida por **ativação de espalhamento** (*spreading activation*, em inglês) é frequentemente utilizada. A ideia geral desta técnica é ativar primeiramente apenas um subconjunto de vértices do grafo e iterativamente ativar os vértices que podem ser alcançados diretamente a partir destes, até que um critério de convergência seja atingido. No capítulo 3 serão apresentados os algoritmos baseados em difusão, os quais utilizam variações desta técnica para o tratamento do problema dos dados esparsos.

## 2.9 Abordagens baseadas em Aprendizado Profundo (*Deep Learning*)

De acordo com a pesquisa apresentada em (ZHANG; YAO; SUN, 2017), abordagens baseadas em aprendizado profundo também têm obtido eficácia em tarefas de recomendação. Para os autores, o aprendizado profundo, no contexto dos Sistemas de Recomendação, é capaz de capturar relacionamentos não lineares e não triviais entre usuários e itens e permitir a codificação de abstrações mais complexas, como a representação de dados em altas camadas.

Ainda de acordo com os autores, os sistemas de recomendação baseados em aprendizado profundo podem ser classificados através de duas perspectivas: modelos de redes neurais e modelos de integração. No que diz respeito aos modelo de rede neural, os autores classificam os sistemas de recomendação com relação às técnicas de redes neurais empregadas, ou seja, os sistemas são classificados de acordo com a quantia de técnicas de aprendizado profundo utilizadas – apenas uma técnica ou multiplas técnicas (duas ou mais). Por outro lado, no tocante ao modelo de integração, os Sistemas de Recomendação são classificados quanto a existência ou não de integração com modelos tradicionais de recomendação, tais como modelos de fatoração de matrizes ou modelos baseados em vizinhança mais próxima.

Nota-se que, de acordo com a revisão literária apresentada em (ZHANG; YAO; SUN, 2017), a base de dados mais utilizada em experimentos com Sistemas de Recomendação baseados em técnicas de aprendizado profundo foi a base de dados *Movielens*, que é considerada densa (apresenta um número significativo de interações

entre usuários e itens, comparado ao número total dos mesmos). Esta, ao que tudo indica, pode ser uma evidência de que Sistemas de Recomendação baseados em aprendizado profundo ainda não têm sido validados massivamente em bases de dados altamente esparsas. Neste contexto, algoritmos baseados em técnicas de difusão de massa mostram-se como uma alternativa interessante.

Segundo os autores de (FU et al., 2018), uma boa acurária nas predições por si só não garante uma experiência satisfatória para os usuários das recomendações pois, apenas com base nos resultados das recomendações, os usuários não são capazes de confiar totalmente no Sistema de Recomendação, uma vez que os métodos colaborativos baseados em modelo são considerados quase uma "caixa-preta" com relação às predições geradas. Por outro lado, os métodos colaborativos baseados em vizinhança apresentam como vantagem uma melhor interpretabilidade, uma vez que explicitam os conjuntos de usuários (ou itens) similares e seus respectivos *ratings*, que explicam melhor aos usuários como as predições foram geradas. Estes autores apresentaram um novo método de filtragem colaborativa baseado em atenção e orientado a item, o qual é capaz de, adaptativamente, capturar os relacionamentos entre um item alvo e os itens presentes no histórico do usuário e gerar as predições de *ratings* a partir destes relacionamentos, sem a necessidade de acesso a informações adicinais. Estes relacionamentos capturados são chamados de *scores* de atenção e são capazes de justificar as predições de *ratings* geradas ao usuário. De forma geral, o modelo baseado em atenção é considerado um método popular de aprendizado profundo, o qual é inspirado pelo mecanismo natural de atenção seletiva do ser humano.

De acordo com as pesquisas realizadas por (ZHANG; YAO; SUN, 2017) e (FU et al., 2018) entre as bases de dados mais utilizadas para experimentos envolvendo aprendizado profundo em sistemas de recomendação estão *Movielens* e *Netflix*, as quais não podem ser consideradas altamente esparsas dado o número de usuários, itens e interações entre ambos – o que é comprovado facilmente pela equação 2.1. Desta forma, a solução para o problema da esparsidade através dos algoritmos baseados em difusão mostra-se como uma alternativa bastante promissora.

## 2.10 Considerações finais

De acordo com Lee e Lee (2015), apesar de algoritmos FC tradicionais obterem bons resultados em termos de acurácia, estes ainda falham com relação à habilidade de recomendar itens que representem novidade aos usuários. Da mesma forma, Kaminskas e Bridge (2016) coloca que pesquisas tradicionais envolvendo Sistemas de Recomendação geralmente focam em verificar o quanto os *ratings* preditos são, de fato, próximos dos verdadeiros *ratings* atribuídos a itens pelos usuários. Os autores ainda reconhecem o grande impacto que outras métricas, tais como *personalization* e *novelty*, têm sobre a qualidade das recomendações.

Neste sentido, algumas pesquisas têm explorado os benefícios da aplicação de alguns processos físicos, tais como a difusão de massa, no contexto dos Sistemas de Recomendação (ZHOU et al., 2010; ZENG et al., 2013; ZHANG; ZENG, 2015; MA et al., 2017). A pesquisa apresentada em Zhou et al. (2010) mostra-se como uma das mais relevantes, isto porque o algoritmo proposto pelos autores serviu de base para diversas extensões capazes de solucionar problemas conhecidos nos algoritmos FC tradicionais. Em especial, a extenção apresentada em Zeng et al. (2013) permitiu resolver de forma efetiva o problema da esparsidade dos dados.

O trabalho descrito nesta dissertação enfatiza o estudo de algumas soluções baseadas em processos físicos de difusão de massa para o problema da esparsidade dos dados. Desta forma, no próximo capítulo será apresentado em detalhes o funcionamento deste tipo de algoritmo, enfatizando o algoritmo híbrido baseado em difusão proposto por Zhou et al. (2010), o qual serviu de base para outras soluções apresentadas na literatura que visaram amenizar este e outros problemas comuns aos Sistemas de Recomendação através de aprimoramentos sobre esta mesma técnica (ZENG et al., 2013; ZHANG; ZENG, 2015; MA et al., 2017).

# 3 ALGORITMOS DE RECOMENDAÇÃO BASEADOS EM DIFUSÃO

Este capítulo apresenta os detalhes do algoritmo híbrido proposto por Zhou et al. (2010), o qual baseia-se no processo físico de difusão de massa e fornece uma solução para o aparente dilema entre acurácia e diversidade em Sistemas de Recomendação. Este dilema tem como objetivo descobrir a combinação apropriada entre métodos baseados em precisão (fornecendo popularidade) e diversidade (fornecendo novidade).

Tal algoritmo, chamado de "hybrid" originalmente pelos autores, será referenciado como $MDHS$ a partir deste momento. Tal algoritmo combina os algoritmos, originalmente conhecidos por, "ProbS" (ZHOU et al., 2007) e "HeatS" (ZHANG; BLATTNER; YU, 2007) para a geração da lista de recomendações, sendo que estes algoritmos serão referenciados como *Mass Diffusion (MD)* e *Heat-Spreading (HS)*, respectivamente, nesta pesquisa.

$MDHS$ representa um sistema de recomendação como um grafo bipartido, definido formalmente como $G = \{U, I, E\}$, onde $U = \{u_1, u_2, ..., u_N\}$, $I = \{i_1, i_2, ..., i_M\}$ e $E = \{e_1, e_2, ..., e_K\}$ são o conjunto de usuários, o conjunto de itens e o conjunto de arestas do grafo, respectivamente.

A partir de um conjunto de dados de treinamento previamente conhecido, o grafo $G$ é construído, atribuindo cada usuário a um vértice $u_N \in U$, cada item a um vértice $i_M \in I$ e, caso o usuário $u_N$ tenha tido interação com algum item $i_M$, uma aresta $e_K \in E$ é inserida em G, tornando $u_N$ adjacente a $i_M$. Interações aqui podem significar uma compra do usuário ou a avaliação de um dado item, por exemplo.

Desta forma, para cada usuário alvo $u$ da recomendação e presente na base de dados, os três passos descritos a seguir são executados:

**Passo 1** Inicialmente, um valor de recurso é atribuído a cada item $i$ no grafo com relação ao usuário $u$ o que é representado por $r(i|u)$, de acordo com a seguinte regra: se existe uma aresta entre $i$ e $u$ então $r(i|u) = 1$ , caso

contrário, $r(i|u) = 0$.

**Passo 2** Na sequencia, ocorre um processo de propagação dos valores de recurso do lado item para o lado usuário no grafo, onde cada usuário $v \in U$ no vértice de destino recebe um novo valor de recurso recalculado $r'(v|u)$ a partir de todos valores de recurso dos itens adjacentes no grafo $G$.

**Passo 3** Na etapa final do processo, ocorre uma nova propagação dos recursos do lado usuário para o lado item no grafo, na qual cada item $i$ recebe um novo valor de recurso recalculado $r''(i|u)$ a partir de todos valores de recurso de usuários adjacentes computados no passo 2.

O cálculo dos novos valores de recurso descritos nos Passos 2 e 3, representados respectivamente por $r'(v|u)$ e $r''(v|u)$, dependem do algoritmo considerado, $MD$, $HS$ ou $MDHS$, além da quantidade de interações que os usuários e itens tiveram na base de dados do Sistema de Recomendação, o que é representado pelo grau dos vértices no grafo, dado por $w(u)$ e $w(i)$, sendo respectivamente o grau do usuário $u$ e o grau do item $i$.

Desta forma, quando é considerado apenas o algoritmo $MD$, os cálculos dos valores de recurso para os passos 2 e 3, são representados, respectivamente, por:

$$r'_{MD}(v|u) = \sum_{i \in I} \frac{r(i|u)}{w(i)}, v \in U, i \in I, \tag{3.1}$$

e

$$r''_{MD}(i|u) = \sum_{v \in U} \frac{r'_{MD}(v|u)}{w(v)}, v \in U, i \in I. \tag{3.2}$$

Por outro lado, quando é considerado apenas o algoritmo $HS$, os cálculos dos valores de recurso descritos nos passos 2 e 3 são representados, respectivamente, por:

$$r'_{HS}(v|u) = \frac{\sum_{i \in I} r(i|u)}{w(v)}, v \in U, i \in I, \tag{3.3}$$

e

$$r''_{HS}(i|u) = \frac{\sum_{v \in U} r'_{HS}(v|u)}{w(i)}, v \in U, i \in I. \tag{3.4}$$

O algoritmo $MDHS$ (vide Algoritmo 1), no entanto, ainda permite a combinação de $MD$ e $HS$, através de um parâmetro de sincronização $\lambda$, o qual é utilizado para mesclar os cálculos das equações 3.1 e 3.3 em $r'_{HB}(u,i)$ (Passo 02) e 3.2 e 3.4 em $r''_{HB}(u,i)$ (Passo 03):

$$r'_{HB}(v|u) = (\sum_{i \in I} \frac{r(i|u)}{w(i)^{\lambda}})/w(v)^{(1-\lambda)}, v \in U, i \in I, \tag{3.5}$$

e

$$r''_{HB}(i|u) = (\sum_{v \in U} \frac{r'_{HB}(v|u)}{w(v)^{\lambda}})/w(i)^{(1-\lambda)}, v \in U, i \in I. \tag{3.6}$$

**Algoritmo 1** – O algoritmo $MDHS$ com $\lambda = 0.5$
(ZHOU et al., 2010)

**Entrada:** user $u$, $G$, $|L|$.
1: **para** cada item $i \in G$ com o qual $u$ teve interação **faça**
2: Inicialize $r(i|u)$ (**Passo 01**)
3: **fim para**
4: **para** cada usuário $v \in G$ **faça**
5: Calcule $r'(v|u)$ com $\lambda = 0.5$ (**Passo 02**, equação 3.5)
6: **fim para**
7: **para** cada item $i \in G$ **faça**
8: Calcule $r''(i|u)$ com $\lambda = 0.5$ (**Passo 03**, equação 3.6)
9: **fim para**
10: list $\leftarrow$ Separe todos os itens que não tiveram interação com $u$ inicialmente e ordene-os pelos seus valores de recurso $r''$
11: $L \leftarrow$ Separe os itens das últimas $|L|$ posições de list e ordene-os de forma decrescente pelos seus valores de recurso $r''$
12: **retorna** $L$

O Algoritmo 1 recebe como parâmetros de entrada o usuário alvo da recomendação $u$, o grafo bipartido de usuários e itens $G$ e o tamanho das listas de recomendação $|L|$. Note que quando $\lambda = 0$ tem-se a aplicação original do algoritmo $HS$ e, quando $\lambda = 1$, tem-se a aplicação original do algoritmo $MD$; portanto qualquer valor entre 0 e

1 permite a combinação de ambos no algoritmo $MDHS$.

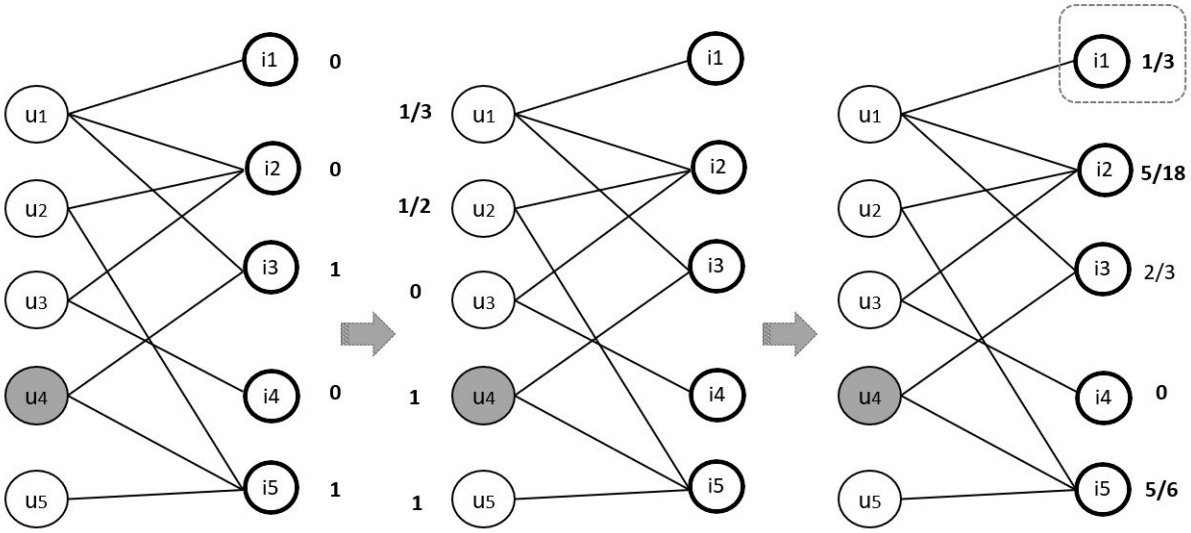
Em Zhou et al. (2010) os autores realizaram alguns experimentos com diferentes bases de dados e valores de $\lambda$ e selecionaram, de forma não automática, o valor de $\lambda$ mais sintonizado aos dados. Os autores consideraram o intervalo $0 \leq \lambda \leq 1$, onde $\lambda \in \Lambda$ e $\Lambda = \{0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0\}$ e sugeriram que um valor intermediário fixo ($\lambda = 0.5$) forneceria a mistura perfeita entre $MD$ e $HS$ em $MDHS$ e obteria melhores resultados.

Após a execução dos três passos descritos anteriormente, todos itens que não tinham inicialmente qualquer interação com o usuário alvo da recomendação, e que ao final do processo receberam um valor de recurso positivo ($r''(i|u) > 0$), são ordenados de forma crescente por $r''(i|u)$. A lista de predição é então formada por todos os itens que correspondem a estes critérios.

Os itens com maiores valores de recurso na lista de predição (últimas posições) compõem a lista de recomendação para o usuário alvo $u$. A lista de recomendação consiste em $|L|$ itens e é ordenada em ordem decrescente pelos valores de recurso dos itens; a primeira posição contém o item mais relevante e a última o item menos relevante.

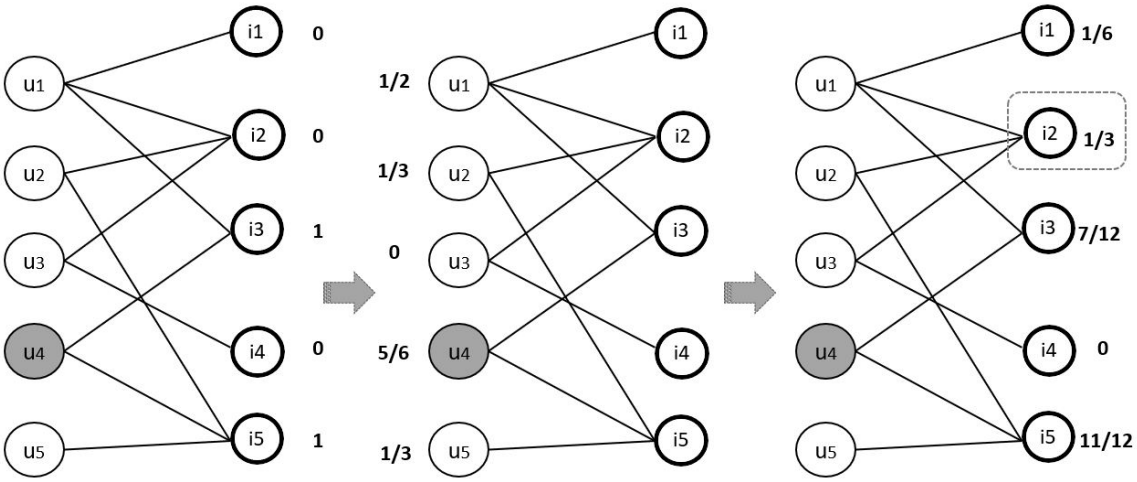
As figuras 5 e 6 ilustram a aplicação do processo de propagação (Passos 1, 2 e 3) no grafo considerando os algoritmos $HS$ e $MD$, respectivamente.

Figura 5: Os três passos de propagação com o algoritmo $HS$ em um grafo bipartido usuário-item.




**Fonte:** adaptação de Zhou et al. (2010).

Figura 6: Os três passos de propagação com o algoritmo $MD$ em um grafo bipartido usuário-item.




**Fonte:** adaptação de Zhou et al. (2010).

Em ambas figuras os usuários são representados por círculos com bordas normais, itens são representados por círculos com bordas em negrito, o usuário alvo da recomendação é indicado por um círculo hachurado e o item mais relevante após a recomendação é marcado com um retângulo pontilhado.

Da figura 5 temos que o usuário alvo da recomendação é $u4$, o qual previamente

interagiu com os itens $i3$ e $i5$. De acordo com o algoritmo HS, os valores de recurso são propagados para o lado usuário utilizando a equação 3.3; neste caso o valor resultante em $u1$ é $1/3$ (soma das propagações de $i1$, $i2$ e $i3$, dividido pelo grau de $u1$, que é 3); e o valor resultante em $u2$ é $1/2$ (soma das propagações de $i2$ e $i5$, dividido pelo grau de $u2$, que é 2). No último passo, a propagação de recursos ocorre do lado usuário para o lado item no grafo de acordo com a equação 3.4; então o novo valor de recurso no item $i3$, por exemplo, é $2/3$ (soma de $1/3$ propagado por $u1$ adicionado a $1$ propagado por $u4$, dividido pelo grau de $i3$, que é 2).

A lista de recomendação gerada ao final do processo sempre considera apenas os itens com os quais o usuário não havia tido interação inicialmente mas que tiveram algum valor de recurso ao final do processo. Portanto, neste caso a lista de recomendação é composta pelos itens $i1$ e $i2$; sendo $i1$ o item mais fortemente recomendado.

Da figura 6 temos que o usuário alvo da recomendação é $u4$, o qual previamente interagiu com os itens $i3$ e $i5$. De acordo com o algoritmo MD, os valores os valores de recurso são propagados para o lado usuário utilizando a equação 3.1; neste caso, $i3$ propaga $1/2$ para os usuários adjacentes no grafo e $i5$ propaga $1/3$, de forma que o valor calculado em $u4$ é $5/6$ No último passo, a propagação de recursos ocorre do lado usuário para o lado item no grafo de acordo com a equação 3.2; então o novo valor de recurso em $i3$, por exemplo, é $5/12$ (propagação de $u4$) adicionado a $1/6$ (propagação de $u1$), o que resulta em $7/12$.

Da mesma forma que no exemplo anterior, a lista de recomendação gerada ao final do processo sempre considera apenas os itens com os quais o usuário não havia tido interação inicialmente mas que tiveram algum valor de recurso ao final do processo. Portanto, neste caso a lista de recomendação é composta pelos itens $i1$ e $i2$; sendo $i2$ o item mais fortemente recomendado.

De acordo com as figuras 5 e 6, nota-se que $MD$ tende a gerar recomendações compostas de itens mais populares (vértices com maiores graus no grafo), enquanto que $HS$ tende a gerar recomendações compostas de itens menos conhecidos (vértices do grafo com menores graus).

Algumas extensões de $MDHS$ foram apresentadas na literatura, em especial

aquela conhecida como *Semi-Local Diffusion (SLD)* (ZENG et al., 2013), que permite que o processo de propagação seja repetido no grafo. Na prática, isto significa que os passos 2 e 3 ocorrem mais de uma vez, sendo possível alcançar vizinhanças ainda maiores de usuários e itens.

Entretanto, tanto $MDHS$ quanto *SLD*, sempre consideram o mesmo valor de $\lambda$ para todos usuários do Sistema de Recomendação, sem considerar nenhuma informação sobre seus perfis. Isto significa que para cada valor de $\lambda$ as listas de recomendação são geradas com a mesma proporção de $MD$ e $HS$ para todos usuários do conjunto de treinamento. Logo, utilizando um conjunto de treinamento, o valor de $\lambda$ que maximiza ou minimiza o resultado das métricas é selecionado experimentalmente, considerando a média ponderada dos resultados para todos usuários.

Acreditamos que utilizando o mesmo valor de $\lambda$ para todos usuários alvo do Sistema de Recomendação sem diferenciá-los através de seus perfis leve a piores resultados. Portanto, nesta dissertação nós propomos um arcabouço para sintonizar o parâmetro $\lambda$ de acordo com o perfil dos usuários e combiná-lo com o algoritmo $MDHS$ para a geração de recomendações personalizadas. Nossa proposta é descrita no próximo capítulo.

# 4 PROPOSTA

A proposta desta pesquisa consiste em um algoritmo que combina $MDHS$ com o parâmetro $\lambda$, especialmente aprendido para o usuário alvo da recomendação, de forma a gerar recomendações personalizadas. Este algoritmo foi nomeado como *User Profile Oriented Diffusion (*$UPOD$*)*, e será referenciado através desta sigla a partir deste momento.

A seção 4.1 descreve como foi feita a reprodução do algoritmo $MDHS$, utilizado como base para o algoritmo proposto $UPOD$. Em seguida, na seção 4.2, são apresentados todos os detalhes a respeito do algoritmo $UPOD$. Por fim,na seção 4.3, são explicados os detalhes de implementação.

## 4.1 Reprodução do algoritmo $MDHS$

Inicialmente foi feita a leitura minuciosa do conteúdo das pesquisas apresentadas em (ZHOU et al., 2007), (ZHANG; BLATTNER; YU, 2007), Zhou et al. (2010) e (ZENG et al., 2013) para entender o funcionamento do algoritmo $MDHS$.

No entanto, para que fosse possível a sua reprodução ainda eram necessários alguns detalhes adicionais que não estavam explícitos nos artigos. Por este motivo foi feito contato com An Zeng e Wei Zeng – autores do artigo (ZENG et al., 2013), os quais nos apoiaram com o esclarecimento de dúvidas e compartilhamento de algumas informações necessárias para a correta reprodução de $MDHS$.

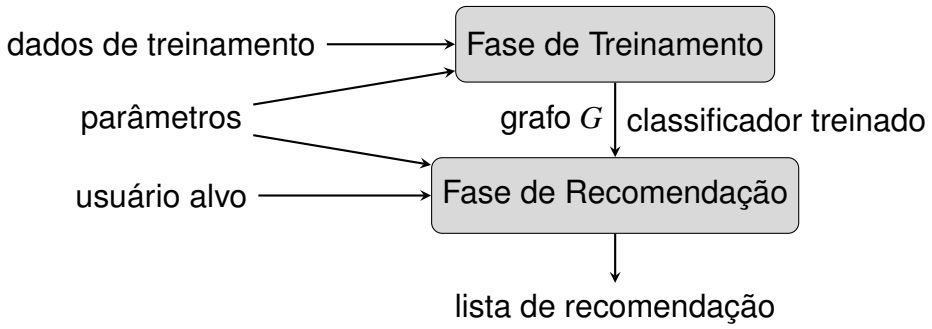
Por fim, para garantir que o algoritmo $MDHS$ havia sido reproduzido corretamente, foram feitos diversos experimentos com as mesmas bases de dados utilizadas pelos autores e comprovou-se que, de fato, os resultados dos artigos originais foram reproduzidos.

## 4.2 Algoritmo $UPOD$

O mecanismo de funcionamento básico do algoritmo consiste em agrupar usuários com perfis similares de acordo com seus atributos e definir, para cada tipo de perfil, o $\lambda$ mais adequado para a mistura dos algoritmos $MD$ e $HS$ internamente no algoritmo $MDHS$. Desta forma, cada usuário alvo terá o $\lambda$ mais apropriado de acordo com seu perfil, combinando novidade e popularidade nas recomendações, de forma personalizada.

$UPOD$ opera em duas fases: fase de treinamento e fase de recomendação, conforme ilustrado na figura 7.

Figura 7: As duas fases de UPOD.

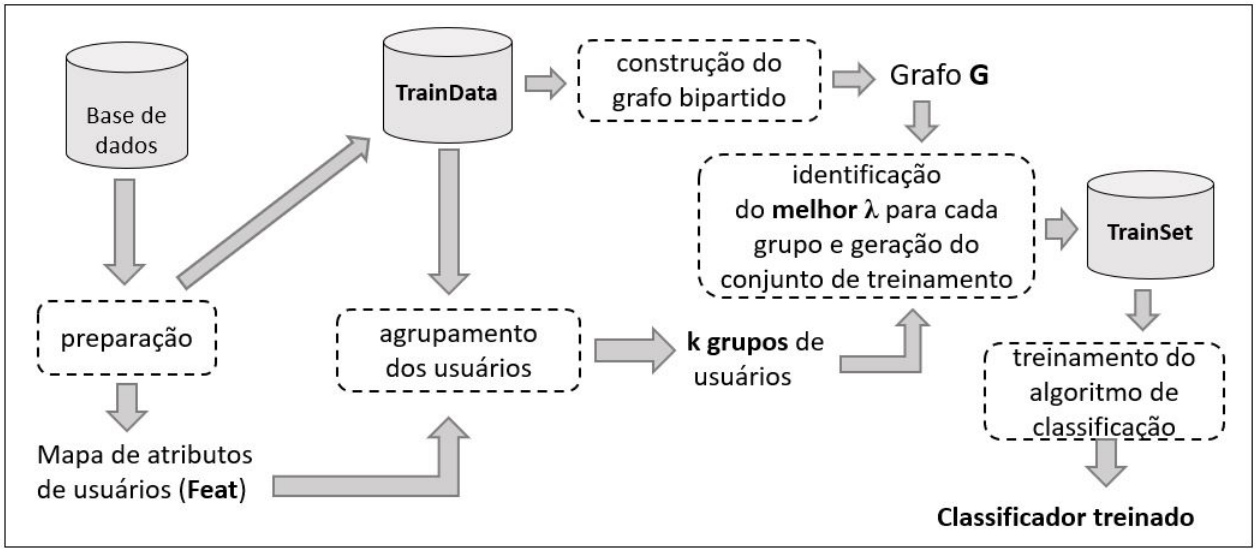



Resumidamente, as fases de treinamento e recomendação consistem em:

- Fase de treinamento
  - Identificar k grupos de usuários similares através dos atributos
  - Descobrir o “melhor” $\lambda$ para cada grupo
  - Associar os atributos ao “melhor” $\lambda$
  - Aprender $\lambda$ a partir dos atributos
- Fase de recomendação
  - Extrair atributos de usuário (conjunto de validação)
  - Gerar recomendação personalizada

Com relação à fase de treinamento é importante ressaltar que existem ainda alguns passos intermediários necessários para sua execução, tais como a preparação da base de dados (remoção de usuários com atributos inválidos, discretização de atributos não categóricos, etc), a extração dos atributos dos usuários e a geração do grafo bipartido necessário para os processos de difusão, por exemplo. De forma geral, todos os passos que compõem a fase de treinamento de UPOD são ilustrados através do esquema a seguir:

Figura 8: Esquema macro da fase de treinamento de UPOD




O algoritmo 2 descreve detalhadamente todos os passos ilustrados no esquema. São recebidos como parâmetros de entrada o $dataset$ sem qualquer tipo de tratamento, o mapa contendo os atributos dos usuários $Feat$, o valor mínimo de grupos $kmin$, o valor máximo de grupos $kmax$, o conjunto de todos valores possíveis para $\lambda$, $\Lambda = \{0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0\}$, e o conjunto de métricas utilizadas para o cálculo do melhor valor de $\lambda$ por grupo, $Metric$.

Primeiramente ocorre o pré-processamento do conjunto de treinamento (Passo 1), que inclui a eliminação de usuários com dados inválidos e/ou ausentes, remoção de data e horário de interação entre usuários e itens, transformação de código postal em nome de países e/ou estados, etc. Ainda neste passo, os valores de atributos são transformados em categóricos. Por exemplo: o atributo de idade de um usuário é transformado em faixas de 5 em 5 anos, e cada faixa representa uma categoria.

**Algoritmo 2** – Fase de treinamento de $UPOD$

**Entrada:** $dataset$, $Feat$, $kmin$, $kmax$, $\Lambda$, $Metric$

1: $TrainData \leftarrow prep(dataset)$
2: $Eval \leftarrow 0$
3: **para** k = $kmin$ até k = $kmax$ **faça**
4: $\quad Clusters_k \leftarrow clustering(TrainData, Feat, k)$
5: $\quad Eval_k \leftarrow evaluate(Clusters_k)$
6: $\quad$ **se** $Eval_k$ é melhor que $Eval$ **então**
7: $\quad\quad Eval \leftarrow Eval_k$
8: $\quad\quad BestClusters \leftarrow Clusters_k$
9: $\quad\quad Best_k \leftarrow k$
10: $\quad$ **fim se**
11: **fim para**
12: $G \leftarrow buildGraph(TrainData)$
13: $TrainSet \leftarrow \{\}$
14: **para** $i = 1$ até $i = Best_k$ **faça**
15: $\quad \lambda_{best}[i] \leftarrow BestLambda(G, BestClusters[i], \Lambda, Metric)$
16: $\quad$ **para** cada $user$ em $BestClusters[i]$ **faça**
17: $\quad\quad userFeat \leftarrow ExtractFeat(user, Feat)$
18: $\quad\quad TrainSet \leftarrow TrainSet \cup \{< userFeat, \lambda_{best}[i] >\}$
19: $\quad$ **fim para**
20: **fim para**
21: $classifier \leftarrow TrainClassifier(TrainSet)$
22: **retorna** $classifier, G$

A partir do Passo 2 até o Passo 11, o conjunto de dados pré-processado é agrupado em $k$ grupos (*clusters*) pela execução do algoritmo $k - modes$ (HUANG, 1998). Este algoritmo utiliza a medida de dissimilaridade para dados categóricos, representa os centróides dos grupos com modas (*modes*) e utiliza um método baseado em frequência para atualizar as modas no processo de agrupamento. A medida de dissimilaridade entre X e Y em $k - modes$ é

$$d(X, Y) = \sum_{j=1}^{m} \delta(x_j, y_j), \tag{4.1}$$

onde $\delta(x_j, y_j)$ é igual a $0$ se $x_j = y_j$ e é igual a $1$ se $x_j \neq y_j$, $j$ corresponde a cada um dos $m$ atributos categóricos que representa $X$ e $Y$.

A moda (*mode*) de um conjunto de $n$ objetos categóricos $X_i$ é o vetor **Q** que minimiza a função $D(X,\mathbf{Q})$:

$$D(X, \mathbf{Q}) = \sum_{i=1}^{n} d(X_i, \mathbf{Q}). \tag{4.2}$$

Logo, no Passo 4 do Algoritmo 2 a função $clustering$ representa o algoritmo $k - modes$, utilizado com $k$ variando de $kmin$ até $kmax$ e, para cada $k$, uma avaliação dos $k$ clusters é, então, executada. Os passos referentes à $k - modes$ são ilustrados no Algoritmo 3 (HUANG, 1998), o qual recebe como parâmetros de entrada o $dataset$ que terá seus usuários agrupados, o mapa contendo os atributos dos usuários $Feat$ e o número de grupos desejado, $k$.

**Algoritmo 3** Algoritmo $k - modes$ ($clustering$)

**Entrada:** $dataset$, $Feat$, $k$

1: Selecione $k$ modas iniciais, uma para cada grupo
2: **para** cada objeto $o$ de $< dataset, Feat >$ **faça**
3: Aloque $o$ ao grupo cuja moda seja a mais próxima de acordo com 4.2
4: **fim para**
5: **para** cada moda de cada grupo **faça**
6: Reteste a dissimilaridade dos objetos com relação às modas atuais
7: **se** a moda mais próxima pertence a outro grupo **então**
8: Realoque o objeto para aquele grupo
9: Atualize as modas de ambos os grupos
10: **fim se**
11: **fim para**
12: $Clusters_k \leftarrow$ conjunto de $k$ grupos finais
13: **retorna** $Clusters_k$

No Passo 5 do algoritmo 2, para cada $k$ grupos retornados pela função de agrupamento que implementa o algoritmo $k - modes$ (Algoritmo 3), a integridade da partição de dados obtida é avaliada. Para este propósito foi utilizado o critério de agrupamento baseado em entropia apresentado em Chen e Liu (2009).

Considere um conjunto de dados $X = \{x_1, x_2, ..., x_n\}$ com $n$ instâncias e $c$ colunas. Cada instância $x_i$ é descrita por $c$ colunas ($x_i = \{x_1^i, x_2^i, ..., x_c^i\}$) e cada coluna de $x_i$ possui um valor do domínio $A_i$, no qual existe um número finito de valores categóricos únicos. Se $a \in A_i$, então a propabilidade de $x_i = a$ é dada por $p(x_i = a)$ e a entropia da coluna de $A_i$ é dada por:

$$H(A_i \mid X) = -\sum_{a \in A_i} p(a \mid X) \log_2 p(a \mid X), \tag{4.3}$$

onde $p(a \mid X)$ é uma propabilidade empírica estimada em $X$. A entropia estimada de todo o conjunto de dados é a soma das entropias das colunas e é representada por $H(X)$.

Além disso, supondo que o conjunto de dados $X$ seja particionado em $k$ grupos, com $C^k = \{C_1, C_2, ...C_k\}$ sendo o conjunto de grupos (partição), $C_k$ é um grupo, $n_k$ é o número de objetos em $C_k$, e $H(C_k)$ é a entropia do grupo. De acordo com Chen e Liu (2005) e Chen e Liu (2009), o critério de agrupamento baseado em entropia é dado por

$$Opt(C^k) = \frac{1}{c}\left(H(X) - \frac{1}{n}\sum_{k=1}^{k} n_k H(C_k)\right). \tag{4.4}$$

Repare que $H(X)$ é fixo e depende do conjunto de dados, de forma que a maximização de $Opt(C^k)$ equivale à minimização da expressão $\frac{1}{n}\sum_{k=1}^{k} n_k H(C_k)$, que é chamada de "entropia esperada" da partição $C^k$. Então, o "melhor k" é aquele que resulta no valor mínimo de entropia esperada entre todos $k$ grupos.

Logo, ao final do laço (Passos 3 a 11 do Algoritmo 2), tem-se em $BestClusters$ o melhor conjunto de grupos para o melhor $k$, dado em $Best_k$.

É necessário, então, determinar o melhor $\lambda$ para cada grupo ($\lambda_{best}$), o qual representa o perfil dos usuários presentes naquele grupo; isto é feito no Passo 15 do

Algoritmo 2 com a chamada da função $BestLambda$, que é detalhada no Algoritmo 4. Este algoritmo recebe como parâmetros de entrada o grafo bipartido $G$, o grupo para o qual deseja-se calcular o melhor $\lambda$, $BestClusters[i]$, o conjunto de todos valores possíveis de $\lambda$, $\Lambda = \{0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0\}$, e o conjunto de métricas utilizadas para obter o melhor $\lambda$, $Metric$.

**Algoritmo 4** - Algoritmo $BestLambda$, que determina o melhor $\lambda$ para um dado grupo $BestClusters[i]$

**Entrada:** $G$, $BestClusters[i]$, $\Lambda$, $Metric$

1: $Eval_\lambda \leftarrow 0$
2: **para** cada $\lambda$ em $\Lambda$ **faça**
3: $\quad RecList_\lambda \leftarrow MD_HS(G, BestCluster[i], \lambda)$
4: $\quad Eval \leftarrow Metric(RecList_\lambda)$
5: $\quad$ **se** $Eval$ é melhor que $Eval_\lambda$ **então**
6: $\quad\quad Eval_\lambda \leftarrow Eval$
7: $\quad\quad \lambda_{best}[i] \leftarrow \lambda$
8: $\quad$ **fim se**
9: **fim para**
10: **retorna** $\lambda_{best}[i]$

Ressaltamos neste ponto que as métricas consideradas em $Metric$ serão detalhadas na Seção 5.3 do capítulo 5.

Uma vez que o valor de $\lambda_{best}[i]$ foi determinado para um dado grupo $i$ no conjunto $BestClusters$, os atributos de cada usuário contidos neste grupo são extraídos para compor um par de treinamento para o algoritmo de classificação. O par $< userFeat, \lambda_{best}[i] >$ é então inserido no conjunto de treinamento $TrainSet$ (Passos 17 e 18 do Algoritmo 2).

Finalmente, utilizando o conjunto $TrainSet$ de treinamento gerado, um algoritmo de classificação é treinado no Passo 21 de forma que, dados os atributos do usuário alvo (ex. idade, gênero, profissão, etc), o classificador treinado possa predizer qual o $\lambda$ mais apropriado para representar o perfil deste usuário.

As saídas da fase de treinamento são o classificador treinado e o grafo $G$ de interações entre usuários e itens na base de dados original. Para os resultados apresentados nesta dissertação foi utilizado o algoritmo de classificação SVM (*Support Vector Machine*) (WITTEN et al., 1999), devido ao mesmo ter apresentado melhor acurácia

nos resultados do que os demais. No entanto, qualquer outro tipo de classificador poderia ser considerado.

O classificador é utilizado para a predição do melhor valor de $\lambda$ para um usuário alvo na fase de recomendação. Esta fase requer como entrada o grafo de interações entre usuários e itens $G$, o mapa de atributos de usuários $Feat$, o classificador treinado construído na fase de treinamento, o tamanho da lista de recomendação $|L|$ e o usuário alvo.

A fase de recomendação de $UPOD$ é ilustrada no Algoritmo 5, o qual recebe como parâmetros de entrada o grafo bipartido $G$, o usuário alvo da recomendação $user$, o mapa contendo os atributos dos usuários $Feat$, o classificador treinado na fase de treinamento $classifier$, e o tamanho da lista de recomendação $|L|$.

**Algoritmo 5** - Fase de Recomendação de UPOD

**Entrada:** $G$, $user$, $Feat$, $classifier$, $|L|$.

1: $userFeatures \leftarrow ExtractFeat\ (user, Feat)$
2: $\lambda_{user} \leftarrow classifier\ (userFeatures)$
3: $recommendList \leftarrow MDHS\,(G, user, \lambda_{user}, |L|)$
4: **retorna** $recommendList$

Primeiro, os valores dos atributos do usuário alvo $userFeatures$ são extraidos no Passo 1 considerando o mapa de atributos de usuários $Feat$. Os valores representados em $userFeatures$ são entradas para o classificador treinado, o qual prediz o valor apropriado de $\lambda$ para o usuário, representado por $\lambda_{user}$ no Passo 2 do Algoritmo 5.

A variável $\lambda_{user}$ reflete o perfil do usuário alvo, definindo a combinação apropriada do algoritmo $MD$, que seleciona os itens populares, com $HS$, que seleciona itens menos conhecidos (novidades). Então, no Passo 3 o algoritmo $MDHS$ é executado sobre o grafo $G$ para o usuário alvo e seu parâmetro de sincronização personalizado ($\lambda_{user}$), sendo gerada a lista de recomendação $L$. Note que agora a lista de recomendação gerada combina itens populares e aqueles pouco conhecidos de acordo com o perfil particular do usuário.

A próxima seção apresenta alguns detalhes da implementação que foi feita para que fosse possível a execução do algoritmo $UPOD$.

## 4.3 Detalhes de Implementação

A implementação do $UPOD$ foi desenvolvida totalmente na linguagem de programação Java e está organizada em dois projetos Java maven, os quais são responsáveis pelo(a):

- Preprocessamento das bases de dados com a remoção de dados inválidos ou incompletos, conversão de código postal em paíseis ou estados, entre outras tranformações, e a criação do grafo bipartido $G$;
- Execução parametrizada dos algoritmos (FC, MDHS e UPOD, usados nos experimentos) e cálculo das métricas de avaliação dos resultados.

Foram feitas adaptações nos projetos para que pudessem ser executados tanto em máquina local quanto também na infraestrutura oferecida pelo *Google Compute Engine*, uma das ferramentas da plataforma em nuvem do Google. Esta alternativa fez com que fosse possível ter diferentes execuções do programa em paralelo.

Isto foi possível pois, durante a implementação, foi criado um arquivo de configuração, que é lido pelo programa principal e que contém todos os parâmetros necessários para a execução, tal como o caminho para os arquivos das bases de dados originais, o algoritmo considerado para a execução, o tamanho das listas de recomendação (|L|), entre outros. Assim, é possível executar de forma paralela o mesmo programa para o algoritmo MDHS e UPOD, por exemplo.

Outro detalhe importante sobre a implementação do algoritmo $UPOD$ é que foi considerada a *API* da biblioteca *Weka* para o treinamento supervisionado do classificador SVM (Passo 21 do Algoritmo 2) e para a predição do valor ideal de $\lambda$ para cada usuário (Passo 2 do Algoritmo 5).

# 5 RESULTADOS EXPERIMENTAIS E DISCUSSÃO

Foram executados diversos experimentos para validar o algoritmo $UPOD$ proposto. Foi feita uma análise comparativa com dois outros algoritmos de recomendação ($MDHS$ e $FC$) e diferentes bases de dados.

Os resultados foram avaliados de acordo com algumas métricas amplamente utilizadas para este tipo de aplicação. Os detalhes de tais experimentos e os resultados obtidos são descritos a seguir.

## 5.1 Bases de Dados

Foram consideradas três bases de dados bem conhecidas na literatura:

**MovieLens** (HARPER; KONSTAN, 2015) trata-se de uma base de dados sobre filmes, contendo 910 usuários, 1.672 itens e 95.579 interações. Os atributos de usuário são a idade, o gênero e a localização.

**Last.FM** (CELMA, 2010). Neste trabalho foi considerado um subconjunto aleatório da base de dados completa sobre músicas, contendo 2.846 usuários, 4.995 itens e 14.583 interações. Os atributos de usuário são a idade, o gênero e o país.

**Book-Crossing** (ZIEGLER et al., 2005) A base de dados Book-Crossing é uma coleção de classificações de livros feitas por usuários. Foi usado um subconjunto aleatório da base de dados completa, contendo 3.421 usuários, 26.811 itens e 35.572 interações. Os atributos de usuário nesta base de dados são a idade e a localização.

Com base na equação 2.1 apresentada na seção 2.1 do capítulo 2, o nível de esparsidade das bases Movielens, Last.FM e Book-Crossing são dados pela Tabela 2. Pode-se observar que Book-Crossing é a base de dados mais esparsa.

Tabela 2: Nível de esparsidade das bases de dados consideradas.

| **Base de dados** | **Esparsidade** |
|---|---|
| MovieLens | 0,9371 |
| Last.FM | 0,9989 |
| Book-Crossing | 0,9996 |

## 5.2 Algoritmos utilizados para comparação

O algoritmo $UPOD$ proposto foi avaliado comparativamente com dois outros algoritmos: (i) o algoritmo base para nosso trabalho, $MDHS$, com $\lambda = 0.5$, seguindo as recomendações dos autores (Algoritmo 1); e (ii) o algoritmo básico de filtragem colaborativa, $FC$, (Algoritmo 6).

O Algoritmo 6 é detalhado em Ricci et al. (2010), Patra et al. (2015) e Yang et al. (2016); e recebe como parâmetros de entrada o usuário alvo da recomendação $u$, o $dataset$, o tamanho da lista de recomendação $|L|$ e o tamanho da vizinhança do usuário $Vz$. É utilizado o Coeficiente de Correlação de Pearson (CCP) (PEARSON, 1920) no Passo 04, cuja equação 2.2 foi previamente detalhada na seção 2.8 desta dissertação. No passo 10 do algoritmo é calculado $pred(u, i)$ dado pela equação 2.4; também detalhada na seção 2.8 do capítulo 2.

**Algoritmo 6** – Algoritmo $FC$
(RICCI et al., 2010), (PATRA et al., 2015), (YANG et al., 2016)

**Entrada:** usuário $u$, $dataset$, $|L|$, $Vz$.
1: $ratingsMatrix \leftarrow$ Extrai a matriz de *ratings* dos usuários e itens da base de dados
2: **para** cada usuário $v \in dataset$, $v \neq u$ **faça**
3: $I_{uv} \leftarrow$ Separe os itens avaliados por ambos $u$ e $v$ em $ratingsMatrix$
4: Calcule $CCP(u, v)$ utilizando $I_{uv}$ e $ratingsMatrix$ (Equação 2.2)
5: **fim para**
6: $List_{users} \leftarrow$ Ordene os usuários decrescentemente de acordo com $CCP$
7: $V \leftarrow$ Separe os primeiros $Vz$ usuários de $List_{users}$
8: $I_u \leftarrow$ Obtenha os itens com os quais $u$ interagiu
9: **para** todos $i_s$, desde que $i \notin I_u$ **faça**
10: Calcule $pred(u, i)$ utilizando $V$ como a vizinhança do usuário (Equação 2.4, (PATRA et al., 2015) )
11: **fim para**
12: $list \leftarrow$ Ordene todos itens decrescentemente de acordo com o valor de $pred(u, i)$
13: $L \leftarrow$ Separe os $|L|$ primeiros elementos de $list$
14: **retorna** $L$

Assim como $MDHS$, o algoritmo $FC$ também foi reproduzido.

## 5.3 Método de avaliação e métricas

Foi considerada a técnica de validação cruzada com $k$ grupos (*k-fold cross-validation*, em inglês) para validar nossa proposta, que se trata de uma técnica de validação de modelo utilizada para avaliar como os resultados de uma análise estatística generalizam para uma base de dados independente. Esta abordagem consiste em dividir aleatoriamente o conjunto de dados em $k$ grupos de tamanhos aproximadamente iguais. O primeiro grupo é tratado como conjunto de validação e o restante ($k - 1$ grupos) como conjunto de treinamento. O procedimento é repetido para cada grupo e é calculada a média dos resultados da validação entre todas execuções para estimar o desempenho preditivo do modelo. Nos experimentos foi usado $k = 10$.

Foram apenas considerados aqueles usuários presentes em ambos os conjuntos, de treinamento e de validação. Estes usuários são chamados de "usuários válidos" e são representados por $U_{valid}$.

Foram utilizadas três métricas para validar as listas de recomendação geradas, descritas a seguir.

### 5.3.1 *Recall*

Esta métrica é utilizada para medir a proporção de itens do conjunto de validação que correspondem a itens presentes na lista de recomendação gerada para o usuário (ZENG et al., 2013; YANG et al., 2016):

$$Re_u(L) = \frac{d_u(L)}{I_u}, \tag{5.1}$$

onde $d_u(L)$ é o número de itens relacionados com $u$ no conjunto de validação e que também estão presentes na lista de recomendação $L$; $I_u$ é o número total de itens relacionados com $u$ no conjunto de validação.

Este processo normalmente é repetido para cada usuário válido, considerando-o como o usuário alvo do teste corrente. O valor final do $Recall$ é calculado como a

média dos valores parciais obtidos para cada usuário alvo:

$$Re(L) = \frac{\sum_{u \in U_{valid}} Re_u(L)}{|U_{valid}|}, \tag{5.2}$$

onde $|U_{valid}|$ representa a quantia de usuários válidos do conjunto de validação.

Quanto maior seu valor, mais itens no conjunto de validação correspondem a itens recomendados.

### 5.3.2 *Precision*

Esta métrica é utilizada para medir a proporção dos itens na lista de recomendação do usuário que correspondem a ele no conjunto de validação (ZENG et al., 2013; YANG et al., 2016):

$$Pe_u(L) = \frac{d_u(L)}{|L|}, \tag{5.3}$$

onde $d_u(L)$ é o número de itens relacionados com o usuário $u$ no conjunto de validação presentes na lista de recomendação e $|L|$ é o tamanho da lista de recomendação. Da mesma forma que *Recall*, o cálculo final desta métrica é feito extraindo a média dos valores parciais obtidos para cada usuário alvo, tal como segue:

$$Pe(L) = \frac{\sum_{u \in U_{valid}} Pe_u(L)}{|U_{valid}|}, \tag{5.4}$$

onde $|U_{valid}|$ representa a quantia de usuários válidos presentes no conjunto de validação.

Quanto maior seu valor, mais itens na lista de recomendação correspondem a itens no conjunto de validação.

### 5.3.3 *Ranking Score*

A métrica *Ranking Score (RS)* é geralmente considerada em trabalhos que exploram algoritmos de recomendação baseados em processos de difusão de massa, tais como em Zhou et al. (2010), Zeng et al. (2013), Zhang e Zeng (2015), Ma et al. (2017).

Para o cálculo do RS é levada em conta a posição em que itens relacionados ao

usuário alvo no conjunto de validação aparecem na lista de recomendação gerada; tal posição é conhecida como *rank* do item, o qual é obtido em função da posição do item $i$ na lista de recomendação, dividido pelo número de itens desconhecidos pelo usuário inicialmente.

Suponha que um determinado item ocupe a 3ª posição da lista de recomendação gerada e existam 100 itens "desconhecidos" pelo usuário, logo o valor de *rank* seria 0.03. Assim, quanto mais próximo das primeiras posições da lista, menor o valor do *rank*. Existem casos, ainda, onde o item relacionado ao usuário $u$ no conjunto de validação não corresponde a itens presentes na lista de recomendação; nestes casos os autores dos trabalhos geralmente consideram algum tipo de ponderação no cálculo de *rank*.

De acordo com Zeng et al. (2013), Yu et al. (2015) e Chen, Zeng e Chen (2015), o valor de RS para um dado usuário $u$ é dado por:

$$RS_u = \sum_{i \in I_u} \frac{rank(i)}{|I_u|}, \tag{5.5}$$

onde $I_u$ é o conjunto de itens relacionados a $u$ no conjunto de validação e $rank(i)$ representa o *rank* do item $i$ na lista de recomendação gerada.

Tal como ocorre nas demais métricas, o resultado final é calculado pela média dos valores parciais obtidos para cada usuário do conjunto de validação:

$$RS = \frac{\sum_{u \in U_{valid}} RS_u}{|U_{valid}|}, \tag{5.6}$$

onde $|U_{valid}|$ representa a quantia de usuários válidos presentes no conjunto de validação e $RS_u$ é o resultado parcial para cada usuário $u$.

Quanto menor seu valor, mais itens relacionados aos usuários no conjunto de validação estarão presentes nas primeiras posições das listas de recomendação.

## 5.4 Resultados e Discussão

Em todos experimentos consideramos $|L| = 30$ como tamanho para as listas de recomendação. No caso do algoritmo FC, o tamanho da vizinhança foi $|Vz| = 30$; e para o algoritmo $UPOD$ o valor de $k_{min}$ foi definido experimentalmente como 100, enquanto que $k_{max}$ foi definido de forma a manter em cada um dos grupos o número máximo de 15 a 20 usuários, portanto $k_{max} = 200$ para todas as bases de dados.

O número de instâncias reservadas para treinamento e testes em cada uma das bases de dados é ilustrado na tabela 3.

Tabela 3: Tamanho das bases de treinamento e testes

| **Base de dados** | **Instâncias para treinamento** | **Instâncias para teste** |
|---|---|---|
| MovieLens | 86217 | 9580 |
| Last.FM | 13068 | 1453 |
| Book-Crossing | 31682 | 3521 |

Após todas as execuções do algoritmo $UPOD$ para cada uma das bases de dados foram calculadas as métricas apresentadas na seção 5.3 e também foi analisada a distribuição dos valores de $\lambda$ no dataset gerado ($TrainSet$) e utilizado para o treinamento do algoritmo de classificação SVM.

As figuras abaixo ilustram esta informação através de histogramas gerados para cada uma das bases de dados:

Figura 9: Distribuição dos valores de $\lambda$ para a base de dados **Movielens**

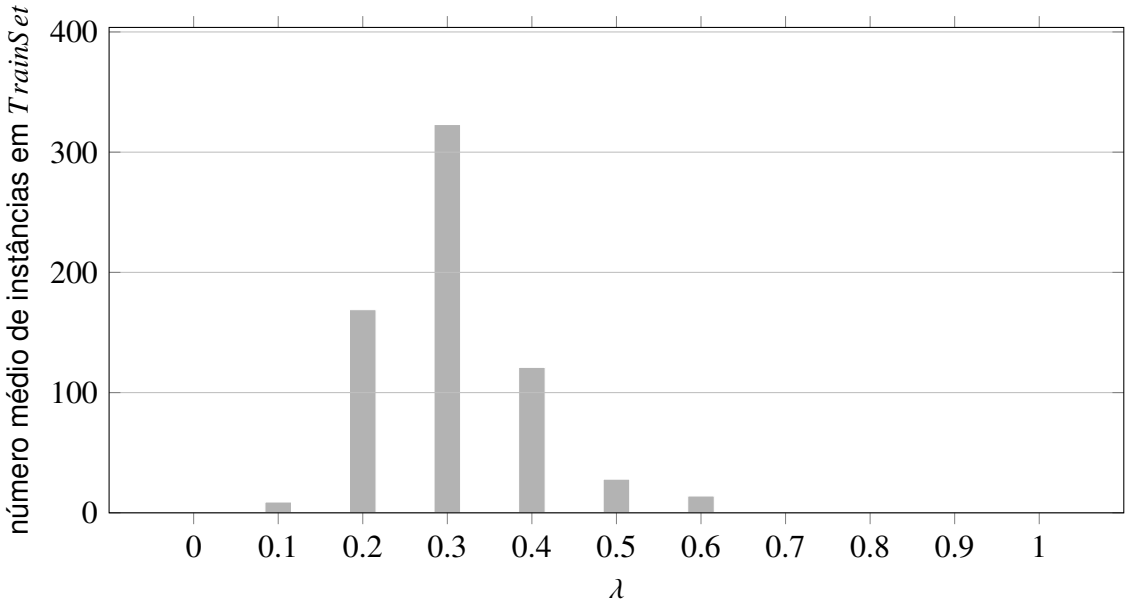



Figura 10: Distribuição dos valores de $\lambda$ para a base de dados **Last.FM**

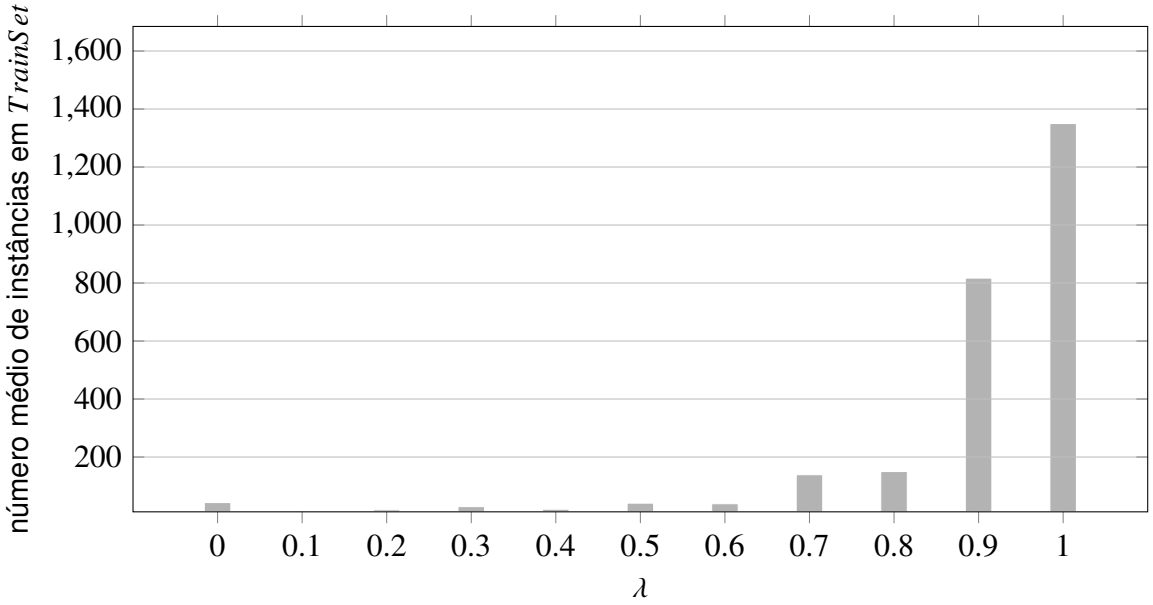
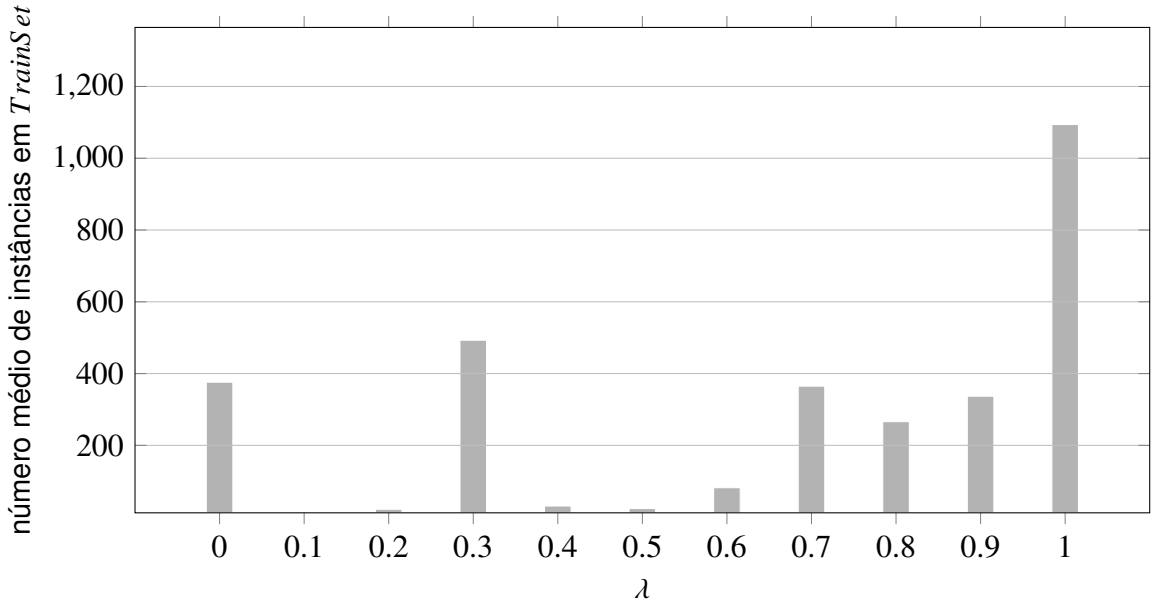


Figura 11: Distribuição dos valores de $\lambda$ para a base de dados **Book.Crossing**




Com base na análise dos histogramas podemos observar que na base de dados Movielens os usuários preferem receber recomendações com maior quantia de novidade dada a maior frequência de $\lambda_s$ com valores abaixo de 0.5, enquanto que na base de dados Last.FM a maioria dos usuários preferem recomendações com maior quantia de itens populares devido à maior frequência de $\lambda_s$ com valores próximos a 1. Na base de dados Book.Crossing, por outro lado, apesar de uma grande quantia de usuários preferirem popularidade nas recomendações, existe uma quantia significativa de usuários que preferem novidade. Estas dados são muito importantes para demonstrar a capacidade do algoritmo apresentado em diferenciar usuários através dos diferentes perfis.

Na sequencia, na Tabela 4 são apresentados os resultados obtidos para cada uma das bases de dados com relação às métricas de avaliação. Em negrito estão aqueles onde $UPOD$ superou os algoritmos $FC$ e $MDHS$, ao mesmo tempo, de acordo com o teste-t com 95% de intervalo de confiança, o que significa que a diferença entre os resultados é considerada estatisticamente significativa.

A Tabela 4(a) apresenta os resultados para a base **Movielens**, a Tabela 4(b) apresenta os resultados para a base **Last.FM** e a Tabela 4(c) apresenta os resultados para a base **Book-Crossing**.

Pode-se observar na Tabela 4 que o algoritmo $UPOD$ supera significativamente os outros algoritmos em relação às métricas consideradas, apesar da diferença entre os resultados não ser estatisticamente significativa em todos os casos. Por exemplo, de acordo com o teste-t, $UPOD$ não supera $FC$ e $MDHS$ com relação à métrica *RankScore* nas bases de dados **Last.FM** e **Book-Crossing**. No entanto, nossa proposta sistematicamente supera os demais algoritmos com relação às métricas *Recall* e *Precision*.

Tabela 4: Resultados Experimentais para as três bases de dados.

| **Algoritmo** | **RS** | | **Re(L)** | | **Pe(L)** | |
|---|---|---|---|---|---|---|
| | avg | $\sigma$ | avg | $\sigma$ | avg | $\sigma$ |
| FC | 0.497 | 0.002 | 0.014 | 0.002 | 0.003 | $4.6x10^{-4}$ |
| *MDHS ($\lambda$ = 0.5)* | 0.080 | 0.001 | 0.299 | 0.008 | 0.086 | 0.001 |
| *UPOD* | **0.074** | **0.001** | **0.308** | **0.007** | **0.091** | **0.002** |

(a) Resultados obtidos para a base de dados **Movielens**. Com $k_{avg} = 197$ e $k_{\sigma} = 4.25$ para o algoritmo $UPOD$, ambas calculadas após a aplicação do critério de clusterização baseado em entropia em cada uma das 10 execuções. O tempo médio de processamento para FC, MDHS e UPOD foi 5 min, 30s e 5s, respectivamente.

| **Algoritmo** | **RS** | | **Re(L)** | | **Pe(L)** | |
|---|---|---|---|---|---|---|
| | avg | $\sigma$ | avg | $\sigma$ | avg | $\sigma$ |
| FC | 0.444 | 0.004 | 0.018 | 0.004 | $9.6x10^{-4}$ | $2x10^{-4}$ |
| *MDHS ($\lambda = 0.5$)* | 0.252 | 0.007 | 0.119 | 0.006 | 0.006 | $2.9x10^{-4}$ |
| *UPOD* | 0.251 | 0.006 | **0.174** | **0.014** | **0.009** | **7.4x10$^{-4}$** |

(b) Resultados obtidos para a base de dados **Last.FM**. Com $k_{avg} = 196$ e $k_{\sigma} = 2.94$ para o algoritmo $UPOD$, ambas calculadas após a aplicação do critério de clusterização baseado em entropia em cada uma das 10 execuções. O tempo médio de processamento para FC, MDHS e UPOD foi 2 min, 18s e 9s, respectivamente.

| **Algoritmo** | **RS** | | **Re(L)** | | **Pe(L)** | |
|---|---|---|---|---|---|---|
| | avg | $\sigma$ | avg | $\sigma$ | avg | $\sigma$ |
| FC | 0.406 | 0.007 | $3.6x10^{-4}$ | $2.4x10^{-4}$ | $3.5x10^{-5}$ | $1x10^{-5}$ |
| *MDHS ($\lambda = 0.5$)* | 0.395 | 0.007 | 0.002 | $4.7x10^{-4}$ | $2x10^{-4}$ | $4.9x10^{-5}$ |
| *UPOD* | 0.394 | 0.008 | **0.003** | **0.001** | **3.76x10$^{-4}$** | **7.3x10$^{-5}$** |

(c) Resultados obtidos para a base de dados **Book-Crossing**. Com $k_{avg} = 192$ e $k_{\sigma} = 8.5$ para o algoritmo $UPOD$, ambas calculadas após a aplicação do critério de clusterização baseado em entropia em cada uma das 10 execuções. O tempo médio de processamento para FC, MDHS e UPOD foi 20 min, 30s e 5s, respectivamente.

# 6 CONCLUSÃO E TRABALHOS FUTUROS

Esta dissertação apresentou o algoritmo $UPOD$ que permitiu a sintonização automática do parâmetro $\lambda$ (grau de mistura) no processo de difusão de massa presente no algoritmo $MDHS$. Tal sintonização torna possível a geração de recomendações personalizadas, combinando popularidade e novidade de acordo com o perfil particular de cada usuário. Mostrou-se, através de vários experimentos utilizando três bases de dados bem conhecidas na literatura, que o aprendizado supervisionado do parâmetro $\lambda$ de acordo com o perfil particular do usuário pode melhorar a qualidade das recomendações em conjuntos de dados altamente esparsos.

Pesquisas futuras podem ser realizadas com o objetivo de incluir os valores de *rating*, atribuídos aos itens no sistema, no cálculo dos valores de recurso durante o processo de difusão de massa no grafo bipartido. Desta forma, itens melhores avaliados no sistema seriam recomendados com maior intensidade do que itens mal avaliados. Outra possível melhoria seria permitir uma quantia maior de propagações de recurso no grafo, a exemplo do que foi feito no algoritmo *Semi-Local Diffusion* (ZENG et al., 2013).

Outros desafios relevantes incluem, por exemplo, a transparência no que tange o papel da explicação de uma recomendação, tornando-a mais aceitável e justificável para o usuário. Aspectos referentes à avaliação dos sistemas de recomendação também oferecem uma gama de dimensões possíveis e importantes que devem ser melhor investigadas. O valor temporal das recomendações também é bastante relevante. Neste caso, o desafio refere-se ao fato de que um determinado conjunto de recomendações pode não ser aplicável para sempre, isto é, existe um intervalo de tempo em que esses itens podem ser recomendados. Isso é evidente, por exemplo, quando se trata de notícias; as pessoas querem ser informadas sobre os eventos e notícias recentes. Enfim, diversos aspectos devem ser integrados no projeto de um sistema de recomendação para que ele possa gerar recomendações cada vez melhores e mais aderentes ao perfil de cada usuário.

# REFERÊNCIAS


AGGARWAL, C. C. **Knowledge-Based Recommender Systems**. Springer, 2016. ISSN 1875-6883. ISBN 9078677048. Disponível em: <http://link.springer.com/10.1007/978-3-319-29659-3_5>.

BAO, J.; ZHENG, Y.; WILKIE, D.; MOKBEL, M. Recommendations in location-based social networks: a survey. **GeoInformatica**, v. 19, n. 3, p. 525–565, 2015. ISSN 13846175.

BEEL, J.; GIPP, B.; LANGER, S.; BREITINGER, C. Research-paper recommender systems: a literature survey. **International Journal on Digital Libraries**, Springer Berlin Heidelberg, v. 17, n. 4, p. 305–338, 2016. ISSN 14321300.

BELLOGíN, A.; CANTADOR, I.; CASTELLS, P. A comparative study of heterogeneous item recommendations in social systems. **Information Sciences**, v. 221, p. 142–169, 2013. ISSN 00200255.

BOBADILLA, J.; ORTEGA, F.; HERNANDO, a.; GUTIÉRREZ, a. Recommender systems survey. **Knowledge-Based Systems**, Elsevier B.V., v. 46, p. 109–132, 2013. ISSN 09507051. Disponível em: <http://dx.doi.org/10.1016/j.knosys.2013.03.012>.

CELMA, O. **Music Recommendation and Discovery in the Long Tail**. [S.l.]: Springer, 2010.

CHEN, B.-l.; ZENG, A.; CHEN, L. The effect of heterogeneous dynamics of online users on information filtering. **Physics Letters A**, Elsevier B.V., v. 379, n. 43-44, p. 2839–2844, 2015. ISSN 0375-9601. Disponível em: <http://dx.doi.org/10.1016/j.physleta.2015.09.019>.

CHEN, K.; LIU, L. The " Best K " for Entropy-based Categorical Data Clustering. In: . [s.n.], 2005. p. 253—-262. Disponível em: <http://dl.acm.org/citation.cfm?id=1116877.1116909>.

____. "Best K": Critical clustering structures in categorical datasets. **Knowledge and Information Systems**, v. 20, n. 1, p. 1–33, 2009. ISSN 02191377.

CHEN, Y.; WU, C.; XIE, M.; GUO, X. Solving the sparsity problem in recommender systems using association retrieval. **Journal of Computers**, v. 6, n. 9, p. 1896–1902, 2011. ISSN 1796203X.

CHOI, K.; YOO, D.; KIM, G.; SUH, Y. A hybrid online-product recommendation system: Combining implicit rating-based collaborative filtering and sequential pattern analysis. **Electronic Commerce Research and Applications**, Elsevier B.V., v. 11, n. 4, p. 309–317, 2012. ISSN 15674223. Disponível em: <http://dx.doi.org/10.1016/j.elerap.2012.02.004>.

CHRISTAKOPOULOU, E.; KARYPIS, G. Local item-item models for top-n recommendation. In: **Proceedings of the 10th ACM Conference on Recommender Systems - RecSys '16**. Association for Computing Machinery, 2016. p. 67–74. ISBN 9781450340359. Disponível em: <http://dl.acm.org/citation.cfm?doid=2959100.2959185>.

FU, M.; QU, H.; MOGES, D.; LU, L. Attention based collaborative filtering. **Neurocomputing**, v. 311, p. 88–98, 2018. ISSN 18728286.

HARPER, F. M.; KONSTAN, J. A. The movielens datasets: History and context. **ACM Trans. Interact. Intell. Syst.**, ACM, New York, NY, USA, v. 5, n. 4, p. 19:1–19:19, dez. 2015. ISSN 2160-6455. Disponível em: <http://doi.acm.org/10.1145/2827872>.

HE, C.; PARRA, D.; VERBERT, K. Interactive recommender systems: A survey of the state of the art and future research challenges and opportunities. **Expert Systems with Applications**, Elsevier Ltd, v. 56, p. 9–27, 2016. ISSN 09574174. Disponível em: <http://dx.doi.org/10.1016/j.eswa.2016.02.013>.

HORSBURGH, B.; CRAW, S.; MASSIE, S. Learning pseudo-tags to augment sparse tagging in hybrid music recommender systems. **Artificial Intelligence**, Elsevier B.V., v. 219, p. 25–39, 2015. ISSN 00043702. Disponível em: <http://dx.doi.org/10.1016/j.artint.2014.11.004>.

HUANG, Z. Extensions to the k -Means Algorithm for Clustering Large Data Sets with Categorical Values. v. 2, n. 3, p. 283–304, 1998.

HUANG, Z.; CHEN, H.; ZENG, D. Applying associative retrieval techniques to alleviate the sparsity problem in collaborative filtering. **ACM Transactions on Information Systems**, v. 22, n. 1, p. 116–142, 2004. ISSN 10468188.

JANNACH, D.; ZANKER, M.; FELFERNIG, A.; FRIEDRICH, G. **Recommender Systems: An Introduction**. 1st. ed. New York, NY, USA: Cambridge University Press, 2010. ISBN 0521493366, 9780521493369.

JAVARI, A.; JALILI, M. A probabilistic model to resolve diversity-accuracy challenge of recommendation systems. **Knowledge and Information Systems**, Springer London, v. 44, n. 3, p. 609–627, 2014. ISSN 02193116. Disponível em: <http://dx.doi.org/10.1007/s10115-014-0779-2>.

KAMINSKAS, M.; BRIDGE, D. Diversity, Serendipity, Novelty, and Coverage. **ACM Transactions on Interactive Intelligent Systems**, v. 7, n. 1, p. 1–42, 2016. ISSN 21606455.

KATARYA, R.; VERMA, O. P. Recent developments in affective recommender systems. **Physica A: Statistical Mechanics and its Applications**, Elsevier B.V., v. 461, p. 182–190, 2016. ISSN 03784371. Disponível em: <http://dx.doi.org/10.1016/j.physa.2016.05.046>.

KIM, H. N.; El Saddik, A. Exploring social tagging for personalized community recommendations. **User Modelling and User-Adapted Interaction**, v. 23, n. 2-3, p. 249–285, 2013. ISSN 09241868.

KOREN, Y.; BELL, R.; VOLINSKY, C. Matrix Factorization Techniques for Recommender Systems. **Computer**, v. 42, n. 8, p. 42–49, 2009. ISSN 0018-9162.

KOTKOV, D.; WANG, S.; VEIJALAINEN, J. A survey of serendipity in recommender systems. **Knowledge-Based Systems**, Elsevier B.V., v. 111, p. 180–192, 2016. ISSN 09507051. Disponível em: <http://dx.doi.org/10.1016/j.knosys.2016.08.014>.

LACERDA, A. Multi-Objective Ranked Bandits for Recommender Systems. **Neurocomputing**, Elsevier B.V., v. 246, p. 1–13. ISSN 09252312. Disponível em: <http://linkinghub.elsevier.com/retrieve/pii/S092523121730228X>.

LEE, K.; LEE, K. Escaping your comfort zone: A graph-based recommender system for finding novel recommendations among relevant items. **Expert Systems with Applications**, Elsevier Ltd, v. 42, n. 10, p. 4851–4858, 2015. ISSN 09574174. Disponível em: <http://dx.doi.org/10.1016/j.eswa.2014.07.024>.

LEE, S.; PARK, S.; KAHNG, M.; LEE, S. G. PathRank: Ranking nodes on a heterogeneous graph for flexible hybrid recommender systems. **Expert Systems with Applications**, Elsevier Ltd, v. 40, n. 2, p. 684–697, 2013. ISSN 09574174. Disponível em: <http://dx.doi.org/10.1016/j.eswa.2012.08.004>.

LEVINAS, C. A. **An Analysis of Memory Based Collaborative Filtering Recommender Systems with Improvement Proposals**. Dissertação (Mestrado) — Universitat Politècnica de Catalunya, September 2014.

LIU, H.; HU, Z.; MIAN, A.; TIAN, H.; ZHU, X. A new user similarity model to improve the accuracy of collaborative filtering. **Knowledge-Based Systems**, Elsevier B.V., v. 56, p. 156–166, 2014. ISSN 09507051. Disponível em: <http://dx.doi.org/10.1016/j.knosys.2013.11.006>.

LU, J.; SHAMBOUR, Q.; XU, Y.; LIN, Q.; ZHANG, G. A web-based personalized business partner recommendation system using fuzzy semantic techniques. **Computational Intelligence**, v. 29, n. 1, p. 37–69, 2013. ISSN 08247935.

LU, J.; WU, D.; MAO, M.; WANG, W.; ZHANG, G. Recommender system application developments: A survey. **Decision Support Systems**, Elsevier B.V., v. 74, p. 12–32, 2015. ISSN 01679236. Disponível em: <http://dx.doi.org/10.1016/j.dss.2015.03.008>.

LU, Z.; DOU, Z.; LIAN, J.; XIE, X.; YANG, Q. Content-based collaborative filtering for news topic recommendation. In: **Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence**. AAAI Press. p. 217–223. ISBN 9781577356998. Disponível em: <https://dl.acm.org/doi/10.5555/2887007.2887038>.

MA, W.; REN, C.; WU, Y.; WANG, S.; FENG, X. Personalized recommendation via unbalance full-connectivity inference. **Physica A: Statistical Mechanics and its Applications**, Elsevier B.V., v. 483, p. 273–279, 2017. ISSN 03784371. Disponível em: <http://linkinghub.elsevier.com/retrieve/pii/S0378437117303588>.

MORADI, P.; AHMADIAN, S. A reliability-based recommendation method to improve trust-aware recommender systems. **Expert Systems with Applications**,

Elsevier Ltd, v. 42, n. 21, p. 7386–7398, 2015. ISSN 09574174. Disponível em: <http://dx.doi.org/10.1016/j.eswa.2015.05.027>.

NIKOLAKOPOULOS, A. N.; KOUNELI, M. a.; GAROFALAKIS, J. D. Hierarchical Itemspace Rank: Exploiting hierarchy to alleviate sparsity in ranking-based recommendation. **Neurocomputing**, Elsevier, v. 163, p. 126–136, 2015. ISSN 09252312. Disponível em: <http://www.scopus.com/inward/record.url?eid=2-s2.0-84924196269{&}partnerID=tZOtx>.

PATRA, B. K.; LAUNONEN, R.; OLLIKAINEN, V.; NANDI, S. A new similarity measure using Bhattacharyya coefficient for collaborative filtering in sparse data. **Knowledge-Based Systems**, Elsevier B.V., v. 82, p. 163–177, 2015. ISSN 09507051. Disponível em: <http://linkinghub.elsevier.com/retrieve/pii/S0950705115000830>.

PEARSON, K. Notes on the history of correlation. **Biometrika Trust**, Oxford University Press, v. 13, n. 1, p. 25–45, 1920.

RICCI, F.; ROKACH, L.; SHAPIRA, B.; KANTOR, P. B. **Recommender Systems Handbook**. 1st. ed. New York, NY, USA: Springer-Verlag New York, Inc., 2010. ISBN 0387858199, 9780387858197.

SANCHEZ-MORENO, D.; Gil Gonzalez, A. B.; Munoz Vicente, M. D.; Lopez Batista, V. F.; Moreno Garcia, M. N. A collaborative filtering method for music recommendation using playing coefficients for artists and users. **Expert Systems with Applications**, v. 66, p. 1339–1351, 2016. ISSN 09574174.

SARWAR, B.; KARYPIS, G.; KONSTAN, J.; REIDL, J. Item-based collaborative filtering recommendation algorithms. In: . [s.n.], 2001. p. 285–295. ISBN 1581133480. ISSN 09501991. Disponível em: <http://portal.acm.org/citation.cfm?doid=371920.372071>.

SHAMBOUR, Q.; LU, J. A trust-semantic fusion-based recommendation approach for e-business applications. **Decision Support Systems**, Elsevier B.V., v. 54, n. 1, p. 768–780, 2012. ISSN 01679236. Disponível em: <http://dx.doi.org/10.1016/j.dss.2012.09.005>.

____. An effective recommender system by unifying user and item trust information for b2b applications. **Journal of Computer and System Sciences**, Elsevier Inc., v. 81, n. 7, p. 1110–1126, 2015. ISSN 00220000. Disponível em: <http://linkinghub.elsevier.com/retrieve/pii/S0022000014001998>.

SRIDEVI, M.; RAO, R. R. DECORS: A Simple and Efficient Demographic Collaborative Recommender System for Movie Recommendation. **Advances in Computational Sciences and Technology**, v. 10, n. 7, p. 1969–1979, 2017. ISSN 0973-6107.

VARGAS, S. Novelty and diversity enhancement and evaluation in recommender systems and information retrieval. In: . [s.n.], 2014. p. 1281–1281. ISBN 9781450322577. Disponível em: <http://dl.acm.org/citation.cfm?doid=2600428.2610382>.

WITTEN, I. H.; FRANK, E.; TRIGG, L.; HALL, M.; HOLMES, G.; CUNNINGHAM, S. J. Weka: Practical machine learning tools and techniques with Java implementations. In: KASABOV, N.; KO, K. (Ed.). **Proceedings of the ICONIP/ANZIIS/ANNES'99 Workshop on Emerging Knowledge Engineering and Connectionist-Based Information Systems**. [S.l.: s.n.], 1999. p. 192–196. Dunedin, New Zealand.

YANG, Z.; WU, B.; ZHENG, K.; WANG, X.; LEI, L. A Survey of Collaborative Filtering-Based Recommender Systems for Mobile Internet Applications. **IEEE Access**, v. 4, p. 3273–3287, 2016. ISSN 2169-3536.

YU, F.; ZENG, A.; GILLARD, S.; MEDO, M. Network-based recommendation algorithms: A review. **Physica A: Statistical Mechanics and its Applications**, v. 452, p. 192–208, 2015. ISSN 03784371.

ZENG, A.; VIDMER, A.; MEDO, M.; ZHANG, Y.-c. Information filtering via hybridization of similarity preferential diffusion processes. 2013. ISSN 0295-5075.

ZENG, W.; ZENG, A.; LIU, H.; SHANG, M.-S.; ZHOU, T. Uncovering the information core in recommender systems. **Scientific Reports**, v. 4, p. 6140, 2014. ISSN 2045-2322. Disponível em: <http://www.nature.com/articles/srep06140>.

ZENG, W.; ZENG, A.; SHANG, M. S.; ZHANG, Y. C. Information filtering in sparse online systems: Recommendation via semi-local diffusion. **PLoS ONE**, v. 8, n. 11, 2013. ISSN 19326203.

ZHANG, F.-G.; ZENG, A. Information Filtering via Heterogeneous Diffusion in Online Bipartite Networks. **PLoS ONE**, v. 10, n. 6, p. e0129459, 2015. ISSN 1932-6203. Disponível em: <http://dx.plos.org/10.1371/journal.pone.0129459>.

ZHANG, H.-R.; MIN, F. Three-way recommender systems based on random forests. **Knowledge-Based Systems**, Elsevier B.V., v. 91, p. 275–286, 2016. ISSN 09507051. Disponível em: <http://linkinghub.elsevier.com/retrieve/pii/S0950705115002373>.

ZHANG, S.; LI, C.; MA, L.; LI, Q. Alleviating the sparsity problem of collaborative filtering using rough set. **COMPEL - The international journal for computation and mathematics in electrical and electronic engineering**, v. 32, n. 2, p. 516–530, 2013. ISSN 0332-1649. Disponível em: <http://www.emeraldinsight.com/doi/abs/10.1108/03321641311296918>.

ZHANG, S.; YAO, L.; SUN, A. Deep Learning based Recommender System: A Survey and New Perspectives. v. 1, n. 1, p. 1–35, 2017. ISSN 15232867. Disponível em: <http://arxiv.org/abs/1707.07435>.

ZHANG, Y.-C.; BLATTNER, M.; YU, Y.-K. Publisher's Note: Heat Conduction Process on Community Networks as a Recommendation Model [Phys. Rev. Lett. 99, 154301 (2007)]. **Physical Review Letters**, v. 99, n. 16, p. 169902, 2007. ISSN 0031-9007. Disponível em: <http://link.aps.org/doi/10.1103/PhysRevLett.99.169902>.

ZHOU, T.; KUSCSIK, Z.; LIU, J.-G.; MEDO, M.; WAKELING, J. R.; ZHANG, Y.-C. Solving the apparent diversity-accuracy dilemma of recommender systems. In: . [S.l.: s.n.], 2010. v. 107, n. 10, p. 4511–4515. ISBN 0027-8424. ISSN 0027-8424.

ZHOU, T.; REN, J.; MEDO, M.; ZHANG, Y. C. Bipartite network projection and personal recommendation. **Physical Review E - Statistical, Nonlinear, and Soft Matter Physics**, v. 76, n. 4, p. 1–7, 2007. ISSN 15393755.

ZHOU, Y.; ZENG, A.; WANG, W. H. Temporal effects in trend prediction: Identifying the most popular nodes in the future. **PLoS ONE**, v. 10, n. 3, p. 1–10, 2015. ISSN 19326203.

ZIEGLER, C.-N. C.; MCNEE, S. M. S.; KONSTAN, J. a. J.; LAUSEN, G. Improving recommendation lists through topic diversification. In: **Proceedings of the 14th international conference on World Wide Web WWW 05**. Association for Computing Machinery, 2005. p. 22–32. ISBN 1595930469. ISSN 1874-3919. Disponível em: <https://dl.acm.org/doi/10.1145/1060745.1060754>.

ZUO, Y.; GONG, M.; ZENG, J.; MA, L.; JIAO, L. Personalized recommendation based on evolutionary multi-objective optimization [Research frontier]. **IEEE Computational Intelligence Magazine**, v. 10, n. 1, p. 52–62, 2015. ISSN 1556603X.

# APÊNDICE

A seguir são apresentados os detalhes do estudo estatístico feito em cada uma das bases de dados.

**Movielens**:

1. Formato em que encontram-se os atributos dos usuários

   I no dataset original:
   **age | gender | occupation | zip code**

   II no dataset após pré-processamento:
   **age | gender | occupation | country_state**

   III exemplo com dados:
   **28 M student US_MI**
   **61 M engineer US_VA**
   **42 M doctor US_KS**

2. Formato em que se encontram os relacionamentos entre usuários e itens

   I no dataset original:
   **user_id item_id rating timestamp**

   II no dataset após pré-processamento:
   **user_id item_id rating**

3. Detalhes de passos realizados durante o pré-processamento da base de dados no algoritmo $UPOD$

   I **são desconsiderados usuários com dados nulos ou vazios**

   II **o timestamp é removido dos relacionamentos**

   III **o “zip code” é convertido para o formato “country_state”**

4. Tamanho do domínio de cada um dos atributos presentes na base de dados

   I **age: 61 (intervalo entre 7 e 73 anos de idade)**

   II **gender: 2 (M ou F)**

   III **occupation: 21**

   IV **country_state: 55**

**Last.FM**:

1. Formato em que encontram-se os atributos dos usuários

   I no dataset original:
   **gender age country signup**

   II no dataset após pré-processamento:
   **gender age country**

   III exemplo com dados:
   **m 20 Czech_Republic**
   **m 19 United_States**
   **m 20 Australia**

2. Formato em que encontram-se os relacionamentos entre usuários e itens

   I no dataset original:
   **user_id item_id artist_name plays**

   II no dataset após pré-processamento:
   **user_id item_id rating**

3. Detalhes de passos realizados durante o pré-processamento da base de dados no algoritmo $UPOD$

   I **são desconsiderados usuários com dados nulos ou vazios**

   II **signup (horário em que usuário se cadastrou) foi desconsiderado**

III **plays (número de vezes que usuário reproduziu a música) foi convertido em ratings, isto foi feito partindo do princípio que um usuário reproduz mais vezes uma música que goste.**

IV **O intervalo do atributo "plays" é [1 – 135392] e ele foi convertido em "rating" com o intervalo [1 – 5]**

V **foram removidos espaços do atributo "country" e adicionado o caractere "_" em seu lugar**

VI **foram desconsideradas idades negativas e acima de 120 anos (existiam essas informações na base original)**

4. Tamanho do domínio de cada um dos atributos presentes na base de dados

   I **gender: 2 (m ou f)**

   II **age: 64 (intervalo entre 1 e 109 anos de idade)**

   III **country: 96**

**Book Crossing**:

1. Formato em que encontram-se os atributos dos usuários

   I no dataset original:
   **location age**

   II no dataset após pré-processamento:
   **mesmo formato**

   III exemplo com dados (dataset já limpo):
   **madrid_spain 31**
   **ontario_canada 43**
   **wisconsin_usa 19**

2. Formato em que encontram-se os relacionamentos entre usuários e itens

   I no dataset original:
   **user_id, item_id, rating**

II no dataset após pré-processamento:
**mesmo formato**

3. Detalhes de passos realizados durante o pré-processamento da base de dados no algoritmo $UPOD$

   I **são desconsiderados usuários com dados nulos ou vazios**

   II **são substituídos espaços em "location" por "_"**

   III **foram desconsideradas idades negativas e acima de 120 anos (existiam essas informações na base original)**

4. Tamanho do domínio de cada um dos atributos presentes na base de dados

   I **location: 465**

   II **age: 80 (intervalo entre 1 e 119 anos de idade)**