Arcabouço Genérico baseado em Técnicas de Agrupamento para Sistemas de Recomendação

Este exemplar corresponde à redação final da Dissertação devidamente corrigida e defendida por Ricardo Luís Zanetti Panaggio e aprovada pela Banca Examinadora.

Campinas, 01 de Outubro de 2010.

Ricardo Torres

Ricardo da Silva Torres (Orientador)

Dissertação apresentada ao Instituto de Computação, UNICAMP, como requisito parcial para a obtenção do título de Mestre em Ciência da Computação.

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA DO IMECC DA UNICAMP
Bibliotecária: Maria Fabiana Bezerra Müller – CRB8 / 6162

Panaggio, Ricardo Luís Zanetti

P191a Arcabouço genérico baseado em técnicas de agrupamento para sistemas de recomendação/Ricardo Luís Zanetti Panaggio-- Campinas, [S.P. : s.n.], 2010.

Orientador : Ricardo da Silva Torres.

Dissertação (mestrado) - Universidade Estadual de Campinas, Instituto de Computação.

1.Sistemas de recuperação da informação. 2.Recuperação da informação. 3.Aprendizado do computador - Técnicas. I. Torres, Ricardo da Silva. II. Universidade Estadual de Campinas. Instituto de Computação. III. Título.

Título em inglês: Cluster-based generic framework for recommender systems

Palavras-chave em inglês (Keywords): 1. Information storage and retrieval systems.
2. Information retrieval. 3. Machine learning – Technique.

Área de concentração: Sistemas de Recuperação da Informação

Titulação: Mestre em Ciência da Computação

Banca examinadora: Prof. Dr. Ricardo da Silva Torres
Prof. Dr. João Paulo Papa (Dco – UNESP)
Prof. Dr. Islene Calciolari Garcia (IC - UNICAMP)

Data da defesa: 01/10/2010

Programa de Pós-Graduação: Mestrado em Ciência da Computação

TERMO DE APROVAÇÃO

Dissertação Defendida e Aprovada em 01 de Outubro de 2010, pela Banca examinadora composta pelos Professores Doutores:

João Paulo Papa

Prof. Dr. João Paulo Papa
Departamento de Computação / Unesp-Bauru

Islene Garcia

Profª. Drª. Islene Calciolari Garcia
IC / UNICAMP

Ricardo Torres

Prof. Dr. Ricardo da Silva Torres
IC / UNICAMP

Instituto de Computação
Universidade Estadual de Campinas

Arcabouço Genérico baseado em Técnicas de Agrupamento para Sistemas de Recomendação

Ricardo Luís Zanetti Panaggio[1]

01 de Outubro de 2010

Banca Examinadora:

- Ricardo da Silva Torres (Orientador)
- João Paulo Papa
  Universidade Estadual Paulista "Júlio de Mesquita Filho" (Unesp)
- Islene Calciolari Garcia
  Universidade Estadual de Campinas (Unicamp)
- Gisele Lobo Pappa
  Universidade Federal de Minas Gerais (UFMG)
- Alexandre Xavier Falcão
  Universidade Estadual de Campinas (Unicamp)

[1]Suporte financeiro de: Bolsa do CNPq (processo 132022/2008-7), de 03/2008 a 02/2010

Resumo

A diferença entre o conjunto de dados disponíveis e o conjunto dos dados que interessam a um usuário é enorme e, em geral, cresce diariamente, uma vez que o volume de dados produzidos todos os dias só aumenta. Identificar todo o conjunto de dados de interesse de um usuário utilizando mecanismos tradicionais é muito difícil – talvez impossível. Nesse cenário, ferramentas que possam ajudar usuários a identificar itens de interesse, como sistemas de recomendação, têm um grande valor.

Esta dissertação apresenta um modelo genérico que pode ser utilizado para a criação de sistemas de recomendação, e sua instanciação utilizando técnicas de agrupamento. Essa dissertação apresenta também a validação desse modelo, a partir de sua implementação e experimentação com dados das bases Movielens e Jester.

As principais contribuições são: definição de um modelo de recomendação baseado em grafos, até onde se sabe mais rico e mais genérico que os encontrados na literatura; especificação e implementação de uma arquitetura modular de um sistema de recomendação baseada nesse modelo, com enfoque em técnicas de agrupamento de dados; validação da arquitetura e do modelo de recomendação propostos, comparando eficácia e eficiência de técnicas de agrupamento de dados em sistemas de recomendação.

Abstract

The difference between the data available and the set of interesting data to a certain user is enormous and, in general, is becoming greater daily, as the amount of data produced increases. Identifying all the interesting data set using traditional mechanisms is difficult – sometimes impossible. In this scenario, providing tools that can help users on identifying items that are of interest, such as recommendation systems, is of great importance.

This dissertation presents a generic model that can be used to create recommender systems, and its instantiation using clustering techniques. It also discusses the validation of this model, by showing results obtained from experiments with data from Movielens and Jester datasets.

The main contributions are: a graph-based generic model for recommender systems, which is more generic and richer than the ones found in literature; the specification and implementation of a modular architecture for recommender systems based on that model, focused on clustering techniques; validation of both model and architecture, by comparing efficiency and effectiveness of clustering-based recommender systems.

Agradecimentos

Gostaria de agradecer a todos que contribuíram para a realização deste trabalho. Impossível fazê-lo sem cometer injustiça, já que muitas pessoas contribuíram. Eu sei que fui injusto, porque não seria possível não sê-lo, então só me resta pedir desculpa, antes de mais nada. Garanto que me esforcei para diminuir ao máximo a injustiça.

Agradeço ao meu orientador pela atenção e apoio durante o desenvolvimento deste trabalho, assim como pelo meu crescimento intelectual e acadêmico. Agradeço também pelo apoio nos momentos difíceis, que não foram poucos. Agradeço a ele, por fim, por dar-me liberdade para desenvolver consideráveis partes desse trabalho, principalmente as que não deram certo. Aprendi muito assim, mais do que teria aprendido indo apenas pelo caminho certo.

Agradeço a todos os professores do Instituto de Computação com quem tive contato em aulas, em conversas e discussões nos corredores ou na sala do café. Cada momento junto com tais incríveis pesquisadores foram de total aprendizado, de coisas simples da vida até o estado da arte da Ciência. Não há, tenho certeza disso, melhor lugar para se estar do que na presença dessas pessoas.

Agradeço em especial à Professora Claudia, por toda a ajuda e apoio durante o período que fiz parte do LIS, e especialmente pelo apoio após minha saída. Seus ensinamentos como mentora do laboratório e de todos nós que pertencíamos a ele não são apenas importantes: foram decisivos em diversos momentos desde que foram passados a nós até agora, e tenho certeza que serão ainda mais importantes durante o resto da minha vida.

Agradeço de forma especial também à Professora Islene, que me ajudou muito em minha passagem de quase três anos no Instituto. Não há professor, com exceção do Professor Torres, que tenha conversado mais tempo comigo. Nossas conversas no corredor, após os Seminários de Software Livre e em tantas outras oportunidades foram sempre apreciadas por mim, e sempre tiveram um impacto positivo na minha forma de ver meu próprio trabalho e minha contribuição para a Ciência.

Agradeço ao (agora Professor) Papa, pela ajuda em desvendar o método de agrupamento de dados que ofereceu os melhores resultados desse trabalho, o OPF. Sem sua ajuda, eu não seria capaz de atingir esses resultados. E sem eles, essa dissertação teria

um valor bem menor.

Agradeço à minha família pelo apoio nas minhas decisões, mesmo e principalmente nas menos acertadas. A ajuda de todos foi muito importante, principalmente do meu pai, Paulo, minha mãe, Viviane, minha irmã, Bruna, minha avó, Ivone, e meus tios Veridiana, Isaías, Maria e Jacob. Um agradecimento extra à minha avó que me ajudou a me manter no caríssimo distrito de Barão Geraldo, permitindo que eu pudesse ter uma qualidade de vida muito melhor, o que com certeza se refletiu na qualidade do trabalho desenvolvido.

Em especial, à minha namorada, Thanuci, por dedicar parte considerável de seu tempo a nós. Agradeço também por aceitar dividir esse pouco tempo com meus infindos finais de semana de trabalho. Não poderia haver companheira mais fiel, paciente e compreensiva nesses momentos. Cada segundo compartilhado comigo foi apreciado ao extremo, tornando minha vida e meu trabalho menos árduos e mais prazerosos.

Agradeço aos colegas que contribuíram com ideias, críticas, sugestões e também, talvez até principalmente, pelos poucos momentos de lazer que tivemos juntos. A todos os colegas dos laboratórios que fiz parte, LIS e Recod, dos outros laboratórios do Instituto, de outros Institutos da Unicamp, os que antes da minha vinda a Campinas e os de toda a vida: vocês fizeram parte de cada etapa desse trabalho. Essa ajuda, mesmo que indireta, jamais será esquecida. Só me nego a listar seus nomes porque seria uma tremenda injustiça esquecer de um amigo sequer. Durante meses compilei essa lista de nomes, e a cada vez que e lia, encontrava novos nomes. Não sou capaz de listá-los, mas garanto que minha gratidão a todos é muito grande.

Gostaria de agradecer a todos os os pesquisadores, que dedicam diariamente seu tempo à Ciência. Graças a todos avançamos diariamente nosso conhecimento. Fico feliz de fazer pesquisa e trabalhar junto desses gigantes, para expandir ainda mais o nosso conhecimento. Esse trabalho só foi possível porque estava apoiado nos ombros de gigantes. Não há outra forma de fazer Ciência. E não há como não agradecer aos que trabalharam antes de mim.

Gostaria de agradecer da mesma forma à comunidade de Software Livre, por acreditar que a liberdade é algo importante e se manter firme nisso. Graças a essa grande comunidade, composta de indivíduos que dividem comigo essa vontade de ser livre, que pude implementar e executar todo o arcabouço criado. Em especial, à comunidade Ruby, por me ajudar em todos os problemas que encontrei e por me aceitar como membro.

Gostaria por fim de agradecer a todo o apoio financeiro recebido por mim, pelo laboratório, pelo Instituto e pela Universidade. Ao CNPq (processo 132002/2008-7), agradeço pela bolsa concedida. À comunidade Ruby, pela bolsa do Rubysoc, que não poderia ter chegado em melhor hora, e que tornou a conclusão desse trabalho possível. Além disso, gostaria de agradecer às agências todas de fomento, CNPq, FAPESP e CAPES, por financiar projetos dos laboratórios, do Instituto e da Universidade.

Sumário

Lista de Tabelas

Lista de Figuras

Capítulo 1

Introdução

Diariamente usuários são expostos a uma quantidade imensa de dados, e em geral não são capazes de absorver nem mesmo uma parte muito pequena deles. Manter-se informado sobre as modificações e adições destes conteúdos torna-se uma tarefa cada vez mais árdua, e o tempo perdido filtrando o que é de interesse só aumenta por conta disso.

Diversas técnicas e metodologias, em várias áreas, vêm sendo desenvolvidas para solucionar – ou ao menos atenuar – os problemas gerados pela existência de um grande volume de dados em sistemas de recuperação de informação: métodos de busca, uso de semântica e metadados e serviços de recomendação.

Recomendação é essencial nos domínios em que há volumes muito grandes de informação, uma vez que analisar todo o domínio pode ser muito custoso e muitas vezes inviável [18]. Nos termos formais utilizados por Gonçalves em seu *framework* 5S [13], pode-se definir recomendação como a identificação de um subconjunto de interesse de uma dada coleção digital que seja do interesse de um dado usuário a partir do conjunto de avaliações dos objetos dessa coleção, produzido pelo próprio usuário ou por outros.

Portanto, um serviço de recomendação é um componente de software que, baseado em avaliações dos objetos de uma coleção digital, produz um subconjunto da coleção que potencialmente seja do interesse de um usuário específico.

Um sistema de recomendação tem como principal função auxiliar um usuário na escolha de itens apropriados de uma grande coleção digital [19], que pode ser composta de diversos tipos de documentos digitais, como fotos, vídeos, artigos e livros. Assim, sistemas de recomendação agregam valor a aplicações, pois colaboram para que os usuários se mantenham atualizados em relação às coleções digitais com as quais têm contato.

Os sistemas de recomendação são comumente divididos em dois grupos: filtragem baseada em conteúdo (*Content-Based Filtering*) [2, 10, 44, 45, 48] e filtragem colaborativa (*Collaborative-Filtering*) [3,14,16,26,46]. Existem, no entanto, sistemas que combinam as técnicas, explorando características positivas de ambas, gerando técnicas híbridas [5, 36,

39]. Filtragem colaborativa consiste em recomendar itens a um usuário baseando-se nas avaliações efetuadas por usuários semelhantes. Já as técnicas de filtragem por conteúdo utilizam itens semelhantes aos avaliados por um usuário para fazer recomendações. Técnicas híbridas tentam compor essas duas vertentes para melhorar a eficácia das recomendações.

Essas três formas de se recomendar são claramente distintas, mas em essência possuem muito elementos semelhantes. Poucos trabalhos tentam explorar essa visão, e em consequência não foram encontrados muitos modelos de sistemas de recomendação que se adaptem facilmente a várias necessidades diferentes. Com um modelo genérico e bem definido, é possível criar sistemas de recomendação com maior facilidade. Pode-se ainda utilizar o modelo para estruturar sistemas existentes e futuros, tornando a sua comparação mais objetiva. Para suprir essa necessidade, esse trabalho propõe um modelo genérico para sistemas de recomendação.

Há várias maneiras de se implementar sistemas de recomendação, e uma técnica comum é o uso de agrupamento da dados. Algumas vantagens são claras no uso de agrupamento de dados em sistemas de recomendação, como a redução da quantidade de elementos a serem tratados e a atenuação de *outliers* e ruídos. Para validar o modelo proposto, foi implementado um sistema de recomendação baseada em agrupamento de dados.

As principais contribuições desta dissertação são:

- definição de um modelo de recomendação baseado em grafos, até onde se sabe mais rico e mais genérico que os encontrados na literatura;
- especificação e implementação de uma arquitetura modular de um sistema de recomendação baseada nesse modelo;
- implementação de um sistema de recomendação baseado nessa arquitetura, com enfoque em técnicas de agrupamento de dados;
- validação da arquitetura e do modelo de recomendação propostos;
- comparação da eficácia e da eficiência de técnicas de agrupamento de dados para sistemas de recomendação.

O arcabouço proposto é mais rico e genérico que os encontrados na literatura em alguns aspectos:

- O arcabouço não foi construído para um determinado tipo de dado, e assim pode ser utilizado para resolver problemas de recomendação com dados de origens diversas;
- O arcabouço engloba todas as técnicas de recomendação conhecidas (filtragem colaborativa, filtragem por conteúdo e abordagens híbridas), de forma que é possível

implementar qualquer uma das técnicas, ou uma combinação delas, de maneira trivial;

- Parte do arcabouço descrito é uma especialização para utilizar agrupamento de dados como base para fazer recomendações; no entanto, é possível fazer novas especializações, ou adaptar as existentes, para utilizar outros mecanismos e métodos para fazer recomendações;

- Em geral, o tratamento de itens e usuários é diferente em todos os modelos encontrados na literatura; com o modelo proposto, o tratamento para as duas entidades centrais de sistemas de recomendação, e eventualmente outras entidades necessárias em algum domínio específico, é o mesmo, podendo ser diferenciado se necessário;

- O modelo proposto não se limita, em nenhum aspecto, a natureza, origem, tipo, forma, estrutura ou qualquer outro aspecto ligado aos dados e suas ligações, assim é possível utilizá-lo para diversos tipos diferentes de dados, em diversas circunstâncias;

- Os algoritmos propostos são apenas estruturações lógicas para tratamento dos dados para recomendação, recebendo como parâmetro funções que fazem o tratamento dos dados e realmente germ as recomendações; definindo de maneiras diversas tais funções, é possível criar muitos recomendadores diferentes.

O restante deste documento é organizado da seguinte maneira: o próximo Capítulo apresenta os trabalhos correlatos; o Capítulo 3 apresenta o novo modelo de dados para sistemas de recomendação; a arquitetura proposta para a construção de sistemas de recomendação é apresentada no Capítulo 4; o Capítulo 5 apresenta os resultados obtidos no trabalho; e por fim o Capítulo 6 apresenta as conclusões do trabalho, possíveis extensões e trabalhos futuros.

Capítulo 2

Trabalhos relacionados

Neste capítulo são apresentados trabalhos existentes na literatura que têm intersecção com o modelo proposto. Inicialmente, será apresentada formalmente a definição de recomendação que será utilizada e as técnicas de recomendações listadas na literatura. Será então apresentada a definição formal de contexto e de outros sistemas de recomendação que se utilizam de informações dessa natureza para gerar recomendações. Por fim, serão apresentados trabalhos em sistemas de recomendação que se utilizam de agrupamento de dados e uma técnica do estado da arte para agrupamento de dados que será utilizada no trabalho, o algoritmo Floresta de Caminhos Ótimos.

2.1 Recomendação

Recomendar pode ser definido de maneira simples como aconselhar, indicar, dar sugestões a respeito de algo. Assim, dado um domínio que possua uma grande quantidade de informação associada, recomendação passa a ser algo essencial, uma vez que analisar todo o domínio pode ser muito custoso e muitas vezes inviável [18].

Formalmente, no seu *framework* 5S [13], no contexto de Bibliotecas Digitais, Gonçalves define recomendação como: "dada uma coleção, um ator e um conjunto de avaliações – produzido pelo próprio ou por outros atores – dos objetos dessa coleção, [deseja-se] produzir um subconjunto de interesse da coleção para esse ator particular".

Seguindo essa definição, um serviço de recomendação é um componente de software que, baseado em avaliações dos objetos de uma coleção digital, encontra um subconjunto de objetos da coleção que potencialmente sejam do interesse de um usuário específico.

Um sistema de recomendação auxilia um usuário na escolha de itens apropriados de uma grande coleção digital [19], que pode ser composta de diversos tipos de documentos digitais, como fotos, vídeos, artigos e livros. Para isso, são identificados itens relevantes de uma dada coleção digital, utilizando similaridade entre itens e usuários e informações

sobre o interesse do usuário baseando-se em dados de seu histórico e comportamento [20]. O objetivo de sistemas de recomendação é, portanto, prover a um usuário específico itens que sejam do seu interesse [19].

Sistemas de recomendação se tornaram uma área de grande relevância a partir do surgimento dos primeiros trabalhos na área, na década de 90. Desde então, tanto a indústria quanto a academia realizam esforços para melhorar as abordagens existentes e desenvolver novas abordagens [1]. Ainda há grande interesse na área pois existe abundância de aplicações práticas que auxiliam o usuário a lidar com sobrecarga de informação [1].

As técnicas utilizadas em sistemas de recomendação podem ser classificadas em dois grandes grupos: filtragem colaborativa (*collaborative filtering*) [3,14,16,26,46] e filtragem baseada em conteúdo (*content-based filtering*) [2,10,44,45,48]. Existem também técnicas híbridas [5, 20, 37, 41, 47], que tentam explorar os pontos fortes das duas técnicas para aumentar a eficácia e eficiência. Detalhes sobre cada uma das técnicas são apresentados nas próximas seções.

2.2 Filtragem colaborativa

Filtragem colaborativa (*collaborative filtering*) [3, 14, 16, 26, 46] é uma técnica de recomendação baseada na suposição de que usuários que tiveram interesse similar no passado, potencialmente terão interesse semelhante no futuro [34]. Assim, para um dado usuário, o sistema de recomendação identifica outros usuários com interesse semelhante e recomenda itens que esses usuários tiveram interesse.

Há algumas maneiras de se classificar técnicas de filtragem colaborativa. Uma delas é a categorização quanto a explicitação das relações itens/usuários. Relações explícitas são aquelas em que o usuário expressa conscientemente sua preferência por um dado item, geralmente em uma escala numérica discreta. Típicas relações explícitas são atribuição de notas a itens, ou identificação de resultados como positivos ou negativos. Relações implícitas são as que o interesse é extraído do comportamento ou interação do usuário com os itens. O comportamento é geralmente extraído de dados de histórico dos acessos do usuário feito a itens da coleção, como ouvir uma música, baixar um artigo ou comprar um livro [3].

Técnicas de filtragem colaborativa podem ser divididas também em abordagens baseadas em modelo e baseadas em memória. Abordagens baseadas em modelo inferem um modelo descritivo do interesse dos usuários e utiliza este modelo para fazer recomendações [46]. Ou seja, assume-se que itens/usuários podem ser agrupados utilizando as informações de interesse. O esforço de técnicas desse tipo é encontrar grupos de interesse, que são utilizados para recomendação. Abordagens baseadas em memória, por outro lado, utilizam itens de usuários com interesse similar para fazer as recomendações [46].

Ou seja, assume-se que usuários que se relacionam com itens similares são boas fontes de novos itens a serem recomendados.

Pela definição de recomendação apresentada anteriormente, o ator é um usuário específico e o conjunto de atribuições de valores aos objetos da coleção são as relações itens/usuários, sejam explícitas ou implícitas. As recomendações geradas a partir dos modelos ou dos dados brutos dessas relações são o subconjunto de interesse do usuário.

Essa abordagem possui alguns problemas clássicos:

- Novo usuário: quando um usuário entra no sistema, ele não avaliou nenhum item ainda; logo, não é possível fazer um casamento entre o perfil desse usuário com os demais;
- Novo item: quando um novo item é inserido no sistema, ele ainda não foi avaliado por nenhum usuário, e então não há informação suficiente para que ele seja recomendado;
- Ovelha Negra: alguns usuários podem ter um comportamento oposto à maioria, tanto por ter opinião diversa como intensionalmente, para atrapalhar a execução do recomendador.

2.3 Filtragem baseada em conteúdo

Filtragem baseada em conteúdo (*content-based filtering*) [2,10,44,45,48] é uma técnica de recomendação baseada na similaridade do conteúdo dos itens [9]. A idéia é que um usuário que teve interesse por um determinado item no passado provavelmente terá interesse por itens semelhantes no futuro. Dessa forma, o sistema de recomendação identifica outros itens parecidos com os que o usuário já demonstrou interesse e os recomenda.

Formalizando filtragem baseada em conteúdo utilizando a definição de recomendação apresentada, o ator é um usuário do sistema de recomendação, o conjunto de atribuições de valores aos objetos das coleções é uma função da similaridade dos itens não relacionados com o usuário para os itens relacionados. Como resultado, o subconjunto de interesse do usuário é a recomendação produzida.

Essa abordagem também possui o problema do novo usuário, pelo mesmo motivo que a filtragem colaborativa. Porém não é sucetível ao problema da ovelha negra, já que não leva em consideração as avaliações entre usuários para recomendar. É também imune ao problema de novo item, pois a similaridade entre itens é medida com relação ao conteúdo dos itens, e não com relação às avaliações feitas e eles.

2.4 Técnicas híbridas

Diversos sistemas de recomendação utilizam uma abordagem híbrida, combinando filtragem colaborativa e filtragem baseada em conteúdo para evitar, ou ao menos diminuir, os problemas das duas técnicas [1, 5, 20, 37, 41, 47].

Existem algumas maneiras de se combinar filtragem colaborativa e baseada em conteúdo, que podem ser classificadas como [1]:

- Implementação das duas técnicas e combinação dos resultados;
- Incorporar algumas características de filtragem baseada em conteúdo em filtragem colaborativa;
- Incorporar algumas características de filtragem colaborativa em filtragem baseada em conteúdo;
- Construção de um modelo que incorpora as duas técnicas.

2.5 Ferramentas para recomendação

Com o uso cada vez maior de sistemas de recomendação, foi necessário criar sistemas em que algoritmos e técnicas de recomendação pudesse ser utilizados por diversas aplicações clientes e em tecnologias e domínios diferentes [32, 33].

Ferramenta	Técnica	Acesso
CoFE	Colaborativa	Java
Duine	**Várias**	Java
EasyUtil	Colaborativa	HTTP
Taste	Colaborativa	**Serviços Web**
MobLife Recommender	Colaborativa	**Serviços Web**
RecS-DL	**Várias**	**Serviços Web**

Tabela 2.1: Ferramentas para recomendação

Em [32] foram analisadas várias ferramentas de infra-estrutura para sistemas de recomendação. A Tabela 2.1 apresenta algumas das ferramentas analisadas por Pedronette e algumas das suas características. É possível observar que algumas ferramentas são independentesde técnica de recomendação e tecnologia de acesso, mas apenas o arcabouço RecS-DL é independentes nos dois aspectos.

Para implementar um sistema de recomendação de maneira interoperável, Pedronete [32], em seu trabalho de Mestrado realizado no Instituto de Computação da UNICAMP, criou a plataforma de serviços de recomendação RecS-DL. Ela foi proposta e implementada para ser independente de domínio de aplicação, de tecnonologia e de técnicas de recomendação, fazendo uso para tal de Serviços Web e de uma arquitetura bastante modular. Além disso, a plataforma foi aplicada e validada no domínio de bibliotecas digitais [31–33].

2.6 Contexto em sistemas de recomendação

Dey [11] define contexto como qualquer informação que pode ser utilizada para caracterizar a situação de qualquer pessoa, local ou objeto que é considerado relevante para a interação entre um usuário e uma aplicação.

Dessa maneira, pode-se modelar contexto na esfera de sistemas de recomendação como o conjunto de dados e metadados que são associados às interações do usuário com itens do sistema. Exemplos de informações desse tipo são tempo (dia da semana, horário) e localização (IP, evidências geográficas).

Dey [11] declara também que um sistema é sensível a contexto se utiliza contexto para prover informações e/ou serviços relevantes a um usuário, onde a relevância depende das tarefas do usuário. Tal objetivo coincide com os objetivos de um sistema de recomendação [42]. Assim, como o objetivo de um sistema de recomendação é fornecer conteúdo relevante ao usuário (Seção 2.1), tornar um sistema de recomendação (mais) sensível a contexto pode trazer benefícios como melhora de eficácia.

Sistemas de recomendação tradicionais geralmente operam sobre as relações entre usuários e itens, mapeando as informações dos itens, usuários e suas relações para algum espaço que represente a opinião de um usuário sobre um dado item [43]. No entanto, alguns trabalhos mostram que uma quantidade adicional de informações, referentes a contexto, pode ser utilizada para melhorar o resultado em sistemas de recomendação [4,24,38,42,43]. Isso se dá pois o interesse de um usuário e as avaliações que faz estão diretamente ligados a situação em que o usuário se encontra. Assim, conhecer o contexto do usuário durante a utilização do sistema permite ter uma informação de maior qualidade a respeito dos interesses do usuário, permitindo um aumento na eficácia das recomendações.

Contexto dos itens e do usuário raramente é utilizado em sistema de recomendação [43]. O uso de contexto em sistemas desse tipo é recente e ainda pouco explorado. Porém, alguns trabalhos mostram que diversas formas de contexto – social [38, 43], pessoal [24, 38, 42], físico [4, 42, 43], estrutural [17]– podem ser utilizadas para melhorar o resultado em sistemas de recomendação. Que contexto utilizar, modelar e mensurar é altamente dependente do domínio da aplicação [43].

Com a adição de contexto a sistemas de recomendação, tem-se o aumento da complexidade [43], uma vez que o volume e a natureza dos dados a serem tratados é muito mais diversa e extensa. No entanto, a adição de contexto a esses sistemas se mostra necessária pois, em algumas situações, a utilidade de certo item a um usuário depende significativamente do tempo ou do local onde o usuário se encontra [1].

Para realizar recomendações são necessárias informações a respeito dos interesses do usuário e das avaliações dos objetos da coleção digital. Entretanto, um fato importante é que o interesse de um usuário e suas avaliações estão diretamente ligados à situação em que o usuário se encontra. Dessa maneira, conhecer o contexto do usuário durante a utilização do sistema permite ter uma informação de maior qualidade a respeito dos seus interesses, permitindo um aumento na eficácia das recomendações.

Sistemas de recomendação tradicionalmente não se utilizam de dados sobre o contexto do usuário e da avaliação dos itens para efetuar recomendações [43]. No entanto, alguns trabalhos mostram que diversas formas de contexto podem ser utilizadas para melhorar o resultado em sistemas de recomendação [4, 24, 38, 42, 43]. A adição de contexto a sistemas de recomendação se mostra necessária pois, em algumas situações, a utilidade de certo item a um usuário depende significativamente do tempo ou do local onde o usuário se encontra [1].

Além de informações relacionadas ao contexto do usuário, muitas vezes a escolha de um item em uma biblioteca digital é direcionada por citações em itens já acessados. Em outro momento, a escolha pode ser feita por um novo item introduzido na biblioteca ter sido escrito por um autor já existente na coleção de itens acessados por um dado usuário. Além disso, o usuário pode se interessar por itens que tenham sido publicados por membros de um mesmo grupo de pesquisa ou de uma mesma instituição. Por isso, utilizar metadados de objetos das coleções digitais, em especial de itens de Bibliotecas Digitais, é algo essencial para uma melhor representação das relações entre os objetos da coleção, permitindo uma medição mais precisa da similaridade entre eles [1]. Trabalhos recentes como [49] focam a utilização desse tipo de metadados para aumentar a eficiência e eficácia de sistemas de recomendação, mas a similaridade entre os itens medida a partir desses metadados são calculadas de maneira bastante específica, dificultando a extensão para novos tipos de metadados identificados.

2.7 Agrupamento de dados

Informalmente, agrupamento de dados, também conhecido como clusterização e análise de agrupamentos/*clusters*, é o ato de descobrir agrupamentos naturais em um conjunto de dados.

Jain [21] oferece uma definição mais formal e precisa para agrupamento de dados:

dada a *representação* de um dado conjunto de n objetos, encontrar k grupos baseados em uma medida de similaridade entre os as representações dos objetos de forma que as similaridades entre objetos do mesmo grupo sejam grandes e de objetos de grupos distintos seja baixa.

Os métodos de agrupamento de dados são divididos em dois tipos: hierárquicos e particionais. Os dois tipos são explicados nas próximas seções.

2.7.1 Agrupamento hierárquico

Um método de agrupamento hierárquico cria uma hierarquia de *clusters* representada por uma estrutura chamada de dendrograma. Essa estrutura é uma árvore onde a raiz corresponde a um único *cluster* contendo todos os objetos e as folhas correspondem a *clusters* que consistem cada um dos objetos invididualmente. Nós intermediários da árvore representam *clusters* com os elementos dos quais é pai.

Existem dois tipos de métodos de cluterização hierárquica: métodos aglomerativos e métodos divisivos. Métodos aglomerativos iniciam das folhas, e vão formando a árvore a cada iteração, juntando nós já existentes em nós intermediários até que reste apenas um nó. Métodos divisivos iniciam da raiz, contendo todos os elementos, e seguem subdividindo o nó em subárvores, até que se chegue a todas as folhas.

A partir do dendrograma, é possível gerar um conjunto de *clusters* fazendo um corte na árvore na altura de um valor específico de similaridade, onde os *clusters* formados seriam os nós que representassem as raízes dessas subárvores. Para conseguir um número k de *clusters*, pode-se procurar por um valor de similaridade que ofereça k *clusters* no corte.

Segundo [22], a maioria dos algoritmos aglomerativos são variantes do *Single-Link* [] e *Complete-Link* [], os algoritmos aglomerativos mais comuns. Esses dois algoritmos são iguais na estrutura, tendo como única diferença a maneira com que os *clusters* são aglomerados.

O método inicia com todos os objetos representando um *cluster* cada um. Em seguida, o algoritmo passa a agrupar o par de *clusters* que for mais próximo, até que reste apenas um *cluster*.

Essa medida de proximidade entre *clusters* pode ser implementada de várias maneiras, e é isso que diferencia o *Single-Link* e *Complete-Link*: no primeiro, das distâncias dos elementos dos dois *clusters*, a utilizada é a menor, enquando no segundo, a maior. Há também outras medidas, como o *Average-Link*, que usa a distância média.

Métodos divisivos iniciam com os objetos formando um único *cluster*. A cada iteração, o algoritmo divide um dos *cluster* em outros menores, de forma que a distância entre os *clusters* formados seja a maior possível, até que todos os objetos formem um *cluster* cada

um. Uma das medidas clássicas para calcular a dissimilaridade em algoritmos divisivos é apresentada em [23].

2.7.2 Agrupamento particional

Um método de agrupamento particional encontra uma partição dos objetos (ou do espaço dos objetos) ao invés de criar uma estrutura de *clusters*.

O algoritmo de agrupamento particional mais usado é o *k-means*, por ser um algoritmo muito simples. Ele inicia com um particionamento randômico e rearranjando o particionamento de acordo com a similaridade entre os itens e os centros dos *clusters* até que algum critério de parada seja atingido.

Um algoritmo particional desenvolvido recentemente no IC/Unicamp tem gerado resultados interessantes em diversos domínios. Tal algoritmo é apresentado a seguir.

Floresta de Caminhos Ótimos

O algoritmo Floresta de Caminhos Ótimos (do inglês *Optimum Path Forest* – OPF) é um algoritmo de classificação supervisionada [27–30, 40] e não-supervisionada (ou agrupamento de dados) [8, 27, 35]. OPF é relevante para classificação e agrupamento de dados pois é rápido, simples, multiclasse, independente de parâmetros e consegue lidar com certo grau de separabilidade entre classes ou agrupamentos [29].

O algoritmo OPF tem como entrada um grafo $G_{OPF} = (V_{OPF}, A_{OPF}, \Psi_{OPF})$, onde V_{OPF} é um conjunto de vértices que representa pontos em algum espaço de características, A_{OPF} é o conjunto de arestas que conecta cada vértice $v_{OPF} \in V_{OPF}$ aos seus n vizinhos mais próximos e Ψ_{OPF} é a função que mapeia cada aresta $a_{OPF} \in A_{OPF}$ ao par não ordenado (v'_{OPF}, v''_{OPF}) de vértices de V_{OPF}. n é um parâmetro do algoritmo, que define a quantidade de vizinhos mais próximos utilizados na adjacência A, e pode ser calculado otimamente [8].

As arestas a_{OPF} do grafo são ponderadas por uma função de distância $d : V_{OPF} \times V_{OPF} \mapsto \mathbf{R}$. Os vértices v_{OPF} do grafo são ponderados por valores de densidade, formando uma função discreta de distribuição de probabilidade $\rho : V_{OPF} \mapsto \mathbf{R}$ (Equação 2.1):

$$\rho(v) = \frac{1}{\sqrt{2\pi\sigma^2}A(v)} \sum_{w \in A(v)} e^{-\frac{d^2(v,w)}{2\sigma^2}} \tag{2.1}$$

onde A é a função de adjacência que mapeia cada vértice v de V_{OPF} para os seus n vizinhos mais próximos, $\sigma = \frac{d_f}{3}$ e d_f é o maior peso dentre as arestas do grafo G_{OPF}.

Agrupamentos são encontrados pela maximização de uma função de valor de caminhos ótimos ρ, que gera uma árvore de caminhos ótimos [8]. Tal árvore tem como raiz um máximo da função de distribuição de probabilidade descrita e é formada pelos caminhos

que têm raiz no máximo da função ρ e valor de densidade mínimo. Cada ponto de máximo define uma zona de influência que agrupa os nós que estão mais fortemente conectados a ele que a qualquer outra zona de influência de outro ponto de máximo [8]. Formalmente, se um caminho no grafo que vai de v a w é denotado por $\pi_{v,w}$, deseja-se maximizar $f(\pi_{v,w})$ para todo $v \in V_{OPF}$, onde

$$f(\pi_{v,v}) = \begin{cases} \rho(t) & \text{se } t \in R \\ \rho(t) - \delta & \text{caso contrário} \end{cases} \tag{2.2}$$

$$f(\pi_{r,w}) = \min\{f(\pi_{r,v}), \rho(w)\} \tag{2.3}$$

onde $\pi_{v,v}$ é um caminho trivial (formado por um único nó), R é o conjunto de raízes formado pelos máximos de ρ , e

$$\delta = \min_{(v,w)\in A|\rho(w)\neq\rho(v)} |\rho(w) - \rho(v)| \tag{2.4}$$

Quanto maior δ, menor o número de raízes $|R|$. O algoritmo maximiza $f(\pi_{r,w})$ de forma que os caminhos ótimos formem uma árvore de caminhos ótimos. Assim, a saída do algoritmo é um mapa de predecessores P sem ciclos que descreve, para cada w fora de R, o seu predecessor $P(w)$ na árvore de caminhos ótimos ou *nil* se $w \in R$.

O pseudocódigo do OPF para agrupamento de dados é apresentado a seguir.

Algoritmo *Agrupamento usando Floresta de Caminhos Ótimos*
Entrada: Grafo $(V_{OPF}, A_{OPF}, \Psi_{OPF})$ e função ρ
Saída: Mapa de rótulos L, mapa de valores de caminhos V, floresta P
($*$ Auxiliares: Fila de prioridade Q de $v \in V_{OPF}$, ordenada por $V(v)$ $*$)
1. $l \leftarrow 1$
2. **para cada** v em V_{OPF}
3. **faça** $P(v) \leftarrow$**nil**
4. $V(v) \leftarrow \rho(v) - \delta$
5. $Q.insere(v)$
6. **enquanto** Q não estiver vazia
7. **faça** $Q.remove(v)$
8. **se** $P(v) =$ **nil**
9. **então** $L(v) \leftarrow l$
10. $l \leftarrow l + 1$
11. $V(v) \leftarrow \rho(v)$
12. **para cada** t em A
13. **faça se** $V(w) < V(v)$
14. **então** $tmp \leftarrow \min\{V(v), \rho(w)\}$

15. **se** $tmp > V(w)$
16. **então** $L(w) \leftarrow L(v)$
17. $P(w) \leftarrow v$
18. $V(w) \leftarrow tmp$
19. $Q.atualiza(w)$
20.

Capítulo 3

Modelo Genérico de Recomendação

Sistemas de recomendação em geral são construídos de maneira *ad-hoc*: com foco em resolver o problema de recomendação de um nicho específico, na maioria das vezes de forma não generalizável e não adaptável.

Não foram encontrados trabalhos na literatura que tenham como foco principal a criação de um modelo genérico para sistemas de recomendação que possa ser usados em qualquer domínio e que seja extensível.

Neste capítulo, um novo modelo de dados para recomendação é proposto, com o objetivo de oferecer uma base genérica para a criação de sistemas de recomendação.

3.1 Modelo de recomendação – Grafo Base

Dados os interesses deste trabalho e o domínio de aplicação apresentados no Capítulo 2, será apresentado um modelo de dados na forma de grafo que constitui a base deste trabalho. O modelo foi definido de maneira genérica e extensível, de modo que se possa aplicá-lo a novos domínios com pouca ou nenhuma alteração.

Nesse modelo, são utilizados grafos para a definição dos elementos centrais (vértices) de um sistema de recomendação e as relações entre tais elementos (arestas).

Como já apresentado na Seção 2.1, um sistema de recomendação possui como elementos principais usuários e itens do sistema de informação que serão alvo de recomendação. Assim, sejam U o conjunto que representa os usuários e I o conjunto que representa os itens do sistema de recomendação.

A partir de U e I, definem-se dois conjuntos de vértices para o modelo de recomendação:

- V_U é um conjunto de nós v_u que representa cada um dos usuários u de U. Para cada u de U existe um correspondente v_u em V_U.

- V_I é um conjunto de nós v_i que representa cada um dos itens i de I. Para cada i de I existe um correspondente v_i em V_I.

Para efetuar o mapeamento de U para V_U é utilizada a função $f_{V_U} : U \mapsto V_U$. O mapeamento inverso é feito através da função inversa de f_{V_U}, $f_{V_U}^{-1}$. O mapeamento de I para V_I é feito de maneira análoga.

Outro ponto muito importante são as relações entre usuários e itens, sobre as quais diversas técnicas de recomendação se apóiam.

Uma dada relação entre usuários e itens é representada pelo conjunto R_{UI_j}, onde $R_{UI_j} = \{r_i | r_i \in U \times I, i \in \mathbf{N}\}$. Uma relação entre itens é representada pelo conjunto R_{II_j}, onde $R_{II_j} = \{r_i | r_i \in I \times I, i \in \mathbf{N}\}$. Por fim, uma relação entre usuários é representada pelo conjunto R_{UU_j}, $R_{UU_j} = \{r_i | r_i \in U \times U, i \in \mathbf{N}\}$.

Como pode existir uma gama variada de relações do mesmo tipo (similaridade entre itens e entre usuários pode ser medida de diversas maneiras, usuários podem se relacionar com itens e várias maneiras diferentes), há a necessidade de representarmos diversas relações diferentes, dentro do mesmo tipo de relação.

Assim, dados os n_{UI} conjuntos de relações entre usuários e itens R_{UI_j} existentes em um sistema, $0 \leq j < n_{UI}$, temos o conjunto $R_{UI} = \bigcup_{\forall j} R_{UI_j}$, superconjunto de todas as relações entre usuários e itens. Do mesmo modo, para todos os n_{UU} conjuntos de relações entre usuários R_{UU_j} existentes em um sistema, $0 \leq j < n_{UU}$, temos o conjunto $R_{UU} = \bigcup_{\forall j} R_{UU_j}$, superconjunto de todas as relações entre usuários. E também, para todos os n_{II} conjuntos de relações entre itens R_{II_j} existentes em um sistema, $0 \leq j < n_{II}$, temos o conjunto $R_{II} = \bigcup_{\forall j} R_{II_j}$, superconjunto de todas as relações entre itens.

A partir de R_{UI_j}, R_{UU_j} e R_{II_j}, definimos conjuntos de arestas para o modelo de recomendação:

- A_{UI_j} é um conjunto de arestas a_{ui_j} que representam cada uma das relações r_{ui_j} de R_{UI_j}. Para cada r_{ui_j} de R_{UI_j} existe uma aresta correspondente a_{ui_j} em A_{UI_j}. A função $f_{A_{UI_j}} : R_{UI_j} \mapsto A_{UI_j}$ faz o mapeamento de R_{UI_j} para A_{UI_j}.

- A_{UU_j} é um conjunto de arestas $a_{u'u''_j}$ que representam cada uma das relações $r_{u'u''_j}$ de R_{UU_j}. Para cada $r_{u'u''_j}$ de R_{UU_j} existe uma aresta correspondente $a_{u'u''_j}$ em A_{UU_j}. A função $f_{A_{UU_j}} : R_{UU_j} \mapsto A_{UU_j}$ faz o mapeamento de R_{UU_j} para A_{UU_j}.

- A_{II_j} é um conjunto de arestas $a_{i'i''_j}$ que representam cada uma das relações $r_{i'i''_j}$ de R_{II_j}. Para cada $r_{i'i''_j}$ de R_{II_j} existe uma aresta correspondente $a_{i'i''_j}$ em A_{II_j}. A função $f_{A_{II_j}} : R_{II_j} \mapsto A_{II_j}$ faz o mapeamento de R_{II_j} para A_{II_j}.

Com base nos superconjuntos R_{UI}, R_{UU} e R_{II}, definimos os conjuntos de arestas que representam todas as relações presentes em um sistema de recomendação:

- A_{UI} é um conjunto de arestas a_{ui} dadas pela união de todas as arestas pertencentes à família de conjuntos A_{UI_j}: $A_{UI} = \bigcup_{\forall j} A_{UI_j}$.
- A_{UU} é um conjunto de arestas a_{uu} dadas pela união de todas as arestas pertencentes à família de conjuntos A_{UU_j}: $A_{UU} = \bigcup_{\forall j} A_{UU_j}$.
- A_{II} é um conjunto de arestas a_{ii} dadas pela união de todas as arestas pertencentes à família de conjuntos A_{II_j}: $A_{II} = \bigcup_{\forall j} A_{II_j}$.

Tendo definidos conjuntos de vértices para usuários (V_U) e itens (V_I) e as relações A_{UI}, A_{UU} e A_{II}, definimos o grafo G_{Rec} que representa todos os elementos relevantes pertencentes a sistemas de recomendação e suas relações:

$G_{Rec} = (V_{Rec}, A_{Rec}, \Psi_{Rec})$, onde $V_{Rec} = V_U \cup V_I$, $A_{Rec} = A_{UI} \cup A_{UU} \cup A_{II}$ e $\Psi_{Rec} : A_{Rec} \mapsto V_{Rec} \times V_{Rec}$ é a função que mapeia cada $a_{Rec} \in A_{Rec}$ a um par ordenado (v'_{Rec}, v''_{Rec}), $v'_{Rec}, v''_{Rec} \in V_{Rec}$.

Como os dados e metadados de usuários e itens e de suas relações podem ser úteis para a realização de recomendações, são definidas funções para obtenção desses dados.

Seja $f_{dU} : U \mapsto D_U$ a função que mapeia cada usuário $u \in U$ para os seus respectivos dados $d_u \in D_U$. A função $f_{dV_U} : V_U \mapsto D_U$, $f_{dV_U} = f_d u \circ f_{V_U}^{-1}$ mapeia cada vértice $v_u \in V_U$ para os dados $d_u \in D_U$ dos usuários u que representam.

Seja $f_{dI} : I \mapsto D_I$ a função que mapeia cada item $i \in I$ para seu conteúdo e metadados $d_i \in D_I$. A função $f_{dV_I} : V_I \mapsto D_I$, $f_{dV_I} = f_d i \circ f_{V_I}^{-1}$ mapeia cada vértice $v_i \in V_I$ para $d_i \in D_I$.

Seja $f_{dR_{UI}} : R_{UI} \mapsto D_{R_{UI}}$ a função que mapeia cada relação $r_{ui} \in R_{UI}$ para seus metadados $d_{r_{ui}} \in D_{R_{UI}}$. A função $f_{dA_{UI}} : A_{UI} \mapsto D_{R_{UI}}$, $f_{dA_{UI}} = f_{dR_{UI}} \circ f_{A_{UI}}^{-1}$ mapeia cada aresta $a_{ui} \in A_{UI}$ para $d_{r_{ui}} \in D_{R_{UI}}$.

Seja $f_{dR_{UU}} : R_{UU} \mapsto D_{R_{UU}}$ a função que mapeia cada relação $r_{u'u''} \in R_{UU}$ para seus metadados $d_{r_{u'u''}} \in D_{R_{UU}}$. A função $f_{dA_{UU}} : A_{UU} \mapsto D_{R_{UU}}$, $f_{dA_{UU}} = f_{dR_{UU}} \circ f_{A_{UU}}^{-1}$ mapeia cada aresta $a_{u'u''} \in A_{UU}$ para $d_{r_{u'u''}} \in D_{R_{UU}}$.

Seja $f_{dR_{II}} : R_{II} \mapsto D_{R_{II}}$ a função que mapeia cada relação $r_{i'i''} \in R_{II}$ para seus metadados $d_{r_{i'i''}} \in D_{R_{II}}$. A função $f_{dA_{II}} : A_{II} \mapsto D_{R_{II}}$, $f_{dA_{II}} = f_{dR_{II}} \circ f_{A_{II}}^{-1}$ mapeia cada aresta $a_{i'i''} \in A_{II}$ para $d_{r_{i'i''}} \in D_{R_{II}}$.

Com as definições acima, é possível, por exemplo, definir grafos que sejam aderentes a modelos de dados de técnicas de recomendação clássicas.

Para um sistema de recomendação que utilize filtragem colaborativa, como apresentado na Seção 2.2, para a geração de recomendações, só é necessário utilizar as relações entre usuários e itens. Assim, o modelo de grafos para essa técnica específica é $G_{CF} = (V_{CF}, A_{CF}, \Psi_{CF})$, onde $V_{CF} = V_{Rec}$, $A_{CF} = A_{UI}$ e $\Psi_{CF} : A_{CF} \mapsto V_{CF} \times V_{CF}$. Note-se que A_{CF} é um subconjunto de A_{Rec} e Ψ_{CF} é um subconjunto de Ψ_{Rec}. Logo, G_{CF} é um subgrafo de G_{Rec}.

Para um sistema de recomendação que utilize a técnica de filtragem baseada em conteúdo, como apresentado na Seção 2.3, são necessárias informações de similaridade de itens, ou seja, das relações entre itens e dos itens relacionados a um dado usuário, ou seja, relações entre usuários e itens. Assim, o modelo de grafos para essa técnica específica é $G_{CBF} = (V_{CBF}, A_{CBF}, \Psi_{CBF})$, onde $V_{CBF} = V_{Rec}$, $A_{CF} = A_{UI} \cup A_{II}$ e $\Psi_{CBF} : A_{CBF} \mapsto V_{CBF} \times V_{CBF}$. Note-se aqui também que A_{CBF} é um subconjunto de A_{Rec} e Ψ_{CBF} é um subconjunto de Ψ_{Rec}. Logo, G_{CBF} é um subconjunto de G_{Rec}.

Assim, G_{Rec} é um grafo genérico para representação das informações importantes associadas a sistemas de recomendação.

3.2 Sistema de recomendação utilizando agrupamento de dados

Há inúmeras técnicas empregadas em sistemas de recomendação, como apresentado na Seção 2.1. Dentre elas, agrupamento de dados é uma das opções.

O uso de agrupamento de dados em sistemas de recomendação é comum devido a algumas vantagens oferecidas:

- A quantidade de elementos a ser avaliada para fazer recomendações diminui, já que se pode limitar aos elementos existentes dentro de um determinado agrupamento apenas, ou somente ao agrupamento;
- A precisão das recomendações tende a aumentar, pois *outliers* e o ruído existentes nas distrubuição dos itens são atenuados nos grandes grupos significativos.

Como o agrupamento de dados pode ser empregado em diversos tipos de dados diferentes, com objetivos diferentes, define-se na Seção 3.3 o algoritmo proposto para geração de recomendações a partir do grafo base utilizando um algoritmo de agrupamento de dados.

3.3 Recomendação sobre o Grafo Base utilizando agrupamento de dados

Como a idéia principal desse trabalho é utilizar toda a informação disponível sobre itens, usuários e suas relações em sistemas de informação para a geração de recomendações, o modelo de dados de recomendação utilizado é bastante rico em informações de todas as naturezas.

Para utilizar toda a riqueza de informações do modelo para geração de recomendações, os agrupamentos de dados devem ser realizados de forma que o volume e qualidade de informações não sejam perdidos.

Uma das maneiras de se alcançar isso é fazendo agrupamento em cada um dos tipos de vértices do modelo (V_U e V_I) utilizando diversas métricas, baseando-se em um ou mais conjuntos de arestas associados aos vértices. Para isso, é necessário alinhar o modelo de dados proposto com o modelo de dados utilizado pela técnica de agrupamento.

Para utilizar o grafo $G_{Rec} = (V_{Rec}, A_{Rec}, \Psi_{Rec})$ apresentado na Seção 3.1 para recomendação utilizando agrupamento de dados, é necessário adaptá-lo para que sirva de entrada para algoritmos de agrupamento e de forma que a resposta reflita as necessidades de um algoritmo de recomendação.

Genericamente, algoritmos de agrupamento de dados possuem como entrada o conjunto de dados que se deseja agrupar e outros parâmetros específicos para cada técnica. Dessa forma, é possível apenas especificar como o conjunto de dados deve ser gerado a partir do grafo base. Outros parâmetros precisam ser analisados e especificados separadamente.

Geralmente, o conjunto de dados agrupado por um algoritmo de agrupamento é um conjunto de pontos p em um espaço S. No grafo base, temos dois conjuntos que podem ser utilizados para agrupamento: V_{Rec} e A_{Rec}.

A partir do conjunto de vértices V_{Rec}, define-se um algoritmo de recomendação que leva em consideração a similaridade entre cada vértice de V_{Rec} para definir os agrupamentos. Como os vértices de V_{Rec} não possuem nenhuma informação associada a eles diretamente, podem-se utilizar as funções f_{dU} e f_{dI}, para obter os dados dos vértices pertencentes aos conjuntos V_U e V_I respectivamente. Sobre esses dados pode-se utilizar o conceito de descritores e extrair características dos vértices, gerando vetores de característica. Tais vetores, finalmente, podem ser utilizados com entrada de um algoritmo de agrupamento de dados.

Como as arestas A_{UU}, A_{II} e A_{UI} e os dados associados às relações representadas por elas, R_{UU}, R_{II} e R_{UI}, estão diretamente ligadas aos conjuntos de vértices V_U e V_I, podem-se utilizar tais dados, fornecidos respectivamente pelas funções $f_{dR_{UU}}$, $f_{dR_{II}}$ e $f_{dR_{UI}}$ na extração de características dos vértices em V_U e V_I. Tendo os vetores de características extraídos dos vértices de V_U e V_I, eles são utilizados também como entrada em um algoritmo de agrupamento de dados.

O resultado dos métodos acima são diferentes agrupamentos dos vértices de V_U e V_I, como relação a diversas métricas diferentes. Tais informações precisam, então, ser unificadas, de forma a gerar uma informação de recomendação para o sistema.

No caso dos vértices de V_I, no momento em que o sistema for recomendar conteúdo para um usuário, seja uma recomendação online ou offline, o sistema atribui um peso a

cada um dos itens desconhecidos pelo usuário. Tal peso é o número de ocorrências daquele item em agrupamentos de itens relevantes para o usuário: grupos que tenham pelo menos um item que tenha sido considerado relevante pelo usuário. Após o cálculo de todos os pesos, os itens são então ordenados e um corte é feito na ordenação. Os itens de peso maior que o corte são então recomendados ao usuário.

Para recomendar conteúdo ao usuário utilizando os agrupamentos feitos sobre os vértices de V_U, o sistema atribui pesos a cada um dos itens desconhecidos. Neste caso, o peso é o número de ocorrências daquele item nos perfis dos usuários que pertençam aos mesmos grupos do usuário. A partir desse ponto, a técnica é similar à descrita anteriormente.

Para cada domínio, a seleção de nós, arestas, informações e métricas a serem utilizados deve ser analisada. Não há uma regra geral com a qual é possível determinar antecipadamente o que trará melhores resultados. Após a seleção, é necessário executar o algoritmo de agrupamento de dados para cada um dos descritores desejados, sobre os respectivos conjuntos de nós e de informações associadas.

Para uma abordagem híbrida, basta executar qualquer grupo de métodos citados acima independentemente e combinar os resultados parciais ou finais. Para combinar resultados parciais, podem-se unir os conjuntos de agrupamentos do mesmo tipo (agrupamentos de V_U e de V_I) e prosseguir com a recomendação. Para combinar os resultados finais, geram-se as recomendações utilizando o método descrito acima e antes de apresentar as recomendações ao usuário, faz-se a união dos conjuntos de recomendações retornados pelos métodos utilizados.

Existem diversos algoritmos de agrupamento de dados descritos na literatura. Neste trabalho, o algoritmo escolhido para agrupar os dados foi o OPF (Seção 2.7.2). A forma proposta para se utilizar o algoritmo OPF para agrupamento de dados nesse trabalho é apresentada na Seção 3.4.2.

3.4 Aplicação de Floresta de Caminhos Ótimos para recomendação

Para efetuar o agrupamento de dados para recomendação como descrito na Seção 3.3, o algoritmo OPF (Seção 2.7.2), considerado o estado da arte em termos de agrupamento de dados passa a ser uma das melhor opções para a realização dessa tarefa no sistema de recomendação.

Como é possível observar na apresentação do OPF (Seção 2.7.2) e da técnica de recomendação utilizando agrupamento de dados proposta na Seção 3.3, *a priori*, não é possível aplicar OPF diretamente sobre o modelo geral de recomendação.

Por isso, é necessário fazer um alinhamento entre o modelo de dados utilizado pelo OPF, resumido na Seção 3.4.1 e o modelo de dados de recomendação proposto. Tal alinhamento é apresentado na Seção 3.4.2.

3.4.1 Modelo de dados do OPF

Como apresentado na Seção 2.7.2, o algoritmo OPF tem como entrada um grafo $G_{OPF} = (V_{OPF}, A_{OPF}, \Psi_{OPF})$, onde V_{OPF} é um conjunto de vértices que representa pontos em algum espaço de características, A_{OPF} é o conjunto de arestas que conecta cada vértice $v_{OPF} \in V_{OPF}$ aos seus n vizinhos mais próximos e Ψ_{OPF} é a função que mapeia cada aresta $a_{OPF} \in A_{OPF}$ ao par não ordenado (v'_{OPF}, v''_{OPF}) de vértices de V_{OPF}. Vértices do grafo G_{OPF} são ponderados pela função $\rho : V_{OPF} \mapsto R$ e as arestas são ponderadas pela função $d : V_{OPF} \times V_{OPF} \mapsto R$.

3.4.2 Adaptação do modelo de recomendação para o modelo de dados do OPF

Para utilizar o grafo $G_{Rec} = (V_{Rec}, A_{Rec}, \Psi_{Rec})$ apresentado na Seção 3.1 para recomendação utilizando OPF, é necessário modificá-lo em alguns aspectos.

As arestas do OPF têm uma semântica específica: determinam a similaridade entre os nós, o que pode ou não ser verdade para alguma classe específica de arestas A_j de A_{Rec}. Dessa forma, alguma modificação deve ser feita de modo que as arestas a serem usadas pelo OPF tenham a semântica correta.

Como descrito na Seção 3.3, em um algoritmo de agrupamento de dados a entrada padrão é, além de parâmetros específicos de cada técnica, o conjunto de dados que devem ser agrupados apenas. Tais dados, como apresentado nessa Seção, são utilizados como entrada do OPF.

Além do parâmetro padrão, OPF recebe também uma função discreta de distribuição de probabilidade ρ, que é calculada utilizando como parâmetro uma função de adjacência A, o tamanho de vizinhança n e um valor d_f da maior aresta existente no grafo. O número de vizinhos mais próximos n pode ser calculado otimamente como citado em [8]. O parâmetro d_f pode ser calculado a partir do conjunto de dados a ser agrupado. A adjacência pode ser definida da melhor forma para o domínio.

Tendo os dados agrupados, o procedimento continua como descrito na Seção 3.3.

Este trabalho oferece um modelo baseado em grafos, genérico e extensível, para representação e exploração de dados referentes à recomendação. Além disso, oferece uma visão de quais são as possibilidades para utilizar o algoritmo de Floresta de Caminhos Ótimos (OPF) para efetuar agrupamentos no nós dos modelos propostos, de forma que se possa

fazer recomendações a partir desses agrupamentos.

Capítulo 4

Arquitetura Proposta para Sistemas de Recomendação

4.1 Arquitetura do sistema de recomendação

A arquitetura proposta do sistema de recomendação foi elaborada tendo em vista a necessidade de criar um sistema de recomendação que tivesse a mesma estrutura externa de um sistema de recomendação clássico (no maior nível de abstração possível, um repositório de dados e um recomendador), mas que pudesse ao mesmo tempo ser extensível, escalável e que pudesse ser utilizada tanto como parte de um sistema de recomendação externo como sendo o agrupador de diversos outros sistemas de recomendação. A Figura 4.1 apresenta a arquitetura proposta.

A arquitetura do sistema de recomendação proposta é dividida em dois módulos principais: o repositório de dados (1) e o recomendador (2), o núcleo do sistema de recomendação. Alguns módulos adicionais, para pré-processamento dos dados (3) e validação do sistema (4), são adicionados para servir de suporte a recomendação e facilitar a execução de experimentos e validação do sistema, respectivamente.

No repositório de dados ficam armazenados os dados de usuários, de itens e das avaliações feitas pelos usuários. Cada um desses dados fica armazenado em um repositório separado, oferecendo informações que podem ser consideradas em diversos tipos diferentes de recomendadores.

O recomendador é o componente responsável por gerar as recomendações para os usuários utilizando os dados do repositório por meio de uma ou mais técnicas de recomendação.

Uma das configurações possíveis do recomendador é o uso uma estrutura hierárquica de recomendadores, apresentada na Figura 4.2. Para isso, são implementados diferentes recomendadores, que serão chamados pelo recomendador externo e que servirão de base

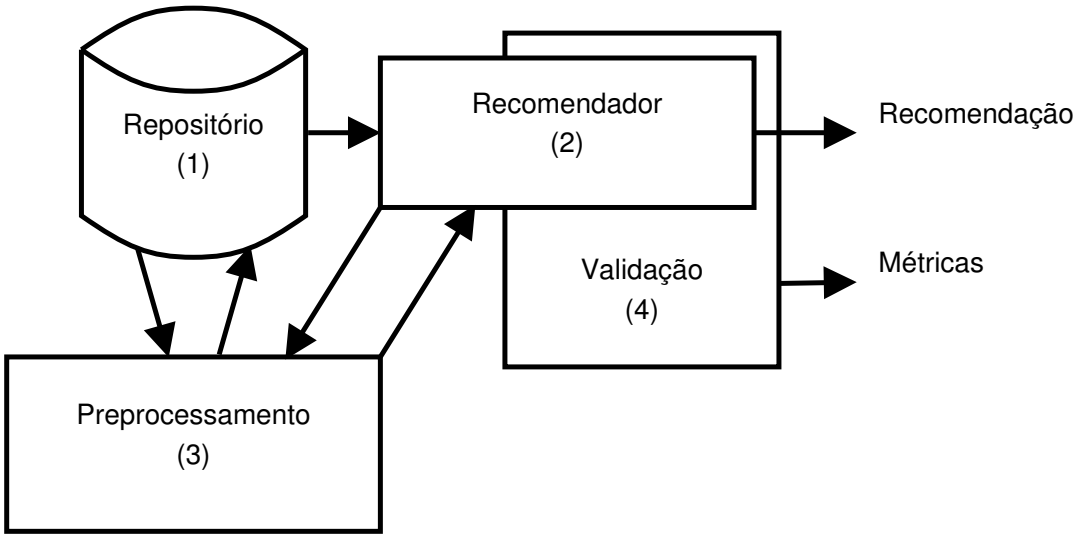

Figura 4.1: Visão geral da arquitetura proposta para um sistema de recomendação. 1 - Repositórios de Dados. 2 - Recomendador. 3 - Preprocessamento. 4 - Validação.

para a composição do resultado por ele retornado. Tais recomendadores (1) podem ler dados dos repositórios e outros dados que tenham sido gerados em alguma etapa anterior de preprocessamento dos dados, e geram valores de avaliação para um conjunto de usuários e itens predeterminados. Os resultados de todos esses recomendadores, assim que gerados, são combinados por um outro componente interno, o agregador de recomendações (2), que utiliza alguma função de agregação para gerar um único resultado de recomendação, que será a saída do recomendador externo.

O componente de Validação serve de base para a criação de experimentos e para validação do sistema, calculando métricas de eficiência, eficácia e outras medidas de interesse. Para validar um sistema de recomendação que utilize a mesma arquitetura, o componente de métodos de validação oferece uma interface padronizada para diversos métodos de validação. Um dos métodos oferecidos é o *k-fold cross-validation*, que gera, a partir do conjunto de dados completo, k subconjuntos de dados, que serão utilizados em k iterações do recomendador, sendo que $k-1$ desses subconjuntos são usados como entrada do recomendador e o subconjunto restante será usada para validação dos resultados obtidos. Para medir a qualidade dos resultados, são utilizadas métricas de validação, também oferecidas pelo componente. Algumas métricas clássicas de sistemas preditivos, como MAE (*Mean Absolut Error*) e RMSE (*Root Mean Squared Error*) já estão implementadas. Em vários métodos de validação, como no *k-fold cross-validation*, é necessária

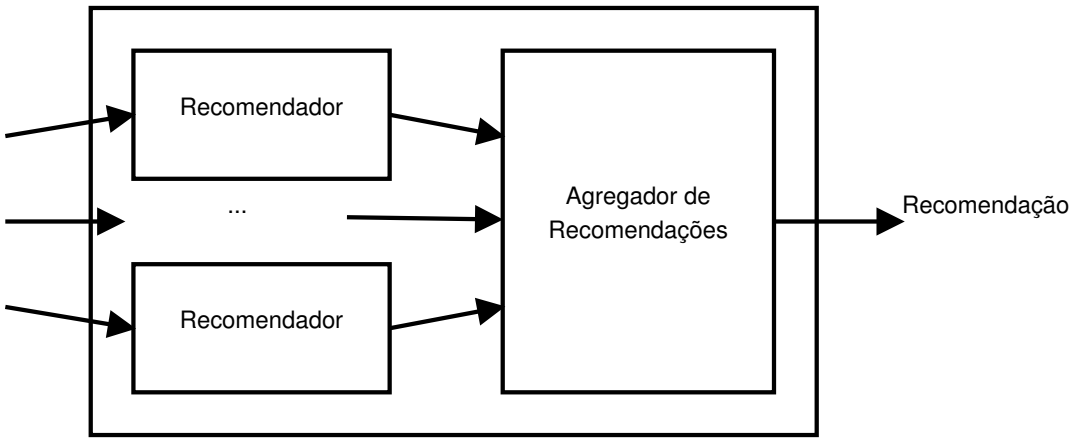

Figura 4.2: Recomendador composto. As recomendações geradas por esse recomendador são geradas a partir de outros recomendadores, e as recomendações geradas por eles são agregadas pelo agregador de recomendações.

uma etapa de agregação dos resultados, para gerar um valor único de avaliação, que é oferecido pelo componente de agregação de validações.

O outro componente externo oferece preprocessamento dos dados, de forma que os recomendadores possam utilizar dados mais refinados ou agregados. Dois componentes internos oferecem essas duas funcionalidades: descritores e técnicas de aprendizado de máquina. Os descritores são formados por duas funções, uma para gerar vetores de características a partir dos dados brutos do repositório e outra para calculo de distância, que geram um valor numérico a partir de dois vetores de características. Descritores são úteis para abstrair os dados e mensurar a semelhança entre eles. O componente de aprendizado de máquina permite agrupar, classificar e perceber outros tipos de padrões nos dados.

Nas próximas seções, cada um desses 4 componentes é detalhado. Na Seção 4.2 é apresentada a organização dos repositórios. O recomendador é descrito e detalhado na Seção 4.3, seguido do agregador de recomendações na Seção 4.4. Na Seção 4.5, o componente responsável pela validação é descrito. Por fim, o componente que encapsula os métodos utilizados no preprocessamento é detalhado na Seção 4.6.

4.2 Repositórios

Dois repositórios são definidos para sistemas de recomendação: os repositórios de dados do sistema e os repositórios de dados agregados. No repositório de dados do sistema

são armazenados os dados originais sobre usuários, itens e todas os informações brutas, originais do sistema ao qual o sistema de recomendação está inserido. O repositório de dados agregados armazena os dados oriundos de processamentos feitos dentro do sistema de recomendação, como as caracterizações dos itens e usuários, as distâncias pré-calculadas e os agrupamentos que possam ser utilizados pelo Recomendador.

4.3 Recomendadores

Um recomendador é um componente que recebe dados dos repositórios de usuários, itens e avaliações e gera um novo conjunto de avaliações, não existentes no repositório original, como apresentado na Seção 4.1.

Para que recomendadores diferentes possam ser utilizados intercambiavelmente, um recomendador deve implementar uma interface padrão predefinida. A seguinte interface deve ser implementada:

```
<Lista de Avaliações> recomendador(
  <Lista de usuários> users,
  <Lista de Itens> items,
  <Lista de avaliações do repositório> ratings,
  <Lista de avaliações a serem geradas> rec,
  <Lista de opções> opt)
```

Na interface, o recomendador recebe as listas dos dados existentes nos repositórios de itens, usuário e avaliações, uma lista contendo os pares de usuários e itens para os quais deve-se estimar a avaliação e uma lista de opções, para que parâmetros próprios de cada recomendador possam ser passados sem ferir a interface padrão. A lista de avaliações a serem geradas é retornada, com os valores de avaliação inferidos pelo pelo recomendador.

O quarto parâmetro da interface, a lista de avaliações a ser preenchida (`rec`) é importante pois não é interessante que recomendações sejam geradas para todos os usuários (exemplos de usuário que não necessitam de recomendação são usuários inativos, bloqueados e banidos) e nem para todos os itens para um dado usuário (por exemplo, itens podem ser filtrados em uma etapa anterior à recomendação para poupar tempo do recomendador). Além disso, em alguns tipos de experimentos como o *k-fold cross-validation*, os dados de avaliação que são utilizados para a validação dos sistema para um dado *fold* são definidos, fazendo com que se poupe tempo do recomendador se as recomendações forem geradas apenas para o subconjunto de interesse.

Na arquitetura proposta, recomendadores podem ser vistos de duas maneiras diferentes: como o núcleo do sistema de recomendação, que lê diretamente os dados dos

repositórios e gera as recomendações, ou como componentes desse núcleo, que terão seus resultados agrupados para compor um único resultado. Não há uma diferença na implementação desses recomendadores, uma vez que qualquer recomendador pode ser composto por outros recomendadores e um agregador de recomendações, como dito em 4.1.

Nas seções a seguir serão apresentados alguns dos recomendadores pospostos. Na Seção 4.3.1 é apresentado um algoritmo de filtragem baseado em conteúdo. Um algoritmo de filtragem colaborativa é apresentado na Seção 4.3.2. A Seção 4.3.3 um algoritmo híbrido.

4.3.1 Recomendador utilizando filtragem por conteúdo e agrupamento de dados

Um recomendador para filtragem baseada em conteúdo e agrupamento de itens foi implementado e seu algoritmo é descrito a seguir.

```
<Lista de Avaliações> recomendador_4.3.1(
  <Lista de usuários> users,
  <Lista de Itens> items,
  <Lista de avaliações do repositório> reczero,
  <Lista de avaliações a serem geradas> ratings,
  <Lista de opções> opt)
```

O *Algoritmo de Filtragem baseada em conteúdo utilizando Agrupameto de dados* parte dos dados padrão de qualquer recomendador da arquitetura e mais dois parâmetros extras: a lista de clusters dos itens (`clusters`) e um valor padrão (`default`). Após o cálculo da média das avaliações dos itens de cada cluster para cada usuário (linhas 1-6), a Lista de Recomendações `rec` é preenchida, para cada par de usuários e itens, utilizando o valor correspondente à avaliação média daquele usuário para o *cluster* em que aquele item está (linhas 7-10). Por fim, a Lista de Recomendações `rec`, já preenchida, é retornada.

Como é possível observar, o algoritmo é bastante simples. Com poucos passos é possível gerar recomendações. Os *clusters* de itens podem ser gerados utilizando qualquer técnica, desde que os clusters formados sejam bons.

4.3.2 Recomendador utilizando filtragem colaborativa e agrupamento de dados

Um recomendador para filtragem colaborativa e agrupamento de dados foi implementado e seu algoritmo é descrito a seguir.

Algoritmo *Algoritmo de Filtragem baseada em conteúdo utilizando Agrupameto de dados*

Entrada: Lista de usuários ***users***, Lista de Itens ***itens***, Lista de avaliações do repositório ***ratings***, Lista de avaliações a serem geradas ***reczero***, Lista de opções ***opt***
Saída: Lista de Avaliações ***rec***

1. **para cada** u em $users$
2. **faça para cada** c em $opt[clusters]$
3. **faça se** número de itens avaliados pelo usuário em c = 0
4. **então** $predrating[u, c] \leftarrow opt[default]$
5. **senão** $predrating[u, c] \leftarrow opt[agg]$ das avaliações dos itens avaliados pelo usuário u que estão no cluster c
6. $rec \leftarrow reczero$
7. **para cada** (u, i) em rec
8. **faça** $c \leftarrow$ cluster de $opt[clusters]$ em que i está
9. $rec[u, i] \leftarrow predrating[u, c]$

```
<Lista de Avaliações> recomendador_4.3.2(
  <Lista de usuários> users,
  <Lista de Itens> items,
  <Lista de avaliações do repositório> reczero,
  <Lista de avaliações a serem geradas> ratings,
  <Lista de opções> opt)
```

O *Algoritmo de Filtragem Colaborativa utilizando Agrupamento de Dados*, como o *Algoritmo de Filtragem baseada em conteúdo utilizando Agrupameto de dados*, parte dos dados padrão de qualquer recomendador da arquitetura e mais dois parâmetros extras: a lista de clusters dos itens (***clusters***) e um valor padrão (***default***). Inicialmente, a médias das avaliações dos itens para cada *cluster* de usuários é calculada (linhas 1-5). Na seqüência, a Lista de Recomendações ***rec*** é preenchida, para cada par de usuários e itens, utilizando o valor médio de avaliação do item para os usuários do mesmo *cluster* (linhas 7-9). Por fim, a Lista de Recomendações ***rec***, já preenchida, é retornada.

Como é possível observar, esse algoritmo é muito similar ao Algoritmo de Filtragem baseada em conteúdo utilizando Agrupamento de Dados 4.3.1. A grande diferença está no tipo de *cluster* utilizado como entrada. Nesse algoritmo, devem-se utilizar *clusters* de usuários, ao contrário do Algoritmo de Filtragem baseada em conteúdo utilizando agrupamento de dados, em que eram usados *clusters* de itens. Tais *clusters* podem ser gerados utilizando características do próprio usuário para medir a similaridade (metadados

Algoritmo *Algoritmo de Filtragem Colaborativa utilizando Agrupamento de Dados*
Entrada: Lista de usuários ***users***, Lista de Itens ***itens***, Lista de avaliações do repositório ***ratings***, Lista de avaliações a serem geradas ***reczero***, Lista de opções ***opt***
Saída: Lista de Avaliações ***rec***

1. **para cada** i em $items$
2. **faça para cada** c em $opt[clusters]$
3. **faça se** número de usuários que avaliou o item i em c = 0
4. **então** $predrating[i, c] \leftarrow opt[default]$
5. **senão** $predrating[i, c] \leftarrow opt[agg]$ das avaliações do item i pelos usuários que estão no cluster c
6. $rec \leftarrow reczero$
7. **para cada** (u, i) em rec
8. **faça** $c \leftarrow$ cluster de $opt[clusters]$ em que u está
9. $rec[u, i] \leftarrow predrating[i, c]$

associados ao usuário, como idade e localização geográfica) ou então as avaliações dos usuários com os itens.

4.3.3 Recomendador híbrido utilizando agrupamento de dados

Um recomendador híbrido que utiliza agrupamento de dados foi implementado e seu algoritmo é descrito a seguir.

```
<Lista de Avaliações> recomendador_4.3.3(
  <Lista de usuários> users,
  <Lista de Itens> items,
  <Lista de avaliações do repositório> reczero,
  <Lista de avaliações a serem geradas> ratings,
  <Lista de opções> opt)
```

O Algoritmo Híbrido utilizando Agrupamento de dados utiliza os mesmo parâmetros dos outros recomendadores da arquitetura proposta mais dois extras: a lista de clusters de usuários e itens (***clusters***) e um valor padrão (***default***). A médias das avaliações dos itens (linha 5) e das avaliações por usuário (linha 4) pertencentes a cada um dos *clusters* são previamente calculadas (linhas 1-5). Na seqüência, a Lista de Recomendações ***rec*** é preenchida (linha 7), para cada par de usuários e itens, utilizando o valor médio das avaliação do item e das avaliações por usuário calculados anteriormente (linhas 7-9), caso

Algoritmo *Algoritmo Híbrido utilizando Agrupamento de Dados*
Entrada: Lista de usuários `users`, Lista de Itens `itens`, Lista de avaliações do repositório `ratings`, Lista de avaliações a serem geradas `reczero`, Lista de opções `opt`
Saída: Lista de Avaliações `rec`
1. **para cada** c em $opt[clusters]$
2. **faça para cada** e em c
3. **faça se** e é um usuário
4. **então** $predratinguser[e] \leftarrow$ média das avaliações feitas por u dos itens de c
5. **senão** $predratingitem[e] \leftarrow$ média das avaliações de i feitas por usuários em c
6. $rec \leftarrow reczero$
7. **para cada** (u, i) em rec
8. **faça se** $predratinguser[u]$ está definido
9. **então** $rec[u, i] \leftarrow opt[agg](predratinguser[u], predratingitem[i])$
10. **senão** $rec[u, i] \leftarrow opt[default]$

existam. Caso contrário, um valor padrão é atribuído (linha 10). No final, a Lista de Recomendações `rec` é retornada.

Esse algoritmo utiliza *clusters* de usuários e itens para gerar recomendações. Tais *clusters* são gerados utilizando-se três funções de similaridade: a similaridade entre os usuários, similaridade entre os itens e de similaridade entre um usuário e um item. Como funções de similaridade entre elementos da mesma categoria, podem ser utilizadas funções como as descritas nas Seções 4.3.2 e 4.3.1. A função de similaridade entre usuários e itens pode ser definida de acordo com a afinidade do usuário com tal item, utilizando, por exemplo, a avaliação do usuário àquele item ou a itens similares.

4.4 Agregador de recomendações

Como apresentado na Seção 4.3, é possível criar recomendadores compostos por outros recomendadores, sendo o resultado do recomendador composto gerado por um módulo de agregação. Esse módulo executa, para cada resultado dos recomendadores simples, uma função de agregação, gerando um único resultado como saída do recomendador.

O módulo de agregação de recomendações é implementado segundo a seguinte interface:

```
<Lista de Recomendações> agregacao_de_recomendacao(
  <Lista de Listas de Recomendações> LRec,
```

```
<Lista de Recomendações> RecZero,
<Função de agregação> agg)
```

onde **Rec** é a saída gerada preenchendo os valores de recomendação na **RecZero**, utilizando a função **agg** sobre elementos de lista de recomendações **LRec**. O agregador de recomendações assume que todas as listas de recomendações de **LRec** contenham os mesmos pares de usuários e itens de **RecZero** e gera a saída do agregador executando a função **agg** sobre a lista formada por todas as recomendações para um dado par. O algoritmo do agregador de recomendações é apresentado a seguir.

Algoritmo *Agregação de Recomendações*
Entrada: Lista de Listas de Recomendações **LRec**, Lista de Recomendações **RecZero**, Função de agregação **agg**
Saída: Lista de Recomendações **Rec**
1. $Rec \leftarrow RecZero$
2. **para cada** $(usuario, item)$ em Rec
3. **faça para cada** $recomendacao$ em $LRec$
4. **faça** $avaliacao \leftarrow recomendacao\{usuario, item\}$
5. $Rec\{usuario, item\} \leftarrow agg(avaliacao)$

Para cada um dos pares de usuários e itens para os quais foram geradas recomendações (linha 2), o agregador de recomendações mapeia o conjunto de avaliações existentes na lista de recomendações **LRec** (linhas 3 e 5) e gera, a partir da função de agregação **agg**, um único valor de avaliação, que será atribuído como o valor de recomendação para daquele item para aquele usuário (linha 6). O conjunto de todas as recomendações geradas por essa agregação é retornado pelo agregador.

4.5 Validação

O módulo de validação é responsável por executar experimentos e para validar o recomendador utilizando os métodos de validação e as métricas que forem necessárias.

O módulo de validação é dividido em dois outros módulos. O primeiro oferece algumas métricas para validar o recomendador. O outro é responsável por validar o sistema, executando o recomendador de acordo com algum método de validação.

O módulo de métricas oferece algumas métricas para calcular a eficácia dos recomendadores. Toda métrica tem a seguinte interface:

```
<Valor da métrica> calcula_metrica(<Recomendação> rec, <Validação> val)
```

onde *rec* é o conjunto de recomendações gerado pelo recomendador e *val* é o conjunto de avaliações que serão usados como base para o cálculo da eficácia. Uma métrica retorna um valor numérico, que indica a qualidade de uma recomendação segundo aquela métrica.

Duas métricas são clássica em sistemas de recomendação: MAE (*Mean Absolut Error*) e RMSE (*Root Mean Squared Error*). Ambas têm como objetivo medir o erro acumulado em todas as recomendações feitas, medindo quão efetivas foram as predições.

O módulo responsável por validar o sistema oferece um mecanismo para validar o sistema. O módulo de validação oferece a seguinte interface:

```
<Lista de Métricas calculadas> validacao(
  <Lista de Usuários> users,
  <Lista de Itens> items,
  <Lista de avaliações> ratings,
  <Recomendador> recommender,
  <Lista de opções do Recomendador> rec_opt,
  <Lista de Métricas> metrics,
  <Agregação de métricas> metric_agg?,
  <Lista de opções> opt)
```

onde as listas *users*, *items* e *ratings* são os dados dos repositórios de usuários, itens e avaliações, respectivamente, o recomendador *recommender* é o módulo que executará o recomendador, *rec_opt* é a lista de opções específicas do recomendador, *metrics* contém todas as métricas que devem ser calculadas, que serão retornadas e *metric_agg* é a função que será usada para agregar os resultados de várias execuções do *recommender*, caso o método de validação execute-o mais de uma vez. Como pode ser desnecessário, a função *metric_agg* é opcional. O último parâmetro, *opt*, é uma lista de opções para que parâmetros diferentes dos definidos possam ser passado para o componente de validação.

Um dos métodos de validação implementados é o *k-fold cross-validation*, que divide o conjunto inicial de avaliações em *k* partes, e utiliza, em *k* etapas, 1 desses *folds* para validar o sistema e os outros *k-1* como entrada, agregando os resultados no final.

O algoritmo utilizado para executar o *k-fold cross-validation* é apresentado a seguir.

```
<Métricas calculadas> k_fold_cross_validation(
  <Lista de Usuários> users,
  <Lista de Itens> items,
  <Lista de avaliações> ratings,
  <Recomendador> recommender,
  <Lista de opções do Recomendador> rec_opt,
  <Lista de Métricas> metrics,
```

```
<Agregação de Métricas> metric_agg?,
<Lista de opções> opt)
```

Algoritmo *K-fold cross-validation*
Entrada: Lista de Usuários `users`, Lista de Itens `items`, Lista de avaliações `ratings`, Recomendador `recommender`, Lista de opções do Recomendador `rec_opt`, Lista de Métricas `metrics`, Agregação de métricas `metric_agg` (opcional), Lista de opções `opt`
Saída: Lista de Métricas calculadas `metriclist`

```
1. se metric_agg não está definida
2.    então metric_agg ← mean
3.          folds ← split(ratings, opt[k])
4. nonaggmetrics ← Matriz com tamanho(metrics) linhas e opt[k] colunas
5. para i ← 0 até k − 1
6.       faça train ← union(folds − folds[i])
7.            validation ← folds[i]
8.            rec ← recommender(users, items, train, zerofill(validation), rec_opt)
9.            para cada m em metrics
10.                 faça nonaggmetrics[m, i] ← m(rec, validation)
11.                      aggmetrics ← Vetor de tamanho(metrics) posições
12.para cada linha l em metrics
13.           faça aggmetrics[l] ← metric_agg(l)
```

O *k-fold cross-validation* divide a lista de avaliações `rating` em `k` listas de tamanho igual (ou no máximo diferindo de 1) usando a função `split` e armazena cada uma dessas `k` listas em `folds` (linha 4). A seguir, para cada uma dessas listas, o recomendador `recommender` é executado utilizando o `fold` atual como conjunto de recomendações a serem geradas (com os valores de avaliação apagados pela função `zerofill`) e todos os outros `folds` como base (linhas 6-9). Gerada a recomendação, as métricas na lista de métricas `metrics` são calculadas para a combinação de `folds` (linhas 10 e 11) e as métricas são agregadas ao final (linhas 12-14). Caso a função de agregação de métricas `metric_agg` não tenha sido definida, o algoritmo assume que a função é uma média simples (linhas 2-3).

4.6 Agrupamento de dados

Como apresentado nas Seções 4.3.2, 4.3.1 e 4.3.3, uma das técnicas de aprendizado de máquina utilizadas para implementar recomendadores nesse trabalho foi agrupamento de dados. O componente de agrupamento de dados é responsável por prover implementações

de diversas técnicas que possam ser usadas por outros componentes do sistema de recomendação.

Dentro do componente são encapsulados, entre algumas particularidades de implementação, apenas os algoritmos e técnicas de agrupamento de dados. Todas as técnicas e algoritmos são implementados utilizando a mesma interface padrão, de forma que possam ser trocados facilmente. A interface é definida da seguinte maneira:

```
<Lista de Agrupamentos> cluster(
  <Lista de objetos> objs,
  <Matriz de distância> dist,
  <Lista de opções> opt)
```

Como apresentado acima, cada técnica recebe três parâmetros: um para os objetos que serão agrupados (*objs*), outro para a matriz de distância dos objetos (*dist*) e um terceiro com a lista de parâmetros e opções próprio de cada técnica (*opt*), caso seja necessário. O retorno de cada técnica é uma lista contendo os agrupamentos gerados pela técnica de agrupamento.

Capítulo 5

Validação

Este capítulo apresenta os resultados da comparação experimental de vários métodos de agrupamento de dados para recomendação, dentro do modelo de sistema de recomendação proposto no Capítulo 3. São apresentados também os detalhes pertinentes a cada um dos experimentos realizados, como a caracterização de usuários e itens, funções de distância utilizadas, métodos de agrupamento de dados, métricas e bases de dados.

Os experimentos realizados comparam métodos de agrupamento de dados apresentados na Seção 2.7, como componente do sistema de recomendação descrito no Capítulo 3.

O objetivo dos experimentos relatados é avaliar a eficácia de cada um dos métodos de agrupamento de dados dentro do sistema de recomendação proposto.

A seguir a definição do problema é revista (Seção 5.1), seguida da metodologia utilizada (Seção 5.2), que compreende as bases de dados que foram utilizadas (Seção 5.2.1), a caracterização de usuários e itens utilizada (Seção 5.2.2), funções de distância (Seção 5.2.3) e métricas de avaliação (Seção 5.2.5). A seguir são apresentados os resultados dos experimentos de avaliação de eficácia (Seção 5.3).

5.1 Definição do problema

Em geral, há duas abordagens para validação de sistemas de recomendação: avaliação do ranking das primeiras N recomendações (*Top-N Recommentation*) e predição de avaliações. A primeira visa gerar um conjunto de itens não avaliados ou novos a usuários, enquanto a segunda tem como objetivo tentar predizer a avaliação que um usuário daria a um dado item. Para sistemas com avaliações explícitas, as duas formas de avaliação são possíveis, enquanto em sistemas com avaliações implícitas, a avaliação de itens novos é mais recomendada.

5.2 Metodologia

Das duas formas possíveis de avaliação de sistemas de recomendação, a predição de avaliações foi utilizada para avaliar a eficácia do recomendador. Na literatura de recomendação em bases com avaliação explícita, é muito mais comum o uso de uma avaliação desse tipo, já que ela utiliza mais informação do perfil do usuário: na avaliação de ranking, apenas a pertinência de um item ao perfil do usuário é considerada, enquanto na predição as avaliações também são utilizadas.

5.2.1 Bases de dados

Existem várias bases de dados utilizadas para avaliar e validar sistemas de recomendação. Entre as mais famosas estão as bases de recomendação de vídeos Movielens [1] [15] e as bases de recomendação de piadas Jester [2] [12]. Essas duas bases foram disponibilizadas mais de uma vez, contando em cada uma dessas ocasiões com conjuntos de dados bastante distintos.

Existem no total 3 bases Movielens:

- *100k Ratings Data Set*: possui 100.000 avaliações de 1682 filmes feitas por 943 usuários;
- *1M Ratings Data Set*: possui 1.000.000 de avaliações para 3900 filmes feitas por 6040 usuários;
- *10M Ratings, 100k Tags Data Set*: 10.000.000 de avaliações e 100.000 *tags* para 10681 filmes feitas por 71567 usuários.

Todas as bases Movielens possuem avaliações inteiras entre 1 e 5 (1, 2, 3, 4 e 5).
A base Jester foi disponibilizada em duas ocasiões:

- *Dataset 1*: 4.1 milhões de avaliações de 100 piadas feitas por 73.421 usuários;
- *Dataset 2* 1.7 milhão de avaliações de 150 piadas feitas por 63.974 usuários.

As duas bases Jester possuem avaliações contínuas entre -10.00 e +10.00, inclusive. A primeira foi coletada entre Abril de 1999 e Maio de 2003, e a segunda entre Novembro de 2006 e Maio de 2009, após a adição de 50 novas piadas ao sistema.

Outras bases são largamente utilizadas em sistemas de recomendação, como Book-Crossing [3] (base de livros) [50] e do concurso NetFlix [4] (outra base de filmes). A primeira

[1] `http://www.grouplens.org/node/73`
[2] `http://eigentaste.berkeley.edu/dataset`
[3] `http://www.informatik.uni-freiburg.de/~cziegler/BX/`
[4] `http://www.netflixprize.com/download`

possui avaliações explícitas e implícitas, o que oferece um conjunto extra de informações para sistemas que utilizem ambas. A segunda, uma base muito grande se comparada a todas as outras (mais de 100 milhões de avaliações para 17770 filmes feitas por 480189 usuários), foi muito utilizada em trabalhos recentes em recomendação, mas foi descontinuada e não está mais disponível [5].

Para esse trabalho, foi utilizada a base *100k Ratings Data Set* do Movielens, por ser uma das bases mais clássicas na avaliação de sistemas de recomendação. Além dela, um subconjunto da *Dataset 1* do Jester também foi gerado para avaliação e possível confirmação de resultados em uma segunda base.

5.2.2 Caracterização de usuários e itens

Para que os dados de usuários e itens possam ser utilizados na arquitetura proposta, é necessário caracterizá-los. A caracterização de entidades de um sistema de recomendação, apresentada na Seção 3.1, é uma maneira de representar tal entidade de maneira tratável pelo sistema de recomendação.

Nos experimentos realizados, tantos os usuários quanto os itens foram caracterizados respectivamente pelos itens que avaliaram e pelos usuários que os avaliaram, juntamente com o valor das avaliações.

O primeiro objetivo dessa abordagem é observar a robustez do modelo genérico de caracterização. Como classicamente apenas os usuários são caracterizados por seu perfil, torna-se interessante observar o comportamento do sistema de recomendação com a mesma caracterização aplicada a itens.

Em filtragem por conteúdo, o conteúdo dos itens é utilizado para caracterizá-lo. Em filtragem coletiva, os itens avaliados por um usuário (ou parte deles, ou ainda um modelo observado a partir deles) é utilizado para caracterizar esses usuários. Nos dois casos, há uma diferenciação explícita entre usuários e itens. Nesse trabalho, itens e usuários são tratados como entidades de mesma ordem no sistema de recomendação, e portanto, representações aplicadas a uma dessas entidades pode ser estendida à outra. Portanto, outro objetivo é analisar a representatividade do modelo proposto.

5.2.3 Distâncias

Em sistemas de resomendação são utilizadas diversas distâncias para medir a similaridade entre itens e perfis de usuários. Algumas das distâncias mais comuns são listadas na Tabela 5.1.

[5] `http://archive.ics.uci.edu/ml/datasets/Netflix+Prize`

Distância	Fórmula
L1	$d_{l1}(x,y) = \sum_{i=1}^{n} \lvert x_i - y_i \rvert$
L2	$d_{l2}(x,y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$
Coseno	$d_{cos}(x,y) = 1 - \frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} x_i^2 \sum_{i=1}^{n} y_i^2}}$
Jaccard	$d_{jac}(x,y) = 1 - \frac{\sum_{i=1}^{n} x_i y_i}{\sum_{i=1}^{n} x_i + \sum_{i=1}^{n} y_i - \sum_{i=1}^{n} x_i y_i}$
Pearson	$d_{\rho}(x,y) = 1 - \frac{\sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n} (x_i - \bar{x})^2 \sum_{i=1}^{n} (y_i - \bar{y})^2}}$

Tabela 5.1: Distâncias analisadas e suas respectivas fórmulas

O uso da distância Pearson é bastante comum [1, 6, 25]. Alguns trabalhos analisam o uso de outras distâncias, e em geral a distância Pearson é considerada a melhor [6], e também sua combinação com outras distâncias apresenta bons resultados [7].

Seguindo a literatura da área, a distância escolhida foi Pearson. Testes preliminares foram feitos e mostraram que para a caracterização de usuários e itens utilizada, essa distância realmente dá melhores resultados que outras, como L1, L2, Jaccard e Cosseno.

5.2.4 Métodos de agrupamento de dados

Foram utilizados 7 métodos de agrupamento de dados nos experimentos realizados. Dos 7 métodos, 6 estão implementados em bibliotecas da linguagem R [6]. A outra biblioteca utilizada foi a *LibOPF* [7], a implementação de referência do OPF, oferecida por seus criadores.

A nomenclatura utilizada na descrição dos resultados dos experimentos foi:

- *agnes (Agglomerative Nesting)* [23]: método aglomerativo que utiliza *Average-Link* para medir a similaridade entre os *clusters*, descrito na Seção 2.7.1;
- *clara (Clustering Large Applications)* [23]: método particional em que em cada iteração executa o *k-means* sobre um subconjunto do conjunto original de dados, e agrupa o resto dos itens ao *cluster* mais próximo;
- *diana (DIvisive ANAlysis Clustering)* [23]: método divisivo, apresentado na Seção 2.7.1;
- *fanny (Fuzzy Analysis Clustering)* [23] método particional baseado no *k-means* que utiliza uma representação nebulosa dos *clusters*;

[6] `http://www.r-project.org/`
[7] `http://www.ic.unicamp.br/~afalcao/libopf/`

Métrica	Fórmula
MAE	$\text{MAE}(x,y) = \frac{\sum_{i=1}^{n}\lvert x_i - y_i\rvert}{n}$
RMSE	$\text{RMSE}(x,y) = \sqrt{\frac{\sum_{i=1}^{n}(x_i - y_i)^2}{n}}$

Tabela 5.2: Métricas utilizadas e suas respectivas fórmulas

- *kmeans (K-Means Clustering)* [23]: método de agrupamento clássico, apresentado na Seção 2.7.2;
- *hclust (Hierarchical Clustering)* [23]: método aglomerativo que utiliza *Single-Link* para medir a similaridade entre os *clusters* sendo formados, descrito na Seção 2.7.1;
- *opf (Optimum Path Forest)* [8]: método particional descrito na Seção 2.7.2.

5.2.5 Métricas

Para avaliar a predição de avaliações, as medidas MAE (*Means Absolut Error*) e RMSE (*Root Mean Squared Error*), citadas na Seção 4.5, são as medidas mais utilizadas [1]. Suas repectivas fórmulas são apresentadas na Tabela 5.2.

As duas métricas medem a qualidade de predições, mais precisamente, sua eficácia ou precisão. Dadas as predições e o *groundtruth*, o MAE é, como o nome diz, a média dos erros ou resíduos absolutos das predições. Dados as mesmas informações, o RMSE é a raiz da média dos erros das predições elevados ao quadrado.

As fórmulas das duas métrica são muito parecidas, mas geram medidas que carregam informações diferentes. Por exemplo, se os erros estiverem mal distribuídos, podemos ter um pequeno valor de MAE, mas um alto valor de RMSE. Dessa forma, as duas medidas são complementares, e ajudam a compreender melhor se as recomendações são boas ou não.

Nesse trabalho, as duas medidas foram utilizadas. No entanto, os gráficos de RMSE foram suprimidos. Em geral, e também nesse trabalho, os gráficos das duas medidas oferecem o mesmo tipo de informação visual.

5.3 Avaliação de eficácia

Alguns experimentos foram realizados para avaliar a eficácia do recomendador utilizando a caracterização de usuários e itens apresentada na seção 5.2.2 na base Movielens e na base

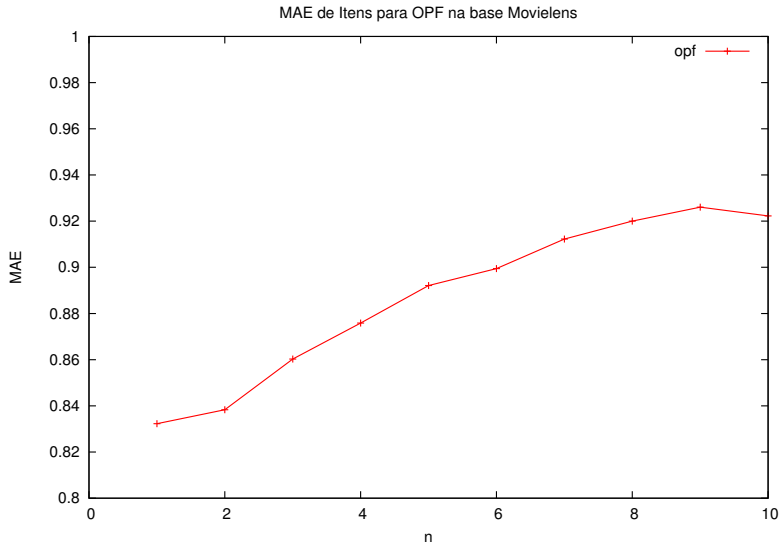

Figura 5.1: Gráfico de MAE em função de n do OPF para itens da base Movielens.

Jester. Foram avaliados vários métodos de agrupamento de dados, com seus parâmetros variados.

Foram utilizados os algoritmos de agrupamento de dados apresentados na seção 2.7. Para o OPF, o parâmetro variado foi o n, que determina quantos vizinhos serão utilizados na adjacência de cada vértice no algoritmo. Nos outros métodos, foi utilizado parâmetro k, que determina o número de *clusters* desejados.

A Figura 5.1 apresenta o gráfico do MAE para o agrupamento de itens utilizando o OPF. A Figura 5.2 apresenta o gráfico do MAE para o agrupamento de itens utilizando vários métodos de agrupamento. Os dois gráficos não foram agrupados pois a grandeza utilizada é diferente. No entanto, para facilitar a comparação dos resultados, o melhor resultado apresentado pelo OPF está indicado por uma reta vertical no gráfico das outras técnicas.

Para valores pequenos de k, o *k-means* é o melhor dos métodos, possuindo resultados muito melhores que os outros métodos até 200 *clusters*. Em torno de 1000 *clusters*, o métodos *hclust*, *agnes* e *k-means* chegam muito próximos aos melhores resultados obtidos, mantendo o valor tanto do MAE quanto do RMSE até 1400 a 1600, quando os resultados voltam a piorar novamente. Os métodos *agnes*, *diana*, *hclust* e *k-means* possuem os melhores resultados de MAE, todos apresentados quando k é grande.

No caso do OPF, a evolução das métricas é diferente porque quanto maior o valor de n, menos *clusters* tendem a aparecer. Da mesma forma que para os outros métodos, quando

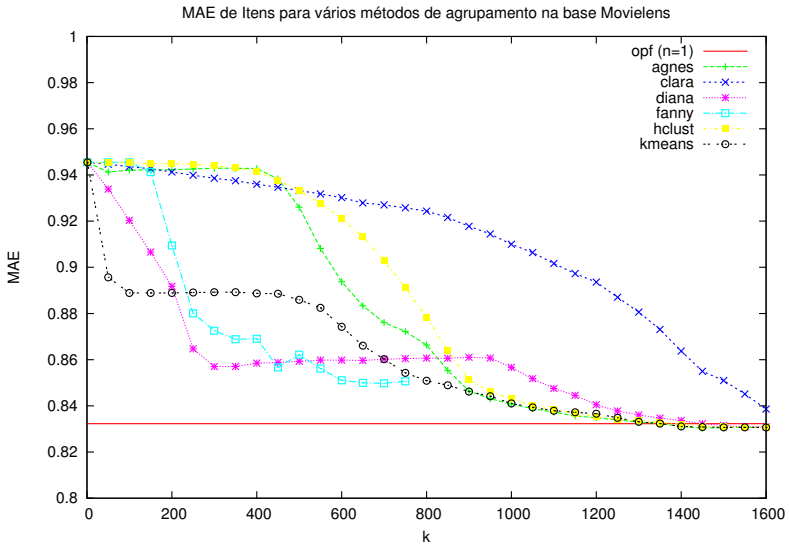

Figura 5.2: Gráfico de MAE em função de k de vários métodos de agrupamento para itens da base MovieLens.

mais clusters, melhor o resultado para o OPF, que apresenta um resultado próximo, porém um pouco pior, dos outros métodos de agrupamento. Apesar da aparente melhora dos resultados com $n = 10$, para valores maiores o gráfico volta a apresentar valores maiores; $n = 10$ é apenas um *outlier*.

A Tabela 5.3 apresenta os valores de k e n para os quais o MAE e RMSE foram mínimos.

A melhora dos resultados com o aumento do número de agrupamentos é esperada, pois a função utilizada para agregar as avaliações dos clusters é a média aritmética. Como a função é simples e não tem o poder de discriminar por si elementos pertencentes ao grupo, as recomendações ficam melhores com o aumento da coesão dos agrupamentos, dada pela diminuição de seu tamanho. Essa análise é sustentada também pelo aumento das medidas de avaliação quando o número de agrupamentos aumenta, pois quando há muitos agrupamentos unitários, perde-se a eficácia, já que não existem grupos.

A Figura 5.3 apresenta o gráfico do MAE para o agrupamento de itens utilizando o OPF. A Figura 5.4 apresenta o gráfico do MAE para o agrupamento de usuários utilizando vários métodos de agrupamento.

A forma das curvas de MAE e RMSE nesse caso é bastante diferente que para os itens, pois as curvas são mais suaves. A convergência se dá para um número de agrupamentos próximo de 600 para todos os métodos menos *clara*. Apesar dessa suavização, o mesmo

Método	k	MAE	k	RMSE
agnes	1450	0.8303	1450	1.0442
clara	1600	0.8386	1600	1.0512
diana	1600	0.8309	1550	1.0452
fanny	700	0.8498	700	1.0604
hclust	1600	0.8306	1500	1.0450
kmeans	1600	0.8306	1500	1.0450
opf	n=1	0.8323	n=1	1.0464

Tabela 5.3: MAE e RMSE do menor k para todos os algoritmos utilizados para agrupar itens na base Movielens

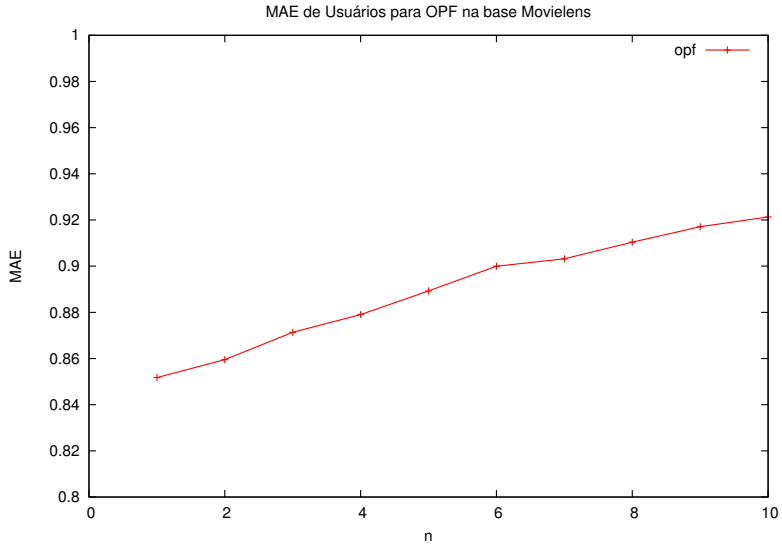

Figura 5.3: Gráfico de MAE em função de n do OPF para usuários da base Movielens.

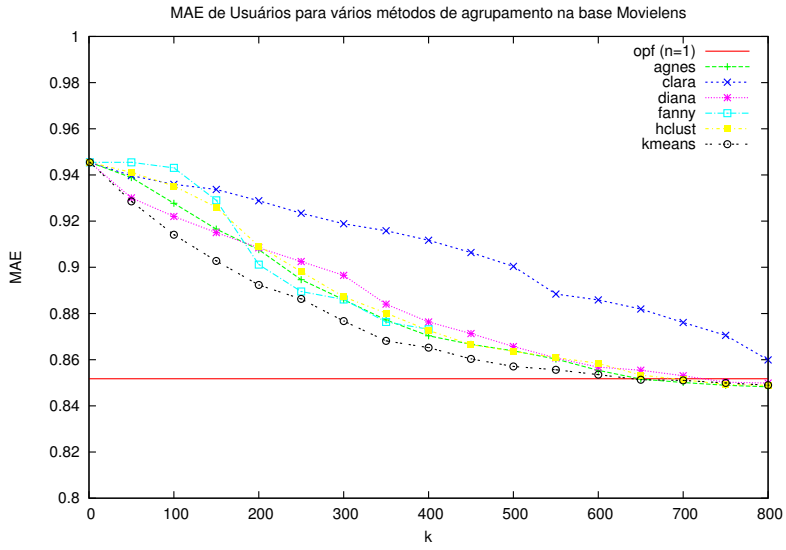

Figura 5.4: Gráfico de MAE em função de k de vários métodos de agrupamento para usuários da base MovieLens.

perfil de evolução é observado. Para o OPF, o melhor resultado acontece com $n = 1$, que é equivalente ao motivo pelo qual os outros métodos conseguem bons resultados com k em torno de 600: o maior número de clusters, proporcionando grupos mais coesos.

A Tabela 5.3 apresenta os valores de k e n para os quais o MAE e RMSE foram mínimos.

Como observado nos resultados para itens na mesma base, o melhor resultado se dá com um número grande de clusters, mas não tão grande que torne a maioria dos clusters unitários.

Na base Jester, os resultados foram similares. A Figura 5.5 apresenta o gráfico do MAE para o agrupamento de itens utilizando o OPF. A Figura 5.6 apresenta o gráfico do MAE para o agrupamento de itens da base Jester utilizando os vários métodos de agrupamento. Da mesma forma que no experimento na base Movielens, os dois gráficos não podem ser agrupados pois os parâmetros variádos são diferentes.

Para $k < 70$, os métodos *k-means* e *hclust* se alternam como o melhor método, com predominância do *k-means*. Entre $k = 70$ e 80, o método *agnes* é melhor. A partir desse valor de k, *agnes*, *clara*, *diana* e *k-means* chegam muito próximos do melhor valor de MAE para esses algoritimos. Nesse caso, como o k máximo é pequeno e a amplitude de valores de avaliação para a base grande (20, uma vez que as avaliações variam entre -10.00 e 10.00), a diferença entre o melhor e o pior valor para todos os métodos é de

Método	k	MAE	k	RMSE
agnes	800	0.8482	800	1.0568
clara	800	0.8599	800	1.0655
diana	800	0.8499	750	1.0585
fanny	400	0.8732	400	1.0768
hclust	800	0.8489	800	1.0574
kmeans	800	0.8490	800	1.0573
opf	n=1	0.8517	n=1	1.0602

Tabela 5.4: MAE e RMSE do menor k para todos os algoritmos utilizados para agrupar usuários na base Movielens

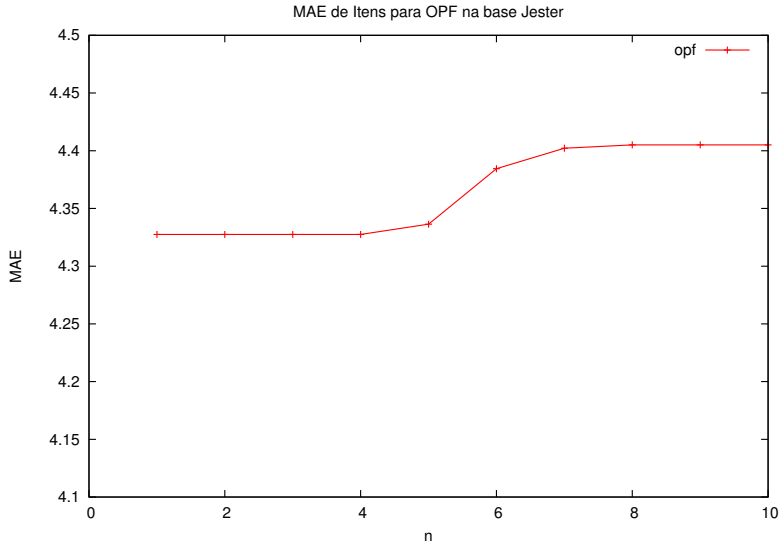

Figura 5.5: Gráfico de MAE em função de n do OPF para itens da base Jester.

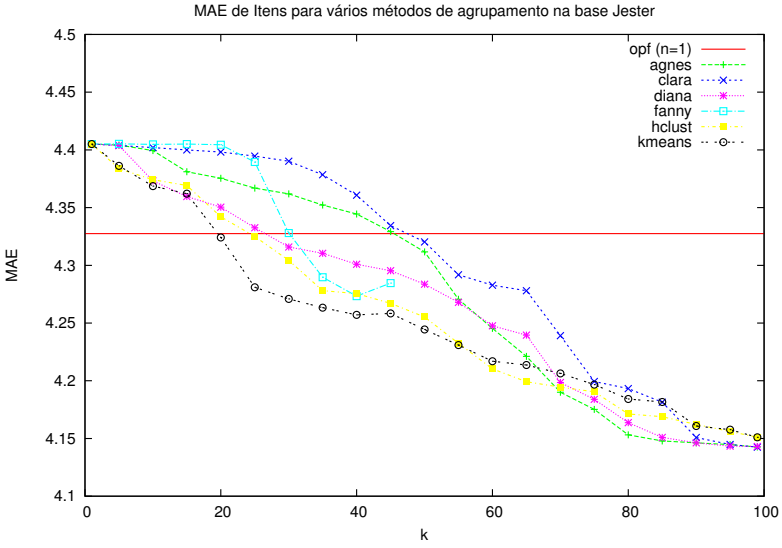

Figura 5.6: Gráfico de MAE em função de k de vários métodos de agrupamento para itens da base Jester.

aproximadamente 1.25%.

No caso do OPF, a evolução das métricas novamente tem um comportamento diferente, já que com o aumento de n menos grupos se formam. O resultado não varia muito nesse caso específico, mantendo o mesmo valor de $n = 1$ a 4, e depois mantendo o mesmo valor de $n = 7$ até 10. Essa pequena variação é reflexo do tamanho da base. Por ser muito pequena e muito diversa, os itens que são vizinhos na média nem adicionam sem reduzem informação.

Da mesma forma que na base Movielens, quando mais grupos, melhor o resultado obtido. Os resultados são bastante estáveis, variando muito pouco, se comparado com a base Movielens, tanto para usuários como para itens.

Os mesmos métodos de agrupamento de dados foram avaliados na base Jester, utilizando os o perfil dos usuários para o agrupamento. A Figura 5.7 apresenta o gráfico de MAE para o OPF. A Figura 5.8 apresenta o gráfico de MAE para a base Jester para os métodos *agnes*, *clara*, *diana*, *fanny*, *hclust* e *k-means*.

O comportamento dos métodos *agnes*, *clara*, *diana*, *fanny*, *hclust* e *k-means* é bastante linear e muito parecido, e a convergência acontece muito próxima de 2000. A base Jester original possui muitos usuários, e há usuários com avaliações bastante densas. Na base utilizada, um subconjunto bem pequeno da base original, a esparsidade aumentou consideravelmente. Dessa forma, a informação existente para criar o perfil do usuário é

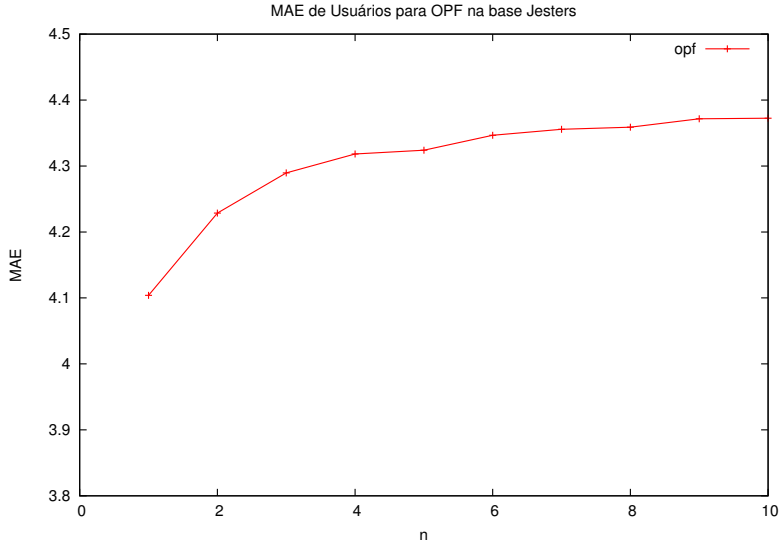

Figura 5.7: Gráfico de MAE em função de n do OPF para usuários da base Jester.

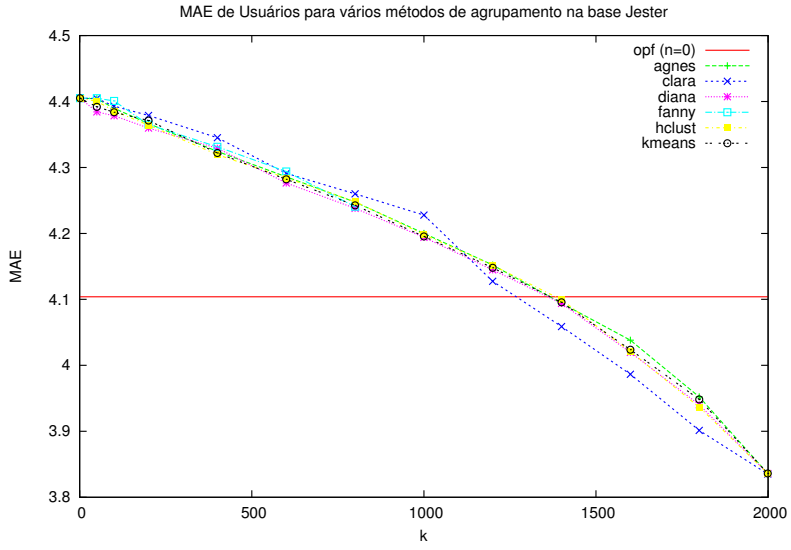

Figura 5.8: Gráfico de MAE em função de k de vários métodos de agrupamento para usuários da base Jester.

pouca, e os grupos formados vão melhorando consideravelmente com o aumento de k. A diferença apresentada entre o melhor resultado e o pior resultado nesse experimento foi a maior dentre todos os experimentos realizados, chegando a quase 3%.

Pelo mesmo motivo da queda sistemática do MAE para os outros métodos de agrupamento, o OPF possui um aumento bastante significativo do MAE com o aumento de n. Com $n = 2$, o número de grupos formados aumentou bastante, e em resposta a qualidade do resultado diminuiu.

A forma das curvas de MAE e RMSE na base Jester são bastante diferentes para itens e usuários, mas o comportamento das curvas de usuários e itens nas duas bases são bastante similares. Apesar de diferenças das bases, o comportamento é similar para os mesmos tipos de perfil. Como o tipo da informação codificada nos perfis de usuário e itens – respectivamente o tipo de item que interessa a um usuário e o tipo de usuário que gosta de um item – é similar, o comportamento esperado e observado é similar.

Em todos os casos, o OPF tem seu melhor resultado com $n = 1$, pois o número de grupos formados é menor. Os outros métodos em geral apresentam melhoras com valores grandes de k, e em geral convergem para o melhor valor de MAE e RMSE com valores próximos do tamanho da base, piorando um pouco do ponto de convergência em diante. Dentre esses métodos, o que teve o melhor desempenho para valores pequenos de k foi o *k-means*. *Clara* teve o desempenho menos consistente. E os métodos *agnes*, *diana*, *hclust* e *k-means* tiveram o melhor resultado no caso geral.

5.4 Avaliação de eficiência

Além de experimentos para avaliar a eficácia do modelo proposto utilizando vários métodos de agrupamento de dados, alguns experimentos foram realizados para avaliar também a sua eficiência. Foram utilizadas as mesmas caracterizações, distâncias e métodos de agrupamento de dados da seção anterior.

Para avaliar a eficiência, o tempo de execução de cada um dos métodos de agrupamento de dados utilizados na avaliação de eficácia foi analisado. Como a divisão em *folds* naqueles experimentos fragmenta as bases utilizadas, e eventualmente o número de usuários e itens presentes no agrupamento pode ser diferente, o que pode mudar de maneira significativa o tempo para execução do agrupamento. Para contornar esse problema, os algoritmos foram executados na base toda, e não apenas em um *fold*. Dessa forma, a variação dos resultados nas várias execuções são menores e os resultados são mais precisos.

O tempo de execução do OPF na base Movielens utilizando o perfil de itens é apresentado na Figura 5.9. Para esse método, o tempo cresce linearmente com o aumento de n. A variação do tempo medido para $n = 4$ foi a maior para esse experimento, o que corrobora para que o ponto seja considerado um *outlier*.

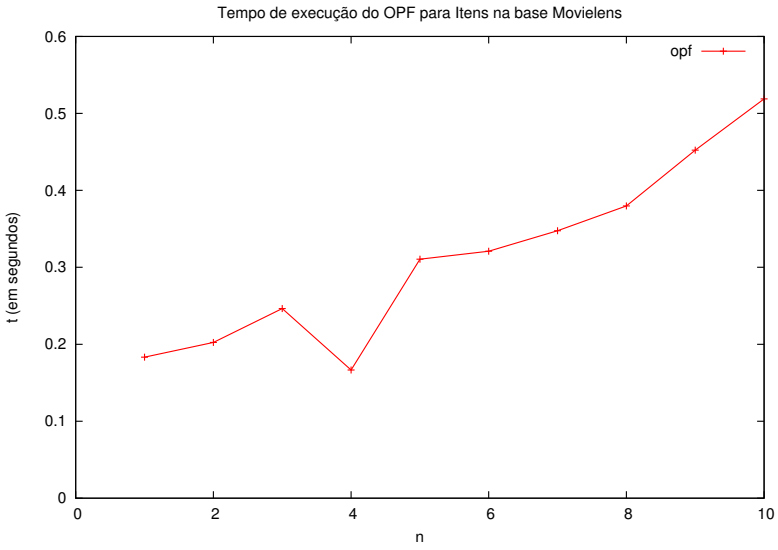

Figura 5.9: Gráfico de tempo em função de n do OPF para itens da base Movielens.

A Figura 5.10 apresenta o tempo de execução dos algoritmos *agnes*, *clara*, *diana*, *hclust* e *k-means* na base Movielens, utilizando o perfil de itens. O método *fanny* tem um comportamento temporal pouco consistente, devido à sua natureza nebulosa. O *k-means* é um dos métodos mais custosos em geral, seguido pelo *clara*. Esses dois métodos têm comportamento parecido, sendo menos custosos com k pequeno e grande, e mais custosos para valores intermediários. Os outros métodos têm tempos muito próximos, tendendo a gastar o mesmo tempo, não importando o valor de k, com os valores de tempo variando entre 10s e 60s para esses métodos. Pode-se observar, no entanto, que o menor tempo dos métodos apresentados na Figura 5.10 é mais de 50 vezes maior que o menor tempo do OPF nesse experimento.

O tempo de execução do OPF na base Movielens utilizando o perfil de usuários é apresentado na Figura 5.11. Da mesma forma que no experimento anterior, o tempo cresce linearmente com o aumento de n.

A Figura 5.12 apresenta o tempo de execução dos algoritmos que têm k como número de agrupamentos esperados na base Movielens, utilizando o perfil de usuários. O método *fanny* novamente apresentou um comportamento temporal pouco consistente, e *k-means* e *clara* também foram os métodos mais custosos, tendo novamente um comportamento temporal similar entre si e com relação ao experimento já apresentado. Os outros métodos também apresentam tempos muito próximos, variando de 5 a 12s, sem influência de k.

A diferença entre o tempo mínimo gasto pelo OPF e o menor tempo dos métodos

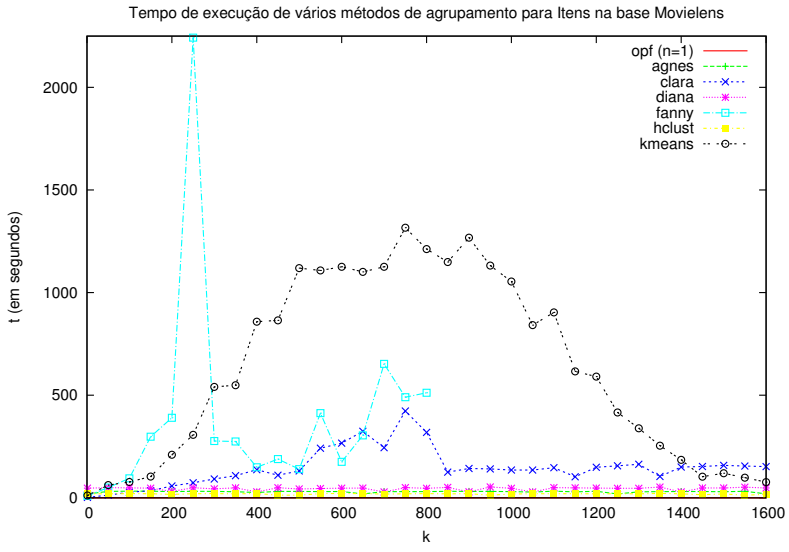

Figura 5.10: Gráfico de tempo em função de k de vários métodos de agrupamento para itens da base Movielens.

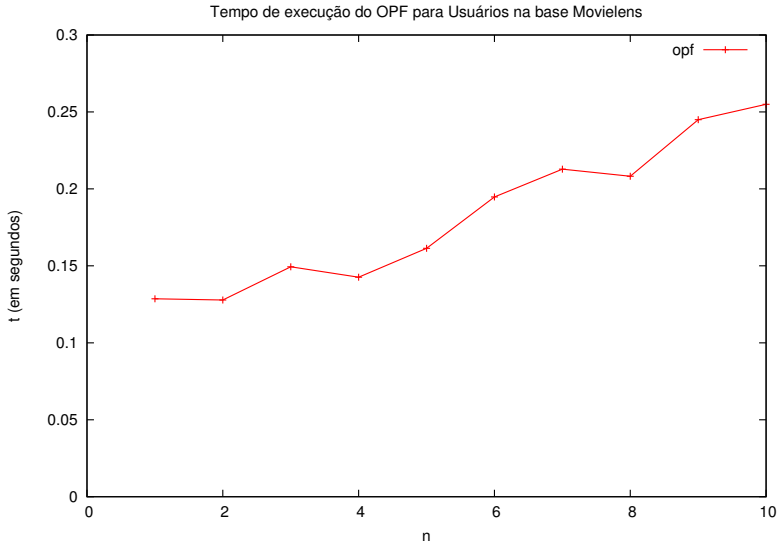

Figura 5.11: Gráfico de tempo em função de n do OPF para usuários da base Movielens.

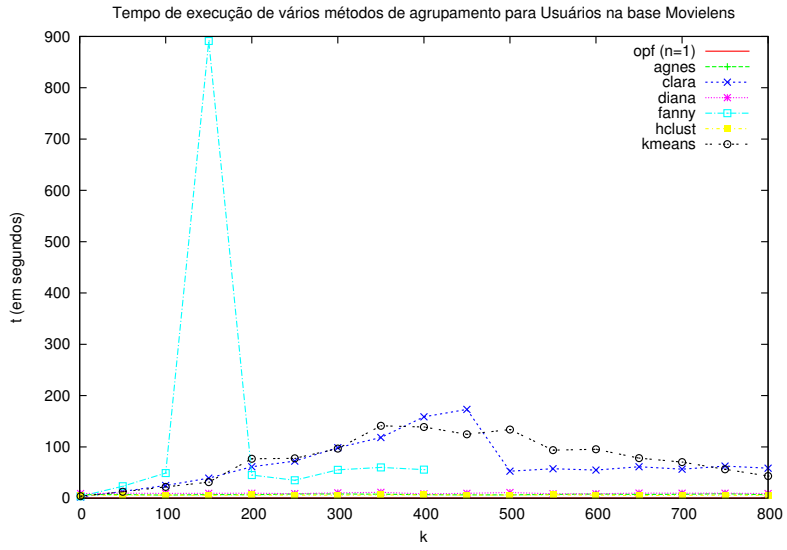

Figura 5.12: Gráfico de tempo em função de k de vários métodos de agrupamento para usuários da base Movielens.

apresentados na Figura 5.12 é também muito grande: OPF é mais de 30 vezes mais rápido nesse experimento, com o menor tempo na casa de 0.15s.

O comportamento temporal dos algoritmos é bastante diferente para itens na base Jester. A Figura 5.13 apresenta o tempo do OPF, e o tempo dos algoritmos *agnes*, *clara*, *diana*, *hclust* e *k-means* é apresentado na Figura 5.14.

O método *fanny* apresenta novamente um comportamento anômalo, com dois pontos completamente fora do esperado. O *clara* é o mais custoso nesse experimento, com seu tempo variando entre o dobro e o triplo dos outros métodos. Os outros métodos têm tempos muito próximos, todos menores que 0.5s, não variando em função de k.

O OPF possui um perfil bastante estranho e aparentemente fora do esperado. Porém, como os tempos são inferiores a 0.15s, uma parte considerável do tempo é gasta em troca de contexto, o que pode ter influenciado o comportamento apresentado. E novamente, o OPF foi mais rápido que todos os outros métodos, um pouco mais de 3 vezes o apresentado pelo algoritmos mais rápido nesse experimento. A diferença nesse caso foi muito menor que nos outros porque a base é bem menor.

O tempo gasto na execução dos mesmos métodos de agrupamento utilizando perfil de usuários na base Jester é apresentado na Figura 5.15 para o OPF e na Figura 5.16 para os métodos *agnes*, *clara*, *diana*, *hclust* e *k-means*.

O método *fanny*, como em todos os outros caso, teve um comportamento bastante dife-

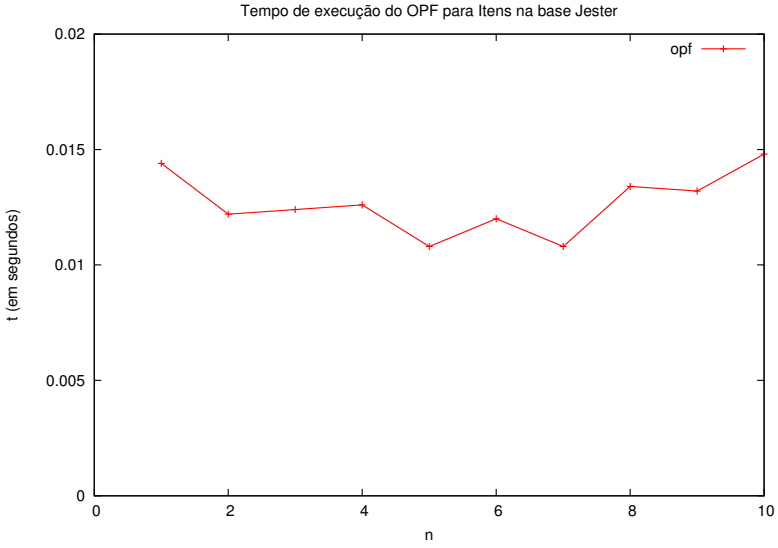

Figura 5.13: Gráfico de tempo em função de n do OPF para itens da base Jester.

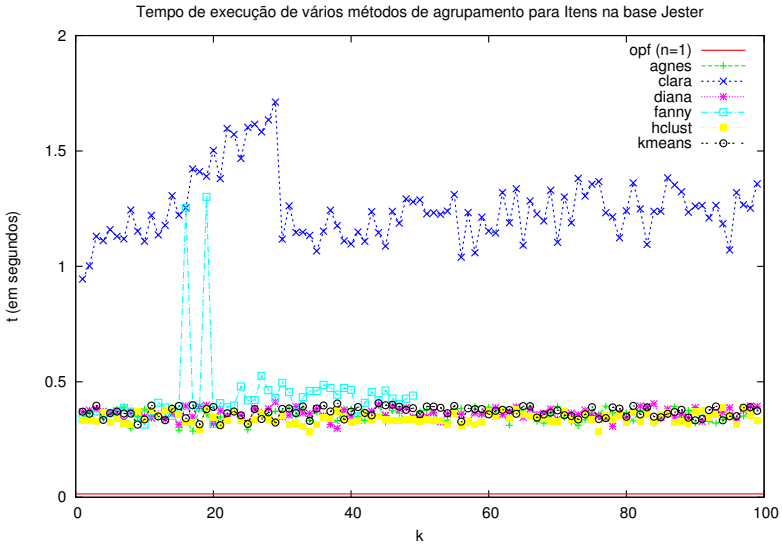

Figura 5.14: Gráfico de tempo em função de k de vários métodos de agrupamento para itens da base Jester.

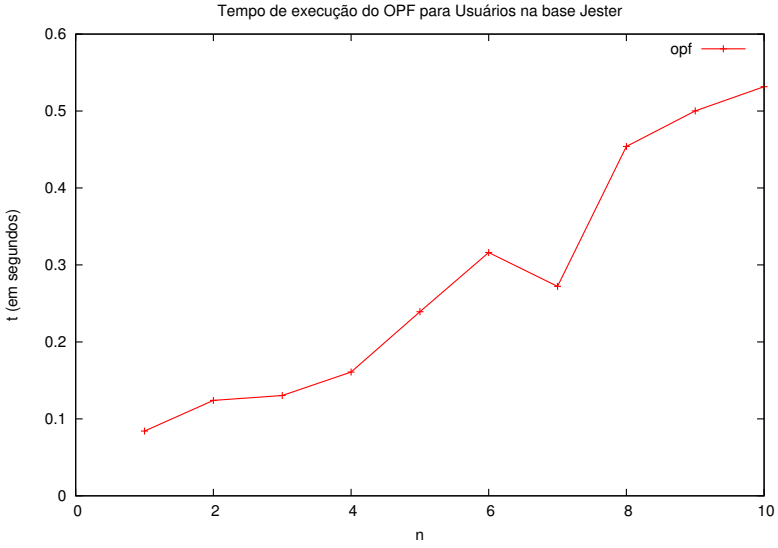

Figura 5.15: Gráfico de tempo em função de n do OPF para usuários da base Jester.

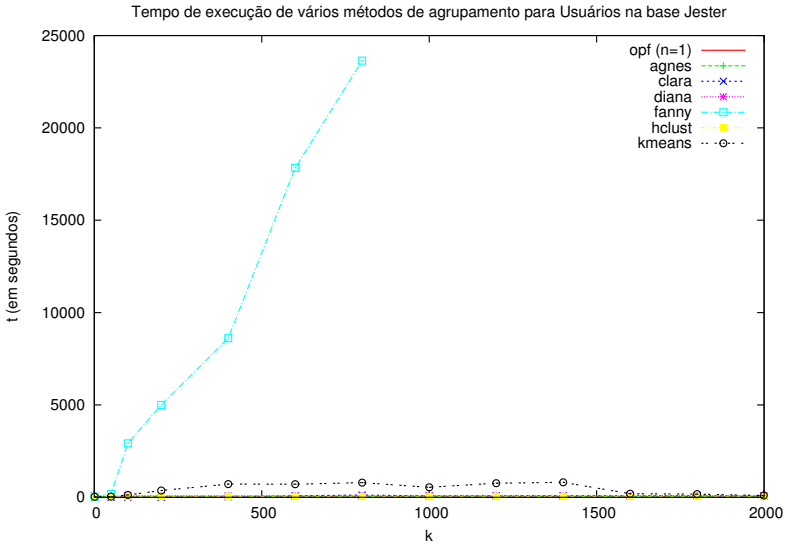

Figura 5.16: Gráfico de tempo em função de k de vários métodos de agrupamento para usuários da base Jester.

rente, nesse caso sendo muito pior que os outros métodos. Esse comportamento especial se deu pelo aumento no tamanho da base. O *k-means* apresentou o mesmo comportamento temporal que nos outros experimentos em bases grandes, com tempo maior para valores intermediários de k e tempo pequeno para os extremos. Os outros métodos tiveram um comportamento temporal similar, com tempos variando de 0.5 a 120s, não variando em função de k.

O OPF apresentou o mesmo comportamento de outros experimentos, aumentando linearmente em função de n. Da mesma forma que nos outros experimentos, foi observada uma queda no tempo para $n = 7$, que também mostrou uma variação muito grande nas várias execuções do algoritmo, caracterizando outro *outlier*. A execução mais rápida se deu com $n = 1$, com tempo inferior a 0.1s, 1/5 da execução mais rápida dos outros métodos.

Analisando apenas o tempo, o OPF foi consistentemente o método mais rápido. Dos outros métodos, *agnes*, *diana* e *hclust* foram os que tiveram um comportamento consistente, invariável com relação a n, muito próximo e comparativamente rápidos, se avaliados contra os outros métodos mais lentos, como o *k-means*.

5.5 Avaliação de conjunta de eficácia e eficiência

As seções 5.3 e 5.4 mostram experimentos executados para avaliar individualmente eficiência e eficácia. Na prática é no entanto é necessário fazer uma análise conjunta dessas duas métricas. Com o aumento de uma, é possível que a outra sofra algum tipo de alteração. E ainda é possível que seja desejado em uma aplicação real que o sistema seja mais rápido ou mais eficaz. De qualquer maneira, é necessário saber se essas duas medidas interferem entre si e qual o tamanho dessa interferência.

Para fazer essa análise, foram combinados os resultados obtidos nos experimentos das seções anteriores. Com tais dados, foram criados gráficos que cruzam informações sobre o tempo de execução dos métodos de agrupamento de dados e o sua eficácia para recomendação, para cada um dos parâmetros utilizados nos experimentos executados. O ponto ideal do gráfico é na origem: um valor baixo de MAE, ou seja, a menor taxa de erro possível, e um tempo baixo.

A Figura 5.17 apresenta o gráfico do MAE obtido para os vários métodos de agrupamento de dados utilizados na recomendação em função do tempo gasto por tais métodos na recomendação. A Figura 5.18 apresenta o mesmo gráfico ampliado para que o tempo dos métodos mais rápidos possam ser melhor analisados.

Como é possível observar, levando em consideração apenas o tempo, o OPF é o melhor método, mais de uma ordem de grandeza mais rápido que o mais rápido dos demais, como já observado em seções anteriores. Levando-se em consideração apenas a eficácia,

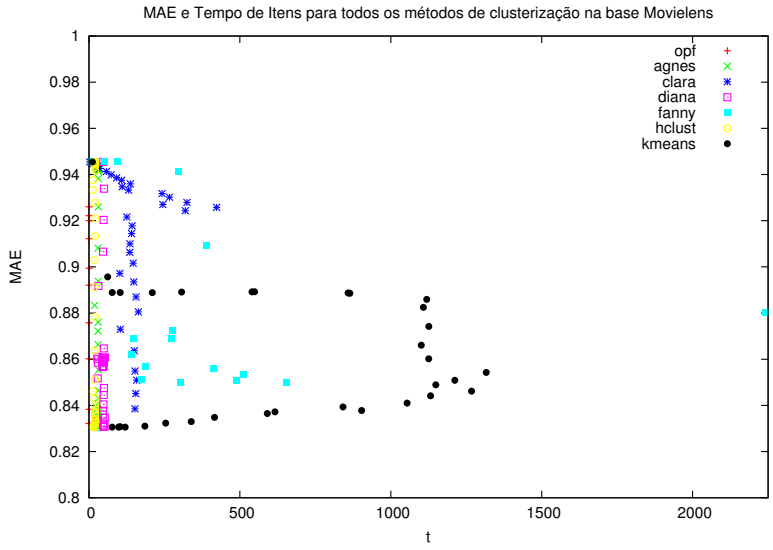

Figura 5.17: Gráfico de MAE e Tempo para todos os métodos de agrupamento avaliando Itens na base Movielens

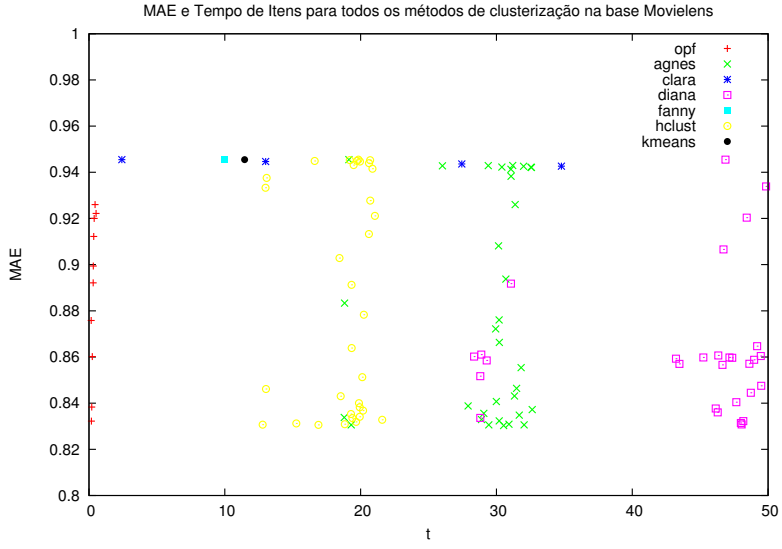

Figura 5.18: Versão ampliada do gráfico de MAE e Tempo para todos os métodos de agrupamento avaliando Itens na base Movielens

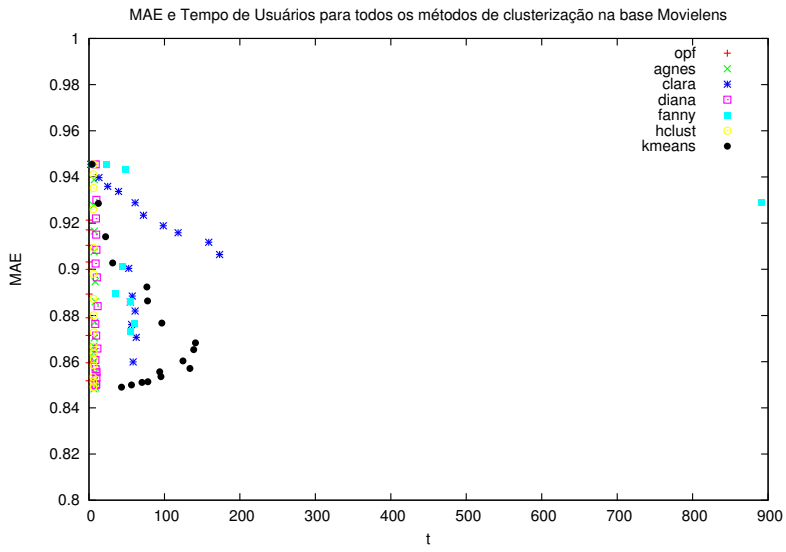

Figura 5.19: Gráfico de MAE e Tempo para todos os métodos de agrupamento avaliando Usuários na base Movielens

praticamente todos os métodos ficam próximos. No entanto, observando o método mais vantajoso nas duas avaliações, o OPF é melhor. Como é muito mais rápido e ainda tem um desempenho equivalente, seria preferível tanto em situações equilibradas, em que sejam necessárias tanto eficácia quanto eficiência, quanto em situações em que se deseje um método mais rápido. O OPF não apresentou o melhor resultado em termos de eficácia nesse caso, mas a diferença foi muito pequena.

A Figura 5.19 apresenta o gráfico do MAE em função do tempo gasto pelos vários métodos de agrupamento utilizados na recomendação utilizando dados de usuários da base Movielens. A Figura 5.18 apresenta o mesmo gráfico ampliado para que o tempo dos métodos mais rápidos possam ser melhor analisados.

Levando-se em consideração apenas o tempo, o OPF é o novamente o melhor método, e também novamente mais de uma ordem de grandeza mais rápido que o mais rápido dos demais. Levando-se em consideração apenas a eficácia, todos os métodos ficam próximos muitos próximos, também repetindo a analise anterior. Mas ao observar as avaliações conjuntamente, o OPF é novamente o melhor, sendo muito mais rápido e com eficácia semelhante. O OPF também não apresentou o melhor resultado em termos de eficácia nesse caso, mas a diferença foi muito pequena.

A Figura 5.21 apresenta o gráfico semelhante aos anteriores para o MAE e tempo para os métodos de agrupamento utilizados no experimentos que utilizou itens da base Jester.

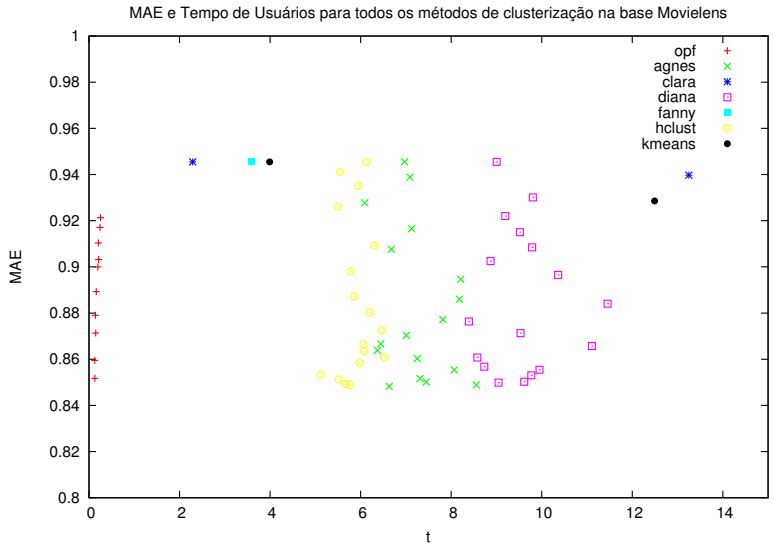

Figura 5.20: Versão ampliada do gráfico de MAE e Tempo para todos os métodos de agrupamento avaliando Usuários na base Movielens

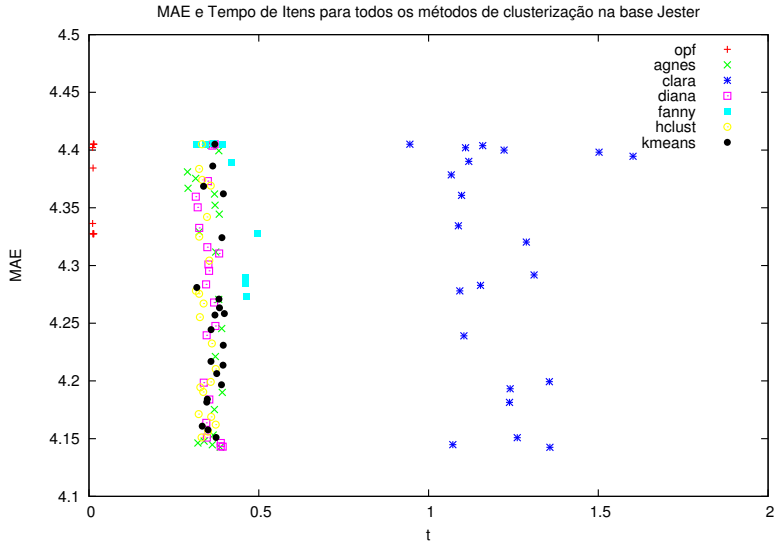

Figura 5.21: Gráfico de MAE e Tempo para todos os métodos de agrupamento avaliando Itens na base Jester

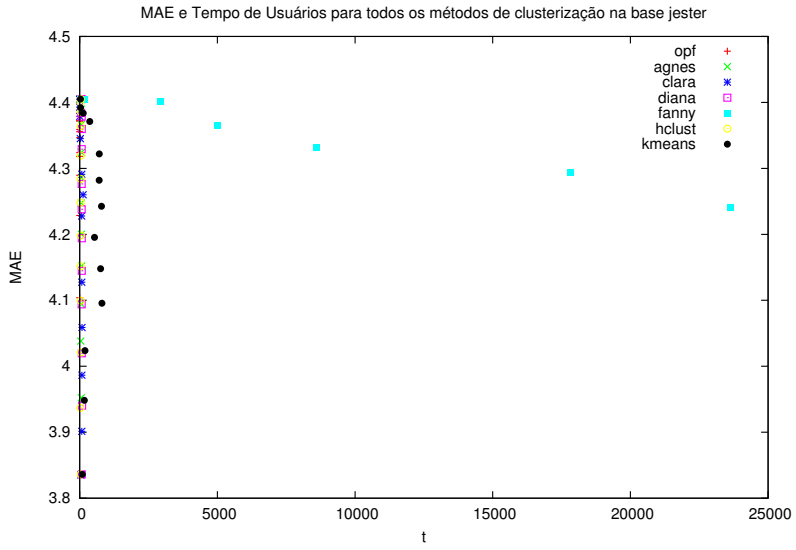

Figura 5.22: Gráfico de MAE e Tempo para todos os métodos de agrupamento avaliando Usuários na base Jester

Da mesma forma que nos casos anteriores, o OPF é muito mais rápido que os demais, porém nesse caso, não tem resultados em termos de eficácia semelhante. Do ponto de vista conjunto, é difícil fazer uma análise não tendenciosa. Se o tempo for uma restrição maior que eficácia, o OPF continua sendo o melhor método. Porém, com a eficácia sendo a restrição mais importante, o OPF perde espaço para ou outros métodos, que tem um desempenho semelhante tanto com relação ao tempo quanto em relação à eficácia.

A Figura 5.22 apresenta o gráfico do MAE em função do tempo gasto pelos vários métodos de agrupamento utilizados na recomendação utilizando dados de usuários da base Jester, e a Figura 5.18 apresenta o mesmo gráfico ampliado.

Nessa configuração há um resultado semelhante ao da análise com itens da mesma base. O OPF é novamente o mais rápido, mas não apresenta um resultado de eficácia tão bom quanto os outros métodos. Com restrições de tempo, continua sendo a melhor opção, porque é novamente mais de uma grandeza mais rápido que os outros métodos, mas em termos de eficácia, os outros métodos tem um desempenho melhor.

De maneira geral, o OPF é o método mais rápido, várias vezes mais rápido que todos os outros métodos. Na base Movielens, possui também um desempenho equivalente em termos de eficácia, sendo o algoritmo ideal para essa base. Na base Jester no entanto, sua eficácia fica abaixo dos outros métodos, tornando seu uso proibitivo em situações em que a restrição maior seja a eficácia. Ainda que não tenha um desempenho tão bom em

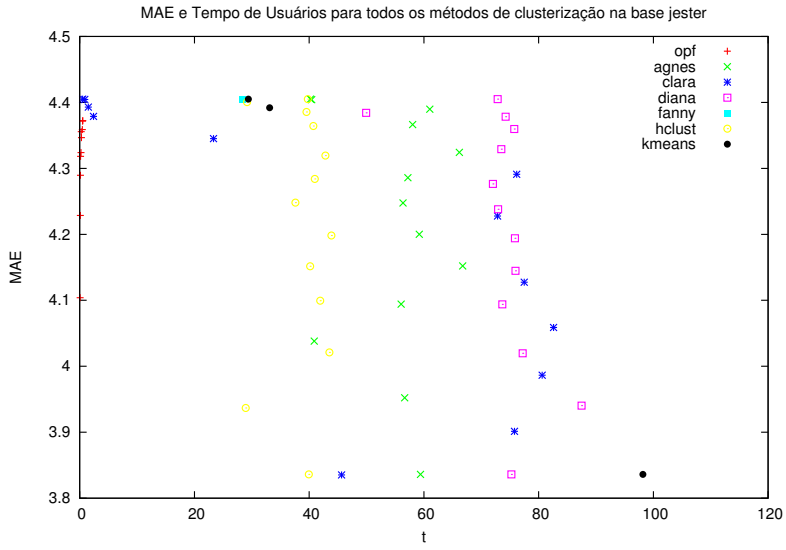

Figura 5.23: Versão ampliada do gráfico de MAE e Tempo para todos os métodos de agrupamento avaliando Usuários na base Jester

termos de eficácia na base Jester, seu tempo é ainda muito menor, fazendo dele uma ótima opção, quando as restrições de eficácia não forem tão estritas.

Capítulo 6

Conclusões

Devido à crescente quantidade de informações disponíveis, sistemas de recomendação são cada vez mais necessários, para ajudar usuários a lidar com tal grande volume de informação. A principal característica de sistemas de recomendação é a capacidade de aprender os interesses do usuário com o uso, e limitar a quantidade de itens que o usuário tenha que analisar para encontrar o que deseja.

Essa dissertação oferece um modelo genérico de recomendação que pode ser utilizado para a criação de sistemas de recomendação, oferecendo os moldes para a sua criação. Além disso, tal modelo serve a comparação de sistemas de recomendação (ou de parte deles) já existentes e futuros, por oferecer um modelo bem definido para o qual se pode mapear tais novos ou futuros sistemas.

Essa dissertação também oferece uma arquitetura de um sistema de recomendação bastante modular, que pode ser utilizada como base para a implementação de sistemas de recomendação, para sua avaliação e validação.

Uma implementação dessa arquitetura foi feita e mostrou sua viabilidade e generalidade.

Utilizando essa implementação, foi feita a avaliação de métodos de agrupamento de dados para recomendação em termos de eficiência e eficácia. Foi observado que o melhor método de agrupamento de dados para recomendação dentre os avaliados, avaliando simultaneamente eficácia e eficiência, é o OPF, por ser muito rápido e oferecer um resultado comparável ao de outras técnicas. Apesar desse resultado, algumas outras técnicas se mostraram mais eficazes quando o tempo não é um limitante.

Por fim, essa dissertação abre espaço para vários outros trabalhos na área de recomendação, seja pelo modelo, pela arquitetura ou pela implementação.

6.1 Contribuições

Em resumo, as principais contribuições desta dissertação são:

- definição de um modelo de recomendação baseado em grafos, até onde se sabe mais rico e mais genérico que os encontrados na literatura;
- especificação e implementação de uma arquitetura modular de um sistema de recomendação baseada nesse modelo;
- implementação de um sistema de recomendação baseado nessa arquitetura, com enfoque em técnicas de agrupamento de dados;
- validação da arquitetura e do modelo de recomendação propostos;
- comparação da eficácia e da eficiencia de técnicas de agrupamento de dados para sistemas de recomendação.

O arcabouço proposto é mais rico e genérico que os encontrados na literatura em alguns aspectos:

- O arcabouço não foi construído para um determinado tipo de dado, e assim pode ser utilizado para resolver problemas de recomendação com dados de origens diversas;
- O arcabouço engloba todas as técnicas de recomendação conhecidas (filtragem colaborativa, filtragem por conteúdo e abordagens híbridas), de forma que pode-se implementar qualquer uma das técnicas, ou uma combinação delas, de maneira trivial;
- Parte do arcabouço descrito é uma especialização para utilizar agrupamento de dados como base para fazer recomendações; no entanto, é possível fazer novas especializações, ou adaptar as existentes, para utilizar outros mecanismos e métodos para fazer recomendações;
- O modelo proposto não se limita, em nenhum aspecto, a natureza, origem, tipo, forma, estrutura ou qualquer outro aspecto ligado aos dados e suas ligações, assim é possível utilizá-lo para diversos tipos diferentes de dados, em diversas circunstancias;
- Os algoritmos propostos são apenas estruturações lógicas para tratamento dos dados para recomendação, com algumas lacunas, onde são tratados os dados e realmente geradas as recomendações; definindo de maneiras diversas tais lacunas, é possível criar muitos recomendadores diferentes.

6.2 Extensões e trabalhos futuros

Várias extensões podem ser feitas a este trabalho. Dentre as quais, destacam-se:

- Extensão do modelo de recomendação para utilizar outras técnicas de aprendizado de máquina para recomendar, como classificação;
- Extensão do modelo genérico para lidar diferentemente com avaliações implícitas e explícitas;
- Adaptação do modelo para uso de contexto nas recomendações;
- Mapeamento dos módulos da arquitetura do recomendador para componentes de um sistema de alto desempenho.

Além de extensões dos modelos e da arquitetura propósta, várias melhorias podem ser implementadas e avaliadas:

- Implementação do sistema de recomendação sobre uma arquitetura distribuída para computação de alto desempenho, com foco em eficácia, eficiência e principalmente escalabilidade dos módulos do sistema de recomendação;
- Implementação de novas técnicas de agrupamento de dados;
- Implementação de novos métodos de agregação para os *clusters* e para as recomendações.

Nessa dissertação, apenas uma pequena porção das possibilidades possíveis de avaliação foram feitas. Há espaço para muitas outras avaliações do modelo proposto:

- Avaliação de novas caracterizações de usuários e itens;
- Avaliação de novas distâncias;
- Avaliação de novos métodos de agrupamento de dados;
- Avaliação de novas funções de agregação;
- Avaliação de novos preditores e recomendadores.

Referências Bibliográficas

[1] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, 17(6):734–749, June 2005.

[2] Y. Bo and Q. Luo. Personalized web information recommendation algorithm based on support vector machine. *The 2007 International Conference on Intelligent Pervasive Computing*, pages 487–490, October 2007.

[3] J. S. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, pages 43–52, 1998.

[4] M. Brunato and R. Battiti. PILGRIM: a location broker and mobility-aware recommendation system. *Proceedings of the First IEEE International Conference on Pervasive Computing and Communications*, pages 265–272, March 2003.

[5] R. Burke. Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, 12(4):331–370, 2002.

[6] L. Candillier, F. Meyer, and M. Boullé. Comparing state-of-the-art collaborative filtering systems. In *MLDM*, volume 4571 of *Lecture Notes in Computer Science*, pages 548–562. Springer, 2007.

[7] L. Candillier, F. Meyer, and F. Fessant. Designing specific weighted similarity measures to improve collaborative filtering systems. In *ICDM*, volume 5077 of *Lecture Notes in Computer Science*, pages 242–255. Springer, 2008.

[8] F. A. M. Cappabianco, A. X. Falcão, and L. M. Rocha. Clustering by Optimum Path Forest and its application to automatic GM/WM classification in MR-T1 images of the brain. *Biomedical Imaging: From Nano to Macro, 2008. ISBI 2008. 5th IEEE International Symposium on*, pages 428–431, May 2008.

[9] R.-M. Chao, J.-T. Huang, and C.-W. Yang. The study of knowledge service-oriented recommendation mechanism - a case of e-learning platform. *Proceedings of 2005 International Conference on Machine Learning and Cybernetics*, 4:2228–2233 Vol. 4, August 2005.

[10] S. Debnath, N. Ganguly, and P. Mitra. Feature weighting in content based recommendation system using social network analysis. In *Proceeding of the 17th international conference on World Wide Web*, pages 1041–1042, New York, NY, USA, 2008.

[11] A. K. Dey. *Providing Architectural Support for Building Context-Aware Applications*. PhD thesis, College of Computing, Georgia Institute of Technology, 2000.

[12] K. Goldberg, T. Roeder, D. Gupta, and C. Perkins. Eigentaste: A constant time collaborative filtering algorithm. *Information Retrieval*, 4(2):133–151, July 2001.

[13] M. A. Gonçalves. *Streams, Structures, Spaces,Scenarios, and Societies (5S): A Formal Digital Library Framework and Its Applications*. PhD thesis, Virginia Polytechnic Institute and State University, November 2004.

[14] A. S. Harpale and Y. Yang. Personalized active learning for collaborative filtering. In *SIGIR '08: Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 91–98, New York, NY, USA, 2008.

[15] J. Herlocker, J. Konstan, A. Borchers, and J. Riedl. An algorithmic framework for performing collaborative filtering. In *Proceedings of the 1999 Conference on Research and Development in Information Retrieval*, August 1999.

[16] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems*, 22(1):5–53, 2004.

[17] R. Holmes and G. C. Murphy. Using structural context to recommend source code examples. In *ICSE '05: Proceedings of the 27th international conference on Software engineering*, pages 117–125, New York, NY, USA, 2005.

[18] Z. Huang, H. Chen, and D. Zeng. Applying associative retrieval techniques to alleviate the sparsity problem in collaborative filtering. *ACM Transactions on Information Systems*, 22(1):116–142, 2004.

[19] Z. Huang, W. Chung, and H. Chen. A graph model for e-commerce recommender systems. *Journal of the American Society for Information Science and Technology*, 55(3):259–274, 2004.

[20] Z. Huang, W. Chung, T.-H. Ong, and H. Chen. A graph-based recommender system for digital library. In *JCDL '02: Proceedings of the 2nd ACM/IEEE-CS joint conference on Digital libraries*, pages 65–73, New York, NY, USA, 2002.

[21] A. K. Jain. Data clustering: 50 years beyond k-means. *Pattern Recognition Letters*, 31(8):651 – 666, 2010. Award winning papers from the 19th International Conference on Pattern Recognition (ICPR), 19th International Conference in Pattern Recognition (ICPR).

[22] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM Comput. Surv.*, 31(3):264–323, 1999.

[23] L. Kaufman and P. J. Rousseeuw. *Finding Groups in Data An Introduction to Cluster Analysis*. Wiley Interscience, New York, 1990.

[24] O. B. Kwon. "I know what you need to buy": context-aware multimedia-based recommendation system. *Expert Systems with Applications*, 25(3):387–400, October 2003.

[25] H. Ma, I. King, and M. R. Lyu. Effective missing data prediction for collaborative filtering. pages 39–46, Amsterdam, The Netherlands, 2007. ACM.

[26] M. O'Connor and J. Herlocker. Clustering items for collaborative filtering. *Proceedings of ACM SIGIR Workshop on Recommender Systems*, August 1999.

[27] J. P. Papa, A. X. Falcão, P. A. V. Miranda, C. T. N. Suzuki, and N. D. A. Mascarenhas. Design of robust pattern classifiers based on optimum-path forests. In *Mathematical Morphology and its Applications to Signal and Image Processing (ISMM)*, pages 337–348. MCT/INPE, 2007.

[28] J. P. Papa, A. X. Falcão, and Celso T. N. Suzuki. Supervised pattern classification based on optimum-path forest. *International Journal of Imaging Systems and Technology*, 19(2):120–131, 2009.

[29] J. P. Papa, A. X. Falcão, and C. T. N. Suzuki. Supervised pattern classification based on optimum-path forest. Technical Report IC-08-20, Institute of Computing, University of Campinas, September 2008.

[30] J. P. Papa, A. A. Spadotto, A. X. Falcao, and J. C. Pereira. Optimum path forest classifier applied to laryngeal pathology detection. *Systems, Signals and Image Processing, 2008. IWSSIP 2008. 15th International Conference on*, pages 249–252, June 2008.

[31] D. C. G. Pedronette and R. da S. Torres. Uma plataforma de Serviços de Recomendação para Bibliotecas Digitais. In *VI Workshop de Teses e Dissertações em Bancos de Dados, XXII Simpósio Brasileiro de Banco de Dados*, pages 51–56, 2007.

[32] D. C. G. Pedronette and R. da S. Torres. Uma plataforma de serviços de recomendação para bibliotecas digitais. Master's thesis, UNICAMP, March 2008.

[33] D. C. G. Pedronette and R. da S. Torres. Uma plataforma de Serviços de Recomendação para Bibliotecas Digitais. In *XXIII Simpósio Brasileiro de Banco de Dados*, pages 253–267, 2008.

[34] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. Grouplens: an open architecture for collaborative filtering of netnews. In *CSCW '94: Proceedings of the 1994 ACM conference on Computer supported cooperative work*, pages 175–186, New York, NY, USA, 1994.

[35] L. M. Rocha, F. A. M. Cappabianco, and A. X. Falcão. Data clustering as an optimum-path forest problem with applications in image analysis. *International Journal of Imaging Systems and Technology*, 19(2):50–68, 2009.

[36] A. I. Schein, A. Popescul, L. H. Ungar, and D. M. Pennock. Methods and metrics for cold-start recommendations. In *SIGIR '02: Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 253–260, New York, NY, USA, 2002.

[37] C. Shahabi and Y.-S. Chen. An adaptive recommendation system without explicit acquisition of user relevance feedback. *Distributed and Parallel Databases*, 14(2):173–192, 2003.

[38] B. Shevade, H. Sundaram, and L. Xie. Modeling personal and social network context for event annotation in images. In *JCDL '07: Proceedings of the 7th ACM/IEEE-CS joint conference on Digital libraries*, pages 127–134, New York, NY, USA, 2007.

[39] Y.-Y. Shih and D.-R. Liu. Hybrid recommendation approaches: Collaborative filtering via valuable content information. *Proceedings of the 38th Annual Hawaii International Conference on System Sciences*, pages 1–7, January 2005.

[40] A. A. Spadotto, J. C. Pereira, R. C. Guido, J. P. Papa, A. X. Falcao, A. R. Gatto, P. C. Cola, and A. O. Schelp. Oropharyngeal dysphagia identification using wavelets and optimum path forest. *Communications, Control and Signal Processing, 2008. ISCCSP 2008. 3rd International Symposium on*, pages 735–740, March 2008.

[41] R. Torres, S. M. McNee, M. Abel, J. A. Konstan, and J. Riedl. Enhancing digital libraries with Techlens+. In *JCDL '04: Proceedings of the 4th ACM/IEEE-CS joint conference on Digital libraries*, pages 228–236, New York, NY, USA, 2004.

[42] M. van Setten, S. Pokraev, and J. Koolwaaij. *Context-Aware Recommendations in the Mobile Tourist Application COMPASS*, pages 235–244. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2004.

[43] W. Woerndl and G. Groh. Utilizing physical and social context to improve recommender systems. In *WI-IATW '07: Proceedings of the 2007 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology - Workshops*, pages 123–128, Washington, DC, USA, 2007.

[44] B. Wu, L. Qi, and X. Feng. Personalized recommendation algorithm based on SVM. *International Conference on Communications, Circuits and Systems*, pages 951–953, July 2007.

[45] Y.W. Wu, Q. Luo, M. Liu, Z.-H. Wu, and L.-Y. Wan. Research on personalized service system in e-supermarket by using adaptive recommendation algorithm. *2006 International Conference on Machine Learning and Cybernetics*, pages 4507–4510, Aug. 2006.

[46] K. Yu, A. Schwaighofer, V. Tresp, X. Xu, and H.-P. Kriegel. Probabilistic memory-based collaborative filtering. *IEEE Transactions on Knowledge and Data Engineering*, 16(1):56–69, 2004.

[47] K. Yu, V. Tresp, and S. Yu. A nonparametric hierarchical bayesian framework for information filtering. In *SIGIR '04: Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 353–360, New York, NY, USA, 2004.

[48] M. Zanin, P. Cano, J. M. Buldú, and O. Celma. Complex networks in recommendation systems. In *CEA'08: Proceedings of the 2nd WSEAS International Conference on Computer Engineering and Applications*, pages 120–124, Stevens Point, Wisconsin, USA, 2008.

[49] D. Zhou, S. Zhu, K. Yu, X. Song, B. L. Tseng, H. Zha, and C. L. Giles. Learning multiple graphs for document recommendations. *Proceedings of the 17th international conference on World Wide Web*, pages 141–150, 2008.

[50] C.-N. Ziegler, S. M. McNee, J. A. Konstan, and G. Lausen. Improving recommendation lists through topic diversification. In *Proceedings of the 14th International World Wide Web Conference (WWW '05)*, May 2005.