Universidade Federal de Campina Grande
Centro de Engenharia Elétrica e Informática
Coordenação de Pós-Graduação em Ciência da Computação

Uma Abordagem Centrada na Filtragem Colaborativa para Redução do Custo Computacional do Método *k-Nearest Neighbors*

Antonio Alexandre Moura Costa

Dissertação submetida à Coordenação do Curso de Pós-Graduação em Ciência da Computação da Universidade Federal de Campina Grande - Campus I como parte dos requisitos necessários para obtenção do grau de Mestre em Ciência da Computação.

Área de Concentração: Ciência da Computação
Linha de Pesquisa: Metodologia e Técnicas da Computação

Orientadores
Angelo Perkusich e Hyggo Almeida

Campina Grande, Paraíba, Brasil

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA CENTRAL DA UFCG

C837a Costa, Antonio Alexandre Moura.

Uma abordagem centrada na filtragem colaborativa para redução do custo computacional do método *k-Nearest Neighbors* / Antonio Alexandre Moura Costa. – Campina Grande, 2014.

74 f. : il. color.

Dissertação (Mestrado em Ciência da Computação) - Universidade Federal de Campina Grande, Centro de Engenharia Elétrica e Informática.

"Orientação: Prof. Dr. Angelo Perkusich, Prof. Dr. Hyggo Almeida".

Referências.

1. *K-Nearest Neighbors*. 2. Filtragem Colaborativa. 3. Desempenho. I. Perkusich, Angelo. II. Almeida, Hyggo. III. Título.

CDU 004.65 (043)

"UMA ABORDAGEM CENTRADA NA FILTRAGEM COLABORATIVA PARA REDUÇÃO DO CUSTO COMPUTACIONAL DO MÉTODO K-NEAREST NEIGHBORS"

ANTONIO ALEXANDRE MOURA COSTA

DISSERTAÇÃO APROVADA EM 26/06/2014

HYGGO OLIVEIRA DE ALMEIDA, D.Sc, UFCG
Orientador(a)

ANGELO PERKUSICH, D.Sc, UFCG
Orientador(a)

LEANDRO BALBY MARINHO, Dr., UFCG
Examinador(a)

MARCOS RICARDO ALCÂNTARA MORAIS, D.Sc, UFCG
Examinador(a)

CAMPINA GRANDE - PB

Resumo

Com o surgimento da *Web 2.0* o volume de informações disponíveis na Internet cresceu acentuadamente, tornando cada vez mais difícil para o usuário alcançar a informação desejada. Sistemas de recomendação surgem como uma alternativa a esse problema, sugerindo conteúdo personalizado. A filtragem colaborativa é uma das abordagens mais eficazes na área de recomendação. Dentre os algoritmos colaborativos, os modelos baseados em fatores latentes constituem o estado da arte na área. Entretanto, tais modelos não conseguem fornecer uma justificativa para o item recomendado, o que em determinados domínios pode tornar a recomendação desinteressante e facilmente ignorada pelo usuário. Diante desse contexto, uma alternativa interessante é o *k-Nearest Neighbors* (kNN), um método simples, popular e capaz de fornecer excelentes resultados. Essa técnica gera recomendações a partir das avaliações dos usuários mais similares (vizinhos mais próximos) ao usuário alvo. Apesar de sua eficácia, o kNN apresenta um custo computacional elevado ao ser executado em grandes bases de dados, tornando sua aplicação inviável em alguns domínios. Neste trabalho objetiva-se melhorar o desempenho do kNN a partir da restrição do espaço de busca dos vizinhos mais próximos. O método proposto utiliza uma heurística de seleção baseada na escolha dos usuários que mais avaliaram itens. Como resultado, constatou-se que utilizando apenas 15% dos usuários na busca dos vizinhos, consegue-se reduzir significativamente o custo computacional, porém mantendo alto nível de acurácia.

Abstract

With the emergence of Web 2.0 the volume of information available on the Internet has grown dramatically, becoming increasingly difficult for the user to achieve the desired information. Recommendation systems emerge as an alternative to this problem, suggesting personalized content. Collaborative filtering is one of the most effective approaches in the area of recommendation. Among the collaborative algorithms, latent factors models are the state of the art in the area. However, such models can not provide a justification for the recommended item, which in some areas can make the recommendation uninteresting and easily ignored by the target user. In this context, an interesting alternative is the k-Nearest Neighbors (kNN), a simple, popular and very robust method. This technique generates recommendations from ratings of the most similar users (nearest neighbors) to the target user. Despite its efficiency, kNN has a high computational cost when executed over large databases, making its application impractical in some domains. In this work we aim to improve the performance of kNN from the restriction of the search space of the nearest neighbors. The proposed method uses a user heuristic selection based on the choice of most rated items. As a result it was found that using only 15% of the neighbors searching space, it was possible to significantly reduce the computational cost, while maintaining high accuracy level.

Agradecimentos

Agradeço a Deus por me dar saúde e paz para conseguir vencer os desafios surgem em meu caminho.

Aos meus pais e irmãos por estarem sempre ao meu lado fornecendo todo o apoio e incentivo necessários.

À minha namorada Mithylenny por todo o amor, compreensão e companheirismo que me forneceu desde a graduação até dias atuais.

Aos colegas do laboratório, principalmente Felipe Barbosa e Reudismam Rolim, pelo auxílio durante toda jornada da pós-graduação.

Ao meus orientadores, Hyggo Almeida e Angelo Perkusich, que foram de fundamental importância para a concretização desse trabalho.

À CAPES pelos investimentos na pós-graduação.

Conteúdo

Lista de Símbolos

AG - *Algoritmo Genético*

EPG - *Eletronic Program Guide*

FBC - *Filtragem Baseada em Conteúdo*

FC - *Filtragem Colaborativa*

ILS - *Iterated Local Search*

IO - *Information Overload*

JSAT - *Java Statistical Analysis Tool*

kNN - *k-Nearest Neighbors*

LSH - *Locality Sensitive Hash*

MAE - *Mean Absolute Error*

MSE - *Mean Square Error*

PBE - *Particionamento Binário do Espaço*

RMSE - *Root Mean Square Error*

SR - *Sistema de Recomendação*

VG - *Vizinho Global*

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

O surgimento da *Web* 2.0 [45], termo criado em 2004 pela empresa *O'Reilly Media*[1], proporcionou uma mudança na forma como a Internet é percebida pelos usuários. Eles deixaram de ser apenas consumidores para se tornarem também produtores de conteúdo, chamados *prosumers* (do inglês *producer and consumer*). Tal mudança acentuou um problema conhecido como *Information Overload* [14, 20, 35], que sobrecarrega os usuários de opções, impossibilitando-os de escolherem a alternativa mais adequada aos seus interesses.

Diante desse contexto os sistemas de recomendação (SRs) [4, 31, 40, 48, 54] ganharam destaque e tornaram-se bastante atrativos, tanto para o setor produtivo quanto para o meio acadêmico. Esses sistemas reúnem técnicas computacionais com o intuito de fornecer itens personalizados (filmes, músicas, livros, etc) aos usuários. Grandes empresas como *NetFlix*[2], *Groupon*[3], *Facebook*[4], *Amazon*[5], dentre outras, utilizam métodos de recomendação em larga escala. Os SRs, geralmente, são divididos em três tipos:

- **Filtragem Baseada em conteúdo(FBC) [40, 43]:** gera recomendações a partir da comparação do perfil do usuário e do conteúdo/descrição dos itens.
- **Filtragem Colaborativa(FC) [10, 29, 57]:** recomenda com base em informações de outros usuários.

[1] www.oreilly.com
[2] www.netflix.com
[3] www.groupon.com
[4] www.facebook.com
[5] www.amazon.com

- **Filtragem Híbrida(FH) [5, 13, 15]:** corresponde a uma combinação de duas ou mais abordagens de recomendação.

A filtragem colaborativa vem se destacando como um dos métodos mais proeminentes na área de recomendação. Essa abordagem recomenda itens a partir das avaliações de usuários com interesses semelhantes ao usuário alvo da recomendação. A ideia da FC é que usuários semelhantes tendem a avaliar itens de forma similar. As avaliações podem ser explícitas, quando o usuário expressa de forma clara seu interesse pelo item, ou implícitas, quando o interesse do usuário é inferido a partir de seu comportamento. Uma das vantagens da FC é a independência sobre o conteúdo dos itens, enquanto que na FBC o processo de recomendação depende da extração do conteúdo, o que em alguns casos pode ser um processo custoso.

Atualmente, o estado da arte da área de recomendação é formado pelos modelos baseados em fatores latentes [3, 38, 42]. Os fatores latentes são variáveis que não são observadas diretamente, porém são inferidas de variáveis observadas. Elas podem corresponder a aspectos da realidade física, porém não é possível especificar quais aspectos são esses. Por exemplo, na área de recomendação fatores latentes podem corresponder ao gênero ou idade do usuário, pois são características podem influenciar na escolha do item, mas não se pode defini-las com certeza. Esses métodos estão entre as melhores técnicas de recomendação, pois conseguem fornecer excelentes resultados mesmo trabalhando com bases de dados grandes e esparsas. Entretanto, tais modelos possuem uma limitação ligada ao fato de não ser possível justificar a recomendação fornecida, ou seja, não há como explicar o porquê de um determinado item ter sido recomendado ao usuário, justamente pelo fato de não ser é possível definir a que estão relacionados os fatores latentes.

Diante desse contexto, o *k-Nearest-Neighbors* (kNN) [36, 39, 61] surge como uma boa alternativa. O kNN é uma técnica supervisionada para classificação de objetos a partir de instâncias mais próximas no espaço de características. Na área de recomendação esse método é bastante utilizado como uma técnica colaborativa para predição de notas (ou avaliações). Mesmo tendo perdido espaço para os modelos de fatores latentes, o kNN ainda é uma das abordagens mais utilizadas devido a sua simplicidade e à obtenção de ótimos resultados.

Umas das vantagens do kNN em relação aos modelos de fatores latentes, refere-se ao fato de ser possível expor ao usuário a origem da recomendação fornecida, uma vez que ela

é gerada a partir das informações dos k vizinhos mais próximos do usuário alvo. Alguns trabalhos [9, 27, 30, 60] mostram que há um aumento do interesse e aceitação da recomendação quando agrega-se uma justificativa ao item recomendado. Por exemplo, no domínio de compras coletivas muitos dos produtos e/ou serviços oferecidos provêm de novos estabelecimentos que almejam conquistar seus primeiros clientes. Do outro lado está o consumidor, indeciso em relação à aquisição da oferta de um estabelecimento com credibilidade ainda não reconhecida. Ao recomendar uma oferta, as chances de um usuário se interessar, provavelmente, serão maiores se ele souber que outros usuários próximos (por exemplo, amigos em sua rede social) também demonstraram interesse pela oferta. Diante disso, o kNN surge como a melhor opção.

1.1 Problemática

O kNN é um método simples e eficaz, porém seu desempenho está diretamente ligado ao tamanho da base de dados, isto é, quanto maior o volume da base, maior será o custo computacional para aplicá-lo. A elevação do custo é ocasionada, principalmente, pelos seguintes aspectos:

- **Requisição de Memória:** por ser uma técnica baseada em memória, o kNN necessita que todos os dados sejam armazenados na memória principal para a execução do algoritmo. Por exemplo, em uma base com 1.000.000 usuários (objetos) o algoritmo pode demandar até 3,7 Terabytes[6] de memória principal, apenas para armazenar os dados em uma Matriz de Similaridade (maiores detalhes no Capítulo 2).

- **Complexidade Computacional:** quanto maior o número de usuários maior a quantidade de cálculos de similaridade necessários.

- **Complexidade de Tempo:** para cada usuário alvo, uma vizinhança distinta é formada. O processo de formação de uma vizinhança consiste em verificar o grau de proximidade, de acordo com uma determinada métrica de similaridade, entre o usuário alvo e cada um dos demais usuários da base. Em seguida, selecionam-se os k usuários com

[6]Dada uma matriz NxN, onde $N = 10^6$ e cada célula de requer 4 bytes de memória, tem-se: $\frac{N^2}{4 \times 10^6} = 3,7\ Terabytes$

maior similaridade (vizinhos mais próximos) ao usuário alvo. Por fim, a classificação ou predição é gerada de acordo com a predominância da informação dos k vizinhos mais próximos.

Com a elevação do custo computacional torna-se mais difícil gerar recomendações que atendam a determinados critérios, pois há situações em que os itens precisam ser recomendados dentro de um dado intervalo de tempo. Por exemplo, atualmente o sistema operacional *Android* possui milhões de usuários ativos em todo o mundo, os quais têm acesso a mais de um milhão de aplicativos disponíveis no *Google Play* (Figura 1.1) [25]. Alguns aplicativos podem ser baixados gratuitamente, enquanto que outros são pagos. Imediatamente após o *download* o usuário pode atribuir uma nota (de 1 a 5) ao *app*, indicando seu grau de satisfação. É comum que *apps* pagos entrem em promoção e passem a ser oferecidos com preço reduzido ou até mesmo gratuitamente. Nesses casos, é importante que as recomendações que envolvem esses itens cheguem aos usuários antes do término da promoção, caso contrário elas perdem seu valor. Dessa forma, é desejável que tais recomendações sejam criadas a partir do momento em que os itens entram em promoção, ou pelo menos o mais próximo disso, pois assim é possível alcançar um maior nível de acurácia.

Gerar recomendações a partir de bases com milhões de usuários e itens é um processo custoso. Para tentar dirimir tal problema, é comum que as recomendações sejam criadas e armazenadas para que posteriormente sejam entregues aos usuários. Porém, esta prática pode ocasionar recomendações menos acuradas, pois quando são geradas em um tempo t e entregues em $t + x$, os dados surgidos no intervalo de tempo x deixam de ser utilizados, o que pode representar uma perda significativa dependendo da velocidade em que novos dados surgem na base. Por exemplo, o número de *downloads* de aplicativos no *Google Play* vem crescendo acentuadamente nos últimos anos. É possível observar na Figura 1.2 [34] que de 2010 a 2013 houve um crescimento de 5.000%. Somente de maio a julho de 2013 foram realizados cerca de 2 bilhões de *downloads*, o que corresponde a uma de média 33,3 milhões de *apps* baixados por dia. No cenário colaborativo há uma tendência dos usuários avaliarem apenas uma pequena quantidade de itens sobre o total ao qual tiveram acesso, porém mesmo diante de tal comportamento o número de avaliações ainda pode ser muito alto devido à elevada taxa de *downloads* diários. Portanto, mesmo em intervalos curtos de tempo é possível haver uma perda significativa de dados.

A *Netflix* é uma empresa norte-americana que aluga filmes e séries de TV via Internet, mediante o pagamento de mensalidade por parte de seus assinantes. Atualmente a companhia possui mais de 44 milhões de membros em 41 países que assistem a mais de um bilhão de horas de filmes e séries de TV por mês [53]. O sistema da empresa permite que seus usuários avaliem filmes e séries para que posteriormente venham a receber itens personalizados. O domínio de filmes é bastante popular na área de recomendação e possui características semelhantes ao do *Google Play*. O *GroupLens Research* [7], grupo de pesquisa da universidade de *Minnesota* nos Estados Unidos, é responsável por um *website* de recomendação de filmes chamado *MovieLens*, no qual parte de seus dados foram divididos e disponibilizados para pesquisa. Duas de suas bases[8] são bastante recorrentes em trabalhos da área. A primeira (*MovieLens* 100K) contém aproximadamente 100.000 avaliações (de 1 a 5), 943 usuários e 1.682 itens. A segunda (*MovieLens* 1M) conta com cerca de 1.000.000 de avaliações (de 1 a 5), 6.040 usuários e 3.952 itens. Existe ainda uma terceira base que possui 10 milhões de avaliações (de 1 a 5), 71.567 usuários e 10.681 itens, porém seu uso não é comum, provavelmente, devido ao maior poder computacional necessário para processamento de todo o conjunto de dados. Por exemplo, apenas para armazenar as similaridades entre os usuários podem ser demandados até 19,08 Gigabytes de memória principal.

Testes realizados utilizando o kNN padrão[9] para predição de notas mostram que para gerar predições para 10% das avaliações das bases *MovieLens* 100K e 1M são necessários aproximadamente 3 e 265 segundos, respectivamente. Na prática, os poucos mais de 4 minutos usados para executar o algoritmo na *MovieLens* 1M não refletem uma real situação, pois a quantidade de dados utilizados pelos sistemas de recomendação de grande empresas tende a ser significativamente maior. Entretanto, é importante destacar que o aumento de tempo observado nos testes, anteriormente citados, serve como um indicador de elevação do custo computacional, pois verifica-se que com o acréscimo de apenas alguns milhares de usuários e itens há um aumento de 8.833% no tempo total de execução. Analisando os conjuntos de dados (*MovieLens* 100K e 1M), tem-se uma diferença de $\times 6{,}4$ para os usuários e $\times 88{,}3$ para o tempo. Considerando que a razão de crescimento dos dados se mantenha constante

[7] www.grouplens.org

[8] Maiores detalhes sobre essas bases de dados podem ser visualizados no Capítulo 5

[9] O termo kNN padrão ou tradicional, neste documento, refere-se ao kNN utilizado na filtragem colaborativa para predição de notas, no qual o espaço de busca dos vizinhos é composto por 100% dos usuários

e tomando como base o número de usuários, estima-se que em um conjunto com 1 milhão de usuários seja necessário 1 semana para executar o kNN. Dessa forma, em domínios como o do *Google Play*, no qual itens podem assumir o status temporário "em promoção", seria inviável executar novamente o método para fornecer recomendações mais acuradas, pois o período promocional já estaria encerrado antes do fim da nova execução.

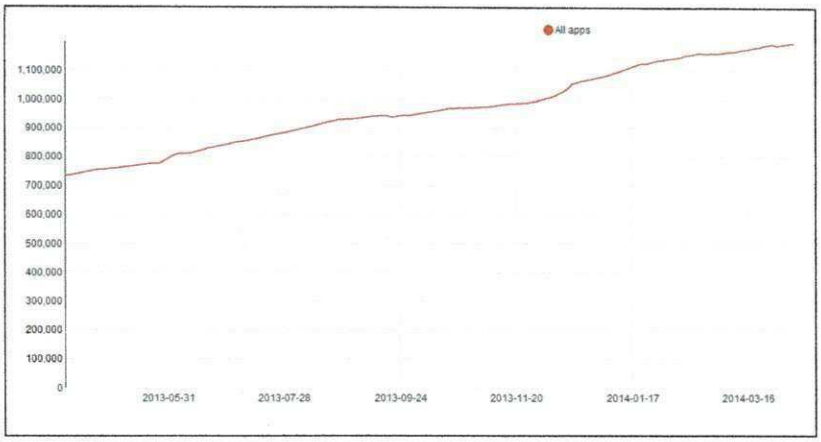

Figura 1.1: Número de Aplicativos Disponíveis no Google Play

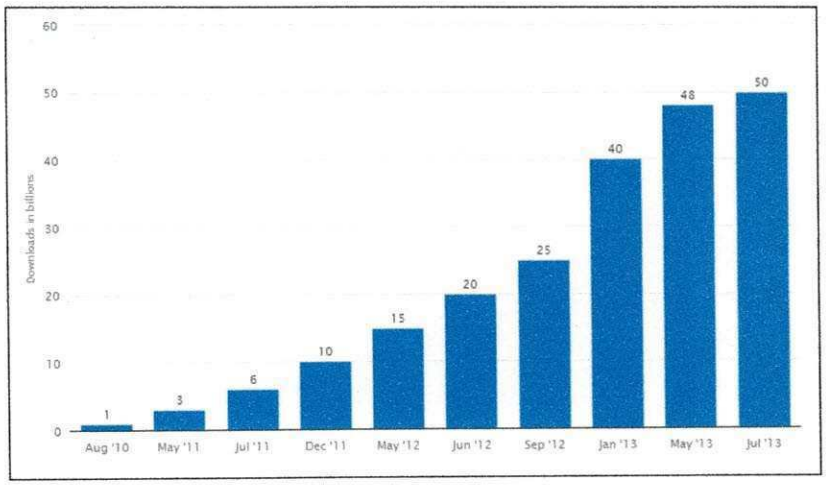

Figura 1.2: Número de Downloads Cumulativos no Google Play

1.2 Objetivos

Neste trabalho objetiva-se propor um método centrado na filtragem colaborativa, derivado do kNN com o intuito de melhorar seu desempenho, diminuindo o custo computacional. Para

isso, utiliza-se o conceito de redução de dados a partir da restrição do espaço de busca dos vizinhos mais próximos.

A validação é realizada através de um experimento, utilizando bases de dados reais da área de recomendação, para avaliar a solução proposta comparando-a com outras abordagens.

Os seguintes objetivos específicos são definidos:

- Reduzir a quantidade de cálculos de similaridade necessários;
- Reduzir o tempo de busca dos vizinhos mais próximos;
- Reduzir a quantidade de memória demandada pelo método;
- Manter o nível de acurácia satisfatório (próximo daquele fornecido pelo kNN tradicional)

1.3 Relevância

Mister se faz destacar que mesmo após o surgimento dos modelos de fatores latentes, a utilização do kNN ainda se faz necessária e apresenta-se como uma ótima opção na área de recomendação, dentre outros aspetos, devido à possibilidade de fornecer recomendações justificadas para aumentar a aceitação dos usuários. A exploração desta técnica de filtragem colaborativa justifica-se em face do intenso processo de globalização que promove de forma constante transformações significativas no setor. Observa-se a expansão do acesso à Internet que contribui para o crescimento acentuado no número de usuários e itens em diversos domínios, como por exemplo o de aplicativos para dispositivos móveis, como já mencionado. Esta expansão acentua o problema do custo computacional do kNN. Além disso, o trabalho desenvolvido é relevante para as pesquisas conduzidas no Laboratório de Sistemas Embarcados e Computação Pervasiva (Embedded) da Universidade Federal de Campina Grande (UFCG).

1.4 Estrutura da Dissertação

Os demais capítulos deste documento estão estruturados da seguinte forma:

- No *Capítulo 2* é apresentado um embasamento teórico sobre sistemas de recomendação, destacando seus tipos e suas respectivas limitações. Em seguida, fundamenta-se sobre o método kNN e sua aplicação na área de recomendação.

- No *Capítulo 3* tem-se a revisão da literatura, em que são discutidos trabalhos desenvolvidos com o intuito de melhorar o desempenho do kNN.

- No *Capítulo 4* apresenta-se uma motivação para o problema a ser abordado pela solução proposta, em seguida descreve-se um método que visa melhorar o desempenho do kNN.

- O *Capítulo 5* tem como foco o experimento realizado. Têm-se a descrição do experimento, ferramentas utilizadas ao longo do trabalho, análise estatística e validação.

- No *Capítulo 6* são apresentadas as conclusões do trabalho, suas limitações e trabalhos futuros.

Capítulo 2

Fundamentação Teórica

Neste capítulo são apresentados os principais conceitos relativos à área de sistemas de recomendação, necessários ao embasamento do trabalho. São detalhados os sistemas de recomendação e suas principais abordagens, destacando-se as vantagens e limitações de cada uma. A fundamentação apresenta maior aprofundamento na filtragem colaborativa e no método *k-Nearest Neighbors* que estão diretamente ligados ao foco do trabalho.

2.1 Sistemas de Recomendação

Os avanços tecnológicos conquistados, principalmente, nas últimas décadas, tornaram o acesso à informação mais rápido e fácil. Diariamente uma grande massa de usuários utilizam motores de busca para tentar encontrar aquilo que desejam, porém a tarefa de entregar a informação exata, embora pareça trivial, é mais complexa do que se imagina. Ao pesquisar em sites de busca, milhares de resultados podem ser retornados, entretanto, apenas uma parcela mínima do que foi encontrado realmente interessa ao usuário. A grande maioria da informação acaba sendo descartada. Tal situação caracteriza um problema conhecido como *Information Overload* (IO), ou seja, representa a incapacidade do usuário de absorver e processar o grande volume de informação ao qual está exposto.

Os sistemas de recomendação surgem como uma alternativa para tentar amenizar parte do problema da IF [10]. Esses sistemas têm como principal objetivo sugerir conteúdo personalizado ao usuário. Os SRs vêm sendo usados em diversos domínios, tais como livros [50], televisão [31], filmes [4], músicas [63] e *e-commerce* [56].

A área de recomendação ganhou ainda mais notoriedade com o surgimento de uma competição oferecida pela *NetFlix*. O concurso conhecido como *NetFlix Prize* [7] foi lançado em 2006 e oferecia um prêmio de um milhão de dólares para quem conseguisse melhorar a acurácia do sistema de recomendação da empresa em 10%. A empresa liberou em outubro de 2006, uma base de dados com aproximadamente 100 milhões de avaliações numa escala inteira de 1 a 5 estrelas, 480 mil usuários, escolhidos de forma aleatória e anônima, e 18 mil títulos de filmes. Os sistemas concorrentes precisavam prever 3 milhões de avaliações. Somente em setembro de 2009 foi anunciada a solução vencedora [37], desenvolvida pela equipe *BellKor's Pragmatic Chaos*. Embora a equipe vencedora tenha conseguido alcançar os 10% de melhoria necessária, sua solução nunca foi implementada pela *Netflix*, pois contava com a combinação de mais de 100 técnicas de recomendação e o ganho adquirido não justificava o esforço para por o sistema em funcionamento.

Os SRs, geralmente, dividem-se em: filtragem baseada em conteúdo, filtragem colaborativa e filtragem híbrida.

2.1.1 Filtragem Baseada em Conteúdo

Sistemas de recomendação baseada em conteúdo constroem suas recomendações a partir da comparação do perfil do usuário com os conteúdos dos itens. Basicamente, o processo de construção das recomendações consiste em montar um perfil - estrutura que representa os interesses do usuário - a partir da extração das características dos itens aos quais o usuário demonstrou interesse no passado. Em seguida, as recomendações são geradas comparando os atributos presentes no perfil do usuário com os atributos extraídos dos conteúdos dos outros itens. Os itens mais compatíveis são recomendados. Segundo Lops et al. [40], as principais vantagens das técnicas baseadas em conteúdo estão relacionadas à independência de usuários, pois o processo de recomendação depende apenas do histórico do usuário alvo e do conteúdo dos itens. Portanto, este tipo de filtragem não somente independe de grande volume de avaliações de outros usuários bem como possibilita a recomendação de itens que ainda não foram avaliados [5].

Limitações

A FBC possui as seguintes limitações:

- **Análise de Conteúdo:** em alguns casos o conteúdo dos itens pode ser escasso, incorreto ou até mesmo difícil de ser extraído. Por exemplo, no domínio de TV Digital uma das principais fontes de informação é o *Eletronic Program Guide* (EPG), porém uma vez que ele é construído manualmente, é possível que um programa seja classificado com categorias incorretas ou sua descrição não reflita totalmente seu conteúdo. Em outros casos, como no domínio de músicas e vídeos, a extração do conteúdo desses itens torna-se uma tarefa mais complexa, pois demanda maior esforço computacional;
- **Super-especialização:** as recomendações para o usuário acabam restritas aos itens similares aos que ele interagiu no passado, uma vez que a construção de seu perfil é baseada nesses mesmos itens;
- **Novos Usuários:** para gerar recomendações eficazes é preciso haver informações suficientes sobre os interesses do usuário, para que assim o sistema possa entender e montar um perfil adequado.

2.1.2 Filtragem Colaborativa

A filtragem colaborativa é uma das abordagens mais recorrentes na área de recomendação [13]. Sistemas colaborativos recomendam itens com base nas avaliações de usuários similares ao usuário alvo. Na FC assume-se que usuários que possuem padrões similares de avaliação tendem a atribuir notas a novos itens de forma semelhante. Umas das principais vantagens das técnicas colaborativas refere-se ao fato de serem completamente independentes da representação do objeto, conseguindo trabalhar bem, mesmo com itens nos quais a extração do conteúdo é complexa, tais como em filmes e músicas [13]. Tal vantagem ocorre pelo fato da FC utilizar apenas as avaliações do usuário. Outro ponto forte diz respeito à diversidade das recomendações, pois a FC consegue sugerir itens diferentes daqueles que o usuário se interessou no passado.

Em geral, algoritmos colaborativos podem ser agrupados em: baseados em memória e baseados em modelos. Os baseados em memória necessitam que o conjunto de dados seja

armazenado na memória principal para gerar as recomendações, são mais fáceis de implementar e conseguem se adaptar melhor às mudanças de interesse do usuário. Em contrapartida, os baseados em modelos geram recomendações utilizando um modelo construído previamente a partir dos dados. Eles conseguem fornecer recomendações mais acuradas, porém a construção do modelo é uma etapa custosa.

Representação dos Dados

Na FC o conjunto de dados é formado tipicamente por um subconjunto de usuários $U = \{u_1, u_2, u_3, ..., u_n\}$ e um subconjunto de itens $I = \{i_1, i_2, i_3, ..., i_m\}$, onde cada usuário $u \in U$ possui uma lista de itens, I_u. Esses dados normalmente são estruturados em forma de uma matriz M (Matriz Usuário-Item), onde cada elemento de M representa uma avaliação de um usuário u para um item i.

$$
\mathrm{M} = \begin{array}{c} \\ u_1 \\ u_2 \\ u_3 \\ \\ u_n \end{array}
\begin{array}{c}
\begin{array}{ccccc} i_1 & i_2 & i_3 & & i_m \end{array} \\
\begin{bmatrix}
a_{11} & a_{12} & a_{13} & \cdots & a_{1m} \\
a_{21} & a_{22} & a_{23} & \cdots & a_{2m} \\
a_{31} & a_{32} & a_{33} & \cdots & a_{3m} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
a_{n1} & a_{n2} & a_{n3} & \cdots & a_{nm}
\end{bmatrix}
\end{array}
$$

Tipos de Avaliações

As avaliações são imprescindíveis para as técnicas colaborativas, pois sem estas não é possível gerar recomendações. As avaliações podem ser:

- **Explícitas:** quando o usuário expressa claramente seu interesse. As formas mais comuns de avaliações explícitas são baseadas em escalas. Por exemplo, a *Netflix* e o *Google Play* permitem a avaliação dos itens utilizando uma escala de 1 a 5 estrelas, na qual quanto maior for o número de estrelas, maior é o interesse do usuário. Em contrapartida, o *Youtube* coleta o interesse do usuário a partir de uma escala binária com as opções "Gostei"e "Não Gostei"(Figura 2.1)[1]. Embora o *feedback* explícito consiga fornecer ótimos resultados aos sistemas colaborativos, em determinados domínios a

[1]Imagem retirada da Internet

coleta desse tipo de dado torna-se uma tarefa difícil de ser realizada, pois os usuários não têm tendência a avaliar uma grande quantidade de itens. Por exemplo, no domínio de TV Digital a interação do usuário com a TV é limitada, pois é feita, geralmente, com uso do controle remoto, o qual possui um conjunto de ações restritas, dificultando qualquer iniciativa de avaliação do usuário;

- **Implícitas:** quando a avaliação é inferida mediante o comportamento do usuário, por exemplo, a partir do tempo de interação com o item ou baseando-se no número de vezes em que o item foi acessado, etc. As avaliações implícitas costumam ser menos intrusivas, pois não exigem uma ação direta. O *feedback* implícito é construído na maioria das vezes sem o conhecimento do usuário, ou seja, sem que ele perceba que seu comportamento está sendo monitorado. Embora esse tipo de avaliação seja mais fácil de ser coletada, normalmente é menos significativa do que a avaliação explícita, pois essa última reflete diretamente o interesse do usuário.

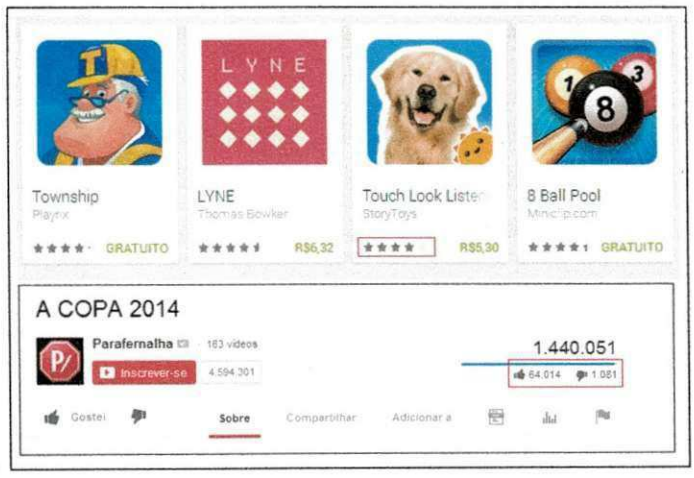

Figura 2.1: Tipos de Avaliação Explícita

Cenários e Métricas de Avaliação

A FC apresenta tipicamente dois cenários: predição de notas e predição de itens. A predição de notas consiste na tarefa de predizer a avaliação do usuário. Nesse cenário, assume-se que os itens com notas mais altas são mais interessantes. As avaliações geralmente são representadas em diferentes escalas, sendo uma das mais comuns a de 1 a 5 estrelas. A

definição da qualidade das predições, nesse cenário, é obtida normalmente com métricas baseadas em erros. Tais medidas são obtidas a partir da diferença entre o valor da predição e o valor real da avaliação atribuído pelo usuário. As mais comuns são:

- **Mean Absolute Error** (MAE): mede a média do desvio absoluto dos erros. Apresenta menos sensibilidade a *outliers*[2]. $r'(u,i)$ é a predição para o usuário u sobre o item i, I é o conjunto de itens para os quais serão realizadas predições e $r(u,i)$ corresponde a avaliação real do usuário.

$$MAE = \frac{1}{|I|}\sum_{i \in I} |r'(u,i) - r(u,i)| \tag{2.1}$$

- **Mean Square Error** (MSE): mede a média do quadrado dos erros. MSE penaliza demasiadamente os *outliers*.

$$MSE = \frac{1}{|I|}\sum_{i \in I} (r'(u,i) - r(u,i))^2 \tag{2.2}$$

- **Root Mean Square Error** (RMSE): mede a raiz quadrada da média do quadrado dos erros. É mais usada que a MSE, pois é medida na mesma unidade dos dados, porém sofre com o mesmo problema dos *outliers*.

$$RMSE = \sqrt{\frac{1}{|I|}\sum_{i \in I} (r'(u,i) - r(u,i))^2} \tag{2.3}$$

Por sua vez, a predição de itens tem como objetivo sugerir uma lista com as *top-n* recomendações. Essa lista contém itens ordenados de acordo com a probabilidade de interesse do item para o usuário. Nesse tipo de cenário, tipicamente, utilizam-se métricas que são úteis quando não se tem interesse na exatidão das predições, mas sim em saber se o usuário alvo gostará ou não do item [17]. Essas medidas se mostram mais adequadas para domínios com avaliações binárias, em que o valor 1 (um) significa que o usuário gostou de um item e 0 (zero) que ele não gostou ou não teve acesso ao mesmo. Nesse panorama, as métricas mais populares são:

[2]São observações que apresentam grande afastamento das demais do conjunto. Na área de recomendação são avaliações com grande erro de predição.

- **Precision**: corresponde à fração de itens recomendados que são realmente relevantes para o usuário, em outras palavras, mede a probabilidade de um item recomendado estar presente no conjunto de itens do usuário. Onde I_T denota os itens que estão de acordo com os interesses do usuário e I_R são os itens retornados na recomendação;

$$PRECISION = \frac{|I_T \cap I_R|}{|I_R|} \tag{2.4}$$

- **Recall**: corresponde à fração de itens relevantes que estão presentes na recomendação, ou seja, mede a probabilidade de um item presente no conjunto de interesse, ser recomendado;

$$RECALL = \frac{|I_T \cap I_R|}{|I_T|} \tag{2.5}$$

- **F-measure**: combina *precision* e *recall* em uma única métrica.

$$F - MEASURE = 2 \times \frac{precision \times recall}{precision + recall} \tag{2.6}$$

Limitações

- **Esparsidade**: corresponde à escassez de avaliações na base de dados. A FC normalmente lida com um grande volume de usuários e itens, que podem facilmente atingir a casa dos milhões, porém a quantidade de avaliações de grande parte dos usuários tende a ser pequena, chegando a alcançar níveis de esparsidade de mais de 90%. Por exemplo, uma das bases de dados utilizadas neste trabalho conta com 943 usuários e 1.682 itens. De um total de 1.586.126 avaliações possíveis a base possui apenas 100.000, apresentando assim um nível de esparsidade de quase 94%. Esse problema dificulta a busca por similaridade entre usuários e acarreta em um declínio da acurácia do sistema;

- **Escalabilidade**: com o crescimento do volume de dados, os sistemas tendem a fornecer recomendações mais acuradas, porém eleva-se significativamente o consumo dos recursos necessários para o processo. Os algoritmos baseados em memória, por exemplo, passam a utilizar quantidades exorbitantes de bytes. A busca por similaridade entre usuários torna-se uma tarefa árdua, aumentando assim o custo computacional para a geração das recomendações;

- **Cold-start**: refere-se à situação em que um novo usuário e/ou item acabou de entrar no sistema. Para fornecer recomendações relevantes ao usuário é preciso que ele tenha avaliado uma quantidade de itens suficiente. De maneira análoga ocorre com o item, sendo preciso que um determinado número de usuários o tenham avaliado.

- **Ovelha Negra**: consiste na dificuldade em encontrar usuários similares para aqueles que possuem um conjunto de avaliação muito peculiar.

2.1.3 Filtragem Híbrida

Os sistemas híbridos combinam duas ou mais técnicas de recomendação com o intuito de minimizar as limitações que elas possuem individualmente [5, 13, 15]. De acordo com Burke e Robin [13], os sistemas híbridos podem ser divididos em sete classes:

(1) **Weighed**: cada abordagem realiza sua própria predição, as quais são posteriormente combinadas em uma única;

(2) **Switching**: uma determinada abordagem é selecionada para fazer a predição quando um dado critério é atingido;

(3) **Mixed**: as predições de ambas as abordagens são apresentadas ao usuário;

(4) **Feature Combination**: um único algoritmo de recomendação recebe como entrada recursos (resultados de uma mineração textual, avaliações implícitas inferidas, etc) de outras técnicas;

(5) **Cascade**: a saída de uma técnica de recomendação é refinada por outra abordagem;

(6) **Feature Augmentation**:a saída de uma técnica de recomendação é a entrada de outra;

(7) **Meta-level**: todo o modelo produzido por uma abordagem é utilizado por outra.

2.2 k-Nearest Neighbors

O kNN é um dos métodos mais populares para buscas associativas. Por ser de fácil entendimento, simples implementação e apresentar excelentes resultados, essa técnica é amplamente

utilizada para os mais diversos problemas de classificação [36], tais como, classificação de textos, imagens, formulação de diagnósticos médicos, etc. Os objetos de interesse são representados por um vetor de característica $v = [a_1(x), a_2(x), a_3(x), ..., a_d(x)]$, onde $a_i(x)$ representa o valor do ith atributo A_i do objeto x. Dada uma instância x, o kNN a rotula de acordo com a classe mais comum entre os vizinhos mais próximos de x. O resultado da classificação é influenciado, principalmente, por dois critérios: métrica de similaridade e o número de vizinhos mais próximos.

A similaridade entre objetos é utilizada para definir o quão próximos eles são e consequentemente selecionar os vizinhos mais próximos do objeto a ser rotulado. No âmbito de classificação as métricas mais comuns são:

- **Distância Euclidiana**: corresponde à distância de uma linha reta entre dois pontos em um espaço euclidiano;

$$D_E(x, y) = \sqrt{\sum_{i=1}^{d}(a_i(x) - a_i(y))^2} \tag{2.7}$$

- **Distância de Manhattan**: corresponde à distância entre dois pontos medida ao longo dos eixos em ângulos retos;

$$D_{M_a}(x, y) = \sum_{i=1}^{d} |a_i(x) - a_i(y)| \tag{2.8}$$

- **Distância de Minkowski**: é uma generalização das duas anteriores. Quando $p = 1$ torna-se uma distância de Manhattan e para $p = 2$ tem-se a distância Euclidiana.

$$D_{M_i}(x, y) = \sum_{i=1}^{d}(|a_i(x) - a_i(y)|^p)^{\frac{1}{p}} \tag{2.9}$$

A definição do número de vizinhos mais próximos (k) não é trivial, pois a quantidade ideal de vizinhos pode variar bastante de acordo com o domínio do problema e a base de dados utilizada. O número k tem impacto direto no desempenho do kNN, pois é com base nas informações dos vizinhos mais próximos que a classificação é realizada. É comum que a decisão seja feita a partir de análise empírica, entretanto, há diversos trabalhos voltados para a escolha automática de k [33, 47].

2.2.1 kNN para Predição de Avaliações

Existem algumas diferenças em relação à aplicação do kNN para classificação de objetos e para predição de notas. No âmbito de classificação geralmente os vetores de características são densos, ou seja, todos (ou quase todos) os atributos do objeto possuem valores, o que faz com que o vetor seja analisado como um todo e em seguida o objeto seja designado a uma única classe. Em contrapartida, no cenário de predição têm-se vetores de características bastante esparsos, em que cada posição nula do vetor corresponde a um item não avaliado, o qual precisa ter seu valor predito separadamente. Outras diferenças relacionam-se às métricas de similaridade utilizadas. Para classificação normalmente utilizam-se medidas de distância, tais como distância Euclidiana e de Manhattan, enquanto que para predição as métricas mais utilizadas são:

- **Correlação de Pearson**: mede a força e a direção da relação linear entre dois vetores;

$$Corr_P(a,u) = \frac{\sum\limits_{i \in I_{au}} (r(a,i) - \overline{r(a)})(r(u,i) - \overline{r(u)})}{\sqrt{\sum\limits_{i \in I_{au}} (r(a,i) - \overline{r(a)})^2} \sqrt{\sum\limits_{i \in I_{au}} (r(u,i) - \overline{r(u)})^2}} \tag{2.10}$$

- **Similaridade do Cosseno**: mede o ângulo entre dois vetores.

$$Sim_C(a,u) = \frac{\sum\limits_{i \in I_{au}} (r(a,i) - r(u,i))}{\sqrt{\sum\limits_{i \in I_{au}} r(a,i)^2} \sqrt{\sum\limits_{i \in I_{au}} r(u,i)^2}} \tag{2.11}$$

Onde $r(a,i)$ é a avaliação do usuário a sobre o item i e $\overline{r(a)}$ é a média das avaliações de a. I_{au} é conjunto de itens avaliados simultaneamente pelos usuários a e u.

O cálculo da similaridade entre dois usuários é uma operação bastante recorrente. Para predizer a avaliação de um usuário para um item é necessário, dentre outras coisas, encontrar os k vizinhos mais próximos do usuário alvo e, para tanto, é preciso computar sua similaridade com cada um dos demais usuários da base. Portanto, tal processo acaba apresentando um elevado custo computacional. A fim de evitar a repetição desnecessária dos cálculos de similaridade, tem-se uma Matriz Triangular chamada Matriz Similaridade (M_S), onde cada elemento dessa matriz representa a similaridade s_{ij} entre um usuário u_i e um usuário u_j.

$$M_S = \begin{array}{c} \\ u_1 \\ u_2 \\ u_3 \\ \\ u_n \end{array} \begin{array}{c} \begin{array}{ccccc} u_1 & u_2 & u_3 & .. & u_n \end{array} \\ \begin{bmatrix} s_{11} & & & & \\ s_{21} & s_{22} & & & \\ s_{31} & s_{32} & s_{33} & & \\ \vdots & \vdots & \vdots & \ddots & \\ s_{n1} & s_{n2} & s_{n3} & \cdots & s_{nn} \end{bmatrix} \end{array}$$

A geração de uma predição consiste basicamente em selecionar os k vizinhos mais próximos do usuário alvo. Em seguida, a partir das notas dos vizinhos sobre o item alvo, realiza-se um cálculo ponderado para encontrar o valor da predição. Esse cálculo pode ser observado na Equação(2.12).

$$r'(a,i) = \overline{r(a)} + \frac{\sum\limits_{u \in U_{au}} sim(a,u)(r(u,i) - \overline{r(u)})}{\sum\limits_{u \in U_{au}} sim(a,u)} \tag{2.12}$$

Onde $r'(a,i)$ é a predição para o usuário a sobre o item i, $sim(a,u)$ é a similaridade entre a e o seu vizinho u, $r(u,i)$ corresponde a avaliação real do usuário, $\overline{r(u)}$ representa a média de avaliações de u. E o conjunto U_{au} representa os k vizinhos mais próximos de a.

Capítulo 3

Revisão da Literatura

Neste capítulo são apresentadas algumas das principais abordagens relacionadas à melhoria do método dos vizinhos mais próximos. De acordo com a revisão da literatura realizada, os trabalhos relacionados foram divididos com base nos tipos de soluções empregadas, as quais são descritas a seguir.

3.1 Redução da Base de Dados

Uma das maneiras de melhorar o desempenho do kNN é reduzindo o custo relacionado à computação da similaridade entre os objetos. Isso pode ser alcançado diminuindo o volume da base de dados, fazendo com que um número menor dos cálculos seja necessário. Essa redução pode ser feita quando se eliminam instâncias e/ou quando se eliminam dimensões nos vetores de características dos objetos. O grau de redução é escolhido de acordo com o objetivo almejado. Quando o intuito é aumentar a acurácia, procura-se excluir apenas os ruídos, ou seja, aquelas instâncias que contribuem negativamente para o resultado final. Quando o foco é aumentar a velocidade, o grau de redução é significantemente maior, pois nesse caso o problema passa a ser visto ligeiramente diferente. A ideia é selecionar a menor quantidade possível de dados para aliviar a computação, porém quanto menos dados menor é o nível de acurácia das recomendações.

A solução proposta por Boumaza e Brun [11] baseia-se na redução do espaço de busca dos vizinhos mais próximos, utilizando um algoritmo de busca estocástica chamado *Iterated Local Search* (ILS) [41]. O ILS retorna um subconjunto de usuários, chamados Vizinhos

Globais (VGs), de onde são retirados os vizinhos de cada usuário alvo. A ideia é acelerar a formação das vizinhanças buscando os vizinhos em um subconjunto restrito, ao invés de buscar em todo o conjunto de dados. O conjunto dos VGs é formado a partir da função de *fitness* representada pela Equação 3.1, onde $e(s)$ consiste em uma combinação (não detalhada pelos autores) de cobertura[1] e erro (MAE) fornecidos pelo conjunto de usuários s.

$$f(s) = \alpha \times e(s) + (1 - \alpha) \times |s| \tag{3.1}$$

Como resultado, o método proposto consegue reduzir o espaço de busca pelos vizinhos mais próximos a 16% do conjunto original, mantendo a acurácia das recomendações bem próxima daquelas conseguidas utilizando todo o conjunto de dados. A principal limitação desse trabalho refere-se ao tempo necessário para encontrar os Vizinhos Globais. A cada iteração do algoritmo ILS, dentre outras coisas, é preciso calcular o MAE fornecido pelo conjunto. Além disso, o *Iterated Local Search* é um método não-determinístico, portanto não há como prever o número de iterações necessárias para encontrar o conjunto final. Dessa forma, o processo de formação dos VG acaba sendo bastante custoso.

Suguna e Thanushkodi [58] propuseram uma melhoria para o kNN fundamentada em dois pontos: diminuição das dimensões dos vetores de características dos objetos e redução nos cálculos da similaridade. Vetores com grandes dimensões causam maior complexidade computacional devido à similaridade que precisa ser computada entre todos os objetos da base. Segundo os autores, os algoritmos de classificação tendem a ficar "confusos" com um maior número de características, pois algumas delas podem não agregar qualidade ao resultado final, ocasionando apenas ruído na solução. A seleção das características é feita por um método, também proposto pelo autores, baseado no algoritmo *Bee Colony Optimization* [59]. Na etapa de classificação utilizam-se Algoritmos Genéticos (AGs) para melhorar a busca pelos vizinhos. Ao invés de verificar todas as similaridades entre os objetos, com o uso de AG, apenas k vizinhos são selecionados a cada iteração. As similaridades (*fitness*) são computadas e o cromossomo (representação do objeto) com maior *fitness* é armazenado como um máximo global. Esse processo é repetido L vezes e ao final o rótulo (classe) do cromossomo mais apto é usado para rotular o objeto de interesse.

Abidin e Perrizo [2] desenvolveram um algoritmo, chamado SMART-TV (SMall Abso-

[1] É dada pela razão entre o número de total de itens e o número de itens para os é possível realizar predição

lute diffeRence of ToTal Variation). O algoritmo apresenta duas fases: pré-processamento e classificação. Na primeira, calcula-se a proximidade entre os objetos a serem classificados e as demais instâncias, utilizando uma função chamada *Vertical Square Set Distance* (VSSD) [1]. VSSD é uma técnica eficiente e escalável usada para medir a variação total de um ponto em relação a um conjunto de pontos na base. Na segunda fase os vizinhos são selecionados de acordo com os valores computados na fase de pré-processamento. A lista de vizinhos é ordenada utilizando a distância Euclidiana e em seguida escolhem-se os k mais próximos para classificação. Basicamente a ideia do método consiste em realizar um cálculo prévio de proximidade utilizando uma métrica com baixo custo computacional, em termos de tempo, para em seguida calcular a real distância, porém apenas entre um número significativamente menor de instâncias, conseguindo assim tornar a etapa de classificação mais rápida.

3.2 Particionamento Binário do Espaço (PBE)

É um método utilizado para subdividir recursivamente o espaço em conjuntos convexos definidos em hiperplanos [32, 49]. Dado um conjunto S de objetos em um espaço de dimensão d. O espaço é dividido em dois subespaços com um hiperplano. Por sua vez, ambos os subespaços são recursivamente particionados. Esse processo continua até que apenas um, ou pelo menos um pequeno número de objetos, restem em cada subespaço. Na Figura 3.1(a) é possível observar um exemplo de PBE de dimensão 2. Na Figura 3.1(b) tem-se uma Árvore PBE que á a representação mais comum desse método.

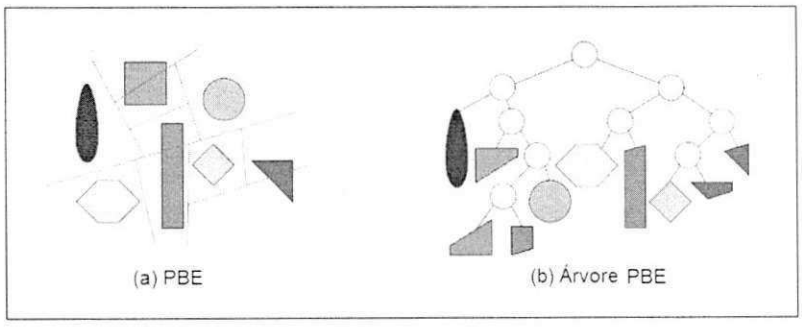

Figura 3.1: Exemplo de Particionamento Binário do Espaço

Uma das aplicações do PBE tem como objetivo tornar mais rápido o processo de busca

pelos vizinhos mais próximos. Os vetores de características do objetos, representados por pontos em um espaço k-dimensional, são particionados de acordo com uma função de distância, normalmente, a Euclidiana. Tem-se como exemplo desse tipo de aplicação uma estrutura bastante recorrente na literatura, utilizadas para busca em espaços multidimensionais, conhecida por *K-Dimensional Tree* (kd-tree).

K-Dimensional Tree

Proposta originalmente por Bentley [8], *kd tree* é uma estrutura de dados para armazenamento de informação a ser recuperada através de buscas associativas, isto é, consiste em uma generalização da árvore binária de busca onde cada nó representa um hiper-retângulo e um hiperplano ortogonal a um dos eixos cartesianos, o qual divide o hiper-retângulo em duas partes. Cada uma delas são associadas a dois nós filhos. A partir da raiz da árvore, todo o espaço de busca é particionado até que o número de pontos no hiper-retângulo atinja um limiar. Os nós folhas são chamados de *buckets* e o limiar é restringido pelo número máximo (*bucket size*) de pontos suportados por um *bucket*. É importante destacar que os pontos são armazenados apenas nas folhas e não nos nós internos [21].

A estrutura da *kd tree* provê um mecanismo de busca eficiente para examinar apenas os pontos mais próximos do ponto de interesse, podendo reduzir o tempo de busca pelos vizinhos mais próximos de $O(N)$ a $O(log\ N)$. No caso de espaços unidimensionais a busca torna-se idêntica a de uma árvore binária, onde cada nó contém um valor de partição em que os pontos são divididos. Todos os pontos com valor menor que o valor de partição pertencem ao filho da esquerda e os demais ao da direita. Em um espaço de dimensão k, apenas uma dimensão (chave) é escolhida para servir como discriminador. O espaço é particionado de acordo com o valor de partição que está diretamente ligado ao discriminador escolhido [21].

A definição do discriminador e do valor de partição representa uma tarefa importante, pois esses dois parâmetros influenciam significativamente o desempenho das buscas. Na proposição original da *kd tree*, Bentley [8] escolheu o discriminador D com base na altura L do nó na árvore, onde $D = (L\ mod\ k) - 1$. Os valores de partição correspondiam aos valores das chaves que eram escolhidas aleatoriamente em cada ponto.

Friedman et al. [22] propôs uma otimização para a *kd tree*, cujo objetivo é minimizar o número de pontos examinados durante a busca. Para isso, o discriminador foi escolhido com

base no desvio padrão máximo das chaves e o valor de partição e dado pela mediana das dimensões (definidas pelo discriminador).

Já Gother et al. [26] desenvolveu um método que corresponde a uma leve variação do trabalho realizado por Friedman et al. [22]. Na construção da *kd tree*, cada nó é dividido em dois usando a mediana (valor de partição) da dimensão com maior variância (discriminador) entre os pontos na subárvore e assim por diante. Como resultado, obteve-se 25% mais velocidade na etapa de classificação.

3.3 Approximate Nearest Neighbors (ANN)

Tradicionalmente a busca pelos vizinhos mais próximos é realizada verificando a similaridade do objeto de interesse com cada uma das demais instâncias da base de dados. Em seguida, constrói-se uma lista em ordem decrescente de similaridade e se escolhem os k primeiros objetos da lista, que formarão os vizinhos mais próximos. Esse tipo de busca é chamada de busca exata, pois retorna fielmente os objetos com maior proximidade. Com o intuito de contornar o problema da complexidade de tempo para formação das vizinhanças surgiram os métodos de busca aproximada que encontram instâncias próximas (vizinhos aproximados) do objeto de interesse sem a necessidade de verificar a similaridade entre todos os objetos. Um dos principais métodos que se baseiam nesse conceito é o *Locality Sensitive Hash* (LSH).

Locality Sensitive Hash

Umas da principais aplicações do LSH é fornecer um método eficiente de busca aproximada em espaços vetoriais de alta dimensão. Segundo Haghani et al. [28], a ideia principal é mapear, com alta probabilidade, objetos similares(representados em um espaço vetorial dimensional d) em um mesmo *hash bucket*, isto é, objetos próximos têm mais chances de terem o mesmo valor de *hash* do que os que estão mais distantes. A indexação é realizada a partir de funções de *hashing* e da construção de diversas tabelas *hash* para aumentar a probabilidade de colisão entre os pontos próximos. A busca pelos vizinhos se dá a partir do *hashing* do objeto de interesse para um *bucket* por tabela *hash*, em seguida os objetos encontrados (vizinhos) são ordenados de acordo com a distância do objeto de interesse.

O trabalho desenvolvido por Gionis et al. [24] utiliza *Locality Sensitive Hash* para melhorar a busca pelos vizinhos de objetos representados por pontos de dimensão d em um espaço de *Hamming* $\{0,1\}^d$. Nos experimentos realizados observou-se que o método proposto conseguiu superar em termos de velocidade estruturas baseadas em Árvores PBE, quando os dados são armazenados em disco.

O algoritmo de LSH vem sendo usado em diversas outras aplicações [12, 16, 46], entretanto, nesses casos há uma limitação fundamental a ser considerada: eles são rápidos apenas quando os pontos estão em um espaço de *Hamming*. Datar et al. [18] propôs uma nova versão do algoritmo LSH que diferentemente dos trabalhos anteriores [12, 16, 24, 46], lida diretamente com pontos no espaço Euclidiano. Para avaliar a solução proposta realizou-se experimentos com bases de dados sintéticas, com vetores esparsos de alta dimensão ($20 \leq d \leq 500$). Como resultado obteve-se um desempenho de até 40 vezes mais rápido que o kd tree (em sua proposição original).

3.4 Considerações Finais do Capítulo

Neste capítulo foram apresentadas algumas das principais abordagens para tentar diminuir o problema do custo computacional do kNN em grande bases de dados. Observou-se que a forma mais comum de se alcançar tal objetivo é modificando o processo de busca dos vizinhos mais próximos.

Os trabalhos pesquisados são voltados, tipicamente, para dois tipos de cenários: classificação de objetos e predição de avaliações. No cenário de classificação, os métodos propostos lidam com vetores densos e de baixa dimensão, enquanto que no de predição de avaliações as abordagens lidam com vetores esparsos de alta dimensão.

Neste trabalho propõe-se um método colaborativo para predição de notas baseado no algoritmo *K-Nearest Neighbors*. A solução proposta visa melhorar o desempenho do kNN a partir da redução do espaço de busca dos vizinhos, com auxílio de uma heurística de seleção de usuários baseada no número de itens avaliados.

Capítulo 4

Solução Proposta

Neste capítulo descreve-se uma abordagem de recomendação centrada na filtragem colaborativa, baseada no método *k-Nearest Neighbors*.

4.1 Visão Geral

A solução proposta foi baseada na abordagem apresentada por Boumaza e Brun [11]. Apesar de conseguir resultados acurados com um conjunto reduzido de dados, o método proposto por Boumaza e Brun apresenta uma limitação em relação ao tempo de execução. A seleção dos Vizinhos Globais é feita com o auxílio de um algoritmo de busca estocástica chamado *Iterated Local Search*. ILS é um método bastante eficiente para problemas de otimização, porém demanda um tempo considerável para alcançar seus resultados. A cada iteração do algoritmo ILS, dentre outras coisas, é preciso calcular o MAE fornecido pelos VGs e esse cálculo por si só já consome bastante tempo. Além disso, o *Iterated Local Search* é um método não-determinístico, por isso não há como prever o número de iterações necessárias para se chegar ao resultado. Desse modo, o processo de formação do conjunto dos VGs acaba sendo bastante custoso.

A abordagem proposta neste trabalho consiste em restringir o espaço de busca dos vizinhos mais próximos. Para isso, é necessário escolher um subconjunto dos usuários com o menor tamanho possível, sem perder a representatividade.

Diante dessa limitação enxergou-se uma oportunidade de aperfeiçoamento do método proposto em [11]. Tomou-se como base a ideia de restrição do espaço de busca dos vizinhos

mais próximos, isto é, ao invés de selecionar os vizinhos percorrendo toda a base de dados, opta-se por procurá-los em um subconjunto reduzido, porém representativo. A representatividade refere-se ao nível de acurácia que pode ser alcançado a partir das avaliações dos usuários escolhidos.

4.2 Heurística de Seleção

O ponto chave da solução proposta reside na escolha de uma heurística para a formação de um espaço reduzido de busca dos vizinhos mais próximos. Diminuindo-se o espaço de busca, a escolha dos vizinhos torna-se menos custosa, pois evita-se a necessidade de busca em toda a população. Por outro lado, perde-se acurácia na recomendação final. A fim de minimizar tal perda faz-se necessária a definição de um conjunto representativo de usuários, os chamados Vizinhos Globais [11]. Diante disso foi proposta uma heurística simples, rápida e eficiente que baseia-se na quantidade de itens avaliados pelos usuários. De acordo com essa heurística o conjunto dos VGs é composto pelos usuários mais ativos, ou seja, aqueles que naturalmente têm maior tendência a avaliar mais itens. Na prática os usuários mais ativos corresponde àqueles que possuem mais itens avaliados em seu histórico. Diversas outras heurísticas para seleção dos usuários também foram avaliadas ao longo da pesquisa. Dentre elas:

- **Média de Similaridade:** os usuários são selecionados de acordo com suas médias de similaridade, isto é, para cada usuário calcula-se sua similaridade com cada um dos demais e em seguida obtém-se a média. Aqueles com as maiores médias são escolhidos;

- **Número de Conexões:** selecionam-se os usuários de acordo com a capacidade de serem vizinhos, ou seja, cada vez que um usuário apresentar um valor positivo de similaridade ele recebe um ponto. Ao final, aqueles com as maiores pontuações são selecionados;

- **Vizinhos mais Recorrentes:** os usuários recebem pontuações de acordo com o número de vezes em que surgem entre os k vizinhos mais próximos de um usuário alvo. Por exemplo, verificam-se os k vizinhos mais próximos e atribui-se um ponto para

cada vizinho da lista. Esse processo é repetido com todos os usuários da base e ao final tem-se a lista dos vizinhos mais recorrentes;

- **Itens Distintos:** cria-se uma lista ordenada de forma decrescente (*Lista A*) de acordo com o número de itens que cada usuário avaliou. Inicia-se uma varredura na *Lista A*, o primeiro usuário é adicionado em uma segunda lista (*Lista B*) e os itens que esse usuário avaliou são adicionados em uma terceira lista (*Lista C*). Ao continuar a varredura verifica-se se o conjunto de itens avaliados pelo próximo usuário difere daqueles presentes na *Lista C*, caso isso ocorra, os itens distintos são adicionados na *Lista C* e o usuário na *Lista B*. O procedimento continua até que o tamanho da *Lista B* atinja a porcentagem de usuários definida ou que a *Lista A* tenha sido varrida por completo. Nesse último caso, se o tamanho da *Lista B* for inferior à porcentagem estabelecida, é feita uma nova varredura na *Lista A* e vão adicionando-se os usuários que ainda não estão presentes na *Lista B* até que esta alcance o tamanho desejado.

Inicialmente foram analisados os custos assintóticos de cada heurística. É possível observar na Tabela 4.1 que a heurística Usuários mais Ativos e a Itens Distintos possuem custos. As demais apresentam custo significativamente, pois para cada usuário faz-se necessários o cálculo de similaridade com todos os outros.

Tabela 4.1: Análise Assintótica das Heurísticas

Heurística	Tempo (segundos)
Usuários mais Ativos	$O(n)$
Itens Distintos	$O(n)$
Média de Similaridade	$O(n^2)$
Número de Conexões	$O(n^2)$
Vizinhos mais Recorrentes	$O(n^2)$

Em seguida foram realizados testes utilizando as bases de dados *MovieLens* 100K e 1M para avaliar as heurísticas. A configuração dos testes segue o modelo descrito no Capítulo 5 que conta 50 execuções para cada heurística seguindo o padrão *10-fold cross validation*. Analisaram-se dois parâmetros: tempo de construção do conjunto dos VGs (dado em segun-

dos) e o erro das predições geradas (obtido pela *Mean Absolute Error*). Como métrica de similaridade optou-se pela correlação de *Pearson*, também utilizada por Boumaza e Brun [11]. Essa métrica é uma das mais recorrentes na literatura na área de recomendação, pois consegue fornecer resultados mais acurados. Nas Tabelas 4.2 e 4.3 têm-se os resultados de cada heurística.

As heurísticas *Média de Similaridade*, *Número de Conexões* e *Vizinhos mais Recorrentes* demandaram tempos de construção semelhantes, pois todas utilizam similaridade para formar seus conjuntos de usuários.

A heurística *Itens Distintos* corresponde a uma variação da heurística proposta com o intuito de reunir usuários com o maior número possível de itens diferentes. Observou-se que os tempos dessas heurísticas foram significativamente menores que as demais. Tal redução deve-se ao fato de utilizarem operações menos custosas para formarem seus conjuntos. Verificou-se também que mesmo havendo aumento dos dados o tempo para construção permaneceu quase constante, em contrapartida naquelas baseadas em similaridade houve um crescimento bastante acentuado. Isso confirma a simplicidade e rapidez de construção da heurística proposta.

Em relação à taxa de erro, as heurísticas dos itens conseguiram fornecer conjuntos mais representativos. Percebe-se que na base *MovieLens* 1M houve uma diminuição nos erros obtidos em todas as heurísticas, colocando-as em um patamar similar no que diz respeito à acurácia, entretanto, o custo computacional daquelas que usam similaridade foi mais elevado.

Tabela 4.2: Resultado da Avaliação das Heurísticas no MovieLens 100K

Heurística	Tempo (segundos)	Erro (MAE)
Usuários mais Ativos	0,001	0,6937
Itens Distintos	0,072	0,6954
Média de Similaridade	1,604	0,7053
Número de Conexões	1,591	0,7046
Vizinhos mais Recorrentes	1,631	0,6994
kNN Padrão	-	0,6826

Tabela 4.3: Resultado da Avaliação das Heurísticas no MovieLens 1M

Heurística	Tempo (segundos)	Erro (MAE)
Usuários mais Ativos	0,002	0,6546
Itens Distintos	1,097	0,6549
Média de Similaridade	92,713	0,6588
Número de Conexões	92,652	0,6579
Vizinhos mais Recorrentes	100,390	0,6553
kNN Padrão	-	0,6500

Os resultados encontrados apontam a heurística dos usuários mais ativos como melhor opção para reduzir o custo computacional do kNN. Acredita-se que isso se deve ao fato desses usuários terem mais chances de serem vizinhos de um número maior de usuários distintos devido à possibilidade de possuírem mais itens em comuns com os demais usuários. Por exemplo, dada uma matriz qualquer Q de usuários e itens, tem-se u_1 como usuário mais ativo. É possível observar que os itens avaliados por u_3, u_5 e u_8 também foram avaliados por u_1 de maneira similar. Dessa forma, u_1 pode ser considerado, simultaneamente, um bom vizinho para u_3, u_5 e u_8.

$$
\mathrm{Q} = \begin{array}{c} \\ u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \\ u_8 \\ u_9 \\ u_{10} \end{array}
\begin{array}{c}
\begin{array}{ccccccccccccccc} i_1 & i_2 & i_3 & i_4 & i_5 & i_6 & i_7 & i_8 & i_9 & i_{10} & i_{11} & i_{12} & i_{13} & i_{14} & i_{15} \end{array} \\
\begin{bmatrix}
4 & & 1 & 3 & 5 & 3 & & 4 & & 4 & 4 & & 2 & 3 & 2 \\
 & & 4 & & & & & 5 & & & 1 & & 4 & & \\
4 & & & & 5 & & & & & & 3 & & 1 & & \\
 & & & & & & 5 & & 2 & 5 & & & 3 & & 4 \\
 & & 1 & 3 & & & & 5 & & & & & & & \\
5 & & & & & & & & & & & & 2 & & 1 \\
4 & & 1 & & & 3 & & & & 5 & & & 2 & & \\
 & & & & & 4 & & & & 4 & & & & 3 & 2 \\
2 & & & & & & & & & & & & 1 & & 3 \\
2 & 1 & & & 3 & & & & 5 & 4 & & & 1 & 3 & 5
\end{bmatrix}
\end{array}
$$

4.3 Configuração dos Parâmetros

Existem dois importantes parâmetros a serem configurados, uma vez que exercem grande influência nos resultados. O primeiro é o número de vizinhos mais próximos k. A maneira mais comum de se determinar o valor de k é mediante uma análise empírica, pois a quantidade ideal de vizinhos pode variar bastante de acordo com o domínio de aplicação. Para a avaliação em questão escolheu-se $k = 30$, pois tal valor conseguiu obter as menores taxas de erros (Figura 4.1)[1].

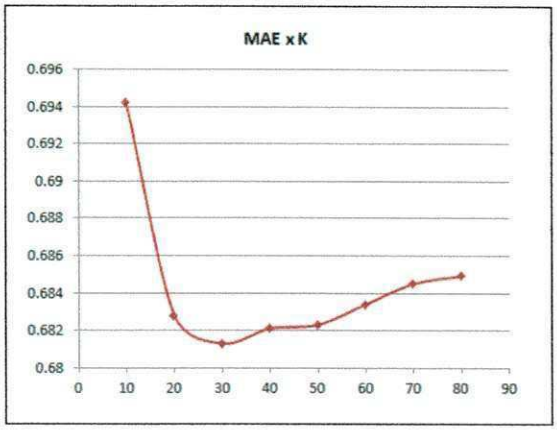

Figura 4.1: Gráfico da Evolução dos Erros de Acordo com a Variação de k

O segundo parâmetro é a porcentagem p de usuários selecionados de acordo com a heurística escolhida. A definição desse parâmetro tem impacto direto no objetivo do trabalho, pois há um *tradeoff* entre o consumo de recursos (tempo e memória) e a acurácia da recomendação. Quanto menor o valor de p menor será o espaço de busca dos vizinhos mais próximos, consequentemente menos cálculos de similaridade serão necessários, pois a proximidade entre dois usuários que não são Vizinhos Globais não é calculada. Além disso, menos memória é demandada, uma vez que a Matriz Similaridade é reduzida. Por outro lado, valores pequenos de p tendem a fornecer resultados menos acurados.

Em Boumaza e Brun [11] utilizou-se $p = 16\%$. Testes foram realizados (Figura 4.2)[2] com a mesma base de dados e escolheu-se um número semelhante ($p = 15\%$), pois tal valor forneceu um balanço ideal entre tempo e acurácia.

[1]Foram realizadas 10 execuções com o kNN padrão, na base *MovieLens* 100K.

[2]Foram realizadas 10 execuções utilizando a heurística dos usuários mais ativos, na base *MovieLens* 100K.

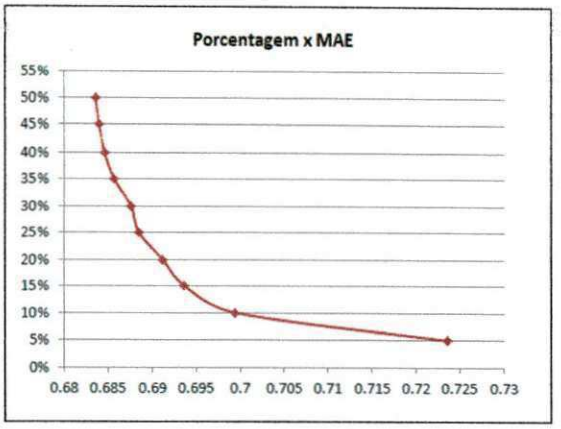

Figura 4.2: Gráfico da Evolução dos Erros de Acordo com a Restrição do Espaço de Busca dos Vizinhos

4.4 Métrica de Similaridade

Como mencionado no Capítulo 2, na área de recomendação a correlação de *Pearson* é uma das mais utilizadas, pois consegue fornecer excelentes resultados. Testes realizados (Figura 4.3)[3] entre a correlação de *Pearson* e a similaridade do Cosseno, indicaram maior nível de acurácia com a utilização da primeira métrica. Por tal motivo, escolheu-se essa métrica para verificação de similaridade entre os usuários.

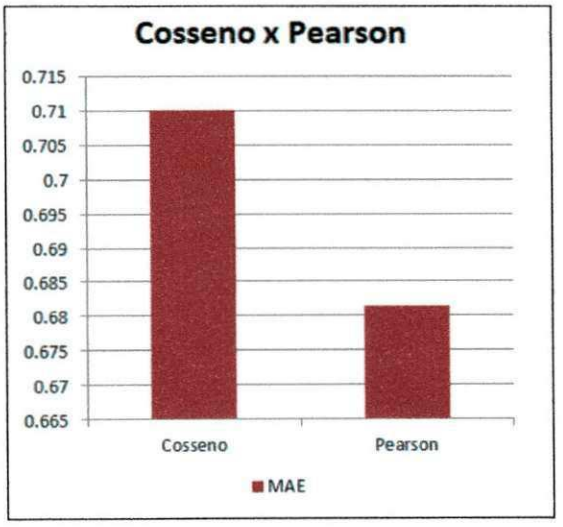

Figura 4.3: Gráfico Comparativo entre a Correlação de Pearson e a Similaridade do Cosseno

Na solução proposta foi utilizada a correlação de *Pearson* em um formato que facilita a

[3]Foram realizadas 10 execuções com o kNN padrão, na base *MovieLens* 100K.

computação, tornando mais rápida a aquisição dos resultados. A fórmula usada é dada pela Equação 4.1.

$$Corr_{P'}(a,u) = \frac{n \sum\limits_{i \in I_{au}} (xy) - \sum\limits_{i \in I_{au}} x \cdot \sum\limits_{i \in I_{au}} y}{\sqrt{[n \sum\limits_{i \in I_{au}} x^2 - (\sum\limits_{i \in I_{au}} x)^2] \cdot [n \sum\limits_{i \in I_{au}} y^2 - (\sum\limits_{i \in I_{au}} y)^2]}} \tag{4.1}$$

Onde $x = r(a,i)$ é a avaliação do usuário a sobre o item i e $\overline{x} = \overline{r(a)}$ é a média das avaliações de a. De forma análoga, tem-se $y = r(u,i)$ e $\overline{y} = \overline{r(u)}$. I_{au} é o conjunto de itens que foram avaliados simultaneamente pelos usuários a e u. Optou-se pela denotação de x e y ao invés de $r(a,i)$ e $r(u,i)$, apenas para facilitar a comparação visual entre as fórmulas.

4.5 Matrizes Similaridade

A heurística de seleção divide a base de dados em dois tipos de usuários: os Vizinhos Globais e os usuários comuns. Uma vez que os vizinhos mais próximos são buscados apenas entre os VGs e não mais em toda a população, o número de cálculos de similaridade diminui, pois não há necessidade de verificar a proximidade entre dois usuários comuns. Sendo assim, foram utilizadas duas Matrizes Similaridade com o intuito de facilitar e acelerar as buscas pelos vizinhos. Um delas armazena as similaridades entre VG (M_{VG}) e a outra entre VG e usuários comuns (M_{UC}).

$$M_{VG} = \begin{array}{c} \\ vg_1 \\ vg_2 \\ vg_3 \\ \\ vg_v \end{array} \begin{array}{c} \begin{array}{ccccc} vg_1 & vg_2 & vg_3 & .. & vg_v \end{array} \\ \begin{bmatrix} s_{11} & & & & \\ s_{21} & s_{22} & & & \\ s_{31} & s_{32} & s_{33} & & \\ \vdots & \vdots & \vdots & \ddots & \\ s_{v1} & s_{v2} & s_{v3} & \cdots & s_{vv} \end{bmatrix} \end{array}$$

$$M_{UC} = \begin{array}{c} \\ uc_1 \\ uc_2 \\ uc_3 \\ \\ uc_c \end{array} \begin{array}{c} \begin{array}{ccccc} vg_1 & vg_2 & vg_3 & \cdots & vg_v \end{array} \\ \begin{bmatrix} s_{11} & s_{12} & s_{13} & \cdots & s_{1v} \\ s_{21} & s_{22} & s_{23} & \cdots & s_{2v} \\ s_{31} & s_{32} & s_{33} & \cdots & s_{3v} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ s_{c1} & s_{c2} & s_{c3} & \cdots & s_{cv} \end{bmatrix} \end{array}$$

4.6 Predição de Notas

A predição de notas foi adaptada a partir do código fonte do método *Weighted User-Based kNN* implementado na ferramenta *MyMediaLite* [23] (uma biblioteca especializada em sistemas de recomendação). A principal diferença reside na forma de seleção dos vizinhos que passou a ser feita no conjunto dos Vizinhos Globais. Um pseudocódigo da função de predição pode ser observado na Figura 4.4. A função recebe como entrada o usuário e o item para o qual deseja-se realizar a predição. Buscam-se os k vizinhos mais próximos do usuário, para cada vizinho encontrado um cálculo ponderado é feito utilizando-se a similaridade entre ambos. Ao final, tem-se o valor de predição para o usuário sobre o item.

4.7 Considerações Finais do Capítulo

Neste capítulo apresentou-se uma abordagem de recomendação centrada na filtragem colaborativa. A solução proposta deriva-se do kNN padrão, com o intuito de diminuir o custo computacional que se eleva com o crescimento do volume dos dados.

O principal ponto da solução reside na definição de uma heurística rápida e eficiente para restringir o espaço de busca dos vizinhos mais próximos. Trabalhos recentes [55, 64] vêm utilizando uma técnica chamada *MapReduce* [19] para melhorar o desempenho do kNN. *MapReduce* é um modelo de programação proposto pelo *Google* com o intuito de paralelizar o processamento de grandes volumes de dados. Devido a restrições de tempo para conclusão

Algoritimo1 Função de Predição

```
1:  function GERARPREDICAO(usuario, item)
2:      resultado ← predicaoBase(usuario, item)
3:      soma ← 0
4:      somaPonderada ← 0
5:      numeroDeVizinhos ← K
6:      listaDeVizinhos ← getVizinhos(usuarios)
7:      for (i = 0; listaDeVizinhos.size(); i++) do
8:          if (itemFoiAvaliado(vizinho, item) == true) then
9:              avaliacao ← getAvaliacao(vizinho, item)
10:             similaridade ← getSimilaridade(usuario, vizinho)
11:             somaPonderada ← somaPonderada + similaridade
12:             soma ← similaridade + (avaliacao − predicaoBase(vizinho, item))
13:             if (−− numeroDeVizinhos == 0) then
14:                 break
15:             end if
16:         end if
17:     end for
18:     if (peso! = 0) then
19:         resultado ← resultado + (soma/somaPonderada)
20:     end if
21:     if (resultado > avaliacaoMaxima) then
22:         resultado ← avaliacaoMaxima
23:     end if
24:     if (resultado < avaliacaoMinima) then
25:         resultado ← avaliacaoMinima
26:     end if
27:     resultado ← arredeondar(resultado)
          return resultado
28: end function
```

Figura 4.4: Pseudocódigo da Função de Predição

da pesquisa não foi possível investigar os efeitos do paralelismo em conjunto com a solução proposta, porém pretende-se, como trabalho futuro, incluir tal investigação no escopo.

Com o método proposto evidenciou-se (Capítulo 5) a redução da quantidade de memória utilizada, a diminuição do número dos cálculos de similaridade bem como o tempo de geração das recomendações, mantendo todavia um nível satisfatório de acurácia.

Capítulo 5

Avaliação

Neste capítulo realizou-se um experimento com o intuito de avaliar a solução proposta. Foram utilizadas duas bases de dados reais de filmes. O foco recai sobre a análise de dois parâmetros principais: tempo de execução e acurácia. Como forma de validação, comparou-se a solução proposta com as principais abordagens que surgiram na literatura para melhorar o desempenho do kNN. Por fim, apresenta-se o experimento realizado, a análise estatística e discutem-se os resultados alcançados.

5.1 Dados Utilizados

Inicialmente foram realizadas buscas com o intuito de encontrar bases públicas do *Google Play* ou outras lojas de aplicativos para dispositivos móveis, porém sem sucesso. Dessa forma, optou-se por bases de um domínio semelhante. Foram utilizadas duas bases de dados reais muito populares na área de recomendação, provenientes do *MovieLens*[1]. Os dados foram coletados durante 07 meses no período de 19 de setembro de 1997 a 22 de abril de 1998. Usuários com menos de 20 filmes avaliados foram excluídos do conjunto. Essa filtragem tem como objetivo remover dados considerados ruídos, ou seja, são informações de pouca ou nenhuma relevância para o conjunto geral. Por exemplo, se um usuário avaliou apenas dois ou três filmes esses dados são insuficientes para gerar um perfil relevante para o

[1]Outra base popular na área de recomendação é a do *NetFlix*, porém é um conjunto de dados de grande volume e devido a limitações de hardware tal base não foi utilizada nesse trabalho. Além disso, a autorização para utilização pública dessa base foi revogada há algum tempo

usuário. Em geral todas as bases para fins acadêmicos, com exceção daquelas destinadas a investigação do problema de *Cold-start*, passam por filtragem semelhantes.

As bases utilizadas possuem as seguintes características:

(a) MovieLens 100K

- 943 usuários;
- 1.682 filmes;
- 100.000 avaliações (escala inteira de 1 a 5);
- Grau de Esparsidade: 93,7%.

(b) MovieLens 1M

- 6.040 usuários;
- 3.952 filmes;
- 1.00.209 avaliações (escala inteira de 1 a 5);
- Grau de Esparsidade: 95,8%.

Ambas as bases apresentam um grau bastante elevado de esparsidade[2], o que significa que as Matrizes Usuário-Item de cada uma possuem uma grande quantidade de itens não-avaliados. Tal situação consequentemente leva a um declínio no desempenho dos algoritmos de recomendação.

5.2 Parâmetros Analisados

O desempenho da solução proposta, bem como dos métodos comparados neste trabalho, foram analisados sob dois parâmetros: tempo de execução (em segundos) e acurácia (MAE).

O tempo de execução foi dividido em duas etapas: fase de preparação e realização. A fase de preparação corresponde a criação de estruturas de dados necessárias para a execução dos métodos como, por exemplo, a Matriz Similaridade para o kNN tradicional e a construção

[2]O grau de esparsidade é dado pela razão entre o número total de avaliações possíveis e a quantidade de avaliações realizadas.

da árvore para o *kd tree*. A etapa de realização consistiu na formação das vizinhança dos usuários alvo e na geração das predições.

Existem diversas formas de avaliar a acurácia de um sistema de recomendação (Capítulo 2). No cenário de predição de notas, tipicamente, utilizam-se métricas baseadas em erro. Boumaza e Brun [11] utilizaram *Mean Absolute Error* para avaliar o método que desenvolveram. Uma vez que a solução proposta neste documento corresponde a uma extensão do trabalho em [11], optou-se por usar a mesma métrica.

5.3 Ambiente e Ferramentas Utilizadas

O experimento foi realizado em uma máquina com processador *Core* i7 2300K de 3,4Ghz, 8Gb de memória RAM DDR3 e sistema operacional *Windows* 7 de 64 bits. A solução proposta foi codificada com auxílio do ambiente de desenvolvimento *Eclipse*[3] (Kepler) em sua versão de 64 bits para desenvolvedores. O Eclipse é um ambiente integrado de desenvolvimento com foco em Java, porém com suporte a outras linguagens de programação.

O método desenvolvido utilizou como base o código fonte da ferramenta *MyMediaLite* [23], uma biblioteca especializada em sistemas de recomendação. Ela foi criada por desenvolvedores da Universidade de *Hildesheim*, localizada na Alemanha. Essa ferramenta fornece algoritmos pertencentes ao estado da arte de dois cenários bastante comuns na filtragem colaborativa: predição de itens e de notas. Além disso, é uma biblioteca de código aberto e muito utilizada em competições na área de recomendação.

Outra ferramenta utilizada no trabalho foi a *Java Statistical Analysis Tool* (JSAT) [52], uma biblioteca codificada em linguagem Java, que possui dezenas de algoritmos de aprendizado de máquina. JSAT foi criado por Edward Raff [4], Mestre em Ciência da Computação pela Universidade *Purdue*, localizada em *West Lafayette*, Indiana. O autor mantém um Blog[5], em que discorre sobre as implementações contidas na biblioteca e discute outros assuntos da área de Aprendizado de Máquina. Algumas das abordagens utilizadas na etapa de validação são oriundas dessa ferramenta.

[3] www.eclipse.org

[4] https://plus.google.com/104543220538859537467

[5] www.jsatml.blogspot.com.br

A análise estatística foi conduzida com o auxílio do R[6], uma ferramenta que reúne um conjunto de técnicas computacionais que facilitam a análise dos dados, cálculo numéricos e produção gráfica.

5.4 Validação

O processo de validação consistiu em comparar a solução proposta com abordagens voltadas para a melhoria do desempenho do kNN. Os métodos foram avaliados no cenário de predição de notas, ao qual destina-se a solução proposta. Os dados utilizados foram divididos em conjuntos disjuntos de treinamento e teste.

5.4.1 Abordagens Avaliadas

Foram selecionados métodos pertencentes ao âmbito de classificação e à área de recomendação. As abordagens divergem, principalmente, na organização e escolha dos vizinhos mais próximos. São elas:

- **k-Dimensional Tree (kd-tree):** implementado na ferramenta JSAT. Corresponde à proposição tradicional;
- **Locality Sensitive Hash (LSH):** implementado na ferramenta JSAT. Foi baseado no algoritmo E2LSH, apresentado por Datar et al. [18];
- **k-Nearest Neighbors (kNN):** corresponde ao kNN padrão para predição de notas, cuja implementação baseou-se no código-fonte da ferramenta *MyMediaLite*;
- **Iterated Local Search (ILS) kNN:** corresponde ao método apresentado por Boumaza e Brun [11]. Na validação utilizou-se também uma restrição 15% para o espaço de busca dos vizinhos.

Utilizou-se o mesmo valor de k para avaliação em ambas as bases de dados. De acordo com testes realizados (Figura 4.1) escolheu-se o k que forneceu o melhor balanço entre acurácia e tempo de busca para o domínio em questão, no caso $k = 30$.

[6]http://www.r-project.org/

O algoritmo para predição é comum a todas as abordagens. No cenário de predição de notas, geralmente as avaliações preditas são obtidas a partir de um cálculo ponderado que envolve as similaridades entre os usuários (Figura 4.4). Observações empíricas indicaram a correlação de *Pearson* como métrica mais acurada para tal tarefa, por isso, optou-se por utilizá-la na etapa de realização. É importante destacar que nessa fase as similaridades foram computadas no instante da predição, com o intuito de evitar qualquer viés, devido a valores previamente calculados.

5.4.2 Divisão dos Dados

Ao se estimar a qualidade de um sistema de recomendação é importante que os dados sejam divididos corretamente [17]. Em geral, os dados são divididos em treinamento e teste. O conjunto de treino normalmente possui a maior parte dos dados e é utilizado para fornecer conhecimento ao método sobre o problema em questão. Em contrapartida, os dados de teste são usados para avaliar o desempenho do método. É fundamental que os conjuntos sejam disjuntos, ou seja, uma avaliação de um usuário para um item que pertença ao conjunto de treino não pode estar presente nos dados de teste e vice-versa.

Uma das maneiras mais comuns de se dividir os dados é a partir do método *n-fold cross-validation* que consiste em separar a base de dados em n partes independentes, dessa forma as partes não se sobrepõem. Cada parte é utilizada apenas uma vez como conjunto de teste e as restantes são usadas para treinamento. De acordo com Cremonesi et al. [17], um bom valor para n é 10. Portanto, as bases de dados do *MovieLens* (100K e 1M) foram divididas seguindo o processo *10-fold cross-validation*, separando-se assim 90% dos dados para treinamento e 10% para teste. Esse processo é repetido 5 vezes, totalizando uma quantidade final de 50 amostras.

5.5 Resultados

Foram realizadas 50 execuções para cada abordagem avaliada. As médias dos resultados estão presentes nas Tabelas 5.1 e 5.2. Os valores detalhados de cada execução podem ser visualizados no Apêndice A.

Tabela 5.1: Resultados das Execuções no MovieLens 100K

Método	Preparação (segundos)	Realização (segundos)	MAE
kd-tree	0,61	21,63	0,8333
LSH	0,07	11,06	0,7528
ILS	1231,46	1,69	0,7083
kNN	1,30	3,05	0,6826
SP	0,52	1,93	0,6937

Tabela 5.2: Resultados das Execuções no MovieLens 1M

Método	Preparação (segundos)	Realização (segundos)	MAE
kd-tree	9,24	1566,77	0,7875
LSH	0,33	454,09	0,7014
ILS	7826,24	67,46	0,6591
kNN	91,37	150,76	0,6500
SP	36,14	64,73	0,6546

A solução proposta e o kNN padrão apresentaram os melhores resultados dentre as abordagens testadas. Com o objetivo de verificar o real desempenho desses dois métodos, foram realizadas novas execuções, porém modificando a função de predição para que fossem utilizadas as similaridades previamente calculadas e armazenadas nas Matrizes Similaridades de cada abordagem. Os resultados das novas execuções podem ser visualizados na Tabelas 5.3 e 5.4. É possível observar uma redução significativa no tempo de realização de ambos os algoritmos.

Tabela 5.3: Solução Proposta x kNN Padrão no MovieLens 100K

Método	Preparação (segundos)	Realização (segundos)	MAE
kNN Padrão	1,42	1,35	0,6826
SP	0,57	0,28	0,6937

5.6 Análise Estatística

Em geral, um dos primeiros passos em um processo de análise estatística corresponde à análise gráfica dos dados envolvidos. O estudo visual fornece informações importantes sobre o comportante dos dados e auxilia na tomada de decisão sobre a aplicação de outros testes

Tabela 5.4: Solução Proposta x kNN Padrão no MovieLens 1M

Método	Preparação (segundos)	Realização (segundos)	MAE
kNN Padrão	91,45	108,66	0,6500
SP	36,10	20,45	0,6546

que eventualmente sejam necessários.

O processo estatístico conduzido neste trabalho, teve apenas como base a análise visual das amostras obtidas no experimento, uma vez que após a observação do comportamento dos dados tornou-se dispensável realizar testes estatísticos adicionais.

O estudo visual foi realizado com o auxílio de *boxplots* [6, 62]. Esse tipo de gráfico permite representar a distribuição de um conjunto de dados a partir de uma linha, que vai desde o valor mínimo ao valor máximo com linhas verticais (ou horizontais) desenhadas no primeiro quartil, na mediana e no terceiro quartil. Os quartis dividem os dados (ordenados) em quatro grupos com aproximadamente 25% dos valores em cada (Figura5.1).

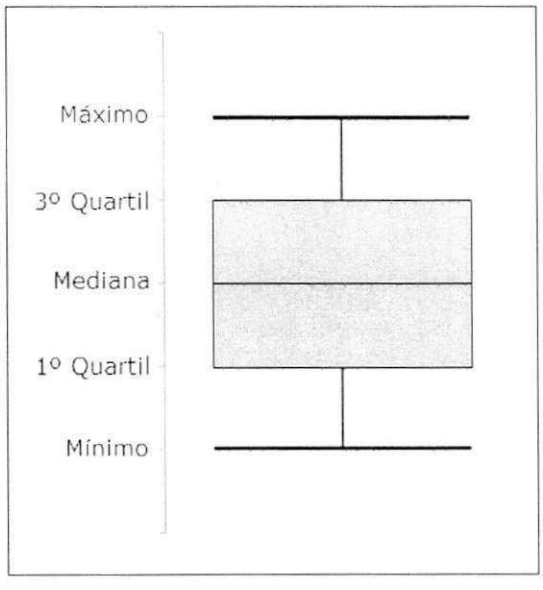

Figura 5.1: Exemplo de um *boxplot*.

Os *boxplots* são bastante utilizados para comparação de grupos de dados. A partir de uma análise visual é possível determinar se há diferenças entre dois ou mais conjuntos. Por exemplo, na Figura 5.2(a) é possível afirmar que os grupos A e B são diferentes (sendo A maior que B), pois não há sobreposição das caixas dos gráficos, o que significa que 75%

dos valores de A estão abaixo de 75% de B. Na Figura 5.2(b) tem-se que 75% dos valores de A são inferiores a 50% de B, dessa forma, é provável que B seja maior que A. Na Figura 5.2(c) há um sobreposição entre as medianas de ambos os grupos, por isso não é possível afirmar se há diferenças entre A e B. Nos dois últimos casos são necessários testes estatísticos adicionais para determinar se realmente há diferenças entres os grupos, enquanto que na Figura 5.2(a) apenas a análise gráfica é suficiente.

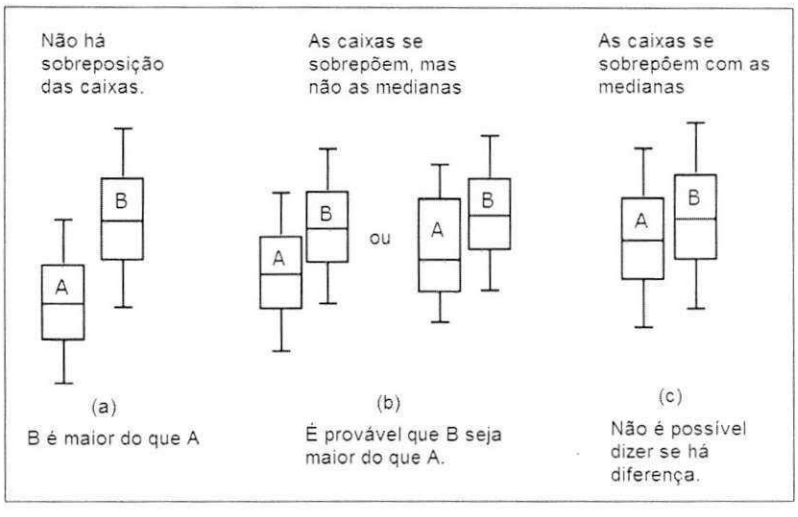

Figura 5.2: Análise de Boxplots

5.6.1 Tempo Total de Execução

Nas Figuras 5.3 e 5.5 têm-se os *boxsplots* dos tempos totais das execuções (tempo de preparação mais realização) dos métodos avaliados no *MovieLens* 100K e 1M, respectivamente. Percebe-se que os gráficos estão pouco legíveis devido a grande diferença dos valores no eixo Y, o que dificulta a realização de um estudo visual. Sendo assim, optou-se pela separação dos *boxsplots* em planos individuais, como é possível observar nas Figuras 5.4 e 5.6.

A análise dos gráficos indicou que os grupos avaliados são diferentes, sendo a solução proposta responsável pelo conjunto de dados com os menores tempos das execuções. Uma vez que não há sobreposição das caixas ou das medianas, não se faz necessária a aplicação de testes adicionais.

5.6.2 Erro das Predições

Nas Figuras 5.9 e 5.10 são apresentados os *boxsplots* dos valores referentes aos erros das predições das abordagens no *MovieLens* 100K e 1M, respectivamente. Observa-se que não há sobreposição entre os gráficos, portanto, conclui-se que os grupos são diferentes. Segundo a análise visual, o kNN padrão foi o método que forneceu o conjunto com os menores erros das predições.

Solução Proposta x kNN Padrão

Nas Figuras 5.7 e 5.8 apresentam-se os gráficos dos tempos das execuções, após a mudança na função de predição com o intuito de verificar o real desempenho da abordagem proposta e do kNN padrão. Com o estudo visual, conclui-se que a solução proposta possui o conjunto com os menores tempos das execuções.

5.7 Discussão

O kNN padrão, SP e ILS são métodos desenvolvidos especificamente com o propósito de gerar recomendações. Essas técnicas utilizam a correlação de *Pearson* para selecionar os vizinhos mais próximos. Essas abordagens conseguiram fornecer resultados mais acurados que as demais. A SP apresentou uma média de erros ligeiramente acima do kNN padrão (apenas 0,7% na maior base). Entretanto, levando-se em conta a redução alcançada do custo computacional, o resultado é bastante expressivo, o que comprova a eficácia da heurística de seleção baseada nos usuários mais ativos.

Quanto a análise dos tempos totais das execuções[7] dos três métodos anteriormente citados, a SP obteve o melhor desempenho, demandando em média 72% menos tempo que o segundo método mais rápido (kNN padrão), quando executados na maior base de dados (*MovieLens* 1M). Esse ganho foi obtido, principalmente, devido à redução da quantidade de cálculos de similaridades realizados na etapa de preparação e à restrição do espaço de busca dos vizinhos. O tempo de preparação do ILS ficou muito acima dos demais, pois sua função objetivo baseia-se, dentre outras coisas, nos erros das predições geradas a partir do conjunto

[7] soma dos tempos de preparação e de realização

dos Vizinhos Globais. A verificação dos erros é uma etapa custosa, porque envolve a formação de vizinhanças distintas para cada um dos usuários. Esse processo é repetido diversas vezes até que o conjunto final dos VGs seja encontrado. Além disso, por ser um método não-determinístico não há controle sobre seu tempo de execução, dessa forma, optou-se por estabelecer um limite de tempo[8] para a *procedure IteratedLocalSearch* (20 e 90 minutos no *MovieLens* 100K e 1M, respectivamente), descrito em [11]. A média dos erros das predições do ILS está próxima daquela fornecida pelo kNN padrão, porém seu acentuado custo computacional pode torná-lo impraticável.

O LSH e o kd-tree são algoritmos de busca associativa, criados com o objetivo de acelerar o processo de seleção de instâncias similares a um objeto de interesse. Esses métodos são reconhecidos na literatura como alternativas eficazes ao método tradicional de busca. Tanto o LSH quanto o kd-tree utilizam distância Euclidiana e são destinados originalmente ao âmbito de classificação de objetos, o qual pode apresentar características diferentes da área de recomendação. De acordo com a literatura, ambos os métodos quando executados em domínios apropriados, são capazes de atingir patamares de tempos logarítmicos. Entretanto, tal comportamento não foi observado durante o experimento, em nenhuma das duas abordagens.

O kd-tree apesar de lidar bem com bases de dados com milhões de objetos, tende a perder significativamente o desempenho com o aumento das dimensões dos vetores de características ($d \geq 10$) [22, 24], problema conhecido como maldição da dimensionalidade (em inglês, *curse of dimensionality*) [44, 51]. Na prática, a maldição da dimensionalidade implica que existe um número máximo de dimensões a partir do qual o desempenho do classificador irá decair ao invés de melhorar, pois os dados se tornam mais esparsos e pouco similares. As bases utilizadas possuem dados esparsos e de alta dimensão, fatores que contribuíram diretamente para o baixo desempenho registrado do kd-tree, tanto em relação aos erros das predições quanto aos tempos das execuções. Isso indica que a forma de estruturação dos dados não fornece bons vizinhos para o domínio em questão.

O LSH é um método que prioriza o tempo ao invés da acurácia, isto é, as buscas retornam

[8]Observou-se que algumas das execuções, no *MovieLens* 1M, ultrapassaram o limite. Isso ocorre devido a possibilidade do algoritmo ficar preso em soluções ótimas locais antes de atingir o trecho de código que finaliza o método pelo tempo excedido.

vizinhos aproximados, diferente da maneira tradicional, no qual verifica-se a similaridade entre todos os usuários para que sejam retornados aqueles com os maiores valores de similaridade. O LSH é indicado para domínios nos quais as buscas aproximadas não exercem grande impacto sobre o resultado do algoritmo, pois ao final têm-se maiores erros, comportamento confirmado pelos resultados alcançados. Por outro lado, o tempo de execução embora tenha sido menor que o do kd-tree, ainda ficou bem distante do kNN padrão.

O kNN padrão, SP e ILS utilizam a Matriz Similaridade para armazenar as proximidades entre os usuários e assim evitar repetições de cálculos desnecessárias. Inicialmente suspeitou-se que a diferença dos tempos das execuções do LSH e kd-tree em relação ao kNN padrão seria devido à utilização da matriz, porém a biblioteca JSAT também utiliza uma estrutura auxiliar, chamada *distcache*, que armazena as distâncias já computadas. Foram realizados testes (não documentados), que mostraram que sem esse mecanismo os tempos das execuções aumentavam significativamente. Portanto, constatou-se que a diferença de desempenho entre os métodos não é ocasionada pela falta de estruturas para armazenamento das distâncias.

5.8 Considerações Finais do Capítulo

Neste capítulo descreve-se o experimento realizado para avaliar a solução proposta. Compara-se o método desenvolvido com abordagens que utilizam o conceito de busca dos vizinhos mais próximos, para classificação de objetos e para geração de recomendações.

A motivação desse trabalho foi elaborada a partir de domínio do *Google Play*, porém o trabalho foi validado em um domínio diferente devido a escassez de dados públicos do domínio de aplicativos móveis. Apesar de ambos os domínios apresentarem algumas características semelhantes (tipo de avaliação, vetores de características esparsos, etc) é que tal situação pode representar uma limitação para abordagem proposta.

Os resultados mostram que o método proposto consegue melhorar o desempenho do kNN tradicional. Através da heurística de seleção dos usuários mais ativos, consegue-se reduzir o número de cálculos de similaridade, a quantidade de memória principal utilizada e o tempo de busca dos vizinhos mais próximos, porém mantendo o nível de acurácia das recomendações satisfatório. Com a redução do custo computacional do kNN, a utilização do algoritmo

torna-se perfeitamente aplicável em grande bases de dados. Retomando o exemplo apresentado na Seção 1.1, no qual para uma base com 1 milhão de usuários seria necessário aproximadamente 1 semana para executar outra vez o algoritmo, observa-se que utilizando a solução proposta esse tempo teria uma redução de 72%, o que tornaria o método viável ao exemplo citado, pois seriam necessários pouco menos de 2 dias para reexecutar todo o método. Isso demonstra a importância de utilizar um espaço de busca reduzido ao invés dos 100% originais. Embora não seja possível afirmar, acredita-se que com o crescimento da base de dados seja possível utilizar porcentagens inferiores aos 15% propostos e conseguir resultados tão acurados quanto os do kNN padrão. Observou-se que mantendo tal porcentagem e aumentando o conjunto de dados, consegue-se diminuir a taxa de erro nas predições, ou seja, é provável que em bases com mais volume, mesmo reduzindo o espaço de busca dos vizinhos mais próximos o erro permaneça constante, o que proporcionaria uma maior redução no custo computacional.

Ademais, constatou-se que os métodos de classificação (kd-tree e LSH) implementados na biblioteca JSAT, não se mostraram adequados para o cenário de predição de notas, pertencente à filtragem colaborativa. Os desempenhos desses métodos se mostraram aquém do esperado tanto em relação ao tempo de execução quanto à taxa de erro nas predições.

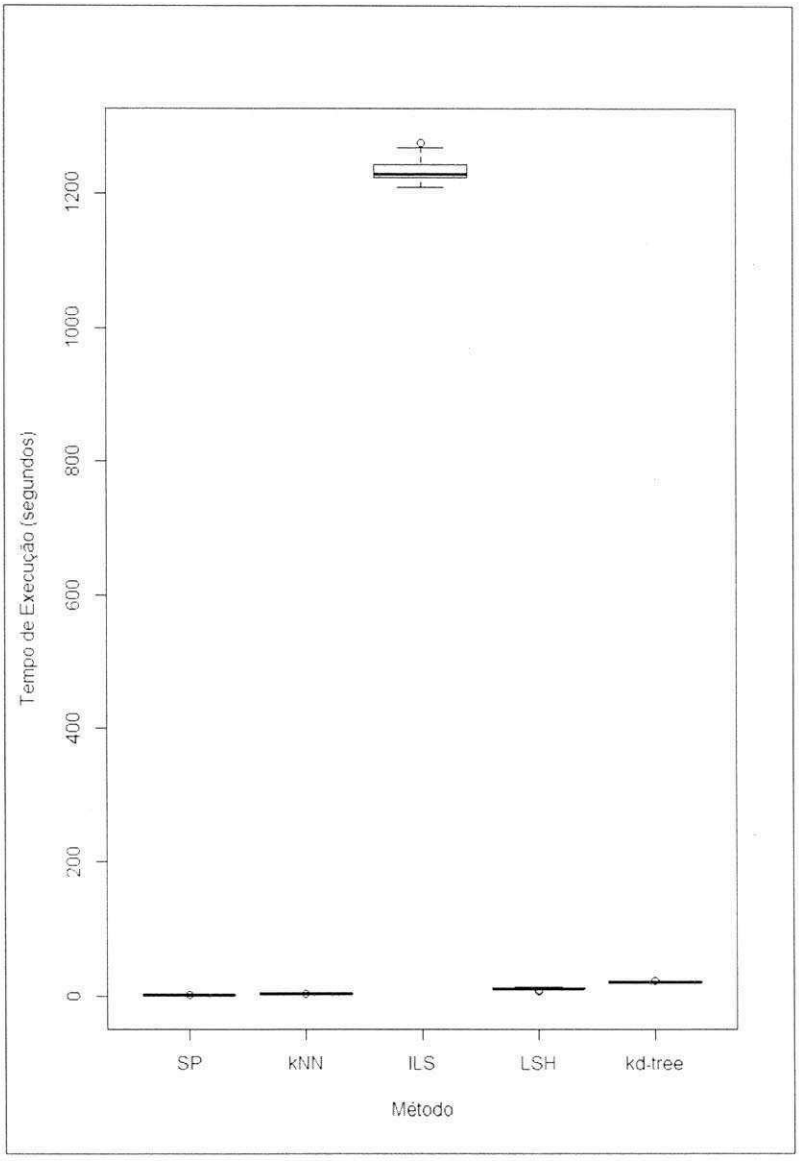

Figura 5.3: Boxplots dos Tempos das Execuções no MovieLens 100K.

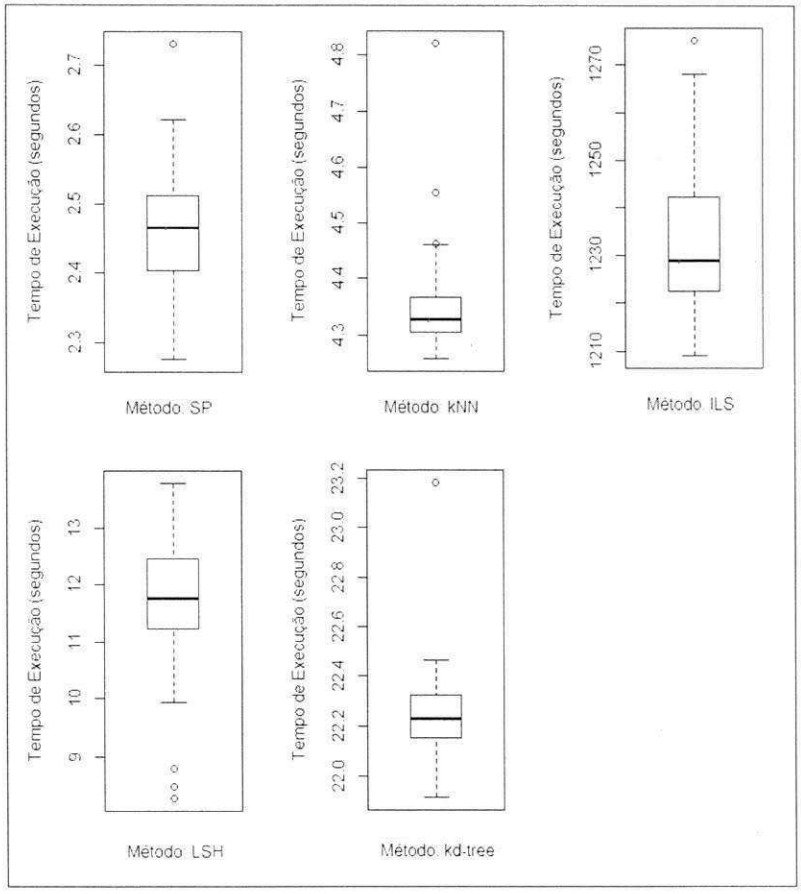

Figura 5.4: Boxplots Individuais dos Tempos das Execuções no MovieLens 100K.

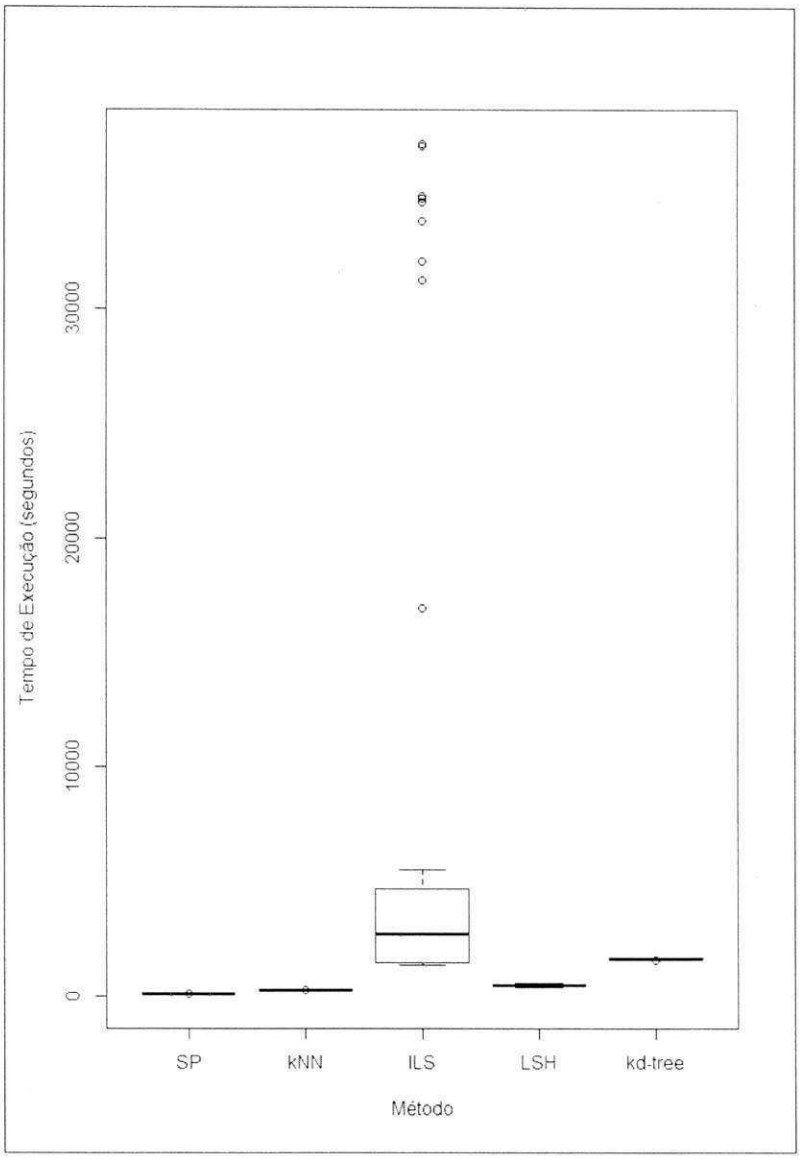

Figura 5.5: Boxplots dos Tempos das Execuções no MovieLens 1M.

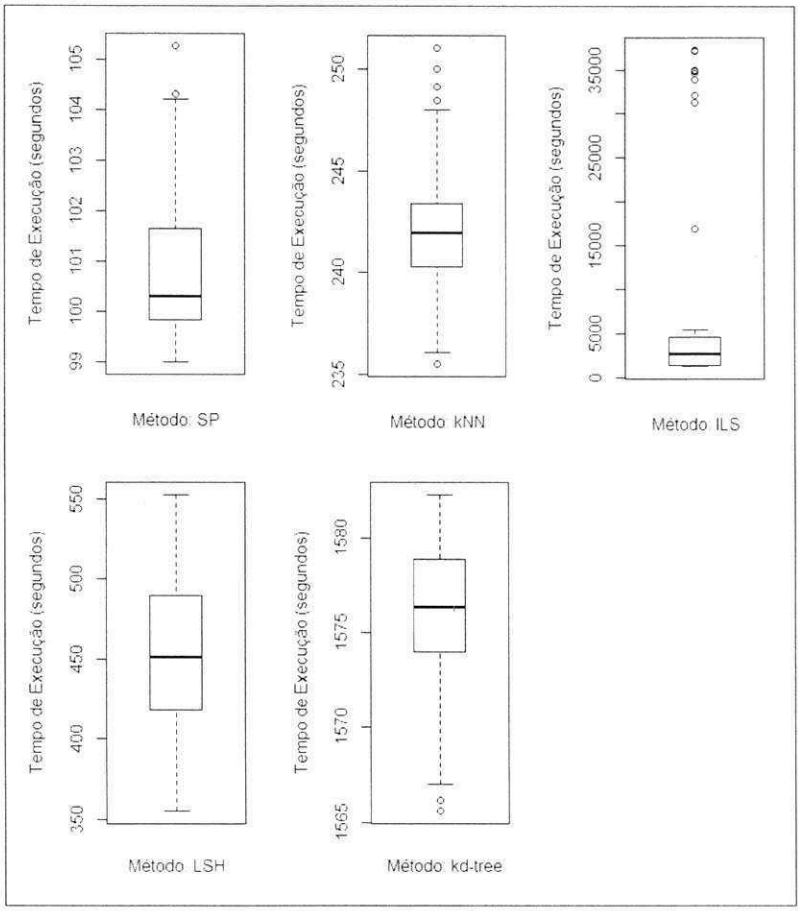

Figura 5.6: Boxplots Individuais dos Tempos das Execuções no MovieLens 1M.

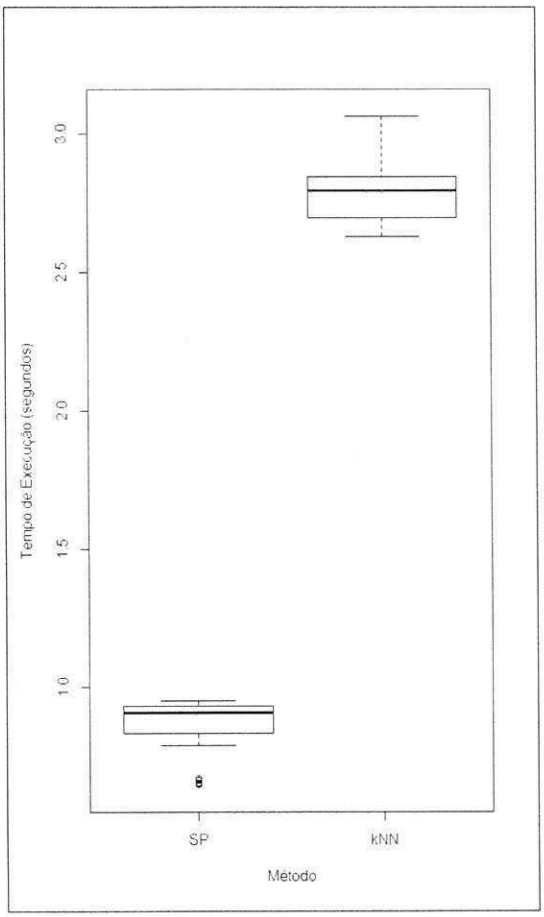

Figura 5.7: Tempos das Execuções SP x kNN no MovieLens 100K (desempenho real).

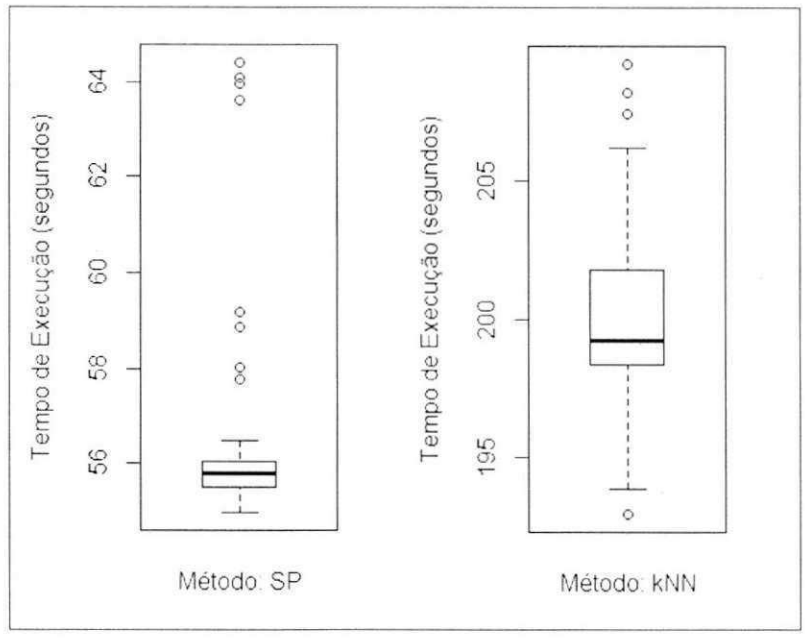

Figura 5.8: Tempos das Execuções SP x kNN no MovieLens 1M (desempenho real).

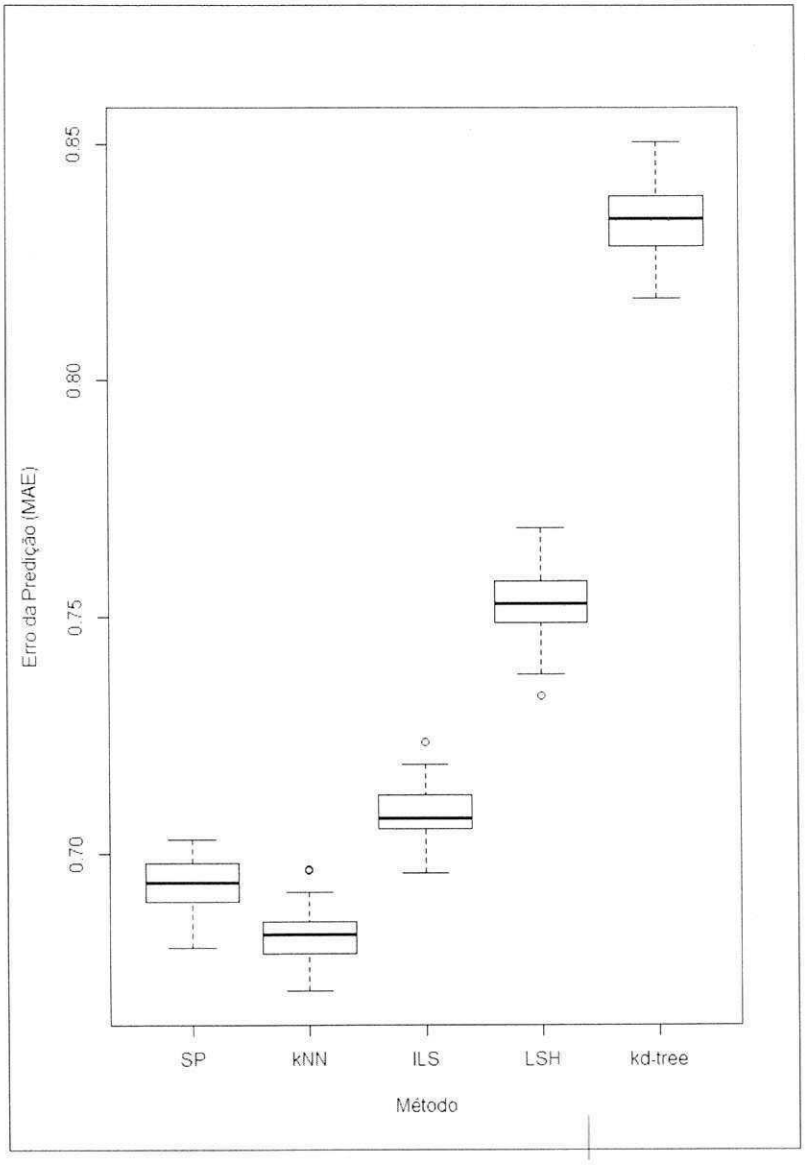

Figura 5.9: Boxplots dos Resultados dos Erros das Predições no MovieLens 100K.

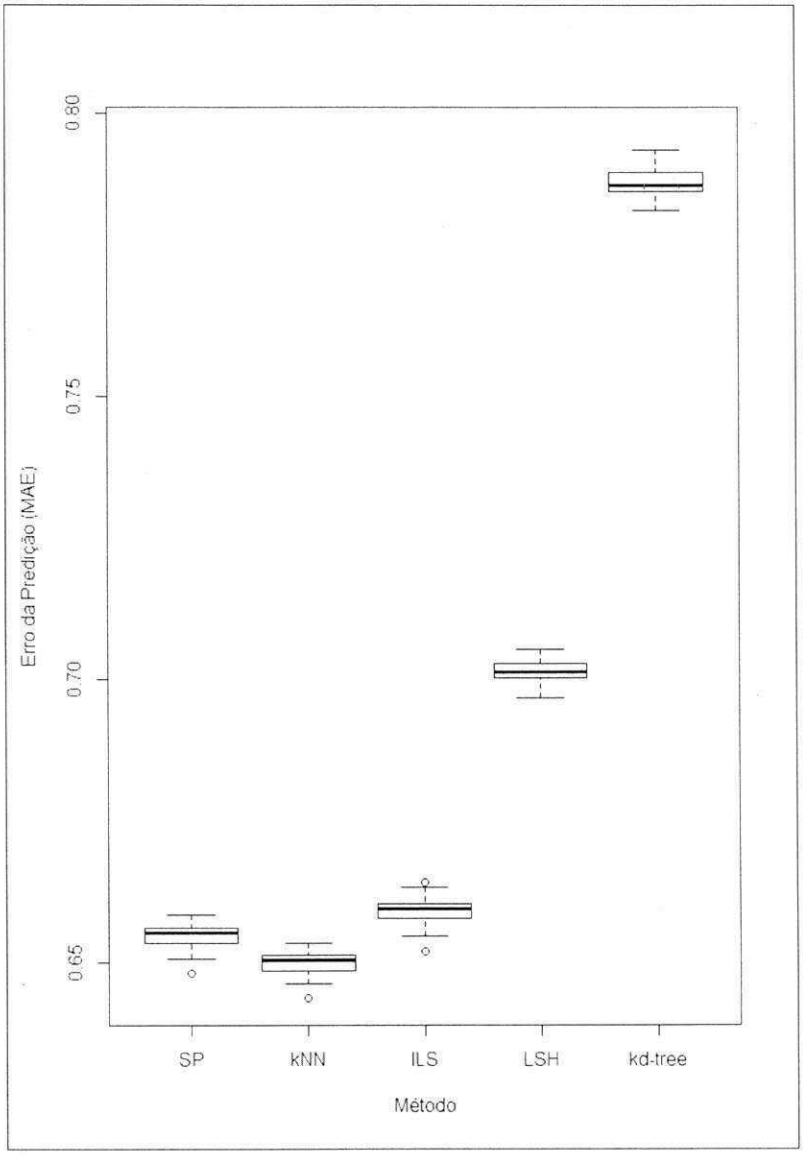

Figura 5.10: Boxplots dos Resultados dos Erros das Predições no MovieLens 1m.

Capítulo 6

Considerações Finais

Neste capítulo expõem-se as considerações finais referentes ao método de recomendação colaborativa desenvolvido e suas principais limitações. Em seguida apresentam-se sugestões para dar continuidade à pesquisa documentada.

6.1 Conclusões

Neste trabalho foi descrita e implementada uma abordagem de recomendação colaborativa com o intuito de melhorar o desempenho do método *k Nearest Neighbors* que é um dos algoritmos mais populares para a realização de buscas associativas. O método apresentado centra-se na ideia da restrição do espaço de busca dos vizinhos mais próximos. Para tanto, foram testadas diversas heurísticas de seleção de usuários e a que forneceu melhores resultados baseou-se na seleção dos usuários mais ativos, ou seja, aqueles que mais avaliaram itens.

Realizou-se um experimento com duas bases de dados reais de filmes, disponibilizadas pelo *MovieLens*. No experimento foram analisados, principalmente, dois parâmetros: tempo de execução (em segundos) e acurácia (*Mean Absolute Error*). A abordagem proposta foi validada comparando-a com vários métodos, tais como, *k-Dimensional Tree*, *Locality Sensitve Hash*, kNN padrão para predição de notas, dentre outros. Como resultado a solução proposta, utilizando apenas 15% do espaço de busca original dos vizinhos mais próximos, obteve um tempo de execução até 72% menor e uma taxa de erro apenas 0,7% maior, em relação ao kNN tradicional.

A heurística de seleção dos usuários mais ativos mostrou-se extremamente rápida na construção do conjunto dos Vizinhos Globais e bastante eficiente em relação à acurácia alcançada.

Como contribuições do trabalho, têm-se:

- Desenvolvimento de um método de recomendação colaborativa baseado no kNN com custo computacional reduzido;
- Proposição de um heurística de seleção rápida e eficaz para acelerar o processo de busca dos vizinhos mais próximos.

6.2 Limitações e Trabalhos Futuros

As principais limitações do trabalhos são:

- A abordagem proposta foi validade apenas em um domínio, o que impossibilita a generalização dos resultados alcançados. A solução proposta visa obter um ganho de desempenho em troca de uma redução da acurácia que no domínio avaliado verificou-se um aumento de 0,7% nos erros das predições (no *MoviLens* 1M), entretanto, não é possível garantir que o nível de acurácia alcançada se mantenha constante em outros domínios.
- A solução proposta destina-se a filtragem colaborativa para predição de notas. Não investigou-se o comportamento em outros cenários, como por exemplo, predição de itens com avaliações implícitas.

Como trabalhos futuros, têm-se:

- Investigar os efeitos do método proposto em bases de dados com milhões de usuários e itens, pois observou-se que a diferença de acurácia entre a solução proposta e o kNN padrão diminuiu com o aumento do conjunto dos dados. Com a base *MovieLens* 100K obteve-se uma diferença absoluta de $0,0111$. Enquanto que com a *MovieLens* 1M a diferença reduziu-se a $0,0046$. Sendo assim, há indícios de que em bases muito grandes o método proposto seja tão ou até mais acurado que o kNN tradicional, porém com um custo computacional significativamente menor;

- A porcentagem p de Vizinhos Globais foi definida a partir de análise empírica e acredita-se que com o crescimento do volume da base, o valor de p possa ser reduzido e ainda assim fornecer bons níveis de acurácia. Portanto, pretende-se criar um método para definição dinâmica de p;
- Investigar os efeitos da inserção de formas de paralelização na busca dos vizinhos mais próximos no espaço restrito;
- Avaliar novas heurísticas de seleção dos usuários;
- Investigar o comportamento em cenários para predição de itens, com avaliações implícitas.

Bibliografia

[1] Taufik Abidin, Amal Perera, Masum Serazi, and William Perrizo. Vertical set square distance: A fast and scalable technique to compute total variation in large datasets. In Gongzhu Hu, editor, *Computers and Their Applications*, pages 60–65. ISCA, 2005.

[2] Taufik Abidin and William Perrizo. Smart-tv: A fast and scalable nearest neighbor based classifier for data mining. In *Proceedings of the 2006 ACM Symposium on Applied Computing*, SAC '06, pages 536–540, New York, NY, USA, 2006. ACM.

[3] Amr Ahmed, Bhargav Kanagal, Sandeep Pandey, Vanja Josifovski, Lluis Garcia Pueyo, and Jeff Yuan. Latent factor models with additive and hierarchically-smoothed user preferences. In *Proceedings of the Sixth ACM International Conference on Web Search and Data Mining*, WSDM '13, pages 385–394, New York, NY, USA, 2013. ACM.

[4] Amos Azaria, Avinatan Hassidim, Sarit Kraus, Adi Eshkol, Ofer Weintraub, and Irit Netanely. Movie recommender system for profit maximization. In *Proceedings of the 7th ACM Conference on Recommender Systems*, RecSys '13, pages 121–128, New York, NY, USA, 2013. ACM.

[5] Ana Belén Barragáns-Martínez, Enrique Costa-Montenegro, Juan C. Burguillo, Marta Rey-López, Fernando A. Mikic-Fonte, and Ana Peleteiro. A hybrid content-based and item-based collaborative filtering approach to recommend tv programs enhanced with singular value decomposition. *Inf. Sci.*, 180(22):4290–4311, November 2010.

[6] Yoav Benjamini. Opening the box of a boxplot. *The American Statistician*, 42(4):257–262, 1988.

[7] J. Bennett and S. Lanning. The netflix prize. In *Proceedings of the KDD Cup Workshop 2007*, pages 3–6, New York, August 2007. ACM.

[8] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Commun. ACM*, 18(9):509–517, September 1975.

[9] Mustafa Bilgic. Explaining recommendations: Satisfaction vs. promotion. In *In Proceedings of Beyond Personalization 2005, the Workshop on the Next Stage of Recommender Systems Research(IUI2005*, pages 13–18, 2005.

[10] Jesus Bobadilla, Antonio Hernando, Fernando Ortega, and Jesus Bernal. A framework for collaborative filtering recommender systems. *Expert Syst. Appl.*, 38(12):14609–14623, November 2011.

[11] Amine M. Boumaza and Armelle Brun. Stochastic search for global neighbors selection in collaborative filtering. In Sascha Ossowski and Paola Lecca, editors, *SAC*, pages 232–237. ACM, 2012.

[12] Jeremy Buhler. Provably sensitive indexing strategies for biosequence similarity search. In *Proceedings of the Sixth Annual International Conference on Computational Biology*, RECOMB '02, pages 90–99, New York, NY, USA, 2002. ACM.

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

[14] Yu-Chen Chen, Rong-An Shang, and Chen-Yu Kao. The effects of information overload on consumers' subjective state towards buying decision in the internet shopping environment. *Electron. Commer. Rec. Appl.*, 8(1):48–58, January 2009.

[15] Sang Hyun Choi, Young-Seon Jeong, and M.K. Jeong. A hybrid recommendation method with reduced data for large-scale application. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 40(5):557–566, Sept 2010.

[16] Edith Cohen, Mayur Datar, Shinji Fujiwara, Aristides Gionis, Piotr Indyk, Rajeev Motwani, Jeffrey D. Ullman, and Cheng Yang. Finding interesting associations without support pruning. In *In ICDE*, pages 489–499, 2000.

[17] P. Cremonesi, R. Turrin, E. Lentini, and M. Matteucci. An evaluation methodology for collaborative recommender systems. In *Automated solutions for Cross Media Content*

and Multi-channel Distribution, 2008. AXMEDIS '08. International Conference on, pages 224–231, Nov 2008.

[18] Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the Twentieth Annual Symposium on Computational Geometry*, SCG '04, pages 253–262, New York, NY, USA, 2004. ACM.

[19] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: Simplified data processing on large clusters. *Commun. ACM*, 51(1):107–113, January 2008.

[20] Angela Edmunds and Anne Morris. The problem of information overload in business organisations: A review of the literature. *Int. J. Inf. Manag.*, 20(1):17–28, February 2000.

[21] Rinie Egas, Dionysius P. Huijsmans, Michael S. Lew, and Nicu Sebe. Adapting k-d trees to visual retrieval. In *Proceedings of the Third International Conference on Visual Information and Information Systems*, VISUAL '99, pages 533–540, London, UK, UK, 1999. Springer-Verlag.

[22] Jerome H. Friedman, Jon Louis Bentley, and Raphael Ari Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Trans. Math. Softw.*, 3(3):209–226, September 1977.

[23] Zeno Gantner, Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. Mymedialite: A free recommender system library. In *Proceedings of the Fifth ACM Conference on Recommender Systems*, RecSys '11, pages 305–308, New York, NY, USA, 2011. ACM.

[24] Aristides Gionis, Piotr Indyk, and Rajeev Motwani. Similarity search in high dimensions via hashing. In *Proceedings of the 25th International Conference on Very Large Data Bases*, VLDB '99, pages 518–529, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.

[25] AppTornado GmbH. AppBrain Stats. `http://www.appbrain.com/stats/`, 2010. [Online; acessado em abril de 2014].

[26] Patrick Grother, Gerald T. Candela, and James L. Blue. Fast implementations of nearest neighbor classifiers. *Pattern Recognition*, 30(3):459–465, 1997.

[27] Ido Guy, Naama Zwerdling, David Carmel, Inbal Ronen, Erel Uziel, Sivan Yogev, and Shila Ofek-Koifman. Personalized recommendation of social software items based on social relations. In *Proceedings of the Third ACM Conference on Recommender Systems*, RecSys '09, pages 53–60, New York, NY, USA, 2009. ACM.

[28] Philippe Cudre-Mauroux Karl Aberer Parisa Haghani. Lsh at large – distributed knn search in high dimensions. 2008.

[29] Jonathan L. Herlocker, Joseph A. Konstan, and John Riedl. Explaining collaborative filtering recommendations. In *Proceedings of the 2000 ACM Conference on Computer Supported Cooperative Work*, CSCW '00, pages 241–250, New York, NY, USA, 2000. ACM.

[30] Jonathan L. Herlocker, Joseph A. Konstan, and John Riedl. Explaining collaborative filtering recommendations. In *Proceedings of the 2000 ACM Conference on Computer Supported Cooperative Work*, CSCW '00, pages 241–250, New York, NY, USA, 2000. ACM.

[31] Shang H. Hsu, Ming-Hui Wen, Hsin-Chieh Lin, Chun-Chia Lee, and Chia-Hoang Lee. Aimed: A personalized tv recommendation system. In *Proceedings of the 5th European Conference on Interactive TV: A Shared Experience*, EuroITV'07, pages 166–174, Berlin, Heidelberg, 2007. Springer-Verlag.

[32] Joaquin Huerta, Miguel Chover, Ricardo Quiros, Roberto Vivo, and Jose Ribelles. Binary space partitioning trees: a multiresolution approach. In *IV*, pages 148–155. IEEE Computer Society, 1997.

[33] Carl Hulett, Andy Hall, and Guangzhi Qu. Dynamic selection of k nearest neighbors in instance-based learning. In Chengcui Zhang, James Joshi, Elisa Bertino, and Bhavani M. Thuraisingham, editors, *IRI*, pages 85–92. IEEE, 2012.

[34] Statista Inc. Cumulative number of apps downloaded from the Google Play. `http://www.statista.com/statistics/281106/`

number-of-android-app-downloads-from-google-play/, 2013. [Online; acessado em abril de 2014].

[35] Ruud Janssen and Henk de Poot. Information overload: Why some people seem to suffer more than others. In *Proceedings of the 4th Nordic Conference on Human-computer Interaction: Changing Roles*, NordiCHI '06, pages 397–400, New York, NY, USA, 2006. ACM.

[36] Liangxiao Jiang, Zhihua Cai, Dianhong Wang, and Siwei Jiang. Survey of improving k-nearest-neighbor for classification. In *Proceedings of the Fourth International Conference on Fuzzy Systems and Knowledge Discovery - Volume 01*, FSKD '07, pages 679–683, Washington, DC, USA, 2007. IEEE Computer Society.

[37] Yehuda Koren. The bellkor solution to the netflix grand prize, 2009.

[38] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, August 2009.

[39] Neal Lathia, Stephen Hailes, and Licia Capra. knn cf: A temporal social network. In *Proceedings of the 2008 ACM Conference on Recommender Systems*, RecSys '08, pages 227–234, New York, NY, USA, 2008. ACM.

[40] Pasquale Lops, Marco Gemmis, and Giovanni Semeraro. Content-based recommender systems: State of the art and trends. In Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor, editors, *Recommender Systems Handbook*, pages 73–105. Springer US, 2011.

[41] Helena R. LourenÃ§o, Olivier C. Martin, and Thomas StÃ¼tzle. Iterated local search. In *Handbook of Metaheuristics, volume 57 of International Series in Operations Research and Management Science*, pages 321–353. Kluwer Academic Publishers, 2002.

[42] Hao Ma, Haixuan Yang, Michael R. Lyu, and Irwin King. Sorec: social recommendation using probabilistic matrix factorization. In James G. Shanahan, Sihem Amer-Yahia, Ioana Manolescu, Yi Zhang, David A. Evans, Aleksander Kolcz, Key-Sun Choi, and Abdur Chowdhury, editors, *CIKM*, pages 931–940. ACM, 2008.

[43] Raymond J. Mooney and Loriene Roy. Content-based book recommending using learning for text categorization. In *Proceedings of the Fifth ACM Conference on Digital Libraries*, DL '00, pages 195–204, New York, NY, USA, 2000. ACM.

[44] Fionn Murtagh, Jean-Luc Starck, and Michael W. Berry. Overcoming the curse of dimensionality in clustering by means of the wavelet transform. *The Computer Journal*, 43(2):107–120, 2000.

[45] San Murugesan. Understanding web 2.0. *IT Professional*, 9(4):34–41, 2007.

[46] Mohammad Norouzi, Ali Punjani, and David J. Fleet. Fast search in hamming space with multi-index hashing. In *CVPR*, pages 3108–3115. IEEE, 2012.

[47] Stefanos Ougiaroglou, Alexandros Nanopoulos, Apostolos N. Papadopoulos, Yannis Manolopoulos, and Tatjana Welzer-Druzovec. Adaptive k-nearest-neighbor classification using a dynamic number of nearest neighbors. In *Proceedings of the 11th East European Conference on Advances in Databases and Information Systems*, ADBIS'07, pages 66–82, Berlin, Heidelberg, 2007. Springer-Verlag.

[48] Deuk Hee Park, Hyea Kyeong Kim, Il Young Choi, and Jae Kyeong Kim. A literature review and classification of recommender systems research. *Expert Syst. Appl.*, 39(11):10059–10072, September 2012.

[49] Mike Paterson and F. Frances Yao. Optimal binary space partitions for orthogonal objects. In David S. Johnson, editor, *SODA*, pages 100–106. SIAM, 1990.

[50] Maria Soledad Pera and Yiu-Kai Ng. What to read next?: Making personalized book recommendations for k-12 users. In *Proceedings of the 7th ACM Conference on Recommender Systems*, RecSys '13, pages 113–120, New York, NY, USA, 2013. ACM.

[51] Vladimir Pestov. On the geometry of similarity search: Dimensionality curse and concentration of measure. *Inf. Process. Lett.*, 73(1-2):47–51, 2000.

[52] Edward Raff. Java Statistical Analysis Tool. `https://code.google.com/p/java-statistical-analysis-tool/`, 2013. [Online; acessado em janeiro de 2014].

[53] Investor Relations. Company Profile. `http://ir.netflix.com/`, 2013. [Online; acessado em abril de 2014].

[54] Paul Resnick and Hal R. Varian. Recommender systems. *Commun. ACM*, 40(3):56–58, March 1997.

[55] Sebastian Schelter, Christoph Boden, and Volker Markl. Scalable similarity-based neighborhood methods with mapreduce. In *Proceedings of the Sixth ACM Conference on Recommender Systems*, RecSys '12, pages 163–170, New York, NY, USA, 2012. ACM.

[56] Sanjeev Kumar Sharma and Ugrasen Suman. Comparative study and analysis of web personalization frameworks of recommender systems for e-commerce. In *Proceedings of the CUBE International Information Technology Conference*, CUBE '12, pages 629–634, New York, NY, USA, 2012. ACM.

[57] Xiaoyuan Su and Taghi M. Khoshgoftaar. A survey of collaborative filtering techniques. *Adv. in Artif. Intell.*, 2009:4:2–4:2, January 2009.

[58] N. Suguna and K. Thanushkodi. An improved k-nearest neighbor classification using genetic algorithm. volume 7, pages 1694–0814, Jul 2010.

[59] N. Suguna and K. Thanushkodi. A novel rough set reduct algorithm for medical domain based on bee colony optimization. *CoRR*, abs/1006.4540, 2010.

[60] P. Symeonidis, A. Nanopoulos, and Y. Manolopoulos. Providing justifications in recommender systems. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 38(6):1262–1272, Nov 2008.

[61] Bin Wang, Qing Liao, and Chunhong Zhang. Weight based knn recommender system. In *Proceedings of the 2013 5th International Conference on Intelligent Human-Machine Systems and Cybernetics - Volume 02*, IHMSC '13, pages 449–452, Washington, DC, USA, 2013. IEEE Computer Society.

[62] David F. Williamson, Robert A. Parker, and Juliette S. Kendrick. The box plot: A simple visual method to interpret data. *Annals of Internal Medicine*, 110(11):916–921, 1989.

[63] Diyi Yang, Tianqi Chen, Weinan Zhang, Qiuxia Lu, and Yong Yu. Local implicit feedback mining for music recommendation. In *Proceedings of the Sixth ACM Conference on Recommender Systems*, RecSys '12, pages 91–98, New York, NY, USA, 2012. ACM.

[64] Chi Zhang, Feifei Li, and Jeffrey Jestes. Efficient parallel knn joins for large data in mapreduce. In *Proceedings of the 15th International Conference on Extending Database Technology*, EDBT '12, pages 38–49, New York, NY, USA, 2012. ACM.

Apêndice A

Resultados do Experimento

Resultados da solução proposta com k = 30						
	MovieLens 100K			MovieLens 1M		
#	Preparação (segundos)	Realização (segundos)	MAE	Preparação (segundos)	Realização (segundos)	MAE
1	0.50	2.12	0.6978	38.65	65.66	0.6559
2	0.69	2.04	0.6909	36.88	67.33	0.6555
3	0.39	1.93	0.6937	35.93	66.02	0.6482
4	0.69	1.90	0.6911	36.87	64.70	0.6555
5	0.69	1.90	0.6989	39.33	65.94	0.6554
6	0.53	1.90	0.6977	36.82	66.60	0.6584
7	0.38	1.95	0.6930	36.56	65.05	0.6555
8	0.44	1.95	0.6853	35.72	63.56	0.6546
9	0.48	1.95	0.6898	35.14	65.42	0.6552
10	0.39	1.92	0.6991	36.29	65.41	0.6524
11	0.44	1.93	0.6969	36.74	65.08	0.6553
12	0.67	1.90	0.7008	35.77	64.28	0.6554
13	0.59	1.98	0.6965	36.32	66.77	0.6530
14	0.50	1.94	0.6988	37.80	65.09	0.6566
15	0.39	1.93	0.6861	36.36	65.60	0.6553
16	0.37	1.92	0.6800	37.85	65.53	0.6534
17	0.37	1.90	0.6871	35.89	65.74	0.6562
18	0.67	1.93	0.6970	37.07	67.09	0.6560
19	0.58	1.97	0.6955	36.00	63.99	0.6506
20	0.61	1.93	0.6916	35.53	63.91	0.6569
21	0.58	1.92	0.6922	35.87	64.50	0.6560
22	0.45	1.95	0.6924	36.18	64.41	0.6537
23	0.58	1.90	0.7004	36.07	64.23	0.6535
24	0.58	1.94	0.6857	35.58	63.43	0.6562
25	0.39	1.92	0.6994	35.60	63.70	0.6546
26	0.56	1.89	0.6903	36.20	64.43	0.6518
27	0.39	1.89	0.7007	35.79	64.26	0.6537
28	0.56	1.92	0.6940	35.58	64.07	0.6538
29	0.55	1.93	0.6904	36.07	64.63	0.6535
30	0.58	1.93	0.6970	35.74	64.33	0.6575
31	0.52	1.93	0.6939	35.93	64.20	0.6543
32	0.56	1.95	0.7018	35.63	64.20	0.6543
33	0.53	1.94	0.6961	35.83	64.48	0.6565
34	0.41	1.92	0.6991	35.63	64.22	0.6553
35	0.55	1.92	0.6992	35.97	64.21	0.6517
36	0.61	1.93	0.6888	35.97	64.24	0.6553
37	0.56	1.92	0.6895	36.11	64.40	0.6528
38	0.45	1.95	0.6871	35.58	64.24	0.6560
39	0.61	1.95	0.6888	35.62	64.14	0.6528
40	0.55	1.92	0.6899	35.87	64.40	0.6560
41	0.56	1.93	0.6995	35.49	63.93	0.6563
42	0.53	1.92	0.6880	35.51	64.10	0.6533
43	0.53	1.94	0.6957	35.39	63.95	0.6533
44	0.52	1.92	0.7028	36.09	64.76	0.6542
45	0.56	1.92	0.6914	35.60	64.30	0.6534
46	0.50	1.98	0.6895	35.66	64.55	0.6564
47	0.56	1.95	0.6954	36.00	64.32	0.6544
48	0.55	1.90	0.6882	35.72	64.04	0.6571
49	0.51	1.95	0.6965	35.77	64.41	0.6531
50	0.47	1.98	0.6952	35.96	64.77	0.6565

Resultados do kNN tradicional com k = 30						
	MovieLens 100K			MovieLens 1M		
#	Preparação (segundos)	Realização (segundos)	MAE	Preparação (segundos)	Realização (segundos)	MAE
1	1.48	3.34	0.6851	96.65	153.34	0.6505
2	1.33	3.06	0.6802	92.66	156.46	0.6500
3	1.53	3.03	0.6783	93.38	155.73	0.6438
4	1.42	3.04	0.6804	93.59	152.18	0.6511
5	1.28	3.03	0.6816	92.88	155.57	0.6504
6	1.26	3.04	0.6780	93.50	152.35	0.6534
7	1.31	3.01	0.6850	93.36	157.67	0.6519
8	1.26	3.04	0.6711	94.33	153.68	0.6528
9	1.31	3.03	0.6855	91.25	149.12	0.6509
10	1.31	3.03	0.6885	91.29	152.72	0.6470
11	1.26	3.04	0.6874	90.03	151.94	0.6492
12	1.28	3.04	0.6918	91.35	146.39	0.6514
13	1.26	3.07	0.6819	90.54	150.63	0.6476
14	1.25	3.01	0.6887	91.60	148.28	0.6519
15	1.26	3.07	0.6764	91.15	145.58	0.6509
16	1.30	3.07	0.6747	90.51	146.20	0.6491
17	1.26	3.03	0.6757	90.59	150.96	0.6517
18	1.26	3.04	0.6883	90.45	146.78	0.6514
19	1.28	3.04	0.6849	91.03	148.92	0.6463
20	1.28	3.04	0.6826	90.43	148.70	0.6513
21	1.36	3.10	0.6839	91.18	151.04	0.6507
22	1.25	3.03	0.6847	91.29	148.97	0.6486
23	1.45	3.01	0.6889	90.84	144.69	0.6505
24	1.33	3.04	0.6738	90.99	145.07	0.6521
25	1.26	3.04	0.6838	91.03	147.02	0.6503
26	1.31	3.03	0.6815	90.62	152.82	0.6477
27	1.26	3.01	0.6848	89.87	149.37	0.6506
28	1.26	3.03	0.6836	91.17	149.28	0.6479
29	1.37	3.09	0.6796	91.34	151.59	0.6494
30	1.26	3.04	0.6863	91.15	150.51	0.6534
31	1.25	3.04	0.6831	90.45	151.23	0.6499
32	1.26	3.07	0.6967	91.73	150.70	0.6486
33	1.31	3.07	0.6915	90.40	152.38	0.6513
34	1.26	3.04	0.6811	90.51	150.91	0.6516
35	1.28	3.03	0.6840	90.68	151.41	0.6473
36	1.28	3.07	0.6852	90.31	148.54	0.6503
37	1.31	3.07	0.6788	90.98	152.41	0.6481
38	1.26	3.04	0.6741	90.45	150.60	0.6526
39	1.26	3.03	0.6736	90.98	151.24	0.6477
40	1.40	3.04	0.6788	91.29	150.62	0.6493
41	1.28	3.06	0.6827	91.48	150.81	0.6520
42	1.28	3.04	0.6754	90.15	152.52	0.6484
43	1.26	3.06	0.6789	90.65	150.76	0.6485
44	1.30	3.11	0.6965	91.03	149.36	0.6503
45	1.34	3.04	0.6825	91.10	152.29	0.6484
46	1.28	3.04	0.6883	91.14	150.23	0.6507
47	1.26	3.07	0.6801	91.01	150.96	0.6503
48	1.26	3.07	0.6761	91.54	151.52	0.6526
49	1.26	3.06	0.6834	91.21	153.01	0.6483
50	1.29	3.04	0.6857	91.54	153.24	0.6512

Resultados do kd-tree com k = 30						
	MovieLens 100K			MovieLens 1M		
#	Preparação (segundos)	Realização (segundos)	MAE	Preparação (segundos)	Realização (segundos)	MAE
1	0.67	22.51	0.8253	10.42	1563.51	0.7866
2	0.70	21.64	0.8375	9.69	1559.71	0.7895
3	0.62	21.70	0.8322	9.55	1566.71	0.7833
4	0.64	21.62	0.8345	9.17	1564.98	0.7914
5	0.66	21.43	0.8461	9.24	1568.54	0.7894
6	0.55	21.61	0.8357	9.53	1558.38	0.7934
7	0.64	21.28	0.8349	9.27	1570.86	0.7876
8	0.66	21.81	0.8181	9.22	1571.31	0.7867
9	0.62	21.75	0.8395	9.17	1569.75	0.7897
10	0.69	21.72	0.8410	9.25	1571.57	0.7865
11	0.64	21.64	0.8297	9.25	1565.10	0.7896
12	0.64	21.68	0.8330	9.31	1567.44	0.7907
13	0.62	21.68	0.8275	9.25	1566.56	0.7861
14	0.69	21.64	0.8391	9.25	1569.68	0.7851
15	0.64	21.65	0.8351	9.25	1569.10	0.7861
16	0.64	21.58	0.8243	9.25	1572.17	0.7865
17	0.64	21.59	0.8330	9.45	1561.56	0.7866
18	0.67	21.67	0.8348	9.20	1569.21	0.7892
19	0.64	21.72	0.8364	9.14	1567.04	0.7853
20	0.62	21.47	0.8213	9.20	1567.09	0.7880
21	0.51	21.58	0.8311	9.17	1557.01	0.7862
22	0.64	21.68	0.8358	9.42	1556.23	0.7878
23	0.66	21.54	0.8374	9.17	1567.38	0.7924
24	0.62	21.75	0.8285	9.16	1566.99	0.7873
25	0.50	21.56	0.8439	9.19	1567.60	0.7897
26	0.55	21.58	0.8441	9.19	1570.44	0.7847
27	0.64	21.54	0.8397	9.19	1571.22	0.7879
28	0.53	21.61	0.8340	9.10	1564.93	0.7831
29	0.64	21.79	0.8389	9.10	1567.26	0.7894
30	0.50	21.82	0.8451	9.13	1569.46	0.7901
31	0.64	21.67	0.8319	9.08	1566.93	0.7882
32	0.64	21.51	0.8424	9.11	1566.70	0.7847
33	0.67	21.56	0.8376	9.16	1568.79	0.7926
34	0.51	21.58	0.8294	9.36	1557.65	0.7837
35	0.64	21.48	0.8505	9.16	1567.79	0.7873
36	0.53	21.65	0.8315	9.11	1563.68	0.7866
37	0.64	21.59	0.8210	9.13	1563.39	0.7845
38	0.67	21.56	0.8174	9.16	1566.80	0.7886
39	0.50	21.48	0.8300	9.16	1568.77	0.7871
40	0.64	21.54	0.8222	9.14	1570.31	0.7894
41	0.64	21.59	0.8281	9.14	1568.40	0.7901
42	0.50	21.72	0.8394	9.16	1567.02	0.7863
43	0.67	21.61	0.8254	9.17	1564.75	0.7828
44	0.64	21.58	0.8323	9.16	1561.58	0.7862
45	0.50	21.56	0.8390	9.16	1566.91	0.7851
46	0.64	21.51	0.8351	9.16	1572.41	0.7877
47	0.64	21.68	0.8221	9.19	1562.31	0.7899
48	0.53	21.58	0.8283	9.17	1570.92	0.7872
49	0.62	21.65	0.8284	9.16	1573.12	0.7853
50	0.64	21.72	0.8383	9.17	1571.78	0.7886

Resultados do LSH com k = 30						
	MovieLens 100K			MovieLens 1M		
#	Preparação (segundos)	Realização (segundos)	MAE	Preparação (segundos)	Realização (segundos)	MAE
1	0.13	11.77	0.7527	0.53	456.56	0.7025
2	0.17	11.73	0.7523	0.34	510.59	0.7010
3	0.04	11.71	0.7562	0.36	508.80	0.6966
4	0.04	10.97	0.7510	0.33	485.65	0.7029
5	0.17	12.71	0.7522	0.33	448.45	0.7039
6	0.17	12.22	0.7528	0.33	552.49	0.7054
7	0.17	13.61	0.7602	0.33	504.97	0.7032
8	0.03	10.00	0.7488	0.33	408.47	0.7006
9	0.17	11.71	0.7535	0.33	380.39	0.7017
10	0.04	12.40	0.7609	0.33	417.55	0.6982
11	0.07	12.40	0.7559	0.33	413.14	0.7003
12	0.17	10.46	0.7687	0.33	389.74	0.7036
13	0.17	11.73	0.7495	0.33	379.17	0.6992
14	0.04	11.75	0.7615	0.33	386.29	0.7013
15	0.18	11.18	0.7490	0.33	486.11	0.7031
16	0.03	12.36	0.7335	0.33	395.20	0.6998
17	0.17	12.46	0.7443	0.33	503.41	0.7028
18	0.17	8.62	0.7638	0.33	449.80	0.7024
19	0.13	11.94	0.7581	0.33	487.31	0.6998
20	0.17	8.32	0.7562	0.33	445.46	0.7027
21	0.17	12.97	0.7455	0.33	460.31	0.7012
22	0.13	12.75	0.7535	0.33	493.02	0.7011
23	0.03	11.70	0.7631	0.33	484.94	0.7050
24	0.05	11.41	0.7379	0.33	431.15	0.7003
25	0.04	9.89	0.7504	0.33	469.62	0.7026
26	0.04	8.25	0.7530	0.33	489.55	0.7007
27	0.03	11.03	0.7645	0.33	428.77	0.6997
28	0.04	11.54	0.7405	0.33	495.36	0.6984
29	0.04	11.11	0.7573	0.33	408.33	0.7010
30	0.03	11.20	0.7522	0.33	463.74	0.7041
31	0.03	12.76	0.7441	0.33	440.83	0.7030
32	0.04	12.63	0.7607	0.33	459.76	0.6982
33	0.03	10.38	0.7655	0.33	452.01	0.7034
34	0.03	12.51	0.7520	0.33	442.84	0.7026
35	0.04	10.70	0.7589	0.33	434.93	0.6975
36	0.03	11.52	0.7561	0.33	504.77	0.7030
37	0.04	12.51	0.7522	0.33	355.18	0.6995
38	0.03	11.65	0.7464	0.33	412.09	0.7021
39	0.04	11.22	0.7532	0.34	432.45	0.7007
40	0.04	11.56	0.7430	0.33	464.04	0.7012
41	0.04	11.59	0.7489	0.33	536.59	0.7030
42	0.06	11.72	0.7553	0.34	403.45	0.7010
43	0.03	13.65	0.7506	0.33	537.59	0.7014
44	0.04	12.19	0.7556	0.33	439.34	0.7027
45	0.11	11.29	0.7459	0.34	455.91	0.7009
46	0.05	11.11	0.7574	0.34	505.43	0.7030
47	0.04	12.01	0.7448	0.33	402.31	0.7005
48	0.04	13.04	0.7576	0.34	518.53	0.7026
49	0.04	13.04	0.7468	0.33	426.29	0.7002
50	0.04	11.22	0.7470	0.33	446.32	0.7022

Resultados do ILS kNN com k = 30						
	MovieLens 100K			MovieLens 1M		
#	Preparação (segundos)	Realização (segundos)	MAE	Preparação (segundos)	Realização (segundos)	MAE
1	1235.34	1.78	0.7125	2376.16	69.44	0.6592
2	1235.79	1.72	0.7097	1375.33	67.00	0.6594
3	1238.07	1.67	0.7107	16869.84	67.88	0.6520
4	1254.94	1.69	0.7053	1291.95	65.02	0.6609
5	1234.90	1.66	0.7131	4293.68	65.24	0.6603
6	1226.09	1.65	0.7058	1391.29	67.21	0.6640
7	1223.37	1.70	0.7062	1284.88	65.27	0.6605
8	1246.94	1.70	0.6958	1373.85	67.07	0.6602
9	1227.30	1.72	0.7066	4707.59	66.94	0.6592
10	1231.97	1.70	0.7102	1396.08	67.55	0.6578
11	1227.04	1.72	0.7073	1371.79	66.82	0.6595
12	1266.36	1.72	0.7123	3536.35	64.80	0.6610
13	1220.91	1.69	0.7136	3839.49	66.66	0.6584
14	1245.43	1.67	0.7133	1290.01	65.41	0.6624
15	1217.58	1.73	0.7036	3789.28	66.93	0.6585
16	1222.92	1.70	0.6992	34594.80	68.77	0.6576
17	1221.90	1.67	0.7021	3129.80	67.74	0.6597
18	1220.23	1.72	0.7134	4610.42	66.83	0.6599
19	1227.10	1.70	0.7122	1391.68	67.30	0.6555
20	1273.37	1.70	0.7114	1388.59	66.89	0.6602
21	1215.54	1.70	0.7058	2050.45	65.30	0.6602
22	1225.40	1.69	0.7094	2851.48	64.91	0.6588
23	1229.69	1.67	0.7190	5031.88	65.30	0.6600
24	1232.03	1.65	0.6988	34773.93	68.49	0.6591
25	1221.75	1.70	0.7078	1400.70	67.53	0.6598
26	1216.82	1.69	0.7003	1283.12	64.85	0.6573
27	1210.73	1.65	0.7151	2248.92	67.13	0.6583
28	1207.47	1.67	0.7075	2935.58	65.71	0.6577
29	1232.93	1.70	0.7052	2290.05	67.74	0.6575
30	1252.32	1.67	0.7063	1367.51	66.85	0.6623
31	1224.81	1.70	0.7060	5398.53	66.57	0.6595
32	1250.51	1.69	0.7237	3012.49	66.94	0.6583
33	1215.90	1.67	0.7147	3620.70	65.69	0.6608
34	1247.63	1.67	0.7073	1394.38	67.19	0.6591
35	1263.84	1.67	0.7167	31185.59	68.25	0.6547
36	1231.54	1.72	0.7097	33775.88	66.66	0.6598
37	1211.51	1.73	0.7050	34855.11	70.28	0.6578
38	1219.88	1.72	0.6979	3192.62	70.76	0.6607
39	1223.20	1.67	0.7009	2327.93	70.49	0.6582
40	1222.26	1.67	0.7027	1480.09	69.45	0.6611
41	1221.33	1.72	0.7156	1460.66	70.76	0.6613
42	1220.20	1.69	0.7022	37154.90	70.70	0.6564
43	1243.03	1.70	0.7059	1417.14	69.15	0.6573
44	1243.01	1.67	0.7102	37038.55	71.16	0.6571
45	1220.20	1.72	0.7045	1429.75	69.71	0.6556
46	1208.53	1.69	0.7154	3188.60	69.71	0.6598
47	1262.40	1.66	0.7059	32009.89	67.39	0.6602
48	1238.44	1.66	0.7070	1343.86	66.74	0.6633
49	1240.58	1.72	0.7122	1390.09	67.53	0.6588
50	1222.45	1.66	0.7137	3098.79	67.60	0.6616

Apêndice B

Avaliação das Heurísticas

Avaliação da Heurísticas - MovieLens 100K							
Usuários mais Ativos				Média de Similaridade			
#	Tempo (segundos)	MAE		#	Tempo (segundos)	MAE	
1	0.013	0.6978		1	1.529	0.7096	
2	0.000	0.6909		2	1.419	0.7042	
3	0.000	0.6937		3	1.911	0.7024	
4	0.000	0.6911		4	1.754	0.7022	
5	0.000	0.6989		5	1.405	0.7154	
6	0.000	0.6977		6	1.666	0.7096	
7	0.000	0.6930		7	1.651	0.7042	
8	0.000	0.6853		8	1.652	0.6911	
9	0.000	0.6898		9	1.577	0.7069	
10	0.000	0.6991		10	1.780	0.7126	
11	0.001	0.6969		11	1.795	0.7056	
12	0.000	0.7008		12	1.499	0.7123	
13	0.000	0.6965		13	1.493	0.7086	
14	0.000	0.6988		14	1.982	0.7036	
15	0.001	0.6861		15	1.684	0.6969	
16	0.000	0.6800		16	1.386	0.6977	
17	0.001	0.6871		17	1.902	0.6971	
18	0.000	0.6970		18	1.519	0.7098	
19	0.001	0.6955		19	1.747	0.7060	
20	0.001	0.6916		20	1.807	0.7050	
21	0.000	0.6922		21	1.740	0.7022	
22	0.001	0.6924		22	1.396	0.7070	
23	0.000	0.7004		23	1.572	0.7103	
24	0.000	0.6857		24	1.741	0.6941	
25	0.001	0.6994		25	1.607	0.7108	
26	0.000	0.6903		26	1.882	0.6991	
27	0.000	0.7007		27	1.403	0.7125	
28	0.000	0.6940		28	1.692	0.7093	
29	0.000	0.6904		29	1.543	0.7019	
30	0.001	0.6970		30	1.683	0.7028	
31	0.001	0.6939		31	1.603	0.7011	
32	0.000	0.7018		32	1.611	0.7199	
33	0.000	0.6961		33	1.607	0.7093	
34	0.000	0.6991		34	1.413	0.7113	
35	0.000	0.6992		35	1.627	0.7121	
36	0.001	0.6888		36	1.547	0.7075	
37	0.000	0.6895		37	1.575	0.7026	
38	0.000	0.6871		38	1.392	0.6956	
39	0.000	0.6888		39	1.389	0.6971	
40	0.000	0.6899		40	1.511	0.7018	
41	0.000	0.6995		41	1.401	0.7085	
42	0.000	0.6880		42	1.638	0.6981	
43	0.001	0.6957		43	1.659	0.7066	
44	0.001	0.7028		44	1.674	0.7126	
45	0.001	0.6914		45	1.440	0.7034	
46	0.000	0.6895		46	1.674	0.7120	
47	0.001	0.6954		47	1.489	0.7008	
48	0.000	0.6882		48	1.661	0.7003	
49	0.001	0.6965		49	1.434	0.7048	
50	0.000	0.6952		50	1.437	0.7094	
Média	**0.001**	**0.6937**		Média	**1.604**	**0.7053**	

Avaliação da Heurísticas - MovieLens 100K							
Número de Conexões				Vizinhos mais Recorrentes			
#	Tempo (segundos)	MAE		#	Tempo (segundos)	MAE	
1	1.487	0.7066		1	1.663	0.7004	
2	1.410	0.7067		2	1.770	0.6986	
3	1.878	0.7039		3	1.790	0.6966	
4	1.744	0.7011		4	1.479	0.6947	
5	1.554	0.7118		5	1.722	0.7125	
6	1.495	0.7019		6	1.713	0.7007	
7	1.516	0.7021		7	1.758	0.6960	
8	1.490	0.6923		8	1.499	0.6908	
9	1.647	0.7045		9	1.522	0.6998	
10	1.587	0.7097		10	1.766	0.7026	
11	1.475	0.7022		11	1.639	0.7061	
12	1.716	0.7136		12	1.646	0.7076	
13	1.492	0.7110		13	1.692	0.7046	
14	1.836	0.7058		14	1.806	0.7003	
15	1.459	0.6959		15	1.789	0.6920	
16	1.759	0.6946		16	1.494	0.6903	
17	1.870	0.6966		17	1.474	0.6905	
18	1.473	0.7106		18	1.474	0.7087	
19	1.702	0.7072		19	1.508	0.6987	
20	1.804	0.7047		20	1.704	0.6963	
21	1.520	0.7080		21	1.777	0.6938	
22	1.637	0.7079		22	1.718	0.7046	
23	1.477	0.7098		23	1.475	0.7018	
24	1.790	0.6950		24	1.513	0.6917	
25	1.464	0.7132		25	1.667	0.7032	
26	1.688	0.6999		26	1.762	0.6928	
27	1.486	0.7146		27	1.817	0.7046	
28	1.550	0.7078		28	1.784	0.6967	
29	1.456	0.7013		29	1.504	0.6945	
30	1.481	0.7060		30	1.758	0.7019	
31	1.480	0.7043		31	1.739	0.6928	
32	1.487	0.7172		32	1.478	0.7149	
33	1.481	0.7079		33	1.506	0.7059	
34	1.458	0.7027		34	1.475	0.7009	
35	1.516	0.7111		35	1.796	0.7063	
36	1.479	0.7054		36	1.501	0.7009	
37	1.620	0.7034		37	1.709	0.6954	
38	1.505	0.6936		38	1.656	0.6908	
39	1.767	0.6958		39	1.479	0.6948	
40	1.780	0.6989		40	1.522	0.6962	
41	1.531	0.7079		41	1.730	0.7027	
42	1.529	0.6970		42	1.696	0.6907	
43	1.711	0.7021		43	1.533	0.7005	
44	1.681	0.7096		44	1.484	0.7036	
45	1.541	0.7006		45	1.483	0.6974	
46	1.738	0.7127		46	1.648	0.7066	
47	1.483	0.7023		47	1.782	0.6937	
48	1.682	0.6944		48	1.681	0.6933	
49	1.533	0.7081		49	1.478	0.7019	
50	1.613	0.7069		50	1.485	0.7052	
Média	**1.591**	**0.7046**		Média	**1.631**	**0.6994**	

Avaliação da Heurísticas - MovieLens 100K							
Itens distintos							
#	Tempo (segundos)	MAE					
1	0.086	0.7027					
2	0.069	0.6901					
3	0.093	0.6925					
4	0.071	0.6924					
5	0.070	0.6994					
6	0.070	0.6972					
7	0.069	0.6935					
8	0.065	0.6885					
9	0.075	0.6923					
10	0.086	0.7009					
11	0.083	0.7009					
12	0.076	0.7017					
13	0.061	0.7000					
14	0.079	0.7049					
15	0.072	0.6917					
16	0.076	0.6790					
17	0.080	0.6882					
18	0.076	0.6991					
19	0.060	0.6949					
20	0.077	0.6992					
21	0.059	0.6901					
22	0.069	0.6952					
23	0.077	0.6991					
24	0.076	0.6886					
25	0.064	0.6955					
26	0.075	0.6917					
27	0.059	0.7000					
28	0.078	0.6928					
29	0.076	0.6924					
30	0.076	0.7001					
31	0.074	0.6927					
32	0.082	0.7052					
33	0.059	0.6995					
34	0.059	0.6985					
35	0.061	0.6980					
36	0.068	0.6950					
37	0.075	0.6916					
38	0.076	0.6886					
39	0.076	0.6908					
40	0.070	0.6928					
41	0.076	0.7013					
42	0.075	0.6922					
43	0.075	0.6945					
44	0.059	0.7063					
45	0.068	0.6933					
46	0.075	0.6940					
47	0.074	0.6968					
48	0.059	0.6881					
49	0.075	0.6956					
50	0.075	0.6985					
Média	**0.072**	**0.6954**					

Avaliação da Heurísticas - MovieLens 1M							
Usuários mais Ativos				Média de Similaridade			
#	Tempo (segundos)	MAE		#	Tempo (segundos)	MAE	
1	0.020	0.6559		1	94.367	0.6593	
2	0.001	0.6555		2	93.178	0.6586	
3	0.002	0.6482		3	92.976	0.6524	
4	0.002	0.6555		4	93.023	0.6591	
5	0.001	0.6554		5	92.851	0.6593	
6	0.001	0.6584		6	92.649	0.6618	
7	0.001	0.6555		7	96.174	0.6606	
8	0.001	0.6546		8	96.236	0.6595	
9	0.001	0.6552		9	92.726	0.6597	
10	0.002	0.6524		10	91.759	0.6567	
11	0.001	0.6553		11	92.103	0.6594	
12	0.001	0.6554		12	92.508	0.6601	
13	0.002	0.6530		13	92.196	0.6571	
14	0.001	0.6566		14	92.305	0.6613	
15	0.002	0.6553		15	92.476	0.6595	
16	0.001	0.6534		16	92.212	0.6578	
17	0.002	0.6562		17	92.555	0.6608	
18	0.001	0.6560		18	91.822	0.6594	
19	0.001	0.6506		19	91.853	0.6554	
20	0.002	0.6569		20	92.290	0.6600	
21	0.001	0.6560		21	92.258	0.6604	
22	0.001	0.6537		22	92.164	0.6576	
23	0.002	0.6535		23	92.118	0.6589	
24	0.002	0.6562		24	92.430	0.6600	
25	0.002	0.6546		25	92.836	0.6596	
26	0.001	0.6518		26	92.353	0.6557	
27	0.001	0.6537		27	92.477	0.6588	
28	0.001	0.6538		28	92.134	0.6568	
29	0.002	0.6535		29	91.978	0.6576	
30	0.001	0.6575		30	92.353	0.6623	
31	0.001	0.6543		31	92.524	0.6585	
32	0.001	0.6543		32	92.696	0.6577	
33	0.001	0.6565		33	92.633	0.6593	
34	0.001	0.6553		34	92.742	0.6597	
35	0.001	0.6517		35	92.773	0.6562	
36	0.001	0.6553		36	92.383	0.6606	
37	0.001	0.6528		37	92.789	0.6567	
38	0.001	0.6560		38	92.181	0.6610	
39	0.001	0.6528		39	91.993	0.6574	
40	0.001	0.6560		40	92.430	0.6597	
41	0.001	0.6563		41	92.337	0.6608	
42	0.002	0.6533		42	92.711	0.6567	
43	0.001	0.6533		43	92.367	0.6567	
44	0.002	0.6542		44	92.976	0.6575	
45	0.001	0.6534		45	93.132	0.6563	
46	0.001	0.6564		46	92.867	0.6599	
47	0.002	0.6544		47	92.181	0.6597	
48	0.001	0.6571		48	92.758	0.6622	
49	0.001	0.6531		49	92.618	0.6582	
50	0.001	0.6565		50	95.191	0.6601	
Média	**0.002**	**0.6546**		Média	**92.713**	**0.6588**	

Avaliação da Heurísticas - MovieLens 1M							
Número de Conexões				Vizinhos mais Recorrentes			
#	Tempo (segundos)	MAE		#	Tempo (segundos)	MAE	
1	94.115	0.6595		1	101.142	0.6562	
2	92.961	0.6576		2	101.091	0.6556	
3	93.319	0.6517		3	101.666	0.6483	
4	93.039	0.6588		4	100.561	0.6560	
5	93.257	0.6587		5	100.474	0.6556	
6	92.586	0.6616		6	100.690	0.6589	
7	92.695	0.6597		7	100.641	0.6567	
8	92.383	0.6594		8	101.667	0.6564	
9	92.882	0.6589		9	101.758	0.6561	
10	92.727	0.6558		10	99.590	0.6533	
11	92.602	0.6580		11	101.602	0.6552	
12	92.212	0.6587		12	100.348	0.6565	
13	94.879	0.6557		13	99.499	0.6542	
14	96.861	0.6609		14	99.439	0.6579	
15	93.928	0.6583		15	100.258	0.6564	
16	92.103	0.6566		16	100.394	0.6537	
17	92.133	0.6585		17	100.771	0.6573	
18	91.806	0.6594		18	100.485	0.6561	
19	92.150	0.6535		19	100.097	0.6514	
20	91.759	0.6592		20	111.618	0.6564	
21	91.993	0.6591		21	99.239	0.6564	
22	92.228	0.6571		22	100.261	0.6541	
23	92.711	0.6578		23	100.153	0.6554	
24	92.227	0.6590		24	100.354	0.6569	
25	92.321	0.6588		25	99.389	0.6558	
26	92.586	0.6542		26	99.738	0.6525	
27	92.305	0.6576		27	126.641	0.6546	
28	92.665	0.6569		28	101.903	0.6537	
29	92.866	0.6559		29	100.392	0.6545	
30	92.680	0.6609		30	101.783	0.6593	
31	92.805	0.6584		31	102.398	0.6554	
32	92.367	0.6571		32	102.279	0.6529	
33	92.243	0.6595		33	102.619	0.6561	
34	92.446	0.6595		34	97.080	0.6565	
35	92.290	0.6556		35	97.230	0.6527	
36	92.321	0.6599		36	96.880	0.6573	
37	92.196	0.6550		37	96.820	0.6538	
38	92.258	0.6591		38	97.070	0.6577	
39	92.742	0.6567		39	97.343	0.6529	
40	92.789	0.6589		40	102.640	0.6555	
41	92.228	0.6599		41	102.300	0.6566	
42	92.602	0.6563		42	96.780	0.6539	
43	92.305	0.6555		43	96.832	0.6537	
44	92.289	0.6564		44	97.230	0.6537	
45	92.414	0.6548		45	96.770	0.6522	
46	92.181	0.6590		46	96.750	0.6557	
47	92.196	0.6588		47	96.712	0.6560	
48	92.836	0.6618		48	96.791	0.6585	
49	92.056	0.6577		49	96.670	0.6542	
50	92.071	0.6594		50	96.650	0.6563	
Média	**92.652**	**0.6579**		**Média**	**100.390**	**0.6553**	

Avaliação da Heurísticas - MovieLens 1M							
Itens distintos							
#	Tempo (segundos)	MAE					
1	2.254	0.6554					
2	1.118	0.6553					
3	1.148	0.6490					
4	1.111	0.6555					
5	1.098	0.6547					
6	1.099	0.6581					
7	1.125	0.6558					
8	1.088	0.6559					
9	1.102	0.6554					
10	1.093	0.6529					
11	1.095	0.6551					
12	1.090	0.6557					
13	1.070	0.6532					
14	1.060	0.6569					
15	1.070	0.6553					
16	1.070	0.6539					
17	1.070	0.6565					
18	1.070	0.6563					
19	1.050	0.6518					
20	1.060	0.6564					
21	1.070	0.6559					
22	1.070	0.6538					
23	1.060	0.6539					
24	1.060	0.6565					
25	1.040	0.6553					
26	1.050	0.6521					
27	1.050	0.6546					
28	1.100	0.6542					
29	1.090	0.6545					
30	1.100	0.6579					
31	1.110	0.6548					
32	1.090	0.6543					
33	1.060	0.6554					
34	1.050	0.6560					
35	1.060	0.6523					
36	1.050	0.6564					
37	1.060	0.6529					
38	1.060	0.6566					
39	1.060	0.6526					
40	1.050	0.6561					
41	1.060	0.6567					
42	1.060	0.6537					
43	1.060	0.6529					
44	1.060	0.6545					
45	1.060	0.6538					
46	1.070	0.6564					
47	1.060	0.6545					
48	1.060	0.6573					
49	1.050	0.6533					
50	1.050	0.6560					
Média	**1.097**	**0.6549**					