

Bruno de Figueiredo Melo e Souza

**Modelos de fatoração matricial para  
recomendação de vídeos**

**DISSERTAÇÃO DE MESTRADO**

DEPARTAMENTO DE INFORMÁTICA  
Programa de Pós-graduação em Informática

Rio de Janeiro  
Agosto 2011

**Bruno de Figueiredo Melo e Souza**

**Modelos de fatoração matricial para  
recomendação de vídeos**

**DISSERTAÇÃO DE MESTRADO**

Dissertação apresentada como requisito parcial para obtenção do grau de Mestre pelo Programa de Pós-graduação em Informática do Departamento de Informática da PUC-Rio

Orientador: Prof. Ruy Luiz Milidiú

Rio de Janeiro  
Agosto 2011



**Bruno de Figueiredo Melo e Souza**

## **Modelos de fatoração matricial para recomendação de vídeos**

Dissertação apresentada como requisito parcial para obtenção do grau de Mestre pelo Programa de Pós-graduação em Informática do Departamento de Informática da PUC-Rio

**Prof. Ruy Luiz Milidiú**  
Orientador  
Departamento de Informática – PUC-Rio

**Prof. Marco A. Casanova**  
Departamento de Informática – PUC-Rio

**Prof. Daniel Schwabe**  
Departamento de Informática – PUC-Rio

**Prof. José Engenio Leal**  
Coordenador Setorial do Centro Técnico Científico  
PUC-Rio

Rio de Janeiro, \_\_\_\_\_ / \_\_\_\_\_ / \_\_\_\_\_

Todos os direitos reservados. É proibida a reprodução total ou parcial do trabalho sem autorização da universidade, do autor e do orientador.

**Bruno de Figueiredo Melo e Souza**

Graduou-se em Engenharia Elétrica com ênfase em Eletrônica e Computação na UFRJ – Universidade Federal do Rio de Janeiro – em 2006. Atualmente trabalha como Coordenador de Tecnologia na Globo.com liderando projetos de infra-estrutura para codificação e distribuição de vídeos para internet.

Ficha Catalográfica

Souza, Bruno de Figueiredo Melo e

Modelos de fatoração matricial para recomendação de vídeos /  
Bruno de Figueiredo Melo e Souza ; orientador: Ruy Luiz Milidiú. –  
Rio de Janeiro : PUC-Rio, Departamento de Informática, 2011.

[], 66 f. : il. ; 30 cm

Dissertação (mestrado)–Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática.

Inclui referências bibliográficas.

CDD: 004

## **Agradecimentos**

À Globo.com que financiou por completo o curso de mestrado.

À minha família, pelo apoio incondicional a todas as minhas escolhas de vida.

Ao meu orientador Ruy Luiz Milidiú, por me guiar nas pesquisas e decisões ao longo do curso.

## **Resumo**

Souza, Bruno; Milidiú, Ruy Luiz . **Modelos de Fatoração Matricial para Recomendação de Vídeos.** Rio de Janeiro, 2011. 66p. Dissertação de Mestrado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

A recomendação de itens a partir do *feedback* implícito dos usuários consiste em identificar padrões no interesse dos usuários por estes itens a partir de ações dos usuários, tais como cliques, interações ou o consumo de conteúdos específicos. Isso, de forma a prover sugestões personalizadas que se adéquam ao gosto destes usuários. Nesta dissertação, avaliamos a performance de alguns modelos de fatoração matricial otimizados para a tarefa de recomendação a partir de dados implícitos no consumo das ofertas de vídeos da Globo.com. Propusemos tratar estes dados de consumo como indicativos de intenção de um usuário em assistir um vídeo. Além disso, avaliamos como os vieses únicos dos usuários e vídeos, e sua variação temporal impactam o resultado das recomendações. Também sugerimos a utilização de um modelo de fatoração incremental otimizado para este problema, que escala linearmente com o tamanho da entrada, isto é, com os dados de visualizações e quantidade de variáveis latentes. Na tarefa de prever a intenção dos usuários em consumir um conteúdo novo, nosso melhor modelo de fatoração apresenta um *RMSE* de 0,0524 usando o viés de usuários e vídeos, assim como sua variação temporal.

## **Palavras-chave**

Filtragem Colaborativa, Sistemas de Recomendação, Aprendizado de Máquina, Modelos de Fatoração Latente, Fatoração Matricial.

## Abstract

Souza, Bruno; Milidiú, Ruy Luiz. **Matrix factorization models for video recommendation.** Rio de Janeiro, 2011. 66p. Dissertação de Mestrado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Item recommendation from implicit feedback datasets consists of passively tracking different sorts of user behavior, such as purchase history, watching habits and browsing activities in order to improve customer experience through providing personalized recommendations that fits into users' taste. In this work we evaluate the performance of different matrix factorization models tailored for the recommendation task for the implicit feedback dataset extracted from Globo.com's video site's access logs. We propose treating the data as indication of a positive preference from a user regarding the video watched. Besides that we evaluated the impact of effects associated with either users or items, known as biases or intercepts, independent of any interactions and its time changing behavior throughout the life span of the data in the result of recommendations. We also suggest a scalable and incremental procedure, which scales linearly with the input data size. In trying to predict the intention of the users for consuming new videos our best factorization models achieves a *RMSE* of 0,0524 using user's and video's bias as well as its temporal dynamics.

## Keywords

Collaborative Filtering, Recommender Systems, Machine Learning, Latent Factor Models, Matrix Factorization.

# Sumário

|     |   |    |
|-----|---|----|
| 1   | Introdução .....                                | 11 |
| 2   | Sistemas de Recomendação.....                   | 16 |
| 2.1 | O Processo de Recomendação .....                | 17 |
| 2.2 | O Problema de Recomendação.....                 | 17 |
| 2.3 | Estratégias de Recomendação .....               | 20 |
|     | Filtragem de Conteúdo .....                     | 21 |
|     | Filtragem Colaborativa .....                    | 22 |
|     | Comparação das Estratégias .....                | 25 |
| 3   | Modelos de Fatoração Matricial .....            | 28 |
| 3.1 | Modelo Matemático.....                          | 29 |
| 3.2 | Fatoração Matricial Incremental.....            | 31 |
| 3.3 | O Fator de Regularização .....                  | 34 |
| 3.4 | Adicionando Particularidades ao Modelo.....     | 35 |
| 3.5 | Adicionando Dinâmicas Temporais ao Modelo ..... | 37 |
| 3.6 | O algoritmo final .....                         | 39 |
| 4   | Aplicação na Globo.com .....                    | 41 |
| 4.1 | Conjunto de Dados .....                         | 41 |
| 4.2 | Métrica de Avaliação.....                       | 45 |
| 4.3 | Experimentos .....                              | 46 |
|     | Quantidade de Passos de Treino .....            | 46 |
|     | Número de Variáveis Latentes .....              | 47 |
|     | Taxa de Aprendizado .....                       | 48 |
|     | Fator de Regularização .....                    | 49 |
|     | Comparação dos Modelos .....                    | 50 |
| 4.4 | Recomendando Vídeos .....                       | 51 |
| 5   | Trabalhos Relacionados.....                     | 53 |
|     | Filtragem Colaborativa .....                    | 53 |
|     | Otimizações de Performance .....                | 55 |
| 6   | Conclusão e Trabalho Futuro .....               | 57 |
| 7   | Referências Bibliográficas.....                 | 60 |

## **Lista de Figuras**

|  |    |
|--|----|
| Figura 2.1 - Processo de recomendação .....                      | 17 |
| Figura 2.2 - Recomendação baseada em filtragem de conteúdo ..... | 21 |
| Figura 2.3 - Filtragem colaborativa na Amazon.com .....          | 23 |
| Figura 2.4 - Processo de formação de vizinhança .....            | 24 |
| Figura 2.5 - Processo de filtragem colaborativa .....            | 25 |
| Figura 3.1 - Fatoração matricial incremental.....                | 32 |
| Figura 4.1 - Catálogo de vídeos do Auto Esporte.....             | 41 |
| Figura 4.2 - Quantidade de passos de treino vs. RMSE .....       | 46 |
| Figura 4.3 - Número de variáveis latentes vs. RMSE .....         | 47 |
| Figura 4.4 - Variável latente 1 vs. variável latente 2 .....     | 48 |
| Figura 4.5 - Efeito da taxa de aprendizado.....                  | 48 |
| Figura 4.6 - Efeito da regularização .....                       | 49 |
| Figura 4.7 - Comparação dos modelos de fatoração .....           | 50 |

## **Lista de Tabelas**

|   |    |
|---|----|
| Tabela 2.1 - Matriz de avaliações para recomendação de filmes ..... | 19 |
| Tabela 3.1 - Um exemplo de matriz usuário item .....                | 28 |
| Tabela 4.1 - Matriz de visualizações de vídeos .....                | 43 |
| Tabela 4.2 - Formatação dos dados de entrada.....                   | 44 |
| Tabela 4.3 - Quantidade de passos de treino vs. <i>RMSE</i> .....   | 47 |
| Tabela 4.4 - Quantidade de variáveis latentes vs. <i>RMSE</i> ..... | 47 |
| Tabela 4.5 - Efeito da taxa de aprendizado .....                    | 49 |
| Tabela 4.6 - Efeito da regularização.....                           | 49 |
| Tabela 4.7 - Comparaçao dos modelos de fatoração .....              | 50 |
| Tabela 4.8 - Exemplo de recomendações para um usuário .....         | 52 |

## **Lista de Algoritmos**

|   |    |
|---|----|
| Algoritmo 1 - Versão básica.....                                | 34 |
| Algoritmo 2 - Versão final com viés e dinâmicas temporais ..... | 40 |

## 1 Introdução

Hoje em dia, o consumidor tem ao seu alcance uma vasta seleção de produtos e conteúdo sendo oferecidos por diversas lojas eletrônicas que se propõem a suprir as mais diversas necessidades e se adequar o máximo possível ao gosto destes consumidores. Assim, oferecer o item mais apropriado para um determinado usuário é um grande diferencial para conseguir aumentar seu nível de satisfação e garantir sua lealdade à loja em questão.

Neste cenário, cada vez mais os principais portais de comércio eletrônico e de conteúdo estão se interessando em sistemas de recomendação capazes de identificar padrões no interesse dos usuários pelos produtos. Isso, de forma a prover recomendações personalizadas que se adéquam ao gosto desses usuários. Além disso, dado que boas recomendações tem um papel fundamental no que se refere à experiência de uso, temos visto as principais referências na venda de produtos *online* disponibilizando essas recomendações como mais um importante eixo para descoberta de conteúdo em seus portais. É o caso da Amazon.com [1] e da Netflix [2].

Estes sistemas são particularmente interessantes para produtos de entretenimento, como filmes, música, seriados, e até conteúdo gerado colaborativamente (ex. Youtube). Os usuários tendem a assistir os mesmos vídeos e cada um deles também está disposto a assistir outros diferentes, dependendo do que lhes é oferecido. Eles também costumam interagir bastante com o ambiente no qual o consumo destes vídeos é feito. Uma dessas formas de interação é indicar explicitamente seu nível de satisfação pelo conteúdo adquirido. Isso gera um volume enorme de dados indicando que conteúdo é oferecido para o usuário consumir e, além disso, como se dá este consumo. A partir destes dados as empresas podem entender o perfil de consumo de seus usuários e ficam habilitadas a fazer ofertas personalizadas para os mesmos.

O crescimento continuo da quantidade de usuários e de produtos lança dois grandes desafios a estes tipos de sistemas. O primeiro é melhorar a qualidade das recomendações que são feitas aos usuários. Dado que boas

recomendações aumentam a quantidade de produtos consumidos, enquanto que recomendações ruins acarretam na perda de usuários. O outro desafio é garantir a escalabilidade dos algoritmos de recomendação. Estes dois desafios tendem a ser conflitantes, pois se o algoritmo gasta pouco tempo na modelagem das recomendações é porque estamos tendo perdas na qualidade das mesmas. Sendo assim, é importante considerar estes dois aspectos simultaneamente no projeto de um sistema destes.

Os sistemas de recomendação costumam seguir um processo bastante específico ao modelar as recomendações para os usuários. Eles recebem o perfil dos usuários e informações sobre os itens ou produtos e produzem as recomendações. Em outras palavras, um sistema de recomendação consiste de dados contextuais, a informação que o sistema possui antes de começar o processo de recomendação, dados de entrada, a informação que o usuário precisa passar para o sistema conseguir gerar as recomendações, e um algoritmo capaz de combinar os dados do contexto com os dados de entrada de forma a modelar adequadamente as recomendações.

Basicamente, as técnicas de recomendação atuais são baseados em duas estratégias distintas: filtragem de conteúdo e filtragem colaborativa. No Capítulo 2 procuramos descrever melhor o funcionamento destes métodos.

Dentre muitas estratégias distintas, os modelos de filtragem colaborativa são, provavelmente, os mais utilizados. Uma das principais vantagens destes modelos é o fato de eles se adaptarem a qualquer domínio, podendo endereçar especificidades dos conjuntos de dados que dificilmente seriam aproveitadas utilizando outros métodos.

O objetivo desta dissertação é avaliar modelos de fatoração matricial que têm se mostrado bastante eficientes no problema de recomendação de vídeos a partir do *feedback* implícito dos usuários num domínio temporal.

Diferentemente dos sistemas de recomendação que utilizam *feedback* explícito dos usuários, não temos nenhuma indicação clara dos usuários a cerca de suas preferências por um vídeo. Em particular, falta-nos a evidência de quais vídeos o usuário não gosta. Também vale lembrar que a percepção e a

popularidade dos vídeos está em constante mudança à medida que novos conteúdos ficam disponíveis nos *sites*. Da mesma forma, a preferência dos usuários também está evoluindo. Isto nos remete à possibilidade de avaliar algumas dinâmicas temporais nos modelos de fatoração a serem propostos neste trabalho e o seu impacto nas recomendações.

Trabalhamos na modelagem de um algoritmo de recomendação que conseguisse identificar características de *feedback* implícito dos usuários em nosso conjunto de dados. Isso porque tratamos estes dados como indicadores positivos de preferência sabendo que os níveis de confiabilidade destes indicadores tendem a variar consideravelmente. Isto nos levou a um modelo de fatoração matricial bastante adaptado para sistemas de recomendação baseados em *feedback* implícito.

Também implementamos um modelo que identifica vieses associadas aos itens ou usuários (*bias*) independente das interações entre os mesmos, e que rastreia alterações no comportamento dos usuários dentro de um eixo temporal no nosso conjunto de dados e avaliamos seu impacto no resultado recomendações.

Esta dissertação está estruturada da seguinte forma:

No capítulo 2 expomos as idéias por trás do funcionamento dos sistemas de recomendação e apresentamos uma explicação detalhada das principais abordagens utilizadas de acordo com cada domínio.

No capítulo 3, alguns modelos de fatoração latente são descritos. As formulações matemáticas, os algoritmos e suas variações são explicados em detalhes.

No capítulo 4 apresentamos os resultados das experimentações feitas com dados do portal de vídeos da Globo.com. A coleta e o esquema de representação de dados e as avaliações dos resultados são discutidos.

No capítulo 5 elencamos alguns trabalhos relacionados ao tema de recomendação e filtragem colaborativa com abordagens bastante interessantes, e também outros estudos mais focados em otimização e ganhos de performance.

Finalmente, no capítulo 6 temos as conclusões extraídas desta dissertação e apresentamos possíveis extensões para este trabalho.

## 2 Sistemas de Recomendação

Sistemas de recomendação são utilizados em portais de vendas e conteúdo com a finalidade de prever a preferência dos usuários. Predições com boa precisão podem resultar em margens de venda maiores e aumentar a satisfação dos usuários. As principais funções dos sistemas de recomendação são analisar os dados dos usuários e extrair informações úteis para futuras predições [4]. Existem diversas técnicas para implementação de sistemas de recomendação, incluindo as baseadas em conteúdo, as colaborativas, entre outras. Para melhorar a performance das predições, estes métodos podem ser combinados em sistemas híbridos.

Existem vários sistemas deste tipo acessíveis pela internet  que visam gerar recomendações para diversos tipos de produtos como musicas, filmes, livros, etc. Por exemplo, sistemas de recomendação são agora parte integral de diversos portais de comércio eletrônico como a Amazon.com e a iTunes Store [5]. De uma forma geral, estes sistemas buscam adquirir opiniões ou preferências sobre itens de um grupo de usuários, e usar estas opiniões para apresentar itens que possam fazer sentido para outros usuários.

A partir desta descrição geral vemos que os sistemas de recomendação precisam de, basicamente, duas coisas para poder funcionar de maneira apropriada:

- 1 Informações sobre as preferências dos usuários
- 2 Um método para determinar se um item é interessante para um usuário

Normalmente, as preferências dos usuários compreendem informações externas como suas características pessoais (idade, sexo, localidade, etc.), seu histórico de interações com o portal em questão, e suas avaliações em cima dos produtos [6]. A forma para determinar se um item é interessante para um usuário ou não depende do tipo de sistemas de recomendação. Neste capítulo, discutiremos algumas técnicas de recomendação comumente utilizadas.

## 2.1 O Processo de Recomendação

Em geral, pode-se dizer que todo sistema de recomendação segue um processo bem definido a fim de criar recomendações. Pensando no processo de recomendação como uma caixa preta, como mostrado na Figura 2.1, podemos identificar duas fontes de informação necessárias como entrada do processo. Estas fontes de informação correspondem aos dados dos usuários e as informações a respeito dos itens e dos usuários. Idealmente, estes dados relacionados ao perfil dos usuários deveriam ser fornecidos explicitamente pelos próprios usuários. No entanto, estas informações também podem ser extraídas de outras fontes como a navegação pelas páginas, itens consumidos ou comprados, forma como os itens foram consumidos, etc.

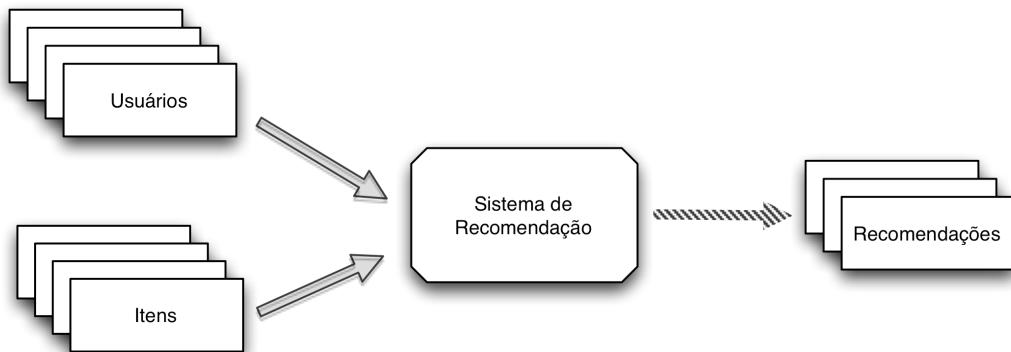


Figura 2.1 - Processo de recomendação

## 2.2 O Problema de Recomendação

Métodos de recomendação tem sido utilizados informalmente há anos. Recomendações positivas e negativas ajudam as pessoas a descobrir coisas novas que eles vão gostar ou evitar alternativas ruins que eles não vão gostar. Isto resulta em economia de tempo e reduz a sobrecarga de informações, que é um problema sério hoje em dia.

Sistemas de recomendação surgiram como uma área de pesquisa independente em meados de 1990 quando os pesquisadores passaram a se concentrar em problemas de recomendação que contavam explicitamente com a estrutura das avaliações [62, 63, 64]. Durante a última década, muito tem sido feito tanto na indústria quanto no meio acadêmico no desenvolvimento de novas

abordagens para sistemas de recomendação e o interesse nesta área ainda permanece.

Informalmente, o problema de recomendação pode ser reduzido a um problema de estimar avaliações para os itens que não foram experimentados por um usuário. Intuitivamente, esta estimativa é geralmente baseada na avaliação dada por este usuário para outros itens e em algumas outras informações relacionadas ao contexto. Uma vez que as avaliações para os itens ainda não avaliados podem ser calculadas, os itens de maior avaliação estimada podem ser recomendados para o usuário.

Formalmente, o problema de recomendação pode ser representado da forma a seguir [13]. O modelo formal de um sistema de recomendação consiste em três itens, um conjunto  $C$  com os usuários, um conjunto  $S$  com os itens passíveis de recomendação, tais como livros, filmes ou restaurantes e uma função de utilidade  $u$ . O espaço de itens possíveis  $S$  pode ser muito grande, com todos os filmes ou livros que poderão ser recomendados. Da mesma forma, o espaço dos usuários também pode ser muito grande, chegando a casa dos milhões, em alguns casos. A parte fundamental do modelo formal é a função de utilidade, que mede a utilidade de um item  $c$  para um usuário  $s$ , isto é,  $u:C \times S \rightarrow R$ . Aqui,  $R$  é um conjunto ordenado, podendo conter inteiros não negativos, ou números reais em um determinado intervalo, o que depende do esquema de representação utilizado no sistema de recomendação.

Usando o modelo descrito, o problema de recomendação é reduzido a escolher um item  $s_i \in S$ , para um usuário  $c_j \in C$  que maximiza a função de utilidade. Em sistemas de recomendação a utilidade de um item é geralmente representada por uma avaliação indicando o quanto um usuário em particular gosta de um item em particular.

Cada usuário do espaço  $C$  pode ser definido usando um perfil. Isso é o chamado perfil de usuário, que contém informações sobre os gostos do usuário para o domínio de recomendação. A representação do perfil do usuário depende da abordagem e do domínio, mas, por exemplo, ela pode incluir características de vários usuários, como idade, sexo, renda, estado civil, entre outros. No caso mais

simples, o perfil de usuário pode conter apenas um único elemento, como o ID de usuário.

Da mesma forma, cada item do espaço  $S$  pode ser definido com um conjunto de características, dependendo do esquema de representação escolhido. Por exemplo, em uma aplicação de recomendação de filmes  $S$  pode ser uma coleção de filmes. Cada um dos filmes em  $S$  pode ser representado não só pelo seu identificador único, mas também por sua informação de conteúdo, como seu título, gênero, diretor, ano de lançamento, atores principais, entre outros, o que é pensado para ter efeito sobre o processo de recomendação.

|          | <b>Matrix</b> | <b>Rocky</b> | <b>Hannibal</b> | <b>Ghost</b> |
|----------|---------------|--------------|-----------------|--------------|
| João     | 1             | 1            | 1               | $\otimes$    |
| Joana    | $\otimes$     | -1           | $\otimes$       | 1            |
| Jaime    | 1             | -1           | -1              | -1           |
| Bernardo | $\otimes$     | 1            | 1               | -1           |

Tabela 2.1 - Matriz de avaliações para recomendação de filmes

A principal preocupação de um sistema de recomendação é que a função de utilidade  $u$  geralmente não é definida no espaço todo de  $C \times S$ . A utilidade é representada tipicamente por avaliações e é inicialmente definida apenas nos itens previamente avaliados pelos usuários. Por exemplo, em um sistema de recomendação de filmes, os usuários inicialmente avaliam algum subconjunto de filmes que eles já viram. Um exemplo de matriz de avaliação dos usuários para os itens para uma aplicação de recomendação de filmes é apresentado na Tabela 2.1, onde as avaliações são especificadas como 1, positiva, ou -1, negativa. O " $\otimes$ " significa que os usuários não avaliaram os filmes correspondentes. Portanto, o mecanismo de recomendação deve ser capaz de estimar ou prever as avaliações dessas combinações usuário-filme e, em seguida, fazer recomendações apropriadas com base nessas predições. Se considerarmos o espaço  $C \times S$  como uma matriz de avaliações, muitas das avaliações não são fornecidas inicialmente, por isso a matriz é dita esparsa. Assim, o trabalho do sistema de recomendação é extrapolar  $u$  para todo o espaço  $C \times S$ . O caminho que é seguido para fazer estas

extrapolações define a função de utilidade. Esta tarefa de extração tem sido abordada de diversas maneiras [64, 65, 66]:

1. Especificando heurísticas que irão definir a função de utilidade empiricamente através da validação de seu desempenho.
2. Estimar a função de utilidade através da otimização de critérios de desempenho pré-definidos, como o erro médio quadrático [13].

### 2.3 Estratégias de Recomendação

As técnicas de recomendação têm algumas possíveis classificações [5, 7, 8].

Especificamente, os sistemas de recomendação possuem:

- dados do contexto, as informações que o sistema tem antes do processo de recomendação iniciar;
- dados de entrada, as informações que os usuários precisam comunicar ao sistema visando a geração das recomendações
- um algoritmo que combina os dados do contexto e os dados de entrada para produzir as sugestões.

Desta forma, as técnicas de recomendação mais comuns podem ser agrupadas em duas categorias distintas. A abordagem de filtragem de conteúdo busca criar um perfil para cada usuário ou produto de forma a caracterizar sua natureza. Por exemplo, para o caso de o produto ser um vídeo, pode-se usar o canal onde o mesmo é exibido, o programa ao qual ele pertence, os atores que participam e outros dados para a formação deste perfil para então recomendar outros vídeos que se encaixem dentro deste perfil. O mesmo acontece na criação dos perfis dos usuários, onde podem ser utilizadas informações como idade, sexo, localização geográfica, ou até mesmo respostas fornecidas em um questionário dedicado. Estratégias baseadas em conteúdo demandam informações adicionais dos produtos que, muitas vezes, não estão disponíveis ou são difíceis de se coletar [10].

Uma outra estratégia é baseada no comportamento prévio dos usuários sem a necessidade de derivarmos um perfil para o mesmo [10]. Esta abordagem é conhecida como filtragem colaborativa. Não há a necessidade de explicitar perfis para cada item ou usuário. A idéia é analisar as relações entre os usuários e as interdependências entre os produtos para poder identificar novas associações entre os produtos e os usuários.

### Filtragem de Conteúdo

A principal idéia por trás dos sistemas de recomendação baseados em conteúdo é sugerir itens para os usuários que sejam similares ao itens que estes usuários já tenham avaliado positivamente em algum momento antes. Recomendações baseadas em filtragem de conteúdo são uma extensão das pesquisas relacionadas a filtragem de informação [11]. Em um sistema baseado em conteúdo, os objetos de interesse são definidos pelas suas características. Por exemplo, o NewsWeeder [12], um sistemas de recomendação de notícias, usa as palavras dos textos para caracterizar os mesmos.

Algoritmos de recomendação baseados em conteúdo aprendem os perfis de interesse dos usuários com base nas características presentes em cada um dos itens que esses usuários avaliaram previamente. Sendo assim, eles constroem os perfis dos usuários a partir do perfil dos itens. O tipo de perfil derivado depende do método de aprendizado empregado. Árvores de decisão, redes neurais, e representações vetoriais são alguns exemplos destes métodos que tem sido comumente utilizados. Estes perfis de usuários são modelos que demandam algum tempo de observação e são atualizados a medida em que novas evidências relacionadas às preferências dos usuários são observadas.

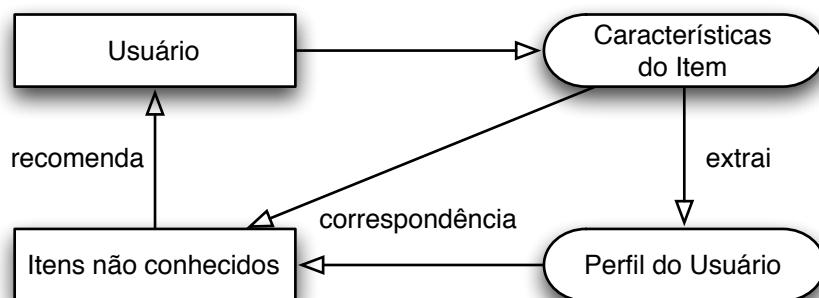


Figura 2.2 - Recomendação baseada em filtragem de conteúdo

Em termos formais, sistemas de recomendação baseados em filtragem de conteúdo estimam a utilidade  $u(c,s)$  de um item  $s$  para um usuário  $c$  baseado nas utilidades  $u(c,s_i)$  indicadas pelo usuário  $c$  para os itens  $s_i \in S$  que são “similares” ao item  $s$  [13].

Um exemplo de sucesso da abordagem de filtragem de conteúdo é o Projeto *Music Genome* [9], usado no serviço de rádio online Pandora.com [35]. Uma pessoa especializada neste tipo de identificação classifica as músicas da rádio levando em consideração centenas de características musicais. Estes atributos esclarecem não apenas a identidade musical de cada faixa como também os fatores relevantes para entender as preferências musicais dos ouvintes.

### Filtragem Colaborativa

Filtragem colaborativa é uma técnica que utiliza o histórico de interações entre itens e usuários visando encontrar relacionamentos entre os mesmos. Um exemplo de relacionamento pode ser dois vídeos sendo visualizados por muitos usuários, indicando que eles são similares. Nenhuma informação contextual a respeito do item em questão é levada em consideração, ou seja, os algoritmos não têm ciência de sobre quais itens ou que tipo de itens eles estão processando.

Em termos formais, sistemas de recomendação baseados em filtragem colaborativa estimam a utilidade  $u(c,s)$  de um item  $s$  para um usuário  $c$  baseado na utilidade  $u(c_j,s)$  do item  $s$  indicadas pelos usuários  $c_j \in C$  que são “similares” ao usuário  $c$  [13]. Por exemplo, num sistema de recomendação de vídeos, para recomendar um vídeo para o usuário  $c$ , o sistema tentará encontrar os “pares” do usuário  $c$ , ou seja, outros usuários com gosto parecido ou que tenham avaliado os mesmos vídeos de forma parecida.

É comum nos sistemas de filtragem colaborativa que os perfis dos usuários sejam armazenados como vetores de itens e suas avaliações para aqueles itens. Esses vetores tendem a aumentar continuamente a medida em que os usuários interagem com o sistema. Alguns sistemas levam em consideração dinâmicas temporais para descontar os desvios no padrão de interesse dos usuários com o passar do tempo [14]. As avaliações podem ser binárias, tais como

gostei ou não gostei, ou valoradas de acordo com o nível de preferência. Dentre os sistemas mais conhecidos que utilizam esta abordagem temos o da Netflix [15], da MovieLens [16], da Amazon [17], entre outros.

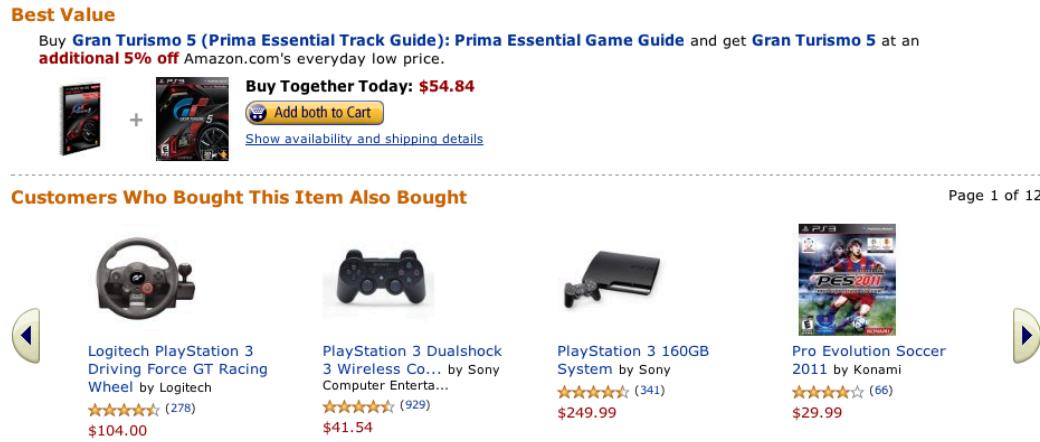


Figura 2.3 - Filtragem colaborativa na Amazon.com

As duas formas mais comuns de implementar uma solução de filtragem colaborativa são os algoritmos de vizinhança ou grafos, e modelos de fatoração latente [3].

Métodos de vizinhança visam computar relacionamentos entre itens ou usuários e construir um modelo baseado num grafo que descreva a vizinhança:

- Métodos item-item constroem os grafos de vizinhança com os vértices conectando itens similares;
- Métodos usuário-usuário constroem os grafos de vizinhança com os vértices conectando usuários com gosto parecido.

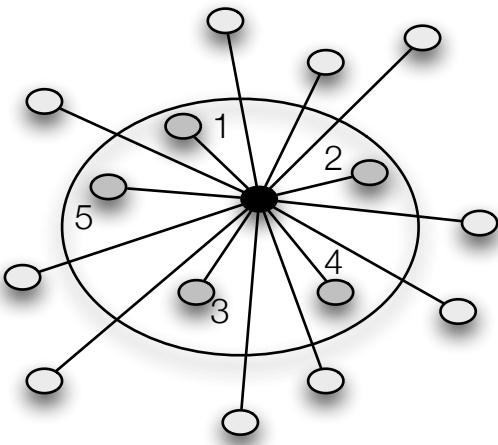


Figura 2.4 - Processo de formação de vizinhança

Estes métodos tem obtido sucesso em muitos domínios, mas seus algoritmos possuem algumas limitações:

1. Esparsidade dos dados. Métodos de vizinhança se baseiam em correspondências exatas, fazendo com que os algoritmos sacrificiem a precisão dos sistemas de recomendação [61]. Uma vez que o coeficiente de correlação é definido apenas entre usuários que avaliaram, pelo menos, dois itens em comum, muitos pares de usuários não têm nenhuma correlação entre si. Na prática, muitos sistemas de recomendação são usados para operar grandes massas de dados, por exemplo, a Amazon.com recomendando livros e o eBay com produtos usados. Nestes sistemas, mesmo os usuários altamente ativos podem ter interagido ou avaliado bem menos que 1% dos itens. Assim, os algoritmos de vizinhança podem ser incapazes de fazer boas recomendações de itens para um determinado usuário.
2. Escalabilidade. Algoritmos de vizinhança demandam um esforço computacional que seja capaz de crescer tanto com o número de usuários quanto com o número de itens. Com milhões de clientes e produtos, um sistema de recomendação *web* padrão executando os algoritmos existentes acabam enfrentando problemas de escalabilidade.
3. Sinônimos. Num cenário real, nomes de produtos diferentes podem se referir a objetos semelhantes. Sistemas de recomendação baseados em

correlação não conseguem encontrar esta associação latente e tratam estes itens como produtos diferentes.

Modelos de fatoração latente tentam explicar o interesse dos usuários por certos itens caracterizando os mesmos em, digamos, 20 a 100 fatores latentes aprendidos nessas relações de interesse. Eles constroem uma aproximação da matriz de interações entre os usuários e os itens e usam essa aproximação para estimar a probabilidade de outras interações. Focamos este trabalho numa especialização do problema de Fatoração de Matrizes Não-Negativas (*NMF*, do inglês *Non-Negative Matrix Factorization*) [41]. Descrevemos em detalhes o funcionamento deste método no capítulo 3.

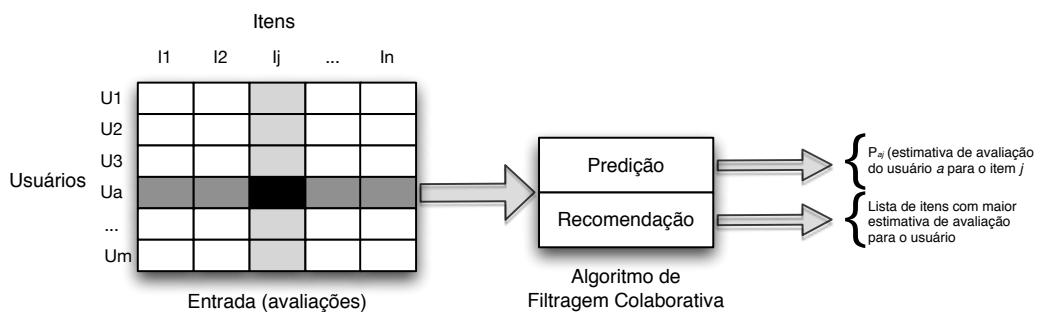


Figura 2.5 - Processo de filtragem colaborativa

Existem muitos desafios para os sistemas de filtragem colaborativa. Seus algoritmos precisam ter performance em conjuntos de dados altamente esparsos, escalar com o aumento do número de usuários e itens, conseguir conceber boas recomendações num curto período de tempo e ainda lidar com outros problemas como o ruído no conjunto de dados e de privacidade.

### Comparação das Estratégias

Há vantagens e desvantagens de ambos os tipos de abordagens para sistemas de recomendação.

A principal desvantagem das abordagens baseadas em filtragem de conteúdo em comparação com as baseadas em filtragem colaborativa é que elas dependem muito dos metadados que descrevem os itens, o que implica que essas informações devem ser obtidas e inseridas no sistema. Assim, a manutenção de um sistema de recomendação baseado em conteúdo requer um esforço

significativo, já que existe a necessidade constante de manter as informações atualizadas ou acrescentar novas informações quando novos itens são adicionados ao sistema.

Uma outra questão é que nem todas as informações podem ser expressas facilmente ou de forma consistente. Algo simples como o gênero de um filme não pode ser utilizado diretamente, pois esta informação nem sempre é bem definida. Os usuários podem ter interpretações diferentes a respeito do gênero ao qual um filme pertence, além disso um mesmo filme pode estar situado entre dois gêneros distintos. Usar os dados de gênero pode, assim, causar problemas quando o sistema se baseia neste tipo de informações.

Abordagens de filtragem colaborativa não dependem destes tipos de informação, o que as torna mais útil em domínios onde não está claro quais atributos dos itens implicam nas preferências dos usuários. A principal deficiência de algoritmos de filtragem colaborativa, porém, é o que se chama de problema de partida a frio. Uma vez que estes algoritmos dependem de informações de avaliação, é impossível recomendar itens que ainda não foram avaliados pelos usuários, ou dar recomendações a usuários que não forneceram qualquer informação de preferência.

Além disso, uma vez que esses sistemas contam com informações de preferência, é necessário que os usuários compartilhem seus gostos. Usuários que estão muito preocupados com a questão de privacidade pode considerar essa necessidade de compartilhamento um problema. Em sistemas de recomendação baseados em conteúdo, não há necessidade de compartilhar as informações entre os usuários, pois os algoritmos só utilizam as avaliações pessoais e semelhanças entre os usuários em termos de meta-informação, e não comparando os padrões de avaliação.

Levando essas vantagens e desvantagens em consideração a conclusão é que diferentes abordagens são mais adequadas em diferentes domínios. Existem tentativas de combinar filtragem colaborativa e filtragem baseada em conteúdo em sistemas de recomendação de filmes visando contornar as deficiências de cada um dos métodos [65, 67], mas por conta da facilidade de manutenção os

algoritmos de filtragem colaborativa têm sido mais utilizados na maioria dos sistemas de recomendação hoje em dia.

### 3 Modelos de Fatoração Matricial



Algoritmos baseados em fatoração matricial são as implementações de maior sucesso dos modelos de fatoração latente. Em termos gerais, esta técnica é capaz de caracterizar tanto itens quanto usuários através de vetores de variáveis latentes inferidas de padrões de avaliação dos itens. Uma alta correspondência entre as variáveis latentes de um item e um usuário levam a uma recomendação. Ela tem se tornado popular nos últimos anos por conseguir combinar uma boa escalabilidade com precisão nas previsões. Além disso, oferece grande flexibilidade para modelar diversas situações do dia-a-dia.

Os sistemas de recomendação dependem de diversos tipos de dados de entrada, os quais são geralmente estruturados numa matriz quadrada onde uma dimensão representa os usuários e a outra os itens de interesse. O objetivo nesses sistemas é prever como os usuários avaliariam um item que eles ainda não avaliaram para então conseguir recomendar itens para os mesmos. Assumindo, por exemplo, que temos 5 usuários e 10 itens, e que as avaliações são 1 para as positivas e -1 para as negativas, a matriz que representa o interesse dos usuários pelos itens toma a forma abaixo (um hífen significa que o usuário ainda não avaliou o item).

|    | I1 | I2 | I3 | I4 | I5 | I6 | I7 | I8 | I9 | I10 |
|----|----|----|----|----|----|----|----|----|----|-----|
| U1 | 1  | -  | -1 | 1  | -  | -1 | -1 | -  | 1  | -   |
| U2 | 1  | -  | 1  | -  | 1  | 1  | 1  | 1  | -1 | -   |
| U3 | -1 | 1  | -  | -1 | -  | -1 | 1  | -  | -  | 1   |
| U4 | -  | 1  | -1 | 1  | 1  | -  | 1  | -  | -1 | 1   |
| U5 | 1  | -1 | 1  | -  | -1 | -  | -1 | -1 | 1  | -1  |

Tabela 3.1 - Um exemplo de matriz usuário item

Os dados mais convenientes para o bom funcionamento desses sistemas são os que o usuário fornece explicitamente como, por exemplo, avaliações sobre seu interesse ou desinteresse por um item em questão. Pilászy e Tikk [38] sugerem que mesmo um número pequeno de avaliações é melhor do que uma vasta quantidade de atributos e metadados dos itens ou dos usuários. A Netflix

coleta estes tipos de dados para seus vídeos através de um sistema de avaliação por estrelas em seu site numa escala que varia de 1 a 5. No TiVo, os usuários indicam suas preferências por séries de TV pressionando botões que indicam avaliações positivas ou negativas sobre o conteúdo assistido. Esses dados explícitos fornecidos pelos usuários geram uma matriz esparsa, já que um único usuário tende a avaliar apenas uma pequena parcela dos itens consumidos.

Assim, a tarefa de prever as avaliações que faltam pode ser considerada como preencher as células vazias (marcadas com hífen) com valores que fossem consistentes com as demais avaliações existentes.

A intuição por traz do uso da técnica de fatoração de matrizes para a resolução deste problema é a de que devem haver algumas variáveis latentes que determinam como um usuário avalia um item. Por exemplo, dois usuários avaliariam positivamente um filme se ambos gostam os atores daquele filme, ou se o filme é um filme de ação, o qual é um gênero que os dois gostam. Então, se conseguirmos descobrir estas características latentes, deveríamos conseguir também prever uma avaliação para um determinado usuário a cerca de um item, pois essas variáveis latentes associadas a um usuário deveriam casar com as de um dado item.

O ponto forte dessa abordagem é que ela permite a incorporação de fontes de informação adicionais. Quando as avaliações explícitas dos usuários não estão disponíveis, esses sistemas podem inferir as preferências dos usuários usando dados implícitos, os quais refletem indiretamente sua opinião através da observação de seu comportamento, como seu histórico de compras, seus vídeos vistos, sua navegação pelo site, suas buscas, ou até mesmo os movimentos do seu mouse. Esses dados implícitos denotam a presença ou ausência de determinados eventos e também são utilizados no preenchimento da matriz.

### 3.1 Modelo Matemático

O conjunto  $U = \{u_1, u_2, \dots, u_p\}$  define os usuários do sistema de filtragem colaborativa e o conjunto  $V = \{v_1, v_2, \dots, v_p\}$  os itens disponíveis. O conjunto  $T$  é chamando de conjunto de treino:

$$T \subset U \times V \times O$$

com  $O$  representando os possíveis valores para as avaliações. O conjunto  $T$  define todas as avaliações conhecidas consistindo de um usuário, um item e a avaliação que o usuário deu para um item.

A função  $e$  explica o mapeamento de um usuário  $u_i \in U$  e um item  $v_j \in V$  para uma avaliação  $r_{ij}$ .

$$e : U \times V \rightarrow O$$

$$e(u_i, v_j) = r_{ij}$$

A função  $e$  é estimada por um modelo  $\hat{e}$  que prevê as avaliações para um dado par usuário-item. O modelo  $\hat{e}$  está relacionado com o correto mapeamento  $e$  por

$$\hat{e}(u_i, v_j) = e(u_i, v_j) + Z$$

onde  $Z$  é o erro entre a predição e a avaliação de fato. Geralmente, supõe-se que  $Z$  segue uma distribuição normal

$$Z \sim N(\mu, \sigma^2)$$

com media  $\mu$  e variância  $\sigma^2$ .

A matriz de avaliações  $R$ , dada por

$$R = (r_{ij})_{i=1, \dots, p, j=1, \dots, q}$$

é a matriz com todas as avaliações, em que não existem células vazias. Na maioria dos casos esta matriz é desconhecida, pois dificilmente em um sistema de recomendação todos os usuários avaliam todos os itens. Geralmente, conhecemos apenas um sub-conjunto desta matriz.

Como falamos anteriormente, o objetivo de um sistema de recomendação é recomendar novos itens que um usuário pode estar interessado. Para isso, um preditor é necessário para aprender a partir das avaliações fornecidas de forma a

prever as avaliações que não foram dadas ainda. Desta forma, uma possível avaliação positiva pode ser dada por um usuário para um item e, por esta razão, o sistema recomendaria este item para o usuário.

Alguns sistemas de recomendação definem o erro médio quadrático (*RMSE*, do inglês *Root Mean Square Error*) como ferramenta para medir os erros de predição. Dessa forma, para um dado conjunto dados de teste  $Q \subset U \times V$  podemos dizer que temos como objetivo minimizar a parte quadrática do *RMSE*.

$$\sum_{(u_i, v_j) \in Q} (\hat{e}(u_i, v_j) - r_{ij})^2 \rightarrow \min$$

Treinar  $\hat{e}$  requer bons ajustes de  $\hat{e}$ . Estes ajustes poderiam ser encontrados minimizando o RMSE para o conjunto de treino. No entanto, utilizar apenas o conjunto de dados de treino para esta validação pode nos levar a um excesso ou especialização do treinamento.

### 3.2 Fatoração Matricial Incremental

O uso de fatoração matricial foi sugerido por Brandyn Webb (também conhecido por Simon Funk) em seu blog [18] durante a competição Netflix Prize [19]. Esta técnica está fortemente relacionada à decomposição de valores singulares (*SVD*, do inglês *Singular Value Decomposition*) [20] [22] [24].

A idéia por trás de uma abordagem de *SVD* clássica é decompor uma matriz de avaliações  $P$  de dimensão  $n \times m$  em um produto de três matrizes,

$$P = A\Sigma B^T$$

onde  $A$  possui dimensão  $n \times n$ ,  $\Sigma$  possui dimensão  $n \times m$  e  $B$  possui dimensão  $m \times m$ . As matrizes  $A$  e  $B$  são ortogonais.  $\Sigma$  é uma matriz diagonal com  $k$  entradas diferentes de zero. Assim sendo, as dimensões efetivas das matrizes  $A$ ,  $\Sigma$  e  $B$  são  $n \times k$ ,  $k \times k$  e  $k \times m$ , respectivamente. Estas  $k$  entradas diagonais  $\varsigma_i$  da matriz  $\Sigma$  são todas positivas com  $\varsigma_1 \geq \varsigma_2 \geq \dots \geq \varsigma_k \geq 0$ . As colunas de  $A$  e  $B$  são, respectivamente, chamadas de autovetores a esquerda e a direita de  $P$ .

Se pegarmos apenas os  $r << k$  valores singulares mais significativos, o

produto destas três matrizes será uma matriz de posto  $r$ , a qual referenciamos por  $P'$  e é a melhor aproximação de  $P$  com posto  $r$  considerando a norma de Frobenius.

$$\|P' - P\|_F^2 = \sum_{i=1}^n \sum_{j=1}^m |p'_{ij} - p_{ij}|^2$$

O uso da técnica *SVD* também traz à tona alguns problemas quando aplicada ao domínio aqui proposto. Geralmente, a matriz de avaliações  $R$  é uma matriz esparsa e a técnica apresentada não funciona com este tipo de matriz. Algumas adaptações tem sido feitas para minimizar este problema, como preencher as entradas em branco com zeros ou utilizando outras técnicas mais sofisticadas [21] [23]. Um segundo problema é o fato de  $R$  ser, tipicamente, grande. Calcular o *SVD* para matrizes grandes requer uma enorme capacidade de processamento computacional.

A abordagem incremental para o *SVD* proposta por Simon Funk funciona consideravelmente bem e é também fácil de se implementar, endereçando diretamente estes problemas. Trata-se de um algoritmo de gradiente descendente que calcula a aproximação utilizando apenas os valores conhecidos de  $R$ .

A técnica é bem parecida com o *SVD* original. Ao invés de decompor  $R$  em três matrizes, focamos apenas em duas matrizes, as matrizes características.

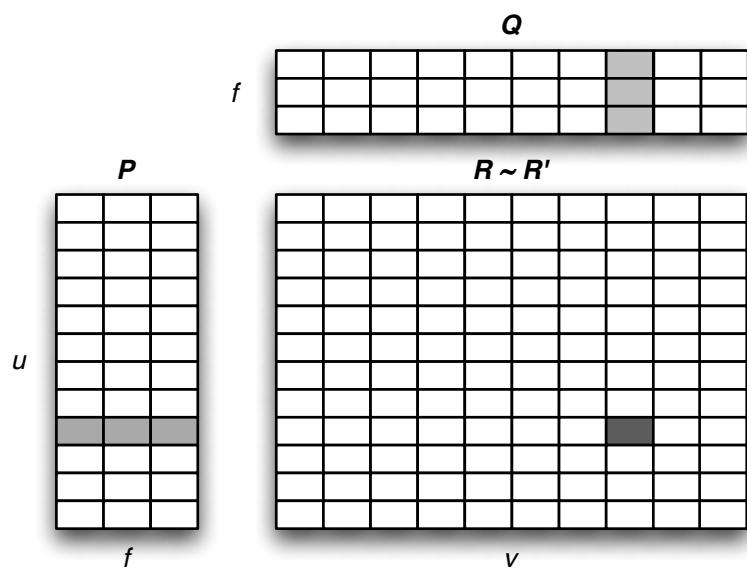


Figura 3.1 - Fatoração matricial incremental

$$R \approx PQ = R'$$

Retornando para o nosso domínio de recomendações, podemos formular o problema da seguinte forma: Suponhamos que temos uma matriz  $R$  ( $u \times v$ ) contemplando as avaliações dos usuários para os itens, e onde  $u$  é o número de usuários e  $v$  o número de itens. E assumamos que queremos considerar  $f$  variáveis latentes. Queremos calcular uma aproximação  $R'$  da matriz  $R$ , tal que  $\|R' - R\|_F$  seja minimizada, e  $R' = P_{u \times f} (Q_{v \times f})^T$ . A  $i$ -ésima linha da matriz  $P$  é o vetor de preferências do usuário  $i$ , e a  $j$ -ésima linha da matriz  $Q$  é o vetor de características do item  $j$ . Dessa forma, conseguimos extrair uma aproximação para os dados desejados, e podemos usá-los para preencher os valores desconhecidos da matriz  $R$  calculando o produto escalar das preferências do usuário pelas características do item

$$\hat{e}(u_i, v_j) = p_i^T q_j = \sum_{k=1}^f p_{ik} q_{kj}$$

onde  $p_{ik}$  e  $q_{kj}$  são componentes dos vetores de preferências do usuário  $i$  e do item  $j$ , respectivamente. Em seguida, podemos calcular o erro quadrático entre a estimativa e a avaliação de fato:

$$\epsilon_{ij}^2 = (r_{ij} - \hat{e}(u_i, v_j))^2 = \left( r_{ij} - \sum_{k=1}^f p_{ik} q_{kj} \right)^2$$

O algoritmo de aprendizado que pode minimizar este erro,  $\epsilon_{ij}^2 \rightarrow \min$ , utiliza o gradiente descendente estocástico para diferenciar a equação em relação às diferentes variáveis latentes e obter as equações de atualização

$$\begin{aligned} p'_{ik} &= p_{ik} + \alpha \frac{\partial \epsilon_{ij}^2}{\partial p_{ik}} = p_{ik} + 2\alpha \epsilon_{ij} q_{kj} \\ q'_{kj} &= q_{kj} + \alpha \frac{\partial \epsilon_{ij}^2}{\partial q_{kj}} = q_{kj} + 2\alpha \epsilon_{ij} p_{ik} \end{aligned}$$

onde  $\alpha$  é taxa de aprendizado. Este número controla o quanto do erro entra no passo de atualização do valor da variável latente em questão. O valor de  $\alpha$  deveria ser consideravelmente pequeno, algo em torno de 0.001. Com a definição

das equações de atualização, podemos definir o algoritmo base abaixo. Esta abordagem [25, 26, 27] combina uma fácil implementação com um tempo de execução relativamente rápido. Sua pseudo-implementação pode ser conferida a seguir.

---

### **Algoritmo 1 - Algoritmo base**

---

Entrada: Matriz de avaliações  $R$  e número de variáveis latente  $f$ .

```

1: Inicialize as matrizes  $P$  e  $Q$  pela primeira vez com um número fixo ou de
   forma randômica
2: para  $k = 1$  até  $f$  faça
3:   repita
4:     para  $\forall(u_i, v_j, r_{ij}) \in T$  faça
5:       calcule  $\varepsilon_{ij}$ .
6:       atualize  $p_{ik}$  e  $q_{kj}$ .
7:     fim
8:     recalcule  $\sum_{(u_i, v_j, r_{ij})} \varepsilon_{ij}^2$ 
9:   até alguma condição de convergência ser atingida
10:  fim
```

Saída: Matrizes  $P$  e  $Q$  com os vetores de variáveis latentes.

---

Algoritmo 1 - Versão básica

Existem diversas variações deste algoritmo. Alguns sugerem treinar uma variável latente de cada vez, outras consideram treinar todas simultaneamente. Esta última forma tende a convergir mais rápido com relação a quantidade total de iterações necessárias. No entanto, em nossa implementação notamos que, apesar de precisarmos de uma quantidade maior de iterações, o primeiro método foi mais rápido pelo fato de conseguirmos otimizar o código cacheando as variáveis latentes conforme elas forem sendo calculadas. Uma vantagem do treino simultâneo é que a variância de cada variável latente não se altera tanto quanto no caso do treino independente. Isto significa que a primeira variável latente acaba tendo um peso bem maior na predição enquanto que as demais tentam apenas prever resíduos menores.

### **3.3 O Fator de Regularização**

Alguns incrementos a este modelo de fatoração matricial tem sido propostos. Um deles consiste em adicionar um fator de regularização  $\beta$  às

equações de atualização. A inclusão deste parâmetro visa suprimir o problema de sobre-treino e, portanto, melhorar a performance das previsões. As equações de atualização ficam assim:

$$p'_{ik} = p_{ik} + \alpha \left( \frac{\partial \varepsilon_{ij}^2}{\partial p_{ik}} - \beta p_{ik} \right) = p_{ik} + \alpha (2\varepsilon_{ij}q_{kj} - \beta p_{ik})$$

$$q'_{kj} = q_{kj} + \alpha \left( \frac{\partial \varepsilon_{ij}^2}{\partial q_{kj}} - \beta q_{kj} \right) = q_{kj} + \alpha (2\varepsilon_{ij}p_{ik} - \beta q_{kj})$$

E a equação de erro passa a ser:

$$\varepsilon_{ij}^2 = \left( r_{ij} - \sum_{k=1}^f p_{ik}q_{kj} \right)^2 + \frac{\beta}{2} \sum_{k=1}^f (p_{ik}^2 + q_{kj}^2) = \left( r_{ij} - \sum_{k=1}^f p_{ik}q_{kj} \right)^2 + \frac{\beta}{2} (\|p\|^2 + \|q\|^2)$$

Chamamos  $\beta \frac{1}{2} \sum_{k=1}^f (p_{ik}^2 + q_{kj}^2)$  de termo de regularização. Assim como a variância das variáveis latentes tende a cair durante o treino, a do parâmetros de regularização  $\beta p_{ik}$  e  $\beta q_{kj}$  também cai à medida em que uma nova variável latente é treinada. Isto significa que a adição destes termos introduz um erro artificial às equações, especialmente para as primeiras variáveis e sua alta variância. E este erro decai com o passar do tempo. Resumindo, este fator desloca o peso na predição das primeiras variáveis latentes para as seguintes e o algoritmo pode treinar com mais iterações e com menos sobre-treino.

### 3.4 Adicionando Vieses ao Modelo

Um dos benefícios de se utilizar uma abordagem de fatoração matricial para filtragem colaborativa é a flexibilidade em lidar com diversos aspectos dos dados do contexto e outros requisitos específicos do domínio de aplicação.

Muito das variações observadas nos valores das avaliações é devido a vieses associados aos usuários ou aos itens em questão, independente de qualquer interação. Por exemplo, alguns dados de sistemas de filtragem colaborativa demonstram uma tendência sistemática de alguns usuários a avaliarem os itens

melhor do que outros usuários. Dessa forma, alguns produtos são percebidos como melhores, ou piores, que outros.

Ao invés de explicar uma avaliação por uma interação na forma  $p_i^T q_j$ , tentamos identificar a porção destes valores que os vieses dos usuários e dos itens poderiam explicar, deixando para as matrizes  $P$  e  $Q$  apenas a porção da interação em si.

Paterek [26] sugere a adição desses vieses ao modelo de predição através da inclusão de duas variáveis, uma para cada item e outra para cada usuário, no cálculo de  $\hat{e}$ . Estas constantes são treinadas simultaneamente. Propomos uma pequena modificação no cálculo de  $\hat{e}$ . Para cada variável latente, disponibilizamos duas variáveis da forma proposta por Paterek e treinamos as mesmas juntamente com as matrizes de variáveis latentes dos usuários e itens.

$$\hat{e}(u_i, v_j) = \sum_{k=1}^f p_{ik} q_{kj} + b_i + b_j$$

Assim, as avaliações observadas são quebradas em três componentes: vieses dos itens, dos usuários e a interação entre o usuário e o item. Isto permite que cada componente indique o seu peso na avaliação prevista.

As equações de atualização para os elementos das matrizes de variáveis latentes dos usuários e itens permanecem inalteradas. As equações de atualização para  $b_i$  e  $b_j$  são

$$b'_i = b_i + \chi \left( \frac{\partial \epsilon_{ij}^2}{\partial b_i} - \delta b_i \right) = b_i + \chi (2\epsilon_{ij} - \delta b_i)$$

$$b'_j = b_j + \chi \left( \frac{\partial \epsilon_{ij}^2}{\partial b_j} - \delta b_j \right) = b_j + \chi (2\epsilon_{ij} - \delta b_j)$$

onde  $\chi$  é a taxa de aprendizagem e  $\delta$  um outro parâmetro de regularização.

A equação para o erro de predição incluindo os novos parâmetros de vieses e regularização passa a ser

$$\varepsilon_{ij}^2 = \left( r_{ij} - \sum_{k=1}^f p_{ik} q_{kj} + b_i + b_j \right)^2 + \frac{\beta}{2} (\|p\|^2 + \|q\|^2) + \frac{\delta}{2} (b_i^2 + b_j^2)$$

Como esses componentes de viés tendem a ter um peso grande na predição das avaliações, uma boa precisão em sua modelagem é fundamental.

### 3.5 Adicionando Dinâmicas Temporais ao Modelo

Até então, os modelos apresentados são todos atemporais. No mundo real, a percepção e a popularidade de um item está mudando constantemente a medida que novos itens surgem. Da mesma forma, as preferências dos usuários evoluem, levando-os a redefinir seu gosto. Sendo assim, o sistema deve levar em consideração esses efeitos temporais no mapeamento da natureza dinâmica e variável das interações entre usuários e itens.

A abordagem de fatoração matricial se adapta bem à modelagem desses efeitos temporais, permitindo que tenhamos uma precisão ainda maior na predição das avaliações. Decompor as avaliações em termos distintos permite ao sistema tratar cada efeito temporal separadamente. Especificamente, podemos considerar os seguintes efeitos: os vieses dos usuários  $b_i(t)$  mudam com o tempo, assim como os vieses dos itens  $b_j(t)$  e as preferências dos usuários  $p_i(t)$ . Por outro lado, não é de se esperar uma significativa variação temporal nas características dos itens  $q_j(t)$ , já que os mesmos são estáticos por natureza.

Neste trabalho, demos uma atenção especial à variação temporal da relevância dos itens. Por exemplo, vídeos podem se tornar populares ou impopulares devido a eventos externos como, por exemplo, a aparição de um determinado ator em uma nova novela. Assim, tratamos os vieses de um item  $b_j(t)$  em nosso modelo de predição como uma função do tempo.

Koren [28] sugere dividir os vieses de um item em fragmentos baseados no tempo. Durante cada período de tempo em um fragmento tem-se a tradução de uma dada particularidade. A decisão de se dividir a linha do tempo em fragmentos deve balancear o objetivo de riqueza em detalhes (pequenos fragmentos) com a necessidade de um número mínimo de avaliações por

fragmento (fragmentos maiores).

$$\hat{e}(u_i, v_j, t) = \sum_{k=1}^f p_{ik} q_{kj} + b_i + b_j(t)$$

Um dia  $t$  estaria associado a um fragmento  $Bin(t)$  de forma que a particularidade do item seja dividida entre uma parte estacionária e uma que evolui com o tempo:

$$b_j(t) = b_j + b_{j,Bin(t)}$$

Em termos práticos,  $b_{j,Bin(t)} \approx MD = B(v \times s)$ .  $B$  contempla as variações temporais para os itens, onde  $v$  é o número de itens e  $s$  o número de fragmentos temporais. A  $j$ -ésima linha da matriz  $M$  é o vetor de características do item  $j$  e a  $b$ -ésima linha da matriz  $D$  é o vetor de características temporais do fragmento  $b$ .

Podemos extrair uma aproximação para o efeito temporal sobre o item desejado pela equação  $b_{j,Bin(t)} = m_j^T d_b = \sum_{k=1}^f m_{jk} d_{kb}$  onde  $t \subset b$ .

A equação para o erro de predição incluindo os novos parâmetros de vieses e regularização passa a ser

$$\varepsilon_{ij}^2(t) = \left( r_{ij}(t) - \sum_{k=1}^f p_{ik} q_{kj} + b_i + b_j + b_{j,Bin(t)} \right)^2 + \frac{\beta}{2} (\|p\|^2 + \|q\|^2) + \frac{\delta}{2} (b_i^2 + b_j^2 + b_{j,Bin(t)}^2)$$

A partir do erro calculado, utilizamos também o gradiente descendente estocástico para diferenciar a equação que contempla as dinâmicas temporais dos itens em relação às suas variáveis latentes e obter as equações de atualização

$$\begin{aligned} m'_{jk} &= m_{jk} + \theta \left( \frac{\partial \varepsilon_{ij}^2}{m_{jk}} - \rho m_{jk} \right) = m_{jk} + \theta (2\varepsilon_{ij} d_{kb} - \rho m_{jk}) \\ d'_{kb} &= d_{kb} + \theta \left( \frac{\partial \varepsilon_{ij}^2}{d_{kb}} - \rho d_{kb} \right) = d_{kb} + \theta (2\varepsilon_{ij} m_{jk} - \rho d_{kb}) \end{aligned}$$

onde  $\theta$  é a taxa de aprendizagem e  $\rho$  um outro parâmetro de regularização.

A fragmentação dos parâmetros funciona bem para mapearmos a

evolução temporal da relevância dos itens. No caso dos usuários, o desafio é maior. É necessário uma resolução detalhada dos vieses dos mesmos para detectar pequenos efeitos temporais. No entanto, é baixa a expectativa por um número suficiente de avaliações dos usuários para cada fragmento de tempo. Nesse caso, diferentes formas têm sido consideradas para parametrizar o comportamento temporal dos usuários [28, 29, 31].

### 3.6 O algoritmo final

Como veremos no próximo capítulo, o conjunto de dados utilizado em nossos experimentos possui algumas peculiaridades que precisaram ser levadas em consideração para que o algoritmo tivesse uma performance aceitável.

Uma das alterações feitas consistiu em definir um número mínimo  $e_{\min}$  de passos de treino para cada variável latente, assim como um número máximo  $e_{\max}$ . Estes dois limites são dinamicamente ajustados dependendo do número de variáveis latentes que estamos treinando. Isto é necessário porque caso contrário o algoritmo encerraria o treino logo após alguns poucos passos. Para as últimas variáveis, é necessário um número de passos de treino maior para garantir que houveram melhorias nas previsões (redução do *RMSE*).

A melhoria no *RMSE* é medida em cinco passos, ou seja, a diferença no *RMSE* para a variável latente  $k$  e o passo  $e$  é dada por:

$$\Delta_k^e = RMSE_k^{e-5} - RMSE_k^e$$

Um mínimo de 5 passos de treino é necessário para cada variável latente.

No caso de o número mínimo de passos de treino ser atingido e o *RMSE* parar de cair, o algoritmo encerra o cálculo daquela variável e parte para a seguinte.

Por fim, armazenamos os valores das variáveis latentes conforme as mesmas iam sendo calculadas para acelerar o cálculo das demais. O algoritmo de treino final é apresentado a seguir.

---

**Algoritmo 2 - Algoritmo final**

---

Entrada: Matriz de avaliações  $R$ , número de variáveis latentes  $f$  e número de segmentos temporais  $b$ .

```
1: initialize as matrizes  $P$  e  $Q$  pela primeira vez com um número fixo ou de forma randômica
2: initialize os vetores de vieses dos usuários  $b_i$  e itens  $b_j$  com um número fixo ou de forma randômica
3: initialize as matrizes  $M$  e  $D$  de dinâmicas temporais para os itens com um número fixo ou de forma randômica
4: para  $k = 1$  até  $f$  faça
5:   para  $e = 1$  até  $e_{\max}$  faça
6:     para  $\forall(u_i, v_j, r_{ij}) \in T$  faça
7:       calcule  $\varepsilon_{ij}$ .
8:       atualize  $p_{ik}$  e  $q_{kj}$ .
9:       atualize  $b_i$  e  $b_j$ 
10:      para  $t = 1$  até  $b$  faça
11:        atualize  $m_{jk}$  e  $d_{kt}$ 
12:      fim
13:      recalcule  $\sum_{(u_i, v_j, r_{ij})} \varepsilon_{ij}^2$  e o RMSE
14:      recalcule  $\Delta_k^e$ 
15:      se  $\Delta_k^e < 0.0001$  ou  $e > e_{\min}$ 
16:        interrompa
17:      fim
18:      armazene o valor da variável calculada
19:    fim
```

Saída: Matrizes  $P$  e  $Q$  com os vetores de variáveis latentes, vetores  $b_i$  e  $b_j$  com os vieses dos itens e usuários e matrizes  $M$  e  $D$  contemplando o efeito temporal nos vieses de um item.

---

Algoritmo 2 - Versão final com viés e dinâmicas temporais

## 4 Aplicação na Globo.com

Neste capítulo descrevemos os experimentos realizados para testar as abordagens apresentadas. Primeiro, descrevemos as principais características de nosso conjunto de dados. Em seguida, os resultados dos experimentos são listados e discutidos.

### 4.1 Conjunto de Dados

Em nossos experimentos utilizamos os dados do portal de vídeos da Globo.com [32]. Separamos os dados de consumo de vídeos nos catálogos do programa Auto Esporte [33] de janeiro a maio de 2011 com algo em torno de 77 mil registros de consumo de 396 vídeos por 3046 usuários, onde cada usuário viu pelo menos 18 vídeos.

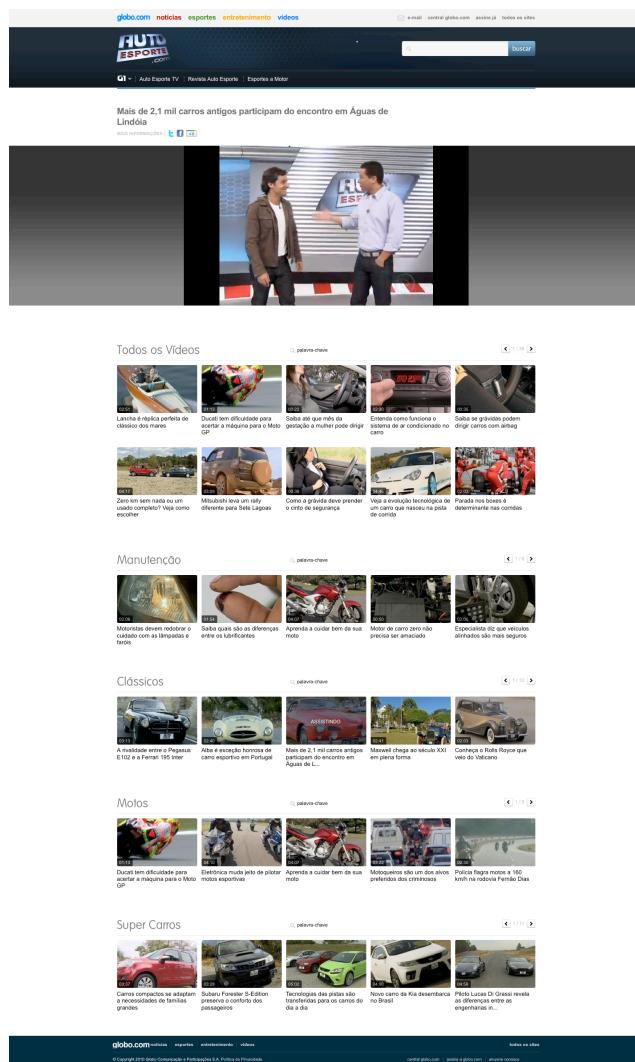


Figura 4.1 - Catálogo de vídeos do Auto Esporte

Os dados foram obtidos a partir dos servidores de borda da Globo.com responsáveis pela entrega dos arquivos de vídeo. A identificação do conteúdo destes arquivos é feita por um ID único que aparece no nome do arquivo.

[http://flashvideo.globo.com/h264/jornalismo/3//sptv/2011/08/09/EF\\_CGJ-RJ\\_T\\_1591109\\_mmp4.mp4](http://flashvideo.globo.com/h264/jornalismo/3//sptv/2011/08/09/EF_CGJ-RJ_T_1591109_mmp4.mp4)

Dado que a experiência de consumo no portal não é autenticada (ou logada), utilizamos o *cookie* do serviço Google Analytics (utilizado para contabilizar as métricas de acesso ao portal) [34] para a identificação dos usuários. Este *cookie* possui um parâmetro chamado *\_utma* que faz a identificação de um visitante único e demora cerca de dois anos para expirar. A forma utilizada para a identificação dos usuários nos remete a dois problemas. O primeiro é que tratamos o mesmo usuário acessando os vídeos de máquinas ou *browsers* diferentes como sendo o mesmo usuário. O segundo é que se, por ventura, o usuário limpar o conteúdo cacheado de seu browser, seu próximo acesso será identificado como o de um outro usuário. Isso introduz ruídos no conjunto de dados que prejudicam os resultados das experimentações.

Considerando as avaliações dos usuários, os sistemas de recomendação costumam funcionar com diferentes tipos de dados de entrada. O mais conveniente é contar com o *feedback* explícito dos usuários, com os usuários indicando explicitamente seus interesses pelos produtos. No nosso caso, estes dados não estão disponíveis. Assim, tivemos que inferir as preferências dos usuários a partir de um *feedback* implícito, que reflete indiretamente a opinião do usuário através da observação de seu comportamento. Dentre os diferentes tipos de *feedback* implícito temos o histórico de compras em um *site* de comércio eletrônico, o histórico de navegação, padrões de busca ou até os movimentos do *mouse*. Por exemplo, um usuário que comprou diversos livros de um mesmo autor provavelmente gosta daquele autor.

Construímos o conjunto de dados de entrada a partir do *feedback* implícito dos usuários usando os registros de visualização dos vídeos no portal. Cada registro de visualização de vídeo em nosso conjunto de dados consiste de uma tupla "data e hora de acesso | identificador do vídeo | identificador do usuário". Não temos nenhum dado sobre o conteúdo do vídeo e características do

usuários. Assim, construímos nossa matriz de avaliações com "1" se um usuário requisitou um vídeo (avaliação positiva) e "-" se ele não viu. Essa matriz é uma matriz esparsa onde, aproximadamente, 94% de seus dados são desconhecidos.

|       | V_ID1 | V_ID2 | V_ID3 | V_ID4 | V_ID5 | V_ID6 | V_ID7 | V_ID8 | V_ID9 | V_ID10 |
|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|--------|
| U_ID1 | 1     | -     | -     | 1     | -     | -     | -     | -     | 1     | -      |
| U_ID2 | 1     | -     | 1     | -     | 1     | 1     | 1     | 1     | -     | -      |
| U_ID3 | -     | 1     | -     | -     | -     | -     | 1     | -     | -     | 1      |
| U_ID4 | -     | 1     | -     | 1     | 1     | -     | 1     | -     | -     | 1      |
| U_ID5 | 1     | -     | 1     | -     | -     | -     | -     | -     | 1     | -      |

Tabela 4.1 - Matriz de visualizações de vídeos

Duas características desses dados implícitos devem ser observadas:

1. Não temos *feedback* negativo. Observando o comportamento dos usuários, podemos inferir que vídeos ele provavelmente gosta e, por isso, escolheu consumir. No entanto, não sabemos se um usuário viu um vídeo e não gostou. Por exemplo, um usuário que não assistiu um trecho de uma edição do programa, pode não tê-lo feito pelo simples fato de não saber da existência do mesmo ou porque o mesmo não estava disponível. Esta assimetria não existe quando estamos lidando com o *feedback* explícito dos usuários, quando os mesmos indicam claramente quais itens eles gostaram e quais não gostaram. Isto tem algumas implicações. Por exemplo, os sistemas que lidam com dados explícitos tendem a focar apenas na informação conhecida - os pares usuário-item para os quais conhecemos as avaliações. Assim, as relações usuário-item restantes, que geralmente constituem a maior parte dos dados, são tratadas como desconhecidas e omitidas da análise. No caso do processamento de dados implícitos, nos concentrar apenas nos dados conhecidos, o *feedback* positivo dos usuários, irá desfigurar a representação do perfil dos usuários. Isto significa que estamos interessados apenas em verificar se um usuário estaria interessado ou não em consumir um vídeo, e não se o mesmo gostou ou não do mesmo.
2. O *feedback* implícito é, por natureza, uma fonte ruidosa. Enquanto

rastreamos passivamente o comportamento dos usuários, podemos apenas supor quais são suas intenções e reais motivos de consumo. Por exemplo, podemos ver o comportamento de consumo para um indivíduo mas isso não necessariamente quer dizer que o mesmo gostou do conteúdo. Ele pode ter começado a consumir e trocado para uma outra aba do navegador ou mesmo saído de perto do ambiente de consumo. Ele pode também ter fechado o *player* de vídeo demonstrando total desinteresse pelo conteúdo que isso não será levado em consideração, dado que estamos extraíndo este comportamento dos *logs* de acesso dos servidores de entrega de vídeos e que só temos a informação de quais arquivos de vídeo foram requisitados.

Separamos o conjunto de dados em dois arquivos, um contendo os dados de treino e outro os dados de teste contendo 5 visualizações de vídeo por usuário dentro do mesmo período de observação. Os conjuntos contendo os dados de treino e de teste são disjuntos. Um exemplo contendo algumas linhas do arquivo com os dados de treino é mostrado a seguir.

---

```
mac179045:auto_esporte brunofms$ tail -10 auto_esporte.2011.data
data e hora de acesso | identificador do vídeo | identificador do usuário
25/May/2011:23:31:46 -0300|1515732|100629313.946488314.1304379721.1306179211.1306369776.4
25/May/2011:23:16:22 -0300|1487030|100629313.452412031.1305484947.1305484947.1306349282.2
25/May/2011:23:41:19 -0300|1515726|156749805.1238557690.1306377303.1306377303.1306377303.1
25/May/2011:23:44:09 -0300|1515730|100629313.1215930022.1305570691.1306374138.1306376677.35
25/May/2011:23:49:05 -0300|1487028|156749805.1238557690.1306377303.1306377303.1306377303.1
25/May/2011:23:54:18 -0300|1486234|156749805.1428959616.1306374823.1306374823.1306374823.1
25/May/2011:23:54:51 -0300|1515726|100629313.1035999351.1300656444.1306361381.1306375927.20
25/May/2011:23:54:45 -0300|1481372|156749805.1238557690.1306377303.1306377303.1306377303.1
25/May/2011:23:56:31 -0300|1481373|156749805.1238557690.1306377303.1306377303.1306377303.1
25/May/2011:23:57:03 -0300|1515726|156749805.1242565478.1306182012.1306209390.1306292155.3
```

---

Tabela 4.2 - Formatação dos dados de entrada

Nosso objetivo é encontrar um vetor  $p_i \in R^f$  para cada usuário  $i$ , e um vetor  $q_j \in R^f$  para cada vídeo  $j$  que irá caracterizar a intenção dos usuários a consumir vídeos. Em outras palavras, as preferências são denotadas como o produto interno  $\hat{e}(u_i, v_j) = p_i^T q_j$ . Estes vetores visam mapear usuários e vídeos num espaço de variáveis latentes onde podem ser diretamente comparados de forma a prover as recomendações.

Nesses experimentos, separamos nosso conjunto de dados em três subconjuntos: o de treino com os dados de visualização usados para treinar os modelos de predição, o de validação para validar os parâmetros usados nos algoritmos de aprendizado destes modelos e um de teste, como usuários e vídeos sem registros de visualizações nos dois anteriores, usado para verificar e comparar a precisão destes modelos. Resumidamente, temos:

- dados de treino (53.167 visualizações)
- dados de validação (12.184 visualizações)
- dados de teste (12.184 visualizações), usado na comparação dos modelos

## 4.2 Métrica de Avaliação

A qualidade de um sistema de recomendação pode ser decidida a partir dos resultados das avaliações. O tipo de métrica utilizada depende do tipo de aplicação de filtragem colaborativa. De acordo com [61], as métricas de avaliação podem ser classificadas em diversas categorias. Para examinar a performance dos experimentos aqui propostos, focamos na categoria de precisão de predição e adotamos a raiz do erro médio quadrático (*RMSE*):

$$RMSE = \sqrt{\frac{1}{n} \sum_{\{i,j\}} (\hat{e}(u_i, v_j) - r_{ij})^2}$$

onde  $n$  é o número total de observações,  $\hat{e}(u_i, v_j)$  é a predição de intenção do usuário  $i$  em visualizar o vídeo  $j$ , e  $r_{ij}$  é a observação de fato. O *RMSE* amplifica a contribuição dos erros absolutos entre as predições e as observações de consumo.

Além de ser bem conhecido e um número simples de se calcular, o *RMSE* tem uma outra propriedade vantajosa que é o fato dele amplificar erros gerados por falso positivos ou falso negativos. E qualquer sistema de recomendação deve saber lidar bem com isso. No entanto, a precisão da predição não resolve diversos outros aspectos importantes de se considerar ao fazer, ou receber, uma recomendação [68].

## 4.3 Experimentos

O método aqui proposto possui diversos parâmetros a serem ajustados. Dentro destes parâmetros, experimentamos diferentes números de variáveis latentes, diferentes taxas de aprendizado para as variáveis latentes e as constantes de vieses, diferentes parâmetros de regularização. Também testamos diferentes modelos de predição.

Visando o desenvolvimento de um protótipo rápido para os testes, a maior parte do código foi escrita em *Python*, usando a biblioteca *numpy* para lidar com objetos tipo *array* muito grandes e para os cálculos mais pesados, dentre outras funcionalidades.

Inicializamos todos os valores da matrizes de preferências  $P$  e de características  $Q$  randomicamente seguindo um modelo de distribuição normal. Uma variável latente é treinada até que a melhoria no *RMSE* seja inferior a 0.0001, ou que um número mínimo de 20 passos de treino e máximo de 50 seja feito no conjunto de dados.

### Quantidade de Passos de Treino

Neste experimento, visamos encontrar uma quantidade de passos de treino razoável para o conjunto de dados em questão. Usamos uma taxa de aprendizado de 0.001 e 20 variáveis latentes. Vale notar que a partir do vigésimo quarto passo passamos a ter uma condição de *overfitting*.

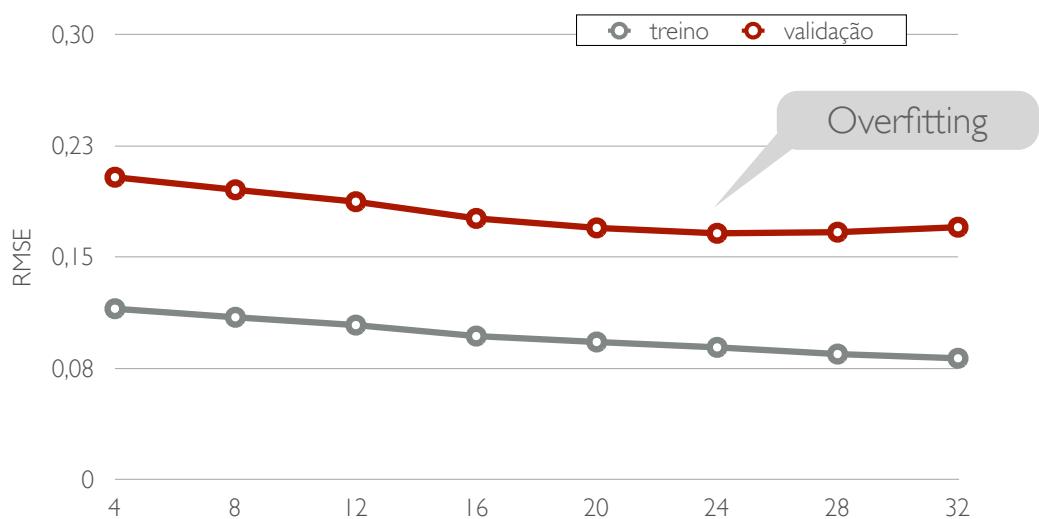


Figura 4.2 - Quantidade de passos de treino vs. *RMSE*

|           | 16 passos | 20 passos | 24 passos | 28 passos | 32 passos |
|-----------|-----------|-----------|-----------|-----------|-----------|
| Treino    | 0.0965    | 0.0926    | 0.0889    | 0.0844    | 0.0816    |
| Validação | 0.1759    | 0.1695    | 0.1658    | 0.1666    | 0.1700    |

Tabela 4.3 - Quantidade de passos de treino vs. RMSE

## Número de Variáveis Latentes

A escolha da quantidade de variáveis latentes ótima  $f$  é crítica para uma boa predição das intenções de visualização. Estamos interessados em um valor de  $f$  que seja grande o bastante para capturar todas as características das interações presentes na matriz e pequena para comportar um tempo de construção do modelo que seja razoável. Encontramos um valor razoável para  $f$  através de sucessivas tentativas com diferentes valores experimentais.

Para este experimento, ajustamos o número mínimo de passos de treino em 20 e o máximo em 50, e usamos uma taxa de aprendizado de 0.001.

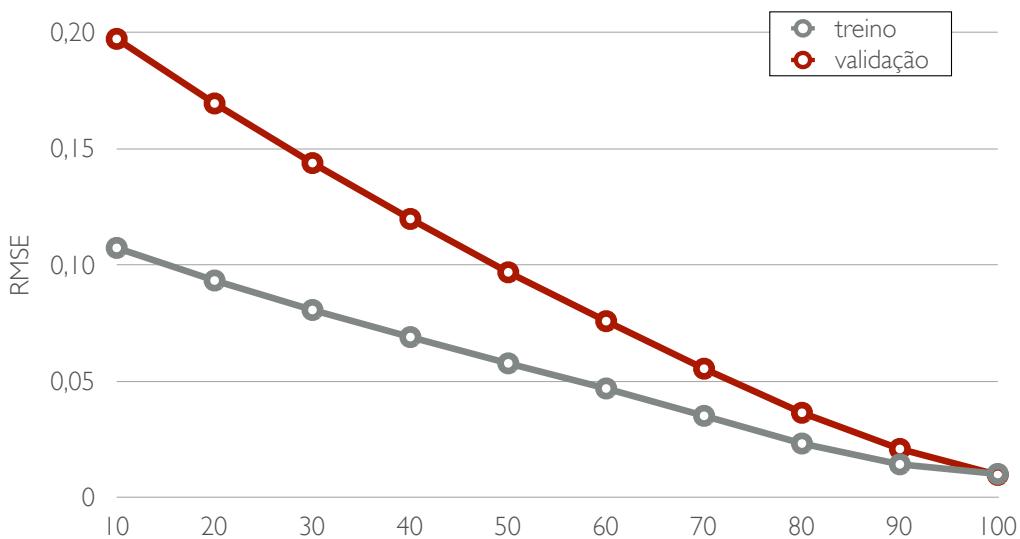


Figura 4.3 - Quantidade de variáveis latentes vs. RMSE

|        | 10 variáveis | 30 variáveis | 50 variáveis | 70 variáveis | 90 variáveis |
|--------|--------------|--------------|--------------|--------------|--------------|
| Treino | 0.1072       | 0.0805       | 0.0576       | 0.0350       | 0.0142       |
| Teste  | 0.1971       | 0.1437       | 0.0967       | 0.0552       | 0.0208       |

Tabela 4.4 - Quantidade de variáveis latentes vs. RMSE

Pegando duas das variáveis latentes resultantes do processo de fatoração matricial, vemos na Figura 4.4 que os vídeos ficam localizado no lugar apropriado com base nas duas dimensões destes vetores latentes. Estes lugares no

gráficos revelam o conteúdo dos filmes que por ali se localizam.

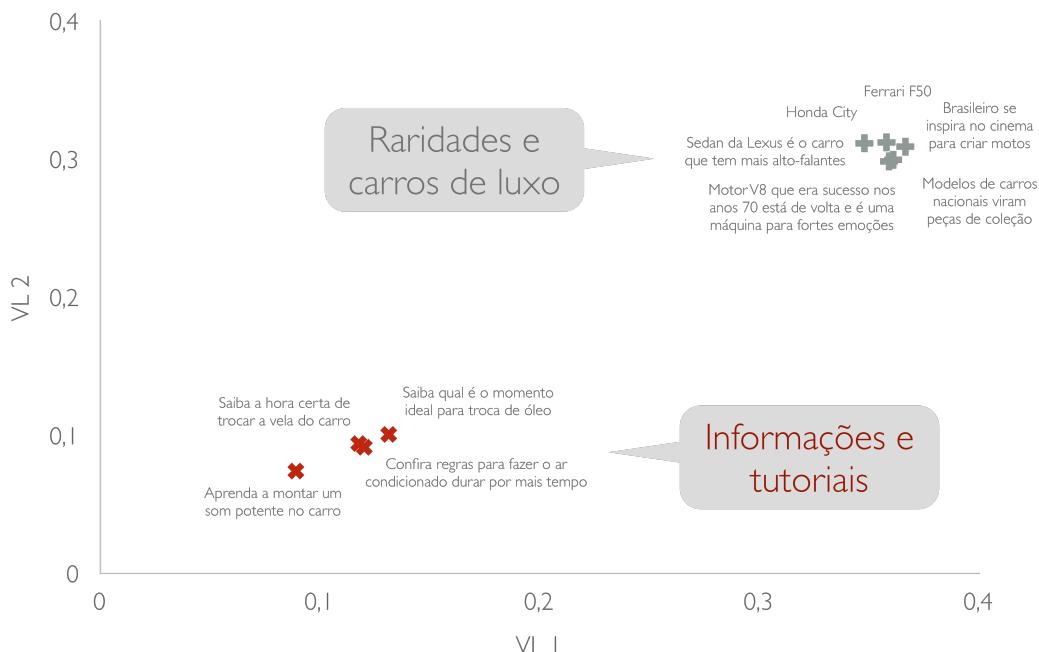


Figura 4.4 - Variável latente 1 vs. variável latente 2

Neste exemplo, podemos identificar claramente 2 categorias: vídeos referentes a carros classificados como raridades ou de luxo, e vídeos relacionados a mecânica de automóveis com informações e tutoriais.

### Taxa de Aprendizado

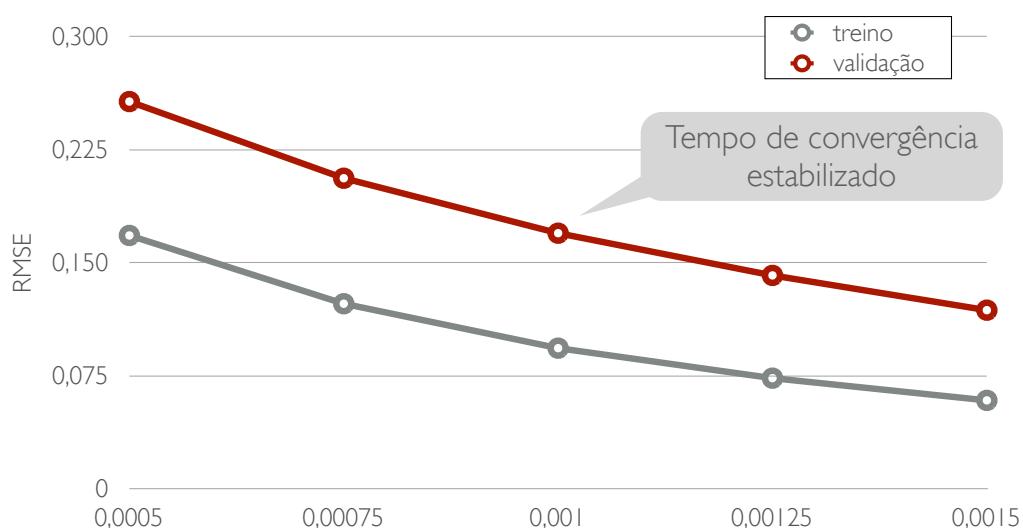


Figura 4.5 - Efeito da taxa de aprendizado

|           | $L = 0.0005$ | $L = 0.00075$ | $L = 0.001$ | $L = 0.00125$ | $L = 0.0015$ |
|-----------|--------------|---------------|-------------|---------------|--------------|
| Treino    | 0.1679       | 0.1226        | 0.0932      | 0.0733        | 0.0585       |
| Validação | 0.2567       | 0.2059        | 0.1694      | 0.1414        | 0.1184       |

Tabela 4.5 - Efeito da taxa de aprendizado

Neste experimento, variamos a taxa de aprendizado com valores em torno de 0,001. Todas as observações foram feitas usando 20 variáveis latentes e um fator de regularização com valor fixo em 0,02.

A Figura 4.5 mostra os valores de *RMSE* para os dados de treino e validação. Quanto menor a taxa de aprendizado, maior o tempo de convergência durante o treino. No entanto, aumentar muito a taxa de aprendizado, pode nos levar rapidamente a uma condição de *overfitting*.

### Fator de Regularização

Neste experimento, variamos a taxa de aprendizado com valores entre 0,015 e 0,025. Todas as observações foram feitas usando 20 variáveis latentes. A taxa de aprendizagem utilizada foi de 0,001. O gráfico a seguir mostra os resultados.

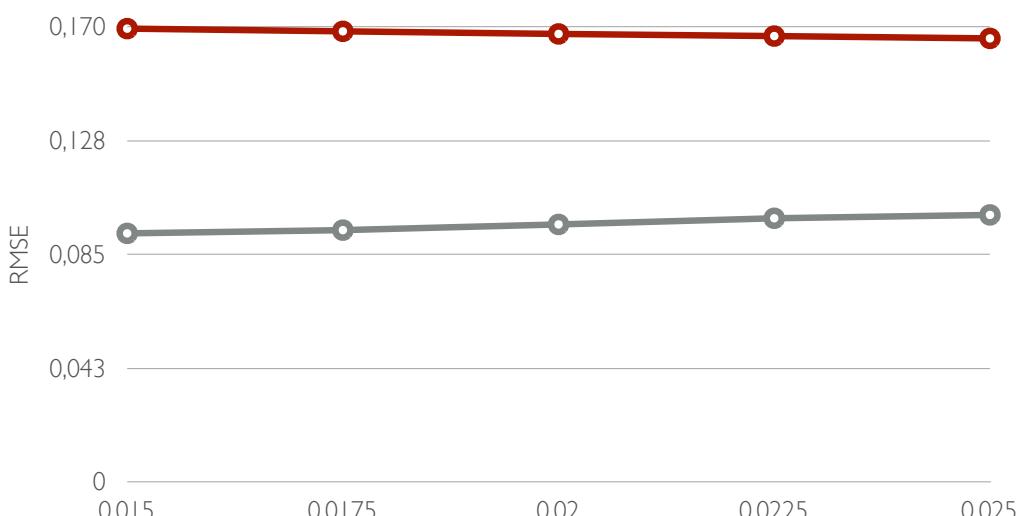


Figura 4.6 - Efeito da regularização

|           | $R = 0.015$ | $R = 0.0175$ | $R = 0.02$ | $R = 0.0225$ | $R = 0.025$ |
|-----------|-------------|--------------|------------|--------------|-------------|
| Treino    | 0.0928      | 0.0940       | 0.0961     | 0.0984       | 0.0997      |
| Validação | 0.1692      | 0.1682       | 0.1673     | 0.1665       | 0.1656      |

Tabela 4.6 - Efeito da regularização

Os resultados mostram claramente que baixos valores para o parâmetro de regularização resultam, mais uma vez, numa condição de *overfitting* nos dados de treino e que o *RMSE* diminui a medida que aumentamos seu valor.

## Comparação dos Modelos

Tentamos diferentes implementações e parametrizações de fatoração matricial. A Figura 4.7 mostra como diferentes modelos e quantidades de variáveis latentes, assim como a evolução dos modelos - fatoração pura, adição de vieses dos vídeos e usuários, adição dos efeitos temporais - afetam o *RMSE*.

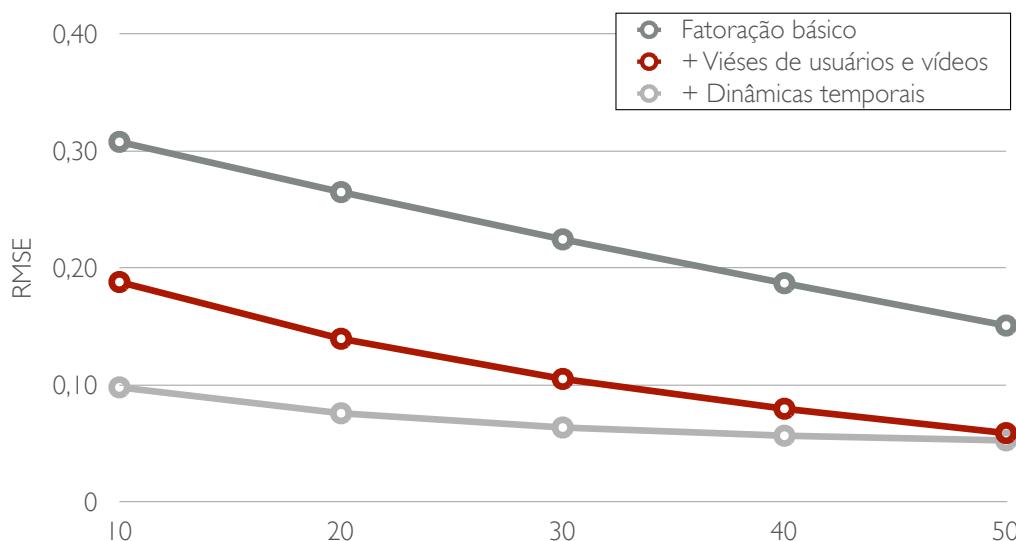


Figura 4.7 - Comparação dos modelos de fatoração

|                     | 10 variáveis | 20 variáveis | 30 variáveis | 40 variáveis | 50 variáveis |
|---------------------|--------------|--------------|--------------|--------------|--------------|
| fatoração simples   | 0.3076       | 0.2647       | 0.2242       | 0.1868       | 0.1506       |
| + viés              | 0.1878       | 0.1393       | 0.1049       | 0.0794       | 0.0586       |
| + efeitos temporais | 0.0978       | 0.0756       | 0.0634       | 0.0564       | 0.0524       |

Tabela 4.7 - Comparação dos modelos de fatoração

Um dos principais desafios destes experimentos foi encontrar uma combinação de parâmetros que desempenhasse bem em cada caso. Esta tarefa nos consumiu grande parte do tempo do projeto.

Para a construção dos nossos preditores ficamos com modelos de fatoração matricial incremental com regularização com uma taxa de aprendizado

de 0,001 e fator de regularização em 0,02. Na inclusão dos vieses dos itens (vídeos) e usuários, usamos uma taxa de aprendizado de 0,0001 com fator de regularização de 0,022 nas equações de atualização. Na adição das dinâmicas temporais dos itens, usamos uma taxa de aprendizado de 0,00003 com fator de regularização em 0,02 nas equações de atualização.

Dentro do período de tempo no qual os registros foram capturados, extraímos 22 fragmentos temporais, ou seja, 1 fragmento para cada semana. Isso, para poder entender o impacto que as dinâmicas temporais desses acessos têm no resultado final das previsões.

Os resultados da Figura 4.7 mostram a evolução do *RMSE* para cada um dos modelos individualmente. Vemos que a precisão melhora à medida em que aumentamos a dimensão (denotada pela quantidade de variáveis latentes) do modelo. Além disso, pode-se observar que, quanto mais refinado é o modelo, com entradas de dados oriundas de outras observações, melhor o resultado.

#### 4.4 Recomendando Vídeos

Para recomendar novos vídeos para os usuários, inicialmente usamos os modelos apresentados para capturar as relações latentes entre os usuários e os vídeos e então computar a previsão de intenção de um usuário assistir a um dado vídeo. Em seguida, usamos estas previsões para gerar uma lista com os  $N$  vídeos a serem recomendados para os usuários.

Começamos com a matriz de avaliações  $R$ , a qual é bastante esparsa. Para capturar os relacionamentos latentes removemos esta esparsidade extrapolando as avaliações dadas pelos usuários para alguns itens para o resto da matriz. Para isso, decomponemos a matriz  $R$  nas matrizes  $P$  e  $Q$ , indicando o peso das interações usuário-vídeo na previsão das intenções, nos vetores  $b_i$  e  $b_j$  com os vieses de usuários e itens, respectivamente e nas matrizes  $M$  e  $D$ , com o peso dos efeitos temporais nas intenções de consumo e usamos estes componentes resultantes para computar a previsão de intenção de qualquer usuário  $i$  ver um vídeo  $j$ :

$$r'_{ij}(t) = \sum_{k=1}^f p_{ik} + q_{kj} + b_i + b_j + b_{j,Bin(t)}$$

Para recomendar vídeos para um dado usuário  $i$ , ordenamos as previsões de intenção de consumo de vídeos para aquele usuário, isto é, os elementos da linha  $i$  da matriz  $R$ . Os vídeos cujas previsões de intenção de visualização estão mais próximas de 1 e que ainda não tiverem sido consumidos pelo usuário em questão serão os recomendados.

---

```
User 100629313.2038689089.1296840090.1299264125.1299717968.27:  
(0.0002619, 'Veja as diferenças dos barulhos de cada motor', '2011-01-30')  
(0.0002840, 'Objetos soltos no carro podem causar graves acidentes', '2010-12-19')  
(0.0004117, 'Chave, cartão ou controle remoto?', '2009-01-18')  
(0.0005802, 'Veja o que levar no kit de emergência para o seu carro', '2010-10-31')  
(0.0007359, 'Rodas e aros', '2009-07-12')
```

---

Tabela 4.8 - Exemplo de recomendações para um usuário

## 5 Trabalhos Relacionados

### Filtragem Colaborativa

O processo de modelagem da informação em sistemas de recomendação envolve diversas áreas de conhecimento. Algumas dessas áreas, por exemplo, estudam a variabilidade nas avaliações dos usuários [48], a evolução nas relações entre os usuários de um conjunto de dados [49], informações fornecidas por usuários externos e especialistas [50], dimensões temporais [28, 51, 52] e o uso de informações espaço-temporais [53].

Um problema típico na pesquisa destes tipos de sistema é a impossibilidade de um usuário avaliar todos os itens disponíveis. Como descrito anteriormente, a consequência disso é que a matriz que relaciona os usuários aos itens acaba ficando com vários campos em branco. Algumas propostas tem sido consideradas na predição destas avaliações [54].

Nos casos de filtragem colaborativa a partir de dados implícitos a informação disponível corresponde às atividades dos usuários que podem ser rastreadas (por exemplo, a compra ou o *download* de um produto). Nestes casos é necessário incluir algum passo de pré-processamento dos dados, os quais tendem a ser binários (presença ou ausência de uma atividade, etc). Seguindo nesta linha, [55] sugere transformar informações temporais sobre o consumo de produtos em pseudo-avaliações. Outros [56, 57] usam informações sobre o quanto frequente cada item é consumido pelos usuários para criar avaliações que possam ser utilizadas como dados de entrada destes sistemas.

[60] considera crucial a utilização dos dados desconhecidos nas observações de comportamento dos usuários na utilização do *feedback* implícito dos usuários durante a montagem da matriz de avaliações, onde a maior parte das avaliações negativas tendem a ser encontradas. Dessa forma, as equações de atualização deveriam considerar todos os pares usuário-item, ao invés de apenas aqueles realmente observados. Este número de termos, que em alguns conjuntos de dados atingem a ordem de bilhões, impede o uso de técnicas de aprendizado com o gradiente descendente estocástico, o qual é amplamente utilizado no caso

de conjunto de dados com *feedback* explícito. Também propõe que sejam utilizadas outras métricas de avaliação que levem em conta a disponibilidade de um item, a competição de outros itens com relação ao primeiro, além de *feedbacks* repetidos. Por fim, compara um modelo de fatoração matricial com o resultado das recomendações geradas por algoritmos base que ordenam os vídeos mais populares dentro de um universo definido e algoritmos de vizinhança item-item.

Algumas abordagens sugerem o aprendizado do espaço latente na presença de dados do usuário, incorporando à equação de predição variáveis demográficas como o sexo e a idade dos mesmos [43]. Outras tentam entender o contexto atual dos usuários de forma a prover recomendações que se adéquam a situação atual dos usuários [44].

Herlocker, Konstan e Riedl [61] sugerem que boas recomendações devem vir acompanhadas com explicações, uma breve descrição com o por que de um produto em específico ter sido recomendado para um usuário. Isto ajuda a melhorar a confiança que os usuários têm no sistemas e a habilidade dos mesmos em colocar as recomendações na perspectiva correta. Além disso, auxilia na investigação de problemas no sistema, reduzindo a incidência de comportamentos inesperados no mesmo.

Durante os últimos anos, temos visto avanços [39, 40] na utilização de relações sociais juntamente com mecanismos de filtragem colaborativa na construção desses sistemas. Diferentes tecnologias baseadas em mineração de dados, aprendizado de máquina, inteligência artificial, entre outras, convergem neste contexto para melhor explorar as relações entre os usuários dos sistemas. Em [40] os autores sugerem que levar em consideração o contexto social e de familiaridade entre os usuários em suas avaliações auxilia na tomada de decisão e, consequentemente, em recomendações com mais qualidade. Em [39] os autores propõem um modelo que combina os laços sociais e as avaliações para aumentar a performance do sistema de recomendação dentro de uma rede social de avaliações, chegando a resultados interessantes.

Shepitsen [58] emprega um algoritmo de personalização para

recomendação em folksonomia baseado numa clusterização hierárquica de *tags*. Outros trabalhos focam em técnicas de passeio aleatório, sendo utilizados em diferentes áreas e se tornando casos de sucesso quando levadas para o campo das recomendações sociais [59].

Golbeck e Hendler [45] endereçaram o problema com uma abordagem baseada na confiança que uns usuários tem nos outros considerando de seus gostos. Eles utilizaram as declarações de confiança explicitadas no *site* de recomendações de filmes *FilmTrust* [46]. O *site* solicita que os usuários avaliem o nível de confiança que eles tem em outros usuários, indicando o gosto de quem eles pretendem seguir e o gosto de quem não seguir. A conclusão é que a confiança que uns usuários tem pelos outros é de grande valor para a qualidade das recomendações.

## Otimizações de Performance

Existe uma vasta gama de trabalhos relacionados a otimização em problemas de recomendação. Uma das primeiras abordagens que explorou a fatoração matricial para filtragem colaborativa foi [10]. Neste artigo, os autores observam que decompor uma matriz qualquer no produto de duas matrizes é uma tarefa de alto custo computacional e propõem o método de mínimos quadrados alternantes (*ALS*, do inglês *Alternating Least Square*).

Algoritmos baseados em *ALS* trabalham iterativamente e cada iteração consiste de dois estágios: no primeiro estágio a matriz  $P$  tem seus valores fixados e o problema de minimização do erro de predição  $\epsilon_{ij}^2$  é resolvido com relação à matriz  $Q$ . No segundo estágio, ocorre o contrário. Em ambos os estágios, o problema pode ser reformulado e resolvido como um problema de mínimos quadrados (*LS*, do inglês *Least Square*) [10, 3]. Essas técnicas de *ALS* tem se mostrado altamente paralelizáveis. Assim, o crescimento contínuo das plataformas de computação paralela, como o *Hadoop* [42], tendem a ser de grande utilidade para rodar estes algoritmos.

Singh e Gordon [47] propõem uma forma de resolver múltiplas tarefas de fatoração matricial simultaneamente. Descrevemos resumidamente a idéia: suponhamos que temos duas matrizes (podendo estar com valores faltando), uma

para a relação usuário-item, descrevendo como os usuários avaliam os itens ( $R$ ), e outra para a relação item-data de consumo ( $S$ ). Podemos fatorar ambas as matrizes separadamente, e extrair o vetor de características dos itens decompondo  $R$ , e outro diferente extraindo  $S$ . A idéia é compartilhar os dois vetores de características dos itens e formular um novo problema de otimização combinando linearmente os dois vetores na equação do erro de predição a ser minimizada. A idéia poderia ser facilmente generalizada para fatorar qualquer quantidade de matrizes de forma simultânea.

## 6 Conclusão e Trabalho Futuro

Hoje em dia temos uma quantidade enorme de informação disponível na internet. Os sistemas de busca nos ~~ajudam~~ razoavelmente a encontrar as informações que queremos. Mas o ideal mesmo é ter algo nos sendo oferecido diretamente sem termos a necessidade de procurar o conteúdo em questão. Inclusive, desde uma simples lista com os links mais acessados nos sites de notícias até recomendações personalizadas que recebemos nos principais sites de comércio eletrônico, estamos bastante acostumados a receber recomendações na Web.

Recomendações podem ser geradas ~~a partir de~~ uma grande diversidade de algoritmos. Enquanto modelos de filtragem colaborativa baseados em itens e usuários são simples e intuitivos, técnicas de fatoração matricial são geralmente mais eficientes pelo fato de conseguirem demonstrar os fatores latentes que não costumam estar explicitados nas relações entre os usuários e os itens.

Nesta dissertação apresentamos algumas abordagens de fatoração matricial para o problema de recomendação de vídeos. Para isso, utilizamos os registros de vídeos vistos nos catálogos da Globo.com como dados implícitos de um modelo incremental de fatoração matricial visando encontrar outros vídeos que pudessem ser oferecidos para usuários com gostos parecidos.

Diferente dos casos onde temos um retorno explícito sobre a preferência dos usuários, nosso algoritmo de fatoração latente deveria utilizar todos os pares usuário-vídeo como entrada. ~~Incluindo~~ aqueles que não estão relacionados a nenhuma observação (indicado como um zero em nosso modelo). Isto é bastante crítico, dado que a forma como foi implementado faz com que as preferências dos usuários sigam uma tendência positivista com relação a todos os itens, não refletindo verdadeiramente seu perfil.

No entanto, usar todos os pares usuário-vídeo como entrada do sistema nos levaria a sérios problemas de escalabilidade. A quantidade de pares tende a ser muito maior que o conjunto de entrada trabalhado, dado que a quantidade de vídeos que um usuário típico assiste é apenas uma pequena fração do total de vídeos em questão. Endereçamos esta questão explorando a estrutura algébrica do

modelo, chegando a um algoritmo de implementação simples e com um tempo de execução que escala linearmente com o tamanho da entrada (dados de visualizações e quantidade de variáveis latentes).

Vimos que um dos principais pontos negativos do método utilizado é a quantidade de parâmetros que precisamos ajustar para chegar ao resultado almejado. Encontrar o conjunto de valores ideal para os parâmetros em questão é uma tarefa que nos consumiu grande parte do tempo do projeto. Os experimentos mostraram também que o *overfitting* causado pela escolha errada destes parâmetros tende a ser um sério problema. Valeria investir mais tempo analisando outras técnicas de regularização que pudessem minimizar o efeito deste problema.

Também mostramos como muitas das variações observadas na preferência dos usuários por alguns vídeos se deve a características dos próprios usuários ou vídeos independentemente de qualquer interação entre eles através da adição de um componente de *bias* ao modelo de aprendizado. Além disso, procuramos entender como a inclusão da evolução temporal da popularidade dos vídeos tende a melhorar significativamente o resultado das previsões.

Dado que o projeto de um sistema de filtragem colaborativa depende fortemente dos dados que alimentam o mesmo, como o tamanho do conjunto de dados, seus requisitos de complexidade, entre outros, enxergamos como uma possível melhoria na continuidade deste trabalho a inclusão de outras fontes de informação sobre o conteúdo e as características dos usuários em questão. Isso poderia endereçar o problema de *cold-start*, onde queremos prover recomendações para usuários sem um histórico prévio de visualizações de vídeo.

Uma outra proposta de continuidade deste trabalho é a paralelização do passo de aprendizado do algoritmo de fatoração. Dependendo do tamanho do conjunto de dados de entrada, a construção do modelo de recomendação pode tomar muito tempo. Alguns estudos [36, 37] relatam ganhos de performance expressivos a partir de algoritmos de aprendizagem paralelizáveis como o *Alternating Least Squares*.

Por fim, achamos que a adição de outras técnicas de predição ao sistema, em questão, como por exemplo algoritmos de vizinhança ou modelos gráficos tende a melhorar a precisão das recomendações. A combinação de diferentes

modelos de filtragem colaborativa se mostrou bastante eficiente durante a competição *Netflix Prize*.

## 7 Referencias Bibliográficas

- [1] G. Linden, B. Smith, and J. York. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing*, 7(1):76-80, 2003.
- [2] J. Bennet and S. Lanning. The Netflix Prize. *Proceedings of the KDD Cup 2007*. ACM Press, 2007.
- [3] Y. Koren, R. Bell, C. Volinsky. Matrix Factorization Techniques for Recommender Systems. *Computer*. 42(8):30-38, 2009.
- [4] Anne Yun-An Chen and Dennis McLeod, Collaborative Filtering for Information Recommendation Systems.
- [5] Schafer, J. B., Konstan, J. and Riedl, J.: 1999, ‘Recommender Systems in E-Commerce’. In: *EC '99: Proceedings of the First ACM Conference on Electronic Commerce*, Denver, CO, pp. 158-166.
- [6] J. Ben Schafer, Nathaniel Good, and Joseph Konstan et. al. Combining collaborative filtering with personal agents for better recommendations. In *Proceedings of the 1999 National Conference of the American Association of Artificial Intelligence*, pages 439–436, 1999.
- [7] Resnick, P., Varian, H., Recommender Systems. *Communications of the ACM*, 40, 3, (1997), 56-58.
- [8] Terveen, L. and Hill, W: 2001, ‘Human-Computer Collaboration in Recommender Systems’. In: J. Carroll (ed.): *Human Computer Interaction in the New Millennium*. New York: Addison-Wesley.
- [9] Tim Westergren, The Music Genome Project, 1999, <http://www.pandora.com/mgp.shtml>, Último acesso em Julho de 2011.
- [10] R. Bell and Y. Koren, Scalable Collaborative Filtering with Jointly Derived Neighborhood Interpolation Weights, *IEEE International Conference on Data Mining*, IEEE, 2007.
- [11] Belkin N. J., Croft W. B., Information filtering and information retrieval:

Two sides of the same coin? // Communications of the ACM 35, 29–39, December 1992.

- [12] Lang K., Newsweeder: Learning to filter news, In Proceedings of the 12th International Conference on Machine Learning, Lake Tahoe, CA, pp. 331-339, 1995.
- [13] Adomavicius G., Tuzhilin A. “Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions”, IEEE Transactions on Knowledge and Data Engineering 17, 734–749, June 2005.
- [14] Billsus D., Pazzani M., “User Modeling for Adaptive News Access”. User-Modeling and User-Adapted Interaction 10(2-3), 147-180, 2000.
- [15] Netflix, <http://www.netflix.com>, Último acesso em Julho de 2011.
- [16] MovieLens, <http://www.movielens.umn.edu>, Último acesso em Julho de 2011.
- [17] Amazon.com, <http://www.amazon.com>, Último acesso em Julho de 2011.
- [18] Funk, Simon. Netflix Update: Try This At Home. <http://sifter.org/~simon/journal/20061211.html>, Último acesso em Julho de 2011.
- [19] Netflix Prize, <http://www.netflixprize.com>, Último acesso em Julho de 2011.
- [20] Herve Abdi. Singular value decomposition (svd) and generalized singular value decomposition (gsvd). In Neil Salkind, editor, Encyclopedia of Measurement and Statistics. Thousands Oaks, CA, 2007.
- [21] Miklo Kurucz, Andras A. Benczur, and Kroly Csalogny. Methods for large scale svd with missing values. In Proceedings of the KDD Cup 2007, pages 54–58. ACM Press, 2007.
- [22] David Bau III and Lloyd N. Trefethen. Numerical linear algebra. Society for Industrial and Applied Mathematics, 1997.
- [23] Michael W. Berry. Large scale sparse singular value computations. Technical report, Department of Computer Science, University of Tennessee, 107

Ayres Hall Knoxville TN, 37996-1301, 2002.

- [24] Carl Eckart and Gale Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218, 1936.
- [25] Y. Koren, “Factorization Meets the Neighborhood: A Multifaceted Collaborative Filtering Model,” Proc. 14th ACM SIGKDD Int’l Conf. Knowledge Discovery and Data Mining, ACM Press, 2008, pp. 426-434.
- [26] A. Paterek, “Improving Regularized Singular Value De- composition for Collaborative Filtering,” Proc. KDD Cup and Workshop, ACM Press, 2007, pp. 39-42.
- [27] G. Takács et al., “Major Components of the Gravity Recom- mendation System,” SIGKDD Explorations, vol. 9, 2007, pp. 80-84.
- [28] Y. Koren, “Collaborative Filtering with Temporal Dynamics,” Proc. 15th ACM SIGKDD Int’l Conf. Knowledge Discovery and Data Mining (KDD 09), ACM Press, 2009, pp. 447-455.
- [29] A. Tsymbal. The problem of concept drift: Definitions and related work. Technical Report TCD-CS-2004-15, Trinity College Dublin, 2004.
- [30] Antonio J. Conejo, Enrique Castillo, Roberto Minguez, and Raquel Garcia-Bertrand. Decomposition Techniques in Mathematical Programming: Engineering and Science Applications. Springer, 2006.
- [31] Y. Ding and X. Li. Time weight collaborative filtering. *Proc. 14th ACM international conference on Information and knowledge management (CIKM’04)*, pp. 485–492, 2004.
- [32] globo.com, <http://www.globo.com>, Último acesso em Agosto de 2011.
- [33] Auto Esporte, <http://g1.globo.com/videos/autoesporte/>, Último acesso em Agosto de 2011.
- [34] Cookies & Google Analytics, <http://code.google.com/apis/analytics/docs/concepts/gaConceptsCookies.html>,

Último acesso em Agosto de 2011.

[35] Pandora Internet Radio, <http://www.pandora.com>, Último acesso em Agosto de 2011.

[36] H. Kim and H. Park. Non-negative matrix factorization based on alternating non-negativity constrained least squares and active set method. Technical report, Technical Report GT-CSE-07-01, College of Computing, Georgia Institute of Technology, 2007.

[37] Jingu Kim, Haesun Park, "Toward Faster Nonnegative Matrix Factorization: A New Algorithm and Comparisons," icdm, pp.353-362, 2008 Eighth IEEE International Conference on Data Mining, 2008.

[38] I. Pilászy and D. Tikk, ‘Recommending new movies: even a few ratings are more valuable than metadata’, in Proc. of the third ACM conf. on Recommender systems, pp. 93–100, New York, New York, USA, (2009). ACM.

[39] A. Said, E. De Luca, and S. Albayrak, “How social relations affect user similarities,” in Proceedings Of The 2010 Workshop On Social Recommender Systems, 2010, pp. 1–4.

[40] P. Bonhard and M. Sasse, “knowing me, knowing youusing profiles and social networking to improve recommender sys- tems,” Bt Technology Journal, vol. 24, no. 3, pp. 84–98, 2006.

[41] D. Lee and H. Seung, “Learning the parts of objects by non- negative matrix factorization,” Nature, vol. 401, no. 6755, pp. 788–791, 1999.

[42] T. White, Hadoop: The Definitive Guide. O’Reilly Press, second ed., 2010.

[43] Y. Seroussi, F. Bohnert, and I. Zukerman, “Personalized rating prediction for new users using latent factor models,” in Proceedings Of The 22nd ACM Conference On Hypertext And Hypermedia. ACM, 2011, pp. 47–56.

[44] Chihiro Ono, Mori Kurokawa, Yoichi Motomura, and Hideki Asoh, ‘A Context-Aware movie preference model using a bayesian network for recommendation and promotion’, in User Modeling 2007, 247–257, Springer-

Verlag, (2007).

- [45] J. Golbeck and J. Hendler, ‘FilmTrust: movie recommendations using trust in web-based social networks’, in Consumer Communications and Networking Conf., 2006. CCNC 2006. 3rd IEEE, volume 1, pp. 282–286, (2006).
- [46] FilmTrust, <http://trust.mindswap.org/FilmTrust/>, Último acesso em Agosto de 2011.
- [47] Ajit P. Singh , Geoffrey J. Gordon, Relational learning via collective matrix factorization, Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining, August 24-27, 2008.
- [48] Xavier Amatriain , Josep M. Pujol , Nava Tintarev , Nuria Oliver, Rate it again: increasing recommendation accuracy by user re-rating, Proceedings of the third ACM conference on Recommender systems, October 23-25, 2009.
- [49] Neal Lathia , Stephen Hailes , Licia Capra, kNN CF: a temporal social network, Proceedings of the 2008 ACM conference on Recommender systems, October 23-25, 2008.
- [50] Xavier Amatriain , Neal Lathia , Josep M. Pujol , Haewoon Kwak , Nuria Oliver, The wisdom of the few: a collaborative filtering approach based on expert opinions from the web, Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval, July 19-23, 2009.
- [51] Neal Lathia , Stephen Hailes , Licia Capra, Temporal collaborative filtering with adaptive neighbourhoods, Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval, July 19-23, 2009.
- [52] Tong Queue Lee , Young Park , Yong-Tae Park, An empirical study on effectiveness of temporal information as implicit ratings, Expert Systems with Applications: An International Journal, v.36 n.2, p.1315-1321, March, 2009.
- [53] Lakshmis Ramaswamy , Deepak P. , Ramana Polavarapu , Kutila

Gunasekera , Dinesh Garg , Karthik Visweswariah , Shivkumar Kalyanaraman, CAESAR: A Context-Aware, Social Recommender System for Low-End Mobile Devices, Proceedings of the 2009 Tenth International Conference on Mobile Data Management: Systems, Services and Middleware, p.338-347, May 18-20, 2009.

[54] Xiaoyuan Su , Taghi M. Khoshgoftaar, A survey of collaborative filtering techniques, Advances in Artificial Intelligence, 2009, p.2-2, January 2009.

[55] Tong Queue Lee , Young Park , Yong-Tae Park, An empirical study on effectiveness of temporal information as implicit ratings, Expert Systems with Applications: An International Journal, v.36 n.2, p.1315-1321, March, 2009.

[56] O. Celma. *Music Recommendation and Discovery in the Long Tail*. PhD thesis, Universitat Pompeu Fabra, Barcelona, Spain, 2008.

[57] L. Baltrunas, X. Amatriain, Towards Time-Dependant Recommendation based on Implicit Feedback. In *Proceedings of the third ACM conference on Recommender systems*, New York, USA, 2009, pp. 423--424.

[58] Andriy Shepitsen , Jonathan Gemmell , Bamshad Mobasher , Robin Burke, Personalized recommendation in social tagging systems using hierarchical clustering, Proceedings of the 2008 ACM conference on Recommender systems, October 23-25, 2008.

[59] Ioannis Konstas , Vassilios Stathopoulos , Joemon M. Jose, On social networks and collaborative recommendation, Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval, July 19-23, 2009.

[60] Yifan Hu , Yehuda Koren , Chris Volinsky, Collaborative Filtering for Implicit Feedback Datasets, Proceedings of the 2008 Eighth IEEE International Conference on Data Mining, p.263-272, December 15-19, 2008.

[61] J. L. Herlocker, J. A. Konstan, and J. Riedl. “Explaining collaborative filtering recommendations”, In Proceedings of the 2000 ACM Conference on Computer Supported Cooperative Work, ACM Press, pp. 241-250, 2000.

- [62] W. Hill, L. Stead, M. Rosenstein, G. Furnas, “Recommending and Evaluating Choices in a Virtual Community of Use”, Proc. Conf. Human Factors in Computing Systems, 1995.
- [63] P. Resnick, N. Iakovou, M. Sushak, P. Bergstrom, J. Riedl, “GroupLens: An Open Architecture for Collaborative Filtering of Netnews”, Proc. 1994 Computer Supported Cooperative Work Conf., 1994.
- [64] U. Shardanand, P. Maes, “Social Information Filtering: Algorithms for Automating ‘Word of Mouth’”, Proc. Conf. Human Factors in Computing Systems, 1995.
- [65] Balabanovic M., Shoham Y., “Fab: Content-based, collaborative recommendation”, Communications of the ACM, 66–72, 1997.
- [66] Resnick P., and Varian H. R., “Recommender systems”, Communications of the ACM 40, 56–58, March 1997.
- [67] Melville, P., Mooney, R. J., & Nagarajan, R., Content-boosted collaborative filtering for improved recommendations. In Eighteenth national conference on artificial intelligence (pp. 187–192), 2002.
- [68] Herlocker, J, Konstan, J., Terveen, L., and Riedl, J. Evaluating Collaborative Filtering Recommender Systems. *ACM Transactions on Information Systems* 22, ACM Press, 5-53, 2004.