

Antônio Guilherme Ferreira Viggiano  
Fernando Fochi Silveira Araújo

**Desenvolvimento de uma biblioteca  
computacional para sistemas de  
recomendação de produtos de lojas de  
comércio online**

**São Paulo, Brasil**  
**6 de novembro de 2014**



Antônio Guilherme Ferreira Viggiano  
Fernando Fochi Silveira Araújo

**Desenvolvimento de uma biblioteca computacional  
para sistemas de recomendação de produtos de lojas  
de comércio online**

Trabalho de Conclusão de Curso apresentado  
ao Departamento de Engenharia Mecatrônica  
da Escola Politécnica da Universidade de São  
Paulo com requisito parcial para obtenção do  
Grau de Engenheiro Mecatrônico.

Universidade de São Paulo  
Escola Politécnica  
Trabalho de Conclusão de Curso

Orientador: Prof. Dr. Fábio Gagliardi Cozman

São Paulo, Brasil  
6 de novembro de 2014



---

Antônio Guilherme Ferreira Viggiano  
Fernando Fochi Silveira Araújo

Desenvolvimento de uma biblioteca computacional para sistemas de recomendação de produtos de lojas de comércio online/ A.G.F. Viggiano; F.F.S. Araújo. – São Paulo, Brasil, 6 de novembro de 2014

87 p.

Orientador: Prof. Dr. Fábio Gagliardi Cozman

Trabalho de Formatura – Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia Mecatrônica e de Sistemas Mecânicos.

1. Inteligência artificial.
  2. Aprendizado computacional.
  3. Comercio eletrônico.
  4. Produtos I. Prof. Dr. Fábio Gagliardi Cozman. II. Universidade de São Paulo. Escola Politécnica. III. Departamento de Engenharia Mecatrônica e de Sistemas Mecânicos
-

Antônio Guilherme Ferreira Viggiano  
Fernando Fochi Silveira Araújo

## **Desenvolvimento de uma biblioteca computacional para sistemas de recomendação de produtos de lojas de comércio online**

Trabalho de Conclusão de Curso apresentado  
ao Departamento de Engenharia Mecatrônica  
da Escola Politécnica da Universidade de São  
Paulo com requisito parcial para obtenção do  
Grau de Engenheiro Mecatrônico.

---

**Prof. Dr. Fábio Gagliardi Cozman**  
Orientador

---

**Prof. Dr. Lucas Antonio Moscato**  
Convidado 1

---

**Prof. Dr. Thiago de Castro Martins**  
Convidado 2

---

**Prof. Dr. Arturo Forner Cordero**  
Convidado 3

---

**Profa. Dra. Larissa Driemeier**  
Convidado 4

São Paulo, Brasil  
6 de novembro de 2014

# Agradecimentos

Agradecemos ao professor Fábio Cozman pela sua orientação e apoio durante todo o projeto. Agradecemos também ao professor Thiago Martins e aos demais orientadores das disciplinas PMR2500 e PMR2550 – Projeto de Conclusão do Curso I e II – por terem nos guiado na elaboração da monografia e por terem sempre exigido trabalhos de alta qualidade. Esse papel é fundamental na valorização do diploma de Engenharia Mecatrônica da Escola Politécnica.



*Make things as simple as possible, but not simpler* (Albert Einstein)



# Resumo

O objetivo deste trabalho é projetar e implementar uma biblioteca computacional para sistemas de recomendações de produto de lojas de comércio, com a finalidade de permitir a fácil implementação de um sistema de recomendação genérico para ser utilizado na geração de email marketing. Para alcançar esse objetivo foi necessário um estudo sobre os principais métodos de cálculo de recomendações, desde as definições dos relacionamentos entre usuário e item às comparações entre itens e seus diversos atributos.

A biblioteca foi desenvolvida utilizando três diferentes algoritmos de recomendação. O algoritmo baseado na ponderação de atributos, que trata-se de um método híbrido entre filtragem colaborativa e filtragem baseada em conteúdo, onde a partir da regressão linear de dados de uma rede social, extraem-se os pesos que determinam a importância de cada atributo dos itens. O segundo método, baseado em no perfil de usuários, leva em consideração o interesse dos usuários por *features*, indiretamente calculado a partir de seu interesse pelos itens. O terceiro método, baseado na correlação usuário-item, é uma variante do método baseado no perfil de usuários. Este método busca os itens com *features* mais similares aos atributos pelos quais o usuário se interessa, através dos atributos.

**Palavras-chaves:** Inteligência artificial, Aprendizado computacional, Comercio eletrônico, Produtos



# Abstract

This is the english abstract.

**Key-words:** latex, abntex, text editoration.



# Lista de tabelas

Tabela 1 – Atributos $a_{if}$	25
Tabela 2 – Avaliações $r_{ui}$	25
Tabela 3 – Avaliações $r_{ui}$	27
Tabela 4 – Atributos $a_{if}$	27
Tabela 5 – Avaliação de sistemas de predição	40
Tabela 6 – Avaliações $r_{ui}$	41
Tabela 7 – Atributos $a_{if}$	41
Tabela 8 – $d_{ij}^f$	44
Tabela 9 – Medidas de distância entre alguns atributos	44
Tabela 10 – $e_{ij}$	44
Tabela 11 – $w_f$	44
Tabela 12 – $s_{ij}$	46
Tabela 13 – $\hat{i}_u$ (FW)	46
Tabela 14 – TF <sub>uf</sub>	46
Tabela 15 – IDF <sub>f</sub>	46
Tabela 16 – $w_{uf}$	46
Tabela 17 – $s_{uv}$	46
Tabela 18 – f <sub>uf</sub>	48
Tabela 19 – $\omega_{ui}$ (UP)	48
Tabela 20 – $\hat{i}_u$ (UP)	48
Tabela 21 – $\omega_{ui}$ (UI)	48
Tabela 22 – $\hat{i}_u$ (UI)	48
Tabela 23 – Parâmetros de influência no desempenho dos algoritmos de recomendação	61



# Lista de símbolos

$k$	Número de vizinhos mais próximos
$N$	Tamanho da lista de recomendação
$\mathcal{U}$	Conjunto de todos os usuários
$\mathcal{I}$	Conjunto de todos os itens
$\mathcal{F}$	Conjunto de todos os atributos dos itens
$u, v$	Usuários
$i, j$	Itens
$f$	Atributos dos itens
$\mathbf{X}_{M \times N}, \mathbf{X}$	Matriz de elementos $x_{mn}$
$\mathbf{x}_N, \mathbf{x}$	Vetor de elementos $x_n$
$\tilde{x}$	Valor ótimo de $x$
$\hat{x}$	Valor estimado de $x$
$ \mathcal{X} $	Número de elementos do conjunto $\mathcal{X}$
$\mathbf{R}, r_{ui}$	Avaliação feita pelo usuário $u$ do item $i$
$\mathbf{A}, a_{if}$	Atributo $f$ presente no item $i$
$\mathbf{S}, s_{ij}, s_{uv}$	Similaridade entre itens $i$ e $j$ ou entre usuários $u$ e $v$
$\mathbf{W}, w_{uf}$	Correlação ponderada entre usuário $u$ e atributo $f$
$\Omega, \omega_{ui}$	Correlação entre usuário $u$ e item $i$
$\mathbf{w}, w_f$	Peso do atributo $f$



# Sumário

<b>1</b>	<b>INTRODUÇÃO</b>	<b>19</b>
1.1	Motivação	20
1.2	Objetivos	21
<b>2</b>	<b>ESTADO DA ARTE</b>	<b>23</b>
2.1	Estado da arte dos problemas	23
2.2	Estado da arte das soluções	26
2.3	Desafios científicos e tecnológicos	26
2.4	Soluções propostas	28
<b>3</b>	<b>METODOLOGIA</b>	<b>31</b>
3.1	Definição da Necessidade	31
3.2	Definição dos Parâmetros de Sucesso	31
3.3	Síntese de Soluções	31
3.4	Detalhamento da Solução	32
3.5	Estruturação do Banco de Dados	32
3.6	Validação Cruzada	33
<b>4</b>	<b>REQUISITOS</b>	<b>35</b>
4.1	Diagrama de Casos de Uso	36
4.2	Diagrama de Atividades	38
4.3	Avaliação de Desempenho	39
<b>5</b>	<b>DETALHAMENTO DE SOLUÇÕES</b>	<b>41</b>
5.1	Algoritmo baseado na ponderação de atributos (FW)	41
5.2	Algoritmo baseado no perfil de usuários (UP)	43
5.3	Algoritmo baseado na correlação usuário-item (UI)	47
<b>6</b>	<b>DESENVOLVIMENTO DA BIBLIOTECA</b>	<b>49</b>
6.1	Recursos acadêmicos	49
6.2	Ferramentas utilizadas	50
6.3	Métodos computacionais	50
6.3.1	Estrutura da biblioteca	50
6.3.2	Algoritmo baseado na ponderação de atributos (FW)	51
6.3.2.1	Determinação de $e_{ij}$	51
6.3.2.2	Determinação de $d_{ij}^f$	51
6.3.2.3	Determinação de $w_f$	52

6.3.2.4	Determinação de $s_{ij}$	52
6.3.2.5	Determinação de $\hat{i}_u$	52
6.3.3	Algoritmo baseado no perfil de usuários (UP)	53
6.3.3.1	Determinação de $w_{uf}$	53
6.3.3.2	Determinação de $s_{uv}$	53
6.3.3.3	Determinação de $\omega_{ui}$	54
6.3.3.4	Determinação de $\hat{i}_u$	54
6.3.4	Algoritmo baseado na correlação usuário-item (UI)	54
6.4	Ambiente de testes	55
7	<b>RESULTADOS</b>	61
7.1	Tamanho da lista de recomendações $N$	61
7.2	Percentual da base de aprendizado $T$	64
7.3	Percentual de avaliações “escondidas” dos usuários-teste na validação cruzada $H$	64
7.4	Valor mínimo para avaliações positivas $M$	67
7.5	Número de vizinhos mais próximos $k$	72
7.6	Conjunto de atributos dos itens $\mathcal{F}$	72
7.7	Medida de distância entre atributos $d^f$	76
7.8	Pesos dos atributos $w_f$	76
8	<b>CONCLUSÃO</b>	79
A	<b>DOCUMENTAÇÃO DA BIBLIOTECA</b>	81
	<b>Referências</b>	85

# 1 Introdução

O comércio on-line se torna cada vez mais importante na vida das pessoas, de forma que a adoção deste método de compra é cada vez mais comum. Estima-se que em 2013 um bilhão de pessoas compraram online (1), gerando uma receita anual de 1,25 trilhão de dólares com expectativas de crescer 17% ao ano até 2017. (2). Para exemplificar, a gigante chinesa Alibaba abriu o seu capital na bolsa de valores americana. Esta oferta pública inicial arrecadou 25 bilhões de dólares, significando a maior oferta pública inicial do mercado acionário americano de todos os tempos. Isso transformou a Alibaba no maior varejista online do mundo, com um valor avaliado em 158 bilhões de dólares (3).

Ao analisarmos o mercado brasileiro, percebemos que se trata de um comércio jovem e com bom potencial. Percebe-se ainda que, no Brasil, o hábito de se fazer compras pela internet não está consolidado. Apenas 19% dos compradores usam esse serviço semanalmente, enquanto em países mais desenvolvidos estes números chegam a 35% na Alemanha e a 39% no Reino Unido. Outro indicador de que o mercado brasileiro ainda é jovem é que 61% dos consumidores de varejo online utilizaram o serviço pela primeira vez nos últimos 4 anos (4).

Como o mercado de varejo online é novo como um todo, este ainda passará por algumas mudanças drásticas em um curto espaço de tempo. Um dos itens-chave destas mudanças é a capacidade de se analisar os dados gerados pelos consumidores. Com estes dados será possível segmentar os clientes mais facilmente e as empresas poderão direcionar suas investidas de forma mais eficiente, chegando ao ponto em que campanhas de marketing e precificação serão totalmente personalizadas (5). Uma das maneiras de se usar estes dados são os sistemas de recomendação.

“Sistemas de recomendação são ferramentas e técnicas de software destinadas a prover sugestões de itens para usuários” (6). O sistema tem o propósito de automatizar o processo de recomendação e auxiliar na tomada de decisão, podendo ser aplicado em diversas áreas da indústria, tais como na indicação de notícias, músicas, relações de amizade ou artigos científicos.

Estes sistemas são utilizados por diversos serviços online e geram um grande impacto quando utilizados corretamente. Em 2012, cerca de 75% dos vídeos assistidos através do site NetFlix foram acessados por meio de recomendações (7). Em 2006, as recomendações representaram 35% dos livros vendidos pela Amazon (8), enquanto em 2007 cerca de 38% das notícias lidas no Google News foram sugeridas por um sistema de recomendação (9).

De modo geral, um sistema de recomendação possui três etapas: a aquisição dos

dados de entrada, a determinação das recomendações e finalmente a apresentação dos resultados ao usuário. A aquisição dos dados de entrada pode ser feita tanto de forma automática quanto manual, e em geral utiliza-se um banco de dados para armazenar essas informações. As sugestões são feitas segundo uma estratégia de recomendação determinada a priori, que pode ser fundamentada nas preferências do usuário, nas características dos itens ou em alguma formulação mista. Finalmente, os resultados são apresentados na interface sob variadas formas, como por exemplo em uma lista dos  $N$  itens mais relevantes para o usuário.

Conforme o tipo específico de itens recomendados, o design do sistema, a interface homem-máquina e o tipo de técnica de recomendação são construídos a fim de prover sugestões mais adequadas.

Os sistemas de recomendação são destinados primeiramente aos indivíduos que não possuem competência ou experiência suficiente para avaliar o grande número de opções do conjunto total de itens. Dessa forma, o sistema é adaptado a cada um dos usuários, de maneira que eles recebam recomendações adequadas ao seu perfil. Essa ideia, amplamente divulgada por um antigo diretor executivo do e-commerce *Amazon.com*, se resume à sua fala de que “se você possui 2 milhões de clientes na web, você precisa ter 2 milhões de lojas na web” (10).

## 1.1 Motivação

Conforme apresentado, a quantidade de lojas de varejo online cresce em ritmo acelerado no Brasil e no mundo. Motivados pela importância econômica dos e-commerces, bem como pela possibilidade de criar um conjunto de ferramentas *open source* que possam ser utilizadas pela comunidade acadêmica e empresarial, propomos como Trabalho de Conclusão de Curso o desenvolvimento de uma biblioteca computacional para sistemas de recomendação de produtos de lojas de comércio online.

Esse pacote computacional é composto de métodos de leitura de dados de histórico de compras e de informações de clientes e produtos, de cálculo de sugestões de itens com base em algoritmos de recomendação e de análise de desempenho das recomendações.

A motivação de se criar uma biblioteca de software decorre principalmente da sua abrangência e capacidade de adaptação, visto que é possível atender a mais casos de uso que um sistema de recomendação completo. De um lado, um sistema de recomendação possui uma finalidade específica – como por exemplo de sugerir notícias para usuários de internet – e uma entrada e saída de dados específica – como por exemplo o fato de as notícias sempre estarem ordenada pelas mais recentes em uma tabela de sugestões. De outro lado, uma biblioteca computacional pode receber qualquer tipo de dados e gerar qualquer saída de dados.

Caso uma empresa ou um acadêmico queira construir seu próprio sistema de recomendação, basta elaborar a conexão entre o pacote apresentado pela dupla, seu banco de dados e a interface gráfica de apresentação de resultados.

As contribuições científica e tecnológica deste trabalho para a Engenharia Mecatrônica estão sobretudo nos campos de inteligência artificial, de sistemas de informação e de automação de processos.

As competências acadêmicas necessárias para a execução desse trabalho envolvem algoritmos e estruturas de dados (abordados em PMR2300 – Computação para Automação), documentação e modelagem de sistemas computacionais (explicados em PMR2440 – Programação para Automação), sistemas de informação e banco de dados (tratados em PMR2490 – Sistemas de Informação) e inteligencia artificial, com enfase em aprendizado de máquina (aprofundados em PMR2728 – Teoria de Probabilidades em Inteligência Artificial e Robótica). As competências técnicas abrangem programação estatística e funcional, demonstradas através da linguagem R.

## 1.2 Objetivos

O objetivo do presente Trabalho de Conclusão de Curso é o desenvolvimento de uma biblioteca computacional para sistemas de recomendação de produtos de lojas de comércio online, e respectiva análise de desempenho das recomendações propostas.

O pacote de software é composto de três diferentes algoritmos de recomendação, além de funções para avaliar a qualidade das sugestões. Neste texto, será feita uma avaliação comparativa entre os três algoritmos. A explicação detalhada dos métodos se encontra no Capítulo 5.

A fim de se poder experimentar a influência de diversos parâmetros na qualidade das recomendações, todas as funções foram desenvolvidas integralmente pela dupla, e nenhuma biblioteca externa foi utilizada. O objetivo do trabalho não é, portanto, o uso de ferramentas de recomendação já disponíveis no mercado, mas sim a elaboração de uma biblioteca que possibilite a construção e análise de um sistema de recomendação próprio. Qualquer e-commerce interessado no assunto pode, portanto, apropriar-se do pacote de software e modificá-lo para atender a suas especificidades e melhorar as sugestões.

Essa ferramenta tem como finalidade a automatização do processo de recomendação de itens, e pode ser aplicada em diversas áreas da indústria, tais como na indicação de notícias, músicas, relações de amizade ou artigos científicos. No nosso trabalho, a biblioteca terá como foco a sugestão de produtos de lojas de comércio online que disponham de um histórico de compras dos usuários e das características dos produtos.

A qualidade das recomendações será avaliada quanto a precisão, abrangência e

tempo de execução. Uma descrição detalhada da avaliação do sistema de recomendação está descrita na Seção 4.3.

Por meio de uma validação cruzada, analisaremos a influência dos principais parâmetros do problema na qualidade das recomendações, como o tamanho do banco de dados, a quantidade de informações de itens e clientes utilizadas na recomendação e outros.

Será discutido o impacto dos principais desafios tecnológicos e científicos dos sistemas de recomendação na nossa proposta de solução, tais como a escalabilidade, a adaptação a novos usuários e a esparsidade dos dados (11).

Ao final, será possível extrair uma validação experimental das diretrizes fundamentais a serem seguidas por e-commerce que desejem desenvolver um sistema de recomendação próprio a partir da biblioteca desenvolvida neste trabalho.

## 2 Estado da Arte

As terminologias *cliente* e *usuário* neste texto serão intercambiáveis e sem distinção semântica, mesmo que na prática essas duas entidades possam ser diferentes. Da mesma forma, *item* e *produto* terão o mesmo significado neste trabalho.

A fim de tornar a formulação mais genérica, também não faremos distinção entre *avaliação positiva* de um item e *compra* de um item. Avaliação positiva é toda avaliação  $r_{ui}$  do item  $i$  feita pelo usuário  $u$  tal que  $r_{ui} > M$ , e avaliação negativa tal que  $r_{ui} \leq M$ , sendo  $M$  um valor mínimo escolhido a priori, indicador de que o usuário  $u$  “gostou” do item  $i$ . No caso de um banco de dados sem avaliações dos produtos, será levada em conta a compra dos itens e será admitida avaliação unitária e valor mínimo nulo. Desta forma, os bancos de dados que contenham informações do tipo “usuário  $u$  avaliou o item  $i$  em  $r_{ui} = 3.54 > M$ ” e aqueles que contenham “usuário  $u$  comprou o item  $i$ , logo  $r_{ui} = 1 > 0$ ” serão tratados equivalentemente. Vale observar que essa definição difere da Referência 12, em que avaliação positiva é aquela tal que  $r_{ui} \geq M$ .

### 2.1 Estado da arte dos problemas

O problema de recomendação pode ser formulado como se segue, adaptado da Referência 13, com notação inspirada no artigo 12:

“Seja  $\mathcal{U}$  o conjunto de todos os usuários e seja  $\mathcal{I}$  o conjunto de todos os itens que podem ser recomendados, tais como livros, filmes ou artigos científicos. Seja  $\ell$  uma função de utilidade, que mede a relevância do produto  $i$  para usuário  $u$ . Em notação matemática,  $\ell : \mathcal{U} \times \mathcal{I} \rightarrow \mathcal{R}$ , onde  $\mathcal{R}$  é um conjunto totalmente ordenado – por exemplo, números inteiros ou números reais dentro de um determinado intervalo, em geral  $\{-1, 0, +1\}$  ou  $[1, 5]$ . O objetivo do sistema de recomendação é determinar o item  $\tilde{i}_u$  que maximize a utilidade  $\ell_{ui}$  do usuário  $u$ .”

$$\forall u \in \mathcal{U}, \tilde{i}_u = \arg \max_{i \in \mathcal{I}} \ell_{ui} \quad (2.1)$$

O problema central da recomendação é que “em geral a função  $\ell$  é desconhecida ou não é definida para todo o espaço  $\mathcal{U} \times \mathcal{I}$ ”, e portanto determinar  $\tilde{i}$  através da Equação 2.1 é inviável.

Em algumas formulações, “a utilidade é descrita pela avaliação  $r_{ui}$  do item  $i$  feita pelo usuário  $u$ ”. Neste caso, o sistema de recomendação busca determinar  $\hat{r}_{ui}$  que melhor se aproxime de  $r_{ui}$ , e a qualidade da recomendação é normalmente descrita pela distância

entre esses dois valores. Em outros sistemas, todavia, a utilidade é descrita diferentemente, de forma que o item com maior valor de  $\hat{r}_{ui}$  não é necessariamente recomendado.

Para lidar com o problema da recomendação, existem três grandes grupos de estratégias de sugestão de itens, segundo as Referências 13, 14:

- Recomendações baseadas em conteúdo: o usuário recebe sugestões de itens similares àqueles pelos quais ele se interessou no passado;
- Recomendações colaborativas: o usuário recebe sugestões de itens que pessoas com preferências semelhantes gostaram no passado;
- Recomendações híbridas: esses métodos combinam características de sistemas colaborativos e baseados em conteúdo. O usuário recebe sugestões de itens compatíveis com seu perfil e de itens do interesse de usuários com perfil similar.

As estratégias de recomendação baseadas em conteúdo exploram os dados dos itens para calcular a sua relevância conforme o perfil do usuário. Suas técnicas de recomendação podem ser classificadas em dois grupos: aquelas baseadas em heurísticas ou memória – fazem a previsão com base em toda a coleção de itens anteriormente classificados pelos usuários – e aquelas baseadas em modelos – utilizam o conjunto de avaliações com o objetivo de descrever a interação entre usuários e itens, tal como em uma regressão linear ou em uma rede Bayesiana.

Na abordagem de sistemas baseados em conteúdo, a recomendação pode ser vista como um problema de aprendizado de máquina, em que o sistema adquire conhecimento sobre o usuário. Muitas vezes é recomendado que o aprendizado seja feito com base no perfil do usuário em uso contínuo, ao invés de forçá-lo a responder diversas perguntas demográficas, como idade, gênero, classe social, etc (15). O objetivo é categorizar novas informações baseadas em informações previamente adquiridas e rotuladas como interessantes ou não pelo usuário. Com estas informações em mãos, é possível gerar modelos preditivos que evoluem conforme aparecem novas informações.

Em sistemas baseados em conteúdo, os itens a serem recomendados podem possuir diversos atributos e formas de classificação. Em documentos como e-mails, *websites* ou comentários de usuários, os textos não têm estrutura definida e a abordagem mais comum para escolher o melhor item é a mineração de informações. O usuário procura por uma lista de termos desejados e o sistema retorna os textos de maior relevância, tal como é feito em um motor de busca (16). Nesses casos, calcula-se a similaridade entre documentos a partir da importância das palavras ou termos similares, como a TF-IDF ou o classificador Bayesiano (17).

Em bancos de dados relacionais, os itens possuem uma categorização pré-definida, e sua relevância depende das suas características, descritas pela matriz de atributos **A**. Cada

*feature* pertence a um conjunto distinto, podendo ser booleano (possui ou não possui), inteiro ou real (preço, data, etc.), ou um coleção finita de valores (marca, modelo, gênero, etc.), como exemplifica a Tabela 1.

Tabela 1 – Atributos  $a_{if}$

	$f_1$	$f_2$	$f_3$	$f_4$
$i_1$	1	50	0.8	P
$i_2$	0	75	0.3	M
$i_3$	1	30	0.4	G

As recomendações colaborativas, por sua vez, tentam prever a utilidade dos itens para cada cliente com base em itens previamente avaliados por outros usuários. Elas podem ser baseadas em usuários, isto é, na escolha de clientes que possuam avaliações similares de produtos, quanto baseadas em itens, na escolha de produtos avaliados similarmente (18).

Mais formalmente, quando a filtragem colaborativa é baseada em usuários, a utilidade  $\ell_{ui}$  de um item  $i$  para um usuário  $u$  é estimada com base nas utilidades  $\ell_{v_k^u}$  dos usuários  $v_k^u \in \mathcal{U}$  que são “similares” ao usuário  $u$ . De maneira análoga, quando baseada em itens, a utilidade  $\ell_{ui}$  é prevista com base nas utilidades  $\ell_{uj_k^u}$ , dado itens  $j_k^u \in \mathcal{I}$  que são “similares” aos itens  $i$ .

Na prática, o cálculo das recomendações para sistemas colaborativos é feito a partir da matriz de avaliações  $\mathbf{R}$ . Isso pode ser exemplificado pela Tabela 2, que possui avaliações de 1 a 5, sendo  $M = 2$ . Em um sistema usuário-usuário, o cliente  $u_1$  receberia recomendação do item  $i_4$ , pois para os itens  $i_2$  e  $i_3$  suas avaliações foram similares às do cliente  $u_2$ . Já para um sistema item-item, o usuário  $u_3$  receberia recomendação do item  $i_3$ , pois este tem avaliações similares às do item  $i_2$ , avaliado positivamente pelo usuário  $u_3$ .

Tabela 2 – Avaliações  $r_{ui}$

	$i_1$	$i_2$	$i_3$	$i_4$
$u_1$	-	4	3	-
$u_2$	-	4	3	5
$u_3$	2	5	-	1

Por fim, as recomendações híbridas combinam aspectos tanto da filtragem colaborativa (baseada em usuários ou em itens) quanto da filtragem baseada em conteúdo, com o objetivo de atingir uma melhor recomendação ou de superar problemas recorrentes nas técnicas individuais, como a esparsidade (*sparsity*) dos dados ou o *cold start* (19).

## 2.2 Estado da arte das soluções

Do ponto de vista do estado da arte das soluções, as variáveis de interesse estão ligadas ao número de usuários no sistema, ao número de itens, à medida de qualidade da recomendação e ao custo computacional (20).

No que se refere à dependência do número de usuários, a filtragem colaborativa baseada em usuários é extremamente efetiva para um baixo número de usuários. A filtragem colaborativa a base de itens é consideravelmente pior para um baixo número de usuários, mas supera todos os outros métodos baseados em memória conforme o número de clientes aumenta.

A dependência do número de itens é, de certa forma, oposta à de usuários: a filtragem colaborativa baseada em itens é extremamente efetiva para poucos itens, enquanto aquela baseada em usuários supera todos os outros métodos baseados em memória para grandes quantidades de itens.

Com relação à medida de qualidade, avaliada a partir da abrangência dos dados, a filtragem baseada em usuários e a baseada em itens mostram uma dependência semelhante. Na análise de menor erro quadrático médio entre o item sugerido e o item efetivamente comprado, todos os métodos de recomendação variam não-linearmente com o número de usuários, itens e acurácia, e de modo geral há um compromisso (*trade-off*) entre a esparsidade (*sparsity*) dos dados e o tempo de processamento.

## 2.3 Desafios científicos e tecnológicos

Um dos maiores desafios tecnológicos dos sistemas de recomendação é, atualmente, o da escalabilidade (15). O sistema de recomendação deve ser flexível no sentido de poder operar igualmente bem tanto em pequenas quanto em grandes bases de dados, que podem chegar até centenas de milhões de clientes (21) e de produtos (22). Isso significa que as recomendações devem ser suficientemente rápidas e ainda assim prover sugestões valiosas aos consumidores.

Um problema muito comum nos sistemas de recomendação é o do *cold start*, que atinge principalmente os sistemas de filtragem colaborativa, grandemente dependentes da matriz de avaliações  $\mathbf{R}$ . Quando itens ou usuários são inicialmente introduzidos no sistema, existe pouca ou nenhuma informação sobre eles. O sistema é incapaz de realizar inferências sobre quais itens recomendar ao novo usuário ou sobre quais produtos são similares ao novo item. Na Tabela 3, por exemplo, o item  $i_{100}$  não possui nenhuma avaliação, e nunca seria recomendado em sistemas puramente baseados em itens. Analogamente, o usuário  $u_3$  também não teria nenhuma sugestão de itens para sistemas puramente baseados em usuários.

Tabela 3 – Avaliações  $r_{ui}$ 

	$i_1$	$i_2$	...	$i_{100}$
$u_1$	5	4	...	-
$u_2$	-	2	...	-
$u_3$	-	-	...	-

Também é uma grande dificuldade dos algoritmos baseados em filtragem colaborativa a esparsidade dos dados. Como a maioria dos clientes interage com uma pequena quantidade de itens, a matriz de avaliações tem em geral menos de 1% dos valores preenchidos, e o sistema deve prever os outros valores (23).

Outro desafio científico é referente à diversidade das recomendações realizadas, também chamado de excesso de especialização (*over-specialization*) (13). Ao mesmo tempo que o sistema deve apresentar itens similares ao que o usuário está procurando, ele também deve sugerir itens que o usuário desconheça ou que nem saiba que poderiam interessá-lo. Esse problema afeta principalmente os sistemas baseados em conteúdo, pois itens com características similares tendem a ser sempre recomendados. Na Tabela 4, o item  $i_2$  seria sugerido para um usuário que tenha avaliado  $i_1$ , apesar de esse não apresentar nenhuma característica diferente do item previamente comprado. Para contornar essa dificuldade, costuma-se introduzir elementos de aleatoriedade na recomendação, por exemplo a partir de algoritmos genéticos (14).

Tabela 4 – Atributos  $a_{if}$ 

	$f_1$	$f_2$	$f_3$
$i_1$	1	50	0.8
$i_2$	1	50	0.8
$i_3$	0	75	0.3

Além desse desafio, existe também o da análise rasa do conteúdo (*shallow content analysis*). O sistema, ao avaliar a característica dos itens, não consegue extraír importantes aspectos para o usuário caso eles não estejam explicitamente descritos na categorização do banco de dados. Isso pode ocorrer, por exemplo, com fatores externos ao produto que influenciem na compra do usuário, como em datas comemorativas, em compras induzidas por propaganda, em compras “impulsivas”, etc. Se um usuário comprou um arranjo de flores no dia das mães, não é necessariamente verdade que ele se interessa por flores. Da mesma maneira, sistemas que ignorem a sazonalidade de certos produtos não recomendariam arranjos de flores para clientes que não tenham comprado itens parecidos, mesmo no dia das mães.

Por fim, um desafio científico que este trabalho enfrentará é a execução de um sistema híbrido do ponto de vista de efemeridade e persistência, ao construir um modelo

de recomendação que integre as preferências de curto e longo termo dos usuários (10). A análise dos dados de compras anteriores, bem como de dados demográficos, deverá portanto ser incorporada à análise de característica dos produtos, a fim de enriquecer a acurácia do sistema (15).

Esse tópico de pesquisa inclui ainda diversos desafios científicos e tecnológicos que não serão tratados no nosso projeto, tais como a preservação da privacidade dos usuários, a criação de modelos de recomendação inter-domínios, o desenvolvimento de sistemas descentralizados operando em redes computacionais distribuídas, a otimização de sistemas para sequências de recomendações, a otimização de sistemas para dispositivos móveis e outros. Um sistema de recomendação inteligente também deveria prever quando enviar uma determinada recomendação, e não agir apenas mediante requisição dos clientes (24).

## 2.4 Soluções propostas

Este Trabalho de Conclusão de Curso aborda três propostas de solução para o problema da recomendação, sendo duas delas retiradas de referências bibliográficas (12, 25), e uma outra apresentada pela dupla. O objetivo é realizar uma análise comparativa entre cada um dos métodos e estabelecer diretrizes para sua aplicação em e-commerce. Os algoritmos propostos estão descritos com maior detalhe no Capítulo 5.

Todas as soluções são algoritmos híbridos, por utilizarem na recomendação tanto a matriz de avaliações **R** quanto a matriz de atributos **A**. Optou-se por dar importância aos algoritmos híbridos em razão de os e-commerce estruturarem seus bancos de dados em torno da descrição dos itens à venda. De modo geral, as tabelas de itens possuem dezenas de atributos, dependendo do ramo de negócios da loja, e pouco detalhe é dado à interação entre o grupo de usuários e itens. A tabela de histórico de compras se limita a informações como data e método de pagamento, e detém pouca informação adicional que possa ser utilizada na recomendação de produtos. Dessa forma supusemos que métodos puramente colaborativos, fundamentados na avaliação dos itens por parte dos usuários, teriam pior desempenho que métodos baseados em conteúdo, que exploram as características dos itens na recomendação.

A solução FW determina a similaridade de dois itens a partir de medidas de distância para cada um dos atributos dos itens, ponderadas por pesos determinados na regressão linear de uma equação descrita pelo interesse dos usuários em cada *feature*.

O método UP parte do princípio que os usuários estão interessados nos atributos dos itens, e traça correlações entre esses dois elementos para obter pesos que servirão de base para o cálculo da similaridade inter-usuários, utilizada na recomendação pelo método da vizinhança (*nearest neighbors*).

A variante UI, elaborada pela dupla, recomenda o melhor item a partir das matrizes de correlação usuário-atributo e atributo-item, a fim de obter a matriz de correlação usuário-item. De antemão espera-se que essa solução tenha desempenho similar ao método de base UP, pois ambos buscam explorar as características dos itens para determinar a preferência do usuário.



# 3 Metodologia

A metodologia de projeto deste Trabalho de Conclusão de Curso foi fundamentada principalmente na Referência 26. Por se tratar de um projeto de Engenharia de Software, foi necessário dar ênfase às etapas iterativas de desenvolvimento dos algoritmos. Esse processo cíclico, com fases de especificação, desenvolvimento e validação, permitiu obter resultados preliminares e os modificar os algoritmos ao longo da disciplina, ajustando detalhes e melhorando o sistema gradativamente (27).

A metodologia de execução do projeto, assim como a de avaliação dos resultados, pode ser consolidada da seguinte maneira:

## 3.1 Definição da Necessidade

Com o crescente número de lojas de comércio online, tornou-se necessário a criação de sistemas que pudessem entender e prever o comportamento de consumidores, a fim de oferecer produtos específicos para cada um deles, aumentando o número de vendas e a satisfação do cliente. Observa-se atualmente que o número de sistemas de recomendação gratuitos, de fácil integração e de código aberto (*open source*) são limitados e não correspondem às necessidades do mercado. Existe, pois, a necessidade da criação de uma biblioteca que possa ser utilizada por e-commerce que desejem estabelecer seu próprio sistema de recomendação ou mesmo por indivíduos interessados na temática da recomendação de itens.

## 3.2 Definição dos Parâmetros de Sucesso

O sucesso do projeto pode ser medido em duas frentes: a primeira, quantitativa, mede a precisão e a abrangência das recomendações; a segunda, qualitativa, avalia se o sistema responde bem aos problemas recorrentes desse tópico de pesquisa, tais como a escalabilidade, o excesso de especialização e outros.

## 3.3 Síntese de Soluções

Nesta fase do projeto, foram propostas possíveis soluções para o desafio da recomendação. Decidiu-se avaliar dois métodos híbridos do meio acadêmico e um outro elaborado pela dupla.

## 3.4 Detalhamento da Solução

Após a escolha dos métodos de recomendação, as soluções foram detalhadas matematicamente segundo uma mesma notação, e a estrutura dos algoritmos foi descrita e exemplificada. Neste ponto, escolheu-se também a linguagem de programação R e a forma de entrada e saída de dados, por meio de arquivos .csv.

A fim de facilitar o pré-processamento dos dados, estabelecemos que seriam necessários dois arquivos. Um deles deve conter a matriz de atributos **A** e o outro, a matriz de avaliações **R**.

$$\mathbf{A} = \begin{bmatrix} a_{i_1 f_1} & a_{i_1 f_2} & a_{i_1 f_3} & \dots \\ a_{i_2 f_1} & a_{i_2 f_2} & a_{i_2 f_3} & \dots \\ a_{i_3 f_1} & a_{i_3 f_2} & a_{i_3 f_3} & \dots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix} \quad (3.1)$$

$$\mathbf{R} = \begin{bmatrix} r_{u_1 i_1} & r_{u_1 i_2} & r_{u_1 i_3} & \dots \\ r_{u_2 i_1} & r_{u_2 i_2} & r_{u_2 i_3} & \dots \\ r_{u_3 i_1} & r_{u_3 i_2} & r_{u_3 i_3} & \dots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix} \quad (3.2)$$

## 3.5 Estruturação do Banco de Dados

Uma vez determinada a forma de entrada de informações, definiram-se os conjuntos de dados a serem utilizados.

O primeiro conjunto de dados abertos é proveniente do sistema de recomendações de filmes MovieLens (<http://movielens.umn.edu>), e é composto de 100 000 avaliações (valores inteiros de 1 a 5) de 943 usuários para 1682 filmes (28). Além disso, cada usuário (idade, sexo, profissão, logradouro) avaliou pelo menos 20 filmes (categoria, ano de publicação). Nessa base de dados, chamada de 100k, o catálogo de filme faz o papel de catálogo de produtos, e o histórico de compras se refere à avaliação dos filmes feita por cada usuário.

O segundo banco de dados é extraído do Internet Movie Database (IMDB), e possui 28 819 filmes. Esse banco está presente na biblioteca `ggplot2` da linguagem de programação R (29).

Na nossa análise, os bancos de dados 100k e IMDB foram utilizados complementarmente. A união desses dois conjuntos deu origem à base 100k-IMDB, composta por 943 usuários, 1682 itens e 25 atributos. Na biblioteca proposta pela dupla, os dados demográficos de usuários não são utilizados.

Ainda na etapa de implementação, confirmamos a validade de cada um dos métodos aplicando-os nas matrizes-referência (Tabelas 6 e 7).

## 3.6 Validação Cruzada

A fim de realizar um estudo comparativo (*benchmarking*) com os artigos de referência, mantivemos a mesma metodologia de avaliação de qualidade do artigo 12.

Em particular, implementamos uma validação cruzada considerando  $T = 75\%$  do banco de dados como base de treinamento ou aprendizado e os 25% restantes como base de testes. Em seguida, mascaramos  $H = 75\%$  das avaliações dos usuários-teste, de modo a medir a qualidade do sistema de recomendação em prever os itens positivamente avaliados. Cerca de uma dezena de parâmetros de interesse foram avaliados para cada um dos métodos (Tabela 23).

Além disso, não fizemos distinção entre valores não observados (*NA value/NULL value*) e avaliações nulas ( $r_{ui} = 0$ ), pois na maioria dos casos essa simplificação é válida. Esse não é o caso, por exemplo, de sistemas em que o usuário pode deliberadamente dar nota zero para um item.

Sabe-se que a extração de um modelo por meio de uma validação cruzada sobre uma mesma base de dados pode gerar *overfitting* (30). Para não cair nesse erro e com foco na reproduzibilidade do trabalho, realizamos todas as amostragens em R utilizando o número 2 como semente aleatória (*state seed*). Dessa forma, os parâmetros calculados para os modelos são sempre os mesmos para qualquer teste de qualidade. Evidentemente, caso se deseje avaliar a performance dos métodos para um outro banco de dados, uma validação cruzada rigorosa deverá ser aplicada.

Como a complexidade dos algoritmos excede o limite dos computadores pessoais da dupla, foi necessário contratar o serviço de computação nas nuvens Amazon Web Services.

Alugamos duas máquinas virtuais do tipo `r3.large`, otimizadas para memória. As máquinas, de especificação 2 vCPU, 15 GB de memória RAM e sistema operacional Amazon Linux AMI release 2014.09 x86\_64, baseado em RHEL Fedora, custaram USD 0,175 por hora de uso. Todos os testes foram realizados em aproximadamente 12 horas, custando apenas USD 4,20 (menos de R\$ 12,00). Uma explicação detalhada da configuração do ambiente de testes se encontra na Seção 6.4.



## 4 Requisitos

A partir dos objetivos deste Trabalho de Conclusão de Curso, é possível extrair os requisitos funcionais da biblioteca de sistemas de recomendação. Esses requisitos ditam principalmente sobre a escalabilidade e o desempenho das recomendações do sistema.

A fim de poder estabelecer uma base comparativa entre o sistema proposto UI e os sistemas de referência FW e UP, serão utilizados os mesmos indicadores de desempenho dos artigos-base: precisão, abrangência e medida  $F_1$  (12, 25). Precisão é a porcentagem de casos corretamente preditos em relação ao tamanho da lista de recomendações. Abrangência é a razão entre o número de itens corretamente preditos e daqueles que foram efetivamente avaliados pelo usuário. A medida  $F_1$ , por sua vez, é a média harmônica entre precisão e abrangência.

Todas essas métricas são dependentes dos diversos parâmetros do problema, como do tamanho da lista de recomendações  $N$ , da quantidade de vizinhos mais próximos  $k$ , e principalmente do banco de dados de teste. Como os artigos de referência não os disponibilizaram integralmente, serão estimados os valores de precisão, abrangência e medida  $F_1$  para o banco de dados da dupla.

Espera-se que a precisão, abrangência e consequentemente a medida  $F_1$  sejam maiores que 20%. Esses valores foram escolhidos por serem superiores aos de algoritmos puramente baseados em conteúdo ou em filtragem colaborativa (12, 25). Na prática, o resultado mais importante é a comparação entre os três métodos para um banco de dados de referência.

Os requisitos funcionais são suportados por requisitos não-funcionais, e estes são determinados pelas restrições sobre o projeto ou execução, tais como desenvolvimento e confiabilidade.

A biblioteca de sistemas de recomendação deverá poder ser utilizada por qualquer e-commerce que disponha de um banco de dados de clientes, produtos e histórico de compras, desde que o formato de entrada seja seguido.

Além disso um requisito não funcional é o desenvolvimento da biblioteca em tecnologias abertas (*open source*) que tenham um alto número de colaboradores, como a linguagem de programação estatística R, a fim de torná-la reutilizável por alunos ou e-commerces interessados.

Por fim, o sistema de recomendação deverá ser escalável e flexível no sentido de poder operar igualmente bem tanto em pequenas quanto em grandes bases de dados.

Apesar serem importantes parâmetros de um sistema de recomendação, a taxa de

recomendações por período de tempo e a escalabilidade estão intimamente relacionados ao orçamento do projeto (31). Pode-se obter virtualmente qualquer *throughput* desejado, contanto que haja investimento equivalente em infra-estrutura computacional. O mesmo não é válido para os parâmetros de qualidade da recomendação, que dependem tão somente dos algoritmos de sugestão. Por esse motivo, os requisitos de projeto não são pautados em taxa de recomendações por tempo.

Com os requisitos do sistema de recomendação definidos, devemos estruturar o seu relacionamento com o administrador do sistema. Para isto determinamos seus casos de uso, classes e atividades.

## 4.1 Diagrama de Casos de Uso

Os casos de uso se dividem em *Avaliar Performance*, *Configurar Banco de Dados*, *Recomendar UI*, *Recomendar UP*, *Recomendar FW*.

O caso de uso *Avaliar Performance* visa avaliar a performance do sistema de recomendação. Esse caso se relaciona com outros três casos de uso. O primeiro deles, *Mascarar Dados*, serve para esconder dados de alguns usuários-teste na matriz de avaliações, para, posteriormente, compará-los às recomendações calculadas pelo sistema. O segundo caso, *Dividir Banco de Treino*, tem o objetivo de dividir o banco de dados em dois, um para o aprendizado do sistema e o segundo para testes. O terceiro caso, *Devolver Indicadores*, torna os indicadores de performance acessíveis ao administrador do sistema.

O caso de uso *Configurar Banco de Dados* tem a utilidade de converter o banco de dados de entrada em matrizes. O caso *Ler Itens*, é o encarregado de ler o arquivo de itens fornecido pelo banco de dados, assim como *Ler Usuários* e *Ler Histórico* são para usuários e histórico de compras. Os casos de uso *Gerar Matriz de Atributos* e *Gerar Matriz de Avaliação* constroem as matrizes de acordo com dados lidos pelos casos de uso de leitura.

Já os casos de uso *Recomendar UI*, *Recomendar UP* e *Recomendar FW*, são os casos de uso em que se geram as recomendações pelos métodos baseados na correlação usuário-item, no perfil de usuários e na ponderação de atributos respectivamente. O método de recomendação baseado na ponderação de atributos necessita de outros três casos de uso, o *Normalizar Matriz*, onde se normaliza as colunas da matriz de distância entre atributos. Esta distância entre atributos é calculada pelos dois outros casos de uso, *Fazer Delta de Kronecker* e *Fazer Índice Jaccard*.

Estes casos de uso foram representados no diagrama de casos de uso (Figura 1).

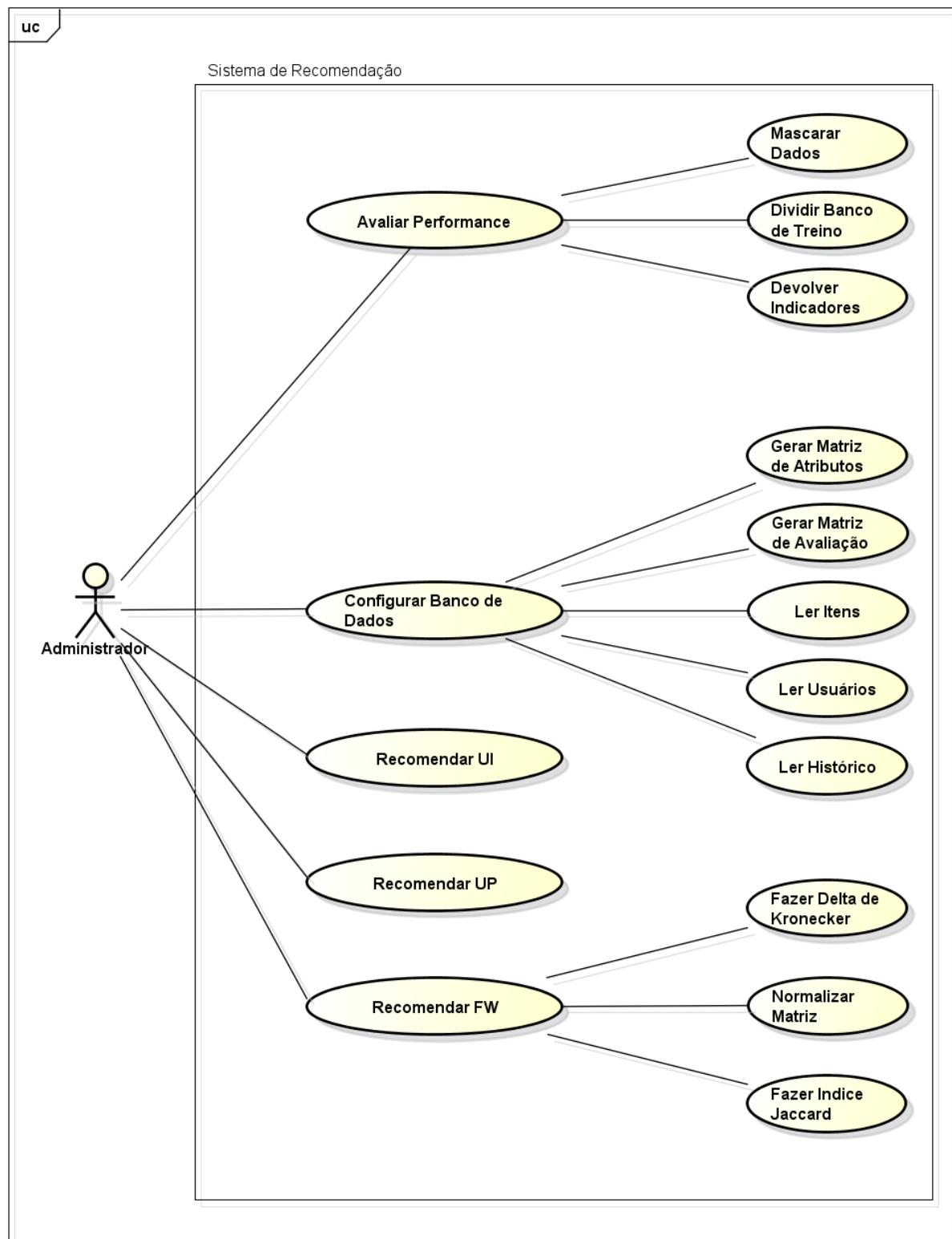


Figura 1 – Diagrama de casos de uso representando os relacionamentos entre o administrador e o sistema de recomendações.

## 4.2 Diagrama de Atividades

Para representar o fluxo de informação foi considerado o diagrama de atividades. Assim é possível visualizar os processos que vão desde a informação fornecida pelo usuário até a geração de recomendações.

O primeiro diagrama de atividade (Figura 2) representa o registro de uma avaliação por parte do cliente, onde o cliente solicita o item para visualizá-lo e a plataforma web faz o pedido de informações ao banco de dados. O banco de dados devolve as informações pedidas e a plataforma o exibe no dispositivo do cliente. Após a visualização, o cliente avalia o item, a plataforma web informa o banco de dados sobre a avaliação. Este registra a avaliação e a plataforma web confirma a avaliação finalizando o processo.

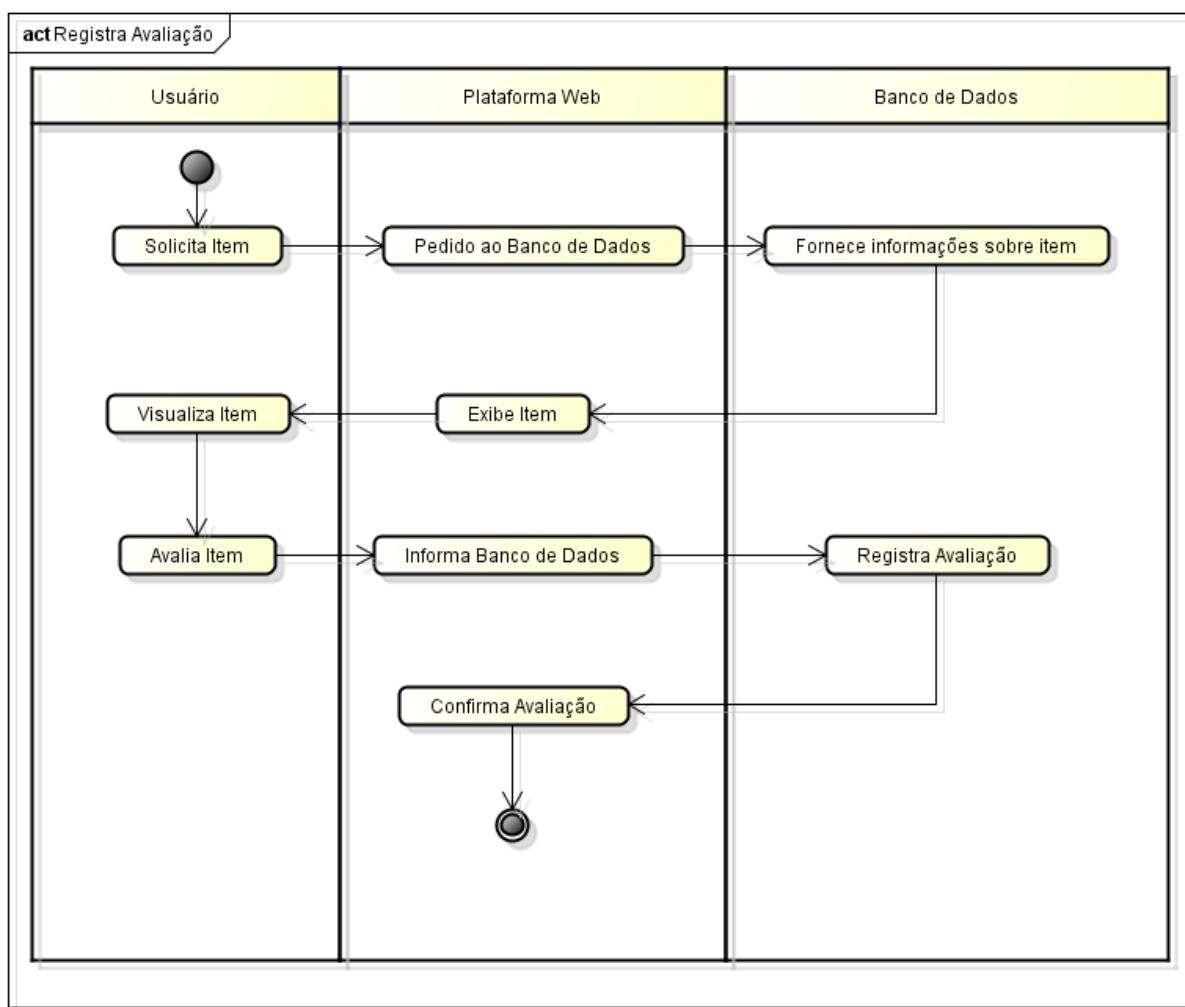


Figura 2 – Diagrama Atividades - Registro de Avaliação

O segundo diagrama (Figura 3), representa o fluxo de informação desde o pedido de uma recomendação pela plataforma web até a finalização da atividade pelo cliente. O primeiro passo é o pedido de uma recomendação, ao banco de dados, pela plataforma web. O banco de dados envia as informações necessárias para o sistema de recomendação que,

de acordo com as regras pré-estabelecidas, as calcula e retorna para o banco de dados. O banco de dados registra estas recomendações e informa a plataforma web, que envia a recomendação ao usuário. O usuário avalia o item recomendado e finaliza o processo.

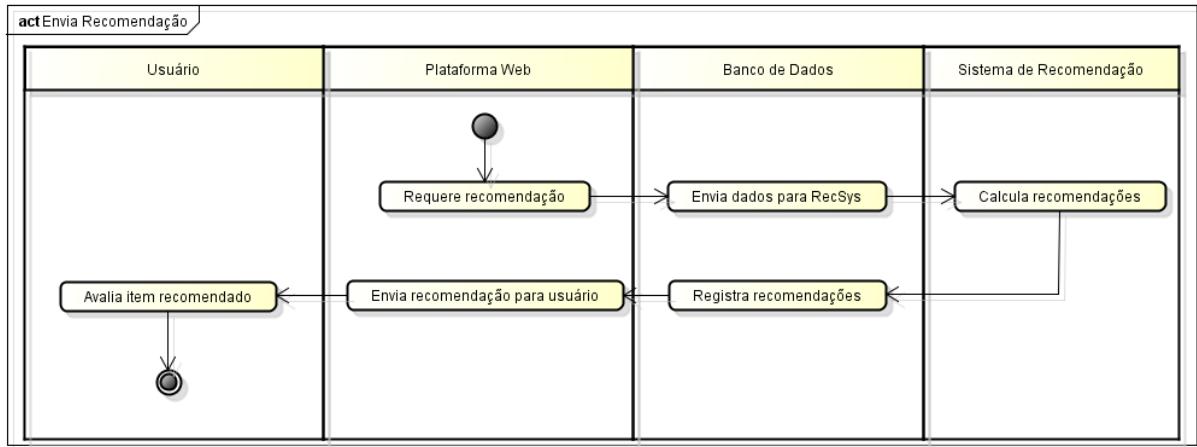


Figura 3 – Diagrama Atividades - Gerar Recomendação

## 4.3 Avaliação de Desempenho

De modo geral, sistemas de recomendação tem o objetivo de apresentar ao usuário itens pelos quais ele possa se interessar. O desempenho de um sistema de recomendação se mede, portanto, na qualidade com a qual ele executa essa tarefa.

Essa qualidade pode ser medida de diferentes maneiras, tal como pela medida de distância entre os produtos recomendados  $\hat{i}$  e aqueles que seriam efetivamente comprados  $i$  pelo cliente em uma validação cruzada (*cross validation*). Outras medidas de predição também podem ser utilizadas, a exemplo de trabalhos de recuperação de informação, tais como acurácia (*accuracy*), especificidade (*specificity*), precisão (*precision*), abrangência (*recall*), medida  $F_1$  ( $F_1$ -score), e outras (32).

No nosso Trabalho de Conclusão de Curso, serão utilizados precisão, abrangência, e medida  $F_1$ . Essas medidas foram escolhidas a fim de se poder estabelecer uma base comparativa com os textos de referência, que também as utilizam, e com algoritmos que fornecem recomendações do tipo lista *top-N* (33). As medidas estão sumarizadas na Tabela 5. As quantidades  $VP$ ,  $FP$ ,  $VN$  e  $FN$  significam o número de verdadeiro e falso positivos e o número de verdadeiro e falso negativos.

Por fim, avaliaremos o desempenho do sistema mediante a mudança nas variáveis de importância do problema, como por exemplo na quantidade de atributos utilizados na recomendação. O tempo de execução também será avaliado em função do algoritmo utilizado e do tamanho do banco de dados.

Tabela 5 – Avaliação de sistemas de predição

Medida	Fórmula	Significado
Precisão	$\frac{VP}{VP+FP}$	Porcentagem de casos positivos corretamente preditos.
Abrangência	$\frac{VP}{VP+FN}$	Porcentagem de casos positivos sobre aqueles que foram marcados como positivos.
$F_1$	$2 \cdot \frac{\text{Precisão} \cdot \text{Abrangência}}{\text{Precisão} + \text{Abrangência}}$	Média harmônica entre precisão e abrangência.

# 5 Detalhamento de Soluções

A fim de facilitar a compreensão dos métodos propostos neste trabalho, serão utilizadas as matrizes de avaliações  $\mathbf{R}$  e de atributos  $\mathbf{A}$  abaixo, adaptadas da Referência 25. Em todos os exemplos, considera-se valor mínimo  $M = 2$ . Os logaritmos são expressos em base 10 e todos os pesos  $w_f$ , descritos a seguir, são utilizados.

Tabela 6 – Avaliações  $r_{ui}$

	$i_1$	$i_2$	$i_3$	$i_4$	$i_5$	$i_6$
$u_1$	-	4	-	-	5	-
$u_2$	-	3	-	4	-	-
$u_3$	-	-	-	-	-	4
$u_4$	5	-	3	-	-	-

Tabela 7 – Atributos  $a_{if}$

	$f_1$	$f_2$	$f_3$	$f_4$
$i_1$	0	1	0	0
$i_2$	1	1	0	0
$i_3$	0	1	1	0
$i_4$	0	1	0	0
$i_5$	1	1	1	0
$i_6$	0	0	0	1

## 5.1 Algoritmo baseado na ponderação de atributos (FW)

O primeiro algoritmo que utilizaremos no sistema de recomendação, adaptado da Referência 12 e denominado ponderação de atributos, *feature weighting* ou FW, trata-se de um híbrido entre filtragem colaborativa e filtragem baseada em conteúdo. A partir da regressão linear de dados de uma rede social (*Internet Movie Database, IMDB*), extraem-se os pesos que determinam a importância de cada atributo dos itens, e é onde ocorre a filtragem colaborativa dos usuários. Após obtenção dos pesos, realiza-se a filtragem baseada em conteúdo para determinar os itens com maior similaridade, que são finalmente recomendados.

Na filtragem baseada em conteúdo, “cada item é representado por um vetor de atributos ou *features*”. A similaridade  $s_{ij}$  entre dois itens  $i$  e  $j$  é dada pela média ponderada

das distâncias entre as *features* dos itens:

$$s_{ij} = \sum_f w_f (1 - d_{fij}) \quad (5.1)$$

As distâncias entre os atributos  $d_f$  são determinadas conforme o tipo de dado avaliado e seu domínio, normalizadas no intervalo  $[0, 1]$ .

Para atributos literais, como categoria, marca, cor, etc., uma possível medida de distância é o delta de Kronecker descrito em 5.2. A similaridade entre as cores “azul” e “vermelho” é, nesse caso, 0, e sua distância é 1. O valor da distância é nulo se e somente se os atributos são idênticos.

Para atributos pertencentes a uma coleção finita de itens, tais como os atores participantes de um filme, é possível estabelecer a similaridade entre dois conjuntos a partir do índice Jaccard, descrito em 5.3. Neste caso, a similaridade entre os conjuntos  $\{\text{Al Pacino, Tom Hanks}\}$  e  $\{\text{Tom Hanks, Marlon Brando}\}$  é  $1/3$ , e a sua distância é  $2/3$ .

$$\delta_{mn} = \begin{cases} 1, & \text{se } m = n \\ 0, & \text{se } m \neq n \end{cases} \quad (5.2)$$

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} \quad (5.3)$$

Vale considerar a correlação entre atributos no cálculo das distâncias: a similaridade de duas marcas de calçado, por exemplo, é maior que a de duas marcas de produtos de categorias diferentes, mesmo que as marcas sejam distintas nos dois casos. Em uma primeira análise, todavia, utilizaremos para a maior parte das *features* as medidas de distância do delta de Kronecker 5.4 (Tabela 8) e do índice Jaccard 5.5. Isso significa que se os atributos de dois itens são idênticos, a distância é nula e portanto a similaridade é máxima. O sumário de algumas medidas de distância que podem ser utilizadas para casos específicos estão na Tabela 9.

$$\begin{aligned} d_{fij} &= 1 - \delta_{ij}^f \\ &= 1 - \delta_{a_{if} a_{jf}} \end{aligned} \quad (5.4)$$

$$\begin{aligned} d_{fij} &= 1 - J^f(i, j) \\ &= 1 - J(a_{if}, a_{jf}) \end{aligned} \quad (5.5)$$

Os pesos  $w_f$  são a priori desconhecidos. A Referência 12 os determina a partir de uma regressão linear do tipo 5.6, onde  $e_{ij}$  é o número de usuários que se interessam

tanto por  $i$  quanto por  $j$ . Esses valores permitem determinar “o julgamento humano de similaridade entre itens”, e pode ser calculado a partir da matriz de avaliações, conforme a equação 5.7 (Tabela 10). O operador booleano  $b_M$ , descrito pela Equação 5.8, nada mais é que uma ferramenta matemática para se poder extrair o número de usuários que avaliaram *positivamente* tanto  $i$  quanto  $j$  a partir de  $\mathbf{R}$ .

$$e_{ij} = w_0 + \sum_f w_f (1 - d_{fij}) \quad (5.6)$$

$$e_{ij} = \sum_u b_M (r_{ui} r_{uj}) \quad (5.7)$$

$$b_M (x) = \begin{cases} 1, & \text{se } x > M \\ 0, & \text{se } x \leq M \end{cases} \quad (5.8)$$

Desta forma, os pesos  $w_f$  são determinados a partir resolução do sistema de equações lineares 5.9 (Tabela 11). Apenas os pesos positivos e com valor absoluto expressivo (maior que um piso arbitrariamente escolhido a posteriori) são utilizados na recomendação.

$$w_0 + \sum_f w_f (1 - d_{fij}) = \sum_u b_0 (r_{ui} r_{uj}), \quad \forall i \neq j \quad (5.9)$$

Calcula-se a matriz de similaridade  $\mathbf{S}$  pela Equação 5.1 (Tabela 12) e recomendam-se os itens similares àqueles já comprados, segundo 5.10 (Tabela 13).

$$\hat{i}_u = \arg \max_{i \in \{i \mid r_{ui} > 0\}, j} s_{ij} \quad (5.10)$$

## 5.2 Algoritmo baseado no perfil de usuários (UP)

O segundo algoritmo, adaptado da Referência 25, é um híbrido entre filtragem colaborativa e filtragem baseada em conteúdo. Os atributos dos itens são ponderados no cálculo de similaridade, com pesos extraídos de um modelo de perfil de usuários, denominado *user profile* ou UP. Esse perfil leva em consideração o interesse dos usuários por *features*, indiretamente calculado a partir de seu interesse pelos itens.

Para se determinar a relevância de  $f$  para  $u$ , deve-se levar em conta não somente a frequência com a qual uma característica aparece, mas também o fato de algumas características estarem contidas na maioria dos itens. Determina-se, então, os pesos  $w_{uf}$ , que mostram a relevância de  $f$  para  $u$ , a partir da medida estatística TF-IDF (*term frequency-inverse document frequency*), presente em formulações de recuperação de informação e mineração de dados (Equação 5.13).

Tabela 8 –  $d_{ij}^f$ 

$f_1$	$i_1$	$i_2$	$i_3$	$i_4$	$i_5$	$i_6$	$f_2$	$i_1$	$i_2$	$i_3$	$i_4$	$i_5$	$i_6$
$i_1$	-	0	1	1	0	1	$i_1$	-	1	1	1	1	0
$i_2$	0	-	0	0	1	0	$i_2$	1	-	1	1	1	0
$i_3$	1	0	-	1	0	1	$i_3$	1	1	-	1	1	0
$i_4$	1	0	1	-	0	1	$i_4$	1	1	1	-	1	0
$i_5$	0	1	0	0	-	0	$i_5$	1	1	1	1	-	0
$i_6$	1	0	1	1	0	-	$i_6$	0	0	0	0	0	-
$f_3$	$i_1$	$i_2$	$i_3$	$i_4$	$i_5$	$i_6$	$f_4$	$i_1$	$i_2$	$i_3$	$i_4$	$i_5$	$i_6$
$i_1$	-	1	0	1	0	1	$i_1$	-	1	1	1	1	0
$i_2$	1	-	0	1	0	1	$i_2$	1	-	1	1	1	0
$i_3$	0	0	-	0	1	0	$i_3$	1	1	-	1	1	0
$i_4$	1	1	0	-	0	1	$i_4$	1	1	1	-	1	0
$i_5$	0	0	1	0	-	0	$i_5$	1	1	1	1	-	0
$i_6$	1	1	0	1	0	-	$i_6$	0	0	0	0	0	-

Tabela 9 – Medidas de distância entre alguns atributos

Atributo $f$	Domínio F	Distância $d_f$
Marca	Literal	$1 - \delta_{ij}^f$
Esporte	Literal	$1 - \delta_{ij}^f$
Gênero	Literal	$1 - \delta_{ij}^f$
Categoria	Conjunto Literal	$1 - J^f(i, j)$
Preço	$\mathbb{R}$	$\frac{ a_{if} - a_{jf} }{\max_{i,j}  a_{if} - a_{jf} }$
Data	$\mathbb{R}$ milissegundos a partir de epoch (34)	$\frac{ a_{if} - a_{jf} }{\max_{i,j}  a_{if} - a_{jf} }$

Tabela 10 –  $e_{ij}$ 

	$i_1$	$i_2$	$i_3$	$i_4$	$i_5$	$i_6$
$i_1$	-	0	1	0	0	0
$i_2$	0	-	0	1	1	0
$i_3$	1	0	-	0	0	0
$i_4$	0	1	0	-	0	0
$i_5$	0	1	0	0	-	0
$i_6$	0	0	0	0	0	-

Tabela 11 –  $w_f$ 

$w_0$	$w_1$	$w_2$	$w_3$	$w_4$
0.41	-0.22	-0.34	-0.03	-

Em nosso caso, TF ou *feature frequency* é a “similaridade intra-usuários”, igual ao número de vezes em que a *feature*  $f$  aparece no perfil do usuário  $u$  (Equação 5.11, Tabela 14). Se o usuário avaliou *positivamente* algum item  $r_{ui}$ , tal que  $r_{ui}$  é superior a um valor mínimo  $M$ , considera-se que  $u$  tem interesse  $\text{TF}_{uf}$  nos atributos  $f$  dos itens  $i$ , representados por  $a_{if}$ .

$$\text{TF}_{uf} = \sum_i b_M(r_{ui} | a_{if}) \quad (5.11)$$

O termo IDF ou *inverse user frequency* é a “dissimilaridade inter-usuários”, relacionada com o inverso da frequência de um atributo  $f$  dentro de todos os usuários (Equação 5.12, Tabela 15).

$$\text{IDF}_f = \log \left( \frac{|\mathcal{U}|}{\sum_u b_0(\text{TF}_{uf})} \right) \quad (5.12)$$

Os pesos  $w_{uf}$ , obtidos na TF-IDF 5.13 (Tabela 16), são utilizados para calcular a similaridade  $s_{uv}$  entre dois usuários  $u$  e  $v$ , conforme as Equações 5.14 e 5.15 (Tabela 17).

$$w_{uf} = \text{TF}_{uf} \text{IDF}_f \quad (5.13)$$

$$s_{uv} = \frac{\sum_{f \in \mathcal{F}_{uv}} w_{uf} w_{vf}}{\sqrt{\sum_{f \in \mathcal{F}_{uv}} w_{uf}^2} \sqrt{\sum_{f \in \mathcal{F}_{uv}} w_{vf}^2}} \quad (5.14)$$

$$\begin{aligned} \mathcal{F}_{uv} &= \mathcal{F}_u \cap \mathcal{F}_v \\ \mathcal{F}_u &= \{f \in \mathcal{F} \mid t_{uf} > 0\} \end{aligned} \quad (5.15)$$

Dispondo-se de  $\mathbf{S}$ , selecionam-se os  $k$  vizinhos mais próximos  $v_k^u$  com maior similaridade  $s_{uv}$ , dentre todos  $v \neq u$ . Posteriormente, determina-se o conjunto  $\mathcal{I}_{v_k^u} = \{i \mid r_{v_k^u i} > M\}$  de itens  $i$  avaliados positivamente por  $v_k^u$ . Em 5.16 avalia-se a frequência total  $f_{uf}$  dos atributos  $f$  para os itens de  $\mathcal{I}_{v_k^u}$  (Tabela 18).

$$f_{uf} = \sum_{i \in \mathcal{I}_{v_k^u}} b_0(a_{if}) \quad (5.16)$$

Por fim, a partir da Equação 5.17 calcula-se o peso  $\omega_{ui}$  (Tabela 19) de cada item e gera-se a lista dos *top-N* produtos a serem recomendados para o usuário  $u$ , conforme 5.18 (Tabela 20).

$$\omega_{ui} = \sum_f a_{if} f_{uf} \quad (5.17)$$

Tabela 12 –  $s_{ij}$ 

	$i_1$	$i_2$	$i_3$	$i_4$	$i_5$	$i_6$
$i_1$	-	0.44	1.00	0.93	0.51	0.17
$i_2$	0.44	-	0.51	0.44	1	-0.32
$i_3$	1.00	0.51	-	1.00	0.44	0.24
$i_4$	0.93	0.44	1.00	-	0.51	0.17
$i_5$	0.51	1.00	0.44	0.51	-	-0.25
$i_6$	0.17	-0.33	0.24	0.17	-0.25	-

Tabela 13 –  $\hat{i}_u$  (FW)

$u_1$	$u_2$	$u_3$	$u_4$
3	5	3	4

Tabela 14 –  $\text{TF}_{uf}$ 

	$f_1$	$f_2$	$f_3$	$f_4$
$u_1$	2	2	1	0
$u_2$	1	2	0	0
$u_3$	0	0	0	1
$u_4$	0	2	1	0

Tabela 15 –  $\text{IDF}_f$ 

$f_1$	$f_2$	$f_3$	$f_4$
0.30	0.12	0.30	0.60

Tabela 16 –  $w_{uf}$ 

	$f_1$	$f_2$	$f_3$	$f_4$
$u_1$	0.60	0.25	0.30	0
$u_2$	0.30	0.25	0	0
$u_3$	0	0	0	0.60
$u_4$	0	0.25	0.30	0

Tabela 17 –  $s_{uv}$ 

	$u_1$	$u_2$	$u_3$	$u_4$
$u_1$	-	0.96	0	1
$u_2$	0.96	-	0	1
$u_3$	0	0	-	0
$u_4$	1	1	0	-

$$\hat{i}_u = \arg \max_{i \in \{i \mid r_{ui} = 0\}} \omega_{ui} \quad (5.18)$$

### 5.3 Algoritmo baseado na correlação usuário-item (UI)

Este método se trata de uma variante da solução UP, e também está embasado no cálculo da preferência do usuário por *features*, medida através do seu interesse pelos itens. O algoritmo UI utiliza as matrizes de correlação ponderada entre usuários e atributos **W** e a matriz de atributos dos itens **A** no cálculo da correlação usuário-item.

A lista dos  $N$  produtos a serem recomendados decorre portanto do cálculo de  $\omega_{ui}$  (Equação 5.19, Tabela 21) e da escolha dos itens que maximizem essa variável para cada usuário (Equação 5.18, Tabela 22).

$$\omega_{ui} = \sum_f w_{uf} a_{if} \quad (5.19)$$

Ao passo que o método *UP* recomenda itens a partir dos  $k$  vizinhos mais próximos, o algoritmo *UI* busca os itens com *features* mais similares aos atributos pelos quais  $u$  se interessa, diretamente através da matriz de atributos.

Espera-se que esse tipo de recomendação forneça sugestões de qualidade similar ao algoritmo original, pois os dois tem a mesma fundamentação inicial. Pode-se observar que, para o exemplo-base, ambos algoritmos forneceram a mesma recomendação para três de quatro usuários.

Tabela 18 –  $f_{uf}$ 

	$f_1$	$f_2$	$f_3$	$f_4$
$u_1$	0	2	1	0
$u_2$	1	3	2	0
$u_3$	1	2	0	0
$u_4$	2	3	1	0

Tabela 19 –  $\omega_{ui}$  (UP)

	$i_1$	$i_2$	$i_3$	$i_4$	$i_5$	$i_6$
$u_1$	2	0	3	0	0	0
$u_2$	3	0	5	0	6	0
$u_3$	0	3	0	2	0	0
$u_4$	0	5	0	3	6	0

Tabela 20 –  $\hat{i}_u$  (UP)

$u_1$	$u_2$	$u_3$	$u_4$
3	5	2	5

Tabela 21 –  $\omega_{ui}$  (UI)

	$i_1$	$i_2$	$i_3$	$i_4$	$i_5$	$i_6$
$u_1$	0.25	0.85	0.55	0.25	1.15	0
$u_2$	0.25	0.55	0.25	0.25	0.55	0
$u_3$	0	0	0	0	0	0.60
$u_4$	0.25	0.25	0.55	0.25	0.55	0

Tabela 22 –  $\hat{i}_u$  (UI)

$u_1$	$u_2$	$u_3$	$u_4$
3	5	-	5

# 6 Desenvolvimento da biblioteca

## 6.1 Recursos acadêmicos

A principal contribuição da Escola Politécnica para o projeto veio das disciplinas de programação para automação (PMR2300 – Computação para Automação e PMR2440 – Programação para Automação) e de banco de dados (PMR2490 – Sistemas de Informação). Além disso, a disciplina optativa PMR2728 – Teoria de Probabilidades em Inteligência Artificial e Robótica abordou a temática do aprendizado de máquina e dos sistemas de recomendação.

Além das disciplinas do curso de Engenharia Mecatrônica, diversos recursos extra-curriculares foram de fundamental importância para o sucesso deste trabalho. Foram aplicados aprendizados práticos de quatro cursos da plataforma online Coursera (<<https://www.coursera.org/>>), sejam relacionados a teoria dos sistemas de recomendação, sejam relacionados a configuração de servidores na Amazon Web Services.

O curso “Redes: Amigos, Dinheiro e Bytes” (Networks: Friends, Money, and Bytes – <<https://www.coursera.org/course/friendsmoneybytes>>), teve papel importante na introdução a temas ligados à rede mundial de computadores. Mais especificadamente, a aula 4 aborda, de maneira simples mas repleta de exemplos, a temática de sugestão de itens através da pergunta “Como o Netflix recomenda filmes?”. Essa aula ajudou-nos a compreender a teoria por trás do algoritmo de recomendação do Netflix detalhado na Referência 35.

Outro curso que influenciou diretamente o nosso Trabalho de Conclusão de Curso foi “Computação para Análise de Dados” (Computing for Data Analysis – <<https://www.coursera.org/course/compdata>>). As quatro semanas de aula ensinaram a leitura de dados formatados em R, o tratamento de dados, o uso de modelos estatísticos, como por exemplo métodos de regressão linear e polinomial, a aplicação de cálculos vetorizados e a construção de gráficos e tabelas.

Aliado a essas aulas, aprendemos também o paradigma funcional, amplamente utilizado em R, durante as sete semanas de “Princípios de Programação Funcional em Scala” (Functional Programming Principles in Scala – <<https://www.coursera.org/course/progfun>>). Todo o pacote foi construído em torno desse padrão de programação, visto que usuários de bibliotecas desejam utilizar métodos e funções genéricas para construção de resultados específicos.

Por fim, o curso de doze semanas de duração “Engenharia de Startup” (Startup Engineering – <<https://www.coursera.org/course/startup>>) nos ensinou a trabalhar com

diversas ferramentas de software necessárias para a realização dos testes de desempenho dos algoritmos. Utilizamos máquinas virtuais, linha de comando Unix, versionamento de código em `git` e editores de texto sem interface gráfica (tais como vi e Emacs). Além disso, o *setup* de máquinas virtuais na Amazon Web Services também era abordada no curso, facilitando a configuração do ambiente de testes e a automatização desse processo.

## 6.2 Ferramentas utilizadas

A programação da biblioteca computacional se deu por meio do ambiente de desenvolvimento integrado RStudio versão 0.98.953 (<http://www.rstudio.com/>). Esse IDE inclui console, editor de texto e corretor de sintaxe que suporta a execução direta de código, bem como ferramentas para visualização de gráficos, depuração de erros e gerenciamento de espaço de trabalho. Além disso, o RStudio está disponível via licença de código aberto AGPLv3 (Afferro General Public License version 3) para os principais sistemas operacionais (Windows, Mac e Linux).

## 6.3 Métodos computacionais

### 6.3.1 Estrutura da biblioteca

A biblioteca está estruturada em quatro seções principais: `db`, onde está o banco de dados MovieLens 100k, `methods`, onde estão os algoritmos de recomendação, `results`, onde estão os métodos de avaliação de qualidade e `setup`, onde estão codificadas funções diversas, tais como leitura de banco de dados e cálculo de medidas de distância.

Código 6.1 – Estrutura da biblioteca

---

```
recsys/
|-- db
|   '-- ml-100k
|       |-- u.data
|       |-- u.item
|       |-- u.user
|       |-- ...
|-- methods
|   |-- common
|       '-- up_ui_w.R
|   |-- fw.R
|   |-- ui.R
|   '-- up.R
|-- results
```

---

```

|   |-- benchmark.R
|   |-- performance.R
|   '-- run_tests.R
'-- setup
    |-- functions.R
    '-- setup.R

```

---

As principais funções da biblioteca estão descritas na documentação do Apêndice A, e o código pode ser obtido através do endereço <<https://github.com/aviggiano/tcc/tree/master/recsys>>.

### 6.3.2 Algoritmo baseado na ponderação de atributos (FW)

O algoritmo de ponderação de atributos possui cinco etapas:

#### 6.3.2.1 Determinação de $e_{ij}$

Em forma matricial, a Equação 5.7 pode ser descrita da forma 6.1.

$$\mathbf{E} = \mathbf{b}_M \left( \mathbf{R}^T \right) \mathbf{b}_M \left( \mathbf{R} \right) \quad (6.1)$$

Em R, isso se traduz por uma simples instrução:

---

Código 6.2 – Determinação de  $e_{ij}$

---

```
e = b(t(r), M) %*% b(r, M)
```

---

#### 6.3.2.2 Determinação de $d_{ij}^f$

Para o cálculo genérico de FW, a medida de distância padrão utilizada é a distância  $L_1$  entre as duas *features* (Equação 6.2, Código 6.3). Para o cálculo específico do banco de dados 100k-IMDB, as medidas de distância (Equação 6.3, Código 6.4) são determinadas segundo a Tabela TODO.

$$d_{ij}^f = |a_{if} - a_{jf}| \quad (6.2)$$

---

Código 6.3 – Determinação de  $d_{ij}^f$  genérico

---

```
d[i,j,] = abs(a[i,]-a[j,])
```

---

$$\begin{aligned}
d_{ij}^f &= |a_{if} - a_{jf}|, \quad f \in \{1, 21, 23, 24, 25\} \\
d_{ij}^f &= J(a_{if}, a_{jf}), \quad f = 2, \dots, 20
\end{aligned} \quad (6.3)$$

---

 Código 6.4 – Determinação de  $d_{ij}^f$  específico
 

---

```

d[i,j,1] = abs(a[i,21]-a[j,21])
d[i,j,2] = abs(a[i,22]-a[j,22])
d[i,j,3] = abs(a[i,23]-a[j,23])
d[i,j,4] = abs(a[i,24]-a[j,24])
d[i,j,5] = abs(a[i,25]-a[j,25])
d[i,j,6] = abs(a[i,1]-a[j,1])
d[i,j,7] = 1-jaccard(a[i,2:20],a[j,2:20])
    
```

---

### 6.3.2.3 Determinação de $w_f$

O vetor  $\mathbf{w}$  é obtido através da regressão linear de Equação 5.9. Na linguagem R, basta aplicar o método `lm`, conforme o Código 6.5.

---

 Código 6.5 – Determinação de  $w_f$ 


---

```

# linear fit e ~ w0 + w (1-d)
D = 1-d
lm.W = lm(e ~ D, x=FALSE, y=FALSE, model=FALSE, qr=FALSE)
W = as.vector(lm.W$coefficients)
    
```

---

### 6.3.2.4 Determinação de $s_{ij}$

Para o cálculo da matriz de similaridade, escolhemos os pesos  $w_f > 0$  e aplicamos a Equação 5.1, para todo  $f > 0$ .

---

 Código 6.6 – Determinação de  $s_{ij}$ 


---

```

if(W[f] > 0) s = s + W[f] * (1-d[,f-1])
    
```

---

### 6.3.2.5 Determinação de $\hat{i}_u$

Os itens a serem recomendados são aqueles para os quais  $s_{ij}$  tem valor máximo e que ainda não tenham sido comprados pelo usuário. Em alguns casos específicos, é desejado que o cliente possa receber recomendações do mesmo produto, e deve-se indicar através do argumento `repick` o valor TRUE.

---

 Código 6.7 – Determinação de  $\hat{i}_u$ 


---

```

not.yet.chosen = if(repick) I.length else union(which(is.na(r[u,])),
      which(r[u,]==0))
iu = intersect(not.yet.chosen, index.top.N(s[i,], N))
    
```

---

### 6.3.3 Algoritmo baseado no perfil de usuários (UP)

O algoritmo baseado no perfil de usuários atributos possui quatro etapas principais:

#### 6.3.3.1 Determinação de $w_{uf}$

O cálculo de  $w_{uf}$  é dado pela TF-IDF de Equação 5.13. Pode-se obter  $\text{TF}_{uf}$  a partir do Código 6.8, e  $\text{IDF}_f$  a partir do Código 6.9. Finalmente, a multiplicação desses dois elementos resulta na correlação usuário-atributo (Código 6.10).

---

Código 6.8 – Determinação de  $\text{TF}_{uf}$

---

```
TF[u,f] = sum(
    b(r[,u] * a[,f], M),
    na.rm = TRUE)
```

---



---

Código 6.9 – Determinação de  $\text{IDF}_f$

---

```
IDFbar = sapply(1:length(TF[,]), function(f) sum(b0(TF[,f])))
IDF = log(length(U)/IDFbar, 10)
```

---



---

Código 6.10 – Determinação de  $w_{uf}$

---

```
w[,u] = TF[,u]*IDF
```

---

#### 6.3.3.2 Determinação de $s_{uv}$

Para obter a matriz de similaridade intra-usuários, basta aplicar a Equação 5.14.

---

Código 6.11 – Determinação de  $s_{uv}$

---

```
Fu = which(TF[,]>0)
Fv = which(TF[,]>0)
Fuv = intersect(Fu, Fv)

s[u,v] = if(length(Fuv) == 0)
          0
        else
          sum((w[,u]*w[,v])[Fuv], na.rm=TRUE) /
          ( sqrt(sum((w[,u]*w[,u])[Fuv], na.rm=TRUE)) *
            sqrt(sum((w[,v]*w[,v])[Fuv], na.rm=TRUE)) )
```

---

### 6.3.3.3 Determinação de $\omega_{ui}$

O cálculo da correlação usuário-item se dá pelo método dos vizinhos mais próximos. Selecionam-se aqueles usuários  $v_u^k$  com maior similaridade  $s_{uv}$  com relação a  $u$  e aplicam-se as Equações 5.16 e 5.17.

Código 6.12 – Determinação de  $\omega_{ui}$

---

```
vuk = index.top.N.not.self(s[u,], u, k, Utest)
Ivuk = unique(unlist(lapply(vuk, function(v) which(r[v,>M]))))

fuf[u,f] = sum(
  b0(a[,f])[Ivuk],
  na.rm = TRUE)

omega = fuf %*% t(a)
```

---

### 6.3.3.4 Determinação de $\hat{i}_u$

Os itens a serem recomendados são aqueles para os quais  $\omega_{ui}$  tem valor máximo e que ainda não tenham sido comprados pelo usuário. Assim como no método FW, caso o cliente possa receber recomendações do mesmo produto, deve-se indicar o argumento `repick` como `TRUE`.

Código 6.13 – Determinação de  $\hat{i}_u$

---

```
repeated = if(repick) NULL else which(r[u,> 0])
iu = index.top.N(omega[u,], N, repeated)
```

---

## 6.3.4 Algoritmo baseado na correlação usuário-item (UI)

Para o algoritmo baseado na correlação usuário-item, a etapa 6.3.3.1 é idêntica à do método UP. Para o algoritmo UI, o cálculo de  $\omega_{ui}$  se dá pela Equação matricial 6.4, de Código 6.14.

$$\Omega = \mathbf{W} \mathbf{A}^T \quad (6.4)$$

Código 6.14 – Determinação de  $\omega_{ui}$

---

```
omega = w %*% t(a)
```

---

## 6.4 Ambiente de testes

Inicialmente, realizamos os testes de qualidade nos nossos próprios computadores pessoais. Todavia, a execução de testes sucessivos exigia muita capacidade computacional, principalmente quanto a memória virtual.

A alocação de objetos e matrizes na memória RAM é muito custosa, principalmente na etapa de determinação de medidas de distância  $d_{ij}^f$  para o algoritmo FW (Equação 5.1). Uma matriz de dimensão  $|\mathcal{I}| \times |\mathcal{I}| \times |\mathcal{F}|$  possui  $1682 \times 1682 \times 25$  elementos (cerca de 71 milhões), e ocupa aproximadamente 500 MB de memória. No total, 4 GB de memória RAM são utilizadas durante todos os testes, fazendo-se necessário o uso máquinas dedicadas.

Por essa razão, realizamos todas as etapas de recomendação e avaliação de qualidade em máquinas *memory-optimized* nos servidores da Amazon Web Services (<http://aws.amazon.com/>). Visto que o serviço é cobrado por hora-máquina, desenvolvemos um *script* de inicialização para instalar todos os pacotes de programação e execução imediata dos testes, permitindo assim reduzir os custos da análise.

As etapas de configuração do ambiente de testes envolvem o cadastro na Amazon Web Services, a criação de uma máquina virtual, a instalação das ferramentas de programação e o descarregamento do código de testes.

Após o cadastro na AWS, deve-se seguir os seguintes passos para a criação de uma máquina virtual:

1. Login na plataforma (Figura 4);
2. Acesso ao serviço de máquinas virtuais Elastic Compute Cloud ou EC2. Para criar uma nova máquina, basta clicar em *Launch Instance* (Figura 5);
3. Escolha da configuração do software da máquina. É possível escolher entre diversos sistemas operacionais, versões e distribuições. Para avançar, deve-se clicar em *Select*. No nosso caso, escolhemos a configuração *Amazon Linux AMI 2014.09.1 (HVM)*, em virtude da facilidade de se instalar pacotes adicionais (Figura 6);
4. Escolha do tipo de máquina. No nosso caso, escolhemos uma máquina otimizada para memória RAM. Em seguida, avançamos em *Next: Configure Instance Details*. Nos *Steps 3, 4 e 5* do serviço, não modificamos nenhuma opção pré-configurada (Figura 7);
5. Criação da chave privada a ser utilizada para conexão com a máquina. Depois da criação de uma nova chave, pode-se descarregá-la através de *Download Key Pair* e finalmente iniciar a máquina em *Launch Instances* (Figura 8);

6. Inicialização da máquina e do DNS público. Esse é o endereço de conexão com a máquina, e pode ser obtido na seção *Description* → *Public DNS* (Figura 9).

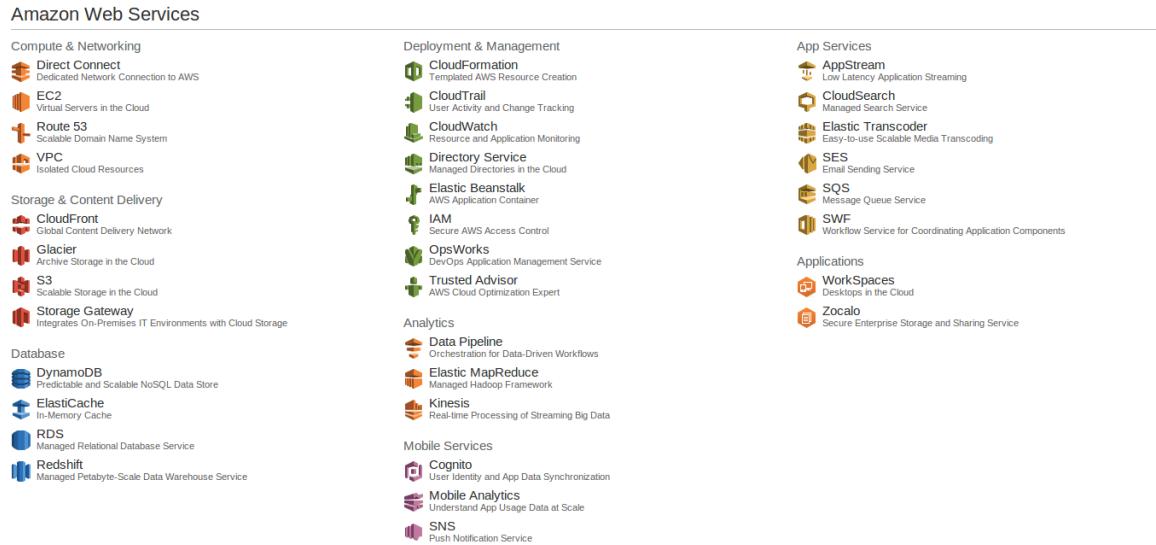


Figura 4 – Serviços da Amazon Web Services

O painel de controle EC2 exibe as seguintes informações:

- Resources:** Usando recursos na US East (N. Virginia) region:
  - 0 Running Instances
  - 0 Volumes
  - 1 Key Pair
  - 0 Placement Groups
  - 0 Elastic IPs
  - 0 Snapshots
  - 0 Load Balancers
  - 5 Security Groups
- Create Instance:** Botão "Launch Instance". Nota: As instâncias serão lançadas na US East (N. Virginia) region.
- Service Health:**
  - Service Status:** US East (N. Virginia): This service is operating normally.
  - Availability Zone Status:**
    - us-east-1a: Availability zone is operating normally.
    - us-east-1b: Availability zone is operating normally.
    - us-east-1c: Availability zone is operating normally.
    - us-east-1e: Availability zone is operating normally.
- Scheduled Events:** US East (N. Virginia): No events.

Figura 5 – Serviços de Elastic Compute Cloud (EC2)

Uma vez criada a máquina, deve-se utilizar a chave privada para realizar um *secure shell* e conectar-se remotamente ao EC2. A partir de um computador pessoal dotado de um interpretador de comandos **bash**, utilizamos a seguinte instrução:

**Step 1: Choose an Amazon Machine Image (AMI)**

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. You can select an AMI provided by AWS, our user community, or the AWS Marketplace; or you can select one of your own AMIs.

Quick Start	Cancel and Exit
My AMIs	
AWS Marketplace	
Community AMIs	
<input type="checkbox"/> Free tier only ⓘ	
 <b>Amazon Linux AMI 2014.09.1 (HVM)</b> - ami-b66ed3de	<b>Select</b> 64-bit
The Amazon Linux AMI is an EBS backed image. It includes the 3.14 kernel, Ruby 2.1, PHP 5.5, PostgreSQL 9.3, Docker 1.2, the AWS command line tools, and repository access to many other packages.	
 <b>Red Hat Enterprise Linux 7.0 (HVM), SSD Volume Type</b> - ami-a8d369c0	<b>Select</b> 64-bit
Red Hat Enterprise Linux version 7.0 (HVM). EBS General Purpose (SSD) Volume Type	
 <b>SuSE Linux Enterprise Server 12 (HVM), SSD Volume Type</b> - ami-aeb532c6	<b>Select</b> 64-bit
SuSE Linux Enterprise Server 12 (HVM). EBS General Purpose (SSD) Volume Type. Public Cloud, Advanced Systems Management, Web and Scripting, and Legacy modules enabled.	
 <b>Ubuntu Server 14.04 LTS (HVM), SSD Volume Type</b> - ami-9ea0a1cf0	<b>Select</b> 64-bit
Ubuntu Server 14.04 LTS (HVM). EBS General Purpose (SSD) Volume Type. Support available from Canonical ( <a href="http://www.ubuntu.com/cloud/services">http://www.ubuntu.com/cloud/services</a> ).	
 <b>Microsoft Windows Server 2012 R2 Base</b> - ami-ba13ab2d	<b>Select</b> 64-bit
Microsoft Windows 2012 R2 Standard edition with 64-bit architecture. [English]	
 <b>Microsoft Windows Server 2012 R2 with SQL Server Web</b> - ami-9c0bb3f4	<b>Select</b> 64-bit
Microsoft Windows Server 2012 R2 Standard edition, 64-bit architecture. Microsoft SQL Server 2014 Web edition. [English]	
 <b>Microsoft Windows Server 2012 R2 with SQL Server Standard</b> - ami-a415aecc	<b>Select</b> 64-bit
Microsoft Windows Server 2012 R2 Standard edition, 64-bit architecture. Microsoft SQL Server 2014 Standard edition. [English]	
 <b>Microsoft Windows Server 2012 Base</b> - ami-3214ac5a	<b>Select</b>
Microsoft Windows Server 2012 Standard edition with Itanium architecture. [English]	

Figura 6 – Escolha do tipo de configuração do software da máquina

**Step 2: Choose an Instance Type**

Filter by: All instance types Current generation ShowHide Columns

Currently selected: r3.large (6.5 ECUs, 2 vCPUs, 2.5 GHz, Intel Xeon E5-2670v2, 15 GiB memory, 1 x 32 GiB Storage Capacity)

Family	Type	vCPUs ⓘ	Memory (GB)	Instance Storage (GB) ⓘ	EBS-Optimized Available ⓘ	Network Performance ⓘ
General purpose	t2.micro	1	1	EBS only	-	Low to Moderate
General purpose	t2.small	1	2	EBS only	-	Low to Moderate
General purpose	t2.medium	2	4	EBS only	-	Low to Moderate
General purpose	m3.medium	1	3.75	1 x 4 (SSD)	-	Moderate
General purpose	m3.large	2	7.5	1 x 32 (SSD)	-	Moderate
General purpose	m3.xlarge	4	15	2 x 40 (SSD)	Yes	High
General purpose	m3.2xlarge	8	30	2 x 80 (SSD)	Yes	High
Compute optimized	c3.large	2	3.75	2 x 16 (SSD)	-	Moderate
Compute optimized	c3.xlarge	4	7.5	2 x 40 (SSD)	Yes	Moderate
Compute optimized	c3.2xlarge	8	15	2 x 80 (SSD)	Yes	High
Compute optimized	c3.4xlarge	16	30	2 x 160 (SSD)	Yes	High
Compute optimized	c3.8xlarge	32	60	2 x 320 (SSD)	-	10 Gigabit
GPU instances	g2.2xlarge	8	15	1 x 60 (SSD)	Yes	High
Memory optimized	r3.large	2	15	1 x 32 (SSD)	-	Moderate
Memory optimized	r3.xlarge	4	30.5	1 x 80 (SSD)	Yes	Moderate

**Cancel** **Previous** **Review and Launch** **Next: Configure Instance Details**

Figura 7 – Escolha do tipo máquina

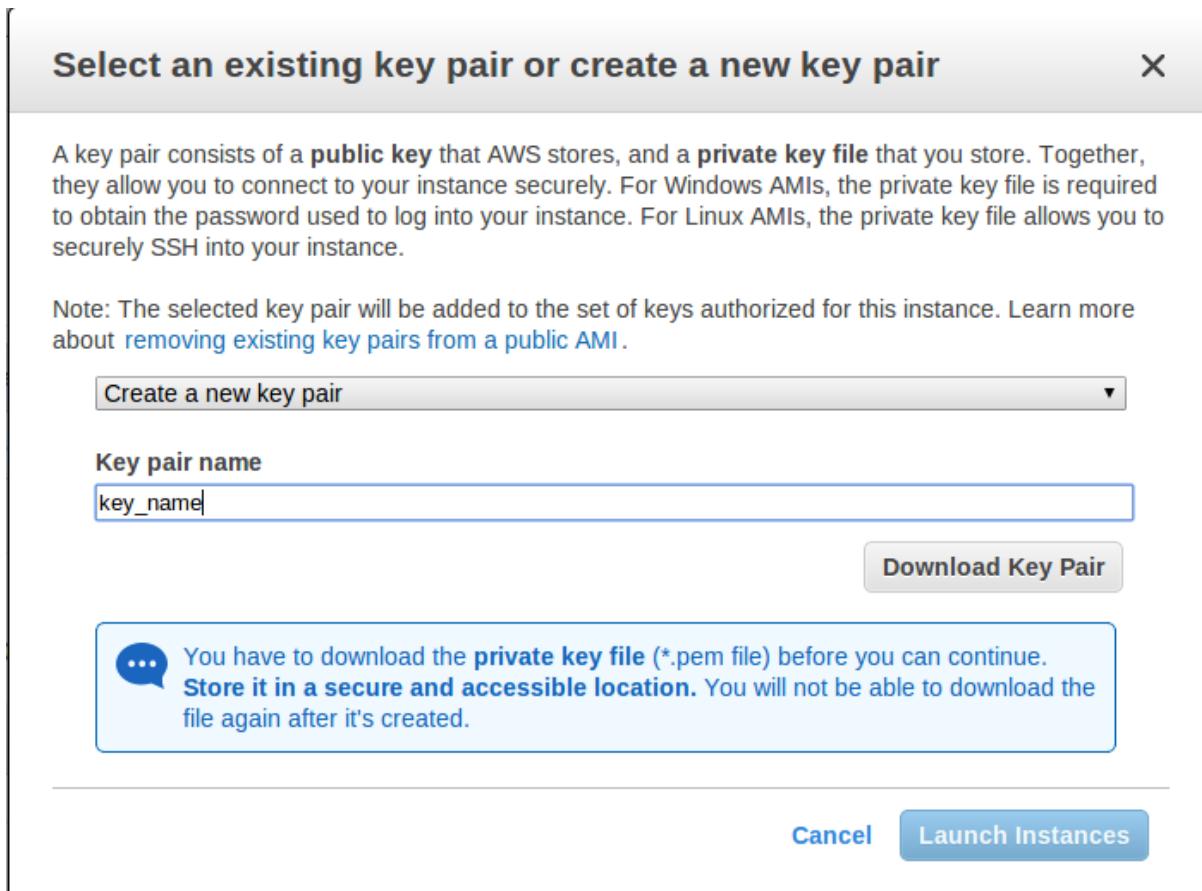


Figura 8 – Criação da chave privada

The screenshot shows the AWS Instances dashboard. A table lists one instance: "i-129954f3" (t2.micro, running, us-east-1a). The "Launch Instance" button is highlighted. Below the table, a detailed view of the instance "i-129954f3" is shown, including its Public DNS ("ec2-54-88-186-106.compute-1.amazonaws.com"), Public IP ("54.88.186.106"), Key Name ("vai0"), and Launch Time ("November 5, 2014 5:38:29 AM UTC-2").

Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS	Public IP	Key Name	Monitoring	Launch Time	Security Groups
	i-129954f3	t2.micro	us-east-1a	running	Initializing	None	ec2-54-88-186-106.com...	54.88.186.106	vai0	disabled	November 5, 2014 5:38:29 A...	launch-wizard-5, view rules

Figura 9 – Criação da máquina e do DNS público (*Public DNS*)

Código 6.15 – Secure shell para conexão com a máquina virtual EC2

---

```
ssh -i ~/Downloads/key_name.pem
ec2-user@ec2-54-88-186-106.compute-1.amazonaws.com
```

---

Em seguida, para a automatização do ambiente de testes e rápida configuração de novas máquinas, criamos um arquivo `script.sh` na linguagem de programação `bash`. Esse `script` instala os pacotes `R` e `git` e cria a chave pública necessária para acessar o servidor onde o código da biblioteca está hospedado. Em virtude de sua popularidade, utilizamos o serviço de hospedagem de códigos abertos GitHub (<<https://github.com/aviggiano/tcc>>).

Código 6.16 – *Script* de configuração do ambiente de testes

---

```
#!/usr/bin/env bash
sudo su          # login como administrador. necessário para instalar
                  pacotes no sistema operacional
yes | yum install R  # instala o pacote R no linux
yes | yum install git # instala o git para descarregar os metodos de
                      recomendacao
ssh-keygen -t rsa    # gera a chave para conectar-se ao GitHub
cat ~/.ssh/id_rsa.pub # imprime a chave publica, que deve ser adicionada nas
                      configuracoes do GitHub
```

---

A saída do `script` é uma chave no seguinte formato:

Código 6.17 – Chave pública

---

```
ssh-rsa AAAAB3NzaC1yc2EAAAQABAAQCB/bxcszoyHzyqtTXp4f11Q30T58Lsb7QLx+7nQ6
y00I0WhK+r5ynSVi0BpTC+2hMrlg1rZTC1ED7Nb+SI9bRvf+1UYW0iVUXtwAVColMNBDIfE7QCWbJm
TmmBLcv9PIoCAvCfrxBh+f1W3hXG388/LIEjZJckPYogho0jPAnFv3IXAGtVniV6cBcTTfKPUnX+n+
6xiqnf4tYQpmPW/mnxk9s3bbEmcE1eYJkrE2IWdzy6EBnR9D4cBW5D8/VMM54xMJzugWZ0//sIjLLT
0oFTTrroiwr+OX2DxqFdgCy8Agx1WZTeGhBAW1nvIVr5WVcWVBSzBCZfg8mYe+zYnbwl ec2-user@
ip-10-168-40-38
```

---

Após a obtenção da chave, deve-se cadastrá-la na página correspondente do GitHub (<<https://github.com/settings/ssh>>), como mostra a Figura 10.

Uma vez tendo habilitado a máquina virtual da AWS para manipulação do repositório da biblioteca, pode-se descarregar o código e executar o `script` de testes de qualidade:

Código 6.18 – Script de execução dos testes de qualidade

---

```
#!/usr/bin/env bash
git clone git@github.com:aviggiano/tcc # clona o repositorio
cd tcc && Rscript recsys/results/run_tests.R  # executa o script de testes
```

The screenshot shows the GitHub 'Personal settings' page with the 'SSH keys' section highlighted. On the left sidebar, 'SSH keys' is selected. The main area displays two existing SSH keys with their titles, creation and last usage dates, and delete buttons. Below this, a form for adding a new key is shown, with a title 'aws' and a large text area containing a public SSH key. A green 'Add key' button is at the bottom of the form.

Need help? Check out our guide to [generating SSH keys](#) or troubleshoot [common SSH Problems](#)

**SSH Keys**

Add SSH key

This is a list of SSH keys associated with your account. Remove any keys that you do not recognize.

● **aws** [Delete]  
Added on May 19, 2014 — Last used on Nov 5, 2014

● **aws** [Delete]  
Added on Jun 10, 2014 — Last used on Jun 30, 2014

**Add an SSH Key**

Title  
aws

Key

```
ssh-rsa AAAAB3NzaC1yc2EAAAQABAAQCbxcsoyHzyqTzXp4fl1Q3OT58Lsb7QLx+7nQ6y0OIoWhK+r5vnSVi0BpTC+2hMrlg1rZTC1ED7Nb+Si9bRvf+1UYW0iVUXtwAVCoLMNBdlfE7QCWbJmTmmBLcv9PloCAvCfrxBh+fw3hXG388/LIEZJckPYogho0PAwFv3IXAGtVmV6cBcTTfKPUnX+np6xqnf4tYQpmPW/mnxk9s3bbEmcE1eYJkrE2IWdzy6EBnR9d4cBW5D8/MMM54xMJuqWZO/sijLLT0oFTTroiwr+OX2DxaFdqCv8Agy1WZTeGhBAW1nvIVt5WVcWVBSzBCZfq8mYe+zYnbwl ec2-user@ip-10-168-40-38]
```

Add key

Figura 10 – Cadastro da chave pública no GitHub

---

# 7 Resultados

Os resultados deste trabalho são a análise de desempenho dos algoritmos propostos, em termos de precisão, abrangência e tempo computacional, mediante a mudanças em suas variáveis de importância (Tabela 23).

Além disso, as metodologias de solução de cada um dos sistemas serão debatidas, de modo a explorar casos de uso particulares e a propor melhorias nos métodos computacionais. Serão respondidas perguntas como “O que acontece com itens ou usuários sem nenhuma avaliação?” e “Qual o desempenho dos métodos para outros bancos de dados?”.

Tabela 23 – Parâmetros de influência no desempenho dos algoritmos de recomendação

Variável	Descrição	Valor padrão
$N$	Tamanho da lista de recomendação	20
$T$	Percentual da base de aprendizado na validação cruzada	75%
$H$	Percentual de avaliações “escondidas” dos usuários-teste na validação cruzada	75%
$M$	Valor mínimo para avaliações positivas	2
$k$	Número de vizinhos mais próximos	10
$\mathcal{F}$	Conjunto de atributos dos itens	Todos atributos
$d^f$	Medida de distância entre atributos	Distância $L_1 \ \cdot\ ^f$
$W$	Quantidade de pesos dos atributos	Todo $w_f > 0$

## 7.1 Tamanho da lista de recomendações $N$

Assim como mostra a literatura, a medida que o tamanho da lista de recomendações aumenta, a precisão cai e a abrangência cresce (Figuras 11 e 12). A primeira decresce com  $N$  porque a quantidade de itens sugeridos se torna excessivamente maior que a quantidade de itens positivamente avaliados pelos usuários-teste. A segunda, por sua vez, cresce com  $N$  porque a probabilidade de sugerirmos itens relevantes para o usuário aumenta quando sugerimos mais itens. Para  $N = |\mathcal{I}|$ , a abrangência atinge 100%, pois todos os itens teriam sido recomendados.

O método UP supera os dois outros algoritmos para todos os valores de  $N$ , tanto em precisão quanto em abrangência, como se observa pelo gráfico das medidas  $F_1$ .

Contrariamente ao esperado, a qualidade de recomendação do algoritmo UI é sensivelmente inferior à do algoritmo UP. Isso se deve ao fato de a correlação usuário-item daquele método colocar ênfase no valor do atributo  $a_{if}$ , mesmo que esses atributos não

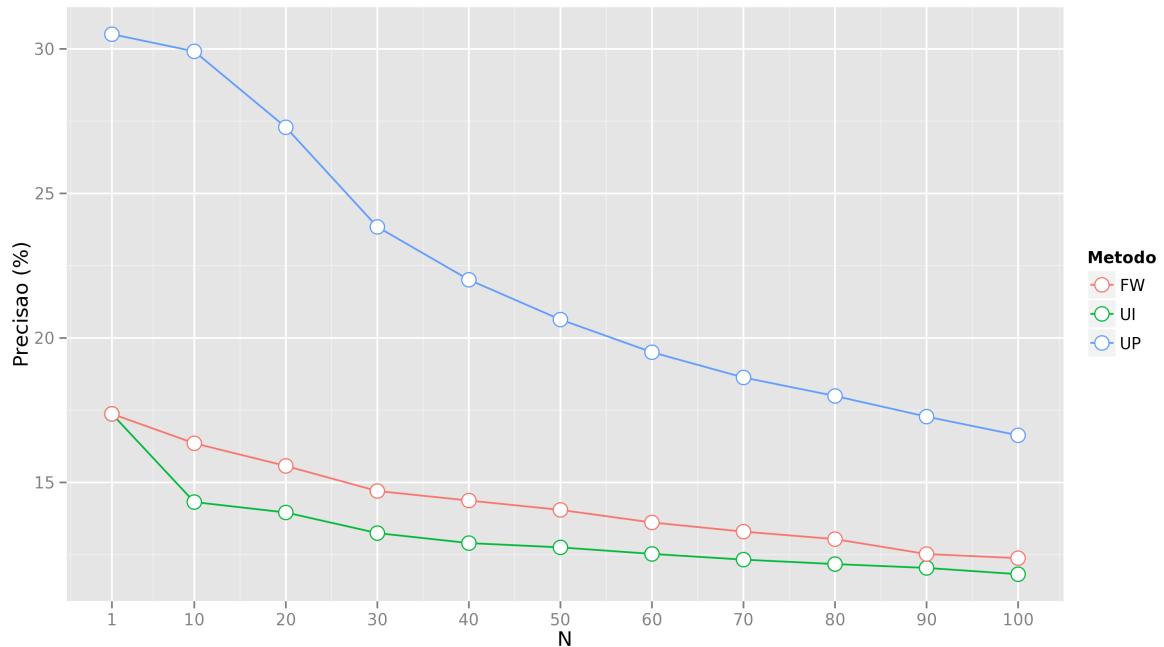


Figura 11 – Precisão em função do tamanho da lista de recomendações  $N$

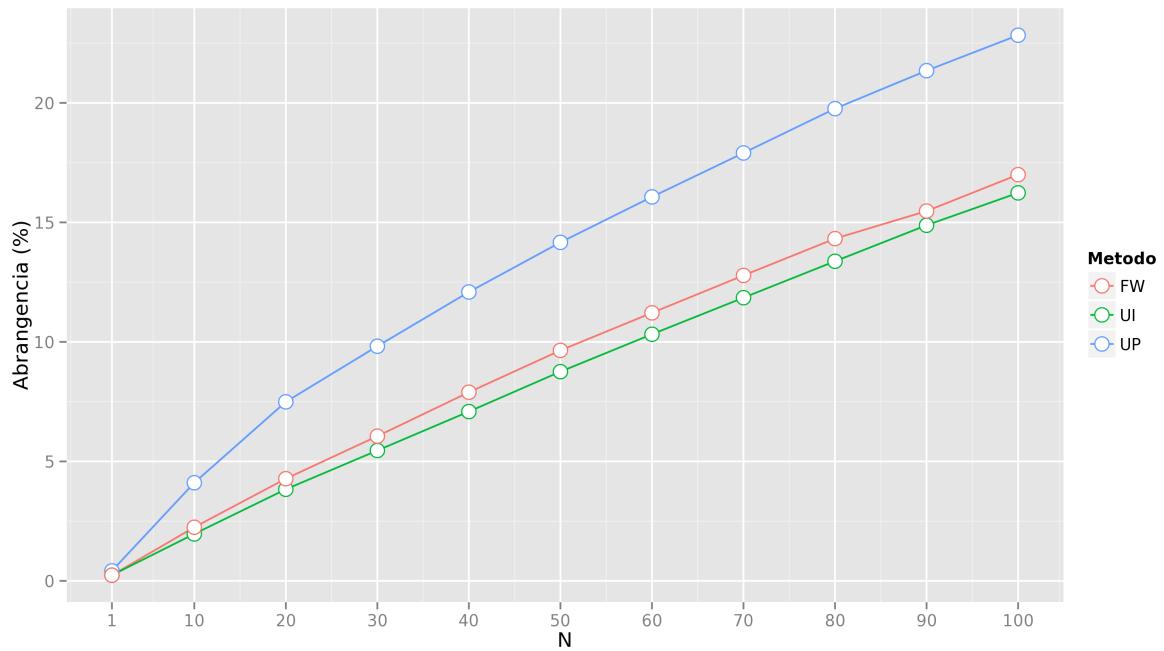
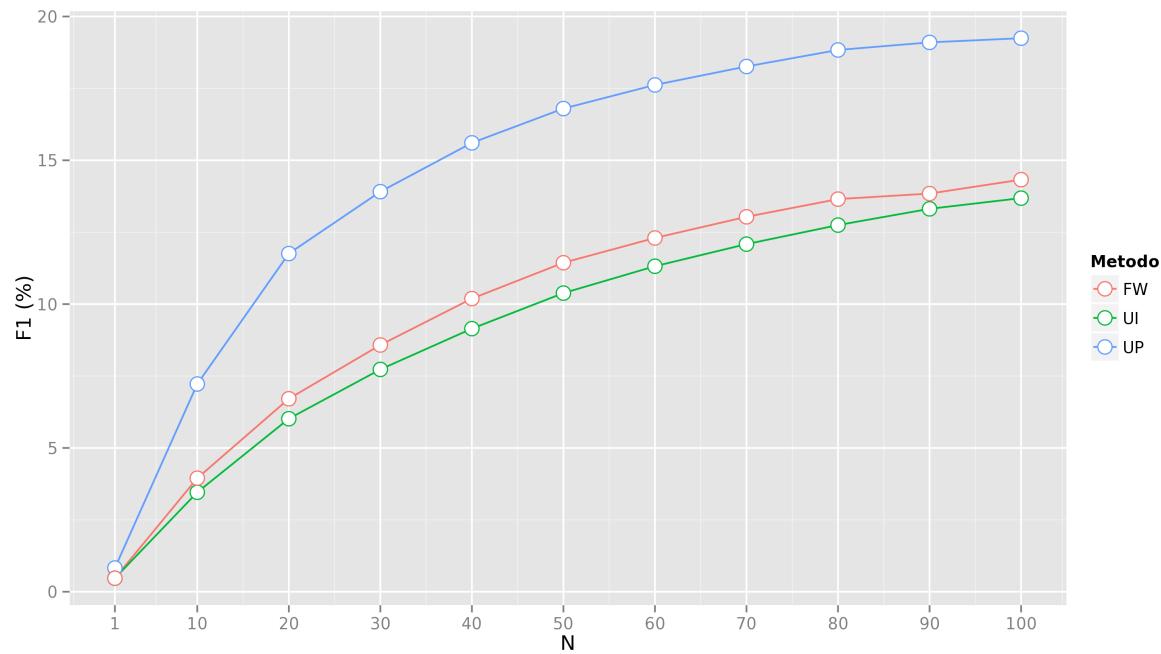
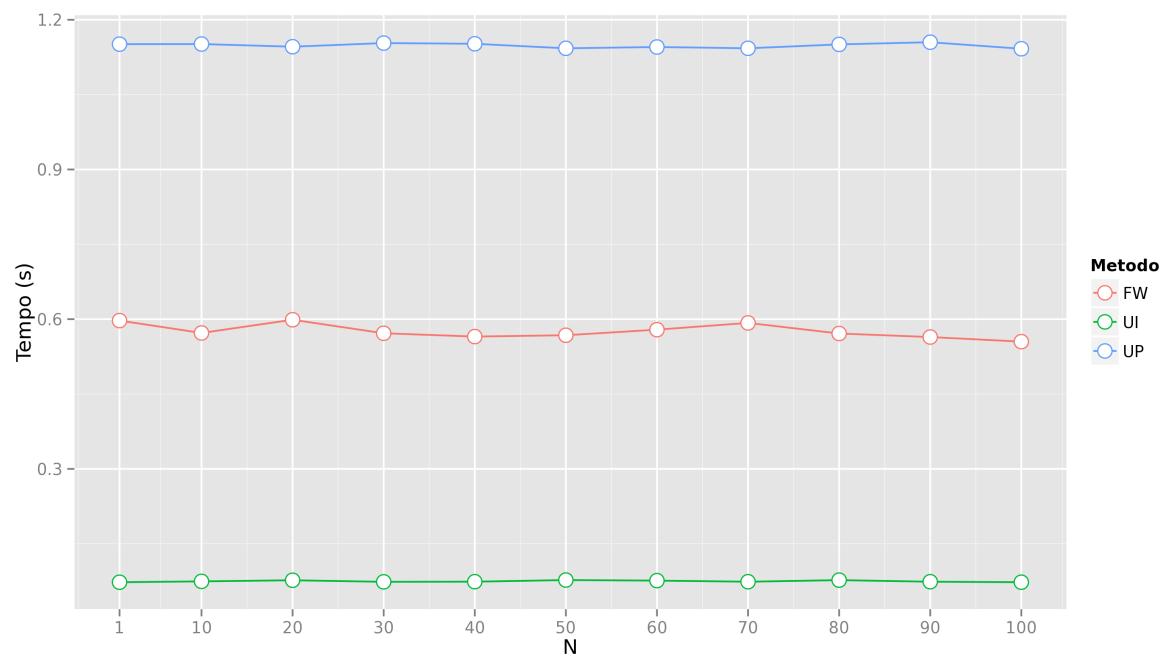


Figura 12 – Abrangência em função do tamanho da lista de recomendações  $N$

Figura 13 – Medida  $F_1$  em função do tamanho da lista de recomendações  $N$ Figura 14 – Tempo de execução em função do tamanho da lista de recomendações  $N$

sejam diretamente proporcionais à preferência do usuário. Esse cálculo é incoerente, por exemplo, para atributos  $f = \text{data}$ : mesmo que o usuário tenha um elevado interesse  $w_{uf}$  por filmes antigos, o valor de  $a_{if}$  não leva em conta se sua preferência é por filmes da década de 1970 ou 1990. Nesse caso, o algoritmo indicaria incorretamente que filmes mais recentes são mais adequados para aquele usuário, porque possuem maior  $a_{if}$ .

A fim de corrigir essa falha no algoritmo UI, seria necessário, por exemplo, aplicar nos atributos  $a_{if}$  uma função  $g_f$  que crescesse no mesmo sentido do interesse do usuário por aquela *feature*. Dessa forma, o cálculo  $\sum_f w_{uf} g_f(a_{if})$  significaria de fato a similaridade entre o usuário  $u$  e o item  $i$  medida através de seu interesse  $g(a_{if})$  pelas *features*  $f$ .

Apesar de alta qualidade das recomendações do método UP, este possui também a maior complexidade computacional. Seu tempo de execução é 2 vezes maior que o do método FW e 4 vezes maior que o do método UI. Todavia, nenhum desses tempos de execução é crítico, tendo em vista que o sistema não seria colocado diretamente à disposição dos clientes, mas que as recomendações seriam enviadas via email, por exemplo.

A fim de melhorar a velocidade das recomendações, a solução mais eficiente é a mudança da linguagem de programação. O uso de linguagens C, C++ ou Python pode melhorar o desempenho computacional em até 500 vezes (36).

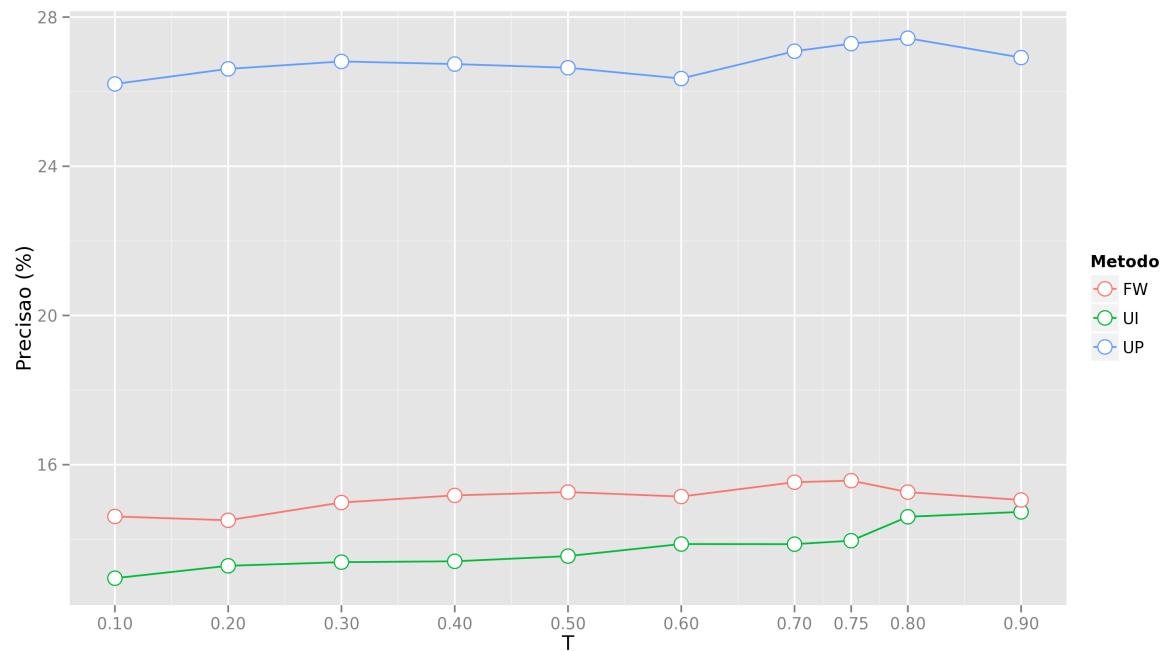
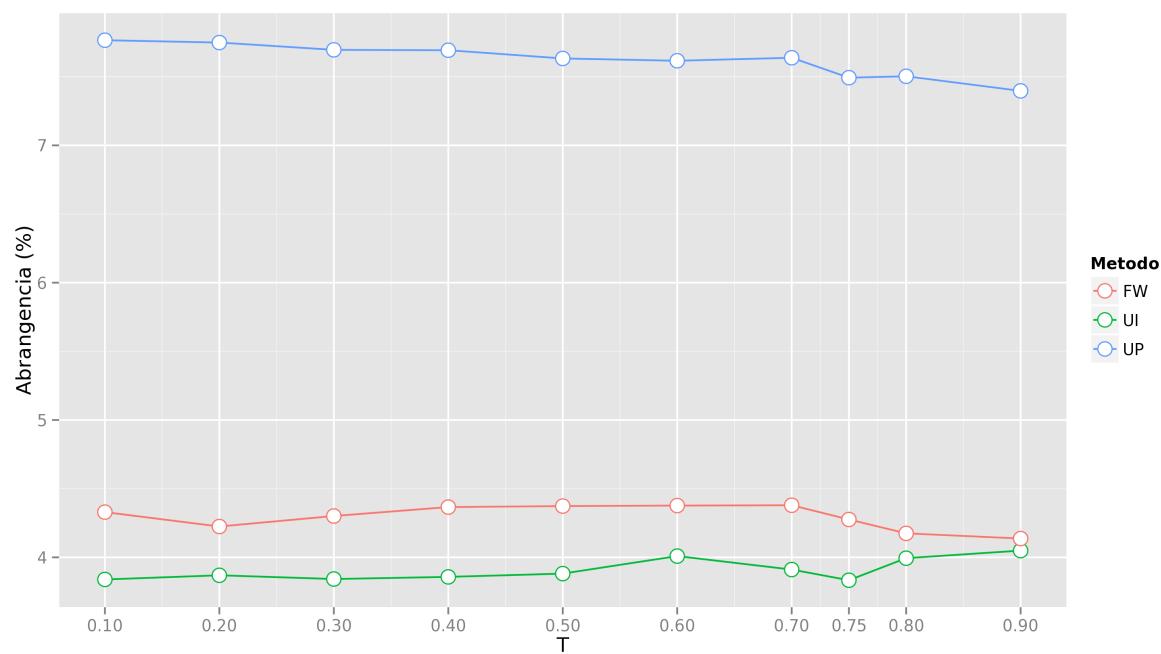
## 7.2 Percentual da base de aprendizado $T$

A medida que o percentual da base de aprendizados aumenta, a precisão de todos os métodos cresce ligeiramente. Isso é consequência do caráter colaborativo dos algoritmos, já que a qualidade da recomendação depende da quantidade total de dados. Entretanto, pode-se observar que a abrangência e a medida  $F_1$  são praticamente constantes para valores crescentes de  $T$  de modo que esse parâmetro não tem grande relevância para o sucesso do sistema de recomendação.

O parâmetro  $T$  não exerce nenhuma influência sobre o tempo de execução dos métodos UP e UI, mas apenas sobre o método FW. Isso ocorre porque a etapa de maior custo computacional (Equação 5.9) é linearmente dependente da quantidade de usuários  $|\mathcal{U}|$ . Quanto menos usuários-teste, mais veloz é o algoritmo.

## 7.3 Percentual de avaliações “escondidas” dos usuários-teste na validação cruzada $H$

Quanto maior o número de avaliações “escondidas”, mais fácil é acertar os itens dos usuários-teste, pois a lista de recomendação é pequena em relação ao total de itens positivamente avaliados pelo usuário. Por esse motivo, a precisão cresce com  $H$  para todos

Figura 15 – Precisão em função do percentual da base de aprendizado  $T$ Figura 16 – Abrangência em função do percentual da base de aprendizado  $T$

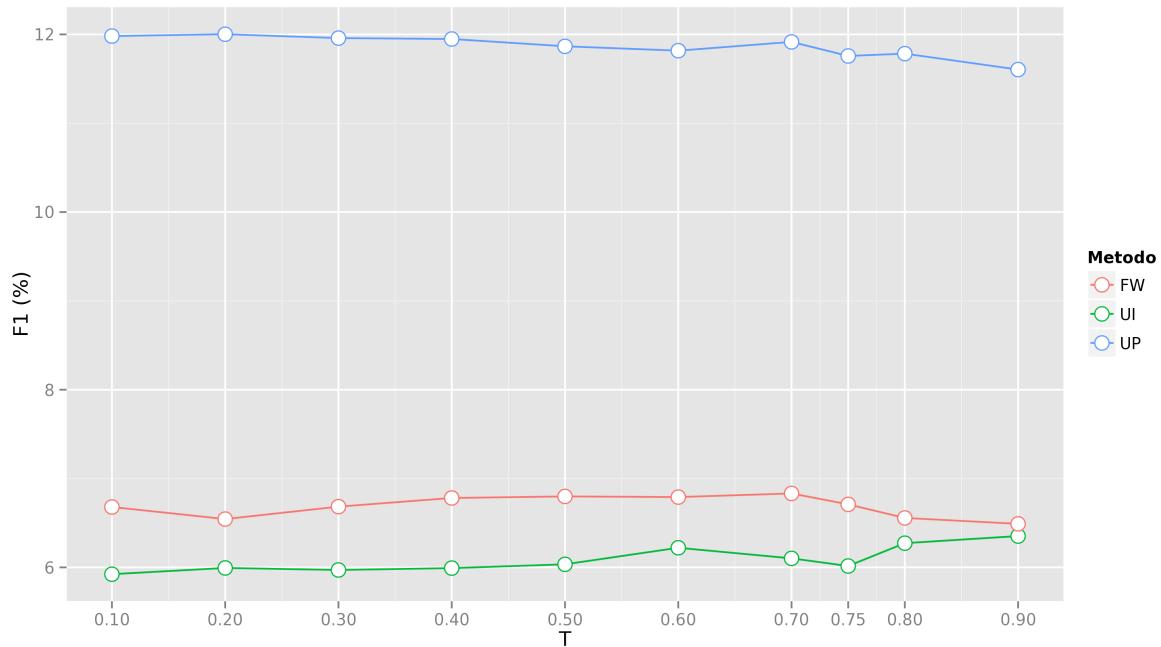


Figura 17 – Medida  $F_1$  em função do percentual da base de aprendizado  $T$

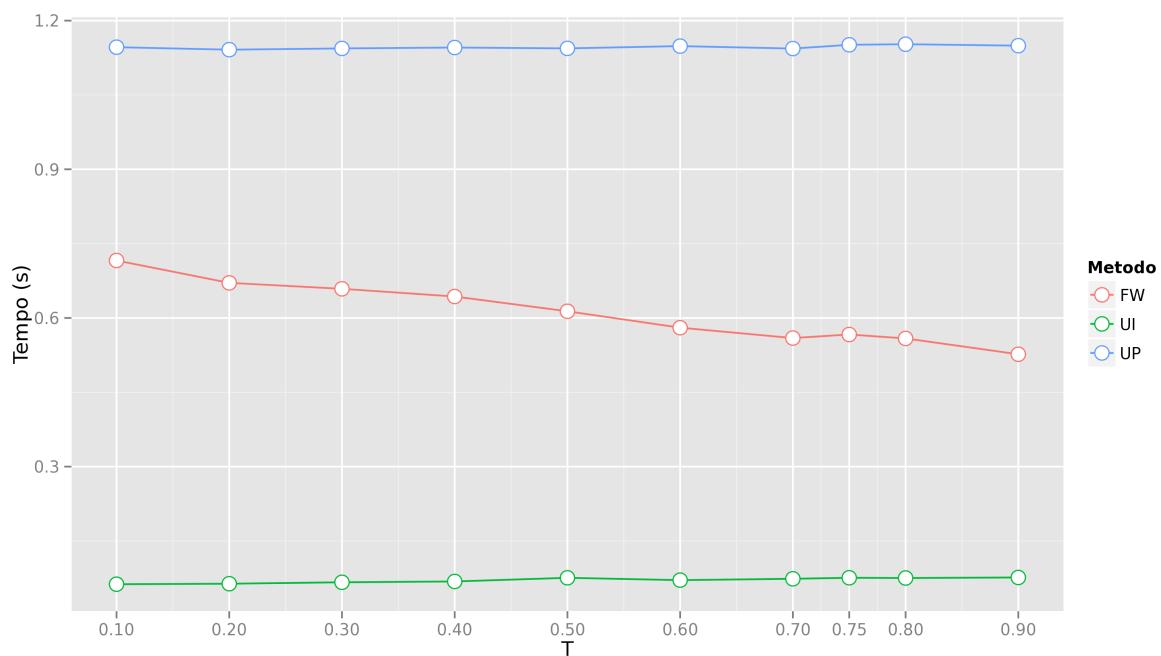


Figura 18 – Tempo de execução em função do percentual da base de aprendizado  $T$

os métodos.

Para o algoritmo FW, a precisão atinge seu máximo em  $H = 75\%$  e depois decresce ligeiramente (Figura 19). Isso ocorre porque o cálculo dos pesos  $w_f$  depende da quantidade de avaliações  $r_{ui}$ . Existe, pois, um compromisso (*tradeoff*) entre facilidade de se acertar itens avaliados quando há muitas avaliações escondidas e a dificuldade de se estimar  $w_f$  quando não há muitos dados de avaliações.

Ao passo que a precisão dos métodos aumenta com  $H$ , a abrangência diminui. Visto que a quantidade de itens da lista *top-N* é fixa, quanto maior o número de itens “escondidos”, mais difícil é de se retornar todos os itens relevantes.

O resultado de uma precisão crescente em função de  $H$  e uma abrangência decrescente é que a medida  $F_1$  possui um ponto de máximo. Para todos os métodos, o valor máximo é tal que  $H = 50\%$  (Figura 21).

Quanto ao tempo de execução, a influência é a mesma do parâmetro  $T$ : para o método FW, a etapa de maior custo computacional (Equação 5.9) é linearmente dependente da quantidade de itens  $|\mathcal{I}|$ . Quanto menos avaliações de itens dos usuários-teste, mais veloz é o algoritmo.

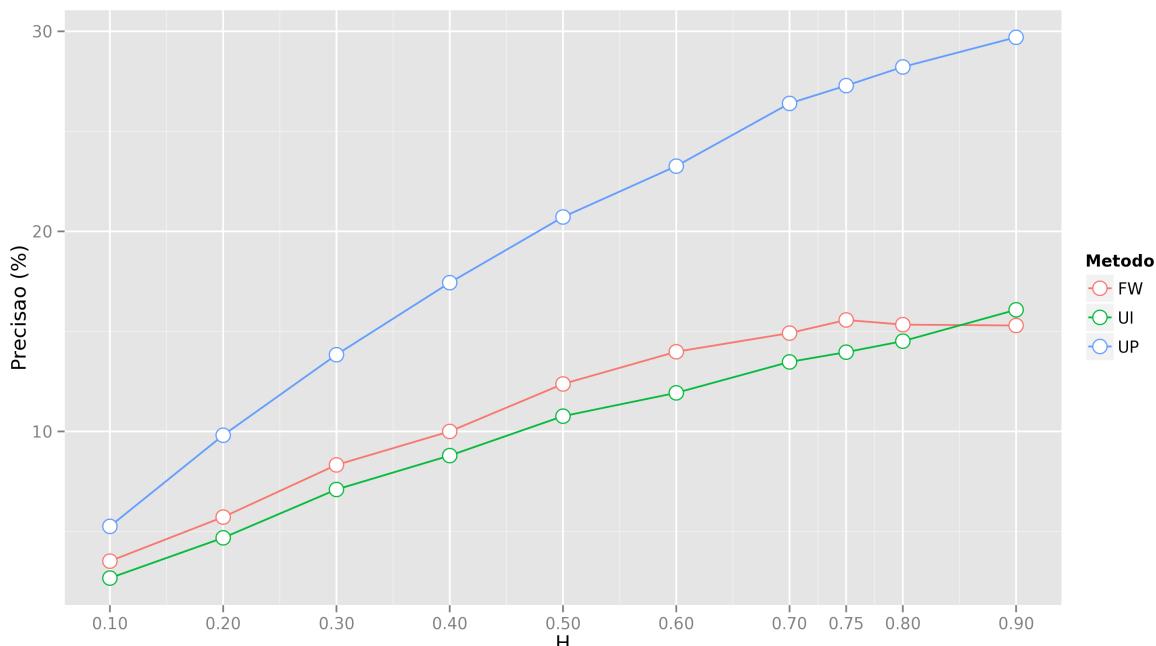


Figura 19 – Precisão em função do percentual de avaliações “escondidas”  $H$

## 7.4 Valor mínimo para avaliações positivas $M$

Contrariamente ao que esperávamos, tornar o algoritmo mais “seletivo” não melhora sua precisão. Apesar de o valor mínimo  $M$  estar intimamente ligado com a noção

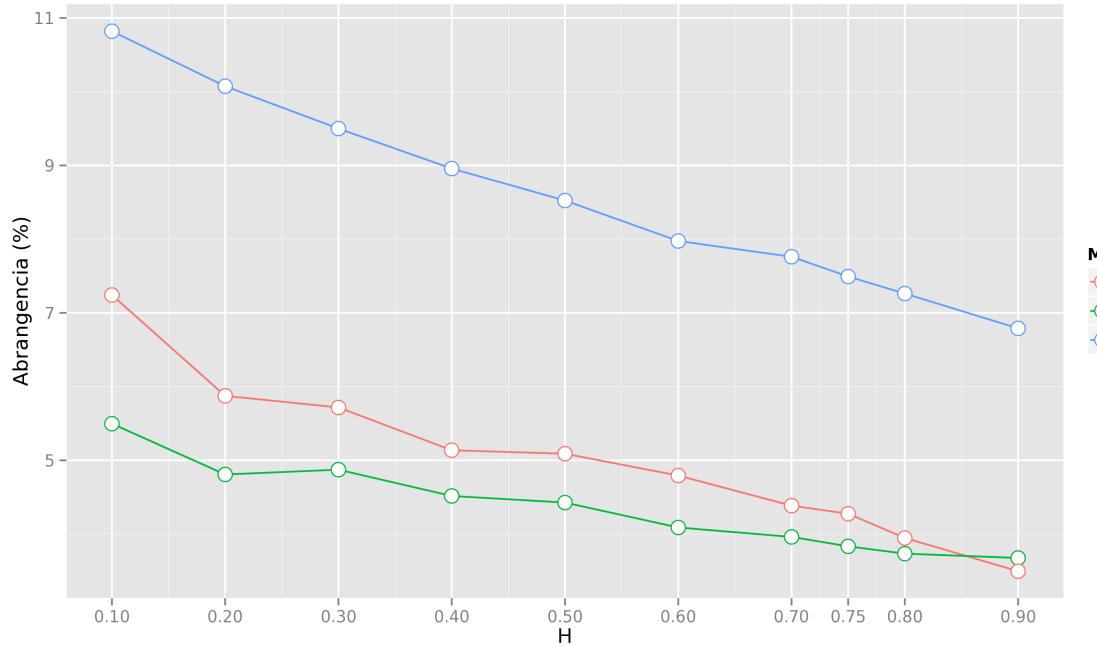


Figura 20 – Abrangência em função do percentual de avaliações “escondidas”  $H$

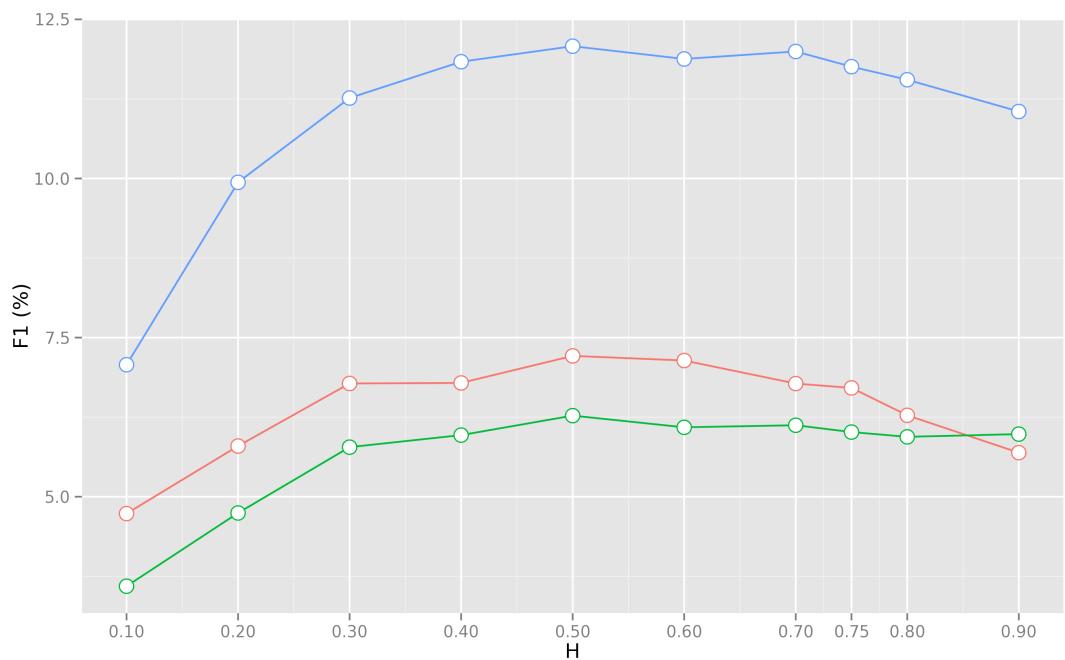


Figura 21 – Medida  $F_1$  em função do percentual de avaliações “escondidas”  $H$

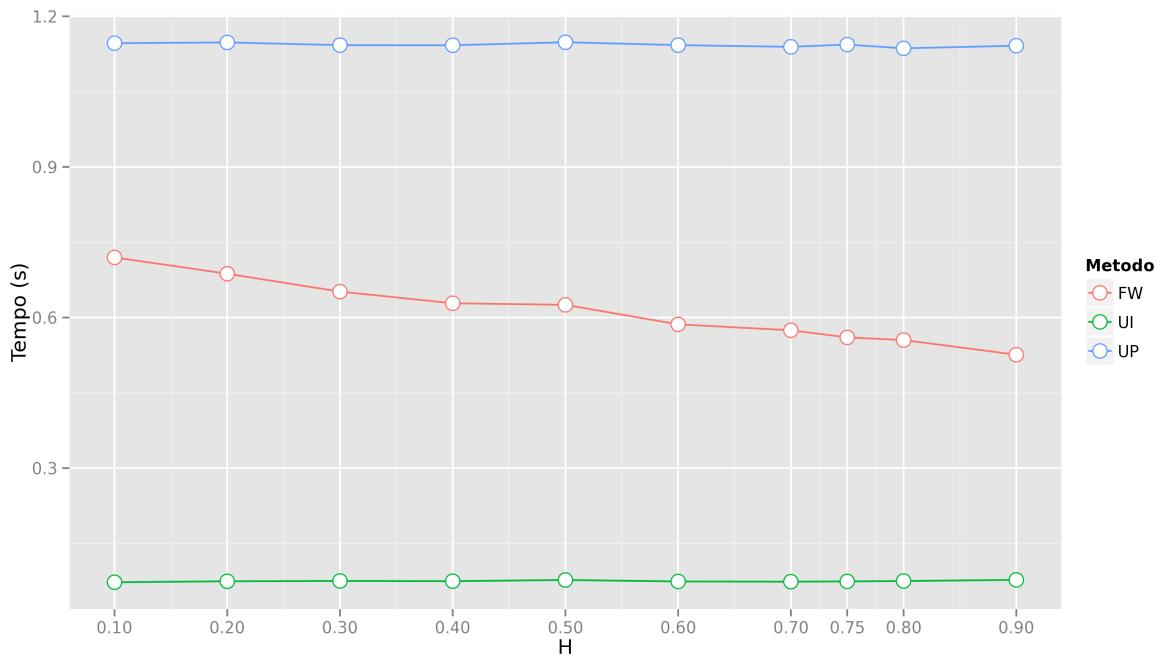


Figura 22 – Tempo de execução em função do percentual de avaliações “escondidas”  $H$

de “avaliação positiva” e de entrar no cálculo de parâmetros importantes dos métodos (Equações 5.7 e 5.11), esse parâmetro pouco influencia a precisão para  $0 \leq M \leq 2$ .

Esse resultado pode ser explicado porque a maioria das avaliações são positivas (Figura 24), e portanto  $b_M$  tem quase o mesmo efeito de  $b_0$ . Isso não ocorre somente pelo fato de os clientes comprarem itens similares a seus gostos, e portanto de raramente se decepcionarem, mas também pelo fato de os usuários terem menos disposição para dar avaliações negativas. Esse fenômeno se chama *hidden feedback*, e se caracteriza pelo fato de que os itens avaliados não são escolhidos ao acaso, mas sim por despertarem aspectos de interesse das preferências do usuário, indo além dos valores numéricos das avaliações (35).

Ao se analisar a abrangência dos métodos, a seletividade influencia na recomendação. Quanto maior  $M$ , menor é a quantidade de itens muito bem avaliados. Estes possuem elevada ponderação/correlação e são facilmente escolhidos pelos algoritmos. Por esse motivo, o desempenho do sistema é melhor.

A complexidade computacional dos algoritmos também depende de  $M$ , já que mais ou menos parâmetros são analisados no cálculo da TF-IDF (métodos UI e UP) e dos pesos dos atributos (método FW).

Um detalhe a se observar é que a precisão é nula e a abrangência é inexistente para  $M = 5$ , já que todas as avaliações  $r_{ui}$  pertencem ao conjunto  $\{1, 2, 3, 4, 5\}$ . Para o algoritmo UI, tanto a precisão quanto a abrangência são nulas para  $M = 4$ .

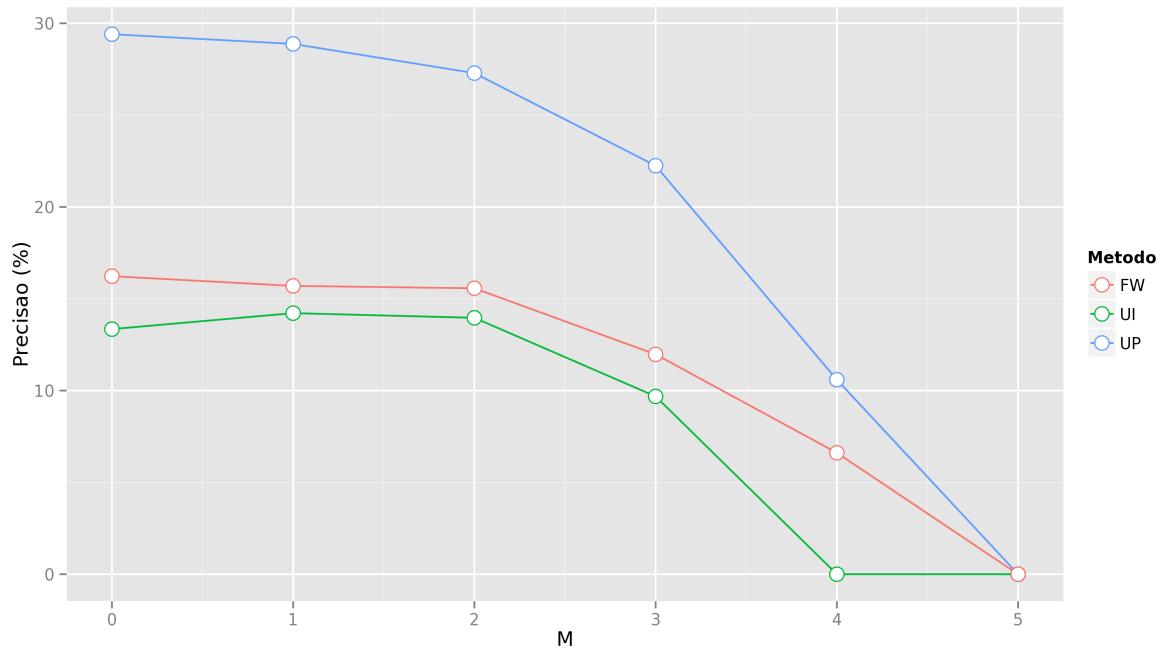


Figura 23 – Precisão em função do valor mínimo para avaliações positivas  $M$

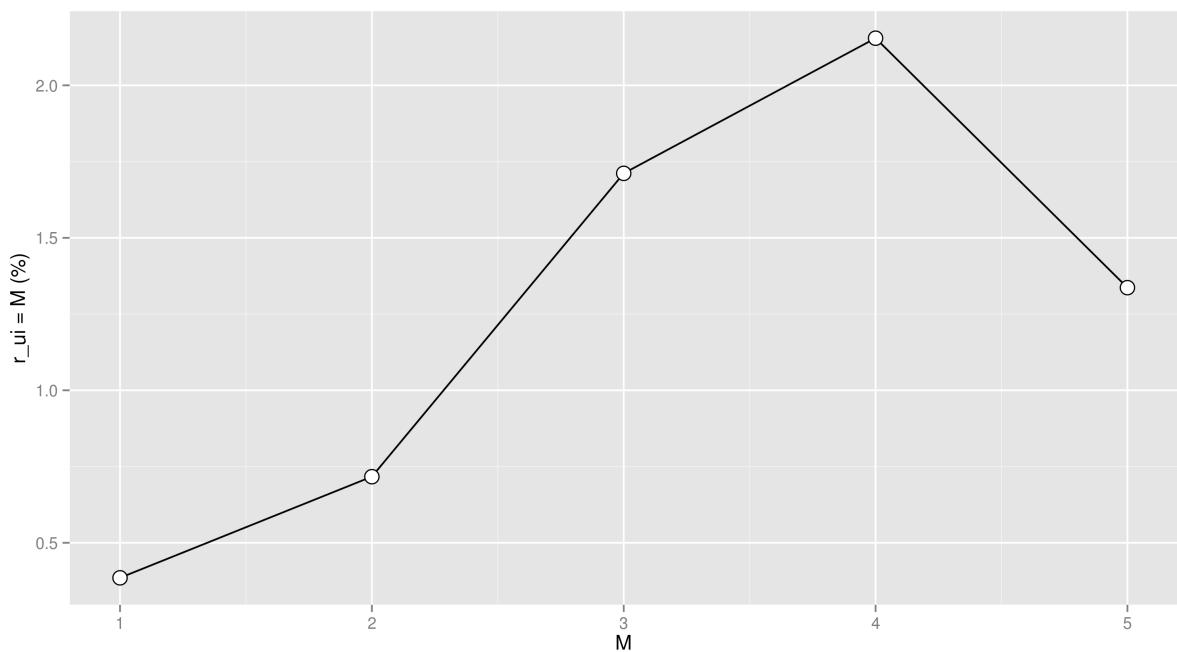
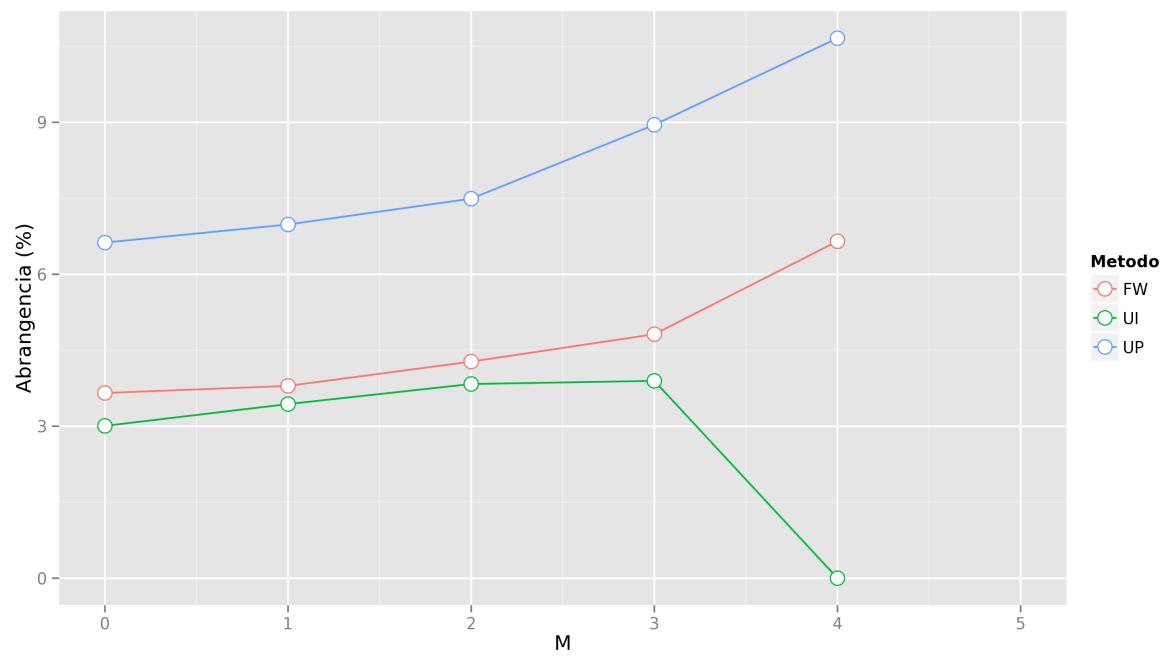
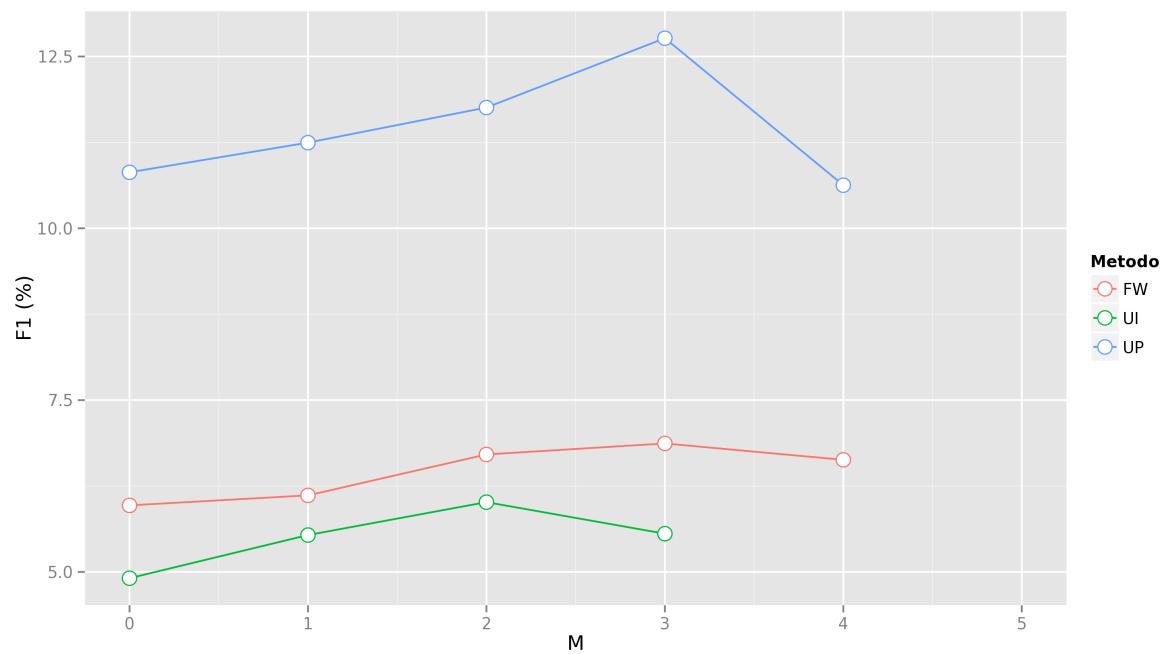


Figura 24 – Percentual de avaliações por valor de  $M$

Figura 25 – Abrangência em função do valor mínimo para avaliações positivas  $M$ Figura 26 – Medida  $F_1$  em função do valor mínimo para avaliações positivas  $M$

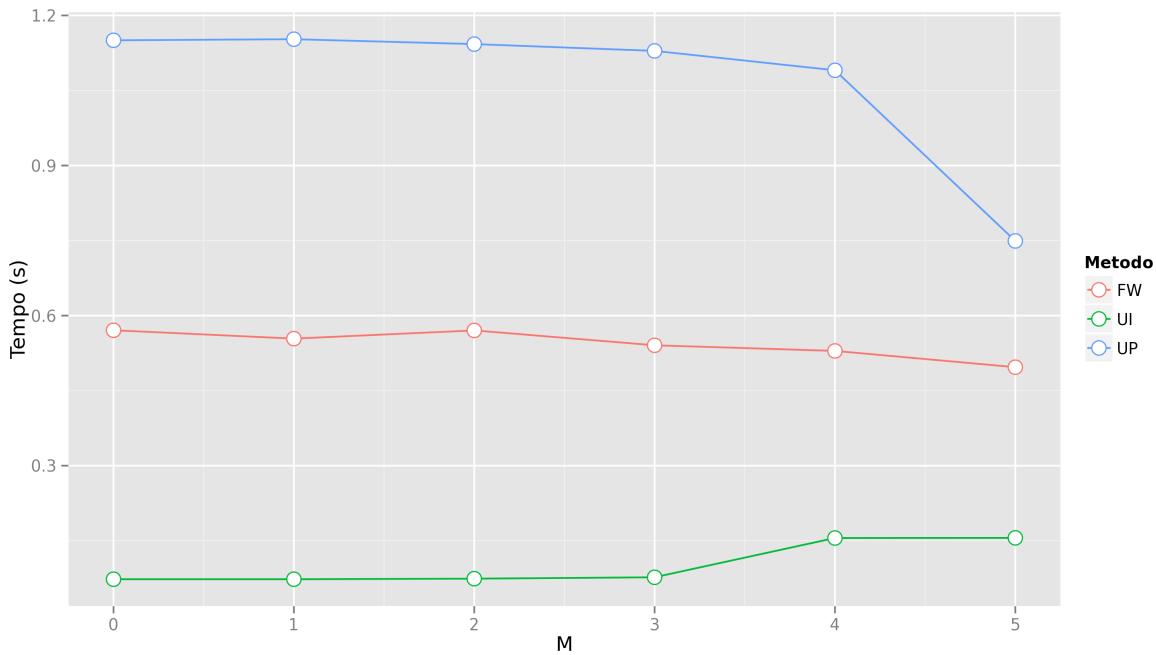


Figura 27 – Tempo de execução em função do valor mínimo para avaliações positivas  $M$

## 7.5 Número de vizinhos mais próximos $k$

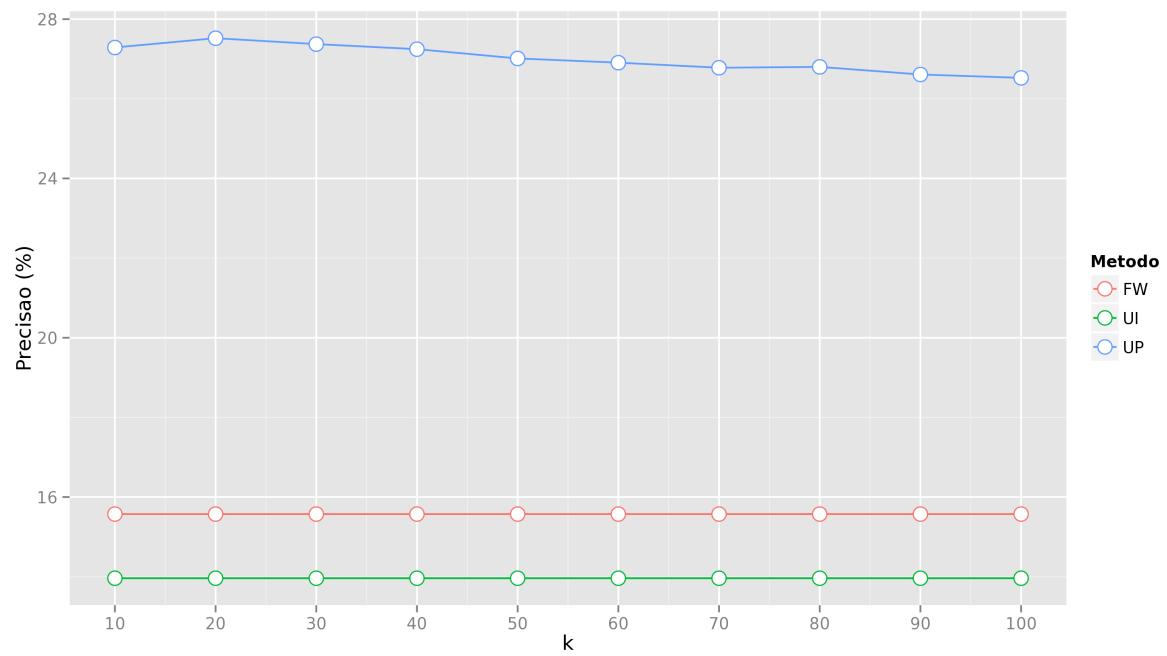
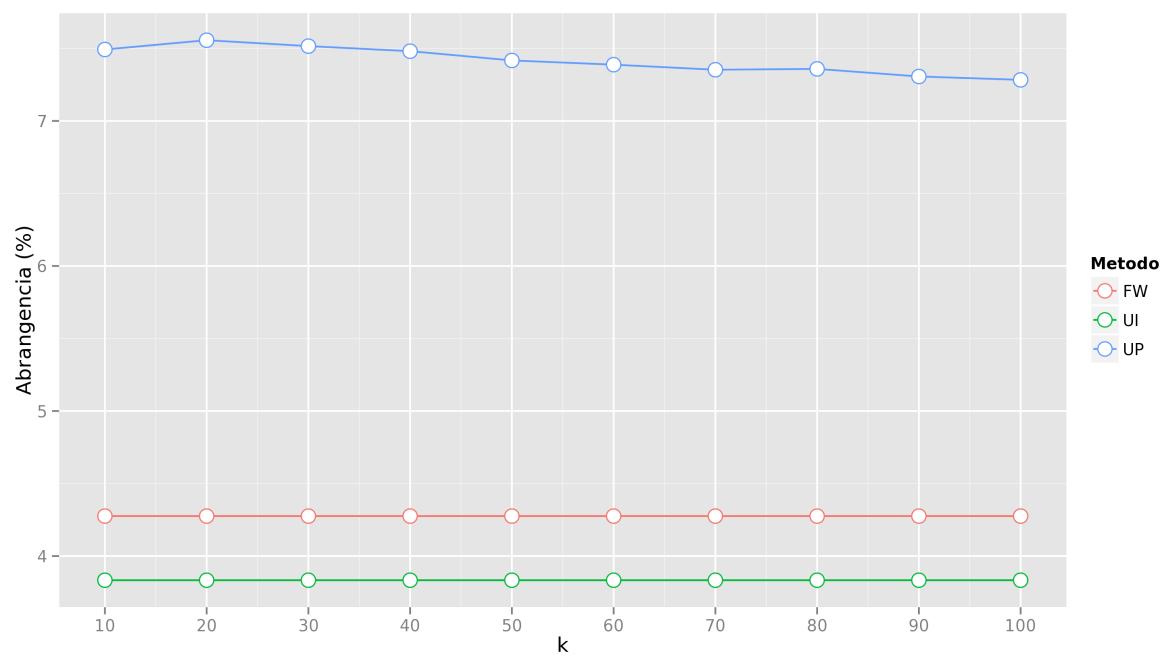
O único método que recomenda itens com base nos vizinhos mais próximos é o UP. Percebe-se que com o aumento de  $k$ , a precisão e a abrangência caem, pois a vizinhança se torna excessivamente grande e repleta de usuários sem muita similaridade com o usuário-teste. Pode-se observar que o valor máximo de precisão e abrangência ocorre para  $k = 20$  (Figura 30).

## 7.6 Conjunto de atributos dos itens $\mathcal{F}$

Para o banco de dados 100k-IMDB, o conjunto de atributos dos itens é  $\mathcal{F} = \{\text{data de lançamento, gênero, ano, duração, orçamento, avaliação, votos}\}$ . A fim de se avaliar a performance dos algoritmos mediante a remoção em determinados atributos, decidimos excluir do conjunto as *features* {data de lançamento, ano}, pois julgamos que elas não eram tratadas corretamente pelos métodos UI e UP.

O resultado desse experimento se observa por exemplo na Figura 31, em que a precisão de todos os métodos melhora substancialmente. Da mesma forma, a abrangência também aumenta para todos os métodos, como se vê na Figura 32.

A conclusão desse experimento é que aumentar a quantidade de atributos dos itens não aumenta necessariamente a qualidade do algoritmo de recomendação. De fato, o algoritmo deve estar preparado para “aprender” quais são as *features* relevantes para cada

Figura 28 – Precisão em função do número de vizinhos mais próximos  $k$ Figura 29 – Abrangência em função do número de vizinhos mais próximos  $k$

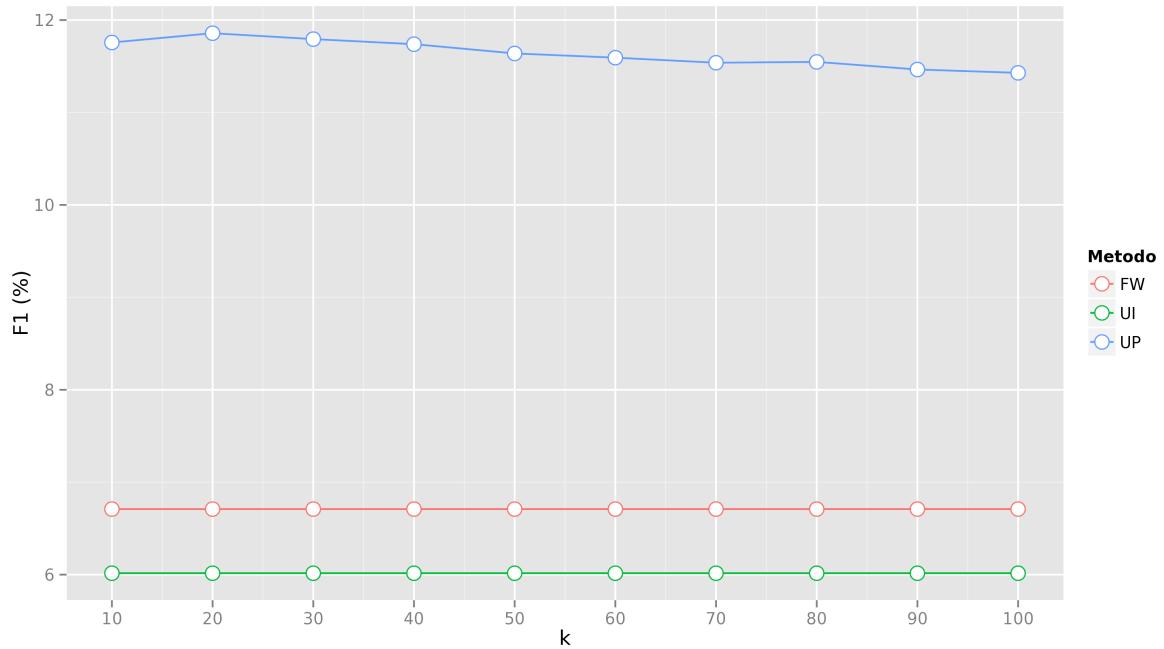


Figura 30 – Medida  $F_1$  em função do número de vizinhos mais próximos  $k$

usuário e eventualmente descartar automaticamente os atributos desnecessários.

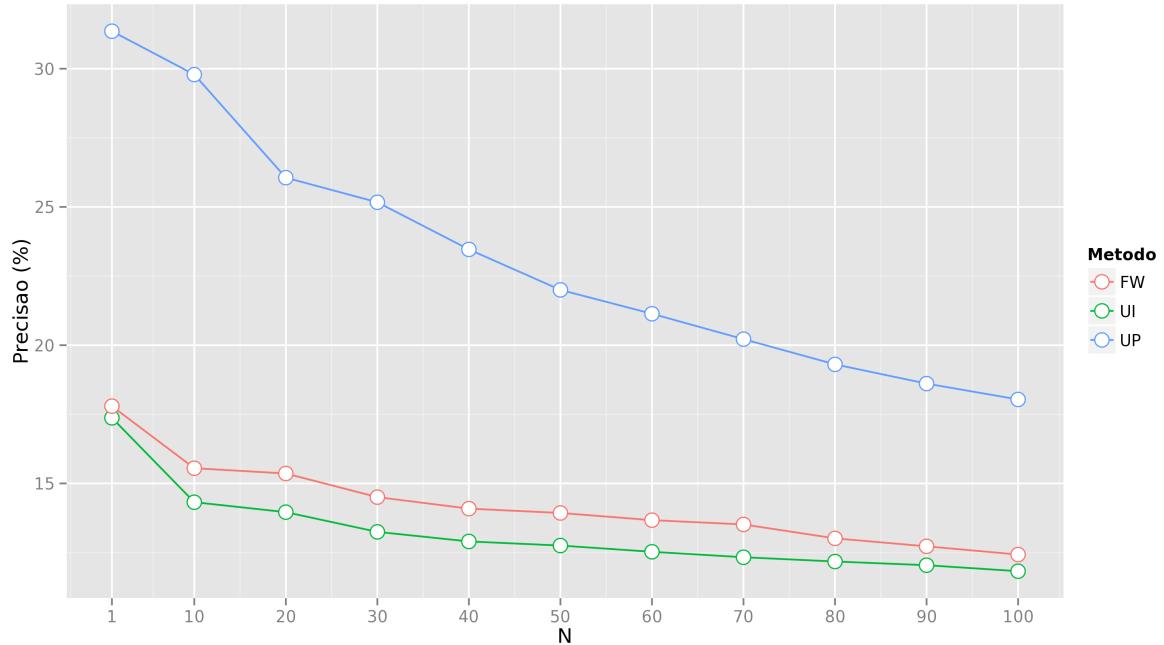
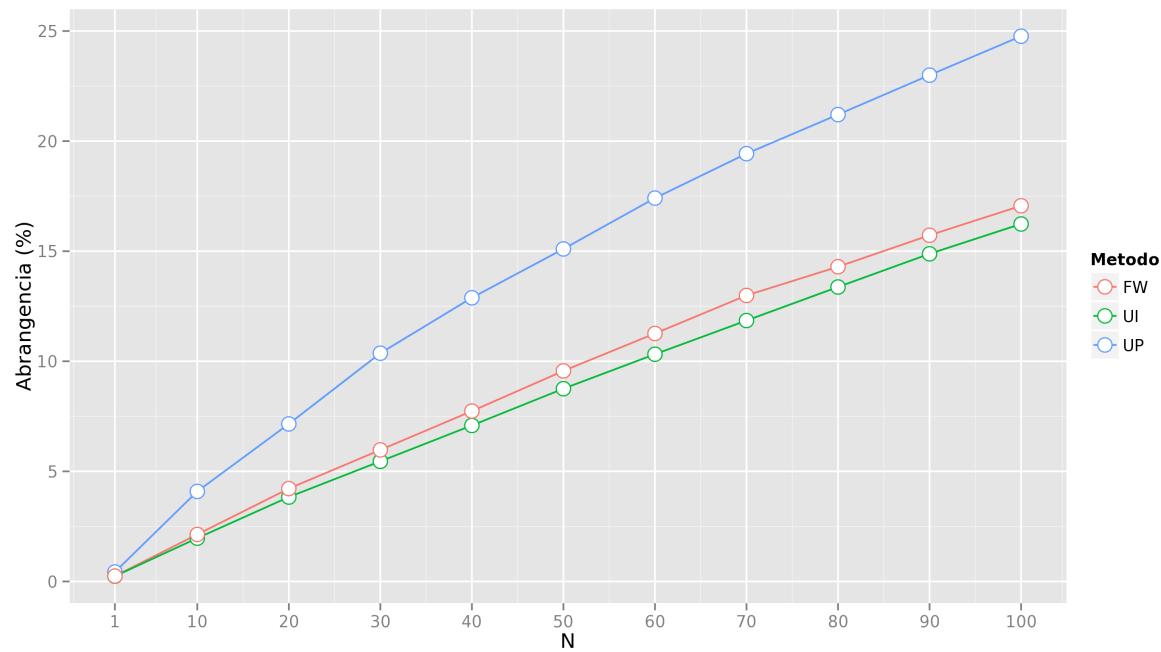
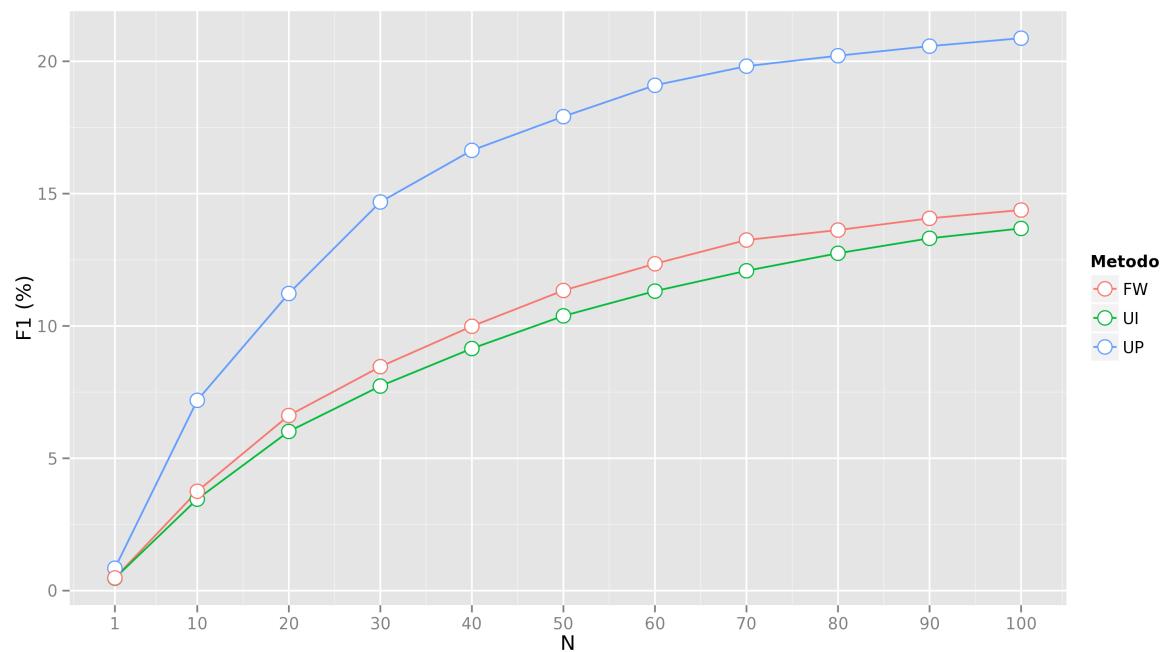


Figura 31 – Precisão em função do tamanho da lista de recomendações  $N$

Figura 32 – Abrangência em função do tamanho da lista de recomendações  $N$ Figura 33 – Medida  $F_1$  em função do tamanho da lista de recomendações  $N$

## 7.7 Medida de distância entre atributos $d^f$

## 7.8 Pesos dos atributos $w_f$

A fim de avaliar a influência da quantidade de pesos utilizados na recomendação do método FW, realizamos os testes selecionando apenas os  $W$  maiores elementos  $w_f > 0$ .

Observa-se que a qualidade da recomendação, tanto em termos de precisão, abrangência e tempo de execução, são aproximadamente independentes da quantidade de pesos  $W$ . Para  $W > 12$ , as medidas de desempenho se tornam constantes.

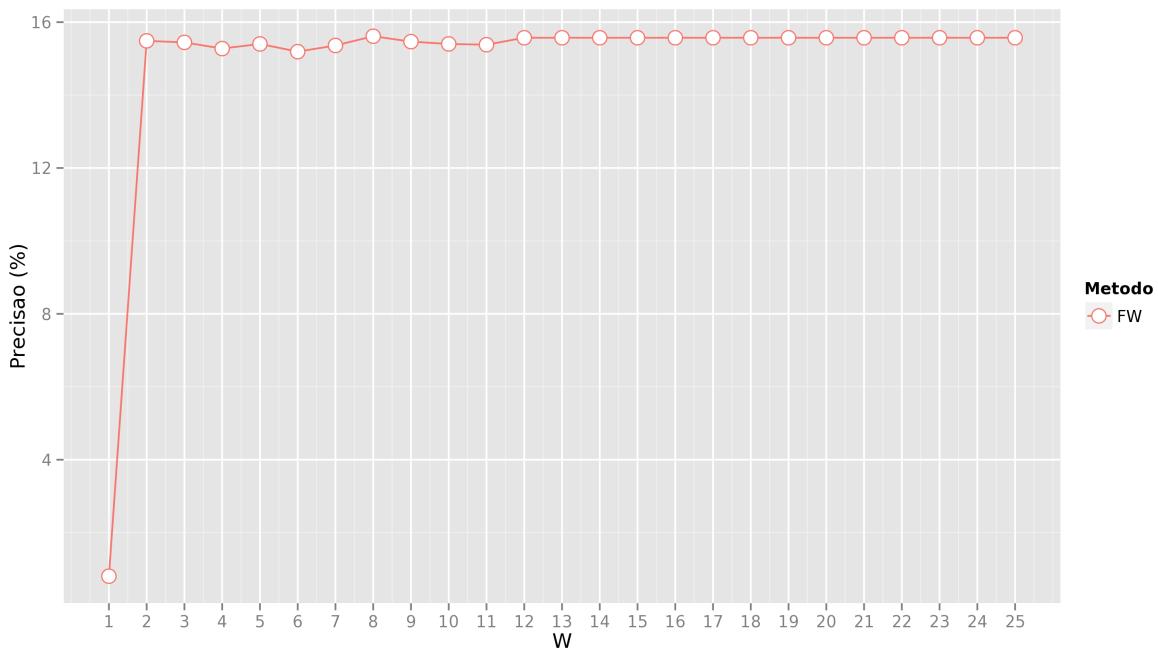
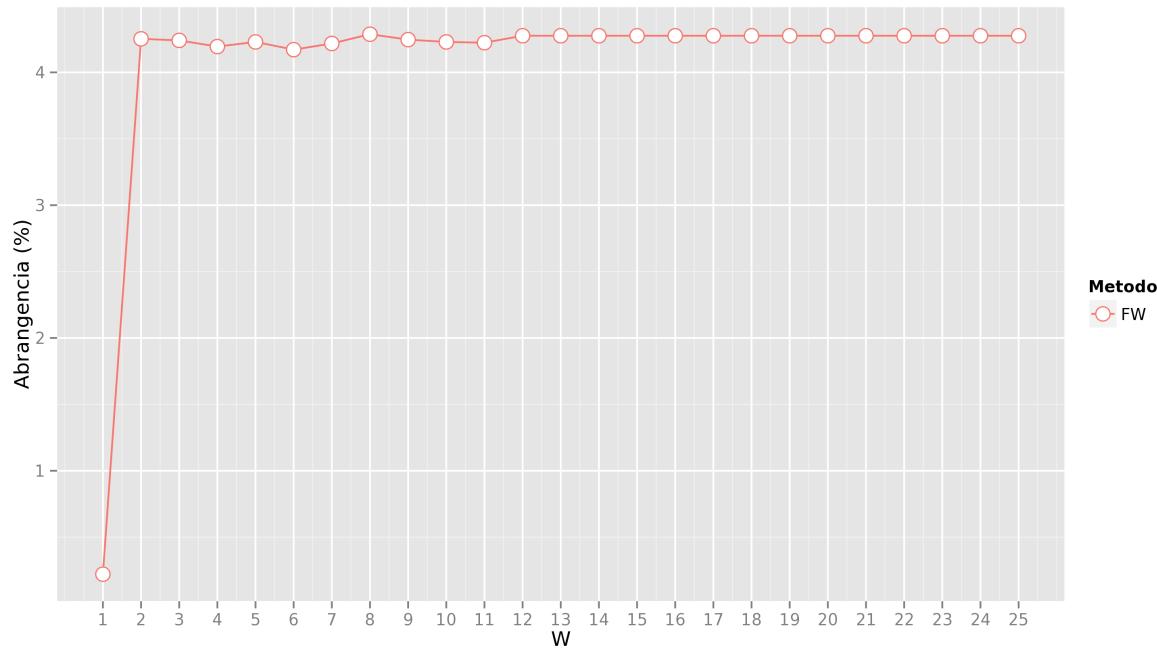
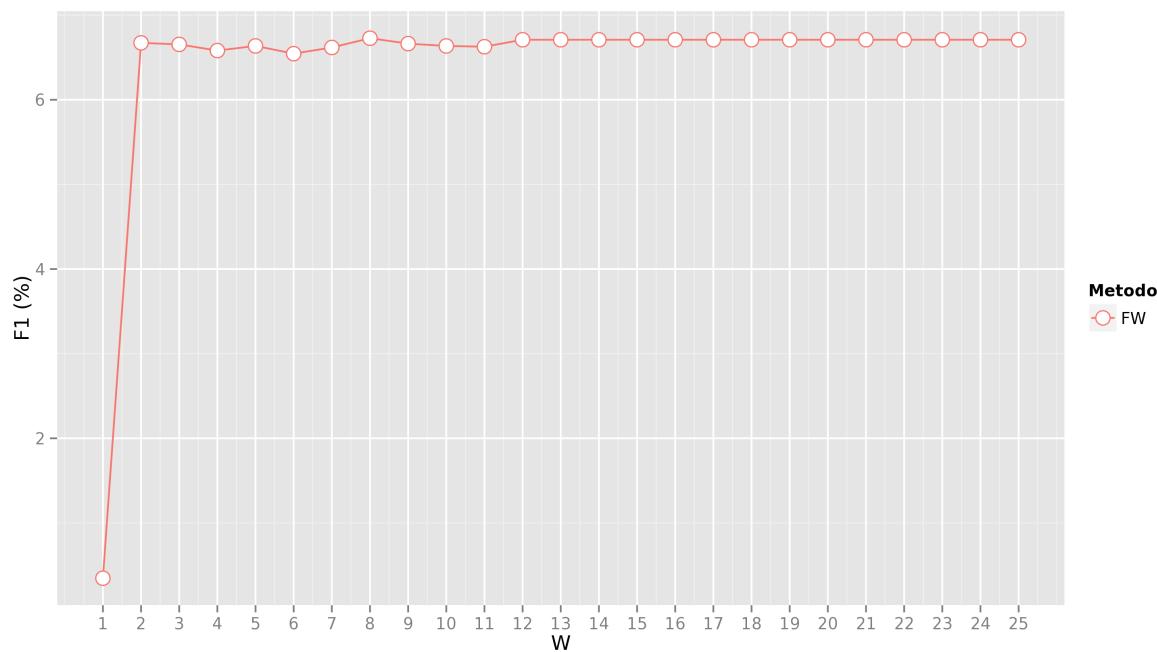


Figura 34 – Precisão em função da quantidade de pesos  $W$

Figura 35 – Abrangência em função da quantidade de pesos  $W$ Figura 36 – Medida  $F_1$  em função da quantidade de pesos  $W$

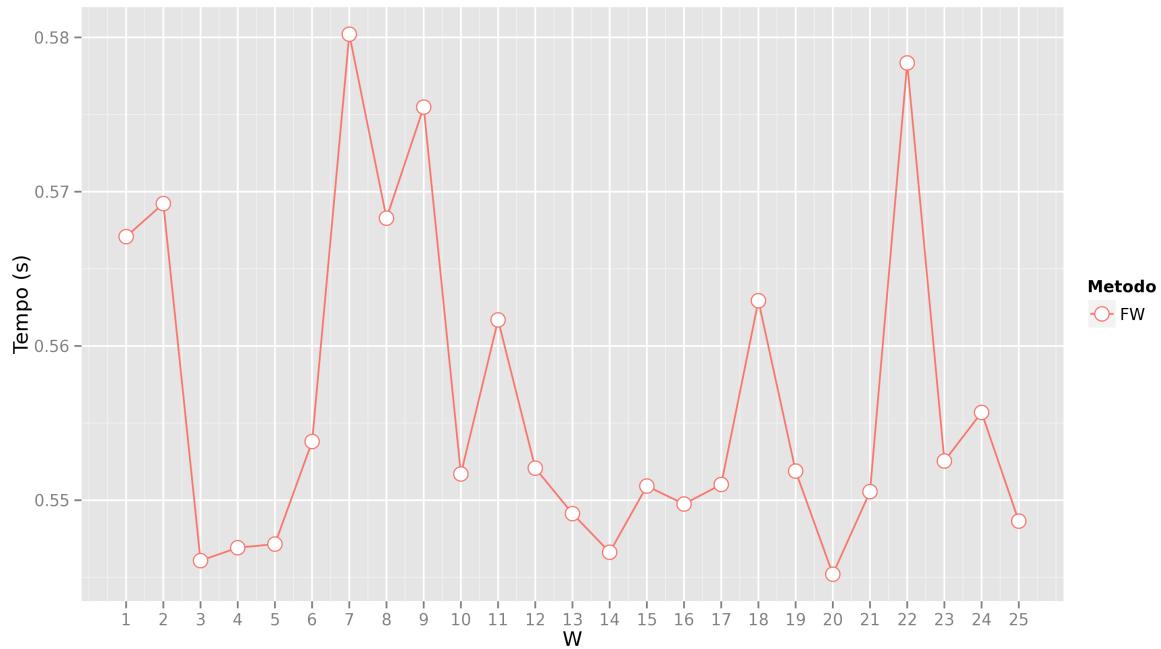


Figura 37 – Tempo de execução em função da quantidade de pesos  $W$

# 8 Conclusão

TODO arrumar

Para os trabalhos futuros, iremos realizar a validação cruzada e avaliar se os requisitos funcionais foram estabelecidos. Em seguida, procuraremos melhorar o sistema de recomendação a fim de torná-lo mais genérico. Buscaremos eliminar restrições quanto a entrada e saída de dados, de forma que elas sejam completamente arbitrárias. O objetivo é que o usuário possa informar ao sistema como é formado sua base, e que todo o tratamento preliminar seja feito automaticamente.

Caso haja tempo, trabalharemos também na construção de um *driver* que possibilite a conexão entre o sistema de recomendação e um banco de dados SQL, sem que seja necessária a etapa intermediária de arquivos csv para aquisição de dados. Planejamos elaborar um *website* para o sistema de recomendação e exportar toda a lógica para um servidor dedicado. Outra melhoria desejada é a reconstrução dos métodos na linguagem de programação C, a fim de melhorar a performance computacional. Dessa forma, o serviço de “sistema de recomendação nas nuvens” estaria completo e poderia ser utilizado por e-commerce reais.



# A Documentação da biblioteca

---

## Código A.1 – setup.R

---

```

read.history
  input      filename, separator, header, col.names
  output     history
  description Lê o arquivo de histórico de compras e retorna uma matriz
              correspondente. Os parametros separator e header indicam a formatação
              do arquivo e são opcionais. Caso header seja FALSE, col.names deve
              conter um arranjo de palavras indicando os nomes das colunas da matriz.

read.item
  input      filename, separator, header, col.names
  output     item
  description Lê o arquivo de descrição dos itens e retorna uma matriz
              correspondente. Os parametros separator e header indicam a formatação
              do arquivo e são opcionais. Caso header seja FALSE, col.names deve
              conter um arranjo de palavras indicando os nomes das colunas da matriz.

read.user
  input      filename, separator, header, col.names
  output     user
  description Lê o arquivo de descrição dos usuários e retorna uma matriz
              correspondente. Os parametros separator e header indicam a formatação
              do arquivo e são opcionais. Caso header seja FALSE, col.names deve
              conter um arranjo de palavras indicando os nomes das colunas da matriz.

get.r
  input      history
  output     r
  description Le a matriz de histórico de compras e retorna a matriz de
              avaliação r_ui

get.a
  input      item
  output     a
  description Le a matriz de descrição de itens e retorna a matriz de
              atributos dos itens a_if

```

---

---

 Código A.2 – performance.R
 

---

```

hide.data
  input      r, Utrain.Utest, HIDDEN, random=FALSE, has.na=TRUE
  output     matriz r com dados mascarados
  description mascara os dados da matriz r para os usuários de teste da
    lista Utrain.Utest. HIDDEN é o percentual de avaliações a serem
    mascaradas. random é o modo de operação; caso seja TRUE, os dados são
    mascarados aleatoriamente para os usuários-teste. has.na indica se os
    itens não avaliados em r são indicados como NA ou como 0.

divide.train.test
  input      r, TRAIN
  output     lista contendo U.train e U.test
  description divide os usuários em duas bases, uma de treinamento de uma
    de testes, segundo a proporção TRAIN

performance
  input      a, r, M=2, k=10, N=20, norm=TRUE, remove=FALSE, method,
            TRAIN=0.75, HIDDEN=0.75, W=FALSE, repick=FALSE
  output     lista contendo precisão, abrangência, medida F1 e tempo de
            execução do método
  description norm indica se a matriz de avaliações deve ser normalizada.
            W, caso diferente de FALSE, indica a quantidade de pesos de atributos a
            serem utilizados no método FW

```

---

 Código A.3 – fw.R
 

---

```

fw
  input      a, r, rtrain.rtest, Utrain.Utest, M, k, N, W
  output     iu
  description Retorna uma lista de recomendação para todos os usuários
            Utest, após obter o modelo a partir dos usuários Utrain

```

---

 Código A.4 – ui.R
 

---

```

ui
  input      a, r, rtrain.rtest, Utrain.Utest, M, k, N
  output     iu
  description Retorna uma lista de recomendação para todos os usuários
            Utest, após obter o modelo a partir dos usuários Utrain

```

---

## Código A.5 – up.R

---



---

```

up
  input      a, r, rtrain.rtest, Utrain.Utest, M, k, N
  output     iu
  description Retorna uma lista de recomendação para todos os usuários
              Utest, após obter o modelo a partir dos usuários Utrain

```

---

### Código A.6 – functions.R

---

```

b
  input      x, y
  output     1 se x > y ou 0 se x <= y
  description A definição da função b_M é diferente da do artigo de
               referência, em que b_Pt(x) é tal que x >= Pt

delta
  input      m, n
  output     1 se m == n ou 0 se m != n
  description Delta de Kronecker

jaccard
  input      xs, ys
  output     número entre 0 e 1
  description Índice Jaccard entre os conjuntos xs e ys

h
  input      matrix, N=6
  output     imprime as primeiras N linhas e N colunas da matriz
  description Caso a entrada seja um vetor, imprime os N primeiros
               elementos.

top.N
  input      xs, N=10
  output     N maiores elementos da lista xs
  description Usado na construção da lista de itens top-N

index.top.N
  input      xs, N=10, ys.remove=NULL
  output     Índice dos N maiores elementos da lista xs
  description ys.remove é uma lista em que se deseja excluir os elementos
               da lista top.N

```

---

```
normalize
  input      matrix, columns=FALSE
  output     matriz normalizada
  description Caso columns seja TRUE, as colunas da são normalizadas
               dependendo do maior valor absoluto de cada uma delas
```

---

# Referências

- 1 EMARKETER. *B2C Ecommerce Climbs Worldwide, as Emerging Markets Drive Sales Higher*. 2013. Disponível em: <<http://www.emarketer.com/Article/B2C-Ecommerce-Climbs-Worldwide-Emerging-Markets-Drive-Sales-Higher/1010004>>. Citado na página 19.
- 2 EMARKETER. *Global B2C Ecommerce Sales to Hit \$1.5 Trillion This Year Driven by Growth in Emerging Markets*. 2014. Disponível em: <<http://www.emarketer.com/Article/Global-B2C-Ecommerce-Sales-Hit-15-Trillion-This-Year-Driven-by-Growth-Emerging-Markets/1010575>>. Citado na página 19.
- 3 MAC, R.; SOLOMON, B. *Alibaba Boosts IPO Price Range, Could Raise Up To \$25 Billion*. 2014. Disponível em: <<http://www.forbes.com/sites/ryanmac/2014/09/15/alibaba-raises-ipo-price-range-could-raise-up-to-25-billion/>>. Citado na página 19.
- 4 COOPERS, P. W. *Total Retail Global Survey of Online Shoppers*. 2014. Disponível em: <<http://www.pwc.com/gx/en/retail-consumer/retail-consumer-publications/global-multi-channel-consumer-survey/index.jhtml>>. Citado na página 19.
- 5 COOPERS, P. W. *The Go-to-Market Revolution - Igniting Growth with Marketing, Sales, and Pricing*. 2014. Disponível em: <[https://www.bcgperspectives.com/content/articles/go\\_to\\_market\\_strategy\\_growth\\_go\\_to\\_market\\_revolution\\_igniting\\_growth\\_marketing\\_sales\\_pricing](https://www.bcgperspectives.com/content/articles/go_to_market_strategy_growth_go_to_market_revolution_igniting_growth_marketing_sales_pricing)>. Citado na página 19.
- 6 RICCI, L. R. F.; SHAPIRA, B. Introduction to recommender systems handbook. In: *Recommender Systems Handbook*. [S.l.]: Springer, 2011. p. 1–35. Citado na página 19.
- 7 AMATRIAIN, X. *Netflix Recommendations: Beyond the 5 stars*. 2012. Disponível em: <<http://techblog.netflix.com/2012/04/netflix-recommendations-beyond-5-stars.html>>. Citado na página 19.
- 8 MARSHALL, M. *Aggregate Knowledge raises \$5M from Kleiner, on a roll*. 2006. Disponível em: <<http://venturebeat.com/2006/12/10/aggregate-knowledge-raises-5m-from-kleiner-on-a-roll/>>. Citado na página 19.
- 9 DAS, A. S. et al. Google news personalization: scalable online collaborative filtering. In: ACM. *Proceedings of the 16th international conference on World Wide Web*. [S.l.], 2007. p. 271–280. Citado na página 19.
- 10 SCHAFER, J. B.; KONSTAN, J.; RIEDL, J. Recommender systems in e-commerce. In: ACM. *Proceedings of the 1st ACM conference on Electronic commerce*. [S.l.], 1999. p. 158–166. Citado 2 vezes nas páginas 20 e 28.
- 11 SARWAR, B. et al. Analysis of recommendation algorithms for e-commerce. In: ACM. *Proceedings of the 2nd ACM conference on Electronic commerce*. [S.l.], 2000. p. 158–167. Citado na página 22.

- 12 SYMEONIDIS, P.; NANOPoulos, A.; MANOLOPOULOS, Y. Feature-weighted user model for recommender systems. In: *User Modeling 2007*. [S.l.]: Springer, 2007. p. 97–106. Citado 6 vezes nas páginas 23, 28, 33, 35, 41 e 42.
- 13 ADOMAVICIUS, G.; TUZHILIN, A. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *Knowledge and Data Engineering, IEEE Transactions on*, IEEE, v. 17, n. 6, p. 734–749, 2005. Citado 3 vezes nas páginas 23, 24 e 27.
- 14 BALABANOVIC, M.; SHOHAM, Y. Fab: Content-based, collaborative recommendation. *Communications of the ACM*, v. 40, p. 66–72, 1997. Citado 2 vezes nas páginas 24 e 27.
- 15 WEI, K.; HUANG, J.; FU, S. A survey of e-commerce recommender systems. In: IEEE. *Service Systems and Service Management, 2007 International Conference on*. [S.l.], 2007. p. 1–5. Citado 3 vezes nas páginas 24, 26 e 28.
- 16 SCHAFER, J. B.; KONSTAN, J. A.; RIEDL, J. E-commerce recommendation applications. *Data Mining and Knowledge Discovery*, v. 5, p. 115–153, 2001. Citado na página 24.
- 17 LOPS, P.; GEMMIS, M. de; SEMERARO, G. Content-based recommender systems: State of the art and trends. In: *Recommender Systems Handbook*. [S.l.]: Springer, 2011. p. 73–105. Citado na página 24.
- 18 LINDEN, G.; SMITH, B.; YORK, J. Amazon.com recommendations: Item-to-item collaborative filtering. *Internet Computing, IEEE, IEEE*, v. 7, n. 1, p. 76–80, 2003. Citado na página 25.
- 19 BURKE, R. Hybrid web recommender systems. In: *The adaptive web*. [S.l.]: Springer, 2007. p. 377–408. Citado na página 25.
- 20 LEE, J.; SUN, M.; LEBANON, G. A comparative study of collaborative filtering algorithms. *arXiv preprint arXiv:1205.3193*, 2012. Citado na página 26.
- 21 TUTOL, L. *Amazon Launches ‘Login and Pay with Amazon’ for a Seamless Buying Experience*. 2013. Disponível em: <<http://services.amazon.com/post/Tx2A98P3EKP62O2/Amazon-Launches-Login-and-Pay-with-Account-for-a-Seamless-Buying-Experience>>. Citado na página 26.
- 22 PALLADINO, V. *Amazon sold 426 items per second in run-up to Christmas*. 2013. Disponível em: <<http://www.theverge.com/2013/12/26/5245008/amazon-sees-prime-spike-in-2013-holiday-season>>. Citado na página 26.
- 23 FENNELL, J. Collaborative filtering on sparse rating data for yelp. com. 2009. Citado na página 27.
- 24 LOPS, P.; GEMMIS, M. de; SEMERARO, G. In: *Recommender Systems Handbook*. [S.l.]: Springer, 2011. Citado na página 28.
- 25 DEBNATH, S.; GANGULY, N.; MITRA, P. Feature weighting in content based recommendation system using social network analysis. In: ACM. *Proceedings of the 17th international conference on World Wide Web*. [S.l.], 2008. p. 1041–1042. Citado 4 vezes nas páginas 28, 35, 41 e 43.

- 26 A Guide To The Project Management Body Of Knowledge (PMBOK Guides). [S.l.]: Project Management Institute, 2004. ISBN 193069945X, 9781933890517. Citado na página 31.
- 27 LARMAN, C.; BASILI, V. R. Iterative and incremental development: A brief history. *Computer*, IEEE Computer Society, Los Alamitos, CA, USA, v. 36, n. 6, p. 47–56, 2003. ISSN 0018-9162. Citado na página 31.
- 28 MOVIELENS. *MovieLens 100k Dataset*. 1998. Disponível em: <<http://files.grouplens.org/datasets/movielens/ml-100k-README.txt>>. Citado na página 32.
- 29 WICKHAM, H. *Movies dataset*. 2006. Disponível em: <<http://docs.ggplot2.org/0.9.3/movies.html>>. Citado na página 32.
- 30 NG, A. Y. Preventing "overfitting" of cross-validation data. In: . [S.l.: s.n.]. Citado na página 33.
- 31 SARWAR, B. et al. Item-based collaborative filtering recommendation algorithms. In: ACM. *Proceedings of the 10th international conference on World Wide Web*. [S.l.], 2001. p. 285–295. Citado na página 36.
- 32 HERLOCKER, J. L. et al. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems (TOIS)*, ACM, v. 22, n. 1, p. 5–53, 2004. Citado na página 39.
- 33 CREMONESI, P.; KOREN, Y.; TURRIN, R. Performance of recommender algorithms on top-n recommendation tasks. In: ACM. *Proceedings of the fourth ACM conference on Recommender systems*. [S.l.], 2010. p. 39–46. Citado na página 39.
- 34 HOPE, C. *Epoch*. 2014. Disponível em: <<http://www.computerhope.com/jargon/e/epoch.htm>>. Citado na página 44.
- 35 LOPS, P.; GEMMIS, M. de; SEMERARO, G. Advances in collaborative filtering. In: *Recommender Systems Handbook*. [S.l.]: Springer, 2011. p. 145–184. Citado 2 vezes nas páginas 49 e 69.
- 36 COOK, J. D. *Benchmarking C++, Python, R, etc.* 2014. Disponível em: <<http://www.johndcook.com/blog/2014/06/20/benchmarking-c-python-r-etc/>>. Citado na página 64.