UNIVERSIDADE FEDERAL DE ITAJUBÁ
PROGRAMA DE PÓS-GRADUAÇÃO EM
CIÊNCIA E TECNOLOGIA DA COMPUTAÇÃO

RSE: um Framework para Avaliação de Desempenho de Sistemas de Recomendação

Paulo Vicente Gomes dos Santos

Durante o desenvolvimento deste trabalho o autor recebeu auxílio financeiro da CAPES e FAPEPE

Itajubá, 10 de maio de 2019

UNIVERSIDADE FEDERAL DE ITAJUBÁ
PROGRAMA DE PÓS-GRADUAÇÃO EM
CIÊNCIA E TECNOLOGIA DA COMPUTAÇÃO

RSE: um Framework para Avaliação de Desempenho de Sistemas de Recomendação

Paulo Vicente Gomes dos Santos

Dissertação aprovada por banca examinadora em 02 de Maio de 2019, conferindo ao autor o título de **Mestre em Ciência e Tecnologia da Computação.**

Banca Examinadora:
Prof. Dr. Bruno Tardiolle Kuehne (Orientador)
Prof. Dr. Luiz Henrique Nunes - IFSP
Prof. Dr. Edvard Martins de Oliveira - UNIFEI

Itajubá
2019

Agradecimentos

Agradeço a minha mãe e minha irmã pelo apoio incondicional nesta caminhada. Ao professor e orientador Bruno Kuehne por me mostrar uma direção a seguir, e pela ajuda e compreensão que enriqueceram grandemente o projeto. Ao professor Stephan Reiff-Marganiec, que em visita ao Brasil disponibilizou parte do seu tempo para propor melhorias ao trabalho. Ao meu primo Rafael pela revisão do artigo. Aos meus demais familiares, professores do Poscomp, colegas de mestrado e GPESC, e pessoal da DDMX. Ainda que indiretamente, todos contribuíram para que este projeto se tornasse realidade.

"Stay hungry. Stay foolish."
(Stewart Brand)

Resumo

Os sistemas de recomendação são filtros que sugerem produtos de interesse para seus clientes, podendo assim causar um grande impacto nas vendas. Atualmente existe uma variedade desses algoritmos, sendo importante escolher a opção mais adequada ao problema em questão. Isso, no entanto, não é uma tarefa trivial. Nesse contexto é proposto o *RSE* (*Recommender Systems Evaluator*): um *framework* que realiza a avaliação de desempenho *offline* dos sistemas de recomendação. O uso da metodologia apropriada é fundamental ao fazer uma avaliação. No entanto isso é frequentemente negligenciado, levando a resultados inconsistentes. O RSE procura abstrair ao máximo a complexidade envolvida no processo, e se baseia em conceitos estatísticos para proporcionar conclusões mais robustas. Os estudos realizados comprovam a sua eficácia, mostrando inclusive que ele pode ser adaptado para ser usado em outros contextos além dos sistemas de recomendação.

Palavras-chaves: Sistemas de Recomendação. Avaliação de Desempenho. Big Data.

Abstract

Recommender systems are filters that suggest products of interest to customers, which may cause positive impact on sales. Nowadays there are a multitude of algorithms, and it is important to choose the most suitable option given a situation. However, it is not a trivial task. In this context, we propose the *Recommender Systems Evaluator* (*RSE*): a framework aimed to accomplish an offline performance evaluation of recommender systems. We argue that the usage of a proper methodology is crucial when evaluating. Yet it is frequently overlooked, leading to inconsistent results. RSE hides the complexity involved in the evaluation and is based on statistical concepts to provide reliable conclusions. Studies conducted proved its effectiveness, demonstrating that it can be adapted to be used in another context rather than recommender systems.

Key-words: Recommender Systems. Performance Evaluation. Big Data.

Lista de ilustrações

Lista de quadros

Lista de tabelas

Lista de códigos

Lista de abreviaturas e siglas

Amp.IC%	Amplitude máxima do intervalo de confiança em porcentagem
Cb.Cat.	Cobertura de Catálogo
Cb.Us.	Cobertura de Usuários
Exp.	Experimento
GPS	*Global Positioning System*
IDC	*International Data Corporation*
IDF	*Inverse Document Frequency*
JVM	*Java Virtual Machine*
nDCG	*Normalized Discounted Cumulative Gain*
NoSQL	*Not Only SQL*
NPS	*Non Personalized Score*
Prod.	Produtividade
RAM	*Random-Access Memory*
RMSE	*Root Mean Square Error*
RSE	*Recommender Systems Evaluator*
SED	*Stream Editor*
SQL	*Structured Query Language*
TF	*Term Frequency*
TI	Tecnologia da Informação

Lista de símbolos

R	Matriz de avaliações
U	Conjunto de todos os usuários
I	Conjunto de todos os itens
s	Saída do experimento
q_F	Efeito produzido pelo fator F
SST	Soma total dos quadrados
SSF	Soma dos quadrados do fator F
F_{inf}	Influência do fator F
H	Largura do intervalo de confiança
N	Quantidade de repetições
$t_{\alpha,N-1}$	Distribuição *Student* com $N-1$ graus de liberdade e probabilidade de erro *alpha*
σ	Desvio padrão
Y	Média
$\omega_{a,u}$	Correlação de *pearson* entre dois elementos
$r_{u,i}$	Avaliação do usuário u no produto i
$\bar{r_u}$	Média de avaliações do usuário u
$s_{u,i}$	previsão ou *score* do usuário u no item i
V	Conjunto de vizinhos do usuário ou produto em questão
f	Frequência do termo do documento
$card$	Cardinalidade
df	Quantidade de documentos cujo termo aparece
μ	Média global
α	Parâmetro de amortecimento

$Rec(u)$	Lista de recomendação do usuário em questão
$TS(u)$	Histórico do usuário em questão
$P@n$	Precisão sob a lista de recomendação
$R@n$	Revocação sob a lista de recomendação
E	Conjunto de usuários que participaram do experimento
T_L	Tamanho padrão da lista
$TResp$	Tempo de resposta
$TRec$	Tempo gasto para fazer a recomendação
$TRespFila$	Tempo de resposta com fila
$TEspFila$	Tempo de espera na fila
GB	Gigabyte

Sumário

1 Introdução

Os Sistemas de Recomendação procuram sugerir itens de interesse para seus usuários, aumentando assim as chances de consumo por parte dos mesmos. Isso é feito principalmente através da análise de históricos, perfis e preferências. Seu uso é bastante comum nos mais diversos ambientes, podendo ser encontrado em lojas digitais de comércio, plataformas de *streaming*, serviços de busca de restaurantes, redes sociais, e outros. Há inclusive a utilização experimental em áreas menos afins, como por exemplo, na manutenção de peças industriais (1). Essa gama de aplicações se deve ao fato de que um bom recomendador é capaz de produzir impacto significativo nas vendas, e também tem se mostrado promissor em outros campos (2). Por exemplo, 75% dos vídeos assistidos na *Netflix* e 35% das compras feitas na *Amazon* provêm dos seus respectivos sistemas de recomendação (3).

Em Sarwar et al. (4), os autores afirmam que o crescimento acelerado de informações e clientes fez com que houvessem três principais desafios para os sistemas de recomendação: produzir sugestões de alta qualidade, realizar muitas recomendações por segundo para milhões de clientes e produtos, e atingir alta cobertura em um cenário de esparsidade dos dados. Muitas vezes, é preciso trabalhar com quantidades exorbitantes de dados, volume esse conhecido como *Big Data* (Seção 1.1). Portanto, a escalabilidade é outro aspecto importante e idealmente não deve causar perda de eficiência.

Devido ao impacto financeiro, é importante que hajam formas de escolher o algoritmo mais adequado ao ambiente de trabalho. Muitas vezes suas propriedades são balanceadas, levando em consideração o custo-benefício envolvido e o objetivo em questão. Percebe-se então que é de suma importância que haja uma maneira de medir o desempenho desses sistemas de forma metódica e detalhada. De fato, a avaliação dos recomendadores é também um grande desafio da área, e embora existam muitas métricas, ainda é uma incógnita como escolher o conjunto mais adequado à situação apresentada (5). Conforme demonstrado em Gunawardana e Shani (6), o uso da métrica errada acarreta na seleção de algoritmos inapropriados.

Nesse contexto tem-se na Avaliação de Desempenho um procedimento que permite identificar pontos problemáticos em um determinado ambiente. Assim, pode-se verificar o impacto causado pelos elementos envolvidos no processo, e apresentar soluções efetivas em ordem de prioridade aos eventos mais críticos. Tudo isso contribui para melhor alocação e aproveitamento dos recursos disponíveis.

A avaliação pode ser *online* ou *offline*. No primeiro caso é testado um sistema que já esteja implantado e funcionando. No segundo, avaliam-se algoritmos que possivelmente serão colocados em produção. Então, a proposta deste trabalho é a criação do *RSE* (*Re-*

commender Systems Evaluator) - ferramenta que realiza a avaliação de desempenho *offline* dos sistemas de recomendação.

É fundamental que a comparação entre os algoritmos seja feita utilizando uma metodologia apropriada. Se isso for negligenciado podem ser obtidas conclusões errôneas. Existem trabalhos semelhantes na literatura, mas os resultados são frequentemente apresentados de forma pouco clara, não sendo possível saber se testes adequados foram considerados ou não.

A avaliação feita pelo RSE é fortemente baseada em conceitos estatísticos, utilizando um conjunto bem organizado de métricas e apresentando resultados de maneira intuitiva. Assim, espera-se contribuir no âmbito de extrair conclusões confiáveis. Como forma de validação, será apresentado um estudo comparativo dos métodos recomendadores clássicos. O *software* está disponível para a comunidade de pesquisa[1].

1.1 Big Data

O rápido crescimento da tecnologia de dispositivos eletrônicos faz com que eles fiquem cada vez menores e mais potentes, e também mais interconectados. Para se ter uma ideia, a capacidade computacional aumentou aproximadamente 6400 vezes entre 1986 e 2007 (7). Esse desenvolvimento todo representa mais do que simplesmente um progresso já esperado, sendo determinante para ocorrer uma mudança no comportamento da sociedade atual: dados estão sendo gerados freneticamente de forma nunca vista antes. Segundo estimativa da *International Data Corporation (IDC)*, uma firma de pesquisa tecnológica, eles crescem a uma taxa de 50% ao ano. Inclusive sua produção está aumentando mais rapidamente que o desenvolvimento da capacidade de armazenamento. A situação tomou tamanha proporção, que desde 2007 já não é mais possível guardar todos os dados digitais produzidos (8).

A esse fenômeno recente deu-se o nome de *Big Data*. De modo formal, *big data* retrata um conjunto de dados que não pode ser percebido, adquirido, gerenciado ou processado pelas ferramentas tradicionais de *TI* dentro de um tempo aceitável (9). O termo é também definido pelo modelo dos 4Vs:

- Volume: como já foi dito, indica a geração de enormes quantidades de dados, em proporções inimagináveis até pouco tempo atrás. Estima-se que por volta de 2.5 exabytes de dados sejam criados diariamente [2] (10). Cada ser humano é literalmente um gerador de dados ambulante, e tem-se também os sensores presentes nas casas, carros, eletrodomésticos, TVs, e muitos outros. Para se ter noção da magnitude desse fenômeno, em 1998 a *Google* já indexava 1 milhão de páginas e algumas milhões de

[1] https://github.com/paulovgs/RSEvaluator

[2] 1 exabyte equivale a 1 bilhão de gigabytes, ou 10^{18} bytes.

buscas. Dez anos depois, esse número cresceu para mais de um trilhão de páginas e 3.5 bilhões de buscas feitas diariamente (11). Em 2011, apenas dois dias foram suficientes para produzir mais dados do que a quantidade criada desde o início da civilização até 2003 (9).

- Velocidade: é imprescindível que a coleta e análise de *big data* seja rápida. Seu processamento em tempo real ou perto disso faz com que uma empresa adquira vantagens significativas em relação a seus concorrentes (10).

- Variedade: essa massa de informações vem das mais diversas formas possíveis, desde dados bem estruturados até aqueles completamente sem nenhuma estrutura. Ela toma a forma de mensagens de texto, áudio, vídeo, imagens, leitura de sensores, *GPS* e muito mais.

- Valor: um dos principais desafios da área é justamente como retirar informações proveitosas dessa imensa massa de dados. O fato é que *big data* quando processado de maneira adequada faz toda a diferença. Por exemplo, entrevistas foram realizadas com executivos de 330 empresas norte-americanas, além da coleta de informações dos seus relatórios anuais. Pôde-se concluir que quanto mais as organizações se caracterizavam como orientada-a-dados, maiores eram seus rendimentos financeiros e resultados operacionais (10). Outro episódio relatado foi que a análise de *big data* na América economizou 300 bilhões de dólares em gastos médicos (9).

Pode-se dizer que o relacionamento entre *big data* e sistemas de recomendação é uma via de mão dupla. Os recomendadores são valiosos nessa nova era, uma vez que um de seus objetivos é justamente diminuir a sobrecarga trazendo somente as informações e serviços mais importantes. Por outro lado, *big data* faz com que a modelagem das preferências dos usuários seja mais precisa e melhor entendida (12). Um fato que justifica essa importância, foi que a *Netflix* lançou em 2006 uma competição que tinha por objetivo criar um recomendador que ultrapassasse em acurácia seu sistema da época. A equipe vencedora atingiu uma melhoria de 10.06% em relação ao antigo e recebeu o grande prêmio de 1 milhão de dólares[3].

1.2 Organização do Texto

O restante do documento está organizado da seguinte maneira: o Capítulo 2 apresenta detalhes sobre os sistemas de recomendação. No Capítulo 3 a avaliação de desempenho é explicada. Na sequência (4) o *RSE* é apresentado. O Capítulo 5 demonstra a utilização do *framework* na prática. Os resultados produzidos pela ferramenta são apresentados em

[3] Netflix Prize <https://www.netflixprize.com/>. Acesso em: 28 ago. 2018.

6. O Capítulo 7 ilustra uma utilização atípica do *software* em um contexto diferente. Por fim, as conclusões e proposições futuras são apresentadas no Capítulo 8.

2 Sistemas de Recomendação

A maioria das pessoas se indagadas sobre o que é um sistema de recomendação, possivelmente diriam que não sabem do que se trata. Mas o fato é que, muito provavelmente, elas já tenham tido uma (ou várias) experiências com esses algoritmos. Isso porque eles se encontram em uma infinidade de aplicações. Se a pessoa já comprou *online*, usa redes sociais, ou mesmo assistiu alguns vídeos na *Internet*, ela já se deparou com um desses algoritmos, mesmo que não tenha percebido. Os recomendadores são filtros que buscam sugerir itens de interesse para seus usuários, aumentando assim as chances de consumo, e consequentemente o lucro ou outra vantagem desejada. Eles aparecem muitas vezes junto com frases características, como "poderá também gostar de" ou "recomendados para você". A Figura 1 ilustra de forma cômica uma dessas situações.

Figura 1 – Ilustração parodiada de um recomendador[1]

Os sistemas de recomendação beneficiam os clientes permitindo encontrar produtos que eles gostam, e em contrapartida ajudam os negócios proporcionando mais vendas (13). A Figura 2 ilustra um sistema contendo 5 pessoas e 4 livros. Ela esquematiza seus elementos principais, que são usuários, itens e suas relações. Estas representam as preferências das pessoas nos produtos. São dadas também informações adicionais de perfis, atributos e conteúdos. Basicamente, o que o algoritmo faz é combinar o que está disponível para modelar os gostos dos seus clientes, e assim ser capaz de fazer sugestões. As Seções 2.1, 2.2 e 2.3 apresentam as principais classificações dos sistemas de recomendação.

[1] Disponível em: <https://www.rinapiccolo.com/piccolo-cartoons/>. Acesso em 16 out. 2018.

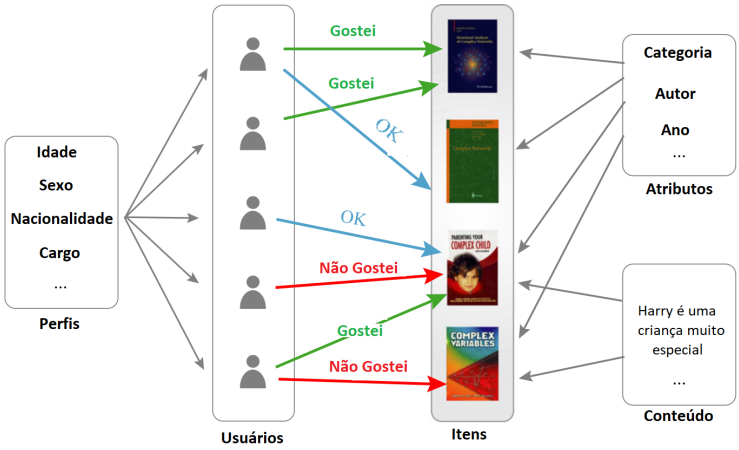


Figura 2 – Principais elementos dos sistemas de recomendação
Fonte adaptada: (5)

2.1 Recomendações Não-Personalizadas

Esse grupo diz respeito a sistemas cujas recomendações não são customizadas para um usuário específico. Portanto, a mesma informação é mostrada para várias pessoas. Há situações em que pouco ou nada se sabe sobre um usuário. Ainda assim é desejável apresentar sugestões, pois elas aumentam as chances de consumo, e se isso de fato ocorrer o sistema é alimentado para futuras possíveis customizações. Para cumprir sua tarefa, os recomendadores não personalizados se valem de dados do produto como popularidade, média de avaliação, se é ou não lançamento, entre outros. Eles são relativamente rápidos de serem computados.

A Figura 3 exemplifica esse caso. Ela mostra a distribuição dos usuários por nota, quantidade total e média de avaliação na página de um dos produtos vendidos pela *Amazon*. Embora nesse caso o sistema não esteja fazendo sugestões, isso é também considerado uma recomendação, pois as informações apresentadas ajudam na tomada de decisão dos usuários. E de fato ele é não personalizado, pois todas as pessoas que estiverem na página desse produto estarão vendo as mesmas informações. Adicionalmente, podem ser mostrados também comentários dos clientes.

Figura 3 – Recomendador não personalizado na página de um dos produtos da *Amazon*[2]

2.2 Recomendações Semi-Personalizadas

As recomendações semi-personalizadas dizem respeito ao conhecimento momentâneo de certa preferência do usuário. Podem ser ainda, personalizações feitas para grupos de clientes. Supondo-se um cenário em que uma pessoa procura por músicas populares em um ambiente dominado por um estilo musical que não reflete o seu gosto. Nesse caso e em outros semelhantes, é interessante dividir a recomendação em grupos, usando informações como situação sócio-econômica, localização, entre outras. Cria-se assim, uma personalização fraca.

Uma segunda situação diz respeito às associações entre produtos, em que são mostrados itens fortemente relacionados àquele que se está pesquisando no momento. Embora tenha um certo nível de customização, ele não reflete o conhecimento das preferências do usuário, e sim o fato de que ele está naquele momento pesquisando determinado produto. Um exemplo é mostrado na Figura 4.

Figura 4 – Associação entre produtos [3]

2.3 Recomendações Personalizadas

As recomendações personalizadas representam as sugestões específicas para cada usuário. Este grupo é o mais amplo, pois são muitas as possibilidades. Para entender de forma detalhada o seu funcionamento, é preciso primeiramente definir a matriz R. Seja U o conjunto de todos os usuários e I o conjunto de todos os itens, a matriz $U \times I \rightarrow R$ representa o grupo de avaliações (ou *ratings*) dadas por um usuário $u \in U$ em um item $i \in I$. Estas avaliações são expressas em uma escala numérica que diz o grau de empatia da pessoa em determinado produto. As células preenchidas indicam que a pessoa já consumiu o item e

[2] Disponível em: <https://www.amazon.com.br/As-Cr%C3%B4nicas-N%C3%A1rnia-Brochura-Lewis/product-reviews/857827069X/ref=dpx_acr_txt?showViewpoints=1>. Acesso em 30 ago. 2018.

[3] Disponível em: <https://www.amazon.com.br/As-Cr%C3%B4nicas-N%C3%A1rnia-Brochura-Lewis/dp/857827069X/ref=cm_cr_arp_d_product_top?ie=UTF8>. Acesso em 30 ago. 2018.

expressou seu gosto. As células vazias são as possíveis recomendações e para as quais o sistema deve gerar predições. Um exemplo pode ser visto na Tabela 1.

Até mesmo sistemas que não utilizam preferências podem se valer da matriz R como forma de consulta do histórico dos usuários. Nesse caso, uma entrada significa que o item foi consumido pela pessoa em questão, mas o valor da avaliação é ignorado. A seguir são apresentados os principais métodos das recomendações personalizadas.

2.3.1 Filtragem Colaborativa

A filtragem colaborativa é uma das formas mais difundidas desses algoritmos. Ela é feita com base nos vizinhos, que representam o grupo de usuários com preferências mais semelhantes àquele que se deseja recomendar. Desta maneira, se assume que pessoas com gostos parecidos no passado tendem a concordar novamente no futuro (5). Os vizinhos são encontrados mensurando a correlação ou grau de similaridade entre todos os pares de usuários do sistema. Este cálculo pode ser feito de diversas maneiras. A correlação de *Pearson* e similaridade pelo cosseno (14) são duas das mais utilizadas (6). Para ilustrar esse raciocínio, será apresentado um exemplo simplificado.

Considere a matriz R da Tabela 1 com três pessoas e três filmes. Deseja-se saber se *Carol* teria interesse por *Casablanca*. Olhando para as avaliações dos outros filmes percebe-se que *Alice* é vizinha de *Carol*, pois elas têm preferências parecidas, ao passo que *Bob* e *Carol* possuem gostos contrários. Assim, o filme *Casablanca* seria uma possível recomendação nesse contexto, uma vez que *Alice* o avaliou com nota alta.

Tabela 1 – Matriz R com escala de 1 a 5

	Alice	**Bob**	**Carol**
Titanic	5	1	5
Toy Story	1	5	2
Casablanca	4	2	?

Fonte adaptada: (5)

De fato, essa situação é refletida no cálculo da similaridade de *Pearson* apresentado na Tabela 2. Ele varia de -1 a +1, o par é mais similar quanto mais próximo ele for de 1, e valores negativos indicam controvérsias nas preferências. Pode ser notado uma alta correlação entre *Alice* e *Carol*, confirmando a análise visual realizada.

Tabela 2 – Similaridade de *Pearson* entre os usuários

	Alice	**Bob**	**Carol**
Alice	–		
Bob	-1	–	
Carol	0.96	-0.96	–

Fonte: Elaborado pelo autor

O método apresentado é chamado de *filtragem colaborativa usuário-usuário*. Há ainda um segundo raciocínio que é baseado na vizinhança entre os itens ao invés de usuários, e é chamado de *filtragem colaborativa item-item*. Se fosse esse o caso, a correlação ficaria como mostrado na Tabela 3. *Casablanca* e *Titanic* possuem alta correlação, e como *Carol* deu nota máxima para este último filme, *Casablanca* novamente seria recomendado. Perceba entretanto, que em um sistema real os resultados produzidos por essas duas abordagens podem diferir.

Tabela 3 – Similaridade de *Pearson* entre itens

	Titanic	**Toy Story**	**Casablanca**
Titanic	–		
Toy Story	-0.98	–	
Casablanca	0.79	-0.86	–

Fonte: Elaborado pelo autor

2.3.2 Filtragem Baseada em Conteúdo

Na filtragem baseada em conteúdo, os itens recomendados são aqueles similares em conteúdo a outros que o usuário gostou no passado, ou até mesmo produtos que casam com atributos da pessoa (15). Por exemplo, tendo informações de gênero em um recomendador de músicas, e se o usuário gostou de *"The Final Countdown"* e *"You Give Love a Bad Name"*, é possível que uma inferência para *Hard Rock* seja feita e portanto seja sugerido *"Walk This Way"*.

Embora esse tipo de filtragem possa ser implementada de várias maneiras, é importante destacar três propriedades: a descrição dos itens, a criação do perfil do usuário e uma forma de compará-los para decidir o que recomendar (16). Na primeira etapa, os itens são representados por seus atributos, que são informações textuais como palavras-chave (palavras pré-definidas que podem assumir um conjunto limitado de valores) e *tags* (palavras dadas livremente por pessoas ou extraídas de um texto). A criação do perfil do usuário pode ser feita através da junção dos atributos dos itens contidos em seu histórico, e para os quais ele demonstrou interesse. Outra alternativa seria permitir por meio de uma interface gráfica que a pessoa construa manualmente a representação de suas preferências. Uma vez que os perfis de itens e usuários foram criados, é preciso compará-los para que o sistema produza a recomendação. Eles são representados como vetores multidimensionais (chamados também de espaços vetoriais), e novamente pode ser usado a computação do cosseno do ângulo entre os dois.

Considere a Figura 5, que mostra o perfil de dois filmes (*Filme* α e *Filme* β) e duas pessoas (*Maria* e *Pedro*) em um espaço vetorial bidimensional. Nesse contexto, *Filme* α seria uma recomendação para *Maria*, pois ambos possuem altas doses de romance e pouca

ação. Na prática, no entanto, as dimensões do espaço vetorial são muito maiores, mas o raciocínio é o mesmo.

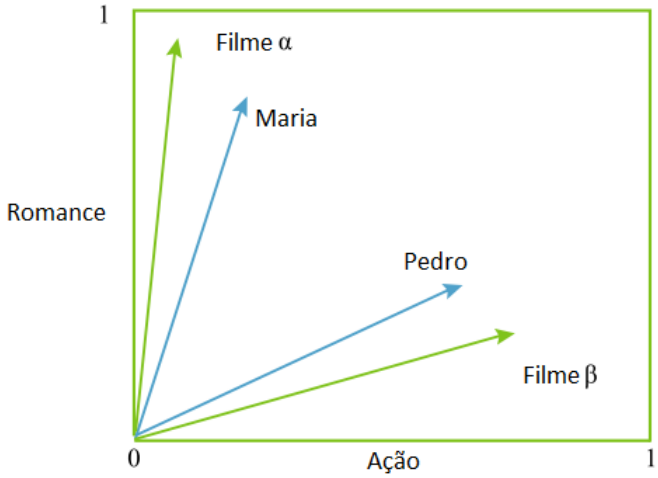


Figura 5 – Itens e usuários representados em um espaço vetorial 2D
Fonte adaptada: (5)

Para que as dimensões sejam criadas de forma adequada, alguns pré-processamentos podem ser necessários. Se os atributos forem advindos de origens públicas, é interessante realizar um procedimento conhecido como *stopping*. Trata-se da remoção de palavras frequentes que não carregam nenhum sentido (pronomes, preprosições, conjunções), como por exemplo "o" e "para" (17). Outra possibilidade seria agrupar palavras com o mesmo significado, de acordo com algum critério de similaridade. Este processo é conhecido como *stemming*, e seu objetivo é criar um termo que reflete o significado geral de coisas como "computação", "computador" e "computadores", fazendo com que elas tenham uma única representação (16). A escolha desse critério deve ser feita com cuidado. Um agrupamento fraco pode não ser eficaz para a maioria dos atributos, ao passo que um agrupamento muito forte acarreta também em efeitos colaterais, como por exemplo entender que "final feliz" e "final infeliz" tenham a mesma representação. Além disso, é preciso ter que lidar com sinônimos (palavras diferentes que têm o mesmo significado), polissemias (a mesma palavra possui mais de um significado) e contexto em que o termo foi usado (16).

Um terceiro conceito bastante utilizado no pré-processamento textual é o *TF*IDF*. O *TF* (*term-frequency*) simplesmente conta a frequência com que a palavra aparece no documento atual. Esse valor é multiplicado pelo *IDF* (*inverse document-frequency*), que representa a raridade da palavra em relação a todos os documentos da coleção. Assim, *TF*IDF* associa um número real a um termo para mensurar a sua relevância em um dado contexto (16).

Apesar da criação dos perfis vetoriais não ser trivial, uma das vantagens da recomendação baseada em conteúdo é que ela não necessita conhecer as preferências dos outros usuários, e por ser fácil mostrar porque determinado item foi recomendado.

2.3.3 Recomendação Baseada em Conhecimento

A recomendação baseada em conhecimento faz uso do conhecimento de domínios específicos e sobre o quão útil um item é para o usuário (18). Ela é utilizada em situações em que os produtos não são consumidos com frequência, ou o tempo é uma variável importante. Por exemplo, a compra de um carro não se dá com a mesma regularidade com que se lê um livro, e a situação econômica de uma pessoa pode variar com o passar dos anos. Esses fatores influenciam na recomendação. Esse tipo de recomendação não necessita do histórico de avaliações, pois o foco está na filtragem de informações e interação humana.

Há basicamente duas linhas de raciocínio: descoberta de similaridade e aplicação ajustada. No primeiro caso a pessoa escolhe um produto do catálogo, uma grande quantidade de candidatos é pega do banco de dados, ordenada por similaridade e as melhores sugestões são mostradas. Já na aplicação ajustada o raciocínio é parecido, com a exceção de que antes da ordenação os candidatos são filtrados para deixar apenas aqueles que satisfazem o critério de ajuste (19).

A Figura 6 mostra um exemplo. A pessoa escolhe um restaurante como ponto de partida, provavelmente um que lhe seja conhecido ou que tenha características que ela aprecia. O sistema então sugere um local similar ao escolhido. Além disso, o usuário tem a opção de refinar a busca escolhendo opções como “Mais Barato”, “Mais Alegre”, entre outras. Ele pode realizar sucessivas interações até ficar satisfeito com o item recomendado.

Figura 6 – Recomendador baseado em conhecimento
Fonte adaptada: (20)

2.3.4 Considerações

Fato é que não existe unanimidade na escolha do melhor método, pois seu uso varia muito do contexto e densidade dos dados disponíveis (5). Todas as técnicas mostradas possuem seus pontos fortes e também suas fraquezas. Segundo os autores de (21), a filtragem colaborativa possui três principais vantagens em relação ao sistema baseado em conteúdo: permite filtrar itens cujo conteúdo não é facilmente analisado por processos automatizados, pode escolher produtos baseado na qualidade e gosto, e é capaz de produzir recomendações mais inusitadas. Entretanto, precisa ser inicializada com grande quantidade de informações para produzir boas recomendações. Quando há pouco ou nenhum histórico dos usuários, o sistema tem dificuldade em fazer as sugestões. Esse problema é conhecido como *cold-start* (22). Além disso, o sistema se torna mais preciso com o crescimento da quantidade usuários, e o consequente aumento do número de avaliações (23). Por fim, a abordagem usuário-usuário não escala bem devido ao gargalo da busca pelos

vizinhos. Os sistemas item-item exploram primeiramente as relações entre os produtos e evitam esse problema, levando assim à criação de recomendadores *online* mais velozes (4).

Já o recomendador baseado em conteúdo não precisa ter o conhecimento dos vizinhos e gera sugestões bastante específicas, o que pode ou não ser positivo dependendo do contexto. As vezes as informações necessárias ao seu funcionamento são difíceis de adquirir ou podem até mesmo não estar disponíveis (24). Sua grande vantagem é que é mais fácil mostrar através dele porque determinado item foi recomendado. Pode haver situações em que empresários tentem fraudar o sistema em prol de benefícios próprios. Através da criação de perfis falsos por exemplo, um dono de hotel pode tentar impulsionar as avaliações do seu próprio estabelecimento, ou fazer com que esses perfis avaliem negativamente seus concorrentes (25). Esses ataques são bastante prejudiciais às avaliações. A abordagem item-item geralmente é mais robusta do que a baseada em usuários, conforme demonstrado em (26). A filtragem baseada em conteúdo, por sua vez, não é afetada por esse tipo de situação (15).

O sistema baseado em conhecimento é mais sensível a mudanças de preferências, mas necessita de um esforço maior por parte do usuário. Todas essas abordagens possuem características distintas, o que sugere que elas sejam complementares ao invés de concorrentes (19). Portanto, na tentativa de minimizar os problemas, é comum a formação de híbridos através da combinação de duas ou mais técnicas, procurando assim aproveitar o que há de melhor em cada uma delas.

Outro ponto a ser evidenciado é que, uma vez que a gama de itens disponíveis em grandes organizações é enorme (muitas vezes ultrapassando milhões de produtos), os usuários avaliam (e conhecem) apenas uma pequena porcentagem dessa quantia. Isso faz com que a densidade da matriz de avaliações seja muito baixa. Nestes casos, é importante que se aplique técnicas de redução das dimensões. Elas ajudam a superar esses problemas transformando o espaço original altamente dimensional em outro com dimensões menores, e preservando as principais características. De fato, os resultados são tão efetivos que já é considerada mais do que simplesmente uma técnica de pré-processamento, e sim uma abordagem no projeto dos sistemas de recomendação (27).

2.4 Entrada de Dados

As informações a respeito dos itens e usuários são a matéria-prima dos sistemas de recomendação. Algumas delas são colocadas diretamente pelos empregados da plataforma. Por exemplo, com a chegada de um novo livro em uma loja virtual, um funcionário terá de entrar com as informações de autor, ano de publicação, idioma, entre outros. Outra forma de entrada é no cadastro de clientes, em que os mesmos fornecem seus dados pessoais.

Em relação às avaliações dos itens, um conceito importante é que a pessoa se sinta

convidada a fazê-las voluntariamente, tornando essa experiência agradável e alimentando o sistema ao mesmo tempo. As estrelas são uma forma de captura bastante utilizada, em que o produto é melhor avaliado quanto maior for a quantidade de estrelas atribuída a ele. Um exemplo pode ser visto na Figura 7, que mostra a média de avaliações que três produtos receberam em um recomendador *online* de filmes.

Figura 7 – Sistema de estrelas [4]

Existem também as avaliações binárias, que são conhecidas por palavras como "gostei" e "não gostei", ou desenhos que expressam a mesma intenção, conforme é mostrado na Figura 8. Tem-se ainda os sistemas unários, quando os objetos são apenas coletados (como em sistemas de compartilhamento de favoritos) ou somente consumidos (como em jornais *online* que não possuem avaliações) ou quando o "gostei" é a única opção (como no Instagram) (5).

Figura 8 – Avaliações binárias [5]

Os casos de avaliações mencionados até aqui são formas explícitas de entrada de dados. No entanto há também alternativas implícitas, como a análise dos cliques do *mouse*, tempo em que um filme permaneceu pausado, padrões de navegação no site, dentre outros. Estas são também fontes de alimentação valiosas para os sistemas de recomendação. Outra possibilidade é fazer o uso do contexto, em que dados adicionais são levados em consideração para melhorar o algoritmo. Tempo, localização e datas comemorativas são apenas algumas das diversas possibilidades. Algumas sugestões podem fazer mais sentido quando o dia está ensolarado do que quando está chovendo, por exemplo.

[4] Figura retirada ao fazer login no recomendador não comercial MovieLens. <https://movielens.org/>. Acesso em: 30 ago. 2018.

[5] Retirado de <https://www.youtube.com/watch?v=soB_zeZhVc0>. Acesso em: 21 out. 2018.

Mesmo com a captura explícita de preferências, o padrão de avaliação pode diferir entre os indivíduos. Em uma escala de 0 a 5, por exemplo, a nota 4 de uma pessoa pode ter o mesmo significado que um 3 de outra. Além disso, o comportamento dos usuários é diferente entre atividades de baixo-risco (ouvir uma música, assistir um filme, etc.) e alto risco (escolher uma casa, planejar uma viajem, etc.). Na primeira, eles tendem a confiar mais em opiniões públicas, e na segunda, tendem a gastar mais tempo interagindo com o sistema e criando seu próprio julgamento (28). Na prática a captura dessas nuances não é tão simples. Por se tratar de um campo amplo e subjetivo, a recomendação se tornou também tópico de interesse de matemáticos, físicos e psicólogos (5).

3 Avaliação de Desempenho

A avaliação de desempenho é um procedimento estatístico usado para quantificar a influência que certos elementos exercem na realização das tarefas do sistema. Ela permite que os avaliadores tirem conclusões importantes e realizem medidas corretivas ou melhorias. A avaliação busca determinar resultados relevantes, como quais são os elementos mais impactantes, se algum deles pode ser descartado, qual a sua combinação ideal, ou a melhor relação custo-benefício possível. Por exemplo, em um sistema cujo os usuários reclamam de lentidão, a avaliação de desempenho auxilia na identificação dos possíveis pontos de falha. Assim seria possível modificar somente os lugares mais críticos, obtendo considerável melhoria na performance e evitando gastos desnecessários. Ela pode ser usada também para comparar diferentes métodos e decidir qual será implantado. Seus termos principais estão especificados a seguir:

- Fatores: são variáveis de entrada que podem influenciar nos resultados observados. Os diferentes valores que eles assumem em um teste são chamados de *níveis*. Quando o fator possui um único nível ele é denominado *parâmetro de entrada*.
- Experimento: representa uma combinação específica e planejada entre os fatores e seus níveis, a qual irá produzir uma saída.
- Variáveis de resposta: representam a saída do experimento, ou a resposta apresentada frente a combinação dos valores de entrada. É o que se deseja medir na avaliação de desempenho.
- Replicação: diz respeito à quantidade de repetições realizada em cada experimento. Mais detalhes são apresentados na Seção 3.2.

3.1 Planejamento da Avaliação

Para que a avaliação seja realizada, é preciso definir os fatores e variáveis de resposta que serão utilizados, os quais dependem do objetivo específico do teste. Por exemplo, imagine que deseja-se medir o desempenho no preparo de café expresso. As variáveis de resposta poderiam ser o tempo de preparo, temperatura do café na xícara e cremosidade. Os fatores e níveis estão especificados no Quadro 1.

Quadro 1 – Fatores e níveis para o exemplo do preparo de café expresso

Fator	**Nível**		
	1	**2**	**3**
Aroma	Frutado	Balanceado	Intenso
Tamanho do café	Pequeno	Grande	—
Máquina	Nespresso	Dolce Gusto	—

Fonte: Elaborado pelo autor

Como pode ser observado, o propósito dessa avaliação é determinar o impacto que os fatores aroma, tamanho e máquina, quando combinados de maneiras diferentes, causam na temperatura do café, na sua cremosidade e tempo de preparo. É preciso também escolher uma estratégia para a realização do teste. Muitas podem ser encontradas na literatura, dentre as quais se destacam o planejamento simples, o planejamento fatorial completo e o planejamento fatorial parcial (29).

- Planejamento simples: inicia-se com uma configuração inicial e os fatores são alterados um por vez. No exemplo dado, seriam necessárias 5 combinações. Embora seja fácil de implementar, não é estatisticamente eficiente e não permite observar a interação entre os fatores, o que pode acarretar em conclusões erradas.
- Planejamento totalmente fatorial: esse caso explora todas as combinações possíveis entre os fatores e os seus níveis. No exemplo tem-se duas máquinas, três aromas e dois tamanhos diferentes, resultando em 12 testes.
- Planejamento fatorial parcial: embora o caso anterior permita verificar o efeito entre todos os fatores, a grande quantidade de experimentos e consequentemente o custo ou tempo gasto podem inviabilizar o projeto. É altamente recomendado reduzir a quantidade de níveis e utilizar pequena quantidade de fatores, se possível. O fatorial parcial é uma simplificação do planejamento totalmente fatorial.

A estratégia implementada neste trabalho é um tipo particular de fatorial parcial, em que todos os níveis são fixados em 2. Muitas vezes o efeito de um fator é unidirecional, fazendo com que a performance aumente (ou diminua) continuamente a medida que este varie do seu mínimo para seu máximo. Assim, testar os valores mínimo e máximo do fator ajuda a determinar se a diferença nos resultados é significativa o suficiente, o que justificaria um experimento mais detalhado (29). Portanto, o modelo fatorial parcial com 2 níveis foi escolhido por ser mais simples e ainda assim permitir tirar conclusões significativas.

No caso do café o aroma possui 3 níveis. Ele teria então um destes níveis descartados, fazendo-se necessária a realização de 8 testes.

3.2 As Repetições

Conforme dito no exemplo anterior, seriam feitos 8 experimentos para o planejamento implementado, uma observação para cada combinação dos fatores. Fato é que todo teste está sujeito a erros experimentais não controlados. Inclusive o próprio processo de medição introduz erros na maioria das vezes. Então corre-se um risco muito grande que essa única observação esteja errada, e se isso acontecesse tudo mais estaria errado, até mesmo as conclusões. Portanto, é indispensável que cada experimento seja repetido. Isso porque um fenômeno curioso ocorre: mesmo com todas as medições erradas, o valor médio sempre estará mais próximo do valor real daquilo que se deseja medir. Portanto, quanto maior o número de réplicas, mais ele se aproximará do valor verdadeiro (30). No infinito essa média seria efetivamente o valor real.

Em contrapartida, muitas repetições podem levar a altos custos. É preciso escolher a menor quantidade de réplicas que possibilite tirar conclusões adequadas, sem que o teste seja dispendioso. No exemplo dado, caso fosse definido 5 repetições, seriam feitos então 40 testes no total.

3.3 Influência dos Fatores

Na avaliação de desempenho, é importante saber qual o impacto na saída causado por cada um dos fatores de entrada, bem como o efeito de suas interações. Mensurar a influência dos fatores permite também saber quais devem ser priorizados e se algum deles pode ser descartado.

Supondo-se um planejamento parcial realizado com dois fatores A e B, seus respectivos níveis (a_1, a_2, b_1, b_2), e uma variável de resposta S. Nesse caso são feitos 4 experimentos para cobrir as combinações possíveis. Para cada nível, atribui-se uma constante unitária multiplicativa positiva e outra negativa. Será fixado, por exemplo, que o primeiro nível seja sempre positivo e o segundo negativo. Assim, o *setup* inicial do teste fica como apresentado na Tabela 4.

Tabela 4 – *Setup* do experimento fatorial parcial

Experimento	**A**	**B**	**S**
1	1 (a_1)	1 (b_1)	$\bar{s_1}$
2	1 (a_1)	-1 (b_2)	$\bar{s_2}$
3	-1 (a_2)	1 (b_1)	$\bar{s_3}$
4	-1 (a_2)	-1 (b_2)	$\bar{s_4}$

Fonte: Elaborado pelo autor

Através de um modelo de regressão, a saída pode ser estimada em função dos efeitos

da entrada. Tem-se portanto (29):

$$s = q_0 + q_A x_A + q_B x_B + q_{AB} x_A x_B \tag{3.1}$$

As constantes q_A, q_B e q_{AB} são respectivamente, o efeito produzido pelos fatores A, B, e pela interação entre eles. As observações quando substituídas no modelo resultam (29):

$$q_0 = \frac{(\bar{s_1} + \bar{s_2} + \bar{s_3} + \bar{s_4})}{4} \tag{3.2}$$

$$q_A = \frac{(\bar{s_1} + \bar{s_2} - \bar{s_3} - \bar{s_4})}{4} \tag{3.3}$$

$$q_B = \frac{(\bar{s_1} - \bar{s_2} + \bar{s_3} - \bar{s_4})}{4} \tag{3.4}$$

$$q_{AB} = \frac{(\bar{s_1} - \bar{s_2} - \bar{s_3} + \bar{s_4})}{4} \tag{3.5}$$

Percebe-se que a saída foi multiplicada pelas constantes -1 e +1, expressando as diferentes variações dos níveis em cada experimento. Cada uma das saídas representa a média das repetições. A partir de então computa-se a soma total dos quadrados SST, que pode ser fracionada na soma dos quadrados de cada efeito. Em seguida a influência dos fatores é medida (29):

$$SST = 2^2 q_A^2 + 2^2 q_B^2 + 2^2 q_{AB}^2 = SSA + SSB + SSAB \tag{3.6}$$

$$A_{inf} = \frac{SSA}{SST} \quad B_{inf} = \frac{SSB}{SST} \quad AB_{inf} = \frac{SSAB}{SST} \tag{3.7}$$

Esse raciocínio será generalizado para k fatores. Como existem dois níveis de variação, tem-se $2^k - 1$ termos. A soma dos quadrados e influência dos fatores ficará como mostrado na Equação 3.8:

$$SST = SSA + SSB + ... + SSK + SSAB + ... + SSAK + ... + SSAB..K \tag{3.8}$$

$$SSA = 2^k q_A^2 \quad ..... \quad SSAB..K = 2^k q_{AB..K}^2 \tag{3.9}$$

Essa é uma maneira bastante útil de quantificar o efeito dos fatores e suas interações no experimento. Considerando a cremosidade no caso do café, por exemplo, poderia ser concluído que ela foi influenciada em 25% pelo aroma, 5% pelo tamanho, 45% pela máquina, 15% pelas interações entre aroma e máquina e 10% pelas relações entre todos eles[1].

[1] Valores fictícios

Esses resultados ajudam o avaliador a adquirir um conhecimento mais profundo sobre a situação apresentada, norteando o processo de avaliação de desempenho como um todo.

3.4 Intervalos de Confiança

Uma forma de interpretar o experimento consiste em resumi-lo em um número que contenha as características chave do conjunto testado. A média é comumente utilizada para essa tarefa. É importante também incluir informações sobre a variabilidade dos dados, de forma que, tendo dois sistemas com a mesma média, será geralmente preferível aquele cujo saída apresenta a menor variação. Ela será contabilizada pela largura do intervalo de confiança, dada por (29):

$$H = \frac{t_{\alpha,N-1} * \sigma}{\sqrt{N}} \tag{3.10}$$

O primeiro termo do numerador representa a distribuição *Student* com $N-1$ graus de liberdade e probabilidade de erro α, σ é o desvio padrão e N a quantidade de repetições realizada. A probabilidade de erro e intervalo de confiança são grandezas complementares. Este último está associado ao trecho que contém o valor do parâmetro estudado com determinada certeza. Por exemplo, se for dito que há 90% de confiança que a média encontra-se entre 5.35 e 6.18, 9 em cada 10 amostras estarão nessa faixa de valores.

Portanto, chega-se naquele supracitado número resumido do conjunto de dados (Y), o qual juntamente com sua variabilidade e a interação entre os fatores, possibilita que conclusões significativas sejam tiradas:

$$Y \pm H \tag{3.11}$$

Assim, dois resultados podem ser comparados de forma eficaz, permitindo identificar qual é mais eficiente ou se são estatisticamente iguais. Uma forma simples e efetiva de fazer isso é colocá-los em um gráfico de barras e interpretá-los visualmente. A Figura 9 apresenta um gráfico desse tipo mostrando três medidas com suas respectivas médias e variabilidades.

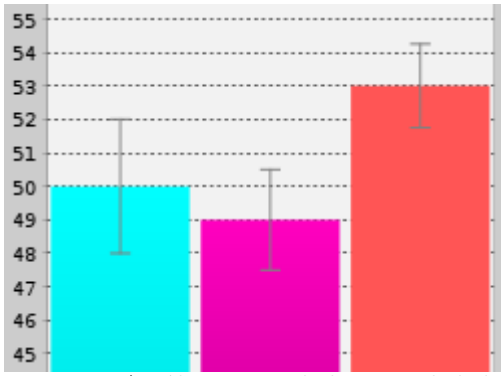


Figura 9 – Análise visual da variabilidade
Fonte: Elaborado pelo autor.

De acordo com Jain (29), as medições das barras azul e rosa são consideradas iguais, porque a variabilidade de pelo menos uma delas se sobrepõe à media da outra. Por outro lado, entre as barras rosa e laranja, a última é significativamente melhor, pois não há sobreposição. Já entre as barras azul e laranja, percebe-se que existe uma leve sobreposição, sem que a média de uma alcance a variabilidade da outra. Nesse caso não é possível tirar conclusões. Um teste com uma quantidade maior de repetições poderia eventualmente diminuir a variabilidade fazendo com que essa interseção deixasse de existir.

É importante ressaltar também que a comparação não é transitiva. Ou seja, se um experimento número 1 for considerado estatisticamente igual ao 2, e este último for igual a um terceiro ensaio, isso não implica necessariamente que os testes 1 e 3 são iguais.

3.5 *Online vs Offline*

Em relação aos sistemas de recomendação, a avaliação pode ser *online* ou *offline*, sendo que a distinção entre esses modos se dá no estado do algoritmo. Se ele já estiver produzindo recomendações em um ambiente real, a avaliação é chamada de *online*. Caso contrário, ela é dita ser *offline*.

Certas características são difíceis de medir *offline*, como a satisfação com as recomendações e a confiança no sistema (5). Muitos sistemas reais realizam testes *online*. Embora eles meçam a performance real do sistema, são custosos de ser executados (6). Por outro lado, o teste *offline* permite comparar maior quantidade de candidatos com baixo custo. É interessante que avaliações *online* sejam realizadas depois de um teste *offline* detalhado, para que o algoritmo seja colocado em um ambiente real somente depois de ter seus parâmetros bem ajustados. Esse processo reduz o risco de causar insatisfação por parte do usuário, o que poderia desencorajar as pessoas a usá-lo novamente (25).

Por isso, é crucial a realização de experimentos *offline*, assumindo que os resultados estejam correlacionados com o comportamento real dos usuários. Seu objetivo é portanto, filtrar os melhores algoritmos para que apenas os mais promissores sejam testados *online*

(6). O experimento *offline* é feito usando um conjunto de dados previamente coletado de um sistema real. Tenta-se simular da melhor maneira possível o comportamento interativo entre usuário e um sistema de recomendação. Nesses testes usa-se a premissa que o comportamento do usuário será similar o bastante com aquele apresentado quando o sistema for colocado em produção. Portanto, o *framework* proposto realiza a avaliação de desempenho *offline* dos sistemas de recomendação. Mais detalhes serão apresentados no Capítulo seguinte.

4 RSE

O RSE (*Recommender Systems Evaluator*) é um *software* escrito em Java que realiza a avaliação de desempenho *offline* dos sistemas de recomendação. Ele procura esconder ao máximo a complexidade envolvida no processo, fazendo com que o avaliador foque na interpretação dos resultados obtidos. O RSE possui ainda um conjunto amplo de fatores e variáveis de resposta, distribuições de carga e modos distintos de avaliação, proporcionando grande flexibilidade no seu uso.

A Seção 4.1 apresenta os trabalhos semelhantes ao *framework* desenvolvido. Nela, algumas características do RSE são evidenciadas em relação aos outros projetos. Na Seção 4.2 sua estrutura é mostrada, e então cada um de seus módulos é explicado em detalhes.

4.1 Trabalhos Relacionados

TagRec (31): *software* escrito em Java que conta com grande variedade de métricas e algoritmos. O seu uso é focado em recomendadores que utilizam *tags*. Ele trabalha com um conjunto de onze possíveis bancos de dados.

RiVal (32): ferramenta Java que segue um fluxo bem definido de etapas: divisão dos dados, recomendação, geração de candidatos e medição da performance. Os dados são carregados através de arquivos, e seus resultados são apresentados numericamente.

Lenskit (33): é uma terceira ferramenta feita em Java, e uma das mais completas do meio. É usada principalmente em algoritmos de filtragem colaborativa. Possui um módulo avaliador com amplo conjunto de medidas.

Outros: o trabalho feito em (34) facilita a adição de novos algoritmos através de *plugins*, e possui interface gráfica que torna simples o seu uso. Entretanto, seu conjunto de métricas é bastante limitado. Outra alternativa é chamada de Idomaar (35). Ele trata os dados de maneira dinâmica, considerando por exemplo o efeito da entrada de novos usuários. WrapRec (36) é um projeto desenvolvido em C# que busca simplificar a análise dos recomendadores. Ele possui rotinas que incorporam os algoritmos de dois *frameworks* já existentes, além de mecanismos para avaliá-los. Um representante da linguagem estatística R é o pacote recommenderlab (37), que conta com funções para avaliação e comparação entre algoritmos, e se concentra em sistemas de filtragem colaborativa.

Todas as abordagens apresentadas são valiosas no contexto da avaliação dos recomendadores. Porém, a maioria usa dados importados diretamente de arquivos. Alguns

mostram os resultados somente na forma numérica, fazendo com que a pessoa tenha um esforço maior para interpretá-los. Uma das propriedades do RSE diz respeito justamente a sua entrada de informações, contando com o armazenamento em uma estrutura padronizada de banco de dados. Assim o teste se assemelha mais a um ambiente de produção. Isso é refletido nas métricas de tempo. Ele faz uso do banco *PostgreSQL*[1], podendo então usufruir de todas as facilidades para manipulação que um sistema gerenciador de banco de dados proporciona, e possibilitando consultas mais complexas. Os resultados são mostrados na forma de gráficos diversos, facilitando a sua interpretação.

Na literatura foi observado que muitas vezes a forma de avaliação não é descrita por completo. Frequentemente não fica claro se foram ou não usados testes estatísticos. Se a metodologia aplicada não for robusta o suficiente, ela pode levar até mesmo a conclusões errôneas. Por conta do embasamento matemático envolvido, espera-se que o RSE possa contribuir proporcionando comparações e resultados confiáveis. Outra característica distinta do *framework* proposto está na forma de distribuição das requisições, que simula intervalos de tempo normais e horários de pico.

4.2 Módulos

O RSE foi projetado em módulos, para que pudesse ser estendido com facilidade. Caso o programador precise de uma funcionalidade específica ou inexistente, ele possivelmente modificará pequenas partes dentro de um módulo, deixando os outros intactos. Seu fluxo de funcionamento é dividido em quatro etapas, ilustradas na Figura 10.

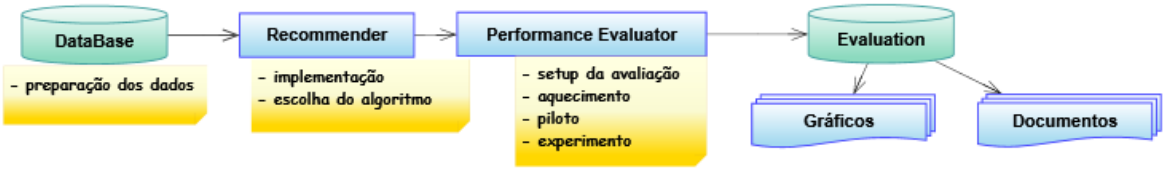


Figura 10 – Etapas de funcionamento do RSE
Fonte: Elaborado pelo autor

A primeira etapa consiste na preparação dos dados. Eles são importados para a estrutura padronizada do RSE. Em seguida, cálculos são feitos e salvos utilizando funções auxiliares do *software*, como por exemplo, a média global de avaliações por item, grau de similaridade entre usuários, e outros. A segunda etapa consiste na escolha de um dos algoritmos já existentes. Alternativamente, o avaliador pode implementar seu próprio método e adicioná-lo ao RSE. Feito isso, passa-se para a terceira parte, em que é realizada a avaliação de desempenho. Por fim, os resultados são armazenados em banco de dados, os quais podem ser apresentados por meio de gráficos ou arquivos de texto. O RSE possui também algumas ferramentas auxiliares, como a geração de *logs* por exemplo.

[1] Disponível em: <https://www.postgresql.org/>. Acesso em: 15 set. 2018.

4.3 Módulo *DataBase*

O objetivo deste módulo é encapsular todo o acesso ao banco de dados e separá-lo das demais partes do sistema. A Figura 11 mostra seu diagrama de classes[2]. A classe *Database* é usada como pilar pelas outras, pois realiza as funções primordiais da comunicação com banco de dados, como por exemplo abertura e fechamento da conexão. Já *Evaluation* possui rotinas que lidam com a avaliação de desempenho propriamente dita. Salvar os experimentos e buscar os resultados são apenas algumas de suas muitas tarefas. As funções estruturais, como criação de tabelas e importação são realizadas pela *Groundwork*.

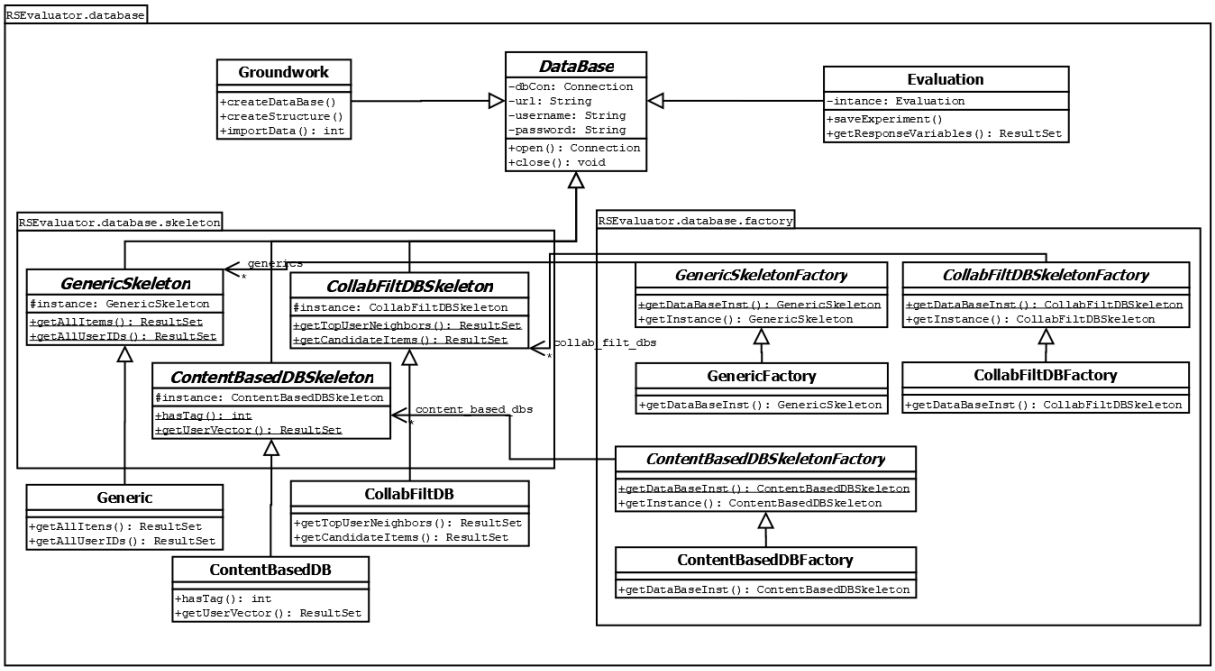


Figura 11 – Diagrama de classes simplificado do módulo *DataBase*
Fonte: Elaborado pelo autor

O restante das classes encontra-se bipartido. O pacote *skeleton* representa a assinatura dos métodos, e as classes que herdam desse grupo são suas respectivas implementações. *Generic* conta com rotinas de comunicação das relações entre usuários, itens e avaliações. *ContentBasedDB* e *CollaborativeFilteringDB* fazem a ponte entre banco e algoritmos recomendadores. Isso possibilita flexibilidade no mecanismo de funcionamento do RSE, pois é permitido modificar ou derivar uma estrutura a partir do *skeleton* sem perder o vínculo que as funções já definidas possuem com o restante do programa. Em outras palavras, o *skeleton* simboliza o conjunto mínimo de métodos indispensáveis à operação normal do RSE, e portanto não podem ser removidos. O que pode ocorrer é a sua sobrescrição ou a adição de novas funções. É evidente que *Evaluation* também é indispensável, a diferença é que a princípio, ela não deveria ser modificada. Por fim, utilizou-se do padrão de projeto Factory (38) para maior maleabilidade na criação das classes.

[2] Alguns métodos, atributos ou parâmetros foram omitidos. Isso também acontecerá nos diagramas subsequentes.

O RSE parte do princípio que o modelo mais simples de recomendação necessita de usuários, itens e suas relações (avaliações ou *ratings*). A estrutura que armazena os dados usados no teste está representada na Figura 12. Assim, todos os algoritmos usam *Items*, *Ratings* e *Users*. Adicionalmente, a filtragem colaborativa precisa das tabelas de similaridade. Já a recomendação baseada em conteúdo se vale de *Tags* e vetores de conteúdo (*User Vector* e *Item Vector*). Este é um modelo genérico que atende a maioria dos casos, mas é permitido que novas tabelas sejam adicionados pelo avaliador conforme sua necessidade.

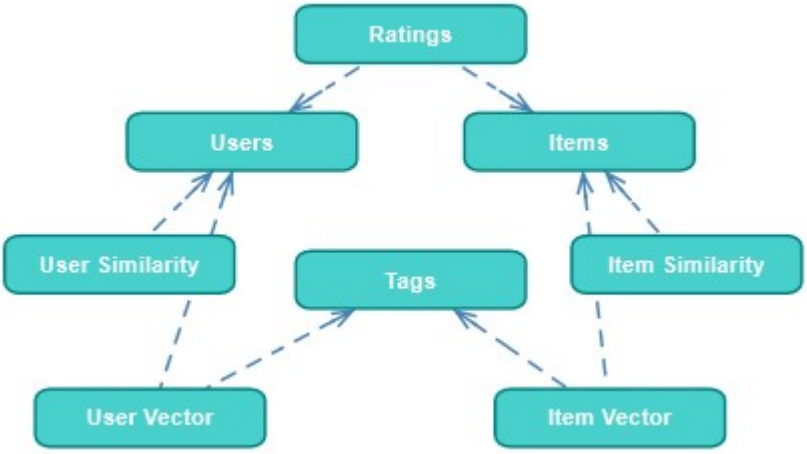


Figura 12 – Estrutura de tabelas para o conjunto de dados teste
Fonte: Elaborado pelo autor

4.4 Módulo *Recommender*

Nesse módulo estão implementados quatro métodos clássicos de recomendação, sendo que dois representam as principais variações da filtragem colaborativa, um terceiro é um recomendador baseado em conteúdo, e um híbrido. Embora existam técnicas mais avançadas, como aquelas que se baseiam em redes sociais e lógica difusa, os clássicos ainda desempenham papel importante em quase todos os tipos de aplicação (12). Há também um método alternativo e semi-personalizado. Porém, nada impede o avaliador de acoplar novos algoritmos se desejável.

A Figura 13 ilustra o diagrama de classes do módulo. Todos os recomendadores são padronizados e inicializados com um identificador do usuário alvo e outras constantes, devendo produzir uma lista com os itens recomendados. Para usar os algoritmos o programador deve simplesmente especificar o nome da classe no arquivo de configuração, bem como os parâmetros necessários. Caso ele queira fazer uma nova implementação, ele deve criar uma classe estendendo *Recommender* e sua fábrica correspondente. Então, terá de implementar a função *score()*, que vai produzir uma lista com os itens recomendados e suas respectivas predições.

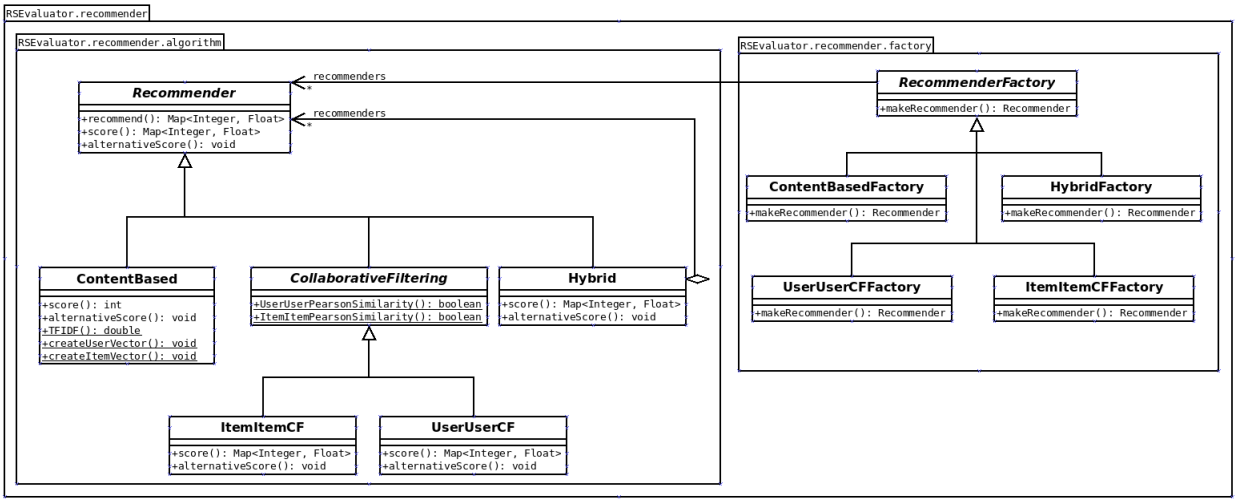


Figura 13 – Diagrama de classes simplificado do módulo *Recommender*
Fonte: Elaborado pelo autor

4.4.1 Implementação da Filtragem Colaborativa Usuário-Usuário

Na filtragem colaborativa três etapas principais são efetuadas: cálculo da similaridade, escolha da vizinhança e candidatos, e cálculo das predições. Para a primeira delas, vários modelos matemáticos podem ser encontrados na literatura. A correlação de *Pearson* foi a escolhida, por apresentar resultados mais satisfatórios do que a similaridade pelo cosseno no ambiente de recomendação (39). A Equação 4.1 calcula o coeficiente de *Pearson* para dois usuários $a, u \in U$ (21). Os itens e as avaliações são representados por $i \in I$ e $r \in R$, respectivamente. Os símbolos $\bar{r}$ e σ representam, respectivamente, a média de avaliação e desvio padrão do usuário em questão.

$$\omega_{a,u} = \frac{\sum_{i=1}^{m}(r_{a,i} - \bar{r_a}) * (r_{u,i} - \bar{r_u})}{\sigma_a * \sigma_u} \tag{4.1}$$

A implementação básica de *Pearson* considera apenas as avaliações que ambos têm em comum. Se houvessem poucos itens avaliados por ambos, a correlação poderia apresentar erroneamente um valor elevado. Por isso, há uma ponderação no denominador, de maneira a atenuar a similaridade para usuários com baixa intersecção de avaliações. Portanto, o somatório do numerador considera apenas itens comuns ao par, mas o denominador considera todos os itens que cada um avaliou. Por se tratar de um cálculo pesado e que cresce exponencialmente com o número de pessoas no sistema, essa etapa é feita previamente e os valores ficam armazenados em um banco de dados.

Para a escolha da vizinhança, deve-se procurar os usuários mais similares ao alvo. Essa busca é feita com base na correlação entre pares calculada na fase anterior. Busca-se n pessoas que possuem as maiores correlações com o usuário alvo, em que n é a quantidade de vizinhos desejada. Os itens candidatos são aqueles contidos nos históricos dos vizinhos, limitados a uma quantidade máxima.

Por fim, o sistema calcula a previsão (ou *score*) para cada item candidato. Ela simboliza a nota que o sistema acredita que o usuário daria para o item, se esse fosse hipoteticamente consumido. As previsões são ordenadas e as maiores irão compor a lista de recomendação. A Fórmula 4.2 é usada para calcular a previsão, na qual $v \in V$ denota os vizinhos de u (21):

$$s_{u,i} = \bar{r}_u + \frac{\sum_{v \in V}(r_{v,i} - \bar{r_v}) * \omega_{u,v}}{\sum_{v \in V} |\omega_{u,v}|} \tag{4.2}$$

4.4.2 Implementação da Filtragem Colaborativa Item-Item

De forma semelhante a Seção 4.4.1, é calculada previamente a correlação de *Pearson* entre todos os pares de itens do sistema. Com base nela, a vizinhança é encontrada para cada produto presente no histórico do usuário alvo. Assim, as previsões são calculadas conforme a Fórmula 4.3 e a lista de recomendação é produzida, na qual $v \in V$ representa os produtos vizinhos ao item i.

$$s_{u,i} = \bar{r}_i + \frac{\sum_{v \in V}(r_{u,v} - \bar{r_v}) * \omega_{i,v}}{\sum_{v \in V} |\omega_{i,v}|} \tag{4.3}$$

4.4.3 Implementação da Filtragem Baseada em Conteúdo

Esse método também possui uma etapa *offline*, que é a computação dos perfis vetoriais de usuários e itens. Para isso, o conjunto de dados precisa primeiramente fornecer a informação de quais *tags* foram aplicadas em cada filme. O algoritmo então realiza o *stemming*, agrupando todos os termos através da escolha de representantes. Considere o exemplo simplificado do Quadro 2, que contém três filmes e suas respectivas *tags*:

Quadro 2 – Exemplo de *tags* aplicadas aos filmes

Filmes	**Tags**
O Mágico Oz	bruxa; bruxaria; fantasia
Harry Potter e o Prisioneiro de Azkaban	bruxo; magia
O Ilusionista	mágico; mistério

Fonte: Elaborado pelo autor

Para realizar o agrupamento, é calculado o grau de semelhança entre as palavras, sendo associado um valor numérico para essa comparação. Pares de palavras cujo valor está acima de um limiar são entendidas como semelhantes. Por exemplo, a concordância entre *bruxo* e *bruxa* resulta em 0.75. Esse cálculo é baseado no trabalho de White (40), e segue as premissas apresentadas pelo autor:

Reflexão verdadeira da similaridade léxica: palavras com pequenas diferenças devem ser reconhecidas como similares.

Robustez ao mudar a ordem da palavra: dois termos que contenham as mesmas palavras em ordens diferentes devem ser entendidos como similares. No entanto, se um termo é apenas um anagrama de letras contidas no segundo, eles devem ser dissimilares.

Independência de linguagem: o algoritmo deve funcionar em diferentes idiomas.

Suponha que o algoritmo elegeu *bruxa*, *fantasia*, *magia* e *mistério* como representantes, sendo que *bruxa* simboliza as palavras *bruxa*, *bruxaria* e *bruxo*. Da mesma forma, *magia* e *mágico* são representados por *magia*. O próximo passo será calcular a relevância de cada representante pela Fórmula (41):

$$TF * IDF(t) = \frac{f(t_d)}{card(d)} * log\frac{card(D)}{df(t)} \tag{4.4}$$

Os termos $f(t_d)$ e $card(d)$ representam, respectivamente, a frequência com que o termo t aparece no documento atual e o número de elementos do mesmo documento[3]. Já $card(D)$ representa a quantidade total de documentos e $df(t)$ simboliza a quantidade de documentos cujo termo t aparece. Para o exemplo apresentado, os valores calculados e normalizados são mostrados na Tabela 5.

Tabela 5 – Representação vetorial dos filmes através das *tags*

Filmes	bruxa	fantasia	magia	mistério
O Mágico Oz	0.491	0.668	—	—
Harry Potter e o Prisioneiro de Azkaban	0.37	—	0.088	—
O Ilusionista	0.37	—	—	1

Fonte: Elaborado pelo autor

Assim criou-se um perfil vetorial para todos os itens, semelhante ao da Figura 5 só que dessa vez com 4 dimensões, sendo que cada um dos termos representados denotam uma dimensão. Os vetores dos usuários são produzidos pela soma dos perfis vetoriais de cada item presente no seu histórico, seguido de normalização. Finalmente, as predições são geradas pelo cálculo do cosseno do ângulo entre os vetores de usuário e item. Esses valores são então colocados na escala definida e ordenados para a geração da lista de recomendação. Dado dois vetores A e B, o cosseno é contabilizado pela divisão do produto escalar pelo produto da distância euclidiana (41):

$$\cos(\theta) = \frac{\sum_{i=1}^{n} A_i \cdot B_i}{\sqrt[2]{\sum_{i=1}^{n} A_i^2} * \sqrt[2]{\sum_{i=1}^{n} B_i^2}} \tag{4.5}$$

[3] No exemplo apresentado, cada documento simboliza um filme.

4.4.4 Implementação Híbrida

Os híbridos são em geral mais populares que os outros métodos, porque conseguem contornar as fraquezas das abordagens individuais (12). No RSE ele é definido como um arranjo de outros algoritmos já existentes. Assim, o híbrido implementado é uma combinação das duas variações da filtragem colaborativa, em que as listas são mescladas e os itens com as maiores previsões vão para a lista de recomendação final.

4.4.5 Implementação da Recomendação Alternativa

Essa é uma segunda estratégia de recomendação, para ser usada quando a primeira não foi capaz de gerar todas as sugestões. O cálculo representa a média das avaliações dadas em um produto. Entretanto, ela é somada com a média global μ, que por sua vez é controlada pelo parâmetro α. Na prática, isso causa um amortecimento que tende para a média global quando existem poucas avaliações, reduzindo assim o peso das mesmas. Quando avaliações suficientes estão disponíveis, esse valor é praticamente irrelevante. O cálculo então é dado pela Equação 4.6 [4]:

$$NPS_i = \frac{\sum_{u \in U_i} r_{u,i} + \alpha\mu}{|U_i| + \alpha} \tag{4.6}$$

A recomendação final é uma espécie de semi-personalização, em que é feito a média entre o valor calculado e a média de avaliações dos produtos consumidos pelo usuário em questão:

$$s_{u,i} = \frac{NPS_i + \bar{r}_{hist}}{2} \tag{4.7}$$

4.5 Módulo *Evaluator*

Nessa parte é realizada a avaliação de desempenho propriamente dita. Seu diagrama de classes é mostrado na Figura 14. Através de fatores, variáveis de resposta e distribuições temporais, *Benchmarker* organiza o experimento e o executa, fazendo todos os tratamentos necessários. Mais detalhes de cada parte são apresentados a seguir.

[4] Retirado do Curso *"Recommender Systems Specialization"*, ministrado por Joseph A. Konstan e Michael D. Ekstrand. Disponível em <https://www.coursera.org/specializations/recommender-systems>. Acesso em: 25 out. 2018

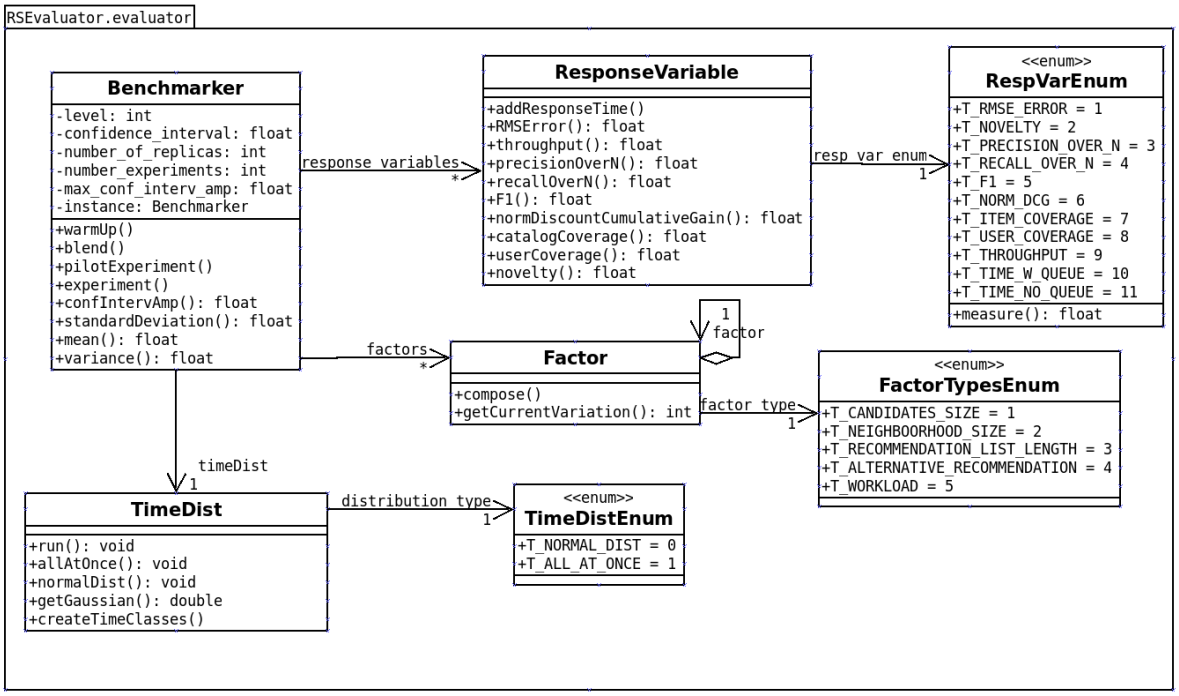


Figura 14 – Diagrama de classes do módulo *Evaluator*
Fonte: Elaborado pelo autor

4.5.1 Fatores e Níveis

O RSE conta atualmente com um conjunto de cinco fatores. São eles:

Carga de trabalho: quantidade de requisições enviadas para o sistema de recomendação. Cada usuário faz uma requisição por vez. Como a escolha das pessoas é um processo aleatório, é possível que um mesmo indivíduo seja atendido mais de uma vez em diferentes iterações. Mesmo sendo esse o caso, as recomendações serão diferentes.

Itens candidatos: são aqueles possíveis de ser recomendados. A lista de recomendação é então, um subconjunto dos itens candidatos encontrados para o usuário em questão. Esse fator limita o número de candidatos que o recomendador deve procurar.

Tamanho da vizinhança: limita a vizinhança de usuários ou itens. Geralmente é utilizado na filtragem colaborativa.

Comprimento da lista de recomendação: tamanho máximo da lista produzida.

Recomendação alternativa: diz se o sistema deve ou não procurar uma segunda estratégia de recomendação, caso a lista gerada pela primeira não esteja completamente cheia.

Há também no RSE o conceito de fator composto. Ele é criado a partir de dois ou mais fatores individuais. Isso significa que todos aqueles que integram o composto sempre

estarão no mesmo nível. Para o módulo avaliador, ele é visto como se fosse somente um. Eles são úteis quando se deseja avaliar a interação de dois fatores com os demais, mas não entre eles mesmos.

Para um melhor entendimento desse mecanismo, a Tabela 6a ilustra uma possível escolha de dois fatores simples e seus níveis, e na 6b é mostrado como eles se comportam em um experimento, em que o valor colorido indica qual nível está ativo em cada teste. Por exemplo, no experimento 4 ambos os fatores estão no segundo nível (2000; Não). Percebe-se que a quantidade de fatores e níveis é determinante para o número de experimentos, pois é preciso cobrir todas as combinações possíveis. Já as Figuras 7a e 7b exemplificam o uso de um fator composto. Nota-se que F_{1comp} varia em relação à F_2, mas seus componentes internos sempre estarão no mesmo nível.

Tabela 6 – Experimento com fatores simples

(a) Possível definição dos fatores

Fator	**Nome**	**Nível**	
		N_1	N_2
F_1	Carga de Trabalho	500	2000
F_2	Rec. Alternativa	Sim	Não

(b) Combinação dos fatores no teste

Fator	**Nível**	**Experimento**			
		1	**2**	**3**	**4**
F_1	N_1	**500**	**500**	500	500
	N_2	2000	2000	**2000**	**2000**
F_2	N_1	**Sim**	Sim	**Sim**	Sim
	N_2	Não	**Não**	Não	**Não**

Fonte: Elaborado pelo autor

Tabela 7 – Experimento com fator composto

(a) Possível definição com fator composto

Fator	**Nome**	**Nível**	
		N_1	N_2
F_{1comp}	Itens Candidatos	10	25
	Tam. Vizinhança	7	20
F_2	Comp. Lista	5	15

(b) Combinação entre fator simples e composto

Fator		**Nível**	**Experimento**			
			1	**2**	**3**	**4**
F_{1comp}	Cand.	N_1	**10**	**10**	10	10
		N_2	25	25	**25**	**25**
	Viz.	N_1	**7**	**7**	7	7
		N_2	20	20	**20**	**20**
F_2		N_1	**5**	5	**5**	5
		N_2	15	**15**	15	**15**

Fonte: Elaborado pelo autor

Mesmo que um fator não seja usado na avaliação, ele ainda pode conter um valor fixo. Assim, ele estaria se comportando como um parâmetro do sistema. Por exemplo, na situação da Figura 6a, o tamanho da lista poderia estar fixo em 10. Isto porque pode ser necessário limitar a lista produzida, mesmo que ela não esteja sendo avaliada como fator.

Todos esses parâmetros são de grande importância nos sistemas de recomendação, e devem estar bem ajustados para uma performance otimizada. Por exemplo, um conjunto de vizinhos pequeno pode gerar recomendações pouco precisas, ao passo que uma grande

quantidade acarreta em lentidão. Encontrar o melhor valor de cada parâmetro é uma das situações que a avaliação de desempenho procura determinar.

4.5.2 Variáveis de Resposta

É preciso cautela ao selecionar as métricas usadas na avaliação. Conforme demonstrado em (6), a escolha de uma medida inapropriada leva a um algoritmo inapropriado. De fato, como fazer essa escolha ainda é uma questão aberta (5). O RSE não diz quais medidas devem ser usadas. No entanto, para tornar essa tarefa um pouco mais simples, as métricas semelhantes encontram-se agrupadas em conjuntos.

Supondo-se que, em um supermercado, haja a recomendação para que os clientes comprem pães e bananas. Por se tratar de produtos bastante comuns, é possível que grande parte dos clientes comprem essas mercadorias com frequência. Esse é um caso em que o sistema de recomendação é muito preciso porém pouco útil, uma vez que as pessoas comprariam mesmo que não houvesse sugestão alguma. Percebe-se que, apesar de ser importante recomendar itens relevantes, a acurácia não deve ser o único critério (ou o mais importante), pois ela dá apenas uma visão limitada do problema em questão.

Outro cenário bastante comum é que os itens mais avaliados inevitavelmente recebem algumas notas baixas, enquanto que itens raramente vistos podem receber avaliações altas. Assim, embora o primeiro seja mais popular que o segundo, ele pode estar com uma avaliação média menor. Portanto, outros critérios devem ser levados em conta, como o lucro que certo produto pode proporcionar ou o tempo que mantém o usuário no site. Por exemplo, vídeos maiores no *Youtube* possibilitam a inserção de mais propagandas, o que faz aumentar as chances de consumo por parte do usuário. Assim, entre dois vídeos de relevâncias próximas, pode ser benéfico recomendar aquele de maior duração.

O RSE possui onze variáveis de resposta divididas em cinco grupos: acurácia, suporte à decisão, métricas centradas no usuário, ranqueamento e performance. Cada um deles visualiza o recomendador de uma perspectiva diferente. Por exemplo, um sistema precisa sugerir séries para um indivíduo chamado *Bob*. As medidas de acurácia verificam o quão preciso o recomendador é em modelar o gosto de *Bob* para um seriado, como por exemplo, *Game of Thrones*. Suporte à decisão procura descobrir se o sistema recomenda ou não séries que *Bob* iria gostar. Havendo ainda uma série desconhecida que, dados as suas atuais preferências, dificilmente lhe seria recomendada, as medidas centradas no usuário procuram saber se esta seria uma boa sugestão para ele. As métricas de ranqueamento procuram entender preferências relativas. Por exemplo, se *Bob* gosta de *Game of Thrones* mais do que *La Casa de Papel*. Por fim, performance mede o tempo gasto para que o recomendador tenha produzido uma lista de séries para *Bob*. Os detalhes de cada grupo são mostrados a seguir.

4.5.2.1 Acurácia

Aqui são medidas as imprecisões das avaliações preditas pelo sistema, através da comparação dessas com as notas dadas pelo usuário. A *Raiz do Erro Médio Quadrático* (*RMSE*) é a variável de resposta implementada nesse grupo. Seja $k = \{u, i\}$ o conjunto das avaliações de tamanho n e para as quais se deseja calcular o erro, $s_{u,i}$ a nota prevista pelo sistema, e $r_{u,i}$ a avaliação dada pelo usuário u no item i, o RMSE é definido pela Equação 4.8 (6):

$$RMSE = \sqrt{\frac{\sum_{(u,i)\in K}(s_{u,i} - r_{u,i})^2}{n}} \tag{4.8}$$

O fato da diferença ser elevada ao quadrado faz com que erros maiores sejam mais penalizados que os menores. Por exemplo, se um erro de meia estrela for computado quatro vezes, ainda assim ele seria menor que um único desvio de uma estrela e meia. Por fim, a raiz quadrada é efetuada para fazer com que encaixe dentro da escala fornecida, tornando a interpretação mais intuitiva.

4.5.2.2 Suporte à Decisão

Avalia o quão eficiente o recomendador é em ajudar o usuário a tomar boas decisões. Os representantes desse grupo são variações de medidas comumente utilizadas na área da recuperação de informações, chamados *Precisão*, *Revocação*, e *F1*. Aplica-se portanto, esses conceitos sobre a lista de itens recomendados.

A precisão representa a porcentagem de produtos da lista que são relevantes e foram escolhidos pelo usuário. Trata-se de retornar a maior quantidade de itens interessantes possível. Já a revocação mede, entre todos os itens escolhidos e relevantes daquela pessoa, quais foram provenientes da recomendação dada pelo sistema. Ela está relacionada a não perder produtos importantes. Caso não seja possível gerar a lista ou o usuário não possua avaliação para teste, os valores de precisão e revocação não serão contabilizados. Por último tem-se F1 (42), que faz um tipo de balanceamento entre as duas. O grande desafio em usá-las nos sistemas de recomendação é encontrar uma boa definição de relevância, uma vez que ela é subjetiva.

Nas fórmulas de precisão e revocação (13), o sufixo "@n" indica que os valores são medidos sob a lista de recomendação $Rec(u)$ de comprimento L (também conhecida como *top-N*). $TS(u)$ representa o histórico ou conjunto de testes de cada usuário, e a intersecção entre essas grandezas mostra que foi recomendado um item para a pessoa, ela aceitou a sugestão e o avaliou de forma positiva.

$$P@n = \frac{\sum_{i=1}^{L} |TS(u)_i \cap Rec(u)_i|}{\sum_{i=1}^{L} |Rec(u)_i|} \tag{4.9}$$

$$R@n = \frac{\sum_{i=1}^{L} |TS(u)_i \cap Rec(u)_i|}{\sum_{i=1}^{L} |TS(u)_i|} \tag{4.10}$$

$$F1 = \frac{2 * P@n * R@n}{P@n + R@n} \tag{4.11}$$

Uma das utilidades dessas métricas é, por exemplo, encontrar o comprimento adequado da lista. Listas mais longas irão melhorar a revocação, e causar uma provável diminuição na precisão. Caso o usuário não necessite de uma lista completa contendo todos os produtos relevantes, a precisão seria a medida mais apropriada. A revocação, por outro lado, seria mais adequada se o objetivo fosse buscar todos os itens importantes dentro de uma região (43).

4.5.2.3 Métricas centradas no usuário

Estão fortemente relacionadas à experiência pessoal ou a até mesmo a objetivos financeiros. Algumas vezes o recomendador não consegue gerar lista para todos os usuários, e também alguns produtos podem nunca ser recomendados. Isso acontece por exemplo, quando a pessoa é nova no sistema e pouca coisa se sabe sobre ela, ou quando o item não é tão popular, entre outras possibilidades. Por isso, o RSE tem implementado a *Cobertura de Catálogo*, a *Cobertura de Usuários* e a *Inovação*. A primeira cobertura mede, dentre o total de recomendações que o sistema deveria produzir ao final do experimento, qual a quantidade de itens que foram de fato sugeridos. A cobertura de usuários mensura a porcentagem de usuários atendidos. Na fórmula, E simboliza o conjunto de usuários que participaram do experimento, e T_L representa o tamanho padrão da lista (44):

$$Cobertura_{catálogo} = \frac{\sum_{u \in E} card(Rec(u))}{T_L * card(E)} * 100 \tag{4.12}$$

$$Cobertura_{usuário} = \frac{card(E_{at})}{card(E)} * 100 \tag{4.13}$$

Outro ponto desses sistemas é que pode acontecer um fenômeno de realimentação, ou seja, itens populares no passado tendem a ser servidos ainda mais no futuro, causando um estreitamento dos gostos e opiniões. É preciso então considerar ferramentas que favoreçam a diversidade, e estudos mostram que isso é possível de ser feito sem ter que sacrificar a performance (45). Para ajudar nessa questão tem-se a *Inovação*, que pode ser entendida como o inverso da popularidade, ou seja, o quão desconhecidos os produtos são para o usuário. No entanto, é preciso que haja um balanceamento com a acurácia, pois mesmo sendo originais as recomendações perdem o sentido se forem irrelevantes (25). Há também grande chance de que o usuário já conheça um item popular, fazendo com que sua recomendação seja redundante. Então, sugerir um produto desconhecido, com o devido

cuidado para que seja também relevante, pode entregar mais valor para o indivíduo (6). Assumindo que a inovação representa o inverso da popularidade, tem-se:

$$Inovação = 1 - \frac{Popularidade_{item}}{card(E_{at})} * 100 \tag{4.14}$$

4.5.2.4 Ranqueamento

São métricas que procuram verificar se a ordem dos itens da lista está de acordo com as preferências do usuário, ou medir a utilidade do ranqueamento feito para a pessoa. Em outras palavras, elas medem a capacidade do algoritmo de modelar preferências relativas. Para esta tarefa foi implementado o *Desconto Normalizado do Ganho Cumulativo* (nDCG)[5]. Nele, as avaliações feitas pelo usuário são colocadas em ordem decrescente e comparadas com o ranqueamento da lista de recomendação. A utilidade de um item é descontada pela posição da lista em que ele se encontra, para dar mais relevância aos que estão no topo. Por causa da variação do comportamento das pessoas, ele é normalizado pelo desconto do ganho cumulativo perfeito, ou seja, o valor atribuído a uma lista que esteja em total acordo com as preferências relativas do usuário.

$$DCG(0,u) = \sum_i \frac{r_{u,i}}{disc(i)} \qquad disc(i) = \begin{cases} 1, & \text{i} \leq 2 \\ \log_2 i, & \text{i} > 2 \end{cases} \tag{4.15}$$

$$nDCG(0,u) = \frac{DCG(0,u)}{DCG(0_u,u)} \tag{4.16}$$

Quanto menor a lista, menos a ordem dos produtos se torna importante, porque aumentam as chances de que o usuário a percorra por completo. Dependendo também da aplicação, a pessoa irá provavelmente consumir um item apenas, ou um conjunto muito pequeno de itens. Então, pode ser esperado que ela dê atenção apenas para os produtos do topo da lista (25). Esses exemplos são capturados pela modelagem de diferentes descontos. No segundo caso, por exemplo, um decaimento mais rápido se adequaria melhor à situação.

Uma característica do nDCG implementado é que ele só considera o cálculo para usuários que possuam pelo menos 3 avaliações em comum entre o conjunto de testes e a lista de recomendação gerada, pois valores menores que esse irão apresentar inevitavelmente nDCGs altos, o que poderia levar a resultados enganosos.

[5] Retirado do curso online *"Recommender Systems Specialization"*, ministrado por Joseph A. Konstan e Michael D. Ekstrand. Disponível em <https://www.coursera.org/specializations/recommender-systems>. Acesso em: 25 out. 2018

4.5.2.5 Performance

Como o processo de recomendação muitas vezes envolve *big data*, e pela necessidade que transações *online* sejam rápidas, é importante medir a velocidade de execução da tarefa. Então, nesse conjunto se encontram as métricas que dizem respeito ao desempenho temporal do algoritmo. O tempo gasto para gerar a resposta para o usuário é medido levando em consideração que ele pode ou não ter ficado na fila de espera (*TRespFila* e *TResp*, respectivamente). Tem-se também a *Produtividade*, que verifica a quantidade máxima de recomendações feitas por unidade de tempo.

$$TResp = TRec_{final} - TRec_{inicial} \tag{4.17}$$

$$TRespFila = TResp + TEspFila \tag{4.18}$$

$$Produtividade = \frac{1}{TResp} * 1000 \tag{4.19}$$

O RSE utiliza dois modos de distribuição da carga de trabalho. Um deles faz com que todas as requisições cheguem exatamente no mesmo instante, simulando um sistema trabalhando em gargalo. Ele ajuda a medir o comportamento do sistema em sua capacidade máxima. O outro modo representa uma distribuição normal de carga. Isso vem da noção que, em um ambiente real, os usuários são atendidos em instantes distintos, de acordo com o horário de solicitação.

Em um sistema de caixas de banco, por exemplo, pode ser que poucas pessoas cheguem no início do dia e sejam imediatamente recebidas pelos caixas, havendo inclusive momentos em que eles estejam ociosos. Conforme o dia passa, o atendimento vai ficando balanceado, com o fluxo de entrada mais ou menos igual ao de saída. Acontecerá também horários de pico e formação de filas. A Figura 15 exemplifica esse tipo de distribuição. A área sombreada denota que aproximadamente 70% da carga está concentrada a um desvio padrão da média. No RSE isso significa que, se estivesse sendo simulada a distribuição de carga em 1 hora, 70% das requisições chegariam entre os minutos 20 e 40. Dois desvios padrão fariam com que fossem distribuídos 95% da carga (entre os minutos 10 e 50), e com três desvios padrão praticamente toda ela seria distribuída.

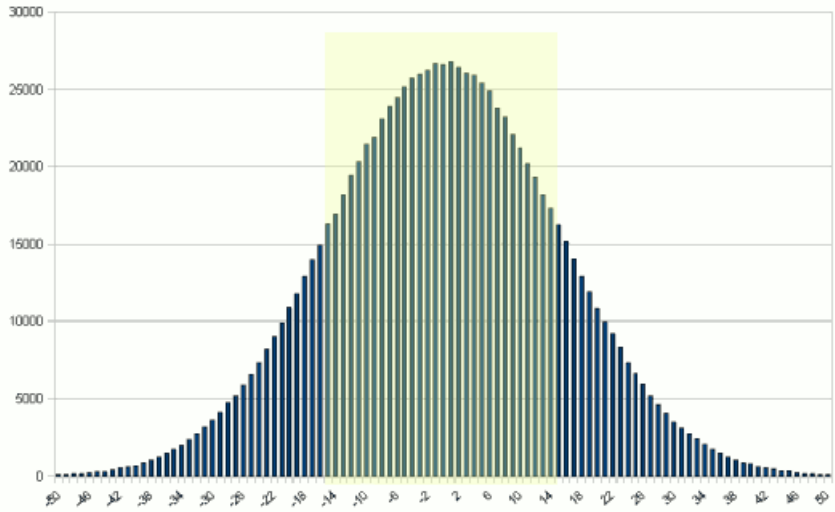


Figura 15 – Distribuição normal da carga de trabalho[6]

4.5.3 Classe *Benchmarker*

Essa é a parte principal da avaliação de desempenho, na qual ocorrem todos os experimentos. O RSE comporta dois modos de avaliação: multi-fatorial 2^k e multinível com um único fator. A forma com que as requisições são distribuídas é a mesma para ambos.

4.5.3.1 Distribuição da Carga

É preciso cuidado para garantir que a distribuição de usuários, itens e avaliações não esteja enviesada. Deve-se procurar fazer com que o teste simule a situação real da melhor maneira possível. Existe a chance de um método desempenhar bem em um conjunto de dados por este estar bastante adequado ao algoritmo. Uma maneira de reduzir a possibilidade desses contratempos é fazer a validação cruzada, na qual os dados de entrada são divididos em dois grupos (treinamento e teste). Um deles é usado na realização dos experimentos e o outro na validação dos resultados. Esse método é exemplificado na Figura 16. Os conjuntos de treinamento e teste variam a cada iteração, dando maior validade estatística aos resultados obtidos. Os dados foram divididos em 5 partes iguais, sendo que 80% foi usado para treinamento e o restante para os testes.

[6] Disponível em: <https://www.javamex.com/tutorials/random_numbers/gaussian_distribution_2.shtml>. Acesso em 19 ago. 2018.

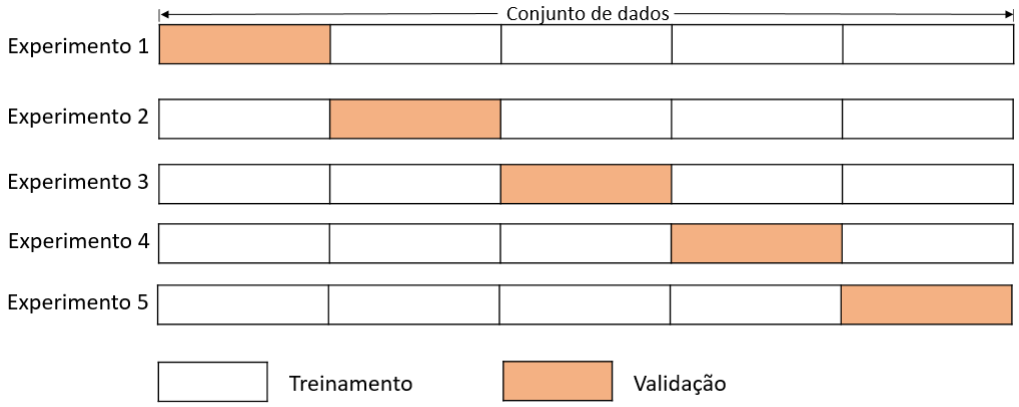


Figura 16 – Validação cruzada dividida em 5 partes
Fonte: Elaborado pelo autor

No RSE, uma ideia semelhante à validação cruzada é aplicada sobre os usuários. No início do experimento, eles são divididos aleatoriamente em cinco grupos de tamanhos iguais. A carga de trabalho é um subconjunto do grupo usado no teste, que varia a cada repetição. Suponha que o conjunto de dados é composto de 10000 usuários e a carga de trabalho é de 500 requisições. O conjunto seria fracionado aleatoriamente em 5 partes iguais de 2000. A cada experimento 500 usuários seriam pegos da parte ativa no teste (bloco cinza da Figura 17).

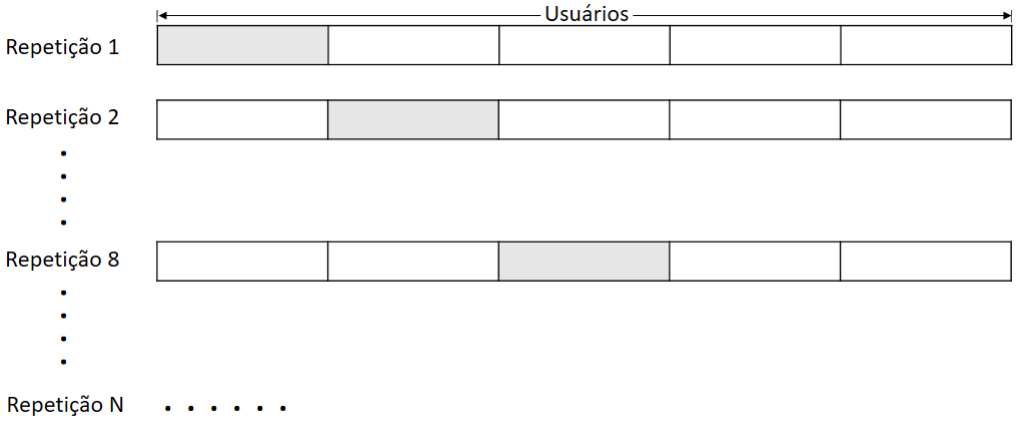


Figura 17 – Fração de usuários ativa em cada repetição
Fonte: Elaborado pelo autor

Nas aplicações tradicionais da validação cruzada, um integrante do conjunto de treinamento não participa do grupo de teste. A diferença aqui é que, uma vez que a lista de recomendação é personalizada, treinamento e validação precisam ser feitos para a mesma pessoa. Portanto, parte das avaliações do usuário são usadas no conjunto de treinamento para produzir a lista (fração conhecida como histórico), e outra parte vai para o grupo de testes. Assim, no treinamento, somente o histórico está disponível e o conjunto de teste representa a simulação dos produtos consumidos após a recomendação ser produzida. Em

outras palavras, é como se no momento do experimento, as avaliações do grupo de teste ainda não existissem. A Figura 18 ilustra esse processo.

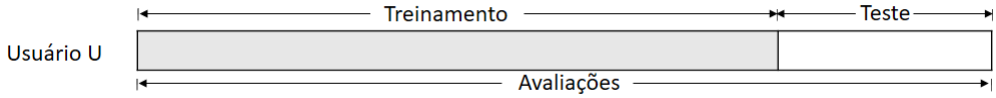


Figura 18 – Avaliações do usuário U divididas em treinamento e teste
Fonte: Elaborado pelo autor

4.5.3.2 Avaliação

É possível que somente uma avaliação de desempenho não seja suficiente para otimizar ou escolher um algoritmo. Isso porque, durante esse processo, o programador adquire novos conhecimentos a respeito da situação, que podem levá-lo a fazer outro teste com configurações diferentes (29). Assim, através de sucessivas avaliações (se for o caso), o RSE procura levar o analista a fazer a melhor escolha frente às diferentes combinações existentes. Para que as possibilidades sejam ampliadas, a ferramenta conta com dois modos de ensaio distintos:

Multi-Fatorial 2^k: se caracteriza pela possibilidade de escolha de vários fatores dentre os disponíveis. Entretanto, para cada um deles, somente dois níveis são permitidos. A justificativa dessa configuração é que ela simplifica o experimento, mas ainda assim permite tirar conclusões significativas. Se um algoritmo A fosse testado em um conjunto de dados X, outro algoritmo B experimentado no conjunto Y, e A obtivesse o melhor desempenho, não seria possível dizer se isso se deu pela superioridade do método ou por causa dos dados de entrada, ou por ambos. Assim, é importante que seja variado apenas um valor por vez (25). Esse teste consiste então na realização de 2^k ensaios, em que k é a quantidade de fatores utilizada. Cada experimento se dá pela variação de um nível, enquanto os outros permanecem fixos.

Multinível: Nesse tipo de configuração, somente um fator é permitido. Porém, vários níveis podem ser escolhidos. A ideia aqui é ter uma visão geral de quais são os melhores valores para o fator em questão. Pode ser usado para otimizar um parâmetro ou até mesmo em conjunto com a avaliação multi-fatorial.

Para que os resultados sejam estatisticamente válidos, é feito um certo número de repetições de cada teste. Os valores de resposta correspondem à média daqueles encontrados em cada repetição. Isso vale para ambos os modos. A classe possui também uma rotina para aquecimento, em que um pequeno ensaio é feito e os resultados são descartados. Seu objetivo é estabilizar a JVM. Há ainda o experimento piloto, projetado para encontrar a quantidade de repetições ideal. Alternativamente, esse número pode ser especificado pelo programador.

A Figura 19 mostra o fluxo realizado pela avaliação de desempenho no RSE. Dadas as configurações necessárias, o algoritmo passará pelas fases de aquecimento (opcional), definição das repetições e experimento. Ao finalizar, os resultados estarão disponíveis para consulta.

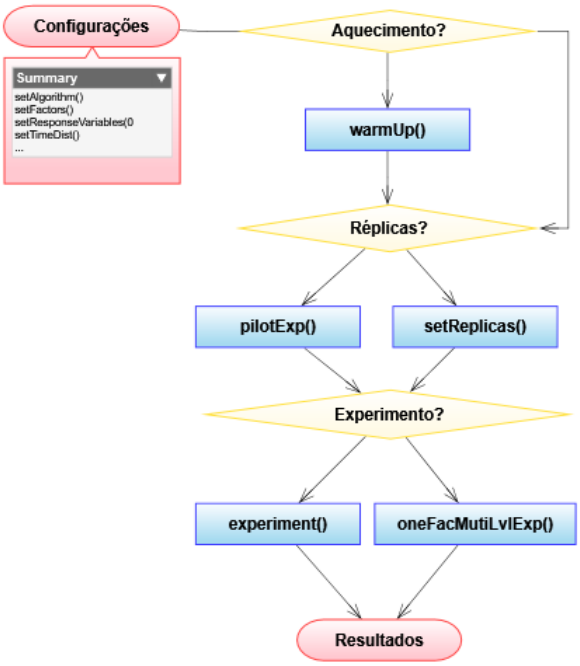


Figura 19 – Fluxo da avaliação realizada pelo RSE
Fonte: Elaborado pelo autor

4.6 Módulo *Reports*

Após ter completado a avaliação, as medições estarão salvas em um banco de dados a parte. Esse é o módulo que apresenta as informações de forma sucinta, para que conclusões possam ser tiradas. Ele consiste, basicamente, de classes para consulta dos dados, realização de cálculos e geração de gráficos. Na Figura 20 pode ser visto a estrutura responsável por armazenar os resultados da avaliação. A Figura 21 mostra o diagrama de classes de *Reports*.

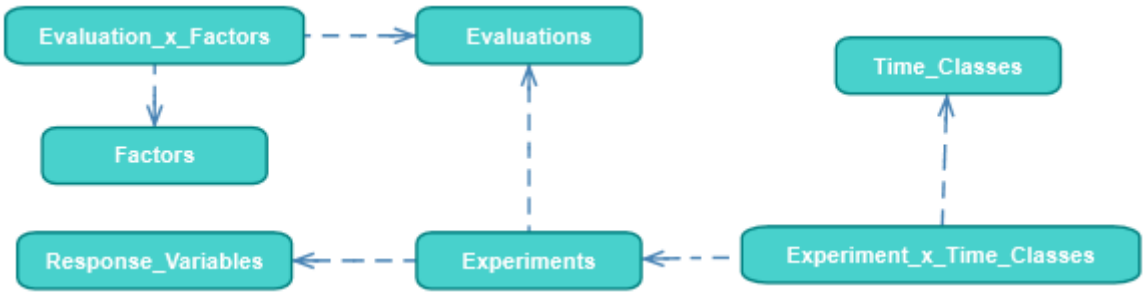


Figura 20 – Estrutura que armazena os resultados da avaliação
Fonte: Elaborado pelo autor

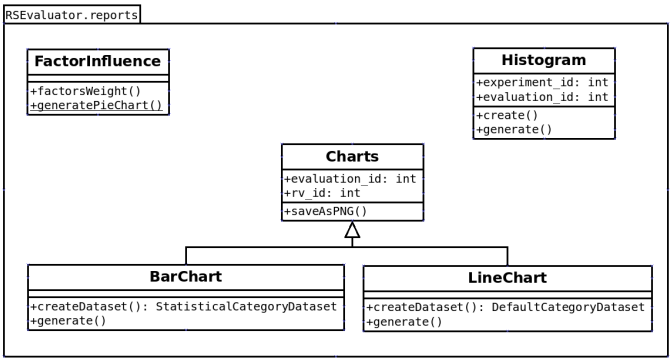
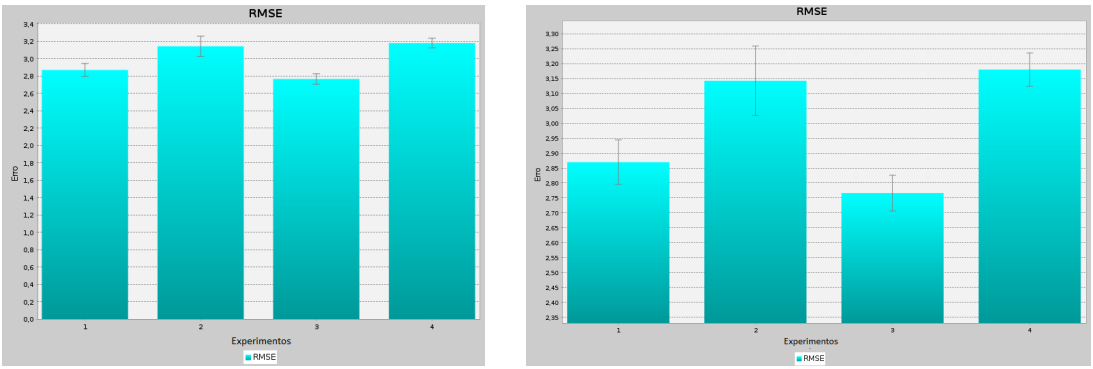


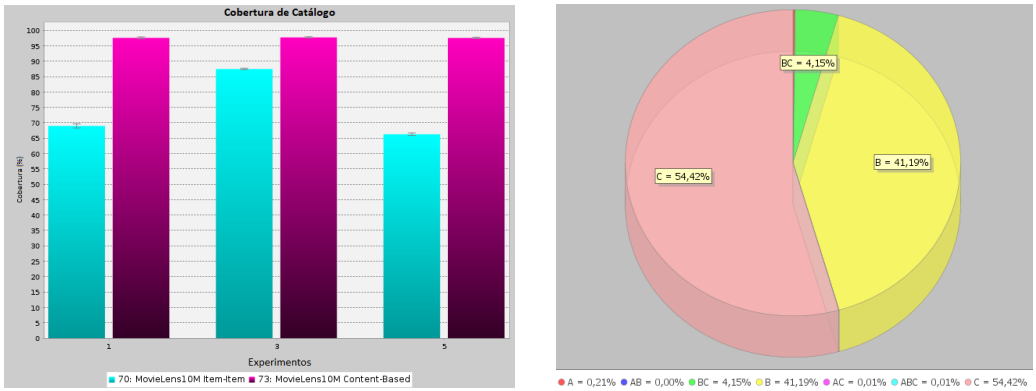
Figura 21 – Diagrama de classes simplificado do módulo *Reports*
Fonte: Elaborado pelo autor

A Figura 22a apresenta um gráfico produzido pelo *framework* para o RMSE. A ferramenta proporciona ainda a aplicação de zoom caso seja necessário. Isso é mostrado em 22b. É possível também fazer a comparação entre avaliações distintas (Figura 23a), ver a influência dos fatores (Figura 23b), histogramas da distribuição de carga (24a) e gráfico de linha para fator multinível (24b).



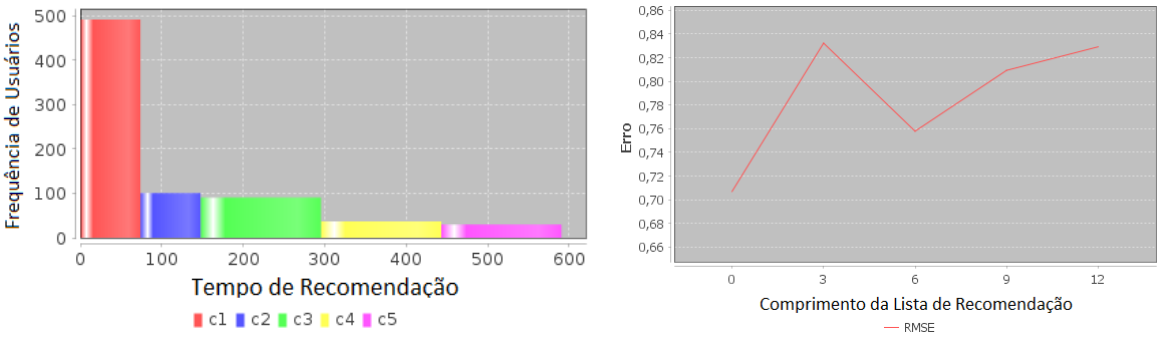
(a) Visualização normal (b) Aplicação de zoom

Figura 22 – Gráfico de barras do RMSE
Fonte: Elaborado pelo autor



(a) Gráfico comparativo (b) Influência dos fatores

Figura 23 – Outros gráficos produzidos
Fonte: Elaborado pelo autor



(a) Distribuição de carga (b) Fator multinível

Figura 24 – Outros tipos de saída
Fonte: Elaborado pelo autor

4.7 Módulo *Utils*

Nesta parte ficam funções auxiliares do sistema. Seu diagrama de classes é apresentado na Figura 25. Como o nome já diz, a classe *User* realiza tarefas relativas ao usuário, como gerar histórico e calcular média de avaliações. Em *Tags* tem-se tratamento e comparação entre palavras, e outros. Há também um gerador de *logs* e um arquivo de configuração. Através dele o analista escolhe o banco de dados que será usado, algoritmo de recomendação, variáveis de resposta, e muitos outros. Por fim, a classe *Utils* disponibiliza algumas outras tarefas diversas.

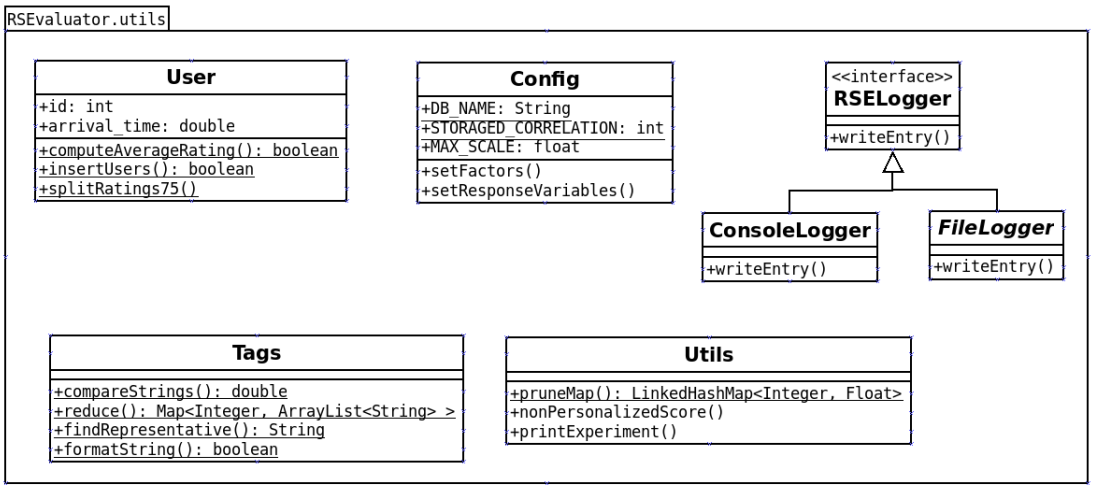


Figura 25 – Diagrama de classes simplificado do módulo *Utils*
Fonte: Elaborado pelo autor

4.8 Considerações Finais

Este Capítulo evidenciou as principais características do RSE. Uma visão geral da sua estrutura foi apresentada, explorando cada um dos seus módulos em detalhes. Percebe-

se que o encapsulamento em funcionalidades semelhantes permite maior flexibilidade ao fazer modificações ou melhorias no *framework*.

O Capítulo 5 apresenta o RSE na prática, mostrando como ele deve ser configurado e exemplificando como fazer uma avaliação de desempenho do ponto de vista do programador/avaliador.

5 RSE - Um Ponto de Vista Prático

Este capítulo visa mostrar o RSE em funcionamento através de exemplos. É seguido o fluxo de etapas necessário para realizar um ensaio, desde a preparação da estrutura, passando pela importação do conjunto de dados, avaliação e produção dos resultados.

5.1 Configurações

A classe *Config* é formada por constantes que representam as personalizações do programador. Ela possui definições de por exemplo, o nome do banco de dados dos testes, nome de alguns identificadores, escala utilizada nas avaliações, entre outras. Algumas de suas constantes são usadas já na importação dos dados, e outras no experimento em si.

O Código 5.1 é uma amostra de como as configurações são feitas. Nesse exemplo será usado o conjunto público de dados *MovieLens10M* (46), extraído de um recomendador online de filmes[1], que conta com aproximadamente 10 mil filmes, 70 mil usuários e 10 milhões de avaliações. Mais detalhes sobre ele são mostrados na Seção 6.2.

```
1  public class Config {
2
3      // nome do banco de dados dos testes
4      public static final String DB_NAME = "MovieLens10M";
5      // personalização do nome da tabela de itens
6      public static final String ITEM_TABLE = "movies";
7      // algoritmo utilizado
8      public static final String RECOMMENDER = "UserUserCF";
9      // escolhe o tipo de distribuição da carga
10     public static final TimeDistEnum DIST_TYPE = T_NORMAL_DIST;
11     // outras definições...
12 }
```

Código 5.1 – Classe que armazena as configurações do RSE
Fonte: Elaborado pelo autor

5.2 Preparando a Estrutura

Esta preparação consiste, primeiramente, em criar duas estruturas no banco de dados: uma para o conjunto de dados que será alvo do teste e outra para armazenar os resultados. Essa última servirá também para todos os ensaios posteriores, e possui por padrão o nome *Evaluation*. O Código 5.2 mostra a criação do banco teste. O comando da linha 1 instancia

[1] MovieLens <https://movielens.org/>. Acesso em 18 out. 2018.

um objeto de *Groundwork*, e faz a conexão com o *PostgreSQL*. Na linha 3 o banco é gerado, e na linha 7 suas tabelas são criadas. Isso resulta na estrutura apresentada na Figura 12.

```
1    Groundwork struct = new Groundwork("");
2
3    if(struct.createDatabase(DB_NAME)){
4
5        struct.setDb(DB_NAME);
6        struct.open(); // abre a conexão no novo banco
7        struct.createStructure(); // criação das tabelas
8
9    }
```

Código 5.2 – Criação do banco que será avaliado
Fonte: Elaborado pelo autor

De modo semelhante, é criada a base para os resultados da avaliação, que pode ser vista na Figura 20. A diferença está na substituição das funções das linhas 3 e 7 por *createEvalDatabase()* e *createEvalStructure()*, respectivamente, e também do nome do banco de dados. No caso do banco *Evaluation*, além de gerar a estrutura, já serão inseridas informações prévias em tabelas referentes aos fatores, classes de tempo e variáveis de resposta. Elas são necessárias para o funcionamento normal da ferramenta.

5.3 Inserindo Dados

O conjunto de entrada pode vir de fontes variadas, e por isso os dados estarão organizados de diferentes maneiras. É preciso colocá-los no formato esperado do RSE (usuários, itens, avaliações). Geralmente dentro de uma base, os arquivos correspondem a tabelas, cada linha a uma entrada, e as colunas são separadas por um caractere distinto. Por exemplo, no *MovieLens10M* as primeiras linhas do arquivo de filmes vêm da seguinte maneira:

- 1::Toy Story (1995)::Adventure|Animation|Children|Comedy|Fantasy
- 2::Jumanji (1995)::Adventure|Children|Fantasy
- 3::Grumpier Old Men (1995)::Comedy|Romance

Aqui, as colunas são respectivamente identificador, título e gêneros. Dois caracteres de dois pontos (::) são usados como separador. Entretanto, o *PostgreSQL* aceita como delimitador apenas um único caractere. Uma forma eficaz de substituí-lo seria usar o comando *SED* do *Linux*. Nesse arquivo, os (::) foram trocados por ponto-e-vírgula (;). O novo delimitador não pode estar presente nem uma única vez no documento. O mesmo

foi feito com o arquivo das avaliações. Uma vez que o objeto *struct* criado anteriormente esteja com a conexão aberta no banco *MovieLens10M*, o Código 5.3 para a importação do conjunto de dados pode ser executado. O último parâmetro especifica as colunas a serem carregadas. Caso o arquivo de dados e tabela destino possuam exatamente as mesmas colunas, o parâmetro pode ser deixado vazio.

```
int rows = struct.importData(
    "movies", // tabela destino
    "/home/aluno/Desktop/MovieLens10M", // caminho da pasta
    "movies.dat", // arquivo de origem
    ';', // separador
    "movie_id, title, genres" //colunas destino
);
```

Código 5.3 – Uso da função de importação de dados
Fonte: Elaborado pelo autor

A próxima etapa realiza a importação dos arquivos dos usuários. No entanto, o *MovieLens10M* não traz explicitamente essa informação. A única informação de usuários disponível são os identificadores presentes no arquivo de avaliações. Para contornar essa situação as avaliações são carregadas primeiro. Em seguida o RSE copia os identificadores de usuários a partir delas. Como trata-se de uma operação de ajuste, é preciso remover a chave estrangeira, e em seguida importar as avaliações de modo semelhante ao dos filmes:

```
// remove a chave estrangeira da tabela ratings
struct.dropConstraint("ratings", "user_id_fkey");
// importação das avaliações
int rows = struct.importData(...);
```

Código 5.4 – Remoção de chave estrangeira seguida de importação
Fonte: Elaborado pelo autor

Agora podem ser criados os usuários e recolocada a chave:

```
User.insertUsers(); // importa os ids para a tabela de usuarios

// cria uma nova chave estrangeira
struct.addForeignKey(
    "ratings", // tabela
    "user_id_fkey", // nome da chave estrangeira
    "user_id", // coluna na qual a chave se refere
    "users", // nome da tabela referenciada
    "user_id" // nome da coluna referenciada
);
```

Código 5.5 – Criação de usuários e de chave estrangeira
Fonte: Elaborado pelo autor

O resultado das operações feitas até agora está mostrado nas Figuras 26 e 27. As colunas vazias serão preenchidas na sequência.

movie_id integer	title character varying	genres character varying	global_avg_rt real	popularity integer	non_personalized_score real
1	Toy Story (1995)	Adventure|Animati...	[null]	[null]	[null]
2	Jumanji (1995)	Adventure|Children...	[null]	[null]	[null]
3	Grumpier Old ...	Comedy|Romance	[null]	[null]	[null]

Figura 26 – Dados inseridos em *movies* até o momento
Fonte: Elaborado pelo autor

user_id integer	global_avg_rt real	history_avg_rt real
1	[null]	[null]
2	[null]	[null]
3	[null]	[null]

(a) Users

user_id integer	movie_id integer	rating real	timestamp integer	is_history boolean
1	122	5	838985046	[null]
1	185	5	838983525	[null]
1	231	5	838983392	[null]

(b) Ratings

Figura 27 – Dados inseridos em *users* e *ratings* até o momento
Fonte: Elaborado pelo autor

A coluna *is_history* de *ratings* determina quais as avaliações farão parte do conjunto de treinamento e quais irão para o conjunto de teste. Ela deve ser preenchida primeiro, pois é usada como base para outros cálculos. Na tabela *users*, duas médias são contabilizadas: uma considera somente as avaliações do histórico e a outra usa todas as notas atribuídas pelo usuário em questão. Os comandos do Código 5.6 irão completar as colunas vazias.

```
1 //coloca 75% das avaliações como histórico e 25% como teste
2 User.splitRatings75();
3 User.computeAverageRating(true); // média global dos usuários
4 User.computeHistoryAverageRating(); // média do histórico de avaliações
5 Utils.itensAverageRating(); // média global dos itens
6 Utils.computePopularity(); // popularidade
7 Utils.nonPersonalizedScore(); // score não-personalizado
```

Código 5.6 – Funções que realizam cálculos diversos
Fonte: Elaborado pelo autor

Após a execução do código, todos os valores nulos foram preenchidos, como pode ser visto nas Figuras 28 e 29.

movie_id integer	title character varying	genres character varying	global_avg_rt real	popularity integer	non_personalized_score real
1	Toy Story (1995)	Adventure|Animati...	3.92877	53059	3.92869
2	Jumanji (1995)	Adventure|Childre...	3.20807	22466	3.2082
3	Grumpier Old ...	Comedy|Romance	3.15039	15111	3.15062

Figura 28 – Amostra de *movies* com todos os valores preenchidos
Fonte: Elaborado pelo autor

user_id integer	global_avg_rt real	history_avg_rt real
1	5	5
2	3.2	3.26667
3	3.93939	3.94

(a) Users

user_id integer	movie_id integer	rating real	timestamp integer	is_history boolean
1	122	5	838985046	false
1	185	5	838983525	true
1	231	5	838983392	true

(b) Ratings

Figura 29 – Amostra de *users* e *ratings* com todos os valores preenchidos
Fonte: Elaborado pelo autor

5.4 Computando Similaridades

Para computar as similaridades, são feitos os cálculos da correlação de Pearson para itens e usuários. Como esta correlação cresce exponencialmente com a quantidade de pessoas do sistema, é interessante limitar a quantia armazenada. Isso é feito através de uma constante presente em *Config*. No exemplo dado, as 2000 correlações mais altas de cada usuário serão armazenadas.

```
1 public static final int STORAGED_CORRELATION = 2000;
```

Código 5.7 – Configuração de quantidade de correlações armazenada
Fonte: Elaborado pelo autor

Mesmo com esse limite, o algoritmo inevitavelmente terá de computar a correlação para todos os pares. Com aproximadamente 70 mil usuários, serão calculados quase 5 bilhões de correlações. No entanto por causa da especificação em *STORAGED_CORRELATION*, apenas 17.5 milhões serão salvas em banco, representando menos de 0.5% do total. Além disso, o algoritmo armazenará somente valores positivos. As funções para as correlações de usuários e itens estão no Código 5.8. Note, entretanto, que elas devem ser executadas separadamente. Isso porque são operações demoradas. Nesse caso, a similaridade entre os usuários demorou aproximadamente 7 dias e 15 horas em um computador comum. O cálculo para os itens demorou por volta de 4 horas e 10 minutos. Uma última observação é que foi preciso alterar a quantidade de memória RAM na JVM, permitindo que fosse utilizado até 3GB.

```
1 CollaborativeFiltering.UserUserPearsonSimilarity();
2 CollaborativeFiltering.ItemItemPearsonSimilarity();
```

Código 5.8 – Computação das similaridades
Fonte: Elaborado pelo autor

O resultado desses comandos está amostrado na Figura 30. A inserção de *tags* e a criação de perfis vetoriais seguem etapas bastante parecidas com o que foi apresentado até aqui.

user_x integer	user_y integer	similarity real
2	50	0.182333
2	71	0.241926
2	84	0.285456

(a) user_similarity

item_x integer	item_y integer	similarity real
1	4	0.287661
1	19	0.154006
1	26	0.188234

(b) item_similarity

Figura 30 – Amostra das tabelas *user_similarity* e *item_similarity*
Fonte: Elaborado pelo autor

5.5 Adicionando um novo algoritmo

Esta Seção apresenta os passos necessários para adicionar um novo algoritmo de recomendação. Primeiramente, é preciso criar a classe estendendo *Recommender*. Sua função principal deverá ser implementada dentro de *score()*.

```
1  public class MyRecommender extends Recommender{
2
3      public MyRecommender(int candidates, int neighbors,
4                           User t_user, int rec_list_length){
5          super(candidates, neighbors, t_user, rec_list_length);
6      }
7      @Override
8      public LinkedHashMap<Integer, Float> score() throws SQLException{
9         // implementação da lógica do algoritmo..
10     }
11     @Override
12     public void alternativeScore
13             (int items_qtd,
14              LinkedHashMap<Integer, Float> recommendation_list
15              )throws SQLException {}
16 }
```

Código 5.9 – Criação de um novo algoritmo
Fonte: Elaborado pelo autor

É necessário também a criação de uma fábrica, que será chamada em tempo de execução e ficará responsável por instanciar *MyRecommender*.

```
1 public class MyRecommenderFactory extends RecommenderFactory{
2     @Override
3     public Recommender makeRecommender(int candidates, int neighborhood,
          User t_user, int rec_list_length){
4         return new MyRecommender(candidates, neighborhood,
5                                  t_user, rec_list_length);
6     }
7 }
```

Código 5.10 – Criação da fábrica para o novo algoritmo
Fonte: Elaborado pelo autor

O novo algoritmo está pronto para ser utilizado, bastando apenas que seja escolhido no arquivo de configuração quando necessário.

```
1 public static final String RECOMMENDER = "MyRecommender";
```

Código 5.11 – Escolha do algoritmo no arquivo de configuração
Fonte: Elaborado pelo autor

5.6 Preparando a Avaliação

Os fatores e variáveis de resposta possuem um tipo enumerado associado. A classe *Config* permite dizer quais deles serão usados no experimento, bem como especificar os valores desejados. Um exemplo é mostrado no Código 5.12.

```
1 public static void setResponseVariables(){
2
3     T_RMSE_ERROR.awake = true; // ativa a variável RMSE
4     T_NORM_DCG.awake = true; // ativa o nDCG
5 }
```

Código 5.12 – Escolha das variáveis de resposta
Fonte: Elaborado pelo autor

O Código 5.13 configura os fatores. A função *fillfactor()* recebe nos dois primeiros parâmetros os valores dos níveis 1 e 2, respectivamente. O último parâmetro identifica o fator escolhido. Em seguida ele é adicionado ao experimento por *addFactor()*. Na criação de um fator composto, eles devem ser acoplados e então adicionados ao experimento. Isso é feito pela função *compose()*.

```
1  public static void setFactors(Benchmarker bcmk){
2
3      // criacao do fator simples
4      Factor workload = fillFactor("250","380",T_WORKLOAD);
5      bcmk.addFactor(workload);
6
7      // criacao do fator composto
8      Factor neighbors = fillFactor("5","25",T_NEIGHBOORHOOD_SIZE);
9      Factor fcomp = fillFactor("30","60",T_CANDIDATES_SIZE);
10     fcomp.compose(neighbors);
11     bcmk.addFactor(fcomp);
12 }
```

Código 5.13 – Configuração dos fatores de entrada
Fonte: Elaborado pelo autor

No caso do experimento multinível, a criação do fator ficaria como no Código 5.14. Da forma como foi especificado na linha 7, o tamanho da vizinhança vai variar a cada 8 unidades entre 10 e 40, resultando em 5 níveis. Há ainda um exemplo de configuração de valor padrão na linha 10.

```
1  public static void setFactors(Benchmarker bcmk){
2
3      Factor neighbors =
4              Factor.createMultiLevelFactor(T_NEIGHBOORHOOD_SIZE);
5      bcmk.addFactor(neighbors);
6
7      Factor.setMultiLevel(10, 40, 8);
8
9      // valor padrao
10     T_RECOMMENDATION_LIST_LENGTH.setDefault("10");
11 }
```

Código 5.14 – Configuração do fator multinível
Fonte: Elaborado pelo autor

5.7 Avaliação de Desempenho

Primeiramente, cria-se uma nova avaliação com o auxílio da classe *Evaluation*. O comando para o aquecimento é mostrado na linha 7 do Código 5.15. Já na linha 13, coloca-se o número de repetições desejado. Alternativamente isso pode ser substituído por *pilotExperiment()* da classe *Benchmarker*. A chamada para a avaliação é feita na linha 14. No caso do experimento multinível ela seria substituída pelo comando comentado na linha 15. Em ambos os casos o valor booleano *true* indica que os resultados deverão ser salvos.

```
Evaluation eval = Evaluation.getInstance();
int id = eval.saveEvaluation("Teste Av.", CONFIDENCE_INTERV);

Benchmarker bcmk = Benchmarker.getInstance();
bcmk.settings(CONFIDENCE_INTERV, eval_id);

bcmk.warmUp(); //aquecimento

Config.setFactors(bcmk); // configura fatores
bcmk.persistFactors();
Config.setResponseVariables(); // configura variáveis de resposta

bcmk.setNumberOfReplicas(10);
bcmk.experiment(true); // experimento multi-fatorial
// bcmk.oneFacMultiLvlExp(true); ensaio multinível
```

Código 5.15 – Avaliação de desempenho
Fonte: Elaborado pelo autor

5.8 Plotando Resultados

Ao realizar a avaliação, o programador dispõe de ferramentas gráficas para visualização dos resultados. O Código 5.16 mostra a criação do gráfico de pizza da influência dos fatores. Os outros gráficos seguem um raciocínio parecido, utilizando a função *generate()* das classes *BarChart*, *LineChart*, ou *Histogram*.

```
FactorInfluence.generatePieChart(
   path, // caminho onde será salvo
   rv_id, // identificador da variável de resposta
   title // título do gráfico
);
```

Código 5.16 – Produção do gráfico da influência dos fatores
Fonte: Elaborado pelo autor

O Capítulo 6 apresenta estudos comparativos dos métodos implementados, no qual os resultados proporcionados pelo RSE poderão ser vistos e interpretados.

6 Validando o RSE

Foram feitos alguns experimentos com os algoritmos implementados com o objetivo de validação do RSE, e também de mostrar formas de tirar conclusões a partir de seus resultados.

6.1 Detalhes do Ambiente

Para executar os ensaios utilizou-se de um computador com Ubuntu 16.04, processador Intel Core2 Quad CPU Q9550 x64, 2.833Ghz e 4GB de RAM. Nenhuma outra tarefa estava sendo executada no momento dos testes. Para que os dados fossem coletados somente quando a JVM estivesse estável, um aquecimento foi realizado através de um experimento inicial com carga reduzida.

6.2 Banco de Dados

Dois bancos de dados foram utilizados para os testes, um de filmes e outro de músicas. Eles serão brevemente descritos.

Segundo Park et al. (47), o domínio dos filmes é o que mais aparece nas pesquisas dos sistemas de recomendação. O banco *MovieLens10M* (46) foi escolhido, por se tratar de um bom exemplo que possui as principais características esperadas de um sistema de recomendação. Tratam-se de informações coletadas de um recomendador real criado por *GroupLens* [1], um grupo de pesquisa da *Universidade de Minesota*. O banco possui 10.000.054 avaliações, 95.580 tags, 10.681 filmes e 71.567 usuários cadastrados. Suas avaliações são feitas na escala de 1 a 5 estrelas, sendo que cada usuário deu nota em pelo menos 20 filmes. Além disso, ele foi incrementado com o *MovieLens Tag Genome* (48) que pertence ao mesmo grupo, e possui 1.128 *tags* distintas totalizando aproximadamente 11 milhões de escores de relevância aplicados aos filmes.

Ainda o trabalho de Park et al. (47) aponta que é preciso mais pesquisa no campo da música. Esse foi um dos motivos que levou a ser escolhido o *R2 - Yahoo!* [2] e também a necessidade de testar a flexibilidade da plataforma na mudança de contexto. É um banco de dados pertencente ao programa *Yahoo Webscope*. Ele possui avaliações de usuários em músicas, bem como informações do artista, gênero e álbum. Foi usada apenas uma pequena parte do banco composta por 36.000 usuários, totalizando 14.157.926 avaliações

[1] <https://grouplens.org>. Acesso em: 18 out. 2018

[2] R2 - Yahoo! Music User Ratings of Songs with Artist, Album, and Genre Meta Information, v. 1.0, <https://webscope.sandbox.yahoo.com>. Acesso em: 18 out. 2018.

em 136.736 artistas. Separou-se 10 avaliações para compor o conjunto de testes de cada pessoa, e o restante ficou sendo parte do seu histórico. Esse conjunto também tinha escala de 1 a 5, porém ela foi dobrada propositalmente para ver se a plataforma lidava bem com essa mudança. Além disso, os nomes das tabelas e colunas variam entre os dois bancos, como por exemplo, a tabela de itens que se chama *movies* no primeiro e *songs* no segundo.

6.3 Experimentos Realizados

Foram elaborados 5 cenários de experimentação para demonstrar a aplicabilidade do RSE, apresentados entre as Seções 6.3.1 e 6.3.5. O experimento da Seção 6.3.1 avalia a performance da filtragem colaborativa item-item. A filtragem baseada em conteúdo é avaliada na Seção 6.3.2. As duas abordagens são comparadas, apresentando seus prós e contras na Seção 6.3.3. Na Seção 6.3.4 um ajuste de parâmetro é feito, e na Seção 6.3.5 tem-se um teste da filtragem colaborativa usuário-usuário, no qual são discutidos assuntos relacionados ao tempo de espera.

6.3.1 Ensaio 01 - MovieLens10M Item-Item

6.3.1.1 Setup

Esse ensaio mediu a performance da *filtragem colaborativa item-item*. Os valores escolhidos para o intervalo de confiança e sua amplitude máxima foram de 90% e 14%, respectivamente. Isso significa que o *software* teve que calcular uma quantidade de repetições que, com 90% de confiança (ou erro máximo de 10%), fizesse com que as medições não ultrapassassem a largura de 14% em torno da sua média. Esse cálculo resultou em 13 repetições. A especificação *relevância* permite que itens façam parte da lista de recomendação somente se estiverem iguais ou acima do valor estabelecido. Mais detalhes das configurações estão no Quadro 3.

Quadro 3 – Configurações do ensaio 01

Variáveis de Resposta	Erro RMSE
	Produtividade
	Cobertura de Catálogo
	Cobertura de Usuários
Fatores (1; -1)	A. Tamanho da Vizinhança (5; 25)
	B. Itens Candidatos (30; 100)
	C. Comprimento da Lista (5; 25)
Algoritmo	Filtragem Colaborativa Item-Item
Banco de Dados	MovieLens10M
Intervalo de Confiança	90%
Amplitude Máxima do IC	14%
Repetições	13
Carga de Trabalho	2500 usuários por repetição
Relevância	3.5
Recomendação Aternativa	Não

Fonte: Elaborado pelo autor

6.3.1.2 Resultados

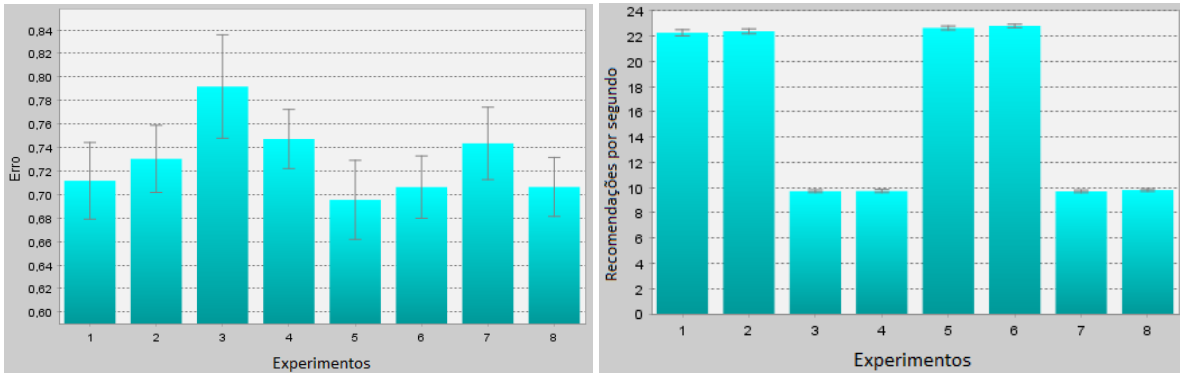
Os resultados medidos estão apresentados na Tabela 8. Foram necessários 8 testes para cobrir todas as possíveis combinações.

Tabela 8 – Resultados experimentais do ensaio 01

	Fatores			Variáveis de Resposta							
Exp.	**A**	**B**	**C**	**RMSE**	Amp.IC %	**Prod.**	Amp.IC %	**Cb.Cat.**	Amp.IC %	**Cb.Us.**	Amp.IC %
1	1	1	1	0.712	9.165	22.271	2.177	74.241	0.944	87.385	0.573
2	1	1	-1	0.730	7.818	22.390	1.712	36.031	1.300	87.462	0.991
3	1	-1	1	0.792	11.110	9.714	2.029	89.329	0.682	94.154	0.582
4	1	-1	-1	0.747	6.733	9.730	2.355	68.433	1.423	94.385	0.530
5	-1	1	1	0.695	9.669	22.654	1.354	73.594	1.031	86.462	1.003
6	-1	1	-1	0.706	7.506	22.814	1.185	35.851	1.447	87.077	0.560
7	-1	-1	1	0.743	8.280	9.705	1.971	88.586	0.695	93.539	0.548
8	-1	-1	-1	0.706	7.103	9.805	1.954	67.902	1.331	93.615	0.535

Fonte: Elaborado pelo autor

Na Figura 31a as medidas do erro RMSE são apresentadas em forma de gráfico. Além disso, foi feito um zoom na escala para facilitar a comparação. Percebe-se que houve pouca variação na acurácia, e o experimento número 3 foi o que apresentou maior erro em valor absoluto. Em relação a ele, os testes 1, 5, 6 e 8 se mostraram ligeiramente melhores. Em termos de produtividade, as respostas mais rápidas estão nos testes 1, 2, 5, e 6. Nesses experimentos chegou-se a pouco mais de 22 recomendações por segundo. A Figura 31b mostra esse caso. A cobertura de catálogo se mostrou bastante satisfatória nos experimentos 3 e 7, chegando a valores próximos de 90% (Tabela 8). Em relação à cobertura de usuários, os testes 3, 4, 7 e 8 obtiveram valores entre 93 e 94.5%.



(a) Resultados com zoom para o RMSE (b) Resultados obtidos para a produtividade

Figura 31 – Resultados gráficos para o RMSE e produtividade

Fonte: Elaborado pelo autor

Uma segunda interpretação no gráfico da produtividade é o fato de que ela foi influenciada principalmente pela quantidade de itens candidatos (fator B), pois as principais variações se deram justamente nas mudanças dos seus níveis. Os testes 1, 2, 5 e 6 foram feitos com 30 candidatos, e os demais com 100. De fato, isso é confirmado pela Tabela 9, que mostra a influência de 99.94% do fator B na produtividade, e apresenta também o cálculo para as outras variáveis de resposta. Alternativamente, eles podem ser mostrados na forma gráfica, como é exemplificado na Figura 32.

Tabela 9 – Influência dos fatores

Fatores	**Variáveis de Resposta**			
	RMSE	**Prod.**	**Cb.Cat.**	**Cb.Us.**
A	30.12	0.03	0.02	0.96
B	38.02	99.94	37.33	98.80
C	4.90	0.01	57.71	0.13
AB	4.31	0.02	0.00	0.00
AC	0.00	0.00	0.00	0.02
BC	22.23	0.00	4.94	0.02
ABC	0.43	0.00	0.00	0.06

Fonte: Elaborado pelo autor

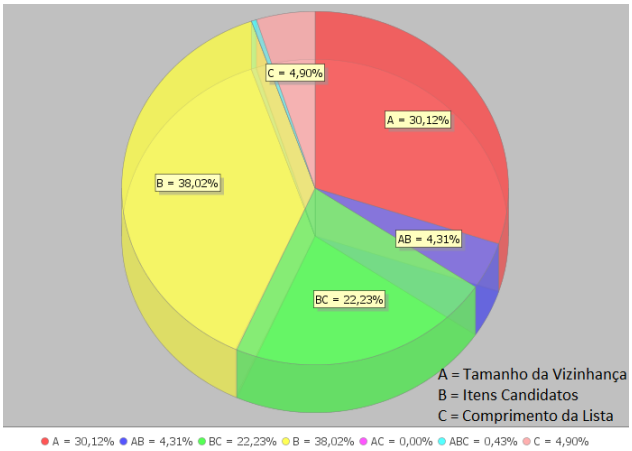


Figura 32 – Influência dos fatores para o RMSE
Fonte: Elaborado pelo autor

Na maioria das vezes não vai haver uma opção que maximize todas as variáveis de resposta. Será preciso então considerar os custos-benefícios envolvidos, e isso depende bastante dos objetivos de uma aplicação real. O Quadro 4 resume as melhores combinações encontradas para cada métrica de resposta.

Quadro 4 – Melhores testes por variável de resposta

Variável de Resposta	**Experimentos**
RMSE	1, 5, 6, 8
Produtividade	1, 2, 5, 6
Cobertura de Catálogo	3, 7
Cobertura de Usuários	3, 4, 7, 8

Fonte: Elaborado pelo autor

A combinação dada pelo experimento 1 apresenta a melhor performance e acurácia. Ela poderia ser implementada caso as coberturas de catálogo e usuários de 74.2% e 87.3% fossem aceitáveis. Se a aplicação demandar que se atinja maiores coberturas, a alternativa 7 se torna mais atrativa. No entanto, isso causaria uma perda de mais de 50% no tempo de resposta e um erro um pouco maior. Escolheu-se portanto as combinações dadas por esses dois testes como sendo os valores possíveis de serem implementados. Elas estão mostradas na Tabela 10.

Tabela 10 – Detalhes dos testes escolhidos no ensaio 01

Fator	**Exp. 1**	**Exp. 7**
Tamanho da vizinhança	5	25
Itens candidatos	30	100
Comprimento da lista	5	5

Fonte: Elaborado pelo autor

6.3.2 Ensaio 02 - MovieLens10M - Filtragem Baseada em Conteúdo

Para fins comparativos, o segundo teste consistiu em fazer um ensaio com as mesmas configurações do primeiro, exceto que foram necessárias 11 repetições, e o algoritmo utilizado foi a *filtragem baseada em conteúdo*. Os resultados podem ser vistos na Tabela 11.

Tabela 11 – Resultados experimentais para a filtragem baseada em conteúdo

Fatores				**Variáveis de Resposta**							
Exp.	**A**	**B**	**C**	**RMSE**	Amp.IC %	**Prod.**	Amp.IC %	**Cb.Cat.**	Amp.IC %	**Cb.Us.**	Amp.IC %
1	1	1	1	0.769	7.824	10.659	4.084	97.612	0.444	97.182	0.455
2	1	1	-1	0.841	2.893	9.855	3.749	73.002	0.419	97.091	0.339
3	1	-1	1	0.731	8.294	3.613	2.216	97.582	0.214	97.000	0.000
4	1	-1	-1	0.794	3.372	3.734	1.215	97.636	0.277	97.000	0.000
5	-1	1	1	0.751	8.429	10.565	7.006	97.832	0.262	97.182	0.455
6	-1	1	-1	0.853	4.275	10.288	3.486	72.925	0.571	97.182	0.455
7	-1	-1	1	0.757	6.161	3.695	1.705	97.584	0.172	97.000	0.000
8	-1	-1	-1	0.786	3.491	3.688	2.370	97.771	0.360	97.182	0.455

Um fato a ser ressaltado aqui é que a filtragem baseada em conteúdo não trabalha com o conceito de vizinhos. Portanto, o tamanho da vizinhança (fator A) não influencia em nada nesse método, mas ele foi mantido de propósito para ver se os resultados medidos comprovariam a sua não relevância. Sendo assim, idealmente todas as entradas referentes à influência do fator A deveriam ser zero. Como pode ser visto na Tabela 12, valores próximos de zero foram medidos para o tamanho da vizinhança em todas as métricas de resposta, com exceção da cobertura de usuários. Uma possível explicação do motivo de os valores não terem sido zero, é que em cada teste os usuários são escolhidos de forma aleatória, e também que a arbitrariedade é um critério de decisão do algoritmo, para que recomendações diferentes possam ser produzidas para uma mesma pessoa. Essas características acabam introduzindo erros que se refletem nesse cálculo da influência. Em relação à cobertura de usuários o cálculo apontou 16.36% de influência, mas essa cobertura se manteve praticamente estável em todos os testes, e portanto sua variação foi insignificante.

Tabela 12 – Influência dos fatores

Fatores	**Variáveis de Resposta**			
	RMSE	**Prod.**	**Cb.Cat.**	**Cb.Us.**
A	0.13	0.02	0.00	16.36
B	20.65	99.55	33.05	45.46
C	68.26	0.13	33.15	1.82
AB	0.59	0.01	0.00	1.82
AC	0.03	0.02	0.00	16.36
BC	6.35	0.20	33.80	16.36
ABC	3.99	0.06	0.00	1.82

Fonte: Elaborado pelo autor

Para interpretar o RMSE, novamente recorreu-se ao zoom do gráfico (Figura 33). Concluiu-se que os melhores resultados se deram nos testes 3, 5 e 7. Já em relação à produtividade e cobertura de catálogo juntas, verifica-se que as melhores respostas estão nas combinações dadas pelos experimentos 1 e 5. A cobertura de usuários se mostrou boa em todos os testes, atingindo aproximadamente 97%. Nesse caso foi possível maximizar todas as variáveis de resposta em um único experimento, e portanto o teste escolhido foi o número 5. O detalhe dessa combinação está mostrado na Tabela 13.

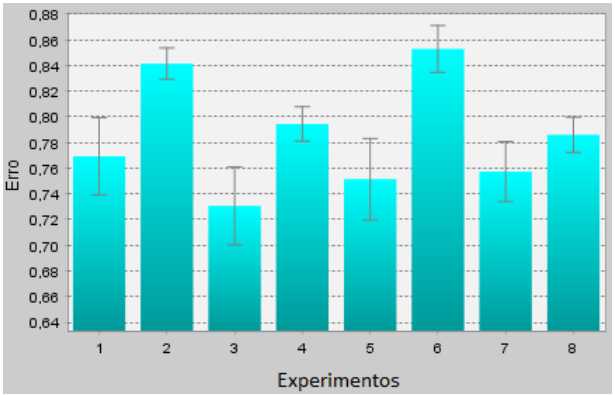


Figura 33 – RMSE do ensaio 02
Fonte: Elaborado pelo autor

Tabela 13 – Detalhes do teste escolhido no ensaio 02

Fator	**Exp. 5**
Tamanho da vizinhança	25
Itens candidatos	30
Comprimento da lista	5

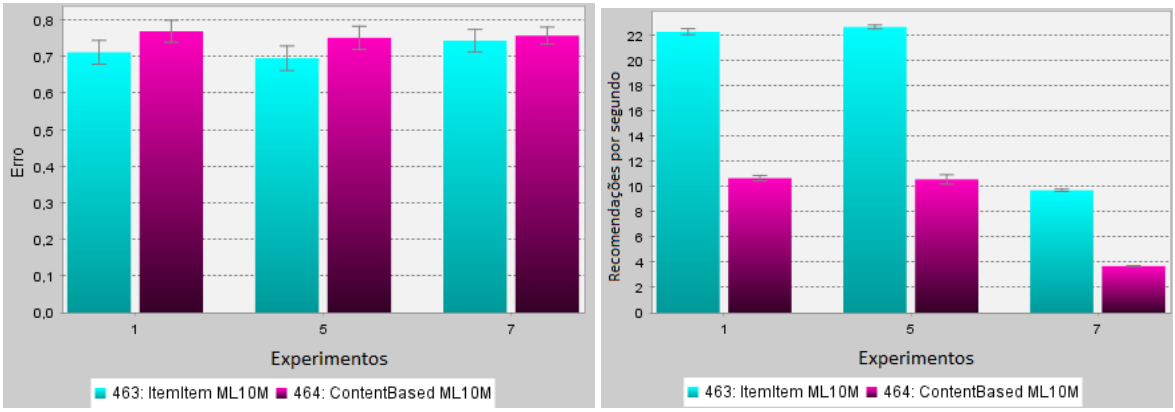
Fonte: Elaborado pelo autor

6.3.3 Comparativo entre os ensaios 01 e 02

Supondo-se que um algoritmo de recomendação foi colocado em produção, tendo sido devidamente testado e com seus parâmetros ajustados para o melhor desempenho. Ao fazer a implementação de um novo recomendador, deseja-se saber se esse deve ou não substituir o primeiro. Uma possível solução seria executar a avaliação de desempenho

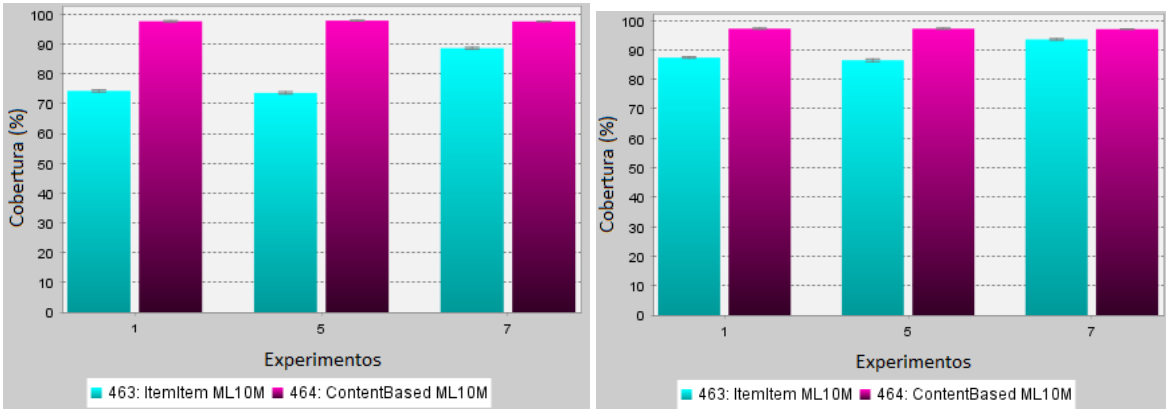
para o novo método e fazer uma comparação entre os dois. Esta é a situação simulada aqui.

Os melhores resultados do ensaio 1 se deram nos experimentos 1 e 7, e em relação ao ensaio 2 a combinação escolhida foi a do teste 5. As Figuras 34 e 35 mostram gráficos comparando os dois experimentos nesses casos. O método baseado em conteúdo apresentou as coberturas mais elevadas e acurácia semelhante. No entanto há uma notável perda na produtividade. Novamente, a escolha final dependeria do objetivo da aplicação. Se a taxa de 10.5 recomendações por segundo fosse suficiente para atender a demanda, a troca de algoritmos seria sim uma possibilidade. Nesse caso, a aplicação estaria melhorando as coberturas de catálogo e usuários, e apresentando valores superiores a 97%. Então, a filtragem baseada em conteúdo seria implementada com os valores dados pela teste número 5, e que pode ser visto na Tabela 13.



(a) Comparativo do RMSE (b) Comparativo da produtividade

Figura 34 – Comparativos entre os ensaios 1 e 2 - RMSE e produtividade
Fonte: Elaborado pelo autor



(a) Comparativo da cobertura de catálogo (b) Comparativo da cobertura de usuários

Figura 35 – Comparativos entre os ensaios 1 e 2 - coberturas
Fonte: Elaborado pelo autor

6.3.4 Ensaio 03 - Otimização de parâmetro através do experimento multinível

A partir dos ensaios 1 e 2, concluiu-se que o algoritmo mais eficaz foi a filtragem baseada em conteúdo com as combinações dadas pelo experimento 5. Suponha, no entanto, que deseja-se saber se é viável aumentar um pouco o comprimento da lista de recomendação sem comprometer os resultados obtidos até o momento. Desta maneira, é necessário realizar um teste multinível variando o tamanho da lista entre 3 e 15. Os outros parâmetros permaneceram fixos, conforme definido no teste 5 do ensaio 6.3.2. Os resultados estão resumidos na Tabela 14.

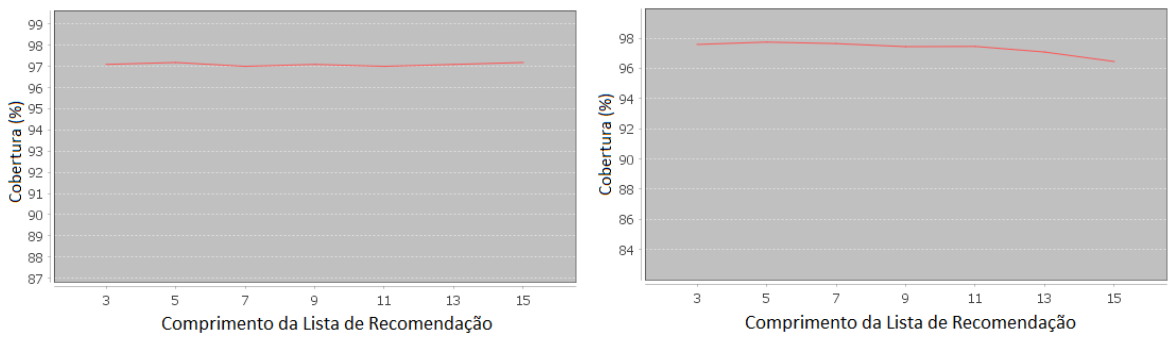
Tabela 14 – Resultados experimentais para o teste multinível

		Variáveis de Resposta							
Exp.	Tam.Lista	**RMSE**	Amp.IC %	**Prod.**	Amp.IC %	**Cb.Cat.**	Amp.IC %	**Cb.Us.**	Amp.IC %
1	3	0.767	8.477	11.184	2.511	97.586	0.383	97.091	0.339
2	5	0.747	10.614	10.944	1.412	97.742	0.303	97.182	0.455
3	7	0.757	9.287	10.789	3.252	97.642	0.288	97	0
4	9	0.805	9.313	10.788	1.362	97.440	0.272	97.091	0.339
5	11	0.776	9.075	10.524	1.851	97.454	0.209	97	0
6	13	0.802	8.194	10.253	3.289	97.083	0.377	97.091	0.339
7	15	0.820	6.827	9.568	2.476	96.455	0.278	97.182	0.455

Fonte: Elaborado pelo autor

A Figura 36 mostra na forma gráfica as respostas para as coberturas. A primeira delas (36a) permaneceu constante em todos os testes. Entre os tamanhos 3 e 11, a cobertura de produtos apresentou variações mínimas (36b). No que se refere a produtividade, houve uma queda constante, a qual é acentuada a partir do tamanho 9 (Figura 37a). Entre todas as medições, o RMSE foi o mais instável. Por isso, recorreu-se também a análise dos intervalos de confiança (Figura 37b). O tamanho 5 (teste 2) apresentou os mesmos erros dos tamanhos 7 e 11 (testes 3 e 5). Entretanto, nada pôde ser concluído em relação ao tamanho 9 (teste 4).

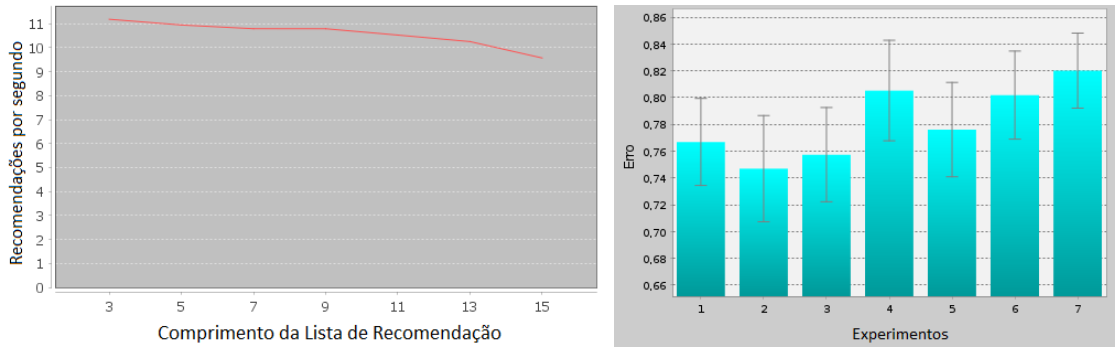
Das análises realizadas, entende-se que a lista de recomendação poderia ter seu tamanho aumentado de 5 para 7 sem que houvesse alteração significativa nos resultados já alcançados. O tamanho 9 seria também uma opção, só que nesse caso haveria uma possível perda na acurácia, e perdas mínimas na cobertura de produtos e na produtividade.



(a) Cobertura de usuários (b) Cobertura de catálogo

Figura 36 – Cobertura de usuários e catálogo para o teste multinível

Fonte: Elaborado pelo autor



(a) Produtividade (b) RMSE

Figura 37 – Produtividade e RMSE do teste multinível
Fonte: Elaborado pelo autor

6.3.5 Ensaio 04 - R2-Yahoo! User-User Collaborative Filtering

Esta avaliação tem por objetivo medir o tempo de recomendação levando em consideração a formação de uma possível fila. Será simulado a chegada de usuários em tempos distintos seguindo uma distribuição normal, havendo momentos em que o sistema possa estar ocioso e horários de pico. Esse é um comportamento parecido com o que um recomendador real enfrentaria ao longo de um dia. O período de tempo considerado aqui será de aproximadamente 7 minutos por repetição, mas os resultados podem ser extrapolados para períodos maiores com cargas proporcionais. O *setup* completo do experimento é mostrado no Quadro 5.

6.3.5.1 Setup

Quadro 5 – Configurações do ensaio 03

Variáveis de Resposta	Cobertura de Usuários
	Tempo de Recomendação (algoritmo)
	Tempo de Recomendação (algoritmo + fila)
Fatores (1; -1)	A. Carga (300; 750)
	B. Composto . Itens Candidatos (30; 60) . Recomendação Alternativa (Não; Sim)
Algoritmo	Filtragem Colaborativa Usuário-Usuário
Banco de Dados	R2-Yahoo!
Intervalo de Confiança	90%
Amplitude Máxima do IC	14%
Repetições	10
Tamanho da Vizinhança	25
Comprimento da Lista	10
Período de Tempo por Repetição	400 segundos
Relevância	6.5

Fonte: Elaborado pelo autor

6.3.5.2 Resultados

A Tabela 15 apresenta os resultados obtidos nesse ensaio.

Tabela 15 – Resultados experimentais para o ensaio 04

	Fatores		**Variáveis de Resposta**			
Exp.	**A**	**B**	**Tempo (ms)**	Amp.IC %	**Tempo com fila (ms)**	Amp.IC%
1	1	1	145.192	2.921	161.174	3.255
2	1	-1	150.376	2.212	167.779	1.950
3	-1	1	143.106	1.534	214.909	7.919
4	-1	-1	147.687	1.331	234.992	5.781

Fonte: Elaborado pelo autor

Se as requisições dos usuários fossem prontamente atendidas pelo sistema, cada pessoa receberia em média uma resposta em aproximadamente 150 ms. Entretanto, esse cenário muda se considerarmos também a formação de filas de espera. Dessa forma, o tempo real de espera do usuário no experimento 1 é na verdade aumentado em 11.01%. Já no experimento 4, que representa uma situação com carga maior, esse aumento chega a 59.11%. Um ponto a ser considerado é que isso se trata de uma média. Pode ter havido pessoas que esperaram mais (requisições próximas ao centro da distribuição normal), assim como outras que sequer ficaram na fila.

A Tabela 16 mostra a influência dos fatores. O fator composto B foi o que mais interferiu no tempo quando a fila é desconsiderada. Já com a fila, a carga de trabalho (A) foi o fator determinante. Esses resultados mostram que de nada adianta ajustar bem os parâmetros do algoritmo se a carga estiver mal dimensionada. O RSE apresenta um histograma com a distribuição dos usuários em cinco classes distintas em função do tempo de recomendação *TRec*. As classes estão especificadas na Tabela 17. Por exemplo, pessoas rotuladas com *c1* são aquelas cujo tempo de espera na fila não ultrapassou metade do tempo de recomendação.

Tabela 16 – Influência dos fatores

Fatores	**Variáveis de Resposta**	
	Tempo (ms)	**Tempo com fila (ms)**
A	19.24	94.24
B	80.46	4.59
AB	0.31	1.17

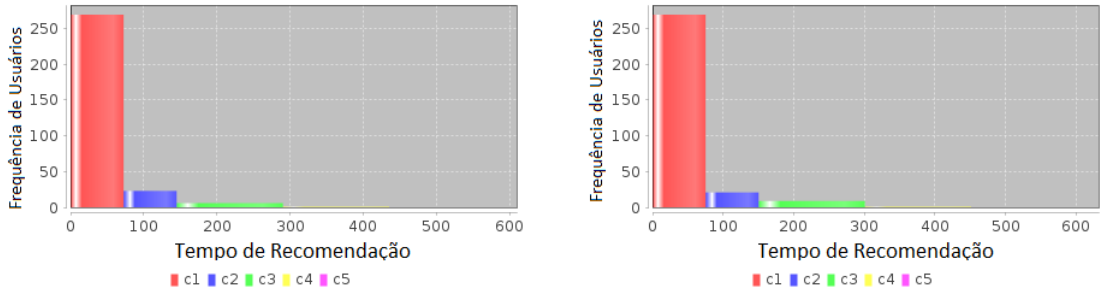
Fonte: Elaborado pelo autor

Tabela 17 – Classes de tempo

Classe	**Limites**
c1	(0 - 0.5) TRec
c2	(0.5 - 1) TRec
c3	(1 - 2) TRec
c4	(2 - 3) TRec
c5	$\geq$ 3 TRec

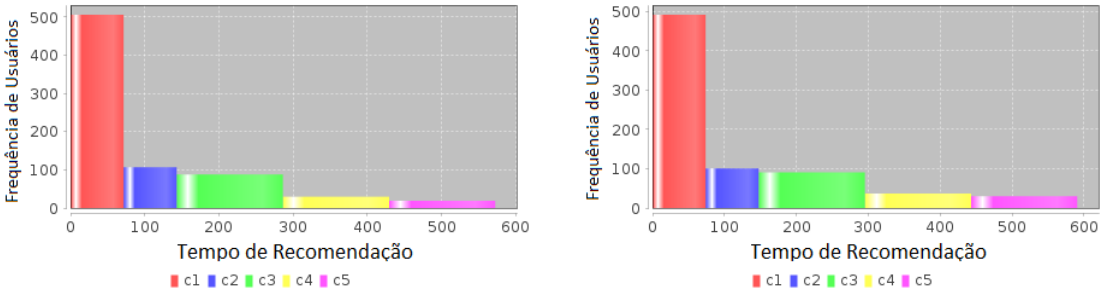
Fonte: Elaborado pelo autor

A distribuição dos usuários nas classes de tempo é apresentada na Figuras 38 e 39. Note que nos teste 1 e 2 a grande maioria da carga ficou entre c1 e c2. Por outro lado nos experimentos 3 e 4, parte considerável dela foi distribuida entre c3, c4 e c5. Isso significa que a quantidade de requisições de 750 causou uma sobrecarga indesejável, pois entre 15 e 20% das pessoas gastaram mais tempo na fila de espera do que com a recomendação propriamente dita.



(a) Experimento 1 (b) Experimento 2

Figura 38 – Distribuição de usuários em função do tempo de espera na fila - testes 1 e 2
Fonte: Elaborado pelo autor



(a) Experimento 3 (b) Experimento 4

Figura 39 – Distribuição de usuários em função do tempo de espera na fila - testes 3 e 4
Fonte: Elaborado pelo autor

Em um cenário real, a especificação de um acordo de nível de serviço juntamente com os resultados apresentados, determinariam a quantidade máxima da carga de trabalho. Por exemplo, se fosse permitido que até 10% das requisições ultrapassassem a classe *c2*, os cenários apresentados nos experimentos 3 e 4 seriam inviáveis.

7 Utilização do RSE fora do contexto original

Embora não fosse um dos objetivos iniciais ao criar o RSE, percebeu-se ao longo do desenvolvimento que grande parte do *framework* poderia ser reaproveitada na avaliação de desempenho de outros tipos de ambientes. É evidente a necessidade da criação de novos elementos dependentes do contexto, como por exemplo algoritmo e estrutura dos dados no banco. Entretanto todo o mecanismo da avaliação, cálculos estatísticos e produção de gráficos podem ser reutilizados.

Assim, surgiu a necessidade de testar o quão difícil seria fazer um novo software, que herdaria o núcleo funcional do RSE e com a finalidade de avaliar o desempenho em outro contexto.

Este Capítulo apresenta um estudo de caso, em que o RSE é utilizado na resolução de um problema real vivenciado por uma empresa parceira da universidade. A empresa utiliza da tecnologia para fazer o gerenciamento de frotas de veículos. Assim, um dos algoritmo de recomendação implementados foi adaptado para a situação em questão. Devido à flexibilidade do RSE, foi possível utilizá-lo também para avaliar o desempenho nesse novo contexto. Mais detalhes são mostrados na Seção 7.1.

7.1 O problema

A *DDMX*[1] é uma empresa que faz o monitoramento inteligente de frotas e gerenciamento logístico. Ela proporciona aos seus clientes a otimização dos serviços prestados e redução significativa nos custos operacionais. Para a realização das tarefas são instalados inúmeros sensores em cada veículo da frota, que coletam dados e os enviam periodicamente à empresa. Eles são automaticamente processados e analisados por softwares, e os resultados são disponibilizados para os clientes.

No entanto, a detecção de possíveis falhas dos veículos ainda não está completamente automatizada. Esta é uma das demandas recentes da empresa. Relacionado a isso, uma segunda questão seria o desenvolvimento de um aplicativo *mobile* que pudesse ser usado pelo técnico como teste inicial no momento da instalação dos sensores do veículo. Após a coleta dos primeiros dados, o aplicativo deveria ser capaz de dizer rapidamente se houve ou não falha. Para isso, ele contaria também com o acesso a informações previamente coletadas de outros veículos.

[1] DDMX <http://ddmx.com.br/>. Acesso em: 13 dez. 2018.

7.2 Algoritmo

Embora os sistemas de recomendação e a detecção de falhas sejam de domínios completamente diferentes, eles compartilham do reconhecimento de padrões como parte da solução dos seus respectivos problemas. Percebeu-se então que a filtragem baseada em conteúdo poderia ser adaptada e usada como *core* do aplicativo demandado pela *DDMX*.

O algoritmo modificado trata os dados coletados de cada veículo como sendo vetores multidimensionais. Assim, ele é capaz de compará-los e encontrar na base de dados os perfis mais similares ao veículo testado. A assunção aqui é que veículos com alto grau de similaridade entre si tendem a apresentar o mesmo comportamento. Por exemplo, se foi encontrado um carro que apresentou falhas e que possui 99% de semelhança com o veículo em questão, pode-se afirmar que o último também possui as mesmas falhas.

7.3 Dados

Os testes foram feitos com dois bancos de dados distintos. O primeiro, denominado *DDMX_1*, conta com aproximadamente 25.000 entradas, que foram coletadas entre os meses de fevereiro e abril de 2018. Praticamente todas as informações disponíveis estão sendo usadas descartando-se apenas as irrelevantes, como identificadores e datas. No total, foram criadas 88 dimensões. Cada dimensão representa uma informação específica coletada pelo sensor, como por exemplo temperatura e pressão do óleo.

Já o segundo banco (*DDMX_2*) possui aproximadamente 36.000 entradas coletadas entre fevereiro e maio de 2018. Entretanto, apenas 13 dimensões foram criadas para cada veículo. Elas foram escolhidas por um especialista da empresa como sendo as mais relevantes ao problema. Ambos os bancos de dados dados foram pré-processados com normalização e retirada de *outliers*.

7.4 Experimentos

Para os ensaios escolheu-se como variáveis de resposta a acurácia e produtividade. A primeira mede a taxa de sucesso do método em detectar falhas, e a produtividade conta o número de processamentos que o algoritmo é capaz de fazer em um segundo. Os fatores utilizados foram os veículos candidatos e a semelhança. O primeiro diz respeito ao tamanho do domínio que o algoritmo vai trabalhar. O segundo indica que o resultado será baseado nos n perfis mais similares ao dado de teste. Por exemplo, se for estabelecido 100 candidatos e semelhança de 5, o método irá analisar 100 perfis de veículos e os resultados serão baseados nos 5 que mais correspondem ao veículo testado.

7.4.1 Ensaios 01 e 02

Os dois primeiros testes tiveram como objetivo obter as primeiras impressões do funcionamento do método, e também comparar em qual dos bancos ele apresentaria melhor desempenho. Para tanto, foram realizados dois ensaios separados, e em seguidas seus resultados foram comparados em um único gráfico. Mais detalhes estão mostrados na Tabela 18. A única diferença entre os ensaios foi o banco utilizado (*DDMX_1* e *DDMX_2*).

Tabela 18 – Configurações dos ensaios 01 e 02

Variáveis de Resposta	Acurácia
	Produtividade
Fatores (1; -1)	A. Candidatos (500; 2500)
	B. Semelhança (1; 5)
Intervalo de Confiança	90%
Amplitude Máxima do IC	14%
Repetições	7
Carga de Trabalho	500 usuários por repetição

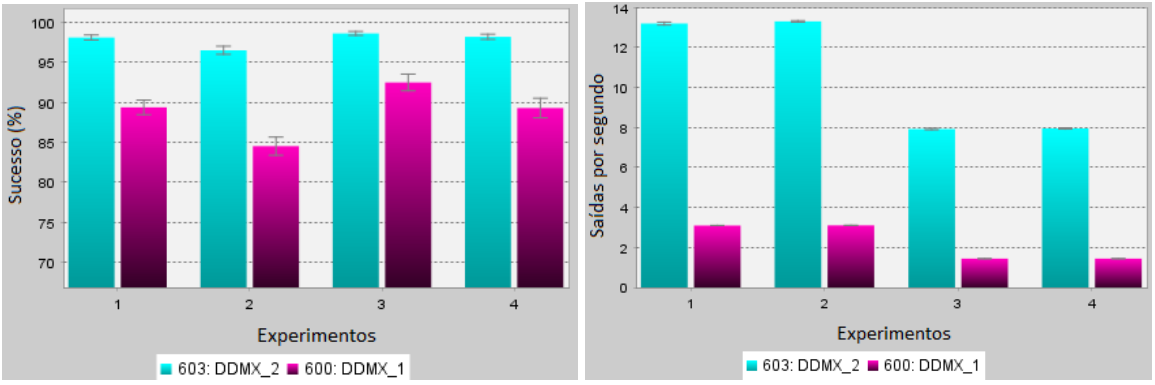
Fonte: Elaborado pelo autor

Como os testes possuem dois fatores de entrada com duas variações cada, são necessárias quatro combinações. Elas estão detalhadas na Tabela 19, e a comparação dos resultados pode ser vista na Figura 40.

Tabela 19 – Combinações feitas nos ensaios 01 e 02

Experimento	Candidatos	Semelhança
1	500	1
2	500	5
3	2500	1
4	2500	5

Fonte: Elaborado pelo autor



(a) Acurácia (b) Produtividade

Figura 40 – Comparativos entre os ensaios 01 e 02

Fonte: Elaborado pelo autor

Percebe-se que as melhores respostas se deram com o banco *DDMX_2*. Além disso, há aparentemente uma pequena diminuição na acurácia quando foi usada a semelhança de

5 (experimentos 02 e 04). A necessidade de uma investigação mais detalhada nesse fator levou à realização do próximo experimento.

7.4.2 Ensaio 03

Este teste foi feito com 8 valores de semelhança diferentes entre 1 e 22. Os candidatos foram fixados em 2500. Os resultados estão na Figura 41, e confirmaram que a semelhança de 1 apresenta de fato a melhor precisão.

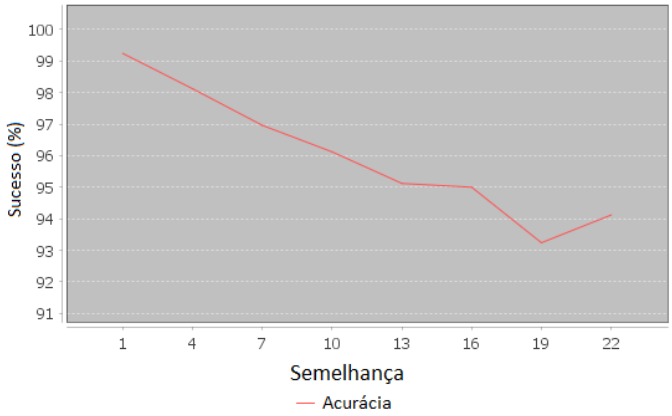
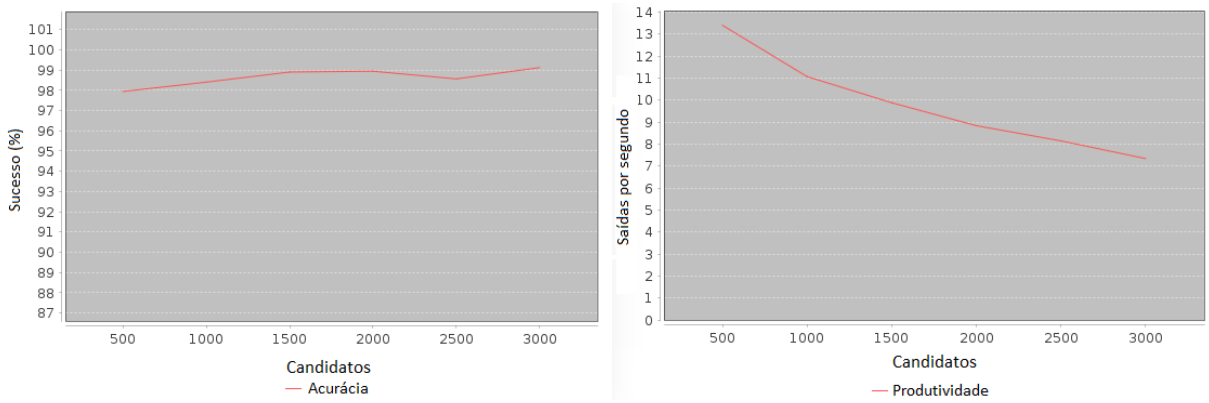


Figura 41 – Acurácia medida para diferentes valores de semelhança

7.4.3 Ensaio 04

É fato que a acurácia tende a aumentar com a quantidade de candidatos. Uma vez que já foi escolhido o banco de dados mais adequado (DDMX_2) e a melhor semelhança (1), o próximo teste verificará em mais detalhes os resultados em relação a quantidade de candidatos, na tentativa de encontrar um bom custo-benefício entre performance e acurácia. O número de candidatos foi variado entre 500 e 3000, e as medições obtidas são apresentadas na Figura 42.



(a) Acurácia (b) Produtividade

Figura 42 – Acurácia e produtividade para diferentes valores de candidatos
Fonte: Elaborado pelo autor

Percebe-se que com 1500 candidatos foi obtido uma acurácia muito próxima de 99%, sendo que esta pouco variou acima dos 1500 para o intervalo medido. Em contrapartida há uma queda constante na produtividade com o aumento do número dos mesmos.

7.5 Resultados

Considerando as análises feitas e os prós e contras envolvidos, a Tabela 20 mostra uma possível escolha dos parâmetros do algoritmo. Foi obtida uma acurácia de 98,9% e produtividade de 9,88 saídas por segundo.

Tabela 20 – Parâmetros escolhidos

Parâmetros de entrada	Valores escolhidos
Banco de dados	DDMX_2
Candidatos	1500
Semelhança	1

Fonte: Elaborado pelo autor

Caso houvesse necessidade de uma acurácia ainda maior, a quantidade de candidatos poderia ser extrapolada. Entretanto, é preciso ter em mente que isso acarretaria em um tempo de resposta maior, muito embora ele possa ser otimizado também com a utilização de um computador adequado. Conforme era esperado, o banco de dados mais completo obteve resultados piores. É possível que as outras dimensões tenham introduzido ruídos, ou ainda uma melhor normalização poderia ter sido mais efetiva.

8 Conclusão

O RSE foi construído para facilitar a avaliação de desempenho dos sistemas de recomendação. Ele conta com um conjunto amplo de métricas, e se baseia em conhecimentos estatísticos para que os avaliadores possam tirar conclusões confiáveis. A ferramenta foi feita em módulos para que pudesse ser facilmente estendida, e os testes demonstraram sua simplicidade de uso e eficácia. Portanto, os seus objetivos foram alcançados.

O *software* realiza experimentos *offline*. Embora testes *online* sejam mais realísticos, eles consomem muito tempo e necessitam ter os dados reais disponíveis, o que muitas vezes não é possível. O RSE pode ser usado para pré-selecionar os melhores algoritmos e/ou parâmetros para um teste real. Assim, somente os candidatos mais fortes participariam do experimento *online*, poupando tempo e recursos financeiros.

Nos testes apresentados, mostrou-se como o RSE pode ser usado na comparação de algoritmos distintos. Além disso, percebeu-se a importância do dimensionamento correto da carga de trabalho, e como o *framework* ajuda nessa tarefa. Outro ponto é que seus gráficos facilitam a interpretação dos resultados, tornando-a mais intuitiva.

É preciso levar em consideração também as diferentes características de cada algoritmo e o objetivo em questão. Essas características foram discutidas na Seção 2.3.4. Extrapolando o contexto dos sistemas de recomendação, o RSE pôde ainda ser usado na avaliação de um segundo domínio. Excluindo as modificações inevitáveis e específicas ao novo ambiente, foram necessárias poucas adaptações no seu núcleo. Ele se mostrou flexível e adaptável ao novo contexto.

É provável que a parte mais trabalhosa do teste tenha sido a preparação inicial do ambiente, no entanto ela é realizada uma única vez. A ferramenta busca abstrair ao máximo a complexidade envolvida na avaliação de desempenho, possibilitando sua execução através de configurações e métodos intuitivos. Ele está disponível publicamente[1], e espera-se que seja útil para desenvolvedores tanto de sistemas de recomendação quanto de outros domínios, permitindo também que esses façam suas próprias mudanças e adaptações. Por fim, a Seção 8.1 propõe direções futuras, no sentido de ampliar as possibilidades de trabalho e versatilidade do *software*.

8.1 Trabalhos futuros e melhorias

A execução dos testes colocou o RSE à prova, concluindo a sua eficácia mas também permitindo a observação de melhorias:

[1] https://github.com/paulovgs/RSEvaluator

Otimização dos dados de entrada: os resultados apresentados no Capítulo 7 mostraram que um conjunto de dados com mais dimensões não necessariamente possui o melhor desempenho. Com o auxílio do RSE, novas avaliações podem ser feitas com pequenos grupos desses eixos, a fim de se obter um conjunto otimizado para o problema abordado. Por fim, este conjunto seria comparado àquele obtido pelo especialista, a fim de melhorá-lo ou validá-lo experimentalmente.

Visões: Massa e Avesani (49) apresentam uma interessante ideia que pode ser aplicada ao RSE: a divisão dos dados de entrada em porções com características distintas, batizadas de "visões". Por exemplo, uma possível carga de trabalho seria composta de três visões, uma com 60% de usuários com mais de 50 avaliações, outra com 35% das pessoas que avaliaram entre 10 e 50 produtos e a última de 5% dos que possuem menos de 10 avaliações. Esse conceito ajudaria a refletir melhor as características do problema em questão, permitindo o estudo mais focado em certas particularidades do algoritmo.

Tamanho do histórico: criar outras proporções de divisão das avaliações entre treinamento e teste.

Correlação: o cálculo atual salva somente os n maiores valores positivos. Adicionar também a possibilidade de armazenar os n maiores valores negativos da correlação implementada, pois há situações em que isso é considerado. Neste momento, a ferramenta faz as similaridades pela correlação de *Pearson*. Outras opções podem ser adicionadas.

Classes de tempo: permitir que o avaliador configure os limites das classes de tempo.

Fatores e variáveis de resposta: disponibilizar novos fatores e mais métricas de cada grupo para ampliar o leque de possibilidades do avaliador. Outro ponto aqui é que a importância da ordem dos itens é diretamente proporcional ao tamanho da lista de recomendação. Isso é modelado com diferentes funções de desconto do nDCG. Portanto, é interessante que outras formas de decaimento sejam disponibilizadas.

Desempenho temporal: foi usado um computador comum para os experimentos, e a performance dos recomendadores poderia ter sido maior se os testes fossem feitos em um servidor apropriado. Ainda que minimamente, certos detalhes que influenciaram no desempenho dizem respeito ao próprio teste. Por exemplo, a coluna *is_history* da tabela de avaliações não existiria em um cenário real, simplificando a consulta. Embora não seja o foco do trabalho, serão estudadas também melhorias na implementação dos métodos.

Experimento: o planejamento multifatorial com dois níveis foi escolhido por ter um ótimo custo-benefício entre simplicidade e resultados. No entanto, os autores de (29)

sugerem que os testes sejam feitos em duas etapas. Na primeira delas, a avaliação seria realizada com uma maior quantidade de fatores e poucos níveis. Dessa forma, seria possível entender o impacto causado por eles. O segundo experimento contaria com um número reduzido de fatores, sendo que aqueles com maior impacto teriam mais níveis. Portanto, para proporcionar avaliações mais completas, pretende-se adicionar também um modo multifatorial com mais de dois níveis de variação.

Interface: criar interface gráfica para o usuário, facilitando ainda mais seu uso.

Bancos *NoSQL*: bancos de dados não relacionais (conhecidos como *NoSQL*), têm se popularizado bastante por apresentar uma performance elevada em ambientes *big data*. Então, seria interessante também estudar a viabilidade e criação de *plugins* para esses bancos.

Referências

1 DAS, S. Maintenance action recommendation using collaborative filtering. *International Journal of Health Policy and Management*, Citeseer, v. 4, n. 2, p. 7–12, 2013. 18

2 GADEPALLY, V. N. et al. *Recommender Systems for the Department of Defense and the Intelligence Community*. [S.l.], 2016. 18

3 MACKENZIE, I.; MEYER, C.; NOBLE, S. How retailers can keep up with consumers. *McKinsey & Company*, 2013. 18

4 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. 18, 30

5 LÜ, L. et al. Recommender systems. *Physics Reports*, Elsevier, v. 519, n. 1, p. 1–49, 2012. 18, 23, 25, 27, 29, 31, 32, 38, 50

6 GUNAWARDANA, A.; SHANI, G. A survey of accuracy evaluation metrics of recommendation tasks. *Journal of Machine Learning Research*, v. 10, n. Dec, p. 2935–2962, 2009. 18, 25, 38, 39, 50, 51, 53

7 HILBERT, M.; LÓPEZ, P. The world's technological capacity to store, communicate, and compute information. *science*, American Association for the Advancement of Science, p. 1200970, 2011. 19

8 REINSEL, D. et al. The expanding digital universe. *White paper, IDC*, 2007. 19

9 CHEN, M.; MAO, S.; LIU, Y. Big data: A survey. *Mobile networks and applications*, Springer, v. 19, n. 2, p. 171–209, 2014. 19, 20

10 MCAFEE, A. et al. Big data: the management revolution. *Harvard business review*, v. 90, n. 10, p. 60–68, 2012. 19, 20

11 BUGHIN, J. Big data, big bang? *Journal of Big Data*, Nature Publishing Group, v. 3, n. 1, p. 2, 2016. 20

12 LU, J. et al. Recommender system application developments: a survey. *Decision Support Systems*, Elsevier, v. 74, p. 12–32, 2015. 20, 43, 47

13 SARWAR, B. et al. *Application of dimensionality reduction in recommender system-a case study*. [S.l.], 2000. 22, 51

14 RODGERS, J. L.; NICEWANDER, W. A. Thirteen ways to look at the correlation coefficient. *The American Statistician*, Taylor & Francis Group, v. 42, n. 1, p. 59–66, 1988. 25

15 MELVILLE, P.; SINDHWANI, V. Recommender systems. In: *Encyclopedia of machine learning*. [S.l.]: Springer, 2011. p. 829–838. 26, 30

16 PAZZANI, M. J.; BILLSUS, D. Content-based recommendation systems. In: *The adaptive web*. [S.l.]: Springer, 2007. p. 325–341. 26, 27

17 SHARMA, D.; JAIN, S. Evaluation of stemming and stop word techniques on text classification problem. *International Journal of Scientific Research in Computer Science and Engineering*, v. 3, n. 2, p. 1–4, 2015. 27

18 KANTOR, P.; RICCI, F.; ROKACH, L. Introduction to recommender systems handbook. *Recommender systems handbook*, p. 1–35, 2009. 28

19 ROBIN, B. *Knowledge-based recommender systems*. [S.l.]: Marcel Dekker, 2000. 28, 30

20 JANNACH, D. et al. *Recommender systems: an introduction*. [S.l.]: Cambridge University Press, 2010. 29

21 HERLOCKER, J. L. et al. An algorithmic framework for performing collaborative filtering. In: ACM. *ACM SIGIR Forum*. [S.l.], 2017. v. 51, n. 2, p. 227–234. 29, 44, 45

22 GUO, H. Soap: Live recommendations through social agents. In: *Fifth DELOS Workshop on Filtering and Collaborative Filtering, Budapest*. [S.l.: s.n.], 1997. 29

23 SHARDANAND, U.; MAES, P. Social information filtering: algorithms for automating "word of mouth". In: ACM PRESS/ADDISON-WESLEY PUBLISHING CO. *Proceedings of the SIGCHI conference on Human factors in computing systems*. [S.l.], 1995. p. 210–217. 29

24 KOREN, Y.; BELL, R.; VOLINSKY, C. Matrix factorization techniques for recommender systems. *Computer*, IEEE, n. 8, p. 30–37, 2009. 30

25 SHANI, G.; WARDANA, A. Evaluating recommendation systems. In: *Recommender systems handbook*. [S.l.]: Springer, 2011. p. 257–297. 30, 38, 52, 53, 57

26 LAM, S. K.; RIEDL, J. Shilling recommender systems for fun and profit. In: ACM. *Proceedings of the 13th international conference on World Wide Web*. [S.l.], 2004. p. 393–402. 30

27 AMATRIAIN, X. et al. Data mining methods for recommender systems. In: *Recommender systems handbook*. [S.l.]: Springer, 2011. p. 39–71. 30

28 CHEN, L.; PU, P. Critiquing-based recommenders: survey and emerging trends. *User Modeling and User-Adapted Interaction*, Springer, v. 22, n. 1-2, p. 125–150, 2012. 32

29 JAIN, R. *The art of computer systems performance analysis: techniques for experimental design, measurement, simulation, and modeling*. [S.l.]: John Wiley & Sons, 1990. 34, 36, 37, 38, 57, 88

30 CAMPOS, G. M. Estatística prática para docentes e pós-graduandos. *São Paulo: Faculdade de Odontologia de Ribeirão Preto*, 2001. 35

31 KOWALD, D.; KOPEINIK, S.; LEX, E. The TagRec framework as a toolkit for the development of tag-based recommender systems. In: ACM. *Adjunct Publication of the 25th Conference on User Modeling, Adaptation and Personalization*. [S.l.], 2017. p. 23–28. 40

32 SAID, A.; BELLOGÍN, A. RiVal: a toolkit to foster reproducibility in recommender system evaluation. In: ACM. *Proceedings of the 8th ACM Conference on Recommender systems*. [S.l.], 2014. p. 371–372. 40

33 EKSTRAND, M. D. et al. Rethinking the recommender research ecosystem: reproducibility, openness, and Lenskit. In: ACM. *Proceedings of the fifth ACM conference on Recommender systems*. [S.l.], 2011. p. 133–140. 40

34 DAYAN, A. et al. Recommenders benchmark framework. In: ACM. *Proceedings of the fifth ACM conference on Recommender systems*. [S.l.], 2011. p. 353–354. 40

35 SCRIMINACI, M. et al. Idomaar: A framework for multi-dimensional benchmarking of recommender algorithms. 2016. 40

36 LONI, B.; SAID, A. Wraprec: An easy extension of recommender system libraries. In: ACM. *Proceedings of the 8th ACM Conference on Recommender systems*. [S.l.], 2014. p. 377–378. 40

37 HAHSLER, M. *recommenderlab: Lab for Developing and Testing Recommender Algorithms*. [S.l.], 2018. R package version 0.2-3. Disponível em: <http://lyle.smu.edu/IDA/recommenderlab/>. 40

38 GAMMA, E. *Design patterns: elements of reusable object-oriented software*. [S.l.]: Pearson Education India, 1995. 42

39 BREESE, J. S.; HECKERMAN, D.; KADIE, C. Empirical analysis of predictive algorithms for collaborative filtering. In: MORGAN KAUFMANN PUBLISHERS INC. *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*. [S.l.], 1998. p. 43–52. 44

40 WHITE, S. How to strike a match. *Development Cycles*, 2004. 45

41 WU, H. C. et al. Interpreting tf-idf term weights as making relevance decisions. *ACM Transactions on Information Systems (TOIS)*, ACM, v. 26, n. 3, p. 13, 2008. 46

42 RIJSBERGEN, C. J. van. *Information retrieval*. 2. ed. London: Butterworths, 1979. Disponível em: <http://www.dcs.gla.ac.uk/Keith/Preface.html>. 51

43 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. 52

44 ANNA, B. Recommender systems-it's not all about the accuracy. *Gab41, Gab41*, 2016. 52

45 ZHOU, T. et al. Solving the apparent diversity-accuracy dilemma of recommender systems. *Proceedings of the National Academy of Sciences*, National Acad Sciences, v. 107, n. 10, p. 4511–4515, 2010. 52

46 HARPER, F. M.; KONSTAN, J. A. The movielens datasets: History and context. *Acm transactions on interactive intelligent systems (tiis)*, ACM, v. 5, n. 4, p. 19, 2016. 62, 71

47 PARK, D. H. et al. A literature review and classification of recommender systems research. *Expert Systems with Applications*, Elsevier, v. 39, n. 11, p. 10059–10072, 2012. 71

48 VIG, J.; SEN, S.; RIEDL, J. The tag genome: Encoding community knowledge to support novel interaction. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, ACM, v. 2, n. 3, p. 13, 2012. 71

49 MASSA, P.; AVESANI, P. Trust-aware recommender systems. In: ACM. *Proceedings of the 2007 ACM conference on Recommender systems*. [S.l.], 2007. p. 17–24. 88