UNIVERSIDADE FEDERAL RURAL DO RIO DE JANEIRO
INSTITUTO DE MULTIDISCIPLINAR

HUGO DINIZ REBELO

Recsys.jl: Um framework de Sistemas de Recomendação

Prof. Filipe Braida do Carmo, M.Sc.
Orientador

Nova Iguaçu, Fevereiro de 2016

Recsys.jl: Um framework de Sistemas de Recomendação

Hugo Diniz Rebelo

Projeto Final de Curso submetido ao Departamento de Ciência da Computação do Instituto Multidisciplinar da Universidade Federal Rural do Rio de Janeiro como parte dos requisitos necessários para obtenção do grau de Bacharel em Ciência da Computação.

Apresentado por:

Hugo Diniz Rebelo

Aprovado por:

Prof. Filipe Braida do Carmo, M.Sc.

Prof. Carlos Eduardo Ribeiro de Mello, Ph.D.

Prof. Leandro Guimaraes Marques Alvim, D.Sc.

NOVA IGUAÇU, RJ - BRASIL

Fevereiro de 2016

Agradecimentos

Primeiramente gostaria de agradecer todo o corpo docente do curso de ciência da computação da Universidade Federal Rural do Rio de Janeiro, em especial ao prof. Filipe Braida que sempre teve a total atenção, compreensão e paciência em todo curso em especial na orientação deste trabalho.

Gostaria de dedicar essa vitoria aos meus pais João Francisco e Maria das Graças que devo absolutamente tudo que tenho na minha vida, são as minhas referências, meus avós Aldevina, José, Rosa, avós que são como meus pais mais velhos, aos meus tios José, Tânia, Elisia, Helena, Regina, Paulo, Sílvia, Edison, Sílvio que também são como meus pais, meus primos que considero todos meus irmãos, os que cresceram comigo como Lucas e Gabriel, os que hoje são meus vizinhos como a Márcia e Pedro, esse por sinal é meu primo, vizinho e ainda é amigo do curso de graduação. Minha prima Juliana, uma pessoa que cresceu junto comigo, passamos pelas mesmas fases da vida juntos, ela é minha irmã gêmea que nasceu alguns dias antes. Minhas primas Mariana e Marina que de distante só as nossas casas.

Não poderia deixar de agradecer aos meus amigos, que sempre estiveram junto comigo, aos meus amigos do Ferreira Viana, Luana, Daniel , Michael, Jessica, Kevyn, Jonathan, amigos tive o prazer conviver durante aqueles belos anos de ensino médio. Ao Kleyton, Raul, Michel, Larissa, Rafael, Egberto, Jéssica, Jamile, Victor Pedro, Liliane, Ygor, Alexsander, Júlio, Suzane, amigos que fiz no curso de Ciência da computação, amigos que muito se esforçaram, pouco dormiram em cada sofrido final de período junto comigo durante esse curso, mas todo esse esforço que muito valeu a pena. A Danielle, Victor e o Pedro Henrique, amigos que muito me fizeram crescer como pessoa e como profissional, nos anos que estivemos juntos no fundão.

RESUMO

Recsys.jl: Um framework de Sistemas de Recomendação

Hugo Diniz Rebelo

Fevereiro/2016

Orientador: Filipe Braida do Carmo, M.Sc.

Com o advento da globalização, começam a aparecer dificuldade nas escolhas de produtos e serviços por causa do grande volume dados apresentados. Isso torna o tema de sistemas de recomendação extremamente atrativo, mas ao mesmo que o tema esta no foco de muitos pesquisadores e instituições, ainda existe uma dificuldade em encontrar disponível implementações de algoritmos clássicos da área para ser utilizados, isso acaba que uma parte significativa do tempo de pesquisa para desenvolvimento de novos algorítimos seja utilizado para desenvolvimento de algo que já exista.

Este trabalho tem como objetivo contribuir com a pesquisa na área de sistemas de recomendação, criando uma ferramenta na qual facilite o uso e o desenvolvimento de técnicas de recomendação baseadas em filtragem colaborativa, de forma a torna a pesquisa de novas técnicas na área de sistema de recomendação mais ágil, sendo um *framework* modular e já tendo algorítimos clássicos implementados. Além disso ele deve ser elaborado de forma que seja fácil sua utilização de forma que possa ser utilizado por um professor em sala de aula.

ABSTRACT

Recsys.jl: Um framework de Sistemas de Recomendação

Hugo Diniz Rebelo

Fevereiro/2016

Advisor: Filipe Braida do Carmo, M.Sc.

With the advent of globalization, some difficulties starts to show up in choices of products and services because of the large volume of data presented. This makes the Recommender Systems topic extremely attractive, but even if this subject is in focus of many researchers and institutions, we still have difficulties finding available implementations of classical algorithms in the area, it turns out to consume a significant part of the research time developing something that already exists, instead of develop new algorithms.

This work have objective to contribute to research in recommendation systems, creating a tool in which facilitates the use and development recommendation techniques in collaborative filtering, in order to make research into new techniques in recommendation systems more agile , being a modular framework and already having implemented algorithms classics. Furthermore it should be developed so that it is easy to use so that it can be used by a teacher in the classroom.

Lista de Figuras

Lista de Tabelas

Lista de Códigos

Lista de Abreviaturas e Siglas

API	Application Programming Interface
GPL	General Public License
K-NN	K-nearest neighbors
MAE	Mean Absolute Error
MIT	Massachusetts Institute of Technology
PDM	Processo de Decisão Markoviano
RMSE	Root Mean Square Error
SVD	Singular Value Decomposition

Sumário

Capítulo 1

Introdução

1.1 Motivação

Com o advento da globalização, começam a aparecer dificuldades nas escolhas de produtos e serviços por causa do grande volumes de dados apresentados. Nos anos de 2008 e 2009 foram produzidos cerca de 6,3 quintilhões de *bytes* todos os dias e surpreendentemente 90% dos dados do mundo foram criados nos últimos anos, decorrente a adesão das grandes empresas à *internet* e criação de redes sociais e dispositivos móveis. [17]

Os sistemas de recomendação nasceram com o objetivo reduzir esta grande quantidade de informação, realizando filtragens de itens baseado no perfil de interesses do usuário. [35] Esses sistemas têm sido utilizados com sucesso para obtenção de informações, produtos ou serviços personalizados, que podem ser ajustados às necessidades específicas de um usuário, possibilitando uma maior eficácia em um processo de buscas.

Para a criação desses sistemas de recomendação é muito interessante que tenhamos ferramentas que facilite o desenvolvimento deles, tendo nelas já implementados os algorítimos clássicos da literatura e que também tenham formas de comparação com diversos algorítimos e com diversas métricas afins de ter uma base comparação desses algorítimos para uma base de dados qualquer.

Visto esse cenário que mostra a importância de sistemas de recomendação atualmente, este trabalho tem como objetivo contribuir com a pesquisa na área de sistemas de recomendação, criando uma ferramenta na qual facilita o uso e o desenvolvimento de técnicas de recomendação baseadas em filtragem colaborativa, de forma a tornar a busca de novas técnicas na área de sistema de recomendação mais ágil, tendo já implementado no *framework* algorítimos clássicos da literatura, as principais métricas de avaliação, e tendo sua estrutura modular de tal forma que mudança de algorítimos não gere retrabalho.

1.2 Objetivo

Os objetivos primários do *framework* Recsys.jl são:

- Ser elaborado de forma modulável no qual tenha o maior reaproveitamento de código possível, fazendo que os modelos já implementados não precisem ser reimplementados.
- A criação de novos módulos deverá ser feita com o mínimo de esforço possível para o criador desse modulo.
- O *framework* precisa ser elaborado de forma que seja fácil sua utilização de forma que possa ser utilizado por um professor em sala de aula.
- Precisa ter métodos de avaliação do modelo, sendo possível a comparação de desempenho com os modelos implementados utilizando *datasets* clássicos da área e a assim mantendo o foco do pesquisador no desenvolvimento e aprimoramento dos modelos.

1.3 Organização do Trabalho

Este trabalho aborda o desenvolvimento do *framework* Recsys.jl ele foi organizado da seguinte forma:

- Capitulo 2: Será vista toda fundamentação teórica exigida para o desenvolvimento do trabalho;
- Capitulo 3: Será visto os projetos relacionados a este, a proposta do Recsys.jl, como ele foi modelado e suas características.
- Capitulo 4: Terá comparação do *framework* Recsys.jl com outros *frameworks* tendo como base de comparação, a forma do desenvolvimento de uma nova técnica, o uso de uma técnica e o desempenho dessa técnica.
- Capitulo 5: Conclusão e trabalhos futuros.

Capítulo 2

Sistema de Recomendação

Hoje nós vivemos em uma sociedade conectada de forma global e em tempo real, na qual produzimos informações como nunca. Em toda a história da humanidade foram produzidos pelo menos 32 milhões de livros, 750 milhões de artigos, 25 milhões de músicas, 500 milhões de imagens, 500 mil filmes, 3 milhões de vídeos e programas de TV, e cerca de 100 bilhões de páginas na web, sendo que, a maior parte dessa produção é referente aos últimos 50 anos de nossa história[37].

A cada ano cerca de 1,5 bilhões de Gigabytes de informações são produzidos e disponibilizados na Internet [2]. Esses valores inviabilizam a avaliação individual de cada informação na web. Por causa desse problema, surge na década de 90 a área de sistemas de recomendação, e esta tem como motivação ajudar usuários a lidar com uma enorme quantidade de dados, de forma a gerar recomendações personalizadas de conteúdo e/ou de serviços. Diversas empresas têm adotado esse tipo de tecnologia como exemplo, a Amazon[1] com livros, Netflix[2] com filmes e a Google[3] com notícias .

O primeiro Sistema de Recomendação, O Tapestry foi desenvolvido pela Xerox Palo Alto no início dos anos 90 e tem como objetivo a seleção de e-mail [13]. Os usuários criam, através de uma interface, regras (filtros) que relacionam suas prefe-

[1]http://www.amazon.com
[2]http://www.netflix.com
[3]https://news.google.com

rências. Através desses filtros os e-mails são selecionados e somente serão enviados para aqueles usuários que estiverem de acordo ao respectivo filtro. Os criadores do sistema Tapestry, Utilizaram a frase "filtragem colaborativa"visando designar um tipo de sistema específico no qual a filtragem de informação era realizada com o auxílio humano, ou seja, pela colaboração entre os grupos de interessados, mas o autor [30] preferiu utilizar o termo "sistema de recomendação"mais geral por duas razões. A primeira é que os recomendadores podem não explicitamente colaborar com os beneficiários, e a segunda razão é que as recomendações podem sugerir itens particularmente interessantes, além de indicar aqueles que devem ser filtrados.

Os sistemas de recomendação auxiliam no aumento da capacidade e eficácia deste processo de indicação já bastante conhecido na relação social entre seres humanos [30]. Em um sistema típico as pessoas fornecem recomendações como entradas e o sistema agrega e direciona para os indivíduos considerados potenciais interessados neste tipo de recomendação. Segundo [7], Eles utilizam técnicas de análise de dados, afim auxiliar os usuários a encontrar itens que eles possam desejar. Isso pode ser relacionado no dia a dia como um atendente de uma loja, no qual para facilitar a escolha do cliente e aumentar a as suas vendes, ele realiza um processo natural de filtragem e recomendação na loja reduzindo muito a quantidade de produtos a ser visto pelo cliente, sendo que os produtos selecionados terão maior probabilidade ser de interesse do cliente.

As formas que os sistemas de recomendação captam informações sobre as preferências de usuários para um conjunto de itens são de forma explicita, isto é quando o usuário índica espontaneamente o que lhe interessa, ou implicitamente que normalmente é feito com monitoramento do comportamento dos usuários (músicas escutada, aplicativos baixados, sites visitados, livros lidos ...) [3].

Segundo [2], podemos formalizar o funcionamento de um sistema de recomendação como: A partir de um conjunto U de usuário de um determinado sistema e um conjunto I de itens que podem ser recomendados. Procura-se um item i que pertence ao conjunto I que máxime a função *utilitaria* que representa a utilidade de um item i para o usuário u. Isto pode expressado pela equação abaixo:

$$c \in C,\ i \in I,\ i_c = argmax(c, i) \tag{2.1}$$

Normalmente a função que mede a utilidade é arbitraria, podendo assumir diversas formas. No exemplo dado na Tabela 2, a função de utilidade é representada por uma nota que varia de 1 a 5, sendo 1 nota que representa o menor grau de utilidade e 5 o maior grau de utilidade.

	Matrix	Harry Potter	Peter Pan	Star Wars
João	5	3	2	5
Maria	4	2	2	3
José	2	5	5	2
Ana	1	3	4	3

Tabela 2.1: Tabela com notas de filmes valendo de 1 a 5 dados por um conjunto de 5 usuários

Normalmente a função de utilidade é definida através de uma nota, e elas são definidas somente para um itens previamente avaliado pelo o usuário, tornando necessário que o sistema de recomendação tenha a capacidade de predizer uma nota de um usuário para um determinado item para ele poder fazer recomendações apropriadas. Essas predições são feitas com diversos métodos e partir desses métodos podemos classificar o sistema de recomendação. Essa recomendação pode ser divida em 6 categorias: [31]

- Baseado em Conteúdo: Utiliza os itens avaliados pelo usuário com o objetivo de identificar características comuns entre eles e assim determinar o conjunto de interesses ou perfil do usuário.
- Filtragem Colaborativa: Utiliza as avaliações de usuários com gostos similares ao dele para gerar uma recomendação. Essa técnica será utilizada no trabalho e sera detalhada na próxima sessão.

- Demográfico: Sistema de Recomendação que utiliza informações demográficas do perfil do usuário para suas recomendações
- Baseado em Conhecimento: Utiliza a preferências dos usuários e o conhecimento sobre o domínio, inferindo as necessidades do usuário.
- Baseado em Comunidade: Utiliza as informações sobre relações entre os usuários para as suas recomendações.
- Métodos Híbridos: Estes métodos combinam duas ou mais técnicas, a principal função da hibridização das técnicas é criar um algoritmo final que contorne as limitações das técnicas individuais.

2.1 Filtragem Colaborativa

2.1.1 Introdução

No início de 1990, a filtragem colaborativa começou a surgir como uma solução para lidar com a enorme carga de *e-mails* que os usuários recebem, o primeiro sistema a adotar filtragem colaborativa, o Tapestry era um sistema de filtragem colaborativa manual, ou seja usuários precisavam criar *queries* em uma linguagem especificamente projetada definindo como o sistema se comportaria, isso acarretava que os usuários precisavam conhecer as outras pessoas para saber quem tinha gostos semelhantes aos seus pois eram eles definiam sua "similaridade"em relação a um outro usuário [13]. O primeiro sistema de filtragem colaborativa automática foi introduzido pelo GroupLens [23, 29], com o objetivo de predizer de forma personalizada, o interesse dos usuários sobre os artigos postados no *Usenet netnews*.

A filtragem colaborativa constitui-se em uma das mais populares técnicas de recomendação, sendo utilizada em muitos sistemas existentes na internet [33]. A técnica se baseia na análise de preferências comuns em um grupo de pessoas. Esses sistemas tentam predizer o interesse de um determinado usuário usando como base, as opiniões de outras pessoas que compartilham de interesses semelhantes. A essência desta técnica está na troca de experiências entre as pessoas que possuem

interesses comuns e possuem "gostos" semelhantes.

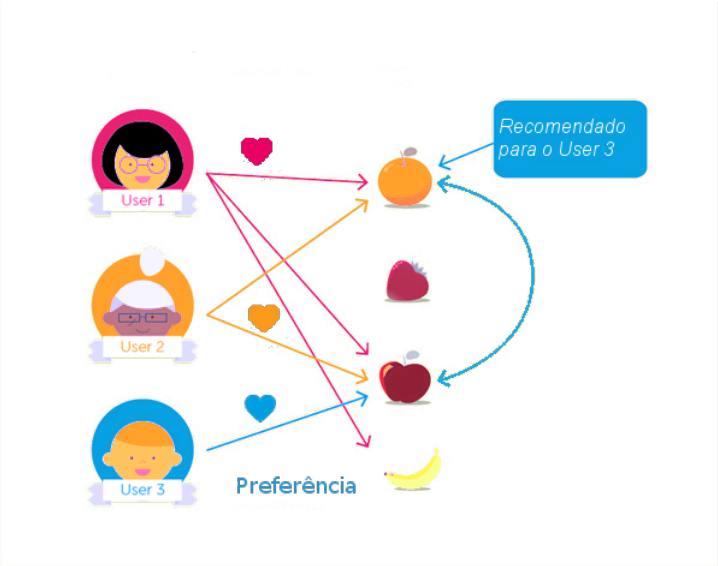
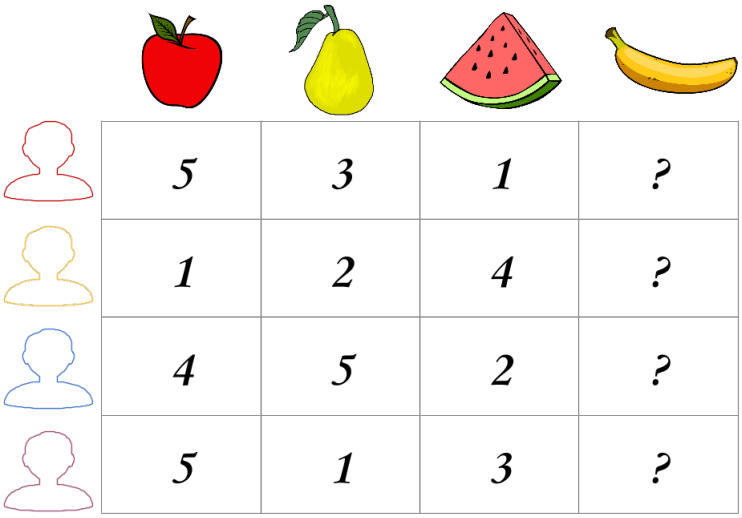


Figura 2.1: Exemplificação de uma recomendação em filtragem colaborativa [15].

Apesar da filtragem colaborativa possuir diversos benefícios sobre a baseada em conteúdo, ela sofre com algumas limitações. Um difícil embora comum desafio para filtragem colaborativa, é o problema de *Cold Start*, ele ocorre quando não é possível fazer recomendações confiáveis devido a uma falta inicial de avaliações. Podemos distinguir três tipos de problemas de *Cold Start*: Inicio da utilização do Sistema de Recomendação, o da adição de um novo item e de um novo usuário.

O problemas de inicio da utilização do Sistema de Recomendação, refere-se a uma quantidade não suficiente de dados (*ratings*) que permitam que recomendações confiáveis possam ser feitas, ou seja não se tem avaliações feitas por usuários.

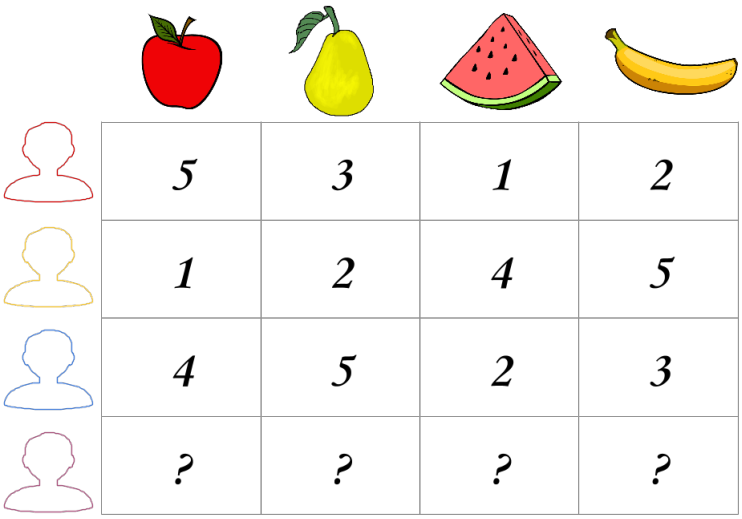
O problema de um novo item surge devido ao fato de que os novos itens introduzidos em um sistema de recomendação, geralmente não têm avaliações já feitas, portanto, eles não serão recomendados [26]. Por sua vez, um item que não é recomendado passa despercebido por grande parte dos usuários do sistema e como eles não têm conhecimento de aquele item pode ser interessante para ele, ele não irá utiliza-lo e avalia-lo. Desta forma, podemos entrar em um ciclo vicioso no qual um conjunto de itens do sistema de recomendação são deixados de fora do processo de recomendações.



5	3	1	?
1	2	4	?
4	5	2	?
5	1	3	?

Figura 2.2: O problema do novo Item [24].

O problema de um novo item tem um menor impacto no sistema de recomendação em relação ao problema de quando se inicia o sistema de recomendação porque os itens podem ser descobertos através de outros meios, como por exemplo, uma divulgação que esse item foi adicionado no sistema.



5	3	1	2
1	2	4	5
4	5	2	3
?	?	?	?

Figura 2.3: O problema do novo Usuário [24].

O problema do novo usuário é uma das grandes dificuldades enfrentadas pelos sistemas de recomendação em funcionamento. Quando os usuários cadastram-se e

ainda não lançaram nenhum voto, eles não podem receber quaisquer recomendações personalizadas com base na Filtragem Colaborativa. Umas das formas comuns de resolver o problema do novo usuário é incentivar ele a dar notas a possíveis itens já conhecidos por ele ou o sistema utilizar usuários no qual já tenham avaliado um numero mínimo de itens.

Um outro desafio que algorítimos de filtragem colaborativa tem que enfrentar é a esparsidade, que foi definida pelo [4] como a baixa quantidade de avaliações em relação a quantidade de usuários e itens. Sistemas de recomendação comerciais, mesmo os usuários ativos utilizam menos de 1%, 2% do total de itens. Um exemplo disso é a base do Netflix completa contem 100.480.507 avaliações de 480.189 usuários, referentes a 17.770 filmes. Tal matriz possui cerca de 8,5 bilhões de células, ou seja 98% dessa matriz é esparsa, isto é, só 2% células realmente contem avaliações.

A escalabilidade é um desafio não só dos algorítimos de Filtragem colaborativa como da maioria do algoritmos de sistema de recomendação. A escalabilidade que segundo [4] é um conceito que implica na capacidade de um sistema suportar um número cada vez maior de elementos. Na filtragem Colaborativa nós temos graves problemas de escalabilidade quando o volume de usuários, itens e avaliações são muito grandes, pois os sistemas tem quer fazem o cálculo da vizinhança para cada cálculo de predição, isso pode gerar um tempo de resposta inaceitável.

Esses desafios podem afetar de maneiras distinta os algoritmos baseados em filtragem colaborativa por causa das diferentes formas desses algorítimos. Esses algoritmos podem ser agrupados em duas classes gerais:[1] algoritmos baseados em memoria e os algoritmos baseados em modelo, eles serão explicados detalhadamente nas próximas seções.

2.1.2 Baseado em Memoria

Algorítimos baseados em memoria ou K Vizinhos mais próximos(*K-Nearest Neighbors*) ou simplesmente K-NN é um algorítimo que permite realizar previsões de valores baseando em valores passados de K elementos mais similares dando um peso

a sua similaridade [2]. O sucesso deste método depende, entre outros fatores, da escolha dos pesos que cada vizinho contribuirá para a predição das avaliações desconhecidas, esse peso é dado a partir de um calculo de quão similar é o vizinho de um determinado elemento, esse calculo de similaridade será explicado nas próximas seções.

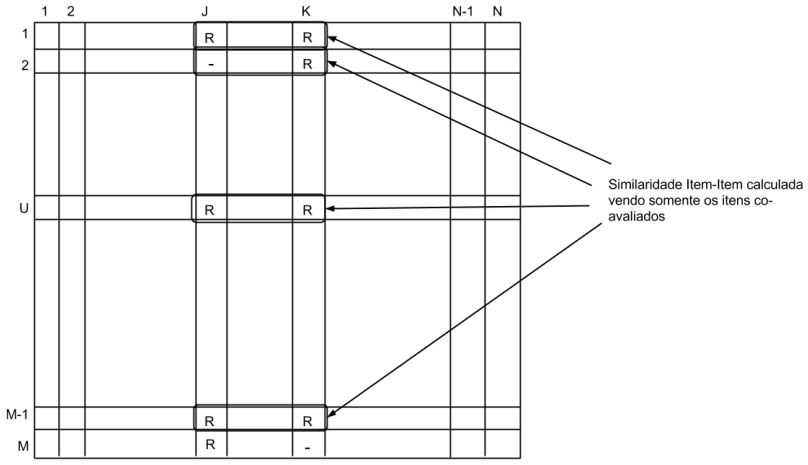


Figura 2.4: Exemplo de itens que foram avaliados em comum [32].

Existem duas formas de realizar a predição usando algorítimos baseados em memoria, essas formas alteram como o agrupamento é realizado. A primeira forma é recomendação baseadas no usuário, essa forma tenta prever uma preferência do usuário u para um item i utilizando a preferência de n usuários similares.

$$P_{ui} = \frac{\sum_{u' \epsilon V} similaridade(u', u) N_{u'i}}{\sum_{u' \epsilon V} similaridade(u', u)} \tag{2.2}$$

A predição P_{ui} ,tal que u é usuário alvo e i é o item alvo é dado por uma media ponderada entre o somatório de $similaridade(u, u')$, onde u é usuário alvo e u' é vizinho ϵ k vizinhos mais próximos do usuário alvo, multiplicado pela preferencia $N_{u'i}$

A segunda é a recomendação baseado no item, essa forma tenta prever qual será a preferência de um item i para usuário u a partir de n itens similares.

$$P_{iu} = \frac{\sum_{i' \epsilon V} similaridade(i', i) N_{i'u}}{\sum_{u' \epsilon V} similaridade(i', i)} \tag{2.3}$$

A predição P_{iu} ,tal que u é usuário alvo e i é o item alvo é dado por uma media ponderada entre o somatório de similaridade $Similaridade(i, i')$, onde u é item alvo e i' é vizinho ϵ k vizinhos mais próximos do item alvo, multiplicado pela preferencia $N_{i'u}$

2.1.2.1 Calculo de similaridade

O principal fator na recomendação por vizinhos mais próximos é a similaridade. Ela é a responsável pela seleção dos vizinhos mais confiáveis que serão utilizados para a previsão e além de fornecer o grau de importância desses vizinhos, ou seja a avaliação de um vizinho mais confiável irá valer mais que a avaliação dos outros vizinhos. Devido a esses fatores, a escolha do método de similaridade impacta diretamente na precisão e no desempenho do algoritmo de recomendação [31] .

A Similaridade utilizando o cosseno, se baseia na representação de objetos y e y' na forma de vetores e a partir disso a similaridade seria o ângulo formado por eles:

$$cos(y, y') = \frac{y^t y'}{\|y\| \, \|y'\|} \tag{2.4}$$

Outra técnica comumente utilizada para o calculo de similaridade é a Correlação de Pearson, a origem desse coeficiente remonta o trabalho de karl Pearson e Francis Galton [9]. Segundo [12] correlação é "uma medida de associação bivariada (força) do grau de relacionamento entre duas variáveis". Sua representação pode ser vista na formula 2.5, sendo que y, y' são os vetores dos elementos a serem comparados e $\bar{y}$ que é a media desses elementos.

$$correlacaoPearson(y, y') = \frac{\sum (y - \bar{y})(y' - \bar{y'})}{\sqrt{(y - \bar{y})^2 (y' - \bar{y'})^2}} \tag{2.5}$$

2.1.3 Baseado em Modelo

Diferente da filtragem colaborativa baseada em memoria que utiliza diretamente as notas dadas pelos usuários para realizar a predição de uma nota por um usuário u para o item i, a filtragem colaborativa baseado em modelo tenta criar um modelo

matemático para predição. Técnicas de aprendizado de máquina ou algoritmos de mineração de dados normalmente são utilizados para a criação de um modelo que consiga identificar os padrões existentes e traduza em um modelo que simule o comportamento dos usuários.

A recomendação baseada em modelo tem sido pesquisada para tentar resolver alguns problemas decorrentes da filtragem colaborativa, como o problema de esparsidade. Do ponto de vista probabilístico, a tarefa de filtragem colaborativa pode ser visto como o cálculo do valor esperado de um voto, dado o que sabemos sobre o usuário [5]. Para o usuário ativo, queremos prever votos em itens ainda não observadas. Se assumirmos que os votos são um valor inteiro com um intervalo para 0 a m, temos:

$$P_{ui} = \sum j * P(v_{ua}, i = j | v_{u_a}, k, k \in I_{ua})) \tag{2.6}$$

em que a expressão é a probabilidade de que o usuário ativo terá um valor para o item j dados as notas úteis anteriormente observados. Dois modelos probabilísticos alternativos para filtragem colaborativa são os modelos de Bayesiano e regressão.

Um modelo Bayesiano é utilizado para agrupar os usuários em grupos que compartilham a mesma distribuição de notas. Os parâmetros do modelo são estimados utilizando a Cadeia de Markov. Recentemente, foram propostos alguns trabalhos envolvendo métodos probabilísticos mais sofisticados, como Processo de Decisão Markoviano (PDM) e Análise Semântica Latente Probabilística [18].

Já os algoritmos baseados em modelos de regressão [36], eles tentam aproximar a nota utilizando este tipo de modelo. Considerando que $X = (X_1, X_2, ..., X_u)$ são variáveis que representam as preferências de um usuário sobre determinados itens. Sendo Λ é uma matriz $u * k$, $N = (N_1, N_2, N_3, ...N_u)$ é uma variável qualquer que representa o ruído das escolhas dos usuários e R é $n * i$ onde R_{ui} é a nota de um usuário u para um item i. Podemos representar o modelo de regressão na Equação abaixo.

$$R = \Lambda * X + N \tag{2.7}$$

2.1.3.1 Modelo de Regressão usando SVD

Decomposição por valor singular (Singular Value Decomposition) comumente chamada de SVD é uma técnica algébrica de fatoração de matrizes. O SVD analisa a matriz em busca de correlações e agrupa os dados correlacionados [28] .

O autor [10] utilizou o SVD para propor uma simplificação do modelo de regressão e a utilização do SVD para construção do modelo. Com isso, surgiu uma classe de algoritmos baseada em modelos que utiliza o conceito de variáveis latentes para a construção de um modelo para recomendação [38, 10].

Modelos que utilizam os fatores latentes são abordagens que tentam explicar a preferência de um item para um usuário caracterizando os mesmos em fatores. Um exemplo disso é no caso de um sistema de recomendação de filmes, nós podemos interpretar os valores latentes de um filme o tão próximo ele é de uma determinada categoria. Já no caso do usuário seria a preferência deste sobre estas categorias.

No trabalho do autor [10], eles utiliza a técnica de Decomposição em Valores Singulares para a inicialização do modelo, com a intenção de gerar K fatores latentes. Com a fatoração temos que para cada usuário u existe um vetor p_u e para cada item i existe um vetor q_i que corresponde os fatores latentes de cada um. Nesse trabalho ele assume que a nota de um usuário u para um item i pode ser aproximada pela multiplicação dos fatores de cada um.

$$r_{ui} = q_i^T * p_u$$

Nota aproximada após a decomposição por valores singulares (2.8)

O maior desafio é que a matriz de notas é esparsa e com isso existem diversos valores faltantes na realização da decomposição. Uma forma de superar esse desafio é após a decomposição, a realização de um procedimento de aprendizado. No trabalho [10] foi utilizado o algoritmo do gradiente descendente, com o objetivo de minimizar o erro quadrático da soma. Simon Funk propôs-se a utilização da constante λ a

regularização da função.

$$\min_{q_i,p_u} \sum_{(u,i)} (r_{ui} - q_i^T P_u)^2 + \lambda(\|q_i\|^2 + \|p_u\|^2)$$

Erro da soma quadrático utilizando o modelo sugerido por Simon Funk (2.9)

O algoritmo do gradiente descendente irá percorrer todas as avaliações do conjunto de treinamento e para cada uma, ele irá calcular o erro em relação à nota prevista . Para cada avaliação do conjunto de treinamento será feita a correção dos parâmetros q_i e p_u ao erro e_{ui}. A variável γ foi utilizada para controlar a taxa de aprendizado.

$$e_{ui} = r_u i - q_i^T * P_u$$
$$q_i \leftarrow q_i + \gamma(e_{ui} P_u - \lambda q_i)$$
$$p_u \leftarrow p_u + \gamma(e_{ui} q_i - \lambda p_u)$$

Algorítimo para a convergência do modelo proposto pelo Simon Funk (2.10)

2.1.3.2 Improved Regularized SVD

Improved Regularized SVD é uma versão Modelo de Regressão usando SVD, que segundo o [27] foi adicionado no Modelo de Regressão usando SVD dois pesos para a Nota aproximada, o parâmetro c_i para cada usuário e o parâmetro d_j para cada filme.

Os parâmetros c_i e d_j são calculados simultaneamente com os parâmetros q_i e p_u e eles são calculados utilizando a variável γ, o erro e_{ui}, a constante λ e o parâmetro $media\_global$ que é nota media do dataset utilizado.

$$r_{ui} = d_j + c_i + q_i^T * p_u$$
$$c_i \leftarrow q_i + \gamma(e_{ui} - \lambda(c_i + d_j - media\_global))$$
$$d_j \leftarrow p_u + \gamma(e_{ui} - \lambda(c_i + d_j - media\_global))$$

Nota aproximada para o *Improved Regularized SVD* e calculo dos parâmetros c_i e d_j (2.11)

Capítulo 3

Recsys.jl

3.1 Motivação

Com a propagação do *e-commerce* em geral, os sistemas de recomendação tornaram-se um tema extremamente atrativo. Através de algoritmos facilmente integráveis a aplicações web, eles agregam valor ao negócio online, promovendo itens de consumo direcionados a um público alvo. Mas ao mesmo tempo, nós temos dificuldade em encontrar disponível implementações dos algoritmos clássicos da literatura, fazendo que pesquisadores precisem implementar esses algoritmos para utilizá-los em suas pesquisas, isso acaba que uma parte significativa do tempo de pesquisa acabe sendo utilizado para desenvolvimento de algo que já exista.

Com o ideal de um *framework* que pudesse ser modulável para facilitar o reuso, tornando a pesquisa de novas técnicas mais ágil e que o código gerado no desenvolvimento nessas técnicas tivessem um alta legibilidade, o desenvolvimento do *framework* foi primeiramente focado na modelagem, e como deveria ser o desenvolvimento das técnicas no *framework*.

A partir disso foi pesquisado uma linguagem que melhor se encaixava com o projeto sendo que deveria ser uma linguagem com desempenho alto, com um nível relativamente alto de abstração e mais próximo possível da linguagem humana. Utilizando esses critérios, foi escolhida a linguagem Julia, uma linguagem cientifica

com um nível alto abstração, linguagem muito próxima da linguagem humana e com desempenho comparável a linguagens de mais baixo nível como o C.

3.2 Trabalhos Relacionados

3.2.1 MyMediaLite

MyMediaLite[1] é uma biblioteca rápida e escalável de sistema de recomendação. Ele aborda dois cenários comuns em filtragem colaborativa: previsão e *ranking*. [11]

A biblioteca oferece algoritmos no estado da arte para essas duas tarefas. A API também contém métodos para atualizações em tempo real de carga e armazenamento dos modelos de recomendação já treinados. MyMediaLite é *open-source*, distribuído nos termos da *GNU General Public License* (GPL) e foi desenvolvida em C. [11]

3.2.2 Lenskit

LensKit[2] é uma plataforma extensível para pesquisas sistemas de recomendação, com ele é possível a experimentação de novas técnicas, utilizando diversas métricas de avaliação e conjuntos de dados comparando com algoritmos existentes sem necessidade de reimplementar-los. Uma plataforma extensível para pesquisas sistemas de recomendação.

Modularidade do LensKit é particularmente notável para a comunidade de pesquisa. Suas implementações de algoritmos são projetados de modo que os componentes individuais (Ex: funções de similaridade) podem ser substituídos sempre que razoável, permitindo melhorias incrementais. Sua plataforma de avaliação também fornece um ambiente consistente para avaliar algoritmos de recomendação de uma maneira incremental. [8]

LensKit fornece implementações dos principais algoritmos filtragem colaborativas. As vantagens do Lenskit são:

[1] http://www.mymedialite.net/

[2] http://lenskit.org/

- a pesquisa de experiência do usuário pode ser incorporada a algoritmos existentes para novos ambientes com um mínimo de codificação

- LensKit é escrito em Java, por isso, é acessível a uma ampla gama de programadores e de fácil utilização em muitos ambientes, e podendo ter algorítimos escritos em Groovy, facilitando o uso e a validação da biblioteca.

3.2.3 Mahout

Apache Mahout[3] é um projeto *open source* da Apache Software Foundation, com o objetivo principal de criar algoritmos de aprendizado de máquina escaláveis sob a licença Apache. O Mahout contém implementações de *clustering*, categorização, CF e programação evolutiva. Além disso, ele usa a biblioteca Apache Hadoop para permitir Mahout escalar na nuvem. [19]

O projeto Mahout foi criado por várias pessoas envolvidas na comunidade Apache Lucene, com um interesse no aprendizado de máquinas e um desejo de ter implementações escaláveis de algoritmos de aprendizado de máquina para agrupamento e categorização, com uma ótima documentação. [19]

O projeto Mahout tem como objetivos:

- Construído e mantido pela a comunidade de usuários e colaboradores de modo a que o código é desenvolvido por colaborador em particular, uma empresa ou universidade.

- Concentra-se no mundo real, o projeto é desenvolvido focando casos de uso prático.

- Sempre fornecer documentação de qualidade e exemplos.

[3]http://mahout.apache.org/

3.2.4 LibRec

LibRec[4] é software Java licenciado sob GPL 2 (versão 1.7 ou superior necessário). Ele facilita o estudo dos dois problemas clássicos de sistemas de recomendação, ou seja, previsão e *ranking* de itens. O LibRec consiste em três componentes principais, ou seja, interfaces genéricas, datasets e algoritmos de recomendação. [11]

- uma plataforma para que outros possam contribuir mais códigos-fonte de outros algoritmos como uma biblioteca de código aberto.
- Multi-plataforma: como um software Java, LibRec pode ser facilmente implementado e executado em qualquer plataforma.
- Para se alterar as configurações do LibRec é preciso somente utilizar o a arquivo de configuração: librec.conf.
- LibRec fornece um conjunto de interfaces de recomendação bem desenhados pelo qual novos algoritmos podem ser facilmente implementadas.

3.2.5 Conclusão

O MyMediaLite e o Mahout Taste, como visto anteriormente tem o foco ferramentas é aplicar os algoritmos básicos em grande bases de dados ou em um sistema em produção. Eles acabam não são adequados tanto no contexto de pesquisa quanto na educação. Já o Lenskit e LibRec, eles são *frameworks* modulares voltado para ensino e pesquisa, tendo objetivos próximos ao do Recsys.jl, apesar deles terem sido implementados em Java, que apesar de ser uma linguagem bem desenvolvida com boa adoção, ela não tem foco cientifico. Cientistas em geral preferem utilizar linguagens dinâmicas de alto nível para o desenvolvimento de seus algorítimos, por exemplo MATLAB, Octave, R, Julia.

[4] http://www.librec.net/

3.3 Linguagem Julia

Computação científica tem requerido, tradicionalmente, alta performance embora muitas das vezes códigos para essa área tem sido desenvolvidos utilizando linguagens de alto-nível que normalmente tem um performance inferior a uma linguagem mais baixo nível. Foi partir dessa lacuna que foi iniciado a Ideia do Desenvolvimento da Linguagem Júlia.

Julia teve a sua primeira versão (0.1) lançada em fevereiro de 2012 e tem como seus principais desenvolvedores o Jeff Bezanson, Stefan Karpinski, Viral Shah, e Alan Edelman. Esse projeto foi desenvolvido Instituto de Tecnologia de Massachusetts (MIT). [34] Hoje Júlia esta na versão 0.42. [6]

Segundos os desenvolvedores Jeff Bezanson, Stefan Karpinski, Viral Shah eles queriam uma linguagem que fosse *open source*, com uma licença liberal. Queriam a velocidade do C com o dinamismo do Ruby. Eles queriam uma linguagem que tivesse macros de verdade como o Lisp, mas com notação matemática óbvia e familiar como o Matlab. Eles queriam algo tão útil para a programação em geral como o Python, tão fácil para estatística como o R, tão natural para o processamento de *string* como o Perl e tão poderoso para a álgebra linear como o Matlab. Algo que seja bem simples de aprender. Eles queriam que fosse seja interativa e facilmente compilada. [21]

3.3.1 Características do Julia

Julia foi desenvolvida sobre a licença MIT, que é uma licença que da ao usuário o uso gratuito do software, o uso para fins comerciais, sua distribuição e a sua modificação, você pode conceder uma sublicença para modificar e distribuir o software a terceiros não incluídos nessa licença. O software é fornecido sem nenhuma garantia sendo então autor não responsabilizada por uma eventual perda ou dano. [20]

Para aumentar a produtividade dos programadores e a portabilidade, cadê vez mais são criadas linguagens de programação de alto nível. Suas características principais são que seu código é mais elaborado, contemplando operações mais complexas

e mais próximas da "lógica humana", o Júlia foi desenvolvido utilizando essas características de uma linguagem de alto nível.

A sintaxe de Julia é muito próxima a do Matlab e consequentemente facilita o aprendizado dos programadores que já utilizam linguagens matemáticas. Enquanto Matlab é bem eficiente para experimentações e explorações de álgebra linear, ele possui limitações para diversas tarefas computacionais fora deste campo relativamente pequeno. Julia mantem a facilidade e expressividade do Matlab para computação numérica de alto nível, mas ultrapassa as limitações comparadas a uma linguagem de programação de propósito geral. [25]

3.4 O Projeto Recsys.jl

3.4.1 Modelagem

Para o desenvolvimento do projeto Recsys de forma simples desenvolvimento e entendimento, foram definidos um conjunto interfaces abstratas para que a partir delas possam ser implementados algoritmos das diversas funcionalidades. Essas interfaces são a Model, Experiment e Dataset.

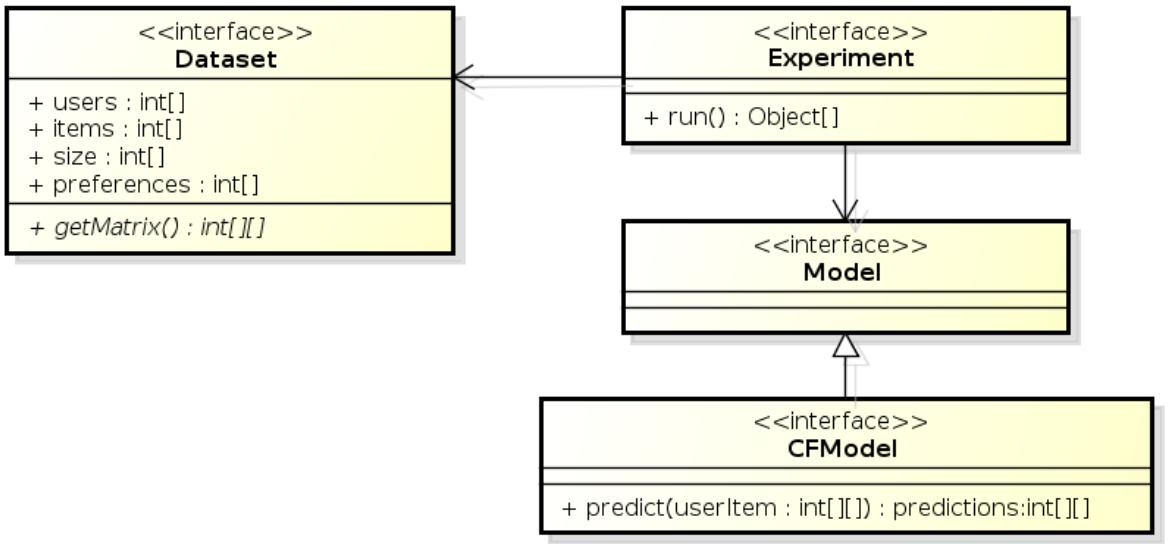


Figura 3.1: Diagrama Exibindo as interfaces do projeto

3.4.1.1 Model

A interface Model, tem o papel de ser a interface da classe na qual se implementa o algorítimo de sistema de recomendação. A interface Model criada especificamente para filtragem colaborativa foi chamada de CFModel, essa interface possui a função predict, na qual gera as predições e as retorna. O método predict tem como entrada os usuários e itens que desejam fazer a predição, essa entrada esta na forma de uma matriz na qual a primeira coluna é definida para os usuário e a segunda para os itens.

Código 3.1: Exemplo de criação de um CFModel

```
1  type NewModel <: CFModel
2
3    predict::Function
4
5    function NewModel(data::Dataset)
6      this = new()
7
8      this.predict = function(data::Array)
9        return zeros(size(data)[1])
10     end
11
12     return this
13   end
14 end
```

3.4.1.2 Experiment

A interface Experiment, como próprio nome já diz, ela faz o papel de ser o experimento, é a interface das classes que implementam uma forma validação do Model utilizando um Dataset qualquer. O método run da interface tem por objetivo a execução do experimento retornando os resultados das métricas utilizadas no experimento.

Código 3.2: Exemplo de criação de um Experimentl

```
1  type ExperimentExample <:Experiment
2
3    run::Function
4
5    function ExperimentExample(dataset::Dataset = Dataset())
6
```

```
7       this = new ()
8       this.run = function()
9         return zeros(10)
10      end
11
12      return this
13    end
14
15  end
```

3.4.1.3 Dataset

A interface Dataset tem o papel de representar a base de dados, ela recupera os dados e se preciso formata os dados para que possa ser utilizada no experimento. ela tem uma função na qual retorna uma matriz onde as colunas são: usuário, item, nota. Os atributos do dataset são *array* de usuários, *array* de itens, *array* de notas e um *array* com tamanho dos *array* de usuários, itens e notas.

Código 3.3: Exemplo de criação de um Dataset

```
1
2   type Dataset
3
4     users::Int32
5     items::Int32
6     size::Int32
7     preferences::Array
8
9     getMatrix::Function
10
11    function Dataset(filename::String)
12      this = new ()
13
14      this.file = readtable(filename, separator = '\t', header = false)
15
16      names!(this.file, [:user, :item, :rating, :time])
17
18      this.users = maximum(this.file[:user])
19
20      this.items = maximum(this.file[:item])
21
22      this.size = size(this.file)[1]
23
24      this.preferences = sort(unique(this.file[:rating]));
25
26      this.getMatrix = function()
```

```
27          return createMatrix(this);
28      end
29
30      return this
31  end
```

3.4.2 Característica do Desenvolvimento

O *framework* Recsys.jl foi desenvolvido embasado nas interfaces citadas na seção anterior, a partir de cada interface surgiram implementações de diferentes tipos que serão detalhados nessa sessão, também serão detalhadas outras características do projeto como as métricas de avaliação .

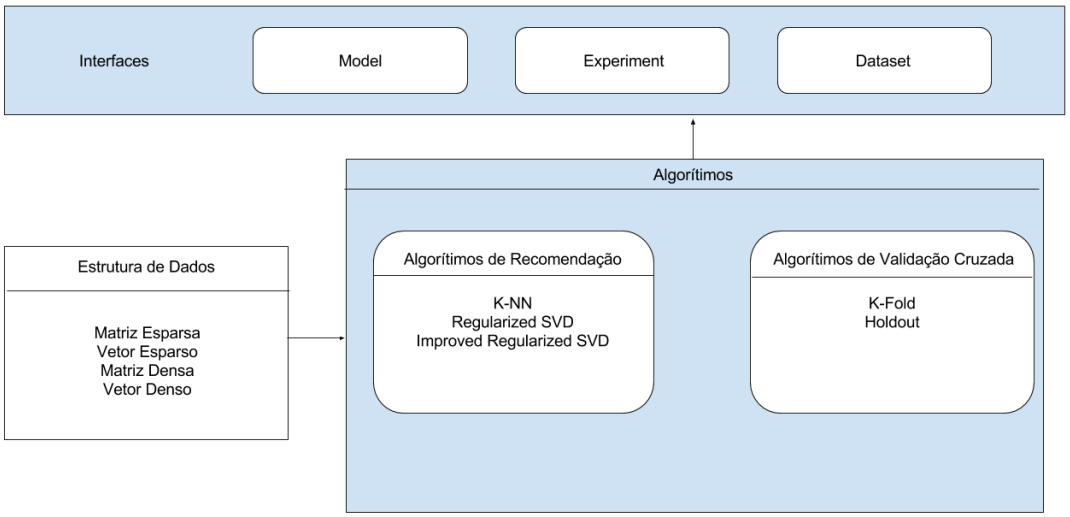


Figura 3.2: Elementos do Recsys.jl

3.4.2.1 Tipos desenvolvidos a partir do Experiment

A partir da experiment, foi implementada duas técnicas de validação cruzada, que é um forma de avaliar a capacidade de generalização de um modelo, a partir de um conjunto de dados. As formas mais simples consistem no particionamento do conjunto de dados em duas subamostras. [22] Essas duas técnicas são:

- K Fold: Consiste em dividir aleatoriamente o conjunto de dados em k subconjuntos mutuamente exclusivos e de tamanho igual(um subconjunto será chamado de fold). Então, o fold é utilizado para a validação e os k - 1 folds restantes serão utilizados na calibração do modelo. Esse procedimento deve ser repetido k vezes até que todos os folds tenham sido utilizados para a validação. [22]

- HoldOut: Consiste em particionar os dados em dois subconjuntos, mutuamente exclusivos, de forma que um será usado para o treinamento e o outro para a validação do modelo. [22]

O método KFold tem como entrada do Construtor o numero de Folds e o dataset que será utilizado. Caso não seja passado os valores para o Dataset e o numero de folds, ele assumira o valor $folds = 8$ e utilizará um Dataset do Movielens de 100K avaliações que será melhor detalhado na seção 4.2.1.

O método Holdout tem como entrada do Construtor qual a porcentagem que será utilizada para treino e qual Dataset será utilizado. Caso não seja passado os valores para o Dataset e o numero da porcentagem, ele assumirá o valor 0.8 para a porcentagem e utilizará um Dataset do Movielens de 100K.

3.4.2.2 CFModel – K-Nearest Neighbor

Foi implementado a partir da interface CFModel, a técnica de Filtragem colaborativa em memoria K-NN, que tem como entrada no método do construtor o Dataset, o tipo do KNN (se ele é baseado no item ou no usuário), o valor do K e a qual método será utilizado para o calculo da similaridade. Caso não seja passado os valores para o tipo do KNN, K e o método de similaridade, o método irá utilizar os valores ": *user*"para o tipo do KNN, 12 para o K e *cosine* para o método de similaridade.

O KNN utiliza um outro tipo chamado de Similaridade, um tipo responsável pela criação da matriz de similaridade, no qual o construtor tem na entrada, o dataset, a quantidade de elementos do qual será a matriz de similaridade (usuário ou item) e a

métrica utilizada para o calculo de similaridade, essa métrica é um método que tem por entrada dois arrays cada um com as notas dadas dos respectivos usuários dados para os respectivos itens. Foram implementados as métricas Correlação de Pearson e Calculo de Similiaridade por Cosseno chamadas de *pearsonCorrelation* e *cosine*.

3.4.2.3 CFModel – Técnicas Baseadas em Modelo

Foi implementado a técnica de Filtragem colaborativa em Modelo de Regressão usando SVD e a Improved Regularized SVD, as técnica foram chamadas respectivamente de $RegularizedSVD$ e $ImprovedRegularizedSVD$.

Essas técnicas tem como entrada no método construtor o Dataset, o K que é o numero de *features*, γ que é taxa de aprendizagem, λ que é a constante de regularização da função e o δ é o coeficiente de erro. Caso não seja passado os valores na chamada da função, os valores que são assumidos por padrão para o $RegularizedSVD$ são: $K = 10$ $\gamma = 0.001$ $\lambda = 0.02$ $\delta = 0.1$. Já para o $ImprovedRegularizedSVD$ são: $K = 10$ $\gamma = 0.016$ $\lambda = 0.05$ $\delta = 0.001$

3.4.2.4 Métricas de avaliação de erro

Para avaliar eficiência de modelos desenvolvidos, foram implementada algumas métricas que criam um resumo das diferenças entre os valores reais (observados) e os valores previstos. Os métodos implementados foram o MAE e o RMSE. Esses métodos tem como parâmetros de entrada um *array* referente ao valores reais e um *array* referente aos valores obtidos pelo método de predição do modelo a ser avaliado.

Mean Absolute Error (Erro absoluto médio) é uma das mais populares métricas de avaliação de sistemas de recomendação. [16] A técnica do calculo do MAE envolve basicamente comparar a nota real com a nota da predição assim obtendo a diferença entre elas (essa diferença sera chamada de erro). A partir dos erros obtidos na comparação das notas reais com as notas da predição se faz uma media desses erros obtendo o índice do MAE.

$$MAE = \frac{\sum |notaReal - notaPrevista|}{quantidadeNota} \tag{3.1}$$

Root Mean Square Error (Erro médio quadrado), assim como o MAE ele utiliza erro das notas reais para a predição, a diferença do MAE para RMSE é a magnitude do erro, o RMSE penaliza o maior discrepância entre o valor real e o valor predito. A técnica do calculo do RMSE envolve basicamente obter os erros na comparação das notas reais com as notas da predição, elevar esse erro ao quadrado obter media e tirar raiz quadrada da media. [3]

$$RMSE = \sqrt{\sum \left(notaReal - notaPrevista\right)^2 / quantidadeNota} \tag{3.2}$$

Capítulo 4

Experimentos

4.1 Introdução

Nas próximas seções serão detalhados, os experimentos em relação ao desenvolvimento e uso de abordagens utilizando o *framework* Recsys.jl.

4.1.1 Uso do Recsys.jl

Para uma análise quanto a legibilidade de um código implementado no Recsys.jl utilizando uma abordagem já desenvolvida, foi feito nesta seção um comparativo com o desenvolvimento do código para a execução do KFold com o algorítimo de filtragem colaborativa em memoria (K-NN) baseado em Usuário.

No Recsys.jl foi preciso passar os parâmetros do K-NN pela entradas construtores dos objetos: Dataset e KFold. No Dataset é preciso passar o endereço do arquivo a ser utilizado no Dataset, no K-Fold foi preciso passar o K, e o objeto Dataset.

Além da passagem de parâmetros, foi implementado uma função que cria o objeto KNN nessa função foi utilizada o Construtor do KNN com o parâmetro da entrada função dataset, e o parâmetro de qual tipo do KNN a ser utilizado, o valor do K e tipo de similaridade a ser utilizada. Para a execução do experimento foi preciso passar a função de criação do KNN e qual fold a ser executado.

Código 4.1: Código do Recsys.jl

```
1  using Recsys
2
3  dataset = Recsys.Dataset("datasets/user.data")
4  experiment = Recsys.KFold(10,dataset)
5
6  function model(data)
7    return Recsys.KNN(data,:user,12,Recsys.unionCosine);
8  end
9
10 result = experiment.run(model, 1)
```

Fazendo um comparativo, no lenskit é preciso implementar script em Groovy A podendo também ser implementado em Java, que a partir do método bind, se passa as classes da abordagem e de seus parâmetros que desejam ser utilizados. Essa forma de passagem requer um conhecimento prévio de como o projeto foi desenvolvido pois é preciso saber a forma que se utiliza os parâmetros do método bind para a passagem da configuração desejada, podendo deixar o código longo e com um problema de legibilidade mesmo sendo escrito em Groovy, que é uma linguagem mas alto nível.

Na biblioteca Librec já tem implementado o método KFold. Para o uso do método KFold e K-NN é somente preciso defini-los no arquivo de configuração librec.conf B . A ideia do LibRec é ter um arquivo de configuração para escolha da abordagem a passagem dos parâmetros necessários da mesma. Isso torna o código simples de ser entendido, pois retira do código de sua aplicação as configurações do LibRec.

A facilidade que o Recsys.jl teve em relação ao fato do código ser legível e simples é porque ele foi implementado linguagem Julia que é de altíssimo nível, proporcionando o código do Recsys.jl curto e simples na hora de definir os parâmetros para execução da abordagem escolhida, como as outras bibliotecas tem seus código escritos em Java, eles escolheram duas formas de tornar o código mais simples: Utilizando uma linguagem *script* para execução da biblioteca que foi o caso do Lenskit, que utilizou o Groovy para tornar a escrita de códigos mais simples e mesmo assim a passagem de parâmetro para abordagem escolhida é feita por chamada de uma função, podendo se utilizada varias vezes, tornando o arquivo longo e difícil de ser lido, ou a utilização de um arquivo de configuração no qual foi o caso do Librec.

4.1.2 Desenvolvimento de uma nova Abordagem

Para saber dificuldade em relação do desenvolvimento de novas técnicas para o Recsys,jl em relação as bibliotecas Lenskit e LibRec, nesta seção avaliaremos quais são os passos básicos para o desenvolvimento de uma nova abordagem.

Para o desenvolvimento de uma técnica de recomendação no Recsys.jl é preciso desenvolver a partir do Model, caso seja uma técnica de Filtragem colaborativa o CFModel e implementar o seu construtor e o método predict(data).

Já para desenvolver no libRec, é preciso implementar uma classe com a interface mais adequada para a técnica a ser desenvolvida e partir dela implementar os métodos: $InitModel()$, $BuildModel()$, implementar o $predict(int\ u, int\ j, boolean\ delimited)$, e adicionar o sua nova técnica no método $getReccomender$ da classe LibRec.java. [14]

O desenvolvimento na biblioteca Lenskit, é preciso seguindo o conjunto de interfaces e classes abstratas do projeto, implementar os métodos dessas classes. Que devido a alta complexidade do projeto faz entendimento desse conjunto de interfaces e classes abstratos serem muito custosos.

A estrutura simples do Recsys.jl é um grande facilitador em relação ao Lenskit, pois apesar dos dois apresentarem uma boa modularidade, mas alta complexidade do projeto gerada por uma grande quantidade de interfaces e classes abstratas geram uma dificuldade no entendimento. Já em relação a biblioteca LibRec, ela tem uma menor modularidade em relação ao Recsys.jl pois a cada técnica que é desenvolvida, é preciso uma alteração no núcleo da biblioteca para que essa técnica ao ser lida do arquivo de configuração possa ser executada. Isso pode tornar o código menos expansível já que toda nova técnica tem que ser colocada no núcleo da ferramenta, que apesar ser um código aberto, sua alteração se mantendo a mesma ferramenta é de responsabilidade dos desenvolvedores da ferramenta. Algo que no Recsys.jl não é preciso, já que o seu núcleo não precisa ser alterado.

4.2 Desempenho do Recsys.jl

4.2.1 Metodologia utilizada

A metodologia utilizada para a avaliação de desempenho do Recsys.jl será levada em conta tempo de execução, as métricas de avaliação de erros MAE e RMSE, e as métricas de avaliação da capacidade de generalização de um modelo KFold e Holdout.

4.2.1.1 Métricas de Avaliação

Com o objetivo de validar o experimento,é necessário utilizar métricas que possibilitem a comparação das diversas abordagens de recomendação desenvolvidas (Cada abordagem dessas podem ser chamadas também de modelo). Dentre as diversas métricas de acurácia foram escolhidas duas métricas, A MAE (Mean Absolute Error) e a RMSE (Root-mean-square deviation).

4.2.1.2 Métricas de Avaliação da capacidade de generalização de um modelo

Umas das principais forma de avaliação da capacidade de generalização de um modelo para o problema de predição(filtragem colaborativa é um problema de predição), nós temos como conceito da validação cruzada o particionamento do conjunto de dados em subconjuntos mutualmente exclusivos, utilizando um ou mais subconjuntos para dados de treinamento e o restante dos subconjuntos para validação(teste). Usaremos as técnicas de validação cruzada K-Fold e Holdout que já foram explicadas nas seções anteriores.

4.2.1.3 Dataset

O Datasets utlizado para os testes foi do Movielens(coletado e disponibilizado pelo GroupLens Research). Esse dataset contêm 100.000 classificações anônimas de aproximadamente 1682 filmes, feitos por 943 usuários do MovieLens.

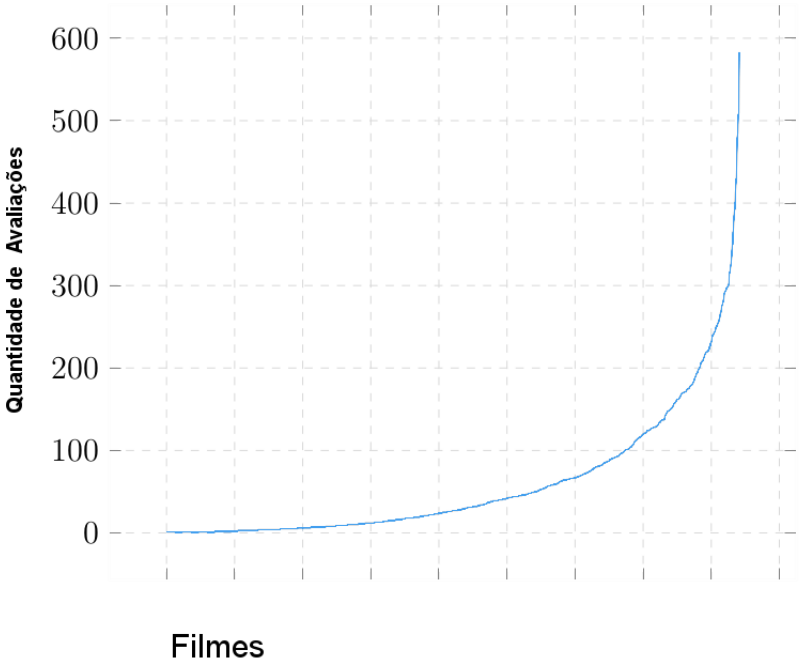


Figura 4.1: Quantidade de avaliação por Item.

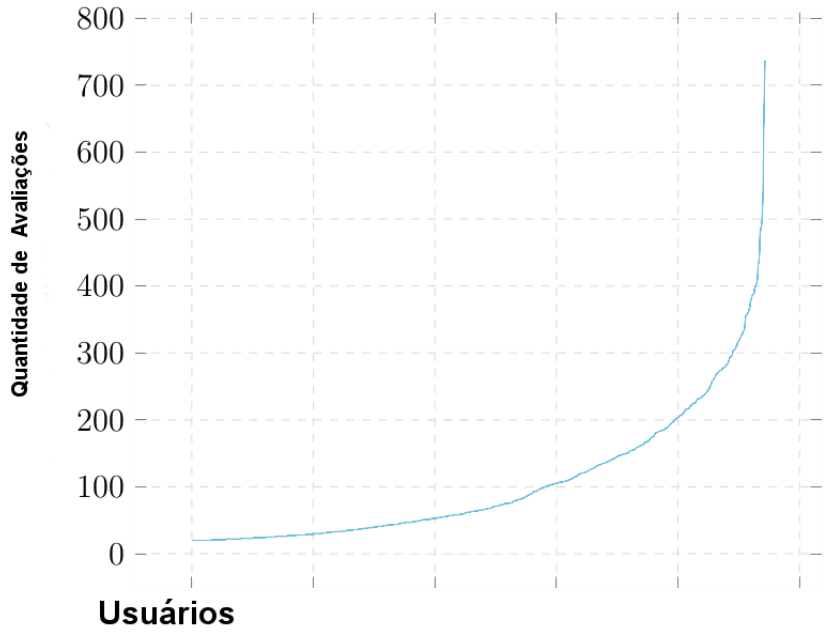


Figura 4.2: Quantidade de avaliação por usuário.

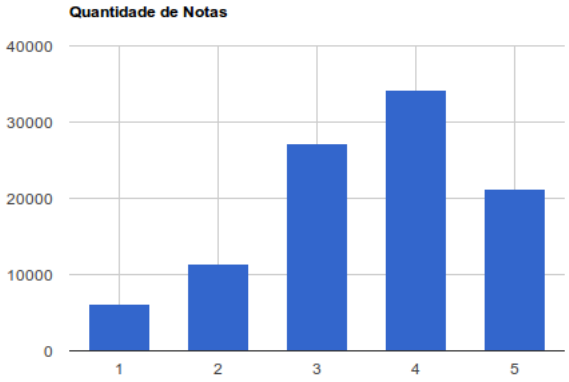


Figura 4.3: Quantidade de avaliações.

4.2.1.4 Ambiente do Experimento

Para a execução dos experimentos foi utilizado uma maquina virtual no Google Cloud Computing, sendo a configuração: 2.5 GHz Intel Xeon E5 v2, 13Gb de ram, 10Gb (Não contando com S.O), Sistema operacional Ubuntu 14.04 e foi utilizado a versão 0.42 da linguagem Julia.

4.2.2 Resultados de técnicas implementadas

Nesta seção será apresentado resultado de algumas combinações de modelos e forma de experimentação, com a intenção de mostrar desempenho das técnicas implementadas no Recsys.jl.

4.2.2.1 Holdout e K-NN

Foi executado o Holdout com o modelo K-NN com as seguintes configurações. Holdout com 80% dos dados para treinamento e 20% para validação, K-NN tem o K com valor 12.

MAE	RMSE	Tempo de Execução
0.746363	0.950648	234.385 mS

Tabela 4.1: Tabela com Resultados para o experimento Holdout K-NN

4.2.2.2 K-Fold e K-NN

Foi executado o K-Fold com o modelo K-NN com as seguintes configurações. K-Fold com 8 folds e o K-NN tem o K com valor 12.

Fold	MAE	RMSE	Tempo de Execução
1	0.735019	0.936529	234.671 mS
2	0.752900	0.951468	235.124 mS
3	0.747038	0.952080	233.753 mS
4	0.748136	0.953544	236.112 mS
5	0.743947	0.948593	234.383 mS
6	0.746639	0.952362	234.456 mS
7	0.745055	0.950646	233.959 mS
8	0.742887	0.946615	233.796 mS

Tabela 4.2: Tabela com Resultados para o experimento K-Fold K-NN

4.2.2.3 Holdout e RegularizedSVD

Foi executado o Holdout com o modelo RegularizedSVD com as seguintes configurações. Holdout com 80% dos dados para treinamento e 20% para validação e RegularizedSVD com features = 10, $\gamma = 0.001, \lambda = 0.02, \delta = 0.1$.

MAE	RMSE	Tempo de Execução
0.768125	0.988429	15.0287 mS

Tabela 4.3: Tabela com Resultados para o experimento Holdout RegularizedSVD

4.2.2.4 K-Fold e RegularizedSVD

Foi executado o K-Fold com o modelo RegularizedSVD com as seguintes configurações. K-Fold com 8 folds e RegularizedSVD com features = 10, $\gamma = 0.001, \lambda = 0.02, \delta = 0.1$.

Fold	MAE	RMSE	Tempo de Execução
1	0.760914	0.980468	13.222 mS
2	0.776480	0.997854	13.149 mS
3	0.772720	0.998646	13.281 mS
4	0.769560	0.995159	12.880 mS
5	0.765369	0.982843	13.227 mS
6	0.771531	0.992858	13.374 mS
7	0.767542	0.985521	13.351 mS
8	0.766910	0.986407	13.360 mS

Tabela 4.4: Tabela com Resultados para o experimento K-Fold RegularizedSVD

4.2.2.5 K-Fold e ImprovedRegularizedSVD

Foi executado o K-Fold com o modelo K-NN com as seguintes configurações. K-Fold com 8 folds e ImprovedRegularizedSVD com features = 10, $\gamma = 0.001, \lambda = 0.02, \delta = 0.1$.

Fold	MAE	RMSE	Tempo de Execução
1	0.767156	0.965291	3.3135 mS
2	0.754223	0.952121	3.1739 mS
3	0.761080	0.953661	3.2632 mS
4	0.763481	0.963351	3.2095 mS
5	0.763433	0.959835	3.2577 mS
6	0.754120	0.954599	3.1670 mS
7	0.765583	0.964671	3.2632 mS
8	0.752482	0.949011	3.2142 mS

Tabela 4.5: Tabela com Resultados para o experimento K-Fold ImprovedRegularizedSVD

4.2.2.6 Holdout e ImprovedRegularizedSVD

Foi executado o Holdout com o modelo ImprovedRegularizedSVD com as seguintes configurações. Holdout com 80% dos dados para treinamento e 20% para validação e ImprovedRegularizedSVD com features = 10, $\gamma = 0.001, \lambda = 0.02, \delta = 0.1$.

MAE	RMSE	Tempo de Execução
0.765144	0.961643	4.96484 mS

Tabela 4.6: Tabela com Resultados para o experimento Holdout ImprovedRegularizedSVD

4.2.2.7 K-Fold e RegularizedSVD em relação a outras bibliotecas

Foi executado o K-Fold com o modelo RegularizedSVD com as seguintes configurações. K-Fold com 8 folds e RegularizedSVD com features = 10, $\gamma = 0.001, \lambda = 0.02, \delta = 0.1$.

Biblioteca	MAE	RMSE	Tempo de Execução
LibRec	0,72152500	0,91201900	4.349 mS
Recsys.jl	0.76887825	0.98996875	13.230 mS
Lenskit	0.75023949	0.91654641	9.519 mS

Tabela 4.7: Tabela com Resultados para o experimento K-Fold Regularized para Recsys, LibRec e Lenskit

4.2.2.8 K-Fold e K-NN em relação a outras bibliotecas

Foi executado o K-Fold com o modelo K-NN com as seguintes configurações. K-Fold com 8 folds e o K-NN tem o K com valor 12.

Biblioteca	MAE	RMSE	Tempo de Execução
LibRec	0,75996700	0,97011400	3.452 mS
Recsys.jl	0.74653087	0.94897963	234.531 mS
Lenskit	0.74139667	0.94632211	2.182 mS

Tabela 4.8: Tabela com Resultados para o experimento K-Fold K-NN para Recsys, LibRec e Lenskit

4.3 Análise dos Resultados

A forma como o Recsys.jl foi modelado permitindo uma altíssima modularização do código e a linguagem Julia ser de altíssimo nível, possibilitou o *framework* ter uma alta legibilidade das técnicas desenvolvidas com a ferramenta, sem a necessidade de um arquivo de configuração que reduz significativamente sua modularidade ou uma complexa estrutura *framework*, dificultando o entendimento do mesmo tornando mais complexo desenvolvimento de novas técnicas.

Em relação a acurácia das técnicas desenvolvidas no Recsys.jl, foi utilizado como critério de avaliação as métricas MAE e RMSE. As técnicas desenvolvidas no Recsys.jl apresentaram um bom desempenho comparadas ao do Lenskit e a do Librec. Um outro comparativo foi em relação ao tempo de execução, no qual foi executados os diferentes *framework* em um único ambiente, neste comparativo o Recsys.jl se mostrou um desempenho um pouco abaixo em relação ao Lenskit e LibRec.

Apesar de desempenho em relação ao tempo de execução ser abaixo das outras ferramentas, o Recsys.jl teve um ganho considerável em relação a facilidade dos desenvolvimentos de novas técnicas com sua simples modelagem, tendo o seus objetivos primários alcançado.

Capítulo 5

Conclusões

5.1 Considerações acerca do Trabalho

Este trabalho propôs um *framework* que fosse elaborado de forma modulável, com o objetivo de quando for implementado uma nova abordagem, os códigos utilizados nessa nova abordagem e que já tenham sido implementados em outra abordagem, não precisem ser reimplementados. Este *framework* precisava também ser elaborado de forma que fosse simples a sua utilização no ensino e pesquisa, sendo possível sua utilização por um professor em sala de aula.

Para tornar Recsys Modulável, foi desenvolvido as interfaces Model, Experiment e Dataset. Essas interfaces dividiram em 3 objetos a forma que se desenvolve a técnica e se avalia a em um determinado cenário com uma determinada base de dados. A Interface Model foi desenvolvida com o objetivo de ser o local aonde será desenvolvida a técnica, a interface Experiment é de qual forma ela será avaliada e a interface Dataset qual a base de dados será utilizada.

Para o *framework* ser elaborado de forma simples de forma ser possível a sua utilização no ensino em sala de aula, foi utilizado uma modelagem simples e escolhi do a linguagem Julia para seu desenvolvimento, uma linguagem de altíssimo nível que torna o código muito simples de ser lido e curto.

5.2 Limitações e trabalhos futuros

Experimentos mostram que apesar de bons resultados, o desempenho em relação ao tempo pode ser melhorado, uma forma de ser feito isso, é a paralelização das técnicas desenvolvidas já que a ferramenta como o LibRec, já utiliza técnicas já paralelizadas.

O desenvolvimento deste trabalho foi feito utilizando a versão 0.3 da Linguagem Julia, mas a linguagem Julia lançou recentemente a versão 0.4 e com ela veio algumas mudanças em métodos utilizados pela ferramenta, sendo preciso a alteração nos métodos que foram marcados como *deprecated* e que são utilizados pelo Recsys.

O Desenvolvimento de outras abordagens para o Recsys.jl, sendo um *framework* muito novo em relação aos outros, Recsys.jl ainda possui uma quantidade muito baixa de abordagens implementadas.

Referências

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

[2] Bernartt, J. L. V., e others. Um sistema de recomendação baseado em filtragem colaborativa.

[3] Bobadilla, J., Ortega, F., Hernando, A., e Gutiérrez, A. Recommender systems survey. *Knowledge-Based Systems 46* (2013), 109–132.

[4] Bondi, A. B. Characteristics of scalability and their impact on performance. In *Proceedings of the 2nd international workshop on Software and performance* (2000), ACM, pp. 195–203.

[5] Breese, J. S., Heckerman, D., e Kadie, C. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence* (1998), Morgan Kaufmann Publishers Inc., pp. 43–52.

[6] Community, J. Julia 0.4 release announcement, 2015.

[7] do Carmo, F. B. Transformando o problema de filtragem colaborativa em aprendizado de máquina supervisionado. *Dissertação de Mestrado. Universidade Federal do Rio de Janeiro. Rio de Janeiro, RJ* (2013).

[8] Ekstrand, M. D., Ludwig, M., Kolb, J., e Riedl, J. T. Lenskit: a modular recommender framework. In *Proceedings of the fifth ACM conference on Recommender systems* (2011), ACM, pp. 349–350.

[9] Figueiredo Filho, D. B., e Silva Junior, J. A. Desvendando os mistérios do coeficiente de correlação de pearson (r). *Revista Política Hoje 18*, 1 (2010).

[10] Funk, S. Disponivel em: http://sifter.org/ simon/journal/20061211.html, acesso em 20 de outubro de 2015, 2012.

[11] Gantner, Z., Rendle, S., Freudenthaler, C., e Schmidt-Thieme, L. Mymedialite: A free recommender system library. In *Proceedings of the fifth ACM conference on Recommender systems* (2011), ACM, pp. 305–308.

[12] Garson, G. D. Structural equations modelling, from statnotes: Topics in multivariate analysis. *North Carolina State University, Retrieved May 25* (2008), 2009.

[13] Goldberg, D., Nichols, D., Oki, B. M., e Terry, D. Using collaborative filtering to weave an information tapestry. *Communications of the ACM 35*, 12 (1992), 61–70.

[14] Guo, G. Librec – getting started, 2015.

[15] Gutierrez, D. D. Disponível em: http://blog.operasolutions.com/bid/387700/data-science-recommender- engines-with-collaborative-filtering, acesso em 10 de outubro de 2015, 2014.

[16] Herlocker, J. L., Konstan, J. A., Borchers, A., e Riedl, J. An algorithmic framework for performing collaborative filtering. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval* (1999), ACM, pp. 230–237.

[17] Hilbert, M., e López, P. The world's technological capacity to store, communicate, and compute information. *science 332*, 6025 (2011), 60–65.

[18] Hofmann, T. Collaborative filtering via gaussian probabilistic latent semantic analysis. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval* (2003), ACM, pp. 259–266.

[19] Ingersoll, G. Introducing apache mahout. *Scalable, commercial friendly machine learning for building intelligent applications. IBM* (2009).

[20] Jeff Bezanson, Stefan Karpinski, V. B. S. Julia license, 2015.

[21] Jeff Bezanson, Stefan Karpinski, V. S. A. E. Why we created julia, 2012.

[22] Kohavi, R., e others. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Ijcai* (1995), vol. 14, pp. 1137–1145.

[23] Konstan, J. A., Miller, B. N., Maltz, D., Herlocker, J. L., Gordon, L. R., e Riedl, J. Grouplens: applying collaborative filtering to usenet news. *Communications of the ACM 40*, 3 (1997), 77–87.

[24] Lázaro, A. d. S. Análise e seleção de algoritmos de filtragem de informação para solução do problema cold-start item.

[25] Medina, M., e Fertig, C. Algoritmos e programação. *São Paulo, Novatec Editora* (2005).

[26] Park, S.-T., e Chu, W. Pairwise preference regression for cold-start recommendation. In *Proceedings of the third ACM conference on Recommender systems* (2009), ACM, pp. 21–28.

[27] Paterek, A. Improving regularized singular value decomposition for collaborative filtering. In *Proceedings of KDD cup and workshop* (2007), vol. 2007, pp. 5–8.

[28] Pereira Filho, J. B. A., e others. Um algoritmo de filtragem colaborativa baseado em svd.

[29] Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., e Riedl, J. Grouplens: an open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM conference on Computer supported cooperative work* (1994), ACM, pp. 175–186.

[30] Resnick, P., e Varian, H. R. Recommender systems. *Communications of the ACM 40*, 3 (1997), 56–58.

[31] Ricci, F., Rokach, L., e Shapira, B. *Introduction to recommender systems handbook*. Springer, 2011.

[32] Sarwar, B., Karypis, G., Konstan, J., e Riedl, J. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web* (2001), ACM, pp. 285–295.

[33] Schafer, J. B., Frankowski, D., Herlocker, J., e Sen, S. Collaborative filtering recommender systems. In *The adaptive web*. Springer, 2007, pp. 291–324.

[34] Sherrington, M. *Mastering Julia*. Pactk Publishing, 2015.

[35] Silva Filho, W. D., e Cazella, S. C. Star: Um framework para recomendação de artigos científicos baseado na relevância da opinião dos usuários e em filtragem colaborativa. *ENIA 2005* (2005), 1042–1051.

[36] Su, X., e Khoshgoftaar, T. M. A survey of collaborative filtering techniques. *Advances in artificial intelligence 2009* (2009), 4.

[37] Tapscott, D., e Williams, A. D. *Wikinomics: How mass collaboration changes everything*. Penguin, 2008.

[38] Teevan, J., Dumais, S. T., e Horvitz, E. Personalizing search via automated analysis of interests and activities. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval* (2005), ACM, pp. 449–456.

Apêndice A

Código do Lenskit para K-NN

Código A.1: Código do Lenskit

```
1  import org.grouplens.lenskit.iterative.*
2  import org.grouplens.lenskit.knn.item.*
3  import org.grouplens.lenskit.mf.funksvd.*
4  import org.grouplens.lenskit.transform.normalize.*
5
6  trainTest {
7      dataset crossfold("ml-100k") {
8          source csvfile("ml-100k/u.data") {
9              delimiter "\t"
10             domain {
11                 minimum 1.0
12                 maximum 8.0
13                 precision 1.0
14             }
15         }
16     }
17
18     algorithm("ItemItem") {
19         bind ItemScorer to ItemItemScorer
20         bind UserVectorNormalizer to BaselineSubtractingUserVectorNormalizer
21         within (UserVectorNormalizer) {
22             bind (BaselineScorer, ItemScorer) to ItemMeanRatingItemScorer
23         }
24     }
25
26     metric MAEPredictMetric
27     metric RMSEPredictMetric
28
29     output "eval-results.csv"
30 }
```

Apêndice B

Código do LibRec para K-NN

Código B.1: Código do LibRec – Codigo Java

```
1  package librec.main;
2
3  import librec.util.FileIO;
4  import librec.util.Logs;
5
6  public class LibRecKFold
7  {
8
9      public static void main(String[] args) {
10         try {
11             new LibRecKFold().execute(args);
12
13         } catch (Exception e) {
14             e.printStackTrace();
15         }
16     }
17
18     public void execute(String[] args) throws Exception
19     {
20         // config logger
21         String dirPath = FileIO.makeDirPath("demo");
22         Logs.config(dirPath + "log4j.xml", true);
23
24         // set the folder path for configuration files
25         String configDirPath = FileIO.makeDirPath(dirPath, "config");
26
27         String configFile = "UserKNNKFOLD.conf";
28
29         long tempoInicio = System.currentTimeMillis();
30
31
```

```
32          // run algorithm
33          LibRec librec = new LibRec();
34          librec.setConfigFiles(configDirPath + configFile);
35          librec.execute(args);
36
37          long tempoFinal = System.currentTimeMillis();
38          Long tempoTotal = tempoFinal-tempoInicio;
39          System.out.println(tempoTotal);
40
41      }
42  }
```

Código B.2: Código do LibRec – Arquivo de configuração

```
1  dataset.ratings.lins=./demo/Datasets/movielens/user.data
2  ratings.setup=-columns 0 1 2 -threshold -1
3  recommender=UserKNN
4  evaluation.setup=cv -k 5 -p off --rand-seed 1 --test-view all
5  item.ranking=off -topN -1 -ignore -1
6  similarity=PCC
7  num.shrinkage=-1
8  num.neighbors=12
9  output.setup=on -dir ./demo/Results/
```

Apêndice C

Código para Gerar os experimentos

Código C.1: Código para Gerar os experimentos .

```
1
2  using Recsys
3
4
5  dataset100K = Recsys.Dataset(string(Pkg.dir("Recsys"), "/datasets/user.data"))
6
7  expHoldout100k = Recsys.HoldOut(0.8,dataset100K)
8
9  expKfold100k = Recsys.KFold(8,dataset100K)
10
11 function createModelRegularedSVD(data)
12   return Recsys.RegularedSVD(data);
13 end
14
15 function createModelImprovedRegularedSVD(data)
16   return Recsys.ImprovedRegularedSVD(data);
17 end
18
19 function createModelKNN(data)
20   return Recsys.KNN(data,:user,12,Recsys.pearsonCorrelation);
21 end
22
23 function testAll()
24   println("100K Recsys.HoldOut Recsys.RegularedSVD")
25   result = expHoldout100k.run(createModelRegularedSVD)
26   file = open("100KHoldOutRegularedSVD.result","w")
27   write(file,string(result))
28   close(file)
29
30   println("100k Recsys.KFold Recsys.RegularedSVD")
31   result = expKfold100k.runAll(createModelRegularedSVD)
```

```
32    file = open("100KKFoldRegularedSVD.result","w")
33    write(file,string(result))
34    close(file)
35
36    println("100K Recsys.HoldOut Recsys.ImprovedRegularedSVD")
37    result = expHoldout100k.run(createModelImprovedRegularedSVD)
38    file = open("100KHoldOutImprovedRegularedSVD.result","w")
39    write(file,string(result))
40    close(file)
41
42    println("100k Recsys.KFold Recsys.ImprovedRegularedSVD")
43    result = expKfold100k.runAll(createModelImprovedRegularedSVD)
44    file = open("100KKFoldImprovedRegularedSVD.result","w")
45    write(file,string(result))
46    close(file)
47
48    println("100K Recsys.HoldOut Recsys.KNN")
49    result = expHoldout100k.run(createModelKNN)
50    file = open("100KHoldOutKNN.result","w")
51    write(file,string(result))
52    close(file)
53
54    println("100k Recsys.KFold Recsys.KNN")
55    result = expKfold100k.runAll(createModelKNN)
56    file = open("100KKFoldKNN.result","w")
57    write(file,string(result))
58    close(file)
59
60  end
61
62  testAll()
```