CENTRO UNIVERSITÁRIO LUTERANO DE PALMAS
ULBRA
COMUNIDADE EVANGÉLICA LUTERANA "SÃO PAULO"
Recredenciado pela Portaria Ministerial nº 3.607 - D.O.U. nº 202 de 20/10/2005

Deise Miranda Borges

DESENVOLVIMENTO DE UMA FERRAMENTA DE RECOMENDAÇÃO AUTOMÁTICA DE PRODUTOS

Palmas

2011

Deise Miranda Borges

DESENVOLVIMENTO DE UMA FERRAMENTA DE RECOMENDAÇÃO AUTOMÁTICA DE PRODUTOS

Trabalho apresentado como requisito parcial da disciplina de Trabalho de Conclusão de Curso I e II (TCC I e TCC II) do curso de Sistemas de Informação, orientado pelo Professor Mestre Fernando Luiz de Oliveira.

Palmas

2011

Deise Miranda Borges

DESENVOLVIMENTO DE UMA FERRAMENTA DE RECOMENDAÇÃO AUTOMÁTICA DE PRODUTOS

Trabalho apresentado como requisito parcial da disciplina de Trabalho de Conclusão de Curso I e II (TCC I e TCC II) do curso de Sistemas de Informação, orientado pelo Professor Mestre Fernando Luiz de Oliveira.

Aprovada em 22 de junho de 2011.

BANCA EXAMINADORA

_____

Prof. M.Sc. Parcilene Fernandes de Brito

Centro Universitário Luterano de Palmas

_____

Prof. M.Sc. Fabiano Fagundes

Centro Universitário Luterano de Palmas

_____

Prof M.Sc Fernando Luiz de Oliveira

Centro Universitário Luterano de Palmas

Palmas

2011

DEDICATÓRIA

Dedico este trabalho à minha mãe, ao meu pai e, à minhas irmãs, por toda a confiança, carinho, compreensão e, apoio.

AGRADECIMENTOS

A Deus, por me dar forças para superar os obstáculos e motivação para desenvolver este trabalho de conclusão de curso.

À minha mãe, que sempre me incentivou a estudar me apoiando quando eu pensava que iria fraquejar, me ensinando que não é errado não sermos perfeitos e que é preciso apenas esforço e persistência para atingir os objetivos.

Ao meu Pai, que me ensinou a nunca desistir dos meus objetivos.

Às minhas irmãs Denise e Danielle, por fazerem parte do meu crescimento e cuidarem da sua irmã caçula. Agradecer principalmente a minha irmã mais velha Denise por estar todo o tempo me apoiando e me incentivando no decorrer do curso e a persistir firme no desenvolvimento do trabalho de conclusão de curso, mesmo estando longe.

Aos meus amigos Douglas, Luane e Naara que fazem parte da minha vida desde o inicio do curso, onde persistimos e enfrentamos os obstáculos juntos, colaborando para que eu pudesse chegar ao final do curso.

Ao meu professor e orientador Fernando Luiz, que teve um papel fundamental no meu crescimento acadêmico, me incentivando a desenvolver trabalhos científicos. Professor que desde o Relatório de Estágio até o Trabalho de Conclusão de Curso me orientou (ótima orientação por sinal), fazendo seus comentários irônicos e divertidos sobre os textos, sempre atento aos erros e apresentando sugestões de melhorias para o trabalho.

Aos meus professores (Família Sistemas de Informação): Fabiano Fagundes – conselheiro e amigo que sempre me animou, desafiou, incentivou a estudar e correr atrás das minhas metas, mostrando que basta apenas querer e se esforçar para alcança-las; Cristina D'Ornellas Filipakis – uma pessoa incrível, exemplo de professora e pessoa que sempre está disposta a ajudar, a aconselhar e escutar; Parcilene Fernandes de Brito – é um poço de conhecimento, que se diverte com algumas ideias sem cabimento das aulas de Gestão Tecnológica; Edeilson Milhomem – com o seu jeito meio “doido”, me ajudou muito no desenvolvimento deste trabalho, discutindo ideias e me mostrando que eu era capaz, bastava apenas acreditar; Madianita Bogo – uma pessoa sensacional sempre com um sorriso no rosto, um coração “mole” e disposta a ajudar seus alunos; Jackson Gomes - que no inicio do curso teve um papel fundamental, incentivando a estudar novas

tecnologias, sempre disposto a ajudar e conversar. Enfim a todos esses professores que tiveram um papel fundamental para o meu crescimento pessoal e profissional.

Agradeço a todos que contribuíram para o desenvolvimento deste trabalho como: Rafael, Jorge, Elizabeth, Roneylson. Em especial a Elizabeth que em meio a tantas ocupações sempre tinha um tempo para contribuir com a correção do texto.

RESUMO

Os Sistemas de Recomendação Personalizados levam em consideração as informações personalizadas de seus usuários, com o intuito de sugerir produtos que estejam de acordo com as suas preferências, diminuindo o esforço do cliente para encontrar o que deseja. Para sugerir produtos de forma personalizada, o Sistema de Recomendação utiliza técnicas de filtragem, as quais processam a informação do usuário e gera a recomendação personalizada. Para entender o processo de funcionamento do Sistema de Recomendação e como a ferramenta desenvolvida irá funcionar, este trabalho aborda os conceitos envolvidos em um Sistema de Recomendação como a taxonomia, construção de perfis, coleta de informações do usuário e técnicas de recomendação (Filtragem Baseada em Conteúdo, Filtragem Colaborativa e Filtragem Híbrida) e Recuperação da Informação. Dentro deste contexto, este trabalho tem como objetivo apresentar os conceitos e etapas envolvidas no desenvolvimento de uma ferramenta de recomendação personalizada, que irá gerar recomendações de produtos de acordo com o perfil do usuário. Esta ferramenta poderá ser agregada a uma aplicação cliente, tal como um Comércio Eletrônico.

PALAVRAS-CHAVE: Sistema de Recomendação, Filtragem, Recuperação da Informação

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS

API - *Application Programming Interface*

CE - Comércio Eletrônico

FBC - Filtragem Baseada em Conteúdo

FC - Filtragem Colaborativa

FH - Filtragem Híbrida

IDE – *Integrated Development Environment*

IDF – *Inverse Document Frequency*

JDBC – *Java Database Conectivity*

PTV – *Personalized TV*

RI - Recuperação da Informação

SR - Sistema de Recomendação

TF – *Term Frequency*

SUMÁRIO

1 INTRODUÇÃO

A diversidade e quantidade de produtos disponíveis para os clientes, tanto no mercado físico como no virtual, é grande. Porém, se por um lado isto pode ser considerado como um grande benefício, pois possibilita ao usuário escolher o que mais lhe convier, por outro, também pode ser visto como um grande problema. Por exemplo, no caso de um Comércio Eletrônico (CE), se essa grande diversidade e quantidade de produtos não estiver bem organizada e não possuir mecanismos eficientes que possibilitem a localização de um produto específico, pode se tornar um problema para o consumidor, que muitas vezes não encontra o produto que realmente deseja, ficando insatisfeito e/ou frustrado.

Por isto, as empresas de Comércio Eletrônico vêm tentando encontrar estratégias para solucionar esse tipo de problema. Uma das respostas encontradas foi a utilização de Sistemas de Recomendação (SR) para que, de certa forma, o sistema possa antecipar uma possível escolha do cliente, recomendando um produto e diminuindo o tempo de procura do produto pelo cliente.

Portanto, um SR, além de facilitar a escolha de um produto, pode possibilitar um aumento nas vendas de um CE, pois um site que demonstra um tratamento diferenciado para o seu cliente e faz recomendações, que em sua maioria são de acordo com as preferências do cliente, pode cativar mais o cliente e fidelizá-lo. Segundo Schafer et. al (1999, p.1), os Sistemas de Recomendação podem também possibilitar o aumento das vendas do CE em três circunstâncias:

- convertendo visitantes em compradores: são os usuários que sempre navegam no site, mas nada compram. Nessas situações o SR pode ajudar o visitante a encontrar o produto que deseja, através da apresentação de sugestões de produtos, ou lista de produtos mais vendidos;
- aumentando a venda adicional: o SR pode ajudar a melhorar a venda adicional (*cross-sell*), que são as vendas “casadas” de produtos, onde o sistema sugere itens adicionais - no momento do processo de compra de algum produto ou baseado no itens do carrinho de compra - de forma que os itens sugeridos estão relacionados com o que está sendo comprado. Dessa forma, se as recomendações forem satisfatórias ao cliente, o valor do pedido pode aumentar;
- ganhando e aumentando a fidelidade: devido ao SR utilizar as preferências do usuário para fazer as recomendações personalizadas, o cliente começa a perceber

que há uma certa preocupação por parte da empresa em conhecê-lo melhor e, assim, apresenta uma interface que se adéque melhor as necessidades do usuário. Desta maneira, quanto mais um cliente utiliza o SR, mais o sistema terá a possibilidade de aprender sobre as preferências do cliente e assim fazer recomendações cada vez mais exatas além de deixar o cliente satisfeito e leal ao site.

Dentro deste contexto, este trabalho tem como objetivo desenvolver uma ferramenta de recomendação automática de produtos que pode servir como um módulo adicional para um site já existente, como, por exemplo, um Comércio Eletrônico. A empresa de CE que adquirir essa ferramenta de recomendação irá fornecer os dados para que o Sistema de Recomendação os processe e, assim, possibilitar que a recomendação personalizada seja gerada. Por exemplo, para a ferramenta de recomendação ser acoplada ao site cliente, será necessário a conexão da mesma com o banco de dados do CE para que se possa obter as informações necessárias para gerar as recomendações. As informações que serão obtidas no banco de dados será o conjunto de avaliações feitas pelos usuários sobre os produtos e o conteúdo da descrição dos produtos avaliados.

Nas seções seguintes serão apresentados os itens necessários para o desenvolvimento do trabalho como: referencial teórico, abordando os principais conceitos do SI, Técnicas de Recomendação; materiais e métodos utilizados; resultados e discussões obtidos; e, por fim, as considerações finais sobre o trabalho.

2 REFERENCIAL TEÓRICO

Esta seção apresentará os conceitos necessários para que seja possível o desenvolvimento da ferramenta de recomendação proposta neste trabalho. Para isso abordará conceitos sobre Sistema de Recomendação e Técnicas de Recomendação.

2.1. Sistema de Recomendação

Segundo Borges (2010, p.14), “pode-se dizer que Sistemas de Recomendações (SR) são utilizados para auxiliar os usuários a identificarem serviços ou produtos de interesse que estejam dentro de uma grande quantidade de opções”. Os SRs podem ser classificados em duas categorias: não-personalizado e personalizado. No SR não-personalizado, as recomendações são realizadas sem levar em conta a preferência do usuário. Um exemplo de recomendação não-personalizada é a lista de produtos mais vendidos. Já um SR personalizado leva em consideração as preferências do usuário para realizar uma recomendação, sendo tais recomendações específicas para cada usuário. Para que isso ocorra o SR deve ser capaz de estar sempre “aprendendo” e identificando as preferências dos usuários.

Para realizar a recomendação o Sistema de Recomendação segue alguns passos, conforme a Figura 1 apresenta a seguir.

Figura 1 - Taxonomia dos Sistemas de Recomendação. Traduzida de (SCHAFER et al., 2001, p.10)

Como mostra a Figura 1, um Sistema de Recomendação recebe como entrada informações sobre o usuário e informações retiradas da participação da comunidade no site. Na sequência utiliza os métodos de recomendação para gerar sua saída, que são as recomendações e, assim, utilizam algumas formas de apresentação da recomendação para o usuário.

Uma das etapas mais importantes para o Sistema de Recomendação é a montagem do perfil do usuário, responsável por definir quais tipos de produtos poderão ser recomendados aos usuários. Esse perfil pode ser organizado a partir do armazenamento de preferências. Na próxima seção será explicada como pode ser realizada a construção de perfis dos usuários.

2.1.1. Construção de Perfis

Dudev et al. (2008 *apud* CASTRO & BARBOSA, 2009, p.17) sugerem três importantes etapas no processo de construção do perfil para o levantamento de informações necessárias no processo de personalização:

- modelagem do usuário – são avaliados quais características do usuário podem ser consideradas importantes para a personalização. Essas características são classificadas como críticas ou não e, assim, separadas em grupos específicos.
- perfil do usuário – a partir das interações do usuário no sistema ou informações diretas cadastradas pelo usuário sobre suas preferências, cria-se um perfil que caracterize o usuário. Nessa etapa o perfil do usuário é composto pelas características que foram identificadas. Dessa forma, esta etapa pode ser compreendida como a formação da base de interesses do usuário, podendo ser construída através da filtragem dos dados do usuário que forem enviados para a ferramenta de recomendação.
- personalização – o perfil do usuário é utilizado para realizar as recomendações.

A primeira etapa para o levantamento das informações dos perfis é a etapa de captação de informação dos usuários, que pode ser chamada de coleta de informação e que será abordada na próxima seção.

2.1.1.1. Coleta de Informação do Usuário

Para construir o perfil do usuário, primeiramente o sistema deve obter informações sobre o mesmo. Posteriormente, essas informações são armazenadas e processadas para que seja criado um perfil que represente o usuário no sistema. Ao final, as recomendações serão feitas de acordo com o perfil criado. A coleta das informações necessárias para montar o perfil do usuário pode ser realizada de duas formas (JUVINA & OOSTENDORP, 2004 *apud* CASTRO & BARBOSA, 2009, p.17):

- coleta explícita: as informações do usuário são captadas de forma direta. O usuário insere de forma explícita informações ao seu respeito, ou seja, informa suas preferências. Alguns exemplos de como essas informações podem ser obtidas, são através das avaliações de produtos, preenchimento de formulários indicando as áreas de interesse, classificando os produtos por relevância (ex: designando nota aos produtos). Essa abordagem é bem interessante, partindo do principio que as

preferências armazenadas para montagem do perfil foram informadas pelo próprio usuário, fazendo com que se tornem mais confiáveis. Porém, essa forma de coleta demanda tempo por parte do usuário, que nem sempre tem disponibilidade ou disposição para informar suas preferências e necessidades.

- coleta implícita: nesse tipo de abordagem, as informações são extraídas de forma implícita, ou seja, não necessita da contribuição direta do usuário. A coleta implícita pode ser feita através do monitoramento de comportamento do usuário no site, analisando as interações. Alguns exemplos de coleta implícita são a observação das páginas visitadas, palavras-chave buscadas, histórico de compras entre outros.

Analisando as duas abordagens explicadas acima, pode-se perceber que a implícita é ideal para usuários que não possuem disponibilidade de contribuir com a alimentação do sistema, pois ela é realizada de forma automática por meio da análise do ciclo de interações do usuário no site. Porém, pode não ser tão confiável como a explícita, pelo fato da decisão de classificar a informação como uma preferência ou não do usuário ser do sistema. Já na abordagem explícita não ocorre o mesmo problema, uma vez que as informações inseridas foram cadastradas diretamente pelo usuário. Apesar de a abordagem explícita coletar informações mais confiáveis, existe um problema também relacionado ao tempo que o usuário pode levar para contribuir com o sistema, informando tais dados.

Na próxima seção serão abordados sobre técnicas de recomendação, que são utilizadas para selecionar os produtos a serem recomendados, de acordo com as preferências do usuário.

2.2. Técnicas de Recomendação

Para realizar as recomendações, o Sistema de Recomendação necessita de uma forma para filtrar as informações relevantes para cada cliente e, assim, apresentar as sugestões. Essa filtragem pode ser feita de três maneiras: Filtragem Baseada em Conteúdo (FBC), Filtragem Colaborativa (FC) e Filtragem Híbrida (FH). A Figura 2 demonstra de forma genérica o funcionamento de um SR utilizando as técnicas de recomendação FBC e FC.

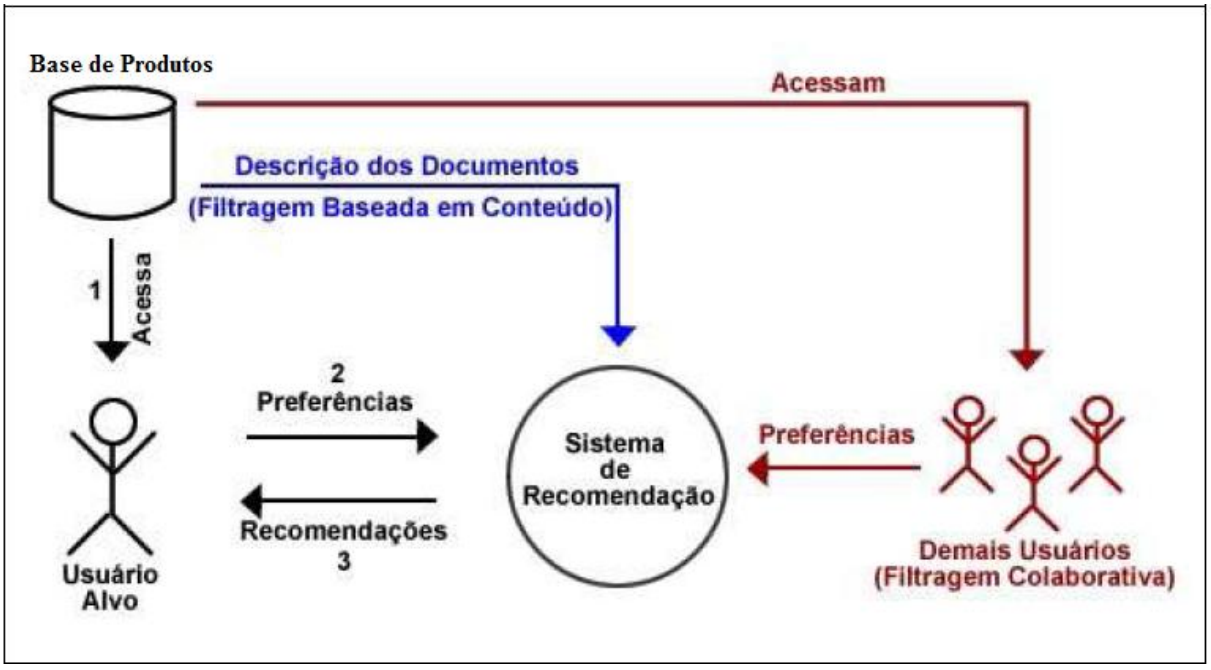


Figura 2 – Técnicas de Recomendação em um Sistema de Recomendação (GODOY NETO, 2009, p.48)

A Figura 2 apresenta um modelo genérico de um Sistema de Recomendação, mostrando o processo realizado ao utilizar a técnica de recomendação. Quando se utiliza a técnica de Filtragem Baseada em Conteúdo é levado em consideração apenas o perfil do usuário e a base de produtos, para fazer o relacionamento do conteúdo dos produtos com as preferências do usuário e assim selecionar os produtos a serem recomendados. Já quando se utiliza a Filtragem Colaborativa, o SI leva em consideração as preferências do usuário e dos demais usuários, para fazer uma comparação entre os perfis e observar quais perfis dos demais usuários são mais similares ao do usuário alvo. A seguir será abordada a técnica de Filtragem Baseada em Conteúdo.

2.2.1. Filtragem Baseada em Conteúdo (FBC)

A FBC parte do princípio que se o usuário consumiu um item anteriormente ele poderá consumir também outros itens similares. Dessa forma, para que se possa saber que itens são similares a outros, é necessário fazer uma análise do conteúdo desses produtos, ou seja, fazer uma correlação dos produtos que já foram consumidos pelo usuário com novos produtos que estão na base.

O perfil do usuário é composto por palavras-chave que juntas formam as preferências do mesmo, onde os itens que já foram consumidos possuem descrições de seus conteúdos e palavras que os identificam.

A FBC é baseada nas técnicas de Recuperação da Informação (RI), que, segundo Baeza-Yates & Ribeiro-Neto (1999, p.1), é uma área que lida com a representação, armazenamento, organização e acesso das informações. Dessa forma, ela proporciona fácil acesso às informações, as quais os usuários necessitam, ou seja, é uma técnica que recupera informações para que os usuários possam ter acesso a elas.

Segundo Baeza-Yates & Ribeiro-Neto (1999, p. 21), existem duas formas de realizar o processo da RI: *ad-hoc* e com filtragem. Na forma *ad-hoc*, existe uma base com a coleção de documentos, no qual são realizadas consultas e retornado o resultado. A *ad-hoc* leva somente em consideração a consulta realizada, independente do usuário. Dessa forma, qualquer usuário que faça uma consulta com as mesmas palavras receberá o mesmo retorno. Diferentemente da forma *ad-hoc*, o tipo de recuperação por filtragem utiliza o perfil do usuário, utilizando a descrição das suas preferências para comparar com a descrição dos itens que estão na base de dados. Dessa forma, pode-se determinar quais documentos são de possível interesse do usuário.

Segundo Baeza-Yates & Ribeiro-Neto (1999, p. 24) os modelos clássicos do processo de RI consideram que cada documento é composto por um conjunto de palavras-chave representativas, que são chamadas de índice de termos, ou seja, conjunto de palavras que identificam os temas do documento. Dessa forma, os índices de termos são utilizados para indexar e resumir o conteúdo do documento. Existem três modelos clássicos que podem ser utilizados no processo de recuperação da informação: Probabilístico, Booleano e Vetorial.

O Modelo Probabilístico tenta estimar a probabilidade de um documento ser relevante para o usuário, dada uma consulta. Ele descreve os documentos atribuindo pesos binários 0 e 1, que representam a ausência ou presença dos termos, respectivamente. Dessa forma, cada termo que esteja presente em um documento é encarado como uma evidência de um documento ser relevante ou não. Assim, é gerado um vetor resultante, baseado na probabilidade de um documento ser relevante ou não para a consulta, que é calculada através do princípio probabilístico de ordenação (GODOY NETO, 2009, p.46).

O Modelo Booleano é baseado na teoria dos conjuntos e álgebra booleana. As consultas são especificadas como expressões booleanas (BAEZA-YATES & RIBEIRO-NETO, 1999, p.25). Dessa forma, essas consultas podem conter operadores lógicos: *and, or e not*. Nesse modelo cada documento é representado por um vetor de índices termos, que possuem pesos binários. Caso o termo esteja presente no documento recebe “1” e, caso contrário, “0”. Assim, por se tratar de um modelo exato, não existe resultado parcial (um documento não pode parcialmente atender a consulta).

Para que um documento seja similar, a comparação da consulta com o documento deve ser verdadeira. Dessa forma, não existe a possibilidade de construir uma lista ordenada pelos documentos mais relevantes, pois nesse modelo não ocorre a parcialidade de relevância de um documento em relação a consulta (considerado uma desvantagem). Sendo assim, um documento atende as necessidade da consulta ou não.

Nesse trabalho será utilizado para implementar a técnica de Recuperação da Informação, o Modelo Clássico Vetorial ou Modelo Espaço Vetorial, como é mais conhecido. Devido a isso, este modelo será abordado de forma detalhada a seguir.

O Modelo Espaço Vetorial é um modelo algébrico, que utiliza o vetor para representar um documento. Segundo Silva (2009, p.26), cada documento é representado por várias palavras diferentes, que são conhecidas como termos. Um termo pode aparecer com certa frequência em um mesmo documento, onde o cálculo dessa frequência resulta no peso não binário que cada termo possui dentre uma lista de termos de um mesmo documento (o peso de cada termo indica o grau de relevância no documento). Assim, um documento pode ser representado por um vetor de *n* dimensões que, em cada coordenada, tem representada a frequência de cada termo. O cálculo do peso de cada termo é de fundamental importância, pois são com esses valores que irá se calcular a similaridade entre os documentos da base e a *query* (consulta tendo o conjunto de palavras-chave sobre as preferências do usuário).

Antes que ocorra o cálculo do peso de cada termo, um documento pode passar por um processo de análise dos dados, que é realizado com o intuito de melhorar o resultado do processo de recuperação da informação na busca por documentos similares. Segundo Godoy Neto (2009, p.47) esse processo é realizado com a aplicação de duas técnicas: *stemming* e *stopwords*, que são responsáveis por

atuar no pré-processamento do texto. Esta fase de pré-processamento é responsável por analisar o texto e excluir os termos irrelevantes e as redundâncias existentes, fazendo com que a estrutura de indexação do texto seja reduzida. A seguir serão explicadas mais detalhadamente as duas técnicas citadas acima:

- *stemming*: é responsável por reduzir as variantes de uma palavra ao seu radical. Assim, essa técnica encontra os termos com significados similares e eliminam seus sufixos, reduzindo o termo a sua raiz. Dessa forma, vários termos que poderiam ser diferentes, devido a sua escrita, são unificados. A Tabela 1, a seguir, demonstra um exemplo da aplicação dessa técnica;

Tabela 1 - Aplicação de Stemming

Termos	**Stemming**
Mundial	Mund
Mundialmente	Mund
Mundo	Mund

A Tabela 1 mostra um exemplo da aplicação da técnica de *stemming*, onde foram eliminados os sufixos “ial”, ”ialmente” e “o”, respectivamente, dos termos “Mundial”, “Mundialmente” e “Mundo”. Ao aplicar a técnica de *stemming* nos termos apresentados, estes serão reduzidos ao seu radical “Mund”.

- *stopwords*: é responsável por eliminar os itens comuns do texto, que aparecem com muita frequência e não trazem significado nenhum para o tema do documento, como artigos, preposições, conjunções e pronomes. A eliminação desses itens não será prejudicial para análise da similaridade entre os textos, pelo fato desses itens serem úteis apenas para trazer um sentido gramatical ao texto. Sendo assim, mesmo que ocorra a eliminação dos artigos, preposições, conjunções e pronomes, os termos que são relevantes ainda terão o mesmo significado. Na Tabela 2 será apresentado um exemplo da aplicação dessa técnica.

Tabela 2 - Aplicação de Stopwords

Documento	**Lista de *Stopwords***	**Termos Relevantes**
Engenheiros ~~do~~ Google testaram ~~um~~ carro ~~que~~ dirige sozinho ~~nas~~ ruas ~~da~~ Califórnia	do	Engenheiros
	um	Google
	que	testaram
	nas	carro
		dirige
	da	ruas
		Califórnia

No exemplo demonstrado pela Tabela 2, é aplicada a técnica de *stopwords*, que retira do documento as palavras sem significado relevante. Neste exemplo, as palavras retiradas foram “do”, “um”, “que”, “nas” e “da”, deixando somente os substantivos ou verbos que possuem significados independentes de outras palavras.

Após a análise dos dados, pode-se obter um documento mais seleto com os termos mais relevantes e, a partir desse momento, seguir para o cálculo do peso de cada termo. Este peso pode ser calculado através do fator chamado Frequência do Termo, *Term Frequency* (TF), onde a formalização do cálculo pode ser representada na equação matemática:

$$tf_{i,j} = \frac{freq}{\max freq}$$

Na equação apresentada acima, a variável *freq* representa a frequência da ocorrência de determinado termo em um documento. Esta frequência é dividida pela máxima ocorrência do termo mais frequente no documento, ou seja, a frequência do termo que mais ocorrer no documento. Desta forma, pode-se obter o peso do termo específico que foi calculado.

É possível também encontrar a frequência inversa do termo, onde a mesma tem como objetivo identificar os termos que ocorrem com frequência em um documento e também aparece muito frequentemente em outros documentos de temas diferentes, denominados termos comuns. Dessa forma, esses termos não

influenciam na classificação de um documento como relevante ou não, pois eles têm os seus pesos diminuídos (menos relevância para o documento).

O cálculo da Frequência Inversa, *Inverce Document Frequency* (IDF), pode ser representado através da fórmula:

$$idf_i = log \frac{N}{n_i}$$

A equação apresentada acima leva em consideração a quantidade total de documentos existentes na Base "*N"* e a quantidade de documentos "*n"* que ocorrem o determinado termo.

Depois de realizado o cálculo da TF e IDF, pode-se relacionar o TF-IDF para obter a medida do peso, que pode ser calculada através da multiplicação da frequência do termo pela frequência inversa, como apresentado na fórmula:

$$w_{i,j} = tf\ x\ idf$$

Após medido o peso de cada termo de cada documento, é possível calcular a similaridade entre dois documentos. Esse cálculo da similaridade entre os documentos pode ser realizado através da fórmula do co-seno, pois segundo Wives (2002, p.39) "cada elemento do vetor é considerado uma coordenada dimensional. Assim, os documentos podem ser colocados em um espaço euclidiano de n dimensões [...] e a posição do documento em cada dimensão é dada pelo seu peso".

O cálculo do co-seno pode ser representado pela equação a seguir.

$$sim(u,d) = \frac{\sum_{i=1}^{n}(w_{u,i}\ \times\ w_{d,i})}{\sqrt{\sum_{i=1}^{n}(w_{u,i})^{\,2}}\ \times\ \sqrt{\sum_{i=1}^{n}(w_{d,i})^2}}$$

Para calcular a similaridade através da fórmula acima, é necessário o cálculo do produto escalar, onde ocorre a somatória do produto dos pesos $w_{u,i}$ e $w_{d,i}$, no qual esses pesos são dos termos que representam os dois documentos "u" (descrição da preferência do usuário) e "d" (documento da base de dados). Segundo Silva (2009, p.26), esse cálculo pode ser representado da seguinte forma:

$$w_u . w_d = w_{u,1}w_{d,1} + w_{u,2}w_{d,2} + \ldots + w_{u,n}w_{d,n}$$

Depois de calculado o produto escalar, divide-se o resultado pelo módulo dos vetores "u" e "d", ou seja, a soma do quadrado de cada peso de um mesmo documento e depois a multiplicação entres os resultados de cada documento.

Segundo Silva (2009, p.27), a soma do quadrado dos n pesos pode ser representada da seguinte forma para os dois documentos:

$$\|wu\| = \sqrt{w_{u,1}^2 + w_{u,2}^2 + \ldots + w_{u,n}^2} = \sum_{i=1}^{n} (w_{u,i})^2$$

$$\|wd\| = \sqrt{w_{d,1}^2 + w_{d,2}^2 + \ldots + w_{d,n}^2} = \sum_{i=1}^{n} (w_{d,i})^2$$

A função do co-seno calcula a similaridade entre os documentos. Essa similaridade é possível de ser verificada através do ângulo (calculado pela função do co-seno) formado entre dois documentos, que varia entre 0 e 1. Quanto menor o ângulo entre os documentos, maior será o co-seno e, consequentemente, maior será o grau de similaridade entre os documentos.

Depois de realizado todo o processo para o cálculo de similaridade entre os documentos, pode-se montar uma lista contendo os documentos mais similares. Esta lista pode ser formada por uma espécie de *ranking* com os documentos dos mais similares aos menos, para que se possa fazer a recomendação para o usuário.

Para um melhor entendimento da técnica, bem como da utilização das fórmulas citadas acima, a Figura 3, a seguir, apresenta um exemplo de dois documentos que foram submetidos às técnicas de *Stopwords* e *Stemming.* Sobre este resultado, serão aplicadas as fórmulas para calcular a similaridade entre os documentos.

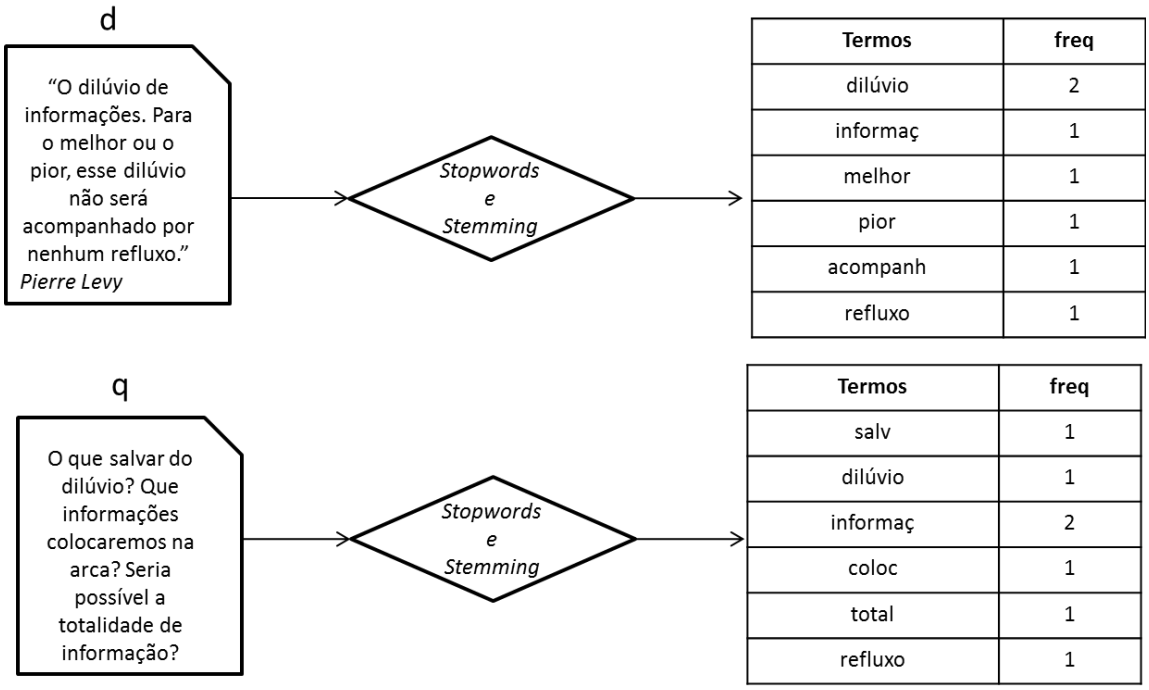


Figura 3 - Aplicação de *Stopwords* e *Stemming*

A Figura 3 demonstra um exemplo de dois documentos "d" (documento da base) e "q" (documento do usuário), no qual são aplicadas as operações sobre o texto e gerado o vetor de termos. O próximo passo para que se possa calcular a similaridade é calcular o peso de cada Termo. Para isso, será utilizado o TF-IDF (*Term Frequency- Inverce Document Frequency*). Para exemplificar como será calculado o peso do termo "dilúvio" do documento "d", considera-se um total de documentos da base "N=30" e o número de documentos em que esse termo ocorre como "n=3".

$$tf_{dilúvio,d} = \frac{2}{2}$$

$$idf_{dilúvio} = \log\frac{30}{3}$$

$$w_{dilúvio,d} = 1 \times 1$$

Depois dos cálculos realizados, pode-se obter um valor de "1" como peso para o termo "dilúvio" do documento "d". Nas Tabelas 3 e 4, a seguir, serão apresentados os dois documentos com seus respectivos pesos dos termos calculados, considerando que o total de documentos da base seja "N=30" e o número de documentos em que esse termo ocorre como "n=3".

Tabela 3 - Peso dos Termos dos Documentos "d" e “q”

Documento "d"		**Documento "q"**	
Termos	**Peso**	**Termos**	**Peso**
dilúvio	1	dilúvio	0,5
informaç	0,5	informaç	1
refluxo	0,5	refluxo	0,5
salv	0	salv	0,5
coloc	0	coloc	0,5
total	0	total	0,5

Depois da extração dos termos e seus respectivos pesos calculados, é necessário a utilização da fórmula do co-seno para calcular a similaridade entre os documentos “d” e “q” apresentados na Tabela 3.

$$sim_{d,q} = \frac{(0{,}5 * 1) + (1 * 0{,}5) + (0{,}5 * 0{,}5) + (0{,}5 * 0) + (0{,}5 * 0) + (0{,}5 * 0)}{\sqrt{0{,}5^2 + 1^2 + 0{,}5^2 + 0{,}5^2 + 0{,}5^2 + 0{,}5^2} \times \sqrt{1^2 + 0{,}5^2 + 0{,}5^2 + 0^2 + 0^2 + 0^2}} = 0{,}68$$

Portanto, de acordo com o cálculo realizado acima, o resultado da similaridade entre os documentos “d” e “q” foi 0,68, ou seja, dentro de um padrão que varia entre 0 e 1, o documento “d” é considerado similar ao documento “q”.

Assim como outras técnicas, a FBC possui suas vantagens e desvantagens:

- Vantagens
    - quando o usuário possui preferências incomuns, a utilização dessa técnica é uma ótima alternativa, pois como se baseia nos produtos já consumidos pelo usuário, não necessita da contribuição de outros usuários com perfis parecidos (o que seria difícil nesse caso);
    - recomendações mais precisas, pelo fato de levar em consideração apenas as preferências do usuário alvo;
    - todos os documentos da base de dados que sejam similares ao perfil do usuário podem ser recomendados.
- Desvantagens
    - não tem a capacidade de inovar em relação as recomendações. Por exemplo, caso um usuário queira consumir um item diferente do que já

consumiu o sistema não terá a capacidade de recomendar, pois essa técnica baseia-se somente nos itens que já foram consumidos;

- quando o usuário é novo e não possui muitos itens consumidos ou avaliados, o sistema pode gerar recomendações imprecisas;
- limitada as características textuais de um produto, ou seja, o conteúdo do produto deve ser um texto ou deve-se inserir de forma manual uma descrição textual do produto contendo suas características.

Na próxima seção será apresentada a Técnica de Recomendação chamada Filtragem Colaborativa, que possui uma abordagem diferente da FBC.

2.2.2. Filtragem Colaborativa (FC)

A Filtragem Colaborativa (FC), ao contrário da FBC, não analisa o conteúdo dos produtos para recomendar algo. No caso, a FC analisa os perfis dos outros usuários com preferências similares para recomendar algo para o usuário alvo. Sendo assim, usuários com perfis semelhantes ao do usuário alvo são analisados e as recomendações são feitas baseadas nos produtos que já foram avaliados por eles. Dessa forma, as recomendações são realizadas por usuários da comunidade que tenham interesses em comum, que podem ser obtidos através das avaliações que tanto o usuário alvo quanto os usuários da comunidade fazem sobre determinado produto.

Segundo Cazella (2006, p.30) “nos sistemas colaborativos a essência está na troca de experiências entre as pessoas que possuem interesses comuns”. Assim, os itens que são recomendados são baseados nas avaliações feitas por outros usuários, e não pelo conteúdo de um produto.

Sistemas que utilizam a FC dependem muito da interação que os usuários da comunidade fazem no site em relação a avaliações de produtos, pois são a partir dessas avaliações que são calculados os usuários com preferências similares (vizinhos próximos). Segundo Burke (2002, p.2), o perfil do usuário em sistemas com a FC consiste em uma matriz de itens e as avaliações do usuário sobre os mesmos.

Os algoritmos colaborativos podem ser agrupados em duas classes: baseados em modelo e baseados em memória (baseado em heurísticas). Os

algoritmos baseados em modelos propõem uma abordagem probabilística para a FC (BREESE et. al 1998 *apud* ADOMAVICIUS & TUZHILIN, 2005, p.738).

Segundo Godoy Neto (2009, p.54), os algoritmos baseados em modelo necessitam da construção prévia de um modelo, formado através da coleção das avaliações do usuário alvo feitas sobre os produtos, e que, assim, formam o conjunto de preferências do mesmo. Esse agrupamento é atualizado, ou seja, reprocessado toda vez que ocorrer uma nova avaliação por parte do usuário alvo, ocasionando um maior custo computacional.

O algoritmo baseado em memória é fundamentado em heurísticas, ou seja, busca os usuários mais similares ao usuário alvo para fazer previsões, com base nas avaliações que foram realizadas, pelos usuários com perfis mais semelhantes ao do usuário alvo, sobre todos os produtos (BREESE et. al 1998 *apud* Adomavicius & Tuzhilin, 2005, p.738). Segundo Sarwar et. al (2001, p.4), esse algoritmo utiliza técnicas de estatística em grupos de usuários da comunidade virtual para poder identificar os clientes com perfis similares ao do usuário alvo e formar o conjunto de "vizinhos". A construção do grupo de usuários com perfis semelhantes é baseada nas avaliações semelhantes entre os clientes da comunidade com o usuário alvo. A Tabela 5, a seguir, apresentará um exemplo de como é a entrada para um algoritmo de FC, em que as linhas da matriz representam os usuários e as colunas os itens avaliados.

Tabela 4 - Similaridade entre Usuários

Usuários	**Item X**	**Item Y**	**Item Z**
Maria	5	4	?
João	1	-	1
José	3	3	5

A Tabela 4 apresenta as avaliações dos usuários sobre alguns itens. Nesta situação o usuário alvo ao qual se deseja fazer a recomendação é "Maria". Para saber quais dos vizinhos possuem o perfil mais próximo é necessário analisar as similaridades das avaliações e tentar predizer a nota que o usuário alvo poderia dar para o item "Z". Assim, de acordo com a nota, o sistema poderia recomendar ou não o "Item Z" ao usuário alvo. Neste exemplo, pode-se observar que o usuário que possui uma avaliação dos itens que mais se assemelha a de "Maria" é o usuário

José, pelo fato das notas que “José” cadastrou para o Item “X” e “Y” (3, 3) serem as mais próximas das notas que “Maria” classificou (5, 4) para os mesmo itens. Dessa forma, é grande a possibilidade do “Item Z” ser oferecido como recomendação para “Maria” baseado na similaridade de perfil.

Para Herlocker (1999, 2000 *apud* CAZELLA, 2006, p.32), a FC pode ser dividida em três pontos principais, que são apresentados na Figura 4, a seguir.

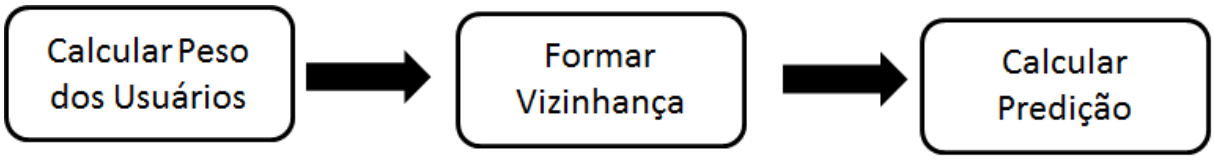


Figura 4 - Pontos Principais da FC

Conforme a Figura 4, o primeiro passo é calcular o peso de cada usuário em relação a similaridade com o usuário alvo (medida de similaridade). Este passo é necessário para que ocorra o segundo passo, que é a fase em que ocorre a formação do conjunto de usuários com perfis mais similares (vizinhança) ao do usuário alvo. Essa vizinhança é formada pelos usuários que possuem pesos maiores em relação à similaridade. Por fim, com a vizinhança do usuário alvo formada, o terceiro ponto é executado para realizar as predições para o usuário alvo, levando em consideração as avaliações dos vizinhos e seus respectivos pesos.

Para que se possa realizar os três passos apresentados acima e, consequentemente, a análise da similaridade, é utilizada a técnica *k-nearest-neighbor* (HERLOCKER, 2000 *apud* CAZELLA, 2006, p.32). Para calcular a similaridade entre os usuários (primeiro passo), pode-se utilizar o coeficiente de correlação de Pearson, que segundo Herlocker (2000 *apud* GODOY NETO 2009, p.57) “apresenta os melhores resultados quando adotada em um sistema de FC”. O cálculo do coeficiente de correlação de Pearson pode ser representado através da seguinte equação:

$$w_{a,u} = \frac{\sum_{i=1}^{m}[(r_{a,i} - \bar{r_a}) \times (r_{u,i} - \bar{r_u})]}{\sqrt{\sum_{i=1}^{m}(r_{a,i} - \bar{r_a})^2 \times \sum_{i=1}^{m}(r_{u,i} - \bar{r_u})^2}}$$

A equação acima representa a fórmula para calcular o coeficiente de correlação de Pearson que é utilizado para calcular o peso (nível de similaridade) de cada usuário da comunidade em relação ao usuário alvo. Nesse caso, a fórmula

verifica a correlação do usuário alvo “a” com outro usuário da comunidade “u”, onde o termo $r_{a,i}$ é a avaliação dada pelo usuário alvo ao item “i” e $\bar{r_a}$ é a média de todas as avaliações do usuário alvo realizadas sobre os produtos. A expressão $r_{u,i}$ representa o conjunto das avaliações feitas por esse integrante da comunidade, sendo que esse conjunto é constituído pelas avaliações dada pelo mesmo a cada item “i”, e $\bar{r_u}$ representa a média das avaliações do usuário “u”. Segundo Cazella (2006, p.32), os resultados da correlação de Pearson retornam um número entre -1 e 1, onde -1 significa total dissimilaridade e 1 total similaridade entre o usuário alvo e o usuário avaliado.

A partir do resultado do cálculo de similaridade entre os usuários e o usuário alvo é formada a vizinhança do mesmo, ou seja, os usuários com perfis mais similares ao do usuário alvo. Dessa forma, é possível calcular a predição (terceiro passo) da nota referente a um item que pode ser recomendado ao usuário alvo, sendo necessária uma combinação das notas dos usuários “vizinhos” com as do usuário alvo. Segundo Cazella (2006, p.32), o cálculo da predição pode ser realizado através da equação:

$$P_{a,i} = \bar{r_a} + \frac{\sum_{u=1}^{n}\left(r_{u,i} - \bar{r_u}\right) \times w_{a,u}}{\sum_{u=1}^{n}\left|w_{a,u}\right|}$$

A equação acima resulta na média ponderada das avaliações atribuídas ao item “i” pelos “n” vizinhos “u” do usuário alvo “a”, onde $\bar{r_a}$ é a média das avaliações do usuário alvo, $r_{u,i}$ a avaliação do usuário “u” sobre o item “i” a ser predito e $w_{a,u}$ é o peso da similaridade entre o usuário “u” e o usuário alvo “a”.

Assim que os três passos citados na Figura 4 forem realizados, é possível predizer a nota que o usuário alvo poderia dar a determinado item que ele ainda não avaliou e, consequentemente, o SR pode avaliar se o item será recomendado ao usuário ou não de acordo com o valor da predição.

A Tabela 5 apresenta um exemplo de entrada para a Filtragem Colaborativa como exemplo da aplicação da técnica, bem como a utilização das fórmulas citadas acima para calcular a similaridade entre os usuários e realizar a predição da nota de um item para o usuário alvo.

Tabela 5 - Exemplo da Aplicação da Filtragem Colaborativa

Usuário	Item 1	Item 2	Item 3	Item 4	Item 5
Joana	5	2	4	1	5
Paulo	1	1	2	2	1
Ana	4	4	3	1	3
Marcos	3	4	3	4	4
Maria	4	1	5	1	?

Analisando a Tabela 5, pode-se perceber, sem a utilização das fórmulas matemáticas, que a usuária “Maria” (usuário alvo) tende a concordar mais com as avaliações da usuária “Joana” e a discordar do usuário “Marcos”. Porém, para que seja provada essa similaridade entre “Joana” e “Maria”, será necessário coletar as avaliações dos itens em comum dos dois usuários: “Maria” {4;1;5;1} e “Joana” {5;2;4;1}.

Para calcular a similaridade entre as duas usuárias é necessário aplicar a fórmula do coeficiente de correlação de Pearson, que terá como médias das avaliações de “Maria” e “Joana”, os respectivos valores: 2,75 e 3. Para que seja possível uma melhor visualização do cálculo do coeficiente de correlação de Pearson, a seguinte fórmula é apresentada para calcular a similaridade entre “Maria” e “Joana”:

$$w_{a,u} = \frac{\big((4-2{,}75)*(5-3)\big)+\big((1-2{,}75)*(2-3)\big)+\big((5-2{,}75)*(4-3)\big)+((1-2{,}75)*(1-3))}{\sqrt{(4-2{,}75)^2+(1-2{,}75)^2+(5-2{,}75)^2+(1-2{,}75)^2\times(5-3)^2+(2-3)^2+(4-3)^2+(1-3)^2}}$$

Através do cálculo da similaridade apresentado acima, pode-se obter o valor de 0,89 para similaridade entre “Maria” e “Joana”. Seguindo a escala que está entre -1 e 1, pode-se entender que o perfil de “Joana” é bem similar ao de “Maria”. Assim, pode-se aplicar a correlação de Pearson nos outros usuários, tendo como base o usuário alvo “Maria”, e formar a vizinhança da mesma. Na Tabela 6, a seguir, são apresentados os usuários com os seus respectivos graus de similaridade.

Tabela 6 - Similaridade entre os Usuários

Usuários	Correlação de Pearson (similaridade)
Joana	0,89
Paulo	0,14
Ana	0,34
Marcos	-0,98

Depois do cálculo das similaridades, o próximo passo é calcular a predição para o "Item 5", que pode ser realizado através da equação de predição, tal como apresentado a seguir:

$$P_{maria,5} = 2{,}75 + \frac{\big((5-3)*0{,}89\big) + \big((1-1{,}5)*0{,}14\big) + \big((3-3)*0{,}34\big) + \big((4-3{,}5)*(-0{,}98)\big)}{0{,}89 + 0{,}14 + 0{,}34 + 0{,}98}$$

Assim, após realizar o cálculo da predição, pode-se prever a nota que a usuária "Maria" daria para o Item 5, caso ela o avaliasse: 3,27.

A FC, além de ter suas vantagens, também apresenta desvantagens, que podem ser explicitadas nos seguintes pontos:

- Vantagens
    - capacidade de inovação nas recomendações, pois como não se baseia nos produtos já consumidos, ele tem a capacidade de recomendar itens diferentes;
    - o sistema não analisa o conteúdo do item. Sendo assim, não é necessário o conhecimento da descrição do item e, consequentemente, evita o trabalho manual que é realizado para inserir descrições de produtos não textuais;
    - recomendações baseadas na avaliação de outros usuários, podendo refletir na qualidade ou credibilidade do item.
- Desvantagens
    - problema com um usuário: quando um novo usuário utiliza o SR é necessário que ele faça um considerável número de avaliações para que o sistema entenda suas preferências;

- problema com novo produto: quando um novo produto é colocado no sistema, fica inviável a recomendação do mesmo até que tal tenha avaliações realizadas sobre ele;
- processamento elevado: caso um produto tenha uma grande quantidade de avaliações, no momento em que for realizado o cálculo dos vizinhos do usuário alvo será necessário um processamento maior;
- usuários incomuns: usuários com preferências diferentes, não muito comuns, não terão muitos vizinhos, fazendo com que as recomendações geradas possam não ser relevantes;
- mudança de interesse: caso o usuário mude de interesse, as avaliações realizadas anteriormente não lhe trarão mais nenhuma significância;
- dependência da seriedade das avaliações: para que as recomendações sejam relevantes para o usuário, as avaliações devem ser realizadas fielmente do ponto de vista do usuário.

Assim como a FBC, a FC possui desvantagens. Por isto, uma maneira encontrada para suprir tanto as desvantagens de uma como da outra é a utilização da Filtragem Híbrida, que será detalhada na próxima seção.

2.2.3. Filtragem Híbrida (FH)

Esse tipo de filtragem procura fazer uma junção das técnicas FBC e FC, combinando os pontos fortes de cada uma, suprindo suas desvantagens e, consequentemente, complementando-se. Segundo Burke (2002, p.6), um SR híbrido combina as técnicas de recomendação com o objetivo de ter um melhor desempenho com menos desvantagens. Vários SRs utilizam a abordagem híbrida, combinando os métodos da FBC e FC, fazendo uma suprir a limitação da outra (ADOMAVICIUS & TUZHILIN, 2005, p.740). A Figura 5, a seguir, apresenta os pontos fracos que são cobertos com a combinação da FBC E FC.

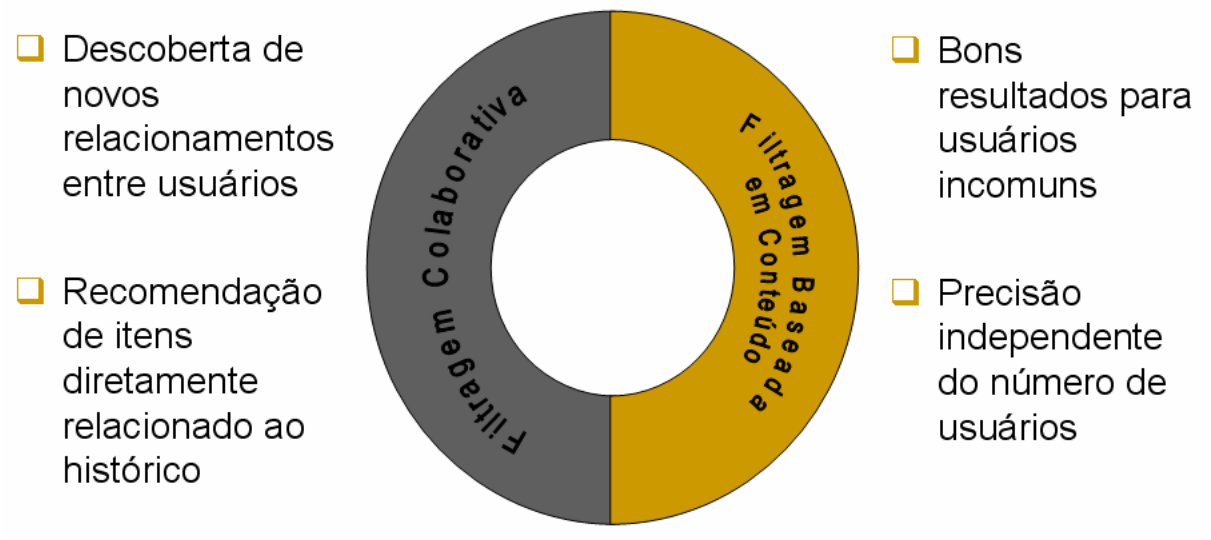


Figura 5 - Abordagem Híbrida combinando FBC e FC (CAZELLA, 2006, p.35)

Na Figura 5 são apresentados os pontos fortes das técnicas FBC e FC que, combinadas, formam a abordagem Híbrida, deixando fora os pontos fracos de cada técnica.

Existem várias formas para se implantar a Filtragem Híbrida, que realiza combinações entre duas ou mais técnicas, a fim de que as desvantagens de uma técnica utilizada isoladamente sejam supridas quando ocorrer a combinação (como no caso da utilização da FBC e FC juntas). As técnicas de hibridização expostas por Burke (2002, p.7) são divididas em:

- ponderada: nesse modelo um item que será recomendado é selecionado a partir dos resultados das combinações das pontuações das técnicas de recomendação que estão presentes no sistema. Dependendo do contexto uma técnica pode receber um peso diferente da outra. A Figura 6 apresenta uma forma de como a técnica ponderada pode trabalhar.

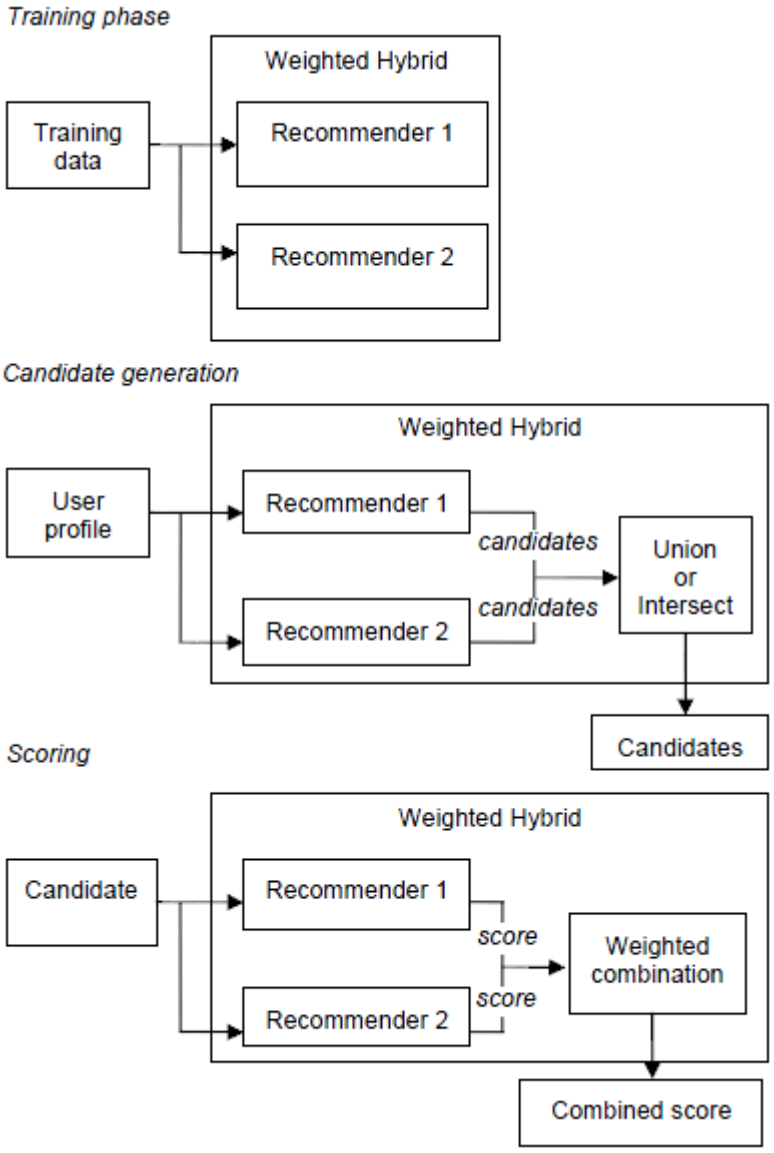


Figura 6 - Sistema Híbrido utilizando a técnica ponderada (BURKE, 2007, p.382)

A Figura 6 exemplifica uma maneira de como a técnica ponderada pode ser utilizada. A primeira etapa consiste na fase de treinamento. Nesta etapa é selecionado um conjunto de dados para que cada técnica de recomendação possa processar e gerar uma previsão. A segunda etapa consiste na geração de dois conjuntos de recomendações candidatas, utilizando, para isto, as preferências do usuário. Após a criação dos conjuntos de recomendações candidatas, é aplicada a união ou intersecção para definir o conjunto final de recomendações candidatas que será utilizado como entrada para a próxima etapa. A terceira e última etapa, aborda o processo de pontuação (*scoring)*, em que cada recomendação candidata do conjunto é avaliada pelas duas técnicas

de recomendação para gerar duas pontuações, que são submetidas a uma combinação linear. Dessa forma, os candidatos serão classificados e apresentados para o usuário de acordo com a pontuação, sendo que a recomendação com pontuação menor fica abaixo das com pontuações maiores.

- alternada: dependendo da situação o sistema alterna a técnica de recomendação utilizada, ou seja, utiliza-se um critério específico para alternar as técnicas. Um exemplo que pode ser citado como critério se refere ao nível de confiança da recomendação. Caso ele seja baixo, pode-se mudar a técnica que está sendo utilizada. A Figura 7 apresenta uma forma de como a técnica alternada pode trabalhar.

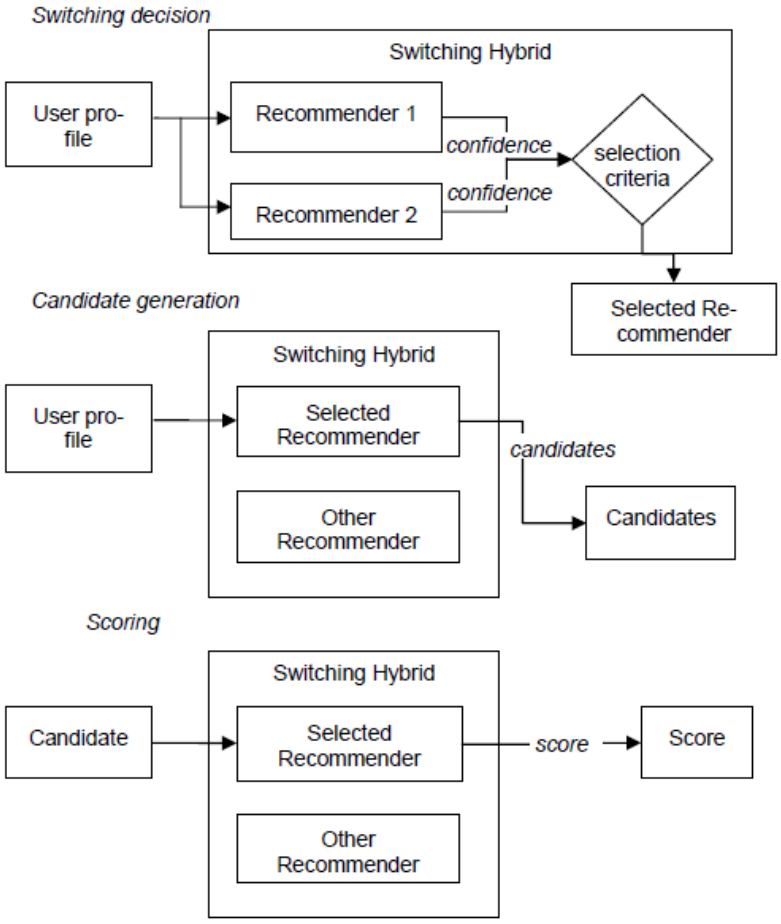


Figura 7 - Sistema Híbrido utilizando a técnica alternada (BURKE, 2007, p.385)

Conforme é apresentado na Figura 7, a técnica alternada seleciona uma técnica de recomendação única dentre as várias que o sistema híbrido tiver, baseando-se na situação da recomendação, confiável ou não, para fazer a mudança. A primeira etapa consiste na escolha da técnica de recomendação que irá gerar as

sugestões, sendo que, primeiramente, cada uma das técnicas de recomendação é aplicada sobre o conjunto de dados para formar as sugestões que possuem níveis de confiança (critério de seleção). A técnica de recomendação que gerou sugestões com nível de confiança maior será a escolhida (o critério de seleção da técnica de recomendação pode variar de acordo com o sistema). A segunda etapa tem como objetivo formar o conjunto de recomendações candidatas, sendo que é aplicada ao conjunto de dados apenas a técnica de recomendação selecionada na primeira etapa. Já na terceira etapa, cada recomendação candidata é aplicada a técnica de recomendação escolhida para gerar a pontuação do candidato, ou seja, representa o quão o candidato à recomendação é similar às preferências do usuário.

- misto: as recomendações de mais de uma técnica de recomendação são combinadas e apresentadas em conjunto. Smyth & Cotter (2000 apud BURKE 2002, p.8) citam o exemplo de um sistema que utiliza o método de recomendação híbrida, que é o Sistema PTV (Personalized TV). Este sistema utiliza a técnica FBC (baseiam-se nas descrições textuais dos programas de TV) e FC (baseiam-se nas preferências dos outros usuários) como técnicas de recomendação, sendo que os resultados gerados por essas técnicas são combinados e apresentados em conjunto. Na Figura 8 é apresentada a forma como a técnica mista pode ser implementada.

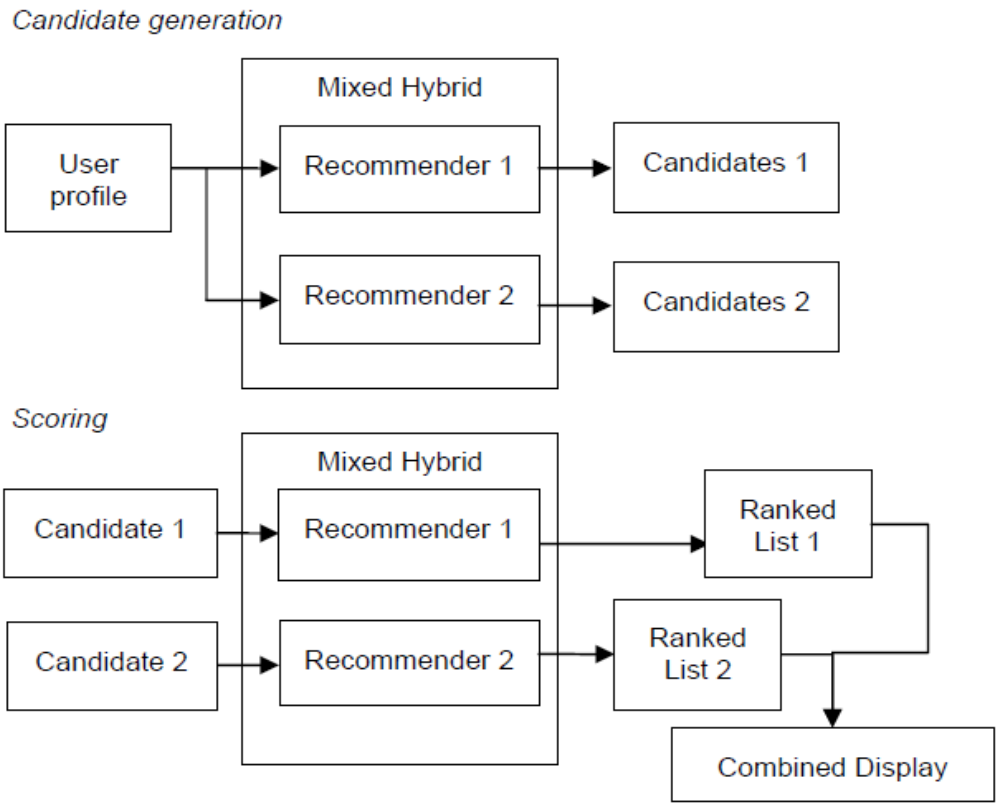


Figura 8 - Sistema Híbrido utilizando a técnica Mista (BURKE, 2007, p.384)

A Figura 8 demonstra como a técnica mista pode ser trabalhada, consistindo em três etapas. A primeira etapa é a de treinamento que, por ser a mesma da técnica ponderada, foi omitida na Figura 8 acima. A segunda etapa tem como objetivo gerar as recomendações candidatas de cada técnica de recomendação, baseando-se no perfil do usuário. Já na terceira e última etapa, as recomendações candidatas geradas na etapa anterior são utilizadas como entrada, para que sejam criadas duas listas de recomendações ordenadas, que são combinadas (formando uma lista única) e apresentadas para o usuário.

- combinação de características: informações de entrada de diferentes fontes são tratadas como colaboração adicional e são combinadas e agregadas a um algoritmo que gerará uma recomendação única. Dessa forma, o método permite que o sistema utilize as informações de uma técnica sem depender totalmente dela, pois há o complemento das informações da outra técnica. A Figura 9, a seguir, apresenta uma das possibilidades de operação da combinação de características.

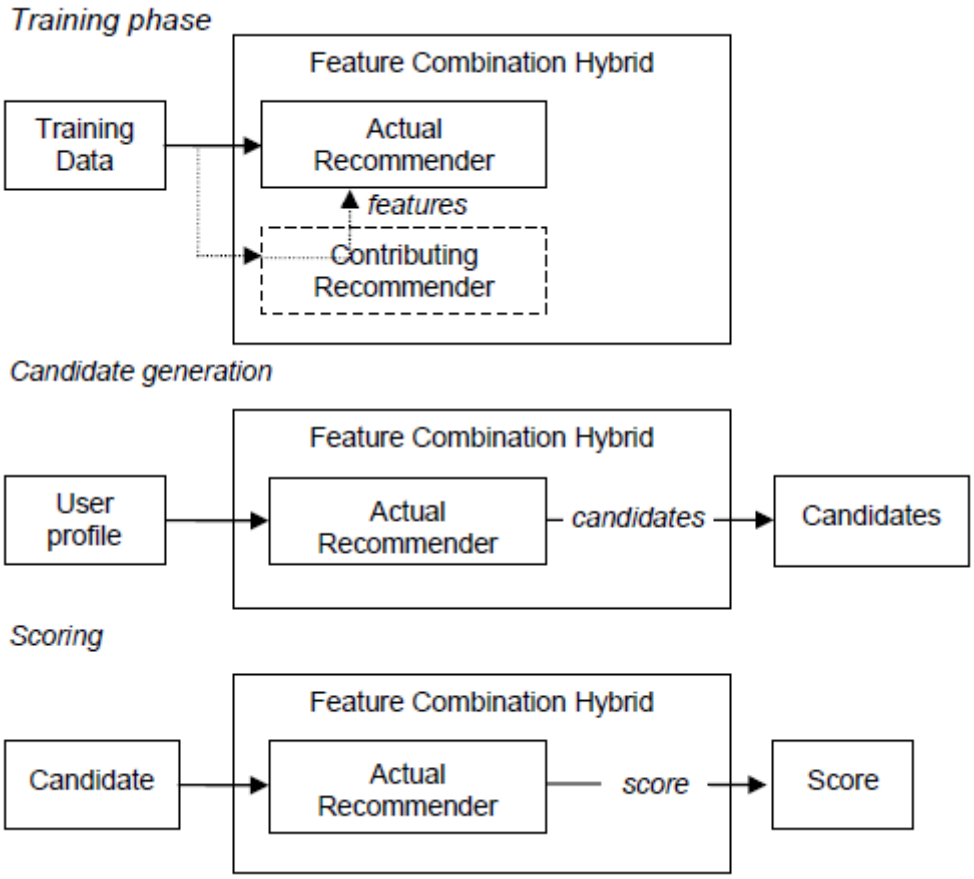


Figura 9 - Sistema Híbrido utilizando a técnica de combinação de características (BURKE, 2007, p.387)

A Figura 9 exemplifica como a combinação de características pode ser abordada. A fase de treinamento dos dados consiste em inserir as características das recomendações, que foram geradas pela técnica contribuinte, na técnica de recomendação atual, para que assim seja gerado o conjunto de dados a serem trabalhados na próxima fase. A segunda etapa é responsável pela geração do conjunto de recomendações candidatas que, por sua vez, baseia-se no perfil do usuário. Já na etapa de *scoring*, cada recomendação candidata é submetida à técnica de recomendação principal, para que seja gerada a pontuação da mesma.

- cascata: no método cascata uma técnica de recomendação é empregada para realizar a primeira fase, que é a de geração das recomendações. Na segunda fase, é aplicada uma segunda técnica de recomendação, que fará o processo de refinamento das recomendações geradas na primeira fase. A Figura 10, a seguir, expõe como a técnica cascata pode trabalhar.

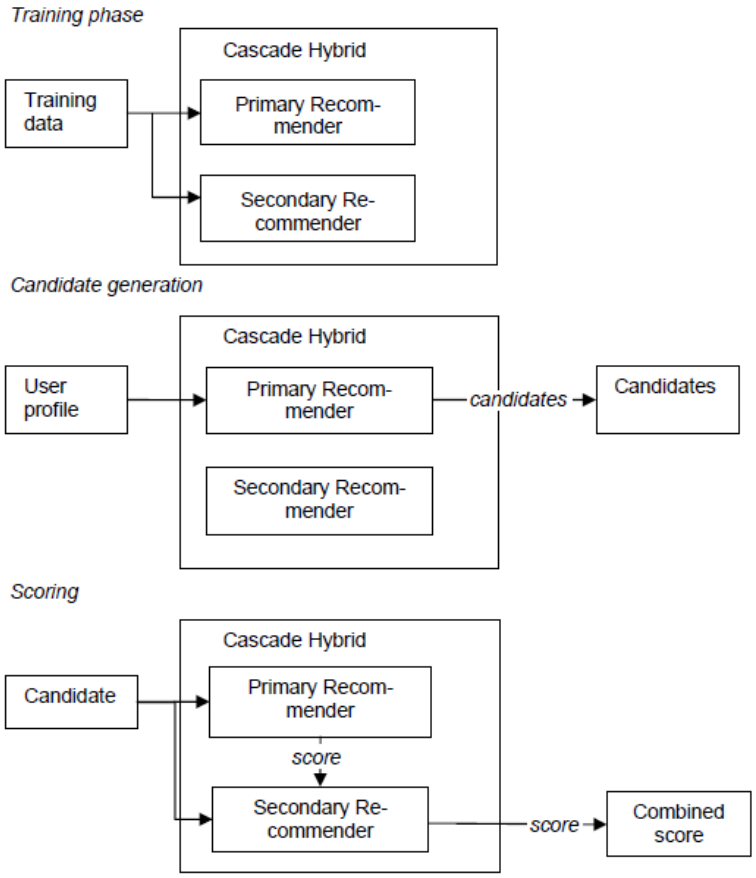


Figura 10 - Sistema Híbrido utilizando a técnica cascata (BURKE, 2007, p.390)

Como pode ser observada na Figura 10, a primeira etapa consiste na fase de treinamento, a qual é responsável por selecionar um conjunto de dados e aplicar as técnicas de recomendação envolvidas na abordagem híbrida. A segunda etapa, responsável pela geração das candidatas a recomendação, é utilizado somente a técnica primária de recomendação, baseando-se no perfil do usuário. Apenas na terceira etapa que a técnica secundária de recomendação será utilizada, com o objetivo de refinar a pontuação da recomendação candidata. Por exemplo, caso duas recomendações candidatas tenham obtido a mesma pontuação, elas serão submetidas à técnica de recomendação secundária e essa pontuação será refinada, podendo ser mudada.

- aumento de característica: uma técnica é utilizada para produzir uma classificação dos itens e a saída gerada serve de entrada para a próxima técnica de recomendação. A Figura 11 apresenta como a técnica de aumento de característica pode ser implementada.

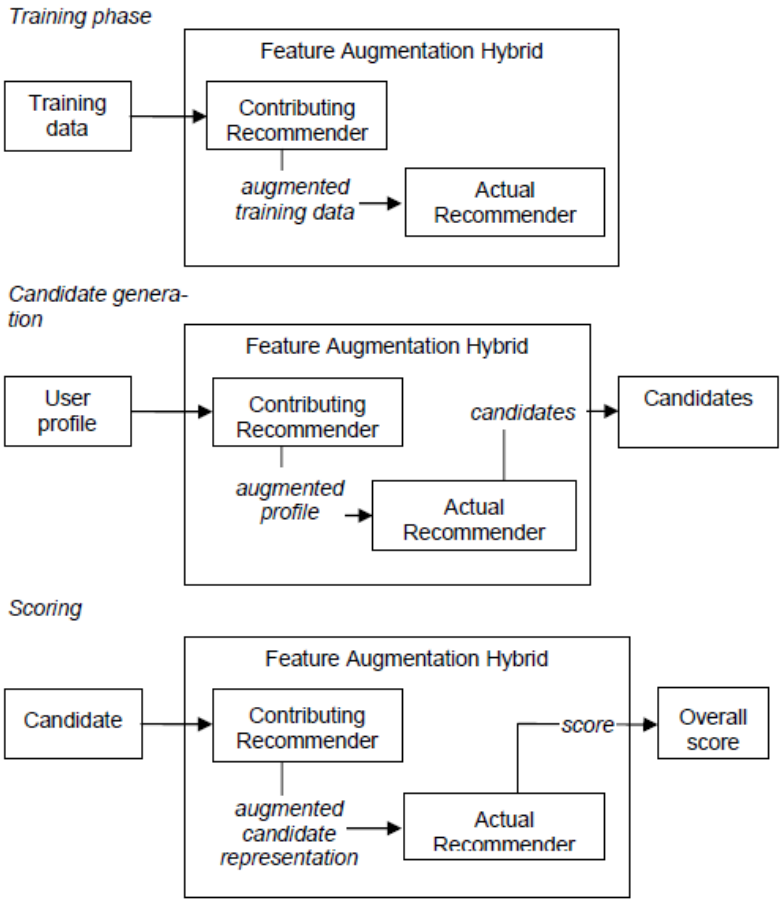


Figura 11 - Sistema Híbrido utilizando a técnica de aumento de característica (BURKE, 2007, p.388)

A Figura 11 demonstra como a técnica de aumento de característica pode ser implementada. A primeira fase tem como objetivo selecionar os dados a serem processados, sendo que a técnica de recomendação contribuinte é aplicada sobre os dados com o objetivo de gerar uma classificação de itens, que servirão como entrada para a técnica de recomendação principal. A próxima etapa é responsável pela geração das recomendações candidatas que, para realizar essa tarefa, utiliza a técnica de recomendação contribuinte para gerar uma classificação das preferências do usuário. Assim, juntamente com a classificação de itens gerada na primeira etapa, esses dados serão processados pela técnica de recomendação principal. Na etapa de *scoring* cada recomendação candidata é submetida à técnica de recomendação contribuinte, com o objetivo de gerar uma classificação para o item, que será processado pela técnica principal e gerada uma pontuação global para o mesmo.

- *meta-level*: é uma forma de combinar as duas técnicas de recomendação (FBC e FC), onde o modelo gerado por uma das duas técnicas serve de entrada para outra. Desta forma, o modelo que foi obtido na utilização de uma técnica gera recursos que servirão de entrada para outro algoritmo. A Figura 12, a seguir, apresenta uma forma de implementar a combinação de características.

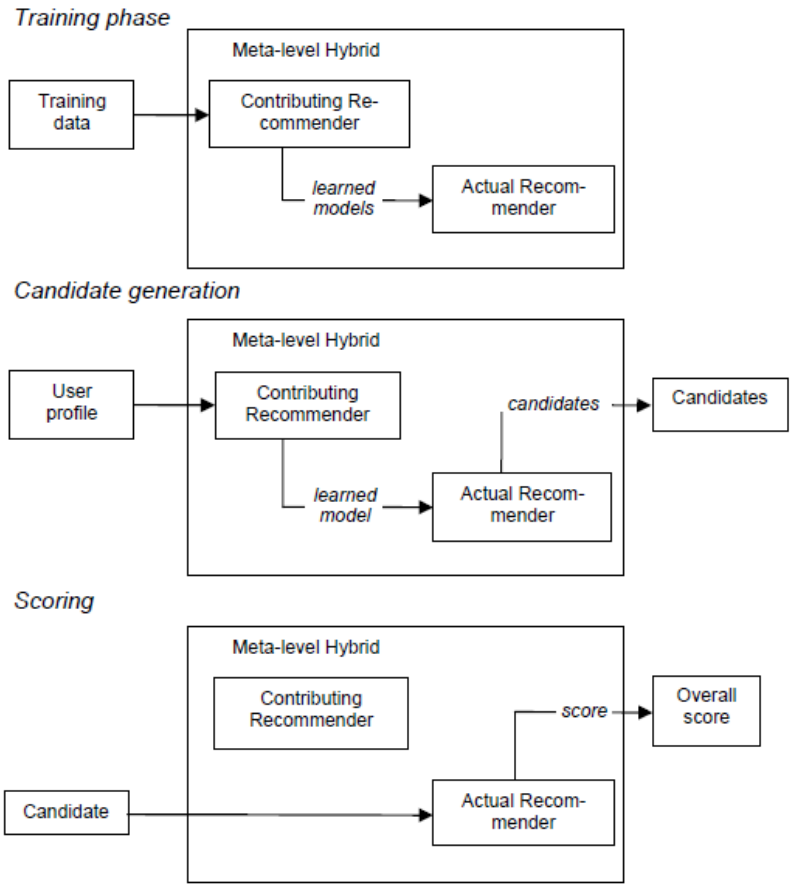


Figura 12 - Sistema Híbrido utilizando a técnica *meta-level* (BURKE, 2007, p.391)

Conforme apresentado na Figura 12, a técnica *meta-level* tem como primeira fase o treinamento. Nesta fase, é necessário selecionar um conjunto de dados, a qual a técnica de recomendação contribuinte é aplicada, tendo como saída um modelo de dados que será utilizado como entrada para a técnica de recomendação principal. A segunda fase consiste na geração das recomendações candidatas. Nesta segunda fase, a técnica de recomendação contribuinte é aplicada para gerar o modelo de dados que servirá de entrada para a técnica principal, para que a mesma gere as recomendações candidatas. Na terceira etapa, apenas a técnica de recomendação principal é utilizada com o objetivo de gerar uma pontuação global para cada recomendação candidata.

Na próxima seção serão apresentados os materiais e métodos utilizados durante o desenvolvimento desse trabalho.

3 MATERIAIS E MÉTODOS

Nesta seção será apresentada a metodologia e os materiais utilizados para desenvolver o trabalho que, juntamente com as orientações, permitiram a conclusão do mesmo.

3.1. Materiais

Os materiais utilizados no desenvolvimento deste trabalho podem ser divididos em duas partes:

- Fontes Bibliográficas – as fontes utilizadas para o desenvolvimento da revisão de literatura foram: monografias, teses, dissertações, artigos e publicações científicas. Os materiais que foram citados neste trabalho podem ser observados na seção 6.
- Software – para implementação da ferramenta de recomendação foi utilizado o ambiente de desenvolvimento NetBeans 6.9, juntamente com a API[1] *Apache Mahout Taste*, implementado com a linguagem de programação Java. As tecnologias e API utilizadas serão melhor detalhadas nas subseções abaixo.

3.1.1. Apache Mahout Taste

Este trabalho utilizou a API[1] *Apache Mahout Taste* para possibilitar a geração das recomendações de acordo com a Filtragem Colaborativa. Esta API é um projeto de código aberto da *Apache Software Foundation*, que fornece uma biblioteca Java para aprendizagem por máquina escalável que suporta grandes quantidades de dados. Os principais algoritmos suportados pela API são: *cluster*, classificação e filtragem colaborativa (INGERSOLL, 2009, ONLINE).

A versão da API *Apache Mahout*, que será utilizada neste trabalho, é a mahout-core 0.3 que foi obtida no site: http://grepcode.com/snapshot/repo1.maven.org/maven2/org.apache.mahout/mahout-core/0.3/.

[1] *Application Programming Interface*

Para utilizar esta API é necessário baixar os seguintes pacotes:

- mahout-core-0.3.jar, disponível em: http://grepcode.com/snapshot/repo1.maven.org/maven2/org.apache.mahout/mahout-core/0.3/;
- uncommons-math-1.0.2.jar, disponível em: http://mirrors.ibiblio.org/pub/mirrors/maven2/org/apache/mahout/uncommons/math/uncommons-math/1.0.2/uncommons-math-1.0.2.jar;
- slf4j-nop-1.6.1.jar, disponível em: http://grepcode.com/snapshot/repo1.maven.org/maven2/org.slf4j/slf4j-nop/1.6.1;
- slf4j-api-1.6.1.jar, disponível em: http://grepcode.com/snapshot/repo1.maven.org/maven2/org.slf4j/slf4j-api/1.6.1.

Posteriormente, é necessário adicioná-los ao projeto, clicando com o botão direito na pasta “bibliotecas” do projeto e escolhendo a opção adicionar JAR.

Dentro da biblioteca mahout-core 0.3 existem várias classes implementadas, das quais, neste trabalho, foram utilizadas:

- DataModel – é uma interface[2] que pode ser implementada pelo MySQLJDBCDataModel e outras classes, e que possibilita representar as informações que serão utilizadas para gerar as recomendações. A utilização da classe MySQLJDBCDataModel é para os casos em que irá utilizar uma base de dados MySql, por meio de um *driver* JDBC (*Java Database Connectivity*) para MySql. O banco de dados deve possuir uma tabela com as seguintes colunas e tipos: chave primária do Usuario (tipo:long), chave primária do Item (tipo:long) e Preferencia (tipo:float). Caso não queira utilizar um banco de dados, pode-se utilizar um arquivo de dados com a extensão txt, dat, csv entre outros. O arquivo deve possuir o formato dividido em colunas como: chave primária do Usuario, chave primária do Item e Preferencias. Assim, quando se utiliza um arquivo como base de dados, a implementação da classe FileDataModel é utilizada. A Figura 13, a seguir, apresenta exemplo da utilização da interface DataModel com a classe MySQLJDBCDataModel.

[2] Consiste em uma classe que contém um conjunto de métodos abstratos que podem ser implementados por outra classe

```
DataModel model = new MySQLJDBCDataModel(data,"ratings","IdUser","IdItem","Preferences");
```

Figura 13 - Implementação com MySQLJDBCDataModel

A Figura 13 apresenta como é criado um modelo de dados através da implementação via MySQLJDBCDataModel, onde o primeiro parâmetro informado é a base de dados que contém as informações; o segundo, a tabela que contém as avaliações; o terceiro, quarto e quinto, são as colunas que contém o id do usuário, id do item e a avaliação, respectivamente. A Figura 14, a seguir, apresenta a utilização da interface DataModel com a classe FileDataModel.

```
DataModel model = new FileDataModel(new File("data.txt"));
```

Figura 14 - Implementação com FileDataModel

A Figura 14 demonstra mais uma maneira de utilizar a Interface DataModel que, neste caso, foi implementada pela a classe FileDataModel, sendo que para criar o modelo de dados é necessário informar como parâmetro apenas o arquivo que servirá como a base de dados.

- UserSimilarity – esta interface pode ser implementada pela classe PearsonCorrelationSimilarity, sendo que este será o algoritmo de correlação de usuário (apresentado na seção 2.2.2). Esta interface irá calcular a semelhança entre os perfis de dois usuários, retornando valores entre -1,0 e 1,0, sendo que 1,0 representa a semelhança total. A Figura 15, a seguir, apresenta um exemplo de como é instanciado um objeto utilizando o algoritmo PearsonCorrelationSimilarity.

```
UserSimilarity usersSimilarity = new PearsonCorrelationSimilarity(model);
```

Figura 15 – Instanciação de um objeto com o algoritmo PearsonCorrelationSimilarity

A Figura 15 demonstra um exemplo da instanciação de um objeto da classe PearsonCorrelationSimilarity, que é uma das classes que implementam a interface UserSimilarity, e recebe como parâmetro um modelo de dados do tipo DataModel.

- UserNeighborhood – as implementações para essa interface têm o objetivo de determinar a vizinhança de usuários semelhantes ao usuário alvo, sendo que as

recomendações serão baseadas nesse conjunto de usuários semelhantes. Na instanciação do objeto pode-se indicar a quantidade limite de vizinhos (usuários com as preferências similares) a qual o sistema irá se basear para calcular a recomendação. Uma das classes que implementa essa interface é a NearestNUserNeighborhood, que possui métodos para calcular um conjunto de "n" usuários que sejam semelhantes ao usuário alvo. A Figura 16, abaixo, expõe um exemplo de como essa interface pode ser utilizada.

```
UserNeighborhood vizinhanca = new NearestNUserNeighborhood(4,usersSimilarity,model);
```

Figura 16 - Utilização da Interface UserNeighborhood

A Figura 16 apresenta um exemplo de como a interface UserNeighborhood é utilizada com a classe NearestNUserNeighborhood. No construtor desta interface é passado como primeiro parâmetro o número máximo de vizinhos sobre o qual irá se basear a recomendação; no segundo parâmentro o algoritmo que será utilizado para calcular a correlação entre os usuário (definida na criação do objeto do tipo UserSimilarity) e o modelo de dados do tipo DataModel.

- Recommender - as implementações para essa interface têm como objetivo gerar as recomendações para o usuário. Umas das classes que implementam essa interface é a GenericUserBasedRecommender, que solicita em seu construtor um DataModel, UserNeighborhood e UserSimilarity. Possui o método recommend, que recebe como parâmetro a chave primária do usuário sobre a qual as recomendações serão calculadas e a quantidade de recomendações a qual se deseja ter. A Figura 17, a seguir, apresenta um exemplo da utilização da interface Recommender e da classe GenericUserBasedRecommender para gerar as recomendações.

```
1    Recommender recomendacao = new GenericUserBasedRecommender(model, vizinhanca, usersSimilarity);
2    List<RecommendedItem> recommendations = recomendacao.recommend(idUsuario, qtdRecomendacao);
```

Figura 17 - Utilização da Interface Recommender

A Figura 17 demonstra um exemplo de como utilizar a interface Recommender juntamente com a classe GenericUserBasedRecommender para gerar as

recomendações. Na linha 1 é instanciado um objeto do tipo GenericUserBasedRecommender por meio de seu construtor, que recebe um parâmetro do tipo DataModel, UserNeighborhood e UserSimilarity. Já na linha 2 ocorre a geração das recomendações, sendo que essas recomendações são geradas no momento em que o método recomend() é chamado, tendo como parâmetro a chave primária do usuário para o qual se deseja gerar as recomendações e a quantidade máxima de recomendações a serem retornadas. Este método retorna uma lista de itens que são do tipo RecommenderItem.

- RecommenderItem - é o tipo de objeto que é retornado quando o método Recommender.recomend() é chamado para gerar as recomendações. Para acessar o conteúdo do objeto, têm-se dois métodos: getItemID() e getValue(), sendo que o primeiro método retorna chave primária do objeto e o segundo, o valor da preferência (possível valor predito na geração da recomendação, que o usuário alvo poderia avaliar o item).

Dessa forma, com as classes e interfaces descritas acima, é possível realizar todo o processo de implementação da Filtragem Colaborativa que será implantada neste trabalho. Na próxima seção será abordada a metodologia seguida para o desenvolvimento deste trabalho.

3.2. Metodologia

Este trabalho, para ser desenvolvido, envolveu diversas etapas. A primeira envolveu um estudo aprofundado sobre os conceitos básicos do Sistema de Recomendação (definição, contexto a qual está inserido e como pode contribuir de maneira positiva para a empresa que o utiliza). A partir disso, pôde-se compreender como é o funcionamento de um SR e identificar as técnicas que fazem a geração da recomendação.

Ainda nesta fase, partiu-se para a compreensão mais detalhada de cada uma das técnicas de recomendações (Filtragem Baseada em Conteúdo, Filtragem Colaborativa e Filtragem Híbrida), que podem ser utilizadas no SR. Dentro deste contexto, nos estudos sobre a FBC, foi necessário entender conceitos sobre Recuperação da Informação, como: modelos clássicos, otimização e limpeza de texto (*Stopword* e *Stemming*) e a fórmula do co-seno. Já na Filtragem Colaborativa, foram estudados conceitos sobre correlação de usuários e predição de avaliações,

bem como a técnica *k-nearest-neighbor* e a fórmula do Coeficiente de Correlação de Pearson. Por fim, na Filtragem Híbrida foram estudados e compreendidos os modelos de FH existentes.

Ao finalizar os estudos sobre os conceitos, foram realizados testes manuais sobre as técnicas, que envolveram cálculos como ângulo do Co-seno (FBC) e Coeficiente de Correlação de Pearson (FC), pois só a partir da realização desses testes seria possível compreender detalhadamente o funcionamento de cada uma das técnicas de recomendação. Portanto, esta etapa foi primordial para o andamento do trabalho, pois são as técnicas que realizam o processo de seleção das sugestões, sendo assim o ponto principal de um SR. Assim, concluída a fase de compreensão das técnicas, pode-se definir quais técnicas melhor se aplicariam para o desenvolvimento da ferramenta proposta neste trabalho.

A próxima etapa envolveu a escolha da linguagem de programação, IDE (*Integrated Development Environment*) e API que iriam auxiliar no desenvolvimento da ferramenta. No caso, foi escolhido a linguagem de desenvolvimento Java com a IDE NetBeans 6.9.1. Para auxiliar no processo do desenvolvimento da FC, foi escolhida a API *Apache Mahout*. Desta forma, para utilização da API *Apache Mahout* foi necessário um estudo sobre como utilizá-la, e que métodos seriam necessários para a implementação da FC.

Posteriormente, foi definido como seria o processo de desenvolvimento da Filtragem Baseada em Conteúdo, que utiliza conceitos da Recuperação da Informação. Inicialmente, foi definido o tipo de Recuperação da Informação (Filtragem), mais especificamente o modelo clássico, baseado no Espaço Vetorial. Depois foi definido como seriam desenvolvidas as técnicas de otimização e limpeza de texto. Para auxiliar no desenvolvimento da técnica de *Stemming*, escolheu-se uma biblioteca chamada PTStemmer, que possui vários algoritmos para aplicação de *stemming* na língua portuguesa, e está disponível em: http://code.google.com/p/ptstemmer/. Para a implementação da técnica de retirada de *stopwords* e do modelo espaço vetorial, buscou-se alguma API que pudesse auxiliar no desenvolvimento, porém não foi encontrada. Desta forma, foi necessário fazer as devidas implementações para finalizar a FBC.

Por fim, foi realizada a junção de todas as etapas de desenvolvimento descritas, resultando na ferramenta. Ao final, foram realizados testes de

funcionalidade e de utilização na referida ferramenta, sendo que os testes foram divididos da seguinte forma:

- Base de Teste: foi desenvolvida uma base de dados para testes composta pelas tabelas produtos (10 produtos), avaliações (51 avaliações) e usuários (10 usuários).
- Filtragem Colaborativa: primeiramente foram feitos testes sobre a FC, por ser a primeira técnica utilizada na Filtragem Híbrida, e por ser ela a responsável por gerar o conjunto de recomendações. Dessa forma, foram realizados cálculos de forma manual para encontrar os usuários com perfis semelhantes ao do usuário alvo, e cálculos de predição para saber quais produtos seriam retornados como recomendações. Dessa forma, foi possível comparar o resultado do teste ao retornado pela ferramenta de recomendação;
- Filtragem Baseada em Conteúdo: com o conjunto de recomendações retornadas pela FC, aplicaram-se técnicas de *Stopwords* e *Stemming* para otimizar o texto, calculou-se a frequência dos termos e se realizou de forma manual o cálculo da similaridade entre um produto retornado pela FC com o perfil do usuário.
- Comparação: após todos os cálculos comparou-se o resultado retornado pelos testes com o retornado pela ferramenta de recomendação e se verificou que o conjunto de recomendações e a ordem retornada foram iguais.

4 RESULTADOS E DISCUSSÃO

Os conceitos estudados sobre sistemas de recomendação e técnicas de filtragem tiveram como objetivo obter fundamentação teórica para permitir que o Sistema de Recomendação desenvolvido pudesse obter de maneira eficaz recomendações personalizadas, que atendam as necessidades dos usuários. Assim, para que essa eficácia fosse atendida a escolha da técnica de recomendação híbrida assumiu um papel importante por possibilitar a utilização de duas ou mais técnicas em conjunto para selecionar as recomendações. Desta forma, este capítulo é voltado para a apresentação do que foi desenvolvido.

4.1. Arquitetura da Ferramenta

A ferramenta irá gerar recomendações partindo do princípio que o site cliente já terá os perfis dos usuários. Dessa forma, esta ferramenta poderá ser acoplada a um site já existente, servindo como um módulo adicional para geração de recomendações. A Figura 18, a seguir, apresenta a arquitetura proposta para o SR.

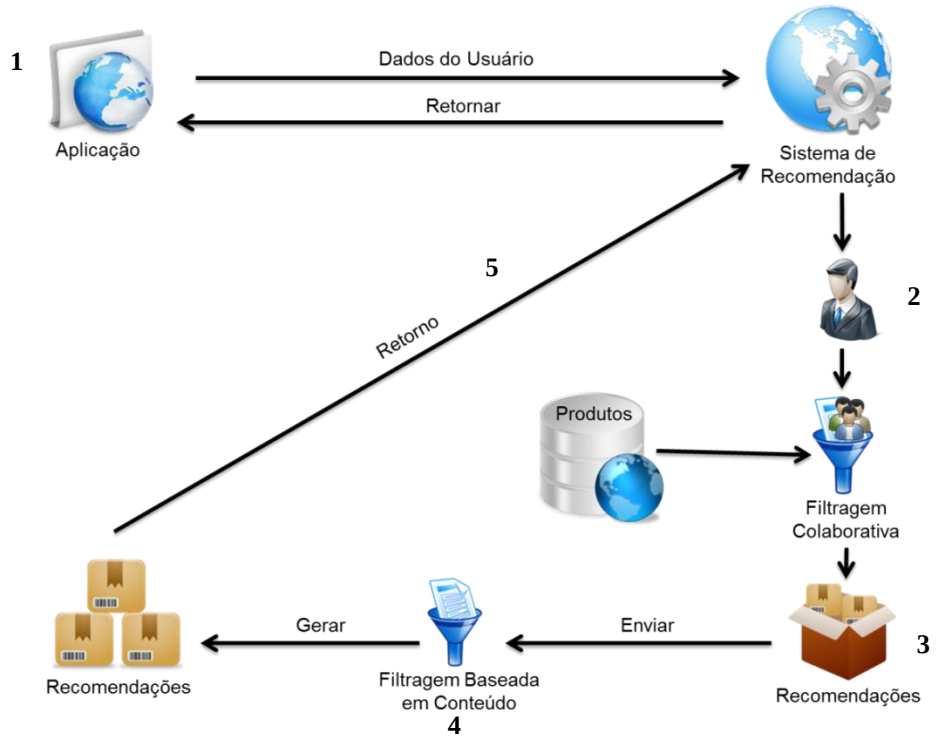


Figura 18 - Arquitetura do Sistema de Recomendação

Como pode ser observado na Figura 18, a arquitetura do sistema é dividida em cinco partes. A primeira etapa consiste na chamada do Sistema de Recomendação por parte da aplicação cliente, sendo que a mesma envia os dados do usuário que se deseja obter a recomendação, juntamente com a quantidade máxima de recomendações que se deseja receber. Nesta solicitação também é necessário passar os dados para a conexão com o banco de dados do site, que será onde a ferramenta de recomendação irá buscar os dados necessários para sugerir as recomendações. Já na segunda etapa, é selecionado o perfil do usuário no banco de dados e chamado o processo de Filtragem Colaborativa para gerar o primeiro conjunto de recomendações. Após a etapa de FC, o próximo passo (terceira etapa) é disponibilizar o conjunto de recomendações geradas na etapa anterior como entrada para a técnica seguinte, Filtragem Baseada em Conteúdo. Na quarta etapa, a FBC irá dar início ao processo de Recuperação da Informação que irá gerar outro conjunto de recomendações, com um nível de personalização mais refinado. Na quinta e última etapa, é retornado para o site o conjunto de recomendações.

4.1.1. Restrições da Ferramenta

A ferramenta de recomendação desenvolvida para este trabalho tem como principais restrições os itens descritos abaixo:

- Banco de dados – esta ferramenta foi desenvolvida para trabalhar com o banco de dados MySql, por ser uma necessidade da API *Apache Mahout* (utilizada para desenvolver a Filtragem Colaborativa).
- Tabelas – é necessário que a aplicação cliente tenha no mínimo duas tabelas: uma que contenha as avaliações dos usuários sobre os produtos e a outra que contenha os produtos vendidos pela loja. É necessário que as chaves primárias das tabelas envolvidas sejam do tipo numérico.

A próxima seção irá demonstrar, de forma detalhada, todas as etapas apresentadas na Figura 18.

4.1.2. Aplicação Cliente

A aplicação cliente irá utilizar o sistema de recomendação como um módulo, que será integrado ao site através de um arquivo jar. Desta forma, para utilizar os recursos da ferramenta, o cliente terá que fazer a chamada de dois métodos:

- ConectarBanco(): este será o método que irá fazer a conexão do Sistema de Recomendação com a base de dados do site. Como parâmetros de entrada ele receberá os seguintes dados:
    - String - usuário do banco;
    - String – senha;
    - String - nome do servidor;
    - Int - número da porta;
    - String - nome da base de dados que contém as informações.

  Como saída, o método retornará o objeto de conexão, que é exigido pelo método recomendar. Esse objeto retornado é do tipo MysqlConnectionPoolDataSource.
- Recomendar(): este método será o responsável por selecionar as recomendações. Como parâmetros de entrada ele receberá os seguintes dados:
    - long - chave primária do usuário;
    - int - quantidade máxima de recomendações;
    - MysqlConnectionPoolDataSource - conexão com o banco de dados (retorno do método anterior);
    - String - nome da tabela que contém as avaliações dos produtos;
    - String - nome da coluna que contém a chave primária do usuário da tabela de avaliações;
    - String - nome da coluna que contém o id do produto avaliado da tabela de avaliações;
    - String - nome da coluna que contém a nota da tabela de avaliações;
    - String - nome da tabela que contém os produtos que foram ou não avaliados;
    - String - nome da coluna que contém a descrição do produto;
    - String - nome da coluna que contém o id do produto na tabela de produto;
    - String - nome da tabela que irá armazenar o conjunto de recomendações;

- String - nome da coluna da tabela de armazenamento das recomendações, que irá armazenar o id do usuário;
- String - nome da coluna da tabela de armazenamento das recomendações, que irá armazenar o id do produto;
- String - nome da coluna da tabela de armazenamento das recomendações, que irá armazenar a data de geração das recomendações.

Este método irá fazer a chamada para gerar as recomendações, que inclui os métodos da Filtragem Baseada em Conteúdo e Colaborativa, que serão as técnicas que irão analisar e processar as recomendações.

Na próxima seção será abordado como o Sistema de Recomendação irá trabalhar para realizar a seleção das recomendações.

4.1.3. Sistema de Recomendação

O Sistema de Recomendação desenvolvido tem o objetivo de recomendar os itens de acordo com o perfil de cada usuário, com intuito de gerar recomendações refinadas. Por isto, ele irá trabalhar com a Filtragem Híbrida como meio de gerar as recomendações para cada usuário. Portanto, para iniciar o processo de recomendação, a ferramenta de recomendação receberá da aplicação cliente os dados para conexão com o banco de dados, que contém o perfil do usuário, bem como a identificação do usuário e a quantidade de recomendações que se deseja receber, conforme dito na seção anterior. O método ConectarBanco(), utilizado para conectar com o banco de dados e retornar o objeto da conexão, e o Recomendar() podem ser chamados, conforme a Figura 19, a seguir.

```
1 MysqlConnectionPoolDataSource data = SR.ConectarBanco("root","123", "localhost", 3306,"sistemarecomendacao");
2 List<Long> recomendacoes = SR.Recomendar(2, 10, data,"ratings","IdUser","IdItem","Preferences",
3     "movies","Genres","MovieId","recomendacoes","IdUsuario","IdProduto","DataCadastro");
```

Figura 19 - Chamada dos métodos ConectarBanco() e Recomendar()

Com pode-se observar na Figura 19, os dois métodos estão na classe SR, que é a classe geral do sistema de recomendação. Na chamada do método ConectarBanco() é retornado um objeto do tipo data, que servirá de entrada para o

método Recomendar(). O método recomendar retornará uma lista de recomendações com o id de cada produto.

Vale ressaltar que, com o objetivo de otimizar o processo de recomendação, a ferramenta irá armazenar o conjunto de recomendações gerados para cada usuário, após ser solicitado pela primeira vez. Desta forma, caso seja solicitado um conjunto de recomendações para um mesmo usuário e este não tenha tido alterações em seu perfil, o sistema não irá gerar as recomendações novamente e, sim, buscar no banco de dados o conjunto de recomendações gerado anteriormente. Para que seja possível a ferramenta de recomendação armazenar os conjuntos de sugestões selecionados, é necessário que a aplicação cliente crie uma tabela contendo o id do usuário e o id do produto, ambos do tipo que receba um valor inteiro, e a data de cadastro do tipo DateTime. As recomendações que serão armazenadas neste banco serão retornadas para o usuário contanto que o período em que elas foram cadastradas no banco não exceda o prazo de sete dias. Caso o período de sete dias seja excedido, na próxima solicitação de recomendações o sistema irá gerar novas sugestões para o usuário e armazená-las novamente. A Figura 20, a seguir, apresenta a estrutura da tabela a ser criada.

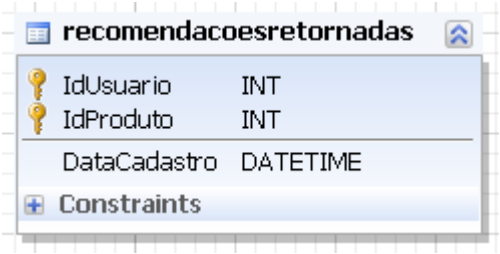


Figura 20 - Tabela para Armazenamento do conjunto de recomendações retornadas

Conforme pode ser observado na Figura 20, a tabela é composta por duas chaves primárias, IdUsuário (representando a identificação do usuário) e IdProduto (representando a identificação do produto), e um campo em que será armazenada a data em que a recomendação foi gerada.

4.1.4. Perfil do Usuário

O perfil do usuário é uma das partes mais importantes para a ferramenta obter a seleção das recomendações. É a partir deste perfil que as sugestões são escolhidas, pelo fato de se tratar de recomendações personalizadas. O perfil do usuário será constituído inicialmente a partir do pressuposto de que todas as avaliações feitas pelo usuário estarão com uma estrutura similar a apresentada na Figura 21, a seguir.

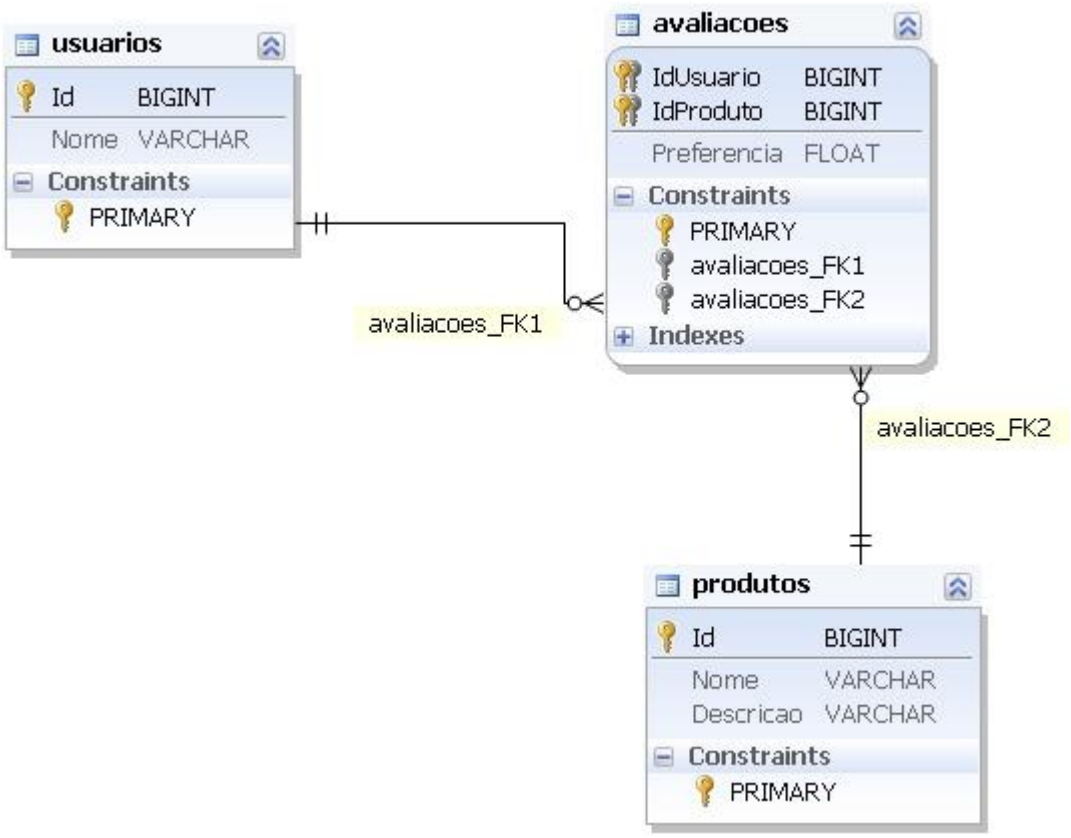


Figura 21 - Estrutura do Perfil do Usuário

Para que seja possível a utilização da ferramenta de recomendação desenvolvida neste trabalho, é necessário ter uma estrutura de perfil do usuário que seja semelhante a apresentada na Figura 21. A estrutura do perfil do usuário deve ser constituída pelas seguintes tabelas:

- usuários – deverá conter os dados de cada usuário, sendo que a chave primária do usuário pode ser de qualquer tipo que aceite um numero inteiro;
- produtos – deverá conter os dados dos produtos, como id de qualquer tipo que aceite um número inteiro, nome e descrição do produto com o tipo de dados literal;

- avaliações – terá todas as avaliações realizadas pelos usuários no site, sendo que a tabela deve conter obrigatoriamente os campos que irão conter o id do usuário do tipo `BIGINT`, `INT` ou qualquer tipo que aceite um número inteiro; id do produto do tipo `BIGINT`, `INT` ou qualquer tipo que aceite um número inteiro; e avaliação do tipo `FLOAT`. Esta tabela é a única que deve obrigatoriamente conter as colunas citadas acima, pois será utilizada na aplicação da Filtragem Colaborativa. Dessa forma, as ligações entre as tabelas são opcionais, podendo haver apenas ligações lógicas.

Para aplicar a Filtragem Baseada em Conteúdo, será necessário que haja uma junção da descrição de todos os produtos que o usuário (usuário alvo das recomendações) já avaliou. Dessa forma, o perfil do usuário para a FBC será um pouco diferente do utilizado na Filtragem Colaborativa, pois ele será composto apenas por descrições de produtos. No entanto, vale ressaltar que para se obter essa junção das descrições é necessário ter a estrutura exposta na Figura 22, pois são dos produtos avaliados que as descrições serão retiradas. A Tabela 7, a seguir, demonstra uma representação do perfil do usuário por avaliações.

Tabela 7- Representação do Perfil do Usuário por Avaliações

IdUsuario	IdProduto	Preferencia
1	2	4
2	2	5
1	3	3
3	2	5
4	4	2
1	5	1

Conforme pode ser observado na Tabela 7, a representação do perfil consiste por uma tabela que contém várias avaliações de diversos usuários, sendo que, neste caso, o usuário selecionado é o com `IdUsuario` “1”. Assim, as linhas que contêm esse id serão selecionadas.

A partir das avaliações feitas pelo usuário “1”, é possível realizar a junção das descrições dos produtos para formar o perfil que será utilizado na Filtragem

Baseado em Conteúdo. A partir da Tabela 7 e da Tabela 8, a seguir, é possível obter o perfil descritivo do usuário, conforme é apresentado na Figura 22.

Tabela 8 - Tabela de Produtos

IdProduto	NomeProduto	Descricao
2	Item A	Tem uma tela de alta definição de 14 polegadas na diagonal.
3	Item B	Tem o processador Intel Core i5 que torna mais rápido o trabalho com vários aplicativos.
4	Item C	Proporciona a liberdade de conectar aparelhos de áudio e vídeo ao seu televisor sem utilizar fios
5	Item D	Explosão de cores, interatividade e tecnologia como você nunca viu.

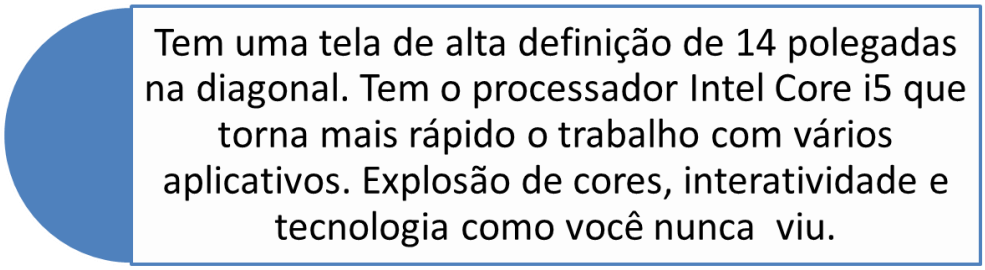


Figura 22 - Perfil utilizado pela Filtragem Baseada em Conteúdo

Como pode ser observado na Figura 22, foi feita uma junção das descrições de todos os produtos que o usuário “1” avaliou, para formar o perfil descritivo do usuário.

Dessa forma, a etapa inicial da formação do perfil (na forma das avaliações) será utilizada para a aplicação da técnica de Filtragem Colaborativa, que necessita

apenas das avaliações dos produtos para gerar as recomendações. Já para a aplicação da Filtragem Baseada em Conteúdo, será necessário o perfil descritivo, pois ela se baseia no perfil do usuário, formado por descrições de conteúdo dos produtos para analisar a similaridade entre um produto e as preferências. Após a coleta do perfil, a próxima etapa consiste na implementação da primeira técnica que irá compor a Filtragem Híbrida, que será abordada na próxima seção.

4.1.5. Filtragem Híbrida

De acordo com a análise das técnicas de recomendação, optou-se por desenvolver a ferramenta de recomendação utilizando a Filtragem Híbrida, pois é a técnica que mescla duas ou mais técnicas, com o objetivo de ter melhores resultados no processo de seleção das recomendações. No caso, o modelo de implementação escolhido para esta referida técnica foi o Cascata.

O modelo Cascata utiliza duas técnicas de recomendação: a primeira técnica tem como objetivo gerar um conjunto de recomendações; já a segunda técnica é utilizada como meio de refinar (ordenar) as recomendações geradas pela primeira. Desta forma, esta técnica será utilizada neste trabalho para realizar um refinamento das recomendações e organizar as sugestões por ordem de similaridade com o perfil do usuário. Na Figura 23, a seguir, será apresentada a arquitetura criada para a Filtragem Híbrida.

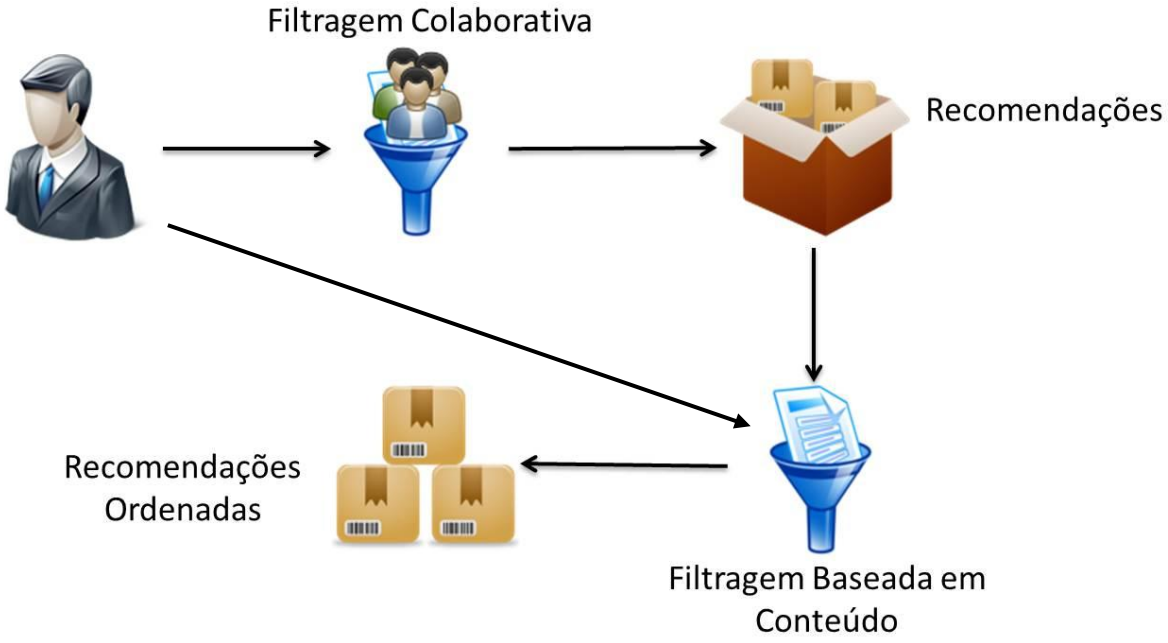


Figura 23 – Arquitetura do Modelo Híbrido

Como pode ser observado na Figura 23, o processo da Filtragem Híbrida irá iniciar pelo perfil do usuário que será requisitado pela Filtragem Colaborativa para gerar o primeiro conjunto de recomendações. Optou-se pela utilização da FC para gerar a lista inicial das recomendações pelo fato desta ser uma técnica que pode trazer produtos diferentes do que já foi consumido pelo usuário. No segundo passo, as recomendações geradas na primeira fase servirão de entrada para a próxima técnica, Filtragem Baseada em Conteúdo. Esta próxima técnica irá analisar a similaridade de cada produto retornado com o perfil do usuário e, assim, retornar o conjunto de recomendações de forma ordenada de acordo com a semelhança.

4.1.6. Filtragem Colaborativa

Conforme definido na modelagem da Filtragem Híbrida (Figura 23), a técnica que irá gerar o conjunto inicial de recomendações será a Filtragem Colaborativa. Esse tipo de Filtragem leva em consideração a semelhança entre os perfis dos usuários para processar o conjunto de sugestões. Sendo assim, para a implementação dessa técnica utilizou-se uma API chamada *Apache Mahout*, que irá auxiliar no processo de desenvolvimento.

Nesse tipo de filtragem, o perfil do usuário deve possuir uma estrutura de tabela que contenha todas as avaliações dos usuários. Dessa forma, o primeiro passo realizado foi executar a importação das bibliotecas *mahout-core-0.3*, *slf4j-api-1.6.1*, *slf4j-nop-1.6.1* e *uncommons-math-1.0.2*. A importação dessas bibliotecas se faz necessária para que seja possível utilizar os métodos da API *Apache Mahout*.

A classe que contém o método para realizar o processo da Filtragem Colaborativa foi constituída através do método *recommender,* que receberá como parâmetro a chave primária do usuário, quantidade de recomendações, o objeto de conexão do banco de dados, nome da tabela que contem as avaliações e os nomes das colunas que contém o id do usuário, id do item e avaliação. Para ser possível compreender a forma pela qual a FC foi desenvolvida, é apresentado na Figura 24, a seguir, o método de *recommender*.

```
1  public static ArrayList<ResultSet> recommender(long idUsuario,int qtdRecomendacao,MysqlConnectionPoolDataSource data,
2          String nomeTabela, String colunaIdUsuario,String colunaIdItem,String colunaAvaliacao ){
3          try
4          {
5              DataModel model = new MySQLJDBCDataModel(data,nomeTabela,colunaIdUsuario,colunaIdItem,colunaAvaliacao);
6
7              UserSimilarity usersSimilarity = new PearsonCorrelationSimilarity(model);
8              usersSimilarity.setPreferenceInferrer(new AveragingPreferenceInferrer(model));
9
10             UserNeighborhood vizinhanca = new NearestNUserNeighborhood(4,usersSimilarity,model);
11             Recommender recomendacao = new GenericUserBasedRecommender(model, vizinhanca, usersSimilarity);
12
13             Recommender cachingRecommender = new CachingRecommender(recomendacao);
14
15             List<RecommendedItem> recommendations = cachingRecommender.recommend(idUsuario,qtdRecomendacao);
```

Figura 24 - Implementação da Filtragem Colaborativa

Na implementação da Filtragem Colaborativa é possível observar na linha 5 que ocorre a instanciação do objeto que irá conter o modelo de dados, através da classe MySQLJDBCDataModel. O construtor desta classe pede como parâmetro um objeto do tipo DataSource, uma String com o nome da tabela, e os nomes dos campos da tabela, que foram informados na chamada do método. Na linha 7 ocorre a criação do objeto que irá definir o algoritmo que será utilizado para realizar o cálculo da correlação entre os usuários, sendo que, neste caso, foi escolhido PearsonCorrelationSimilarity, que teve o seu cálculo abordado na seção 2.2.2. Já na linha 8 é informado também o modo como irá ser deduzida uma preferência do usuário em relação a um item. Na linha 10 é criado o objeto que irá conter a forma como

será calculada a vizinhança do usuário alvo. A interface UserNeighborhood é implementada pela classe NearestNUserNeighborhood, recebendo como parâmetro em seu construtor um valor do tipo int que representa a quantidade máxima de usuários similares, um objeto do tipo UserSimilarity (instanciado na linha 7) e um objeto do tipo DataModel (instanciado na linha 5). Na linha 11, é instanciado um objeto que irá conter todos os requisitos para gerar a recomendação, contendo os objetos instanciados nas linhas 5, 7 e 10. Dessa forma, o último passo para que o processo de seleção das recomendações ocorra está na linha 15, onde o método recommend é chamado, recebendo como parâmetro um valor do tipo long, que representa o id do usuário e um valor do tipo int, representando a quantidade máxima de recomendações que se deseja obter (lembrando que esses valores foram passados como parâmetro na chamada do método que realiza o processo da Filtragem Colaborativa). Assim, o método retorna a lista de recomendações gerada, sendo que essa lista será composta apenas pelos ids dos produtos.

Com a implementação da FC finalizada, o próximo passo a ser realizado consistiu na implementação da Filtragem Baseada em Conteúdo, que será abordada na próxima seção.

4.1.7. Filtragem Baseada em Conteúdo

Após ter sido gerado o conjunto inicial das recomendações, o próximo passo a ser realizado é a aplicação da Filtragem Baseada em Conteúdo. Esse tipo de filtragem parte do pressuposto de que se o usuário já consumiu determinado produto uma vez, ele irá comprar produtos semelhantes, sendo que essa semelhança é medida através da descrição textual entre os itens.

A FBC utiliza conceitos da Recuperação da Informação em sua implementação. Sendo assim, foi necessária a escolha de um modelo clássico de RI que seria implementado no desenvolvimento da técnica. Dessa forma, o modelo definido foi o Espaço Vetorial, sendo que as técnicas de otimização e limpeza de texto serão *Stopwords* e *Stemming*. Na Figura 25, a seguir, são apresentados os passos que foram seguidos para o desenvolvimento da técnica de filtragem.

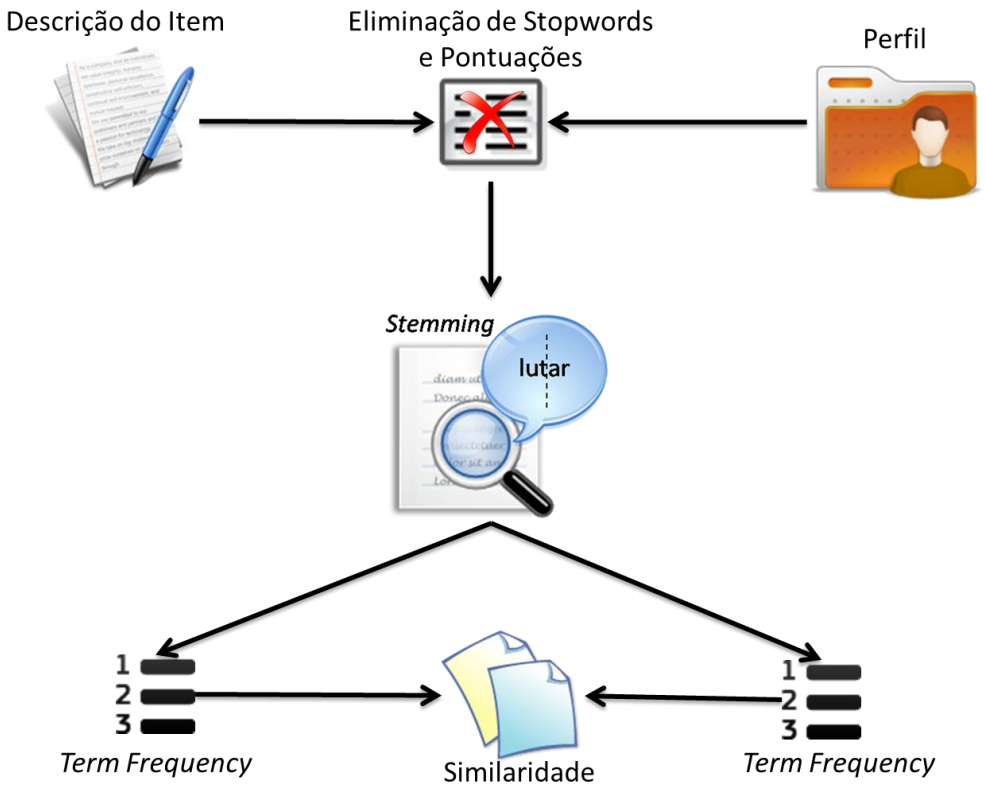


Figura 25 - Filtragem Baseada em Conteúdo

Conforme é apresentado na Figura 25, o primeiro passo para a implementação da FBC foi obter as descrições dos produtos. Tais descrições foram obtidas através dos produtos retornados pelo conjunto de recomendações geradas na seção anterior, acessando a coluna que tem descrição de cada produto. Ainda na primeira etapa, também está incluída a formação do perfil do usuário (resultado da junção da descrição de todos os produtos já avaliados pelo usuário), para que assim, tanto o perfil como a descrição do produto fossem submetidos a aplicação das técnicas de *Stopwords*, eliminação das pontuações e *Stemming*, técnicas aplicadas para otimização do texto e limpeza de termos desnecessários. Terminada esta etapa, obtiveram-se duas listas de termos: uma representando o perfil e outra representando a descrição do produto. Assim, o próximo passo foi calcular a frequência de cada termo contido nas listas, resultando em dois vetores que em cada posição tem um objeto com os atributos: termo e o peso correspondente. Por fim, o último passo foi a implementação do cálculo de similaridade entre o perfil e o produto, feito através da fórmula do co-seno.

Conforme a Figura 25, para o desenvolvimento dessa técnica foram necessárias várias etapas, as quais serão detalhadas nas próximas seções.

4.1.7.1. Otimização e Limpeza do Texto

Antes de analisar a similaridade entre dois textos, é necessário aplicar técnicas que realizam a otimização e redução do texto, eliminando termos desnecessários em relação ao tema abordado, como conjunções, artigos, preposições entre outros. Para esta tarefa foi criada uma classe chamada de OperacoesTexto, contendo todos os métodos que envolvem operações sobre o texto, como *Stopwords*, eliminação de pontuação e *Stemming*.

O primeiro método desenvolvido foi a eliminação de *Stopwords*. O primeiro passo da sua implementação foi definir um arquivo contendo uma lista de palavras como conjunções, preposições, disjunções entre outros. Dessa forma, com a lista de *stopwords* definida, ao realizar a análise do texto se for encontrada qualquer ocorrência de uma palavra que está contida na lista, a mesma deve ser retirada do texto analisado. A Figura 26, a seguir, apresenta parte da lista de *stopwords* definida para este trabalho.

a	aquilo	dele	disse	esta	fim
à	as	deles	disso	está	foi
agora	até	dentro	disto	estamos	for
ainda	atrás	depois	dito	estão	foram
alguém	através	desde	diz	estará	fosse
algum	bem	desligado	diz	estas	fossem
alguma	bom	dessa	dizem	estava	grande
algumas	cada	dessas	dizer	estavam	grandes
alguns	caminho	desse	do	estávamos	há
ali	coisa	desses	dos	este	horas
ambos	coisas	desta	dos	estes	iniciar
ampla	com	destas	e	esteve	inicio

Figura 26 - Lista de *Stopwords*

Como pode ser observado na Figura 26, foi criada uma lista de palavras, através de uma análise empírica sobre os termos da língua portuguesa que não trazem significado para o conteúdo de um texto, que fazem parte da lista de *Stopwords* e que serão eliminadas dos textos analisados. Definida a lista, o próximo passo foi o desenvolvimento do método RemoverStopWords(), apresentado na Figura 27, a seguir.

```
1  public static ArrayList<String> RemoverStopWords(String textoCompleto)
2  {
3      File fileStopWords = new File("M:\\Ulbra\\9º Período\\TCC2\\Stopwords\\ListaStopwords.txt");
4      String textoRemovidoStop = textoCompleto.toUpperCase();
5      ArrayList<String> resultStop= new ArrayList<String>();
6      if(fileStopWords.exists())
7      {
8          BufferedReader br = new BufferedReader(new FileReader(fileStopWords));
9          String linha="";
10         while((linha=br.readLine())!=null)
11         {
12             linha=linha.toUpperCase();
13             linha=linha.trim();
14             textoRemovidoStop=textoRemovidoStop.replaceAll("\\b"+linha+"\\b"," ");
15
16         }
17         br.close();
18         String[] textoFinal = textoRemovidoStop.split(" ");
19         for(int i=0;i<textoFinal.length;i++)
20         {
21             if(!textoFinal[i].isEmpty())
22                 resultStop.add(textoFinal[i]);
23
24
25         }
```

Figura 27 - Remoção de *Stopword*

Conforme pode ser observado na Figura 27, o método recebe como parâmetro um valor do tipo String, sendo que o mesmo representa o texto a ser analisado para a retirada de *stopwords*. Na linha 3, é indicado onde está o arquivo que contém a lista de palavras a serem retiradas. Já na linha 4, o conteúdo da variável que contém o texto é transformado em letras maiúsculas com o método toUpperCase(). Essa transformação é necessária para o momento da comparação das palavras do texto contido na variável com o texto do arquivo. Da linha 6 à 13, ocorre o processo de leitura do arquivo, sendo que para cada linha do arquivo (correspondente a uma palavra) é aplicada a transformação para letra maiúscula e retirado os espaços. Na linha 14, é verificado se a palavra obtida no arquivo está contida no texto e, se a resposta for positiva, a palavra é substituída por um espaço

em branco. Essa localização e substituição da palavra é feita através da utilização do método replaceAll(), que recebe como parâmetro uma expressão regular, que informa a palavra que deve ser localizada e substituída, e o valor de substituição. Após o processo de eliminação dos *stopwords*, o próximo passo é colocar cada termo em uma posição do vetor. Para isso, na linha 18 é aplicado o método split(), informando como parâmetro de separação o espaço em branco e, assim, transformando a String em um vetor de palavras. Por fim, da linha 19 a 22 são verificadas quais posições do vetor não estão vazias e adicionadas no vetor final que será o retorno do método.

Terminada a etapa do desenvolvimento da técnica de *Stopwords*, a segunda técnica desenvolvida foi a eliminação de pontuações. Ela é responsável por remover as pontuações existentes no texto. Na Figura 28, a seguir, será apresentado como o método foi implementado.

```
1  public static String RemoverPontuacao(String textoCompleto)
2  {
3      try
4      {
5          textoCompleto=textoCompleto.replaceAll("[\\?!;.,_\\-|/\\\\(){}\\[\\]<>]"," ");
6          return textoCompleto;
7
8      }
```

Figura 28 - Eliminação de Pontuação

Como é apresentado na Figura 28, o método recebe como parâmetro uma String contendo o texto a ser tratado. Na linha 5, é aplicado o método replaceAll(), que recebe como parâmetro uma expressão regular, que irá verificar se o texto possui alguns dos símbolos informados na expressão e substituir pelo texto informado no segundo parâmetro que, no caso, é um espaço em branco. Dessa forma, o retorno do método é o texto devidamente tratado, sem pontuações.

A terceira e última técnica desenvolvida nessa classe foi o método de *Stemming*. Ele é responsável por reduzir uma palavra ao seu radical para que assim, as variações de palavras com o mesmo radical sejam as mesmas. Para sua implementação foi utilizada uma biblioteca que fornece algoritmos para a aplicação de *Stemming*, chamada *PTStemmer*, que está disponível em: http://code.google.com/p/ptstemmer/. Utilizou-se o algoritmo de *Orengo* para aplicar a técnica de redução ao radical, por se tratar de um algoritmo para a língua

portuguesa. Na Figura 29, a seguir, será apresentado o método implementado para aplicar a técnica.

```
1   public static ArrayList<String> Stemming(ArrayList<String> texto) throws PTStemmerException
2   {
3       Stemmer stemmer = Stemmer.StemmerFactory(StemmerType.ORENGO);
4       ArrayList<String> result = new ArrayList<String>();
5       for(int i=0;i<texto.size();i++)
6       {
7          result.add(PTStemmerUtilities.removeDiacritics(stemmer.getWordStem(texto.get(i))));
8
9       }
10      return result;
11  }
```

Figura 29 - Método de Stemming

Conforme pode ser observado na Figura 29, o método recebe como parâmetro uma lista de palavras do tipo ArrayList<String> (que representam um texto). Cada termo contido na lista será submetido à técnica de *Stemming*, para que o mesmo seja reduzido ao seu radical. Dessa forma, na linha 3 é instanciado um objeto do tipo Stemmer (classe da biblioteca *PTStemmer*) e atribuído a ele o algoritmo que será utilizado para aplicação do Stemming. No caso da linha 3, é atribuído o algoritmo Orengo. Da linha 5 a 9, foi criado um laço de repetição com a finalidade de percorrer a lista de palavras de forma que, para cada posição do vetor (cada palavra), seja aplicado o método getWordStem(), o qual irá reduzir o texto ao seu radical.

Após a implementação das três técnicas (*Stopwords*, Remoção de Pontuação e *Stemming*) o desenvolvimento da classe OperacoesTexto foi finalizada. Dessa forma, todas as operações sobre o texto realizadas no desenvolvimento da ferramenta de recomendação utilizou métodos explicados nessa seção.

Na próxima seção será apresentada a classe que trabalha com a frequência de ocorrência das palavras no texto.

4.1.7.2. Cálculo da Frequência dos Termos

Para realizar esta tarefa, foi criada uma classe chamada TermosFrequencia que é responsável por calcular o peso de cada termo dentro de um texto. A forma pela qual esta classe foi definida pode ser vista na Figura 30, a seguir.

```
1  public class TermosFrequencia
2  {
3      private String Termo;
4      private double Freq;
5      public TermosFrequencia(String termo, double freq )
6      {
7          this.Termo=termo;
8          this.Freq=freq;
9      }
10     public String getTermo()
11     {
12         return Termo;
13     }
14     public void setTermo(String termo)
15     {
16         this.Termo = termo;
17     }
18     public double getFreq()
19     {
20         return Freq;
21     }
22     public void setFreq(double freq)
23     {
24         this.Freq = freq;
25     }
```

Figura 30 - Definição da Classe TermosFrequencia

Conforme pode ser observado na Figura 30, a classe TermosFrequencia é composta pelos atributos Termo (representa a palavra) e Freq (representa a ocorrência do termo no texto) com os seus respectivos métodos de acesso, *get* e *set*. Portanto, um objeto do tipo TermosFrequencia irá conter um termo e a frequência relacionada.

Para realizar o cálculo da frequência dos termos foi criado dentro desta classe o método *TF()*, que é o responsável por receber uma lista de palavras e calcular a ocorrência da mesma dentro desta lista, ou seja, calcular o *Term Frequency* (citado na seção 2.2.1). A forma como o método foi definido pode ser observada na Figura 31, a seguir.

```
1   public static ArrayList<TermosFrequencia> TF(ArrayList<String> texto)
2   {
3       ArrayList<TermosFrequencia> tf = new ArrayList<TermosFrequencia>();
4       int cont;
5       int maior=0;
6       for(int i=0;i<texto.size();i++)
7       {
8           cont = 0;
9           for(int j=0;j<texto.size();j++)
10          {
11              if(texto.get(i).equals(texto.get(j)))
12              {
13                  cont++;
14                  if(cont>1)
15                      texto.remove(j);
16              }
17          }
18          if(cont>maior)
19              maior=cont;
20          tf.add(new TermosFrequencia(texto.get(i),cont));
21      }
22      for(int i=0;i<tf.size();i++)
23      {
24          double peso = tf.get(i).getFreq()/maior;
25          tf.get(i).setFreq(peso);
26      }
```

Figura 31 - Método TF

Como pode ser observado na Figura 31, o método recebe como parâmetro uma lista de termos do tipo ArrayList<String>. O primeiro passo para o cálculo da frequência dos termos (linha 3) foi criar uma lista do tipo ArrayList<TermosFrequencia> (definição da classe demonstrada na Figura 30), sendo que cada posição do vetor será composta por um objeto contendo um atributo termo e outro freq (peso relacionado ao termo). Na linha 4 foi criado um contador para realizar a contagem das ocorrências dos termos dentro da lista e, na linha 5, foi criada uma variável para guardar a ocorrência máxima de um termo dentro do texto. Da linha 6 a 15 foram criadas duas estruturas de repetição (for). No primeiro laço de repetição, será selecionada uma posição do vetor texto, que foi passado como parâmetro para o método. Já no segundo, o mesmo vetor que está sendo utilizado na primeira estrutura será percorrido, a partir da primeira posição, para verificar a quantidade de ocorrência do termo selecionado no primeiro *loop*. Na linha 18 é analisado se a ocorrência do termo verificado no momento é maior que o valor contido na variável maior. Caso seja verdadeiro, a variável recebe o valor da quantidade de ocorrência do termo verificado no momento. Na linha 20, o termo é adicionado na lista com o

tipo ArrayList<TermosFrequencia> (criada na linha 3) juntamente com a sua frequência relacionada (mas com o peso ainda não calculado). Já na linha 22 é criado um *loop* para percorrer o ArrayList<TermosFrequencia>, que contém os termos e suas frequências relacionadas para calcular o peso de cada termo. Dessa forma, na linha 24 é realizado o cálculo do peso, sendo que este cálculo consiste em dividir a frequência do termo (obtida através do método getFreq()) pela frequência máxima de ocorrência de um termo dentro do documento (valor contido dentro da variável maior). Por fim, na linha 25 o peso calculado é atribuído ao seu respectivo termo. O próximo passo a ser seguido no desenvolvimento da Filtragem Baseada em Conteúdo foi a implementação da Classe SimilaridadeConteudo, que será apresentada na próxima seção.

4.1.7.3. Cálculo da Similaridade entre os Conteúdos

Na Filtragem Baseada em Conteúdo as recomendações são geradas analisando-se a similaridade entre o perfil textual do usuário com a descrição dos produtos contidos na base de dados. Dessa forma, para calcular essa similaridade foi utilizada a fórmula do co-seno que retorna um resultado entre 0 e 1. Assim, quanto mais próximo de 1 for o resultado mais semelhante é a descrição do produto ao perfil do usuário.

Antes de realizar o cálculo da similaridade é necessário que os textos (perfil do usuário e descrição do produto) passem por todos os processos descritos nas seções anteriores, pois é necessário ter os dois vetores de termos com suas respectivas frequências, representando cada um dos textos. Após obter os vetores é necessário ordenar o vetor representante da descrição do produto de acordo com o vetor do perfil do usuário, ou seja, o vetor da descrição do produto só irá conter os termos que estejam também no vetor do perfil do usuário. Caso exista um termo no perfil do usuário que não tenha na descrição do produto, esse termo é adicionado ao vetor de descrição do produto com peso 0 (zero). Nas Tabelas 9 e 10, a seguir, será apresentado um exemplo de como é feita essa ordenação.

Tabela 9 - Formação Original dos Vetores

Documento "d"		**Documento "q"**	
Termos	**Peso**	**Termos**	**Peso**
dilúvio	1	Salv	0,5
informaç	0,5	Dilúvio	0,5
melhor	0,5	Informaç	1
pior	0,5	Coloc	0,5
acompanh	0,5	Total	0,5
refluxo	0,5	Refluxo	0,5

Tabela 10 - Vetores Ordenados

Descrição Produto		**Perfil do Usuário**	
Termos	**Peso**	**Termos**	**Peso**
dilúvio	1	dilúvio	0,5
informaç	0,5	informaç	1
refluxo	0,5	Refluxo	0,5
salv	0	Salv	0,5
coloc	0	Coloc	0,5
total	0	Total	0,5

Para implementação da ordenação descrita acima foi necessário desenvolver um método chamado OrdenarArrayDocumento(), que recebe como parâmetro duas listas do tipo ArrayList<TermosFrequencia>, uma lista representando o perfil do usuário (perfil), e a outra representando a descrição do produto (documento). Na Figura 32, a seguir, pode-se observar como o método foi implementado.

```
1   public static ArrayList<TermosFrequencia> OrdenarArrayDocumento(ArrayList<TermosFrequencia> perfil,
2       ArrayList<TermosFrequencia> documento)
3   {
4       ArrayList<TermosFrequencia> documentoOrdenado = Util.IniciarArray(perfil.size());
5       boolean adicionou=false;
6       for(int i=0;i<perfil.size();i++)
7       {
8           for(int j=0;j<documento.size();j++)
9           {
10              if(documento.get(j).getTermo().equals(perfil.get(i).getTermo()))
11              {
12                  documentoOrdenado.set(i,documento.get(j));
13                  adicionou=true;
14              }
15
16          }
17          if(documentoOrdenado.size()>0 && !adicionou)
18          {
19              TermosFrequencia termo = new TermosFrequencia(perfil.get(i).getTermo(),0);
20              documentoOrdenado.set(i,termo);
21
22          }
23          adicionou=false;
24      }
25      return documentoOrdenado;
26  }
```

Figura 32 - Ordenação das listas

Conforme pode ser observado na Figura 32, na linha 4 é instanciado uma lista do tipo ArrayList<TermoFrequencia> que irá conter a representação da descrição ordenada de acordo com a lista de termo do perfil. Na linha 6 é criado um *loop* para percorrer a lista perfil. Da linha 8 a 16 é criada uma segunda estrutura de repetição para percorrer a lista documento, que deve ser ordenada de acordo com a lista perfil. Dessa forma, dentro desta estrutura é verificado se existe algum termo da lista documento que é igual ao termo selecionado no primeiro laço de repetição. Caso exista algum termo igual, este termo e seu peso relacionado é adicionado na lista documentoOrdenado (criada na linha 4) na posição i selecionada. Das linhas 17 a 22 é verificado se o termo selecionado da lista perfil não foi adicionado à lista documentoOrdenado (se o termo não foi adicionado, então o mesmo não existe dentro da lista documento). Caso a verificação seja verdadeira, o termo é adicionado na mesma posição em que o termo do perfil está na lista documentoOrdenado com peso 0 (zero), pois esse termo não existe na lista documento. Por fim, na linha 25 é retornada a lista documentoOrdenado, representando a descrição do produto ordenada.

Após ordenar a lista representante da descrição do produto com a representante do perfil do usuário, já é possível realizar o cálculo da similaridade entre os dois textos, utilizando a fórmula do co-seno (seção 2.2.1). Na Figura 33, a seguir, será apresentada a forma pela qual o método da similaridade foi desenvolvido.

```
1   public static double SimilaridadeCosseno(ArrayList<TermosFrequencia> perfil,
2       ArrayList<TermosFrequencia> documentoOrdenado)
3   {
4       double numerador = 0;
5       double denominadorPerfil=0;
6       double denominadorDocumento=0;
7       double similaridade=0;
8       for(int i=0;i<perfil.size();i++)
9       {
10          numerador+=perfil.get(i).getFreq()*documentoOrdenado.get(i).getFreq();
11          denominadorPerfil+=Math.pow(perfil.get(i).getFreq(), 2);
12          denominadorDocumento+=Math.pow(documentoOrdenado.get(i).getFreq(),2);
13      }
14      denominadorPerfil=Math.sqrt(denominadorPerfil);
15      denominadorDocumento=Math.sqrt(denominadorDocumento);
16      similaridade=numerador/(denominadorPerfil*denominadorDocumento);
17      return similaridade;
18  }
```

Figura 33 - Similaridade entre os Conteúdos

O método SimilaridadeCosseno irá receber como parâmetro duas listas do tipo ArrayList<TermosFrequencia>, uma representando o perfil e a outra representando a descrição do produto ordenada. Nas linhas 4 a 7 são criadas as variáveis necessárias para fazer o cálculo do co-seno. Como a lista do perfil do usuário e a do produto serão do mesmo tamanho (pois a lista do produto está ordenada com o perfil), na linha 8 é feito um *loop* percorrendo a lista perfil e, na linha 10 inicia-se o processo do cálculo do numerador da fórmula do co-seno. A cada ciclo do for é feita a soma da multiplicação do peso de um termo da lista perfil com o peso de um termo (na mesma posição) da lista documentoOrdenado. Das linhas 12 a 15 são realizados os cálculos para o denominadorPerfil e denominadorDocumento, sendo que para cada um, primeiramente, é feita a soma dos pesos elevados ao quadrado e depois é retirada a raiz quadrada do resultado da soma. Na linha 16, é onde ocorre o cálculo da similaridade, onde se divide o numerador pelo denominador (que é o denominadorPerfil multiplicado pelo denominadorDocumento) e o resultado é o que indica o quão próximo o produto é do perfil do usuário.

Para guardar a pontuação de similaridade que cada item teve em relação ao perfil do usuário foi criada a classe RecomendacaoBaseadaConteudo, que possui dois atributos, IdProduto e Pontuacao, com os seus respectivos métodos de acesso (*get* e *set*). Essa classe se fez necessária para armazenar as pontuações de cada item, para que fosse criada uma lista contendo os produtos com suas respectivas

pontuações de similaridade, para que depois fosse feito uma ordenação por relevância das recomendações retornadas ao usuário.

Na próxima seção será apresentada a classe geral que irá chamar todos os métodos para gerar as recomendações.

4.1.8. Chamadas dos métodos

Esta seção irá apresentar como foram feitas as chamadas dos métodos necessários para gerar as recomendações, além de demonstrar a lógica de funcionamento da ferramenta.

O método geral responsável por gerar as recomendações é o Recomendar(), mas antes de chamar esse método é necessário chamar o método ConectarBanco(), para que a ferramenta estabeleça conexão com o banco de dados do cliente. A forma pelo qual o método foi definido pode ser visto na Figura 34, a seguir.

```
1   public static MysqlConnectionPoolDataSource ConectarBanco(String user,String password,
2       String serverName,int portNumber,String databaseName)
3   {
4
5       MysqlConnectionPoolDataSource data = new MysqlConnectionPoolDataSource();
6       data.setUser(user);
7       data.setPassword(password);
8       data.setServerName(serverName);
9       data.setPortNumber(portNumber);
10      data.setDatabaseName(databaseName);
11      data.setAutoReconnect(true);
12      data.setConnectTimeout(28800);
13      return data;
14  }
```

Figura 34 - Conexão com o Banco de Dados

O método apresentado na Figura 34 recebe como parâmetro o usuário de conexão com o banco de dados (String), senha (String), nome do servidor de banco de dados (String), porta de conexão (int) e o nome do banco de dados (String), que se deseja conectar para realizar o processo de geração das recomendações.

Após a chamada do método de conexão com o banco de dados o próximo método a ser chamado será o Recomendar(), que irá gerar as recomendações e retornar uma lista de ids dos produtos recomendados por ordem de relevância. A forma pelo qual este método foi definido pode ser visto na Figura 35.

```
public static List<Long> Recomendar(long idUsuario,int qtdRecomendacoes,MysqlConnectionPoolDataSource data,
    String nomeTabelaAvaliacao, String colunaIdUsuario,String colunaIdItem,String colunaAvaliacao,
    String NomeTabelaProduto,String colunaDescricaoProduto, String colunaIdProduto, String NomeTabelaArmazenarRecomendacao,
    String NomeColunaArmIdUser, String NomeColunaArmIdItem, String NomeColunaArmData ) throws SQLException
    {
        DBManager dbm = new DBManager();
        ResultSet rs = (ResultSet) dbm.getRecomendacaoUser(idUsuario, data);
        ArrayList<Long> recomendacoesFinais=new ArrayList<Long>();
        if(rs.next())
        {
            long resultado = new Date().getTime()-rs.getDate("DataCadastro").getTime();
            resultado = resultado/(24*60*60*1000);
               if(resultado<7)
               {
                   while(rs.next())
                   {
                       recomendacoesFinais.add(rs.getLong("IdProduto"));
                   }
                   return recomendacoesFinais;
               }
               else
               {
                   dbm.DeleteRecomendacoesUser(idUsuario, data,NomeTabelaArmazenarRecomendacao,NomeColunaArmIdUser);
                   return GerarRecomendacao(idUsuario,qtdRecomendacoes,data,nomeTabelaAvaliacao, colunaIdUsuario,
                       colunaIdItem,colunaAvaliacao, NomeTabelaProduto,colunaDescricaoProduto, colunaIdProduto,
                       NomeTabelaArmazenarRecomendacao, NomeColunaArmIdUser, NomeColunaArmIdItem, NomeColunaArmData);
               }
        }else{
               return GerarRecomendacao(idUsuario,qtdRecomendacoes,data,nomeTabelaAvaliacao, colunaIdUsuario,colunaIdItem,
               colunaAvaliacao, NomeTabelaProduto,colunaDescricaoProduto, colunaIdProduto, NomeTabelaArmazenarRecomendacao,
               NomeColunaArmIdUser, NomeColunaArmIdItem, NomeColunaArmData);
        }
```

Figura 35 - Recomendação de Produtos

Ao ser chamado, o método Recomendar() irá verificar se já existe algum conjunto de recomendações já geradas para o usuário em questão. Caso exista, ele irá verificar se a data em que a recomendação foi gerada ultrapassa o prazo máximo de sete dias (prazo estipulado pelo Sistema de Recomendação desenvolvido neste trabalho para guardar um conjunto de recomendações). Se o prazo ainda não tiver ultrapassado, ele apenas busca no banco de dados o conjunto de recomendações geradas para aquele usuário. Caso contrário, ele irá apagar o conjunto de recomendações antigo e chamar o método GerarRecomendacao() para gerar um novo conjunto de recomendações e armazená-lo no banco.

Para gerar as recomendações o método Recomendar() faz uma chamada para o método GerarRecomendacao(), que é o método que contém todas as chamadas para os demais métodos responsáveis pela geração da recomendação. O primeiro método a ser chamado é o responsável pela Filtragem Colaborativa, que irá retornar uma lista de produtos, que servirá de entrada para Filtragem Baseada em Conteúdo. A FBC, por sua vez, irá guardar em uma mesma variável a junção das descrições de todos os produtos avaliados pelo usuário a qual a recomendação é direcionada e que irá representar o perfil deste usuário. Após chamar o método da Filtragem Colaborativa a próxima etapa será criar uma estrutura de repetição para percorrer a

lista de recomendações retornada pela primeira técnica e, assim, verificar a similaridade de cada produto da lista com o perfil do usuário. Dessa forma, para realizar esse processo será necessário chamar os métodos, descritos nas seções anteriores, na seguinte ordem:

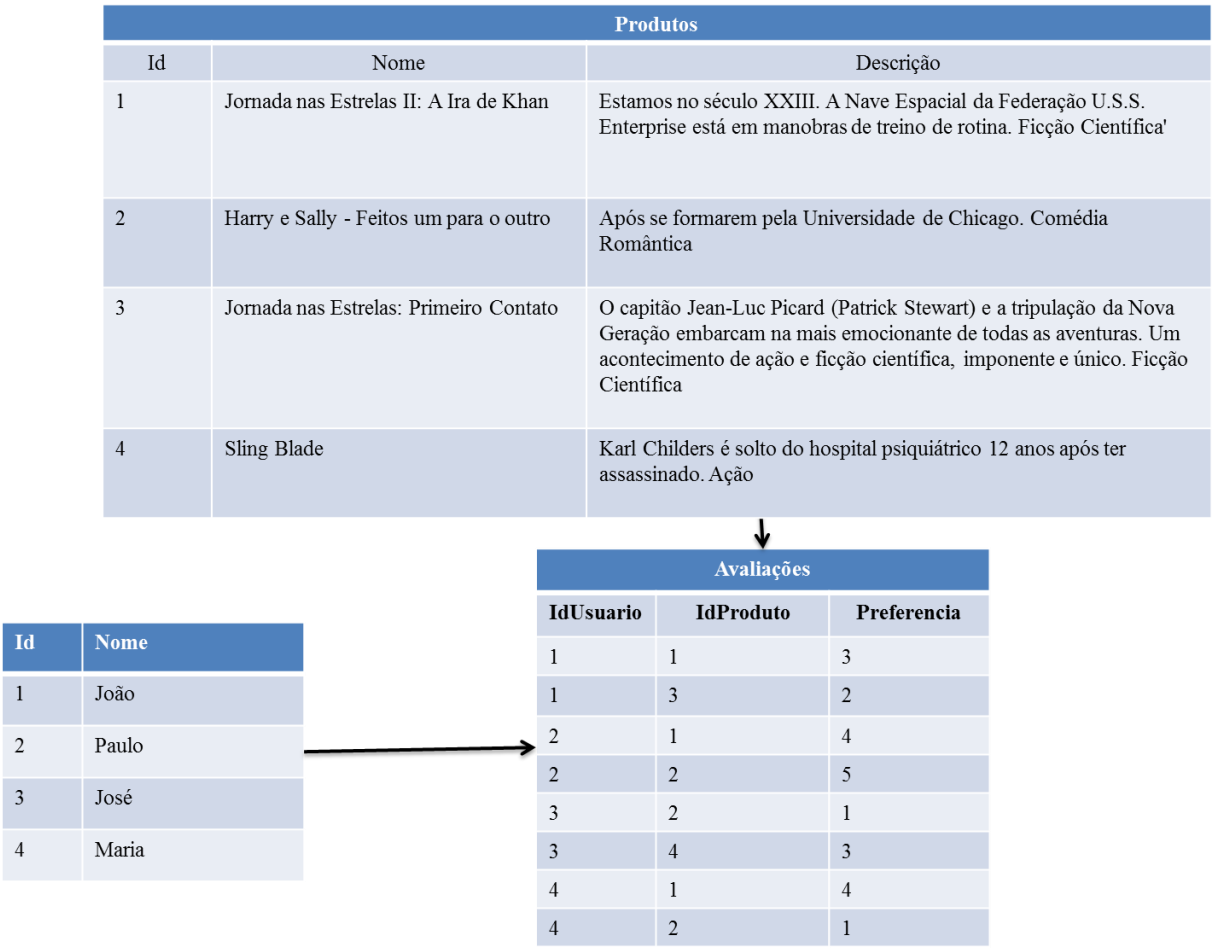
1. RemoverPontuacao – este método foi chamado para aplicar a remoção de pontuações tanto para o perfil do usuário quanto para o produto (retornado pela FC) que seria analisado a similaridade.
2. RemoverStopWords – após remover as pontuações o próximo passo foi a remoção dos *Stopwords*, que são palavras que não trazem significado para o texto.
3. Stemming – como última etapa da otimização do texto, foi aplicada a técnica de *Stemming*, a fim de reduzir cada palavra ao seu radical.
4. TF – após a otimização e limpeza do texto, foi calculada a frequência e peso de cada termo dentro do texto.
5. OrdenarArrayDocumento – a penúltima etapa foi ordenar a lista representante do produto a ser verificada a similaridade de acordo com a lista representante do perfil do usuário.
6. SimilaridadeCosseno – por fim, calculou-se a similaridade entre o produto e o perfil do usuário, e se adicionou a chave primária do produto e o resultado da similaridade a uma lista do tipo `RecomendacaoBaseadaConteudo`, que contém atributos `IdProduto` e `Pontuacao`, para que, ao final, fizesse uma ordenação por relevância e retornasse ao usuário as recomendações ordenadas.

Com a lista de recomendação gerada, a próxima etapa foi armazenar essas recomendações associando-as ao usuário alvo. Dessa forma, a próxima vez em que for solicitado um conjunto de recomendações para um usuário que já tem recomendações armazenadas no banco de dados, o sistema apenas irá buscar na base de dados e retornar as recomendações associadas a aquele usuário, evitando que ocorra um novo processamento para geração das recomendações. Vale ressaltar que essas recomendações serão válidas por até sete dias. Passado esse prazo, será feita uma nova geração de recomendações, quando solicitado, para o usuário.

Na próxima seção serão apresentados os testes realizados sobre a ferramenta.

4.2. Testes

Esta seção será responsável por apresentar os testes realizados sobre a ferramenta proposta neste trabalho. Desta forma, buscou verificar se o resultado retornado pela ferramenta de recomendação está correto. No entanto, para que isto fosse possível, criou-se uma base de dados pequena com 10 produtos (filmes), 10 usuários e 51 avaliações. Na Figura 36, a seguir, são demonstradas partes dessas tabelas.



Produtos

Id	Nome	Descrição
1	Jornada nas Estrelas II: A Ira de Khan	Estamos no século XXIII. A Nave Espacial da Federação U.S.S. Enterprise está em manobras de treino de rotina. Ficção Científica'
2	Harry e Sally - Feitos um para o outro	Após se formarem pela Universidade de Chicago. Comédia Romântica
3	Jornada nas Estrelas: Primeiro Contato	O capitão Jean-Luc Picard (Patrick Stewart) e a tripulação da Nova Geração embarcam na mais emocionante de todas as aventuras. Um acontecimento de ação e ficção científica, imponente e único. Ficção Científica
4	Sling Blade	Karl Childers é solto do hospital psiquiátrico 12 anos após ter assassinado. Ação

Avaliações

IdUsuario	IdProduto	Preferencia
1	1	3
1	3	2
2	1	4
2	2	5
3	2	1
3	4	3
4	1	4
4	2	1

Id	Nome
1	João
2	Paulo
3	José
4	Maria

Figura 36 - Base de Dados de Teste

Optou-se por uma base de dados pequena pelo fato de que realizar os testes na ferramenta de forma manual, envolvendo os cálculos matemáticos da Filtragem Colaborativa e Filtragem Baseada em Conteúdo tendo uma base dados grande ficaria inviável por causa da quantidade de dados e cálculos.

Utilizando a ferramenta de recomendação desenvolvida neste trabalho, foi solicitado um conjunto de recomendações para o usuário com chave primária igual a

“3” e com quantidade de recomendação desejada igual a “3”. Dessa forma, a Figura 37, a seguir, apresenta o resultado gerado pela Ferramenta.

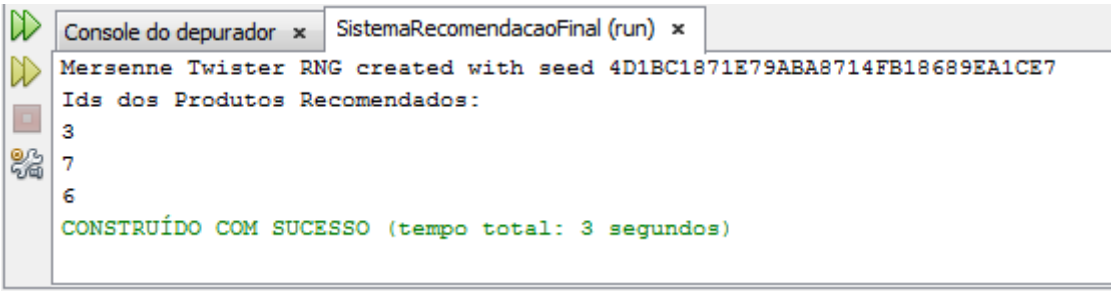


Figura 37 - Resultados Retornados

Conforme apresentado na Figura 37, as chaves primárias dos produtos retornados foram: 3 (Jornada nas Estrelas: Primeiro Contato), 7 (Jornada nas Estrelas IV: A Volta Para Casa) e 6 (Dois Homens e um Destino).

Tendo como premissa o resultado retornado pela ferramenta de recomendação, os testes foram realizados tendo como parâmetros de entrada o mesmo usuário e a mesma quantidade de recomendações informadas quando se utilizou a ferramenta de recomendação. Dessa forma, os testes foram divididos em 3 etapas:

- primeira etapa - constituiu em realizar os cálculos de similaridade para cada usuário da base, para formar o conjunto de usuários com perfis similares ao do usuário alvo (usuário com chave primária 3). Dessa forma, para cada usuário foram levados em consideração apenas os produtos avaliados em comum com o usuário alvo para realizar o cálculo da similaridade (Seção 2.2.2). Assim, a fórmula aplicada para calcular a similaridade entre os usuários foi a Correlação de Pearson, que foi aplicada para todos os usuários. Na fórmula, a seguir, é apresentado um exemplo do cálculo da similaridade entre o usuário alvo (chave primária 3) e o usuário com chave primária igual “4”.

$$w_{a,u} = \frac{((2-1{,}7)*(4-2{,}3)) + ((1-1{,}7)*(1-2{,}3)) + ((2-1{,}7)*(2-2{,}3))}{\sqrt{(2-1{,}7)^2 + (1-1{,}7)^2 + (2-1{,}7)^2 \; * \; (4-2{,}3)^2 + (1-2{,}3)^2 + (2-2{,}3)^2}} = 0{,}8$$

  Como pode ser observado na fórmula acima, a similaridade entre o usuário “3” e o usuário “4” é de 0.8, que está dentro de uma escala entre -1 e 1. Assim, o usuário “4” pode ser considerado com o perfil bem similar ao do usuário alvo. Dessa forma, a fórmula da Correlação de Pearson foi aplicada para toda a base de dados, resultando na Tabela 11, a seguir.

Tabela 11 – Nível de Similaridade com o Usuário 3

Chave Primária dos Usuários	Correlação de Pearson (similaridade)
1	1,0
9	1,0
4	0,8
7	0,3
8	- 0,3
10	- 0,3
5	- 0,5
2	- 1,0
6	- 1,0

Por meio da Tabela 11 é possível identificar o nível de similaridade de todos os usuários da base em relação ao usuário 3, possibilitando definir um ponto de corte, sendo que somente os três usuários mais próximos serão utilizados para gerar as recomendações. Portanto, serão utilizados apenas os usuários 1, 9 e 4.

- segunda etapa – consiste em buscar os produtos que já foram avaliados pelos três usuários, mas não pelo usuário alvo. Dessa forma, foram encontrados três produtos que são os itens 3 (Jornada nas Estrelas: Primeiro Contato), 6 (Dois Homens e um Destino) e 7 (Jornada nas Estrelas IV: A Volta Para Casa). Assim, o próximo passo é calcular a predição da nota que o usuário alvo daria para estes produtos. A forma pela qual o cálculo da predição foi definido pode ser visualizado na Tabela 12, a seguir.

Tabela 12 - Cálculo de Predição de Notas

Produtos	Predição
Jornada nas Estrelas: Primeiro Contato	$P_{3,3} = 2 + \frac{((2-3{,}7)*1) + ((3-3{,}7)*1) + ((4-2{,}3)*0{,}8)}{1+1+0{,}8} = 1{,}6$
Dois Homens e um Destino	$P_{3,6} = 2{,}3 + \frac{(5-3{,}7)*1}{1} = 3{,}6$
Jornada nas Estrelas IV: A Volta Para Casa	$P_{3,3} = 2 + \frac{((5-3{,}7)*1) + ((4-2{,}3)*0{,}8)}{1+0{,}8} = 3{,}5$

- terceira etapa – com o conjunto de recomendação já formado, esta etapa consiste em calcular a similaridade entre o conteúdo dos produtos retornados com o perfil do usuário. Dessa forma, utilizou-se o perfil do usuário constituído de todas as descrições dos produtos avaliados por ele, que pode ser visualizado na Figura 38, a seguir.

Estamos no século XXIII. A Nave Espacial da Federação U.S.S. Enterprise está em manobras de treino de rotina. Ficção Científica; Após se formarem pela Universidade de Chicago. Comédia Romântica; Karl Childers é solto do hospital psiquiátrico 12 anos após ter assassinado. Ação; Um bando de terroristas apodera-se de um importante aeroporto internacional. Ação

Figura 38 - Perfil Textual do Usuário

Assim, após obter o perfil descritivo do usuário, o próximo passo é aplicar as técnicas de otimização de texto como *Stopwords* e *Stemming* e calcular a frequência dos termos. A Tabela 13, a seguir, apresenta parte da representação do perfil do usuário com os termos e seus respectivos pesos.

Tabela 13 - Representação do Perfil do Usuário

Termos	Pesos
assassin	0.5
aca	1.0
band	0.5
terror	0.5
apod	0.5
import	0.5
aeroport	0.5
nav	0.5
espacial	0.5
feder	0.5

Após obter a representação, em forma de termos, do perfil e dos produtos, inicia-se a ordenação da lista de termos que representa o produto com a lista de termos do perfil do usuário. Na Tabela 14, a seguir, são apresentadas parte das representações do perfil do usuário 3 e do produto a qual irá ser utilizado para analisar a similaridade, o filme Jornada nas Estrelas: Primeiro Contato (chave primária 3).

Tabela 14 - Representação dos Termos Ordenados

Perfil do Usuário 3		Produto 3	
Termos	**Peso**	**Termos**	**Peso**
manobr	0,5	manobr	0
trein	0,5	trein	0
rotin	0,5	rotin	0
ficca	0,5	ficca	1,0
cientif	0,5	cientif	1,0
form	0,5	form	0

Conforme apresentado na Tabela 14, pode-se observar que os termos que não existem na descrição do produto e existem no perfil do usuário, são adicionados à lista do produto com o peso 0. Após a ordenação das listas iniciou-se o cálculo da similaridade entre o perfil e o produto, conforme a fórmula a seguir.

$$sim_{3,3} = \frac{\ldots + (0{,}5 * 0) + (0{,}5 * 0) + (0{,}5 * 0) + (0{,}5 * 1{,}0) + (0{,}5 * 1{,}0) + (0{,}5 * 0) + \cdots}{\sqrt{\ldots + 0{,}5^2 + 0{,}5^2 + 0{,}5^2 + 0{,}5^2 + 0{,}5^2 + 0{,}5^2} \times \sqrt{0^2 + 0^2 + 0^2 + 1^2 + 1^2 + 0^2 + \cdots}} = 0{,}34$$

Dessa forma, a similaridade entre o perfil do usuário com chave primária 3 e o produto de chave primária 3 (Jornada nas Estrelas: Primeiro Contato) é de 0,34, dentro de uma escala de 0 a 1. Na Tabela 15, a seguir, é apresentada a similaridade entre os três produtos retornados pela Filtragem Colaborativa e o perfil do usuário especificado no inicio dos testes.

Tabela 15 - Similaridade

Produto	Similaridade
6	0,17
7	0,34
3	0,34

Como pode ser observado na Tabela 15, ao ordenar os resultados por relevância, o retorno das recomendações para o usuário ficaria da seguinte forma: 3 (Jornada nas Estrelas: Primeiro Contato), 7 (Jornada nas Estrelas IV: A Volta Para Casa) e 6 (Dois Homens e um Destino). Comparando ao resultado retornado pela ferramenta de recomendação com o resultado dos testes, pode-se observar que os dois resultados são iguais, comprovando o funcionamento da ferramenta.

Na próxima seção serão apresentadas as considerações finais sobre o desenvolvimento deste trabalho.

5 CONSIDERAÇÕES FINAIS

Nesse trabalho foram abordados os conceitos sobre Sistemas de Recomendação, Coleta de Informação do Usuário, Técnicas de Recomendação e Recuperação da Informação. As compreensões destes conceitos foram de grande importância para que fosse possível a definição e o desenvolvimento da ferramenta de Recomendação proposta neste trabalho. Com base nestes estudos foi definido que a Recuperação da Informação seria baseada na recuperação por filtragem e o modelo clássico do processo da RI seria o Modelo Espaço Vetorial. Em relação às técnicas de recomendação foi utilizada a Filtragem Híbrida no modelo cascata, sendo que as técnicas escolhidas para o desenvolvimento da FH foram a Filtragem Colaborativa e a Filtragem Baseada em Conteúdo. A Filtragem Híbrida no modelo cascata utiliza uma técnica de recomendação para gerar o primeiro conjunto de recomendações que, por sua vez, será utilizada como entrada para segunda técnica de recomendação que irá refinar (ordenar) o primeiro conjunto de recomendações e assim as retornar para a aplicação cliente.

Para realizar o desenvolvimento da ferramenta foi necessária a implementação de duas técnicas de recomendação que compõe a Filtragem Híbrida, a Filtragem Colaborativa e a Filtragem Baseada em Conteúdo, respectivamente. A primeira técnica foi implementada com o auxílio da API *Apache Mahout*, o que tornou o desenvolvimento dessa técnica mais rápida, apesar do tempo destinado ao estudo da utilização da API. No decorrer dos estudos sobre a API, encontrou-se dificuldade em relação a documentação, pois a maioria das documentações existentes estão desatualizadas, não estando de acordo com a versão da biblioteca. Já a segunda técnica envolveu o desenvolvimento de diversas tarefas como: remoção de *Stopwords*, de *Stemming*, cálculo da frequência, ordenação das listas e o cálculo da similaridade entre dois textos, finalizando, assim, o desenvolvimento da FBC.

Após realizar a implementação da ferramenta de recomendação, pode-se perceber que ao trabalhar com uma base de dados muito grande o processamento

para gerar as recomendações torna-se lento, pela grande quantidade de processamento que é feito para calcular as recomendações. Dessa forma, a saída utilizada para amenizar esse problema foi armazenar o primeiro conjunto de recomendações gerado para o usuário, sendo que na próxima requisição pelas recomendações do mesmo usuário, não serão geradas novas recomendações e sim serão buscadas no banco de dados. Vale ressaltar que foi estipulado um tempo de validade para essas recomendações armazenadas, sendo que, ao ser finalizado, as recomendações serão geradas novamente.

Para realizar os testes, optou-se por criar e utilizar uma pequena base de dados, devido aos cálculos que teriam que ser feitos manualmente, que tornam-se inviáveis caso seja trabalhado com uma grande base de dados. Dessa forma, ao realizar os testes pode-se comprovar a funcionalidade da ferramenta. Assim, também verificou-se que quanto maior a base de dados de um sistema de recomendação for, maior será a quantidade de cálculos matemáticos que serão realizados, aumentando assim o processamento e o tempo para retornar o resultado.

Por fim, vale ressaltar que o resultado do presente trabalho poderá ser adicionado a uma aplicação já existente, desde que sejam atendidos alguns requisitos da ferramenta, para gerar recomendações para seus usuários, contribuindo para aumentar as vendas do Comércio Eletrônico.

5.1. Trabalhos Futuros

Com relação a trabalhos futuros, propõe-se utilizar a computação distribuída para melhorar o desempenho da ferramenta de recomendação. Desta forma, o processamento será dividido entre várias máquinas, fazendo com que o processamento para gerar as recomendações em uma base de dados grande seja rápido. Assim, independente do tamanho da base de dados, o tempo para gerar o conjunto de recomendações sempre será otimizado.

Outra proposta de melhoria seria implementar uma forma de deixar a ferramenta genérica, ou seja, independente da plataforma de banco de dados. Em seu estado atual, a ferramenta desenvolvida trabalha apenas com o Banco de dados MySql, devido à limitação imposta pela API utilizada para desenvolver a Filtragem Colaborativa. Por isto, a solução seria encontrar alguma outra API ou desenvolver os métodos para aplicar a FC, já que todos os conceitos necessários para realizar esta tarefa já foram estudados.

Por fim, outra proposta seria realizar a implantação da ferramenta em um ambiente real e fazer uma análise dos resultados gerados pela ferramenta de recomendação, aplicando testes de qualidade no conjunto de recomendações retornadas.

6 REFERÊNCIAS BIBLIOGRÁFICAS

ADOMAVICIUS, Gediminas; TUZHILIN, Alexander. Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions. **IEEE Transactions on Knowledge and Data Engineering**, Piscataway, v.17, n.6, p. 734-749, 2005. Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=3631DFDE0403B3DBFB7E2BDDEC749731?doi=10.1.1.107.2790&rep=rep1&type=pdf>. Acesso em: 30 de maio de 2010.

BAEZA-YATES, Ricardo, Ribeiro-Neto, Berthier, Modern Information Retrieval, **ACM Press**, New York, USA, 1999.

BORGES, Deise Miranda. **Estudo de Técnicas de Recomendações Automáticas de Produtos.** 2010. 85p. Relatório de Estágio (Graduação em Sistemas de Informação) – Centro Universitário Luterano de Palmas, Palmas.

BURKE, Robin. Hybrid Recommender Systems: Survey and Experiments**. User Modeling and User-Adapted Interaction.** Massachusetts, novembro de 2002. p. 331-370. Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.88.8200&rep=rep1&type=pdf>. Acesso em: 30 de maio de 2010.

BURKE, Robin. Hybrid web recommender systems. **In:___**.**The adaptative web**. Berlim: Springer-Verlag, 2007. cap. 12, p.377-408.

CASTRO, Saulo de Souza Guerra Ferreira de; BARBOSA, Tiago Martins. **RAPIDos:** Recomendação Automática de Produtos Interessantes em Dispositivos Móveis. 2009. 61p. TCC (Graduação em Ciência da Computação) – Universidade de Brasília, Brasília. Disponível em:

<http://monografias.cic.unb.br/dspace/bitstream/123456789/191/1/Monografia_Saulo_Tiago_FINAL.pdf>. Acesso em: 22 de fevereiro de 2010.

CAZELLA, Sílvio César. **Aplicando a Relevância da Opinião de Usuários em Sistema de Recomendação para Pesquisadores**. 2006. 180p. Tese (Doutorado em Ciências da Computação) – Universidade Federal do Rio Grande do Sul, Porto Alegre. Disponível em: <http://hdl.handle.net/10183/8424>. Acesso em: 20 de agosto de 2010.

GODOY NETO, Mario. **RAVE au DISCO:** Recomendação automática de vídeo como auxílio no processo de disseminação do conhecimento. 2009. 148p. Dissertação (Mestrado em Ciência da Computação) – Universidade Federal de Pernambuco, Recife. Disponível em: <http://www.bdtd.ufpe.br/tedeSimplificado//tde_busca/arquivo.php?codArquivo=6959>. Acesso em: 19 de março de 2010.

INGERSOLL, Grant. **Introdução ao Apache Mahout**. 2009. Disponível em: <http://www.ibm.com/developerworks/br/java/library/j-mahout/>. Acesso em: 15 de fevereiro de 2011.

SCHAFER, J. Ben; KONSTAN, Joseph; RIEDL, John. Recommender Systems in E-Commerce. In: ACM CONFERENCE ON ELECTRONIC COMMERCE, 1999, Denver, Colorado. **Anais**... p. 158-166. Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.39.2552&rep=rep1&type=pdf>. Acesso em: 05 de abril de 2010

SCHAFER, J. Ben; KONSTAN, Joseph A.; RIEDL, John. E-Commerce Recommendation Applications. **Data Mining e Knowledge Discovery**, Massachusetts, n. 5, jan. 2001. Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.43.1280&rep=rep1&type=pdf>. Acesso em: 19 de abril de 2010.

SILVA, Edeilson Milhomem da. **SWEETS:** um Sistema de Recomendação de

Especialistas aplicado a Redes Sociais. 2009. 145p. Dissertação (Mestrado em Ciência da Computação) – Universidade Federal de Pernambuco, Recife.

WIVES, Leandro Krug**. Tecnologias de Descoberta de Conhecimento em Texto Aplicadas à Inteligência Competitiva**. 2002.116p. Tese (Doutorado em Computação) – Universidade Federal do Rio Grande do Sul, Porto Alegre. Disponível em: <http://www.scribd.com/doc/35627173/TECNOLOGIAS-DE-DESCOBERTA-DE-CONHECIMENTO-EM-TEXTOS-APLICADAS-A-INTELIGENCIA-COMPETITIVA>. Acesso em: 14 de outubro de 2010.