Sistemas de Recomendação em Rapid Miner:

um caso de estudo

por

Sérgio Francisco dos Santos Morais

Dissertação de Mestrado em

Análise de Dados e Sistemas de Apoio à Decisão

Orientada por:

Prof. Dr. Carlos Soares

Faculdade de Economia

Universidade do Porto

2012

Aos meus Pais …

Nota biográfica

O candidato nasceu na freguesia de Mafamude, concelho de Vila Nova de Gaia, a 5 de Julho de 1979.

Em 2004, licenciou-se em Engenharia Informática (ramo de Computadores e Sistemas) pelo Instituto Superior de Engenharia do Porto (ISEP) com uma média final de 16 valores.

A nível profissional, teve desde o final da sua licenciatura, uma atividade ininterrupta na área de Informática, mais precisamente na análise e desenvolvimento de *software* para *Web*, numa empresa de *software* de gestão da região do Porto.

Em 2009, tomou a iniciativa de concretizar um desejo antigo: o de ingressar no Mestrado de Análise de Dados e Sistemas de Apoio à Decisão (MADSAD). Esse desejo implicou um novo e intenso desafio na minha vida: o de conciliar a vida profissional com a académica. Em 2012, prevê concluir o seu mestrado.

Agradecimentos

Aproveito este espaço para agradecer a todas as pessoas que direta ou indiretamente contribuíram e me ajudaram na elaboração desta dissertação.

A todas pelo seu apoio e incentivo, com um agradecimento especial:

- Ao meu orientador, Prof. Dr. Carlos Soares, pelo interesse, ajuda e apoio que me prestou. Pelo fornecimento dos dados que foram usados na componente experimental. Também pela disponibilidade que teve para esclarecimento de dúvidas que foram surgindo e na revisão das diversas versões do relatório;

- Aos meus pais pela força que me deram e pela paciência que tiveram para suportar dias e noites sem a minha atenção.

Este trabalho foi financiado em parte pelo projeto Palco 3.0. Foi um projeto em Co-Promoção de I&DT, co-financiado pelo Quadro de Referência Estratégico Nacional (QREN-AdI Palco 3.0/3121 PONORTE)

A todos o meu muito obrigado!

Resumo

No comércio eletrónico, a quantidade de produtos e serviços disponíveis é grande.

Considerando cada sítio individualmente, é frequente existirem produtos que nunca são apresentados aos clientes, apesar de serem interessantes para eles. Isto é verdade em casos em que o cliente já ambiciona há muito por eles ou mesmo sem ter conhecimento da sua existência. Estes casos representam uma oportunidade perdida, com prejuízo tanto para os clientes como para a empresa.

Os sistemas de recomendação (SR) têm como objetivo descobrir e apresentar itens (produtos, serviços ou conteúdos) que vão de encontro às preferências e necessidades dos utilizadores. Tornam-se assim uma poderosa ferramenta para as empresas com comércio eletrónico, que permite rentabilizar o seu negócio e ao mesmo tempo facilitar a vida ao consumidor na satisfação das suas necessidades.

Dentro das várias técnicas de SR, a filtragem colaborativa (FC) é das mais utilizadas hoje em dia. Parte de um pressuposto muito simples: as aquisições feitas por utilizadores com os mesmos gostos de um dado utilizador, têm grande probabilidade de serem também do agrado deste.

Apesar da importância destes métodos, eles não estão implementados nalgumas das ferramentas de *Data Mining* (DM) mais populares, como o Rapid Miner (RM). O objetivo deste projeto é a implementação de um método de FC no RM usando apenas os operadores disponibilizados pela ferramenta. A implementação é testada num problema de recomendação de conteúdos numa rede social de música.

Palavras chave: sistemas de recomendação, filtragem colaborativa, comércio eletrónico, preferências, utilizadores, Rapid Miner

Abstract

A large number of products and services are available through electronic commerce.

Considering an individual site, there are often products which are not presented to its customers, despite being interesting to them. This is true in cases when the customers already desire them or even when they are not aware of their existence. This cases represents a lost opportunity, affecting both the customers as the company.

Recommender systems (RS) aim to find and present items (products, services and contents) that are suitable to fulfill customer preferences and needs. They represent a powerful tool for companies involved in e-commerce, enabling them to monetize their business while making the life of their customers easier by better satisfying their needs.

Among the existing RS techniques, collaborative filtering (CF) is the most widely used. They are based on a simple assumption: the purchases made by other users with preferences similar to a given user are likely to be interesting for the latter.

Despite the importance of these methods, they are not available in many of the most common data mining tools, such as Rapid Miner (RM). The goal of this project is to implement a CF method in RM using only the operators which are available in the tool. The implementation is tested in a problem of recommending content in a social network focused on music.

Keywords: recommended systems, collaborative filtering, e-commerce, preferences, users, Rapid Miner

Índice geral

Índice de figuras

Índice de tabelas

Lista de abreviaturas

SR - Sistema de Recomendação

FC - Filtragem Colaborativa

GUI - *Graphic User Interface*

XML - *eXtensible Markup Language*

IR - *Information Retrieval*

DM - *Data Mining*

RM - Rapid Miner

LSI - *Latent Semantic Indexing*

SH - Sistemas Híbridos

FCUB - Filtragem Colaborativa *User Based*

FCIB - Filtragem Colaborativa *Item Based*

XML - *Extensible Markup Language File*

CSV - *Comma-Separated Values*

1 - Introdução

Este capítulo introdutório consiste basicamente na apresentação da dissertação. Indica qual o seu enquadramento e contexto, a motivação do projeto. Descreve o problema abordado e os seus objetivos. Por fim, segue-se uma descrição da estrutura e organização da dissertação.

1.1 - Enquadramento e contexto

Cada vez mais, a quantidade de informação disponível na *Internet* cresce a um ritmo alucinante. A facilidade e a liberdade de inserir conteúdos fomentam esse crescimento sem nos apercebermos.

Por natureza, o ser humano não é capaz de acompanhar esse crescimento em tempo útil, tornando difícil encontrar e selecionar a parte da informação que mais lhe interessa.

Se por um lado, o acesso à informação é mais facilitado, por outro, os utilizadores são muitas vezes confrontados com conteúdos redundantes, irrelevantes, ofensivos ou mesmo falsos.

Para o utilizador, que está interessado numa pequena parte dos dados, torna-se imprescindível ter soluções que permitam classificar em níveis de importância as informações retornadas, tendo em vista uma procura de informação mais eficiente, restrita e direta ao assunto.

Devido ao grande volume de informação existente, é necessário esforço por parte das entidades envolvidas no processo de tomada de decisão. Numa época de grandes exigências e de rigor, em que o tempo é um bem precioso, a aquisição rápida da informação relevante é um valor acrescido.

É sempre bom ter alguém ou algum agente virtual que nos possa ajudar na hora de escolher determinados itens que vão de encontro às nossas necessidades ou preferências.

Assim, urge a necessidade de ferramentas e de técnicas que, de uma forma inteligente e eficiente permitam ajudar o utilizador na procura de informação útil na *Web*.

Para colmatar ou minimizar estes problemas, surgiram os SR, como sendo um conjunto de técnicas que ajudam o utilizador na filtragem dessa mesma informação.

1.2 – Motivação

O RM é uma ferramenta de DM com sucesso crescente. No entanto, o RM não tem de base algoritmos nem operadores específicos para a criação de SR.

No início deste projeto, existiam alguns processos de SR para o RM desenvolvidos por terceiros (disponíveis no site MyExperiment[1]). Estes processos apresentam algumas limitações: não são fáceis de usar, estão muito dependentes de um formato específico de dados que não é muito comum e são pouco flexíveis.

Depois de este projeto estar em desenvolvimento foi conhecida uma extensão[2] completa de SR para RM. No entanto, esta implementação é apoiada em código externo ao RM, o que dificulta a sua compreensão e impossibilita a introdução de alterações.

1.3 – Problema proposto e objetivos

O objetivo principal desta tese consiste em avaliar as capacidades do RM na implementação de "raiz" do algoritmo de FC *user based* (FCUB) usando apenas os operadores base do RM.

Pretende-se que este processo seja mais completo e flexível relativamente aos disponibilizados no MyExperiment. Por outro lado, pretende-se que facilite o seu desenvolvimento no futuro. Por exemplo, deve ser possível substituir a função de distância utilizada aqui por outras que entretanto sejam disponibilizadas no RM.

1.4 – Estrutura da dissertação

A dissertação está estruturada tendo em vista uma melhor organização e sequência lógica dos assuntos abordados.

Assim sendo, esta dissertação é constituída por seis capítulos, incluindo a Introdução, Conclusões e Referências.

No capítulo atual é apresentado o enquadramento e contexto, a motivação, o problema proposto e objetivos e por fim, a estrutura da dissertação.

[1] http://www.myexperiment.org
[2] A extensão faz parte do projeto e-lico: http://www.e-lico.eu/recommender-extension.html

O segundo capítulo incide sobretudo na componente teoria dos SR: além da sua definição é apresentado o contexto histórico, a importância no comércio eletrónico, formas de aquisição de informação (implícita e explícita), divisão dos vários métodos em categorias, exemplos reais de utilização e as vantagens e desvantagens dos SR. Este capítulo foca a avaliação em SR e à metodologia de avaliação usada neste processo.

O terceiro, intitulado 'Filtragem Colaborativa em Rapid Miner', faz uma introdução teórica à tecnologia usada e descreve detalhadamente passo a passo todo o processo implementado.

O quarto capítulo, intitulado 'Resultados Experimentais' tem como objetivo validar o processo usando dados de um caso real. Começa por fazer uma descrição dos dados usados, bem como uma breve análise exploratória dos mesmos. Segue-se a descrição das condições experimentais e por fim a apresentação e discussão dos resultados.

Nos dois capítulos restantes segue-se, respetivamente as conclusões e as referências.

2 - Sistemas de Recomendação

Este capítulo faz uma introdução teórica aos SR: a sua definição, contexto histórico, a sua importância no comércio eletrónico, formas de aquisição de informação, divisão dos vários métodos em categorias, avaliação em SR, exemplos reais da sua utilização e por fim discussão de algumas vantagens e desvantagens.

2.1 - Definição

De uma forma resumida, pode-se dizer que é uma tecnologia que usa técnicas estatísticas e de descoberta de conhecimento com o objetivo de fazer recomendações de itens[1] a utilizadores baseando-se num histórico de atividades anteriores.

A tarefa de recomendação pode ser vista como um problema de previsão: o sistema tenta prever a utilidade de determinados itens a um utilizador e depois ordena-os de acordo com os valores de utilidade previstos. A utilidade de um item é normalmente representada por um valor numérico que reflete o grau de interesse previsto do utilizador nesse item.

O resultado de um SR é normalmente um conjunto de itens ordenados decrescentemente pela utilidade prevista para um determinado utilizador (Mobasher, 2007).

Genericamente, um SR é constituído pelos seguintes componentes:

- **Item** - algo que pode ser recomendado ao utilizador;
- **Utilizador** - entidade a quem foram ou vão ser feitas recomendações de itens;
- **Conhecimento** - o que é adquirido pelo sistema sobre os procedimentos de recomendação;
- **Histórico** - registo das atividades resultantes da interação ente os utilizadores e os itens (compras, votações, visitas, acessos, etc.).

[1] Neste caso, quando se refere a itens, pode-se igualmente referir a produtos, serviços, pessoas ou mesmo páginas *Web* dependendo do contexto a que se está a referir.

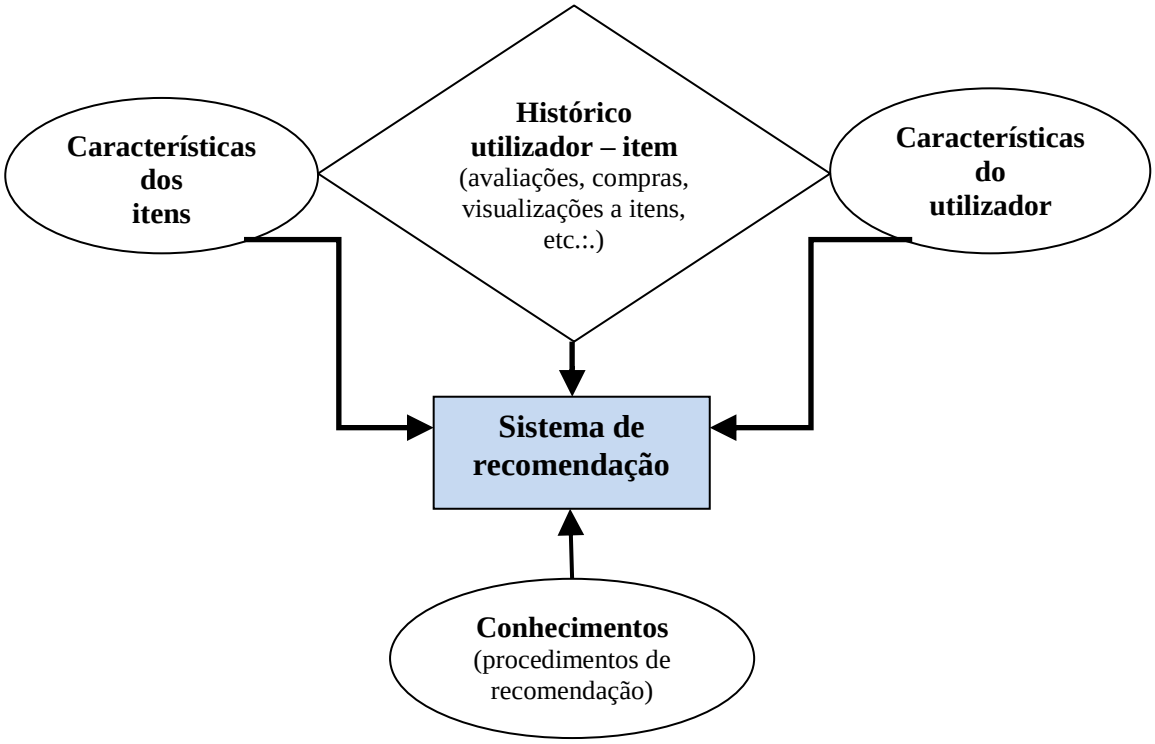

Figura 2.1 – Componentes de um SR

2.2 – Contexto histórico

Os SR tornaram-se uma importante área de investigação desde meados da década de 90 (Adomavicius e Tuzhilin, 2005).

Segundo Oliveira (2007), o primeiro SR divulgado publicamente foi o *Tapestry* (em 1992). Consistia na seleção de documentos de *email*. Neste sistema, os utilizadores criavam regras (filtros) que representavam as suas preferências pessoais. Estas regras eram então comparadas com conteúdos de documentos e aqueles que respeitassem as regras eram então enviados para os respetivos utilizadores. Além disso, o *Tapestry* possibilitava aos utilizadores a apreciação de documentos (se gostaram ou não). Isto permitia que utilizadores com gostos semelhantes acedessem aos documentos apreciados positivamente uns dos outros, colaborando na filtragem das mensagens (Pereira, 2007).

2.3 – Importância no comércio eletrónico

O comércio eletrónico é atualmente das áreas que maior proveito tem da utilização dos SR. Usa diferentes técnicas para divulgar os produtos/serviços mais adequados aos seus clientes, para deste modo aumentar o seu lucro (Oliveira, 2007).

Permite por exemplo que, o comércio eletrónico se torne personalizado ao cliente, resultando eventualmente num maior volume de negócio e rentabilidade. Permite às organizações estabelecer melhores relações com os seus clientes ao dar-lhes exatamente o que eles procuram, no momento certo de modo a minimizar a taxa de abandono.

Permite de forma bastante explícita aplicar técnicas de *marketing* tais como o *cross-selling* e *up-selling*: quando um cliente compra uma máquina fotográfica digital pode lhe ser recomendado a compra de produtos relacionados, como por exemplo cartões de memória, tripé, bolsa para a máquina, etc. Por outro lado, quando um cliente compra um computador portátil pode-lhe ser sugerido a compra de um computador idêntico mas ligeiramente mais caro com melhores características (mais memória, melhor placa gráfica, etc.) bem como a realização de ofertas promocionais quando há a previsão que o cliente tende a comprar um produto numa empresa concorrente.

A importância e o impacto económico desta área de investigação reflete-se por exemplo no Netflix Prize[2]. Ou seja, num concurso/desafio de um milhão de dólares para quem conseguisse implementar um SR com uma taxa de erro no mínimo 10% mais baixa do que o existente na Netflix (Hashsler, 2011).

A criação de novas técnicas e a avaliação dos SR nos problemas do mundo real têm sido uma área ativa de investigação (Melville e Sindhwani, 2010).

2.4 – Formas de aquisição de informação

Para efetuar recomendações, os algoritmos de SR precisam de dados (*inputs*) para processar. Ou seja, precisam de conhecer minimamente os utilizadores (itens preferidos, dados pessoais, gostos, áreas de interesse, etc.) e os itens envolvidos (características, palavras-chave, etc.).

[2] Mais informações em http://www.netflixprize.com

A qualidade e a relevância dos dados são extremamente importantes para que os resultados sejam os mais corretos possível.
Mas, nem sempre se deve utilizar todos os dados existentes, sob pena de os SR se tornarem lentos e menos eficientes. Por isso, é importante fazer uma seleção para perceber quais são os *inputs* que são realmente importantes para o sistema em causa.

Segundo Adomavicius e Tuzhilin (2005), a obtenção de informação pode ser feita de duas formas:

2.4.1 – Explícita

O utilizador é questionado sobre os dados que o sistema necessita para fazer recomendações. É normalmente sob a forma de questionários, avaliações, gostos (*likes*), etc..
Apesar de esta ser a forma mais fácil para o sistema[3], não deve ser demasiado intrusivo porque pode aborrecer o utilizador com demasiadas perguntas e este nem sempre estar disponível nem motivado para responder. Devem ser feitas questões que realmente interessem. Por exemplo, se estivermos numa loja *online* de produtos de higiene, é de extrema importância perguntar o sexo do utilizador para que não se efetuem recomendações sem sentido, como por exemplo, *aftershaves* a senhoras. Por outro lado, perguntar qual o estado de saúde em casos de recomendação para compra de filmes, não é relevante nem faz sentido.
Este tipo de aquisição de informação pode ser vantajoso para o caso de novos utilizadores (quando ainda não lhe foram feitas recomendações nem se sabe nada acerca dos seus gostos ou preferências). Outras das vantagens é o facto de se poder obter informações mais concretas, variadas e provavelmente mais corretas; exigir menos "trabalho" por parte dos sistemas para obter essas informações.

[3] Pode-se denominar também de sistema reativo (Anand e Mobasher, 2005).

Os tipos de informação que podem ser obtidos são:

- **Dados pessoais:**

  Elementos como a idade, sexo, estado civil, habilitações literárias, naturalidade, etc. São importantes porque existem diferenças de preferências e necessidades entre grupos de pessoas: jovens e idosos, mulheres e homens ou casados e solteiros;

- **Preferências temáticas ou áreas de interesse**:

  O sistema pode verificar quais os itens que estão englobados nas áreas de interesse definidas pelo utilizador, recomendando aqueles que mais se ajustem nessas áreas. Por exemplo, se o utilizador indicar que gosta de culinária há maior probabilidade que lhe seja recomendado um livro sobre receitas culinárias do que um livro sobre astronomia.

  A figura 2.2 apresenta um exemplo real deste tipo de aquisição de informação:

Figura 2.2 – Exemplo real de aquisição de informação explícita por preferências temáticas

- **Avaliação de itens:**

  O utilizador pode indicar a sua opinião sobre um ou um conjunto de itens. A escala utilizada para a avaliação pode ser numérica, como por exemplo, de 1 a 5 em que (1 - mau, …, e 5 - excelente), binária que permite uma avaliação mais limitada, mas mais concreta (gosto/não gosto, etc.) (Vozalis e Margaritis, 2003).

A figura 2.3 apresenta dois exemplos reais do tipo de aquisição de informação explícita:

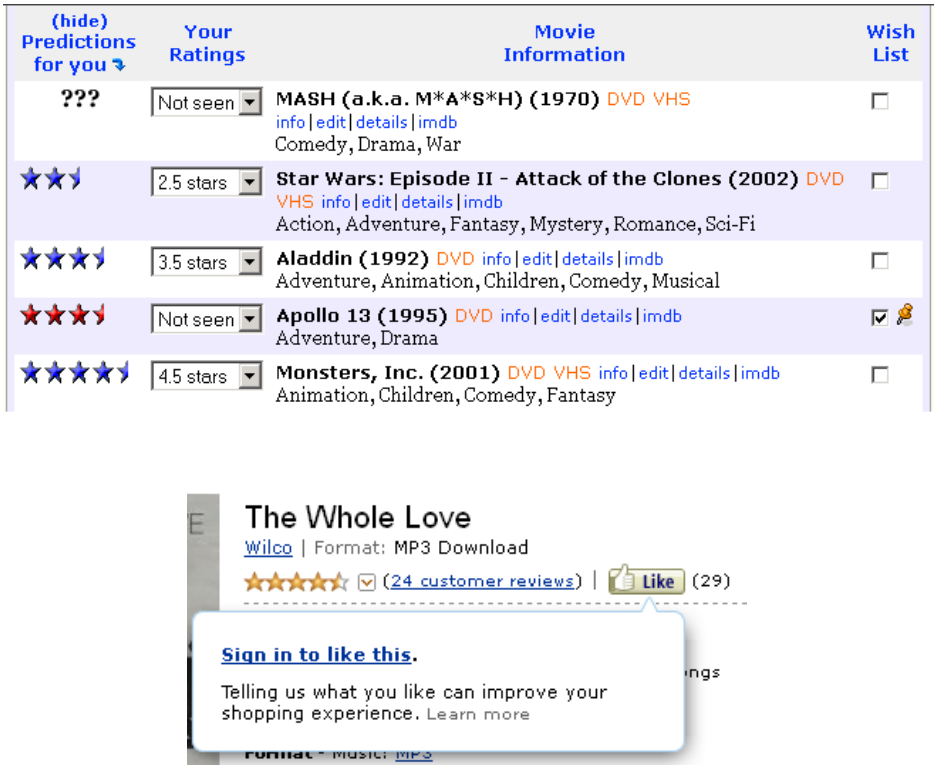

Figura 2.3 – Alguns exemplos reais de aquisição de informação explícita por avaliação de itens

2.4.2 - Implícita

Neste caso, o utilizador não é questionado diretamente sobre os dados que o sistema[4] necessita para recomendar.

O sistema terá de monitorizar todas as ações/comportamentos relevantes do utilizador e tentar inferir desses dados os seus os gostos e preferências. Cada interação do utilizador contribui para a recolha de informações.

Segundo Nichols (1997), existe uma série de ações que pode ser interpretada como avaliações implícitas: comprar, avaliar, gravar/imprimir, apagar, referenciar, responder, marcar, ler, consultar, sequência de páginas *Web* que visitou, tempo de permanência em cada página, etc.

Tem como vantagens o facto de não obrigar o utilizador a perder tempo a responder aos questionários. Por outro lado, tem a desvantagem de poder inferir preferências erradas

[4] Pode ser chamado de sistema proactivo (Anand e Mobasher, 2005)

(por exemplo, cinco minutos na mesma página pode significar que o utilizador se interessa ou que saiu para tomar café).

2.5 – Categorização

Os métodos de recomendação podem ser divididos em categorias. Mas, existem opiniões divergentes quanto forma de categorização dos SR. Diferentes autores têm a sua perspetiva.

A opinião mais consensual é a divisão em três tipos de filtragem (Adomavicius e Tuzhilin, 2005). Alguns autores, como Montaner (Cazela *et al.*, 2010), defendem que existe uma quarta abordagem: a filtragem demográfica.

2.5.1 – Filtragem baseada em conteúdo (*content based filtering*)

Na filtragem baseada em conteúdo (FBC), a recomendação de um item a um utilizador baseia-se na semelhança com itens adquiridos ou vistos por este no passado (Adomavicius e Tuzhilin, 2005). A semelhança tem em conta as descrições de conteúdo dos itens (um conjunto de termos ou palavras, sinais áudio, imagens, etc.) que representam os seus atributos ou palavras-chave (Adomavicius e Tuzhilin, 2005). Por exemplo, um livro pode ser caracterizado pelo tema, autor, editora, etc.

O perfil do utilizador e de cada item podem ser representados por vetores de palavras e seus pesos. Assim, é possível determinar quais as palavras mais relevantes para um determinado utilizador.

A ideia base é "*comparar conteúdos dos itens de forma a recomendar itens semelhantes aos que o utilizador achou relevantes no passado*" (Venâncio e Nóbrega, 2011).

Figura 2.4 – Filtragem baseada no conteúdo

Por exemplo, num site de visitas virtuais, se o utilizador indicar que gosta de ver a Torre dos Clérigos há maior probabilidade que goste de ver a Sé do Porto do que por exemplo o Estádio do Dragão ou a Casa da Música, visto que os primeiros tratam-se de monumentos antigos. Quando alguém ouve uma música, são analisadas as suas características intrínsecas (género, grupo, etc.) e é-lhe recomendado uma música dentro de mesmo género ou do mesmo grupo musical.

A identificação de similaridades entre itens permite por exemplo, recomendar livros do mesmo autor, filmes do mesmo género, etc..

A FBC é muito utilizada nas recomendações de textos, de páginas *Web* ou outros conteúdos essencialmente textuais.

2.5.1.1 - Vantagens e desvantagens da FBC

Algumas das vantagens deste tipo de filtragem:

- A sugestão de um item a um utilizador não requer que esse item seja avaliado por outros utilizadores, pois o critério de recomendação baseia-se apenas na comparação das descrições de conteúdo, independentemente se o item foi ou não escolhido por mais alguém;
- Todos os itens são comparados com o perfil do utilizador, tendo por isso, todos igual probabilidade de serem recomendados.

No entanto, existem algumas limitações:

- Super-especialização. Um utilizador com preferências muito limitadas (poucas ou muitas na mesma categoria de itens), sofre o problema da pouca variabilidade de sugestões.

  Se por exemplo, demonstrar interesse em filmes de terror, dificilmente ser-lhe-á recomendado outras categorias de filmes, por potencialmente interessantes que sejam para o utilizador.

  Para resolver este problema, é possível introduzir alguma aleatoriedade no sistema, de forma a surgirem outras recomendações diferentes mas que façam sentido;
- Tendo em conta o conceito de FBC, há o risco de que as recomendações sejam muito semelhantes às anteriores (Adomavicius e Tuzhilin, 2005).

Por exemplo, não é ideal recomendar todas as notícias que abordem o mesmo tema, pelo que se deve dar sugestões que sejam suficientemente diferentes;

- Este tipo de sistemas têm dificuldades em identificar termos sinónimos como por exemplo 'carro' e automóvel'. Não permite por exemplo, verificar se um artigo está ou não bem escrito (Adomavicius e Tuzhilin, 2005).

  Para resolver este problema pode ser usado um dicionário de sinónimos (Valdemir, 2008).

2.5.2 – Filtragem colaborativa (*collaborative filtering*)

A filtragem colaborativa[5] (FC) foi proposta para resolver falhas da FBC (Valdemir, 2008).

Utiliza uma metodologia diferente, uma vez que não requer conhecimento de conteúdo dos itens (Valdemir, 2008).

Como o próprio nome indica (colaborativo), as recomendações feitas a um utilizador são baseadas nos itens que lhe são desconhecidos mas que foram relevantes para utilizadores com perfis semelhantes de preferências. Se conhecermos alguém que tenha os mesmos gostos do que nós, é muito provável que uma nova aquisição dessa pessoa seja também do nosso interesse. É esta a ideia subjacente deste método de filtragem.

Os algoritmos de FC foram apresentados por Breese *et al*. (1998) divididos em duas classes: baseados em memória e em modelo:

2.5.2.1 – Baseados em memória (*memory based*)

Como o próprio nome indica, nesta classe de métodos, os dados são guardados em memória e são calculadas medidas de similaridade entre utilizadores e/ou entre itens, de cada vez que é solicitada uma recomendação.

[5] O termo 'filtragem colaborativa' foi usado pela primeira vez em 1992, por David Goldberg no seu artigo entitulado "*Using collaborative filtering to weave na information tapestry*" quando projetou o sistema Tapestry (Segaran, 2007).

Podemos considerar as seguintes abordagens:

1 - User Based

Com uma função que mede a similaridade entre utilizadores (eq. 2.1 ou eq. 2.3), é possível determinar quais os utilizadores com os mesmos gostos. Assim, verifica-se os itens que estes mais apreciam para que se possa recomendar ao utilizador da sessão.

Ao usar recomendações de outros utilizadores é possível tratar qualquer tipo de conteúdo e recomendar itens, mesmo que esses não sejam semelhantes aos itens já avaliados pelo utilizador. Assim, é possível recomendar filmes a um utilizador mesmo que este só tenha visto livros.

Os dados são normalmente representados por uma matriz de dimensões *n* por *m* (*n* utilizadores e *m* itens). Cada célula da matriz $r_{i,j}$ pode conter um valor que relaciona o utilizador *i* ao item *j*. Estes valores têm uma determinada escala e significado, desde intervalares (ex.: 1 a 5) que pode corresponder a uma avaliação ou até dados binários (0 e 1) que indica se um item foi visto (é da preferência) ou não de um determinado utilizador.

Independentemente da escala e do significado dos valores das células, estas matrizes são normalmente esparsas. Isto é, comparativamente com a sua dimensão, muito poucas são as células preenchidas (assumindo que os zeros não são representados) porque muitos dos utilizadores não tem relação com muitos dos itens.

A tabela 2.1 apresenta um exemplo de uma matriz binária, em que o valor 1 indica que os utilizadores compraram os itens indicados na intersecção das respetivas células.

Neste caso, o utilizador da sessão (que pede recomendações) é o Pedro:

	item 1	item 2	item 3	item 4	item 5	item 6
Paulo		1			1	
João	1	1				
Márcia			1	1	1	
Carlos			1			
Ana	1			1		
Pedro	?	1			?	

Tabela 2.1 – Exemplo simples da aplicação da FC na recomendação de itens

Se quisermos recomendar um item ao Pedro, temos de procurar outros utilizadores com preferências semelhantes ao Pedro. Neste caso, o Paulo e o João já compraram itens que o Pedro também comprou (item 2 – representado na tabela 2.1 a azul). Assim, é recomendado ao Pedro itens que o Paulo e o João compraram, mas que o Pedro ainda não comprou (que neste caso é o item 1 e o 5 – representado na tabela 2.1 a vermelho).

Tipicamente, o processo de FCUB passa pelos seguintes passos:

- Determinar o índice de similaridade entre o utilizador da sessão e cada um dos outros;
- Selecionar os *k* utilizadores com mais alto valor do índice de similaridade calculado anteriormente;
- Calcular o *score* ponderado de cada item, tendo em conta apenas os *k* utilizadores mais semelhantes;
- Dos itens anteriores, escolher apenas os *n* itens com maior *score*, mas que não estejam presente no utilizador da sessão.

Umas das medidas mais usadas para medir o índice de similaridade entre utilizadores é o coeficiente de correlação de *Pearson* (Melville e Sindhwani, 2010) (eq. 2.1).
O valor resultante $W_{a,u}$, indica a similaridade das avaliações dadas aos itens pelo utilizador *u* e o da sessão *a*.

$$w_{a,u} = \frac{\sum_{i \in I} \left(r_{a,i} - \bar{r}_a\right)\left(r_{u,i} - \bar{r}_u\right)}{\sqrt{\sum_{i \in I} \left(r_{a,i} - \bar{r}_a\right)^2 \sum_{i \in I} \left(r_{u,i} - \bar{r}_u\right)^2}} \quad (2.1)$$

onde *I* é o conjunto de itens avaliados por ambos os utilizadores, $r_{u,i}$ é a avaliação dada pelo utilizador *u* ao item *i* e $\bar{r}_u$ é a média das avaliações dadas pelo utilizador *u* (Melville e Sindhwani, 2010).
Para aplicar este método, é necessário que haja pelo menos dois itens avaliados em comum. Se houver apenas um, o denominador é zero e o resultado é impossível de calcular.

Os valores de $W_{a,u}$ variam entre -1 e 1, em que 1 indica similaridade total e -1 dissimilaridade total.

A previsão (*score*) do item *i* para o utilizador *a*, pode ser definida da seguinte forma (eq. 2.2):

$$\rho_{a,i} = \bar{r_a} + \frac{\sum_{u \in K} \left(r_{u,i} - \bar{r_u}\right) \cdot w_{a,u}}{\sum_{u \in K} w_{a,u}} \quad (2.2)$$

em que $W_{a,u}$ é o valor da similaridade entre os utilizadores *a* e *u*, e *K* é o conjunto dos vizinhos mais próximos (vizinhança) do *a*.

Outra medida muitas vezes usada é a similaridade do cosseno (eq. 2.3). Se considerarmos cada utilizador como um vetor multidimensional no espaço de itens, podemos usar o cosseno do ângulo formado por cada par de vetores como medida de similaridade entre dois utilizadores:

$$sim\,(u,v) = \cos(\vec{u},\vec{v}) = \frac{\vec{u} \,.\, \vec{v}}{\|\vec{u}\| \,.\, \|\vec{v}\|} \quad (2.3)$$

em que $\vec{u}$ e $\vec{v}$ representam os vetores da cada um dos utilizadores.

No caso particular de os dados serem binários, a similaridade do cosseno pode ser simplificada pela seguinte expressão (eq. 2.4) (Gunawardana e Shani, 2009):

$$sim\,(u,v) = \frac{N_{u,v}}{\sqrt{N_u} \cdot \sqrt{N_v}} \quad (2.4)$$

em que $sim\,(u,v)$ corresponde ao valor da similaridade entre os utilizadores u e v;

$N_{u,v}$ é o número de itens comuns entre os utilizadores u e v;

N_u é o número de itens que contêm o utilizador u;

N_v é o número de itens que contêm o utilizador v

Para a determinação dos *scores* dos itens usa-se a expressão (eq. 2.5):

$$p_{a,i} = \sum_{j \in Ka} sim(a,j) \,.\, w_{i,j} \qquad (2.5)$$

em que $p_{a,i}$ corresponde ao *score* obtido para o item *i* para o utilizador *a* (utilizador da sessão); *Ka* corresponde ao conjunto de vizinhos mais próximos do utilizador *a*; $sim(a,j)$ corresponde à similaridade entre os utilizadores *a* e *j*; $w_{i,j}$ indica se o item está ou não (1 ou 0), respetivamente no utilizador *j*.

No entanto, outras medidas de similaridade podem ser usadas, como por exemplo, a correlação de *Spearman*, correlação τ de *Kendal*, diferença média quadrática, entropia, etc. (Melville e Sindhwani, 2010).

2 - *Item based*

Com o aumento progressivo do número de utilizadores e de itens no sistema (na ordem dos milhões em alguns casos) (Melville e Sindhwani, 2010), o processo de recomendação *online* torna-se cada vez mais demorado para a FCUB.

De cada vez que é solicitada uma recomendação, comparar um utilizador com todos os outros e depois comparar todos os itens que cada utilizador avaliou, torna-se uma tarefa lenta. Além disso, é normal que possa haver poucos itens em comum o que torna difícil a tarefa de encontrar utilizadores semelhantes.

Este problema gera um custo computacional maior para determinar as similaridades entre utilizadores em tempo real, causando uma latência inaceitável e consequentemente tempo de resposta maior.

Para minimizar este problema, Lindem, Smith e York propuseram em 2003 uma variante do método padrão: a FC baseada no item (*item based collaborative filtering*) (FCIB) (Melville e Sindhwani, 2010). Em vez de a similaridade ser calculada entre utilizadores como acontece na FCUB, é calcula entre itens, não em termos do seu conteúdo, mas em termos de preferências que lhes são dadas pelos utilizadores.

Assim, o sistema recomenda os itens mais semelhantes aos itens dos "vizinhos" do utilizador da sessão.

Devido ao facto dos relacionamentos entre itens não alterarem tão frequentemente como entre utilizadores, muitos dos cálculos podem ser feitos *offline* e *à priori*, tornando os

SR mais rápidos *online*. Os cálculos podem ser feitos quando o sistema tem uma sobrecarga baixa de tráfego ou até mesmo num computador diferente da aplicação principal.

Uma das formas de calcular a similaridade entre os itens *i* e *j* ($w_{i,j}$) pode ser definida pela seguinte expressão (eq. 2.6):

$$w_{i,j} = \frac{\sum_{u \in U} \left(r_{u,i} - \bar{r}_i\right)\left(r_{u,j} - \bar{r}_j\right)}{\sqrt{\sum_{u \in U} \left(r_{u,i} - \bar{r}_i\right)^2} \sqrt{\sum_{u \in U} \left(r_{u,j} - \bar{r}_j\right)^2}} \quad (2.6)$$

onde, *U* é o conjunto de todos os utilizadores que avaliaram ambos os itens; $r_{i,j}$ é a avaliação dada pelo utilizador *u* ao item *i* e $\bar{r}_i$ é a média das avaliações obtidas pelo item *i* (Melville e Sindhwani, 2010).
O *score* do item *i* para o utilizador *a* pode ser calculado pela seguinte expressão (eq. 2.7):

$$\rho_{a,i} = \frac{\sum_{j \in K} r_{a,j}\, w_{i,j}}{\sum_{j \in K} |w_{i,j}|} \quad (2.7)$$

onde *K* é o conjunto dos vizinhos mais próximos do utilizador a (Melville e Sindhwani, 2010).

2.5.2.2 – Baseados em modelos (*model based*)

Em vez de usarem cálculos de distâncias ou similaridades, estes algoritmos caracterizam-se pela criação de modelos.
Estes modelos são criados a partir do histórico das preferências do utilizador e usados mais tarde na recomendação (Breese *et al.*, 1998; Oliveira, 2007), enquanto que na FC o histórico é usado no momento da recomendação.
Os algoritmos baseados em modelos utilizam uma aproximação probabilística para determinar o valor esperado de uma previsão. Para o utilizador ativo pretende-se prever a votação para itens ainda não vistos por ele.

Assumindo que os votos têm um valor inteiro que varia entre 0 e m, temos (Breese *et al.*, 1998):

$$p_{a,j} = E(v_{a,j}) = \sum_{i=0}^{m} P(v_{a,j} = i \,|v_{a,K}, k \in I_a) i$$

em que p é a probabilidade de o utilizador ativo realizar determinado voto no item j dados os votos previamente observados (Breese *et al.*, 1998).

De uma forma geral, estes algoritmos produzem recomendações mais rápidas, apesar de precisarem de algum tempo para criar o modelo.

Podem ser usados diversos algoritmos para obter os modelos, tais como métodos de agrupamento, redes *bayesianas*, regras de associação ou modelos de factorização de matrizes (Breese *et al.*, 1998).

2.5.2.3 - Vantagens e desvantagens da FC

Pode-se considerar como algumas das vantagens da FC (Valdemir, 2008):

- São independentes das características intrínsecas dos itens, permitindo que, em conjunto possam ser recomendados itens de diversas categorias, como por exemplo: livros, CDs, filmes, etc.;
- O facto de a capacidade de produzir recomendações de qualidade depender da avaliação (positiva ou negativa) feita por outros utilizadores. O sistema por si só não consegue distinguir um item de qualidade dos outros apenas pelo seu conteúdo;
- Outra das vantagens é a capacidade de o sistema surpreender o utilizador com recomendações inesperadas, mas com qualidade. Esta ação normalmente é chamada de *serendipity*.

Apesar de todas as suas vantagens, apresenta as suas próprias limitações:

- **Esparsidade**

  O número de utilizadores e de itens tende a ser enorme. Na maior parte dos casos, o número de avaliações já obtidas é muito mais pequeno em relação ao número de avaliações que necessitam de ser previstas o que se vai traduzir em

muitas combinações utilizador/item não relevantes (células com zero ou valor indefinido).

Por exemplo, num SR de filmes, os filmes que forem avaliados por poucos utilizadores são raramente recomendados, mesmo que estes atribuam pontuações elevadas.

- **Falso Vizinho ou 'Ovelha negra'**

  Acontece quando utilizadores são reconhecidos como semelhantes para o sistema, mas na realidade, esses utilizadores não possuem preferências parecidas para outros itens. Pode acontecer que utilizadores avaliem alguns itens iguais e nos perfis dos mesmos possuam poucos itens. Os utilizadores com mais itens avaliados em comum têm muito mais probabilidade de realmente serem semelhantes (Valdemir, 2008).

- **Inicialização (*cold start problem*)**

  Este problema divide-se em dois subproblemas (o novo utilizador e o novo item):

  - O utilizador precisa de avaliar um número suficiente de itens para que o SR possa perceber os seus gostos e apresentar recomendações mais próximas possíveis das suas preferências. Para resolver este problema, a maioria dos sistemas usa a técnica da recomendação híbrida, combinando a FBC com a FC (discutido na secção 2.5.3);
  - Enquanto um novo item não for recomendado por um número considerável de utilizadores, o sistema não estará apto a recomendá-lo. Este facto, pode novamente ser colmatado com a utilização de um sistema híbrido (discutido na secção 2.5.3).

2.5.3 – Sistemas híbridos

Os sistemas híbridos (SH) são a combinação de pelo menos dois tipos de filtragem. O caso mais comum é a junção da FC com a FBC (Melville e Sindhwani, 2010).

A motivação principal para a criação dos SH é o aumento da qualidade das recomendações (Pereira, 2007) porque permite colmatar as desvantagens de cada um dos tipos de filtragem e aproveitar as suas vantagens.

Segundo Adomavicius e Tuzhilin (2005), existem diferentes formas de agregar estes dois tipos de filtragem:

- Combinar as recomendações obtidas pelos dois tipos de filtragem em separado:

  Nesta forma, existem dois cenários: combinar as recomendações obtidas numa única, usando uma combinação linear ou um esquema de votos; avaliar para cada recomendação qual dos sistemas gera recomendações mais exatas, determinando o grau de confiança de cada uma e escolher o maior ou o mais consistente com as avaliações realizadas pelo utilizador no passado.

- Incorporar algumas características da FBC na FC:

  Utiliza-se a FC mas acrescenta-se características da FBC, como a manutenção do perfil relativo às características dos itens que cada utilizador já avaliou. Esta estratégia permite reduzir o problema da escassez de itens comuns e resolve o problema dos novos itens pois podem ser sugeridos com base no perfil do utilizador.

  Por vezes, são utilizados mecanismos automáticos (*filterbots*) que realizam análises baseadas no conteúdo, funcionando assim como agentes inteligentes no sistema de FC. Perante isto, os utilizadores que tiverem preferências semelhantes aos *filterbots* obtêm recomendações mais exatas.

- Incorporar algumas características da FC na FBC:

  Nesta forma, os sistemas usam normalmente a técnica de redução da dimensionalidade aplicada ao grupo de perfis baseados no conteúdo. Podem utilizar *latent semantic indexing* (LSI)[6] para criar perfis de utilizadores tipo CF, onde os utilizadores são representados como termos de vetores,

[6] Permite descobrir padrões relativos à forma como as palavras são utilizadas num conjunto de documentos. Ao agrupar as palavras podem surgir co-ocorrências que caracterizam grupos de documentos.

aumentando a performance comparativamente aos sistemas com FBC (Adomavicius e Tuzhilin, 2005).

Neste caso, a técnica LSI é aplicada sobre o conjunto de perfis (quer sobre o perfil de CF, quer sobre o perfil do sistema FBC) em vez do conjunto de documentos, com o objetivo de obter as semelhanças entre perfis (Soboroff, 1999).

- Desenvolvimento de um modelo único unificador de recomendação:

  Como o próprio indica, esta forma resume-se à combinação de duas das técnicas para a criação de um modelo de recomendação. É exemplo disso a junção de um sistema de FBC com um classificador baseado em regras.

  Esta forma permite resolver os problemas do novo utilizador e do novo item que se verifica na fase inicial da FC, quando ainda não existe informação suficiente para gerar recomendações fidedignas (Adomavicius e Tuzhilin, 2005).

2.5.3.1 - Vantagens e desvantagens dos SH

Além das vantagens referidas anteriormente, pode-se acrescentar que, em alguns casos fornece recomendações mais precisas porque estes sistemas ultrapassam algumas das limitações das abordagens FBC (tais como a esparsidade) e da FC.

Diversos artigos científicos compararam empiricamente a performance dos SH com os sistemas de FBC e de FC puros e demonstraram que os SH podem fornecer recomendações mais precisas e robustas (Adomavicius e Tuzhilin, 2005).

Por outro lado, tem como uma das desvantagens o facto de serem sistemas mais complexos e caros de implementar.

2.6 – Avaliação em SR

A avaliação é uma etapa muito importante na conceção de um SR.

Permite, através de estudos empíricos, dar indicações se o modelo fornece boas ou más recomendações e deste modo analisar o comportamento e a performance do sistema quando testado num cenário real.

Através da avaliação podem ser comparadas diferentes técnicas e abordagens de recomendação de modo a refiná-las e torná-las mais eficazes. As conclusões podem ser tiradas com base nos resultados obtidos em cada uma das abordagens.

No contexto dos SR existem diversas medidas que podem ser usadas na avaliação. Umas são mais apropriadas do que outras consoante vários fatores, como por exemplo o tipo de dados a usar: binários ou intervalares.
Algumas dessas medidas são: *mean absolute error* (MAE); *mean squared error* (MSE); *precision*, *recall* e F1 (discutidas mais à frente); Curvas ROC; *coverage*; *learning rate*; *novetly*; entre outras. Destas, *precision*, *recall* e F1 são das mais usadas (Alípio *et al.*, 2011; McCarey *et al.*, 2003; Vozalis e Margaritis, 2003). Estas três medidas surgiram há algumas décadas atrás em sistemas de *Information Retrieval* (IR).

2.6.1 – Metodologia de estimação da performance

Existem várias metodologias de estimação da performance que são adotadas em diversos artigos científicos da área.
Uma das possíveis metodologias (que pode ser usada nas abordagens baseadas em memória e em que a estrutura de dados é uma matriz binária utilizadores *vs* itens) pode ser descrita da seguinte forma:

1. O conjunto total de utilizadores é dividido em dois subconjuntos: treino e teste. Escolher um dos utilizadores (que contenha pelo menos dois itens). Esse utilizador será denominado de ativo (u_a) e pertencerá ao conjunto teste. Os restantes compõem o conjunto treino, tal como representado na figura 2.5:

		i_1	i_2	…	i_m
Conjunto treino	u_1	1	0	0	1
	u_2	0	1	1	0
	…	1	1	1	0
Conjunto teste	u_a	0	1	0	1

Figura 2.5 - Divisão do conjunto de utilizadores em dois subconjuntos: treino e teste

2. Dos n itens que pertencem ao utilizador ativo, deve-se, de forma aleatória "esconder" (passar para 0) no máximo $n-1$ itens (para deixar pelo menos um observado – ver figura 2.6). Os itens relevantes são os escondidos.

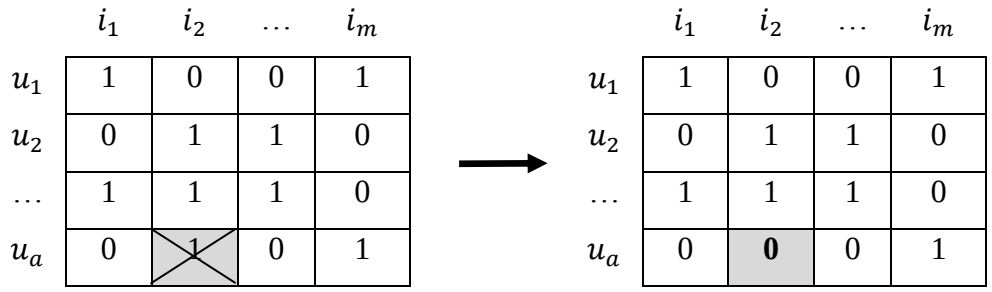

Figura 2.6 - Esquema resumido que representa parte da metodologia usada na avaliação

Por esta razão é que, um utilizador para ser ativo tem de ter pelo menos dois itens iniciais: um para esconder e o outro para manter observado.

3. O algoritmo de FC é executado (tendo em conta os pressupostos anteriores) com o objetivo de recomendar itens ao utilizador ativo;
4. A partir dos itens recomendados pelo algoritmo, é contabilizado o número de acertos, isto é, dos itens escondidos quantos foram recomendados. Este valor é normalmente chamado de número de *hits*.

2.6.2 – Medidas de avaliação

No caso mais simples, as medidas *precision* e *recall* podem ser adaptadas para o contexto dos SR da seguinte forma (Karypis, 2000):

$$precision = \frac{numero\ de\ hits}{N_r}$$

$$recall = \frac{numero\ de\ hits}{N_e}$$

em que $numero\ de\ hits$ corresponde ao número de itens comuns entre os conjuntos de relevantes e de recomendados; N_r ao número de itens recomendados e N_e ao número de itens relevantes.

Os resultados de ambas as métricas variam entre 0 e 1 (quanto mais alto for o valor melhor é a performance do modelo).

Pode-se dizer que a *precision* é a proporção de itens relevantes que foram recomendados de entre todos os recomendados e o *recall* é a proporção de itens relevantes que foram recomendados de entre todos os relevantes.

Se o valor da *precision* for 1 significa que todos os itens recomendados são relevantes, no entanto, não se sabe se todos os relevantes foram recomendados. Por outro lado, o valor 1 no *recall* significa que todos os itens relevantes foram recomendados, mas não se sabe quantas recomendações não são relevantes.

Estas duas medidas são inversamente relacionadas (Kim *et al.*, 2005) e sozinhas não permitem avaliar completamente o problema. Aumentando o número de itens recomendados, o *recall* tem a tendência a aumentar porque maior é a probabilidade de os itens relevantes serem selecionados. Ao mesmo tempo a *precision* diminui (Kim *et al.*, 2005).

Por isso, há a necessidade de uma nova medida (F1) que combine as duas anteriores. Esta medida é definida pela seguinte expressão:

$$F_1 = 2 \,.\, \frac{precision \;.\; recall}{precision + recall}$$

Tal como a *precision* e *recall*, F1 varia igualmente entre 0 e 1. Quanto maior for o valor melhor é o resultado da avaliação. É esta a medida que vai ser usada para avaliação da performance do algoritmo de recomendação.

Se pretendermos generalizar esta metodologia para mais do que um utilizador ativo (de cada vez), as fórmulas podem ser adaptadas para a seguinte forma (Kim *et al.*, 2005):

$$recall = \frac{\sum_{i \in A} |H_i \cap Top\_N_i|}{\sum_{i \in A} |H_i|}$$

$$precision = \frac{\sum_{i \in A} |H_i \cap Top\_N_i|}{N.|A|}$$

em que:

H_i é a lista de itens relevantes para o utilizador *i*;

N é o número total de itens recomendados para cada utilizador;

Top_N_i é a lista de recomendações para o utilizador *i*;

A é o número de utilizadores que tem pelo menos um item relevante;

O resultado final é assim um valor médio que tem em conta todos os utilizadores que foram contabilizados nos cálculos.

2.7 – Exemplos reais

Esta secção tem como objetivo apresentar alguns exemplos reais da aplicação de SR. Existem diversos casos de sucesso, desde sites de comércio eletrónico a sites de entretenimento:

1 - Amazon (http://www.amazon.com)

É atualmente o exemplo mais citado em toda bibliografia sobre este tema. É um site mundialmente conhecido que permite a venda *online* de diversos tipos de produtos.

Utiliza algoritmos colaborativos que personalizam o aspeto (lista de recomendações) do site de acordo com as preferências e interesses de cada cliente. Por exemplo, a apresentação de conteúdo informático para um engenheiro informático ou a apresentação de artigos de bebé para futuras mães.

Existe um apontador denominado "*your recommendation*", que leva o cliente a uma página onde pode filtrar as suas recomendações por áreas de interesse, avaliar produtos recomendados e ver a relação entre os itens (Ramos, 2010).

Figura 2.7 - Exemplo de uma página de recomendação do Amazon

De notar pela figura 2.7, a existência de algumas frases/expressões muito usadas no contexto dos SR, como por exemplo: "*avalie*", "*clientes que compraram x também compraram y*", "*explore itens semelhantes*".

2 - Palco Principal (http://palcoprincipal.sapo.pt)

O Palco Principal (PP) é uma rede social de música onde os artistas, os ouvintes e os profissionais do mundo da música se encontram. Oferece diferentes funcionalidades aos utilizadores registados: divulgar trabalhos musicais, ouvir músicas, criar as suas próprias *playlists*, adicionar músicas a essas *playlists* de acordo com as suas preferências, descartar músicas que lhe são recomendadas pelo sistema, entre outras (Palco Principal – site oficial, 2012).

O sistema recomenda novas músicas a um ouvinte baseando-se em preferências de ouvintes com os mesmos gostos musicais (FCUB) além de existir também algoritmos de FCIB.

Por outro lado, o sistema também percebe quais as preferências de um determinado ouvinte sabendo quais as músicas que ele incluiu na sua *playlist*. A partir dessas músicas o sistema vai procurar músicas semelhantes a essa (baseando no seu conteúdo), e são essas músicas que vão ser recomendadas (Palco Principal – blog, 2010).

Há medida que as músicas forem adicionadas à *playlist*, o sistema tornar-se cada vez mais robusto e eficaz e produzirá melhores recomendações (Palco Principal – blog, 2010).

Figura 2.8 - Exemplo de uma página de recomendação de músicas do Palco Principal

3 - Netflix (https://signup.netflix.com/global)

É um serviço *online* de aluguer de filmes em DVD (Melville e Sindhwani, 2010). Utiliza algoritmos colaborativos no seu SR chamado *Cinematch.*

A escala típica usada na avaliação dos filmes é de 1 a 5 em que 1 significa que não gosta e 5 que gosta muito (Melville e Sindhwani, 2010).

A base de dados é gigantesca. Em Abril de 2010, continha cerca cem milhões de avaliações dadas por cerca de meio milhão de utilizadores a milhares de títulos de filmes (Melville e Sindhwani, 2010).

Dada a importância que os SR tem no seu negócio, há algum tempo atrás, a empresa lançou um desafio, chamado 'Desafio Netflix' que premiava com um milhão de dólares quem apresentasse um SR com uma taxa de erro 10% mais baixa do que o atual. Tendo em conta o incentivo pelo prémio apareceram muitos candidatos com várias abordagens para a resolução do problema.

A análise dos resultados obtidos permitiu a elaboração de vários artigos científicos.
A equipa vencedora utilizou técnicas de factorização de matrizes (*matriz factorization*) usando álgebra linear e análise estatística de matrizes.

Figura 2.9 - Exemplo de um ecrã do Netflix

4 - Redes Sociais

A utilização de SR é também muito comum nas redes sociais tais como o *Facebook* ou *Hi5*. Permitem por exemplo, recomendar novos amigos ou outro tipo de sugestões.
Os algoritmos de recomendação utilizados em grande parte das redes sociais são baseados em FC e no conceito de vizinhança, mais precisamente nos ***k*** vizinhos mais próximos, definidos através da correlação entre utilizadores (Ramos, 2010).

Figura 2.10 - Exemplo de sugestão usado na rede social Facebook

2.8 – Desvantagens e problemas associados

Além das respetivas desvantagens inerentes a cada um dos tipos de filtragem, existem algumas que são generalistas:

1 - Questões de privacidade dos dados

Em geral, os SR não causam problemas mas, quando usados sobre dados controversos (como por exemplo, o sexo, raça, religião, orientação sexual para categorizar os indivíduos) podem causar preocupações como a invasão da privacidade.

Estas práticas podem ser contra a legislação anti-discriminação.

2 - Escalabilidade (*scability*)

Em muito casos, o número de utilizadores e de itens pode chegar à ordem dos milhões (Adomavicius e Tuzhilin, 2005). Por isso, é necessário ter estruturas de dados sofisticadas e arquiteturas avançadas para poder lidar com muitos utilizadores e itens ao mesmo tempo e a necessidade de haver algoritmos eficientes para extrair informação neste tipo de cenários.

Para dar resposta em tempo útil é necessário o uso de outras soluções para minimizar este problema, como por exemplo, o uso de redes *bayesianas*, técnicas de agrupamento, decomposição em valores singulares (*Singular Value Decomposition*) para reduzir o número de utilizadores e de itens.

3 – Filtragem Colaborativa em Rapid Miner

Este capítulo começa com uma breve apresentação da tecnologia usada. Segue-se uma descrição detalhada do processo implementado em RM (incluindo as suas diferentes fases, tendo como foco principal o algoritmo de FCUB).

3.1 - Tecnologia usada (Rapid Miner)

O RM é uma ferramenta de DM, *Text Mining*, análise de dados e *business intelligence*.
É desenvolvido em Java, permitindo a sua utilização versátil em qualquer sistema operativo e ambiente de trabalho.
O projeto RapidMiner começou em 2001 por Ralf Klinkenberg, Ingo Mierswa e Simon Fischer na Unidade de Inteligência Artificial da Universidade de Dortmund (Alemanha). Em 2006, Ingo Mierswa e Ralf Klinkenberg fundaram a empresa Rapid-I[1] (Rapid Miner – brochura oficial).

Esta ferramenta está disponível em duas versões (Rapid Miner – brochura oficial):

- *Community Edition*: gratuita, mas limitada em termos de funcionalidades e recursos. Mas, mesmo assim é usada em milhares de aplicações em todo o mundo;
- *Enterprise Editon*: é a versão profissional do *software* que, além de todas as vantagens da versão *Community* contêm soluções empresariais específicas para utilizadores profissionais. Possui igualmente capacidades avançadas de criação de relatórios e serviços específicos de garantia e assistência.

Na implementação do algoritmo de FC desta tese foi usada a versão *Community Edition* 5.2.00.

O RM é usado em investigação, educação, projetos experimentais e em aplicações industriais.

[1] http://rapid-i.com

De acordo com o jornal de DM '*KDnuggets*', o RM é líder mundial no *software* de código aberto de DM. É usado em milhares de aplicações em mais de 40 países por grandes companhias mundiais (Rapid Miner – brochura oficial).

Entre outras funcionalidades, o RM oferece aos seus utilizadores (Batista, 2010):

- Uma solução global para o desenho e implementação de um processo de DM. Permite a definição de todas as fases incluídas na metodologia CRISP-DM, classificadas e hierarquizadas segundo as tarefas descritas: seleção de dados, integração, transformação, pré-processamento, implementação de sofisticadas técnicas de modelação, validação dos modelos e algoritmos, etc.;
- Uma interface gráfica muito intuitiva e flexível para o desenho de um processo de DM;
- Mais de 500 operadores que implementam as mais diversas técnicas e algoritmos. Ligação direta com a biblioteca de classes de aprendizagem automática *Weka*, uma das mais utilizadas na comunidade científica especializada;
- Acesso às mais diversas fontes de dados: Excel, Access, Oracle, IBM DB2, Microsoft SQL Server, Sybase, Ingres, mySWL, Postgres, SPSS, etc.;
- Mais de 20 métodos para visualização de dados e modelos;
- Repositórios de processos, dados e meta-dados.

Todos os processos de DM podem ser executados na interface gráfica no modo GUI (*Graphic User Interface*). Cada um deles é armazenado num ficheiro XML (*eXtensible Markup Language*) (Batista, 2010).
Um processo é representado num sistema em árvore de operadores ou por um ambiente gráfico de fluxo de processo (*workflow*). Em ambos os casos, a estrutura do processo é ainda descrita internamente em XML, o que permite adicionalmente o desenvolvimento do processo nesta linguagem (Rapid-I GmbH, 2010).

3.2 – Implementação do algoritmo de recomendação

Antes de descrever a implementação do algoritmo de recomendação e com vista a contextualizar esta secção, todo o processo vai ser brevemente esquematizado na figura 3.1:

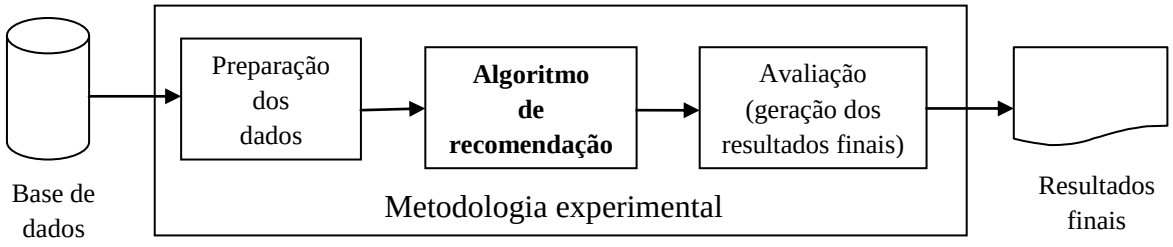

Figura 3.1 – Esquema resumido de todo o processo

Todo o processo foi construído usando uma estrutura modular. Após a leitura dos dados, a fase de preparação permite converte-los para uma estrutura adequada ao algoritmo de recomendação. Por fim, serão processados e impressos os resultados finais de avaliação. Nesta secção será descrito com detalhe apenas o algoritmo de recomendação (que é a parte principal da implementação).

Sempre que possível os operadores que compõem o processo foram agrupados em subprocessos tendo em conta o contexto e a tarefa para que foram destinados, permitindo uma interpretação e visualização mais fácil (por exemplo, um subprocesso para determinar os vizinhos mais próximos de um utilizador, outro para calcular os *scores* dos itens preferidos pelos vizinhos mais próximos, etc.).

Alguns operadores, pela sua natureza já são subprocessos (como por exemplo, o 'Loop Examples').

Houve a preocupação em definir nomes de atributos que sejam genéricos ('id_user' e 'id_item') para que o processo possa ser usado em qualquer contexto de dados.

Os parâmetros usados no algoritmo são os seguintes:

- `numero_recomendacoes`: Número de itens a recomendar para cada utilizador;
- `numero_vizinhos_mais_proximos`: Número de utilizadores com maior valor do índice de similaridade relativamente ao utilizador atual;
- `numero_itens_escondidos`: Número de itens a esconder em cada utilizador.

O valor mínimo para cada um dos parâmetros anteriores é 1.

A figura 3.2 ilustra o tipo de dados de entrada para o algoritmo. O primeiro atributo corresponde sempre ao utilizador (‘id_user’) e os restantes atributos aos itens (‘i1’, ‘i2’, … ‘i_n’).
As células da matriz estão sempre preenchidas e como se trata de um contexto binário, podem tomar valores (1 ou 0).

id_user	i1	i2	i3	i4	i5	i6	i7	i8	i9
1	0	0	0	1	1	0	1	0	0
2	0	1	0	0	0	1	1	0	0
3	0	0	1	1	0	0	0	0	0
4	0	0	0	0	0	0	1	1	1
5	1	0	0	1	0	0	0	0	0
6	1	1	1	0	1	0	0	0	0
7	0	0	1	1	0	0	0	0	0
8	0	1	1	0	0	0	0	0	0
9	1	1	1	1	0	1	0	0	0

Figura 3.2 - Exemplo do tipo de dados de entrada no algoritmo

O conjunto de operadores que fazem parte do algoritmo vão ser executados uma vez para cada uma das linhas (utilizadores) da matriz, através do operador ‘Loop Examples’. O índice de cada linha é referenciado pela variável `indice_linha`.

A figura 3.3 apresenta o conjunto de operadores que compõem o nível superior do algoritmo.

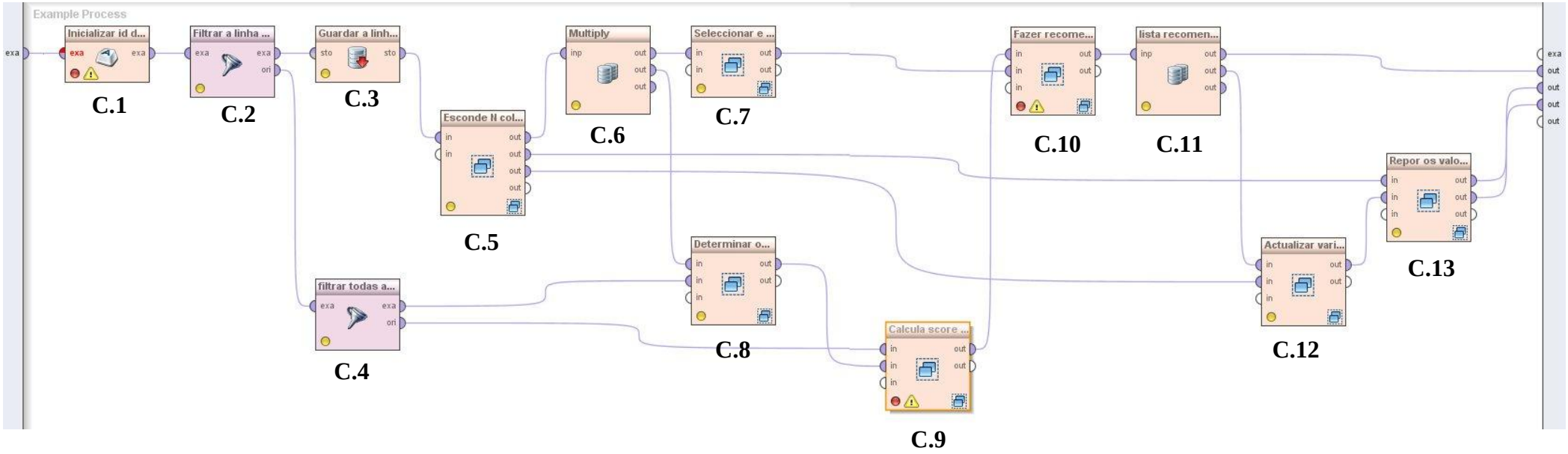

Figura 3.3 - Nível superior do algoritmo de FC

O algoritmo inicia-se com o operador 'Extract Macro' (C.1) que atribui a uma macro com o nome `valor_linha`, o valor da coluna 'id_user' no índice (utilizador) atual.

Os parâmetros usados por este operador (C.1) foram os seguintes:

macro	valor_linha
macro type	data_value
attribute name	id_user
example index	%{indice_linha}

3.2.1 - Selecionar os dois subconjuntos de utilizadores: atual e os restantes

Esta parte do algoritmo concretiza o ponto 1 da secção 2.6.1.
Assim, o passo C.2 permite selecionar apenas o utilizador atual (conjunto teste). Para isso, foi usado o operador 'Filter Examples' com os seguintes parâmetros:

condition class	atribute_value_filter
parameter string	id_user=%{valor_linha}

Os dados originais são passados para outro operador (C.4) idêntico ao C.2, mas, neste caso para selecionar os restantes utilizadores (conjunto treino).
No operador C.4 foram usados os seguintes parâmetros:

condition class	atribute_value_filter
parameter string	id_user=%{valor_linha}
invert filter	true

De notar, que estes parâmetros são quase idênticos aos do operador C.2. A única diferença é que a opção 'invert filter' está selecionada para que sejam selecionadas as linhas que não verificam a condição.

A linha do utilizador atual (seleccionado em C.2) irá ser guardada em memória através do operador 'Remember' (C.3) na variável `linhaSessao`. Esta operação é necessária

para que, posteriormente seja possível alterar nesta linha os itens que foram sorteados como escondidos (a descrever no subprocesso C.5).

3.2.2 - Esconder *n* itens do utilizador atual

Esta parte concretiza o ponto 2 da secção 2.6.1.

Esta etapa não faz parte do algoritmo de recomendação propriamente dito. Mas, pela sua função não pode ser isolada do algoritmo.

O passo C.5 permite esconder aleatoriamente *n* itens do utilizador atual, passando as células de 1 para 0. O valor de *n* é o que está definido no parâmetro `numero_itens_escondidos`.

A figura 3.4 apresenta o conjunto de operadores necessários para esconder alguns dos itens na linha do utilizador atual:

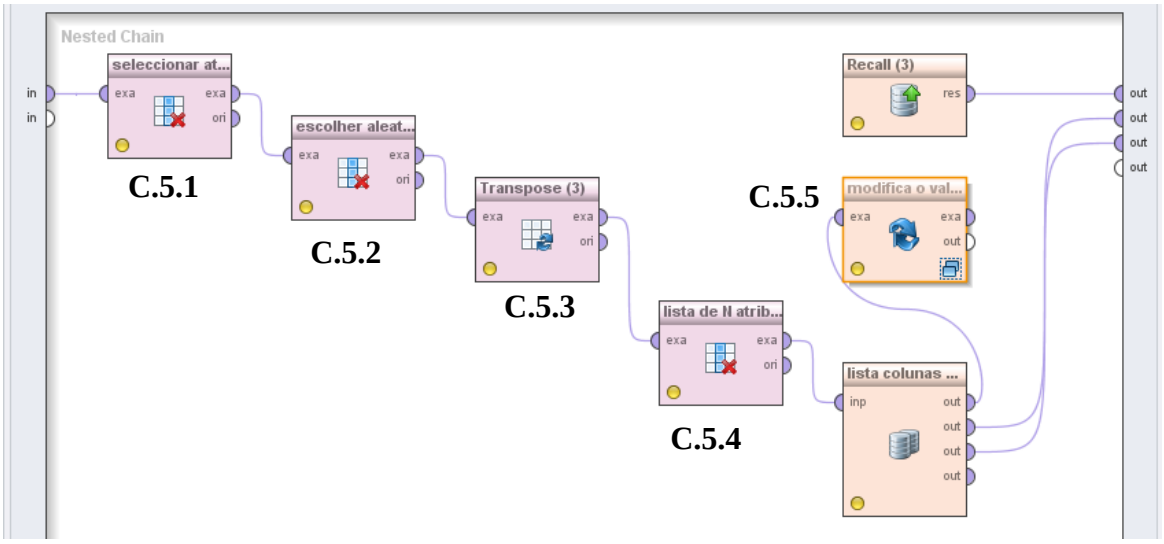

Figura 3.4 - Subprocesso que permite esconder itens do utilizador atual

Este subprocesso tem como *input* a linha que corresponde ao utilizador atual, que originou do passo C.3.

Em C.5.1, este subprocesso começa por seleccionar da linha actual apenas os atributos (itens) que tenham valor = 1.

É usada a seguinte condição:

attribute filter type	numeric_value_filter
numeric condition	= 1

Dos itens selecionados anteriormente, o operador 'Select by Random' (C.5.2) escolhe aleatoriamente alguns para esconder. O número de itens a esconder é o que está definido no parâmetro `numero_atributos_escondidos`.
Seguidamente, os passos C.5.3 e C.5.4 permitem que, a linha que contêm os nomes dos itens escondidos seja transposta de forma a que esses nomes fiquem dispostos (cada um numa linha) e numa só coluna (com o nome 'id') para facilitar o passo posterior.

Um possível resultado do passo C.5.4 é apresentado na figura 3.5:

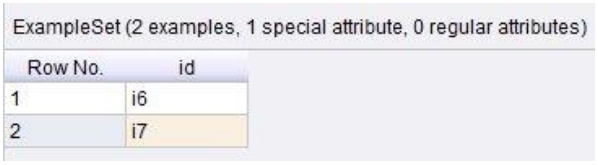

Figura 3.5 - Exemplo da estrutura de dados resultante da seleção dos itens a esconder

Em C.5.5, surge novamente o operador 'Loop Examples'. Têm como objetivo percorrer cada uma das linhas que contêm os nomes dos itens escondidos (ver figura 3.6) para modificar os valores da linha do utilizador atual entretanto guardada numa macro no passo C.3.

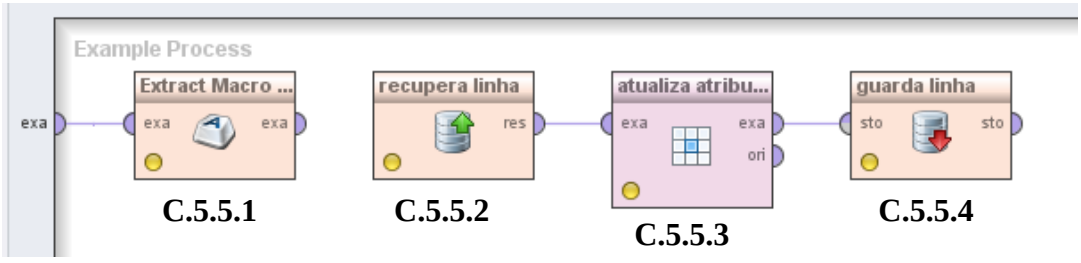

Figura 3.6 - Operador 'Loop examples' para percorrer cada uma das linhas e modificar o conjunto de dados original.

Este operador começa por percorrer cada uma dessas linhas, atribui a uma macro (de nome 'id_coluna') o valor da linha atual (que corresponde ao nome de um dos itens escondidos). Em C.5.5.2, a linha actual é recuperada pelo operador 'Recall' através da variável `linhaSessao`, que foi guardada pela primeira vez no passo C.3.

Em C.5.5.3 o operador 'Set Data' é o que permite esconder os atributos. Isto é, passar do valor 1 para 0. Este operador usa os seguintes parâmetros:

example_index	1
attribute_name	%{id_coluna}
value	0

Depois de cada alteração, cada linha vai ser actualizada em memória pelo operador 'Remember' (C.5.5.4) para voltar a ser recuperada na próxima iteração, e assim sucessivamente para todos os itens escondidos.

3.2.3 - Determinar os *k* vizinhos mais próximos

O subprocesso indicado no passo C.8 permite calcular os *k* vizinhos mais próximos entre o utilizador atual e os restantes.

Recebe dois *inputs*: linha com os itens observados (depois de alguns serem escondidos) e o outro são as restantes linhas que resultaram do passo C.4.

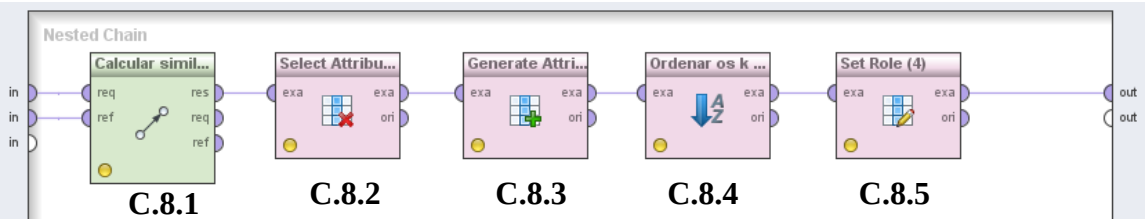

Figura 3.7 - Subprocesso que permite determinar os *k* vizinhos mais próximos de um utilizador

O operador 'Cross Distances' (C.8.1) calcula as distâncias entre o utilizador atual e os restantes. Do vasto leque de métricas de distância disponíveis, é usada a similaridade do cosseno.

Como se pretende um número restrito de utilizadores com maior similaridade, é necessário selecionar a opção ‘only top k’ e definir o referido número. Neste caso, corresponde ao valor do parâmetro `numero_vizinhos_mais_proximos`.

A figura 3.8 apresenta um possível *output* do operador ‘Cross Distances’ (C.8.1).

Row No.	request	document	distance
1	1	4	0.40825
2	1	2	0.40825
3	1	6	0.35355

Figura 3.8 - Possível *output* do operador ‘Cross Distances’

A coluna ‘request’ corresponde ao identificador do utilizador atual, ‘document’ aos identificadores dos outros utilizadores de comparação e ‘distance’ corresponde à distância propriamente dita.

Os resultados estão num formato que não é o pretendido. Por isso, é necessário fazer um tratamento especial para mudar o formato. O operador seguinte ‘Select Attributes’ (C.8.2) permite selecionar apenas os atributos ‘document’ e ‘distance’.

Seguidamente, o operador ‘Generate Attributes’ (C.8.3) permite gerar novos atributos, baseado nos dois já existentes. Assim, são gerados dois novos atributos e descartados os existentes. O novo atributo ‘id_user’ vai ter o valor do atributo ‘document’ e ‘similaridade’ o de ‘distance’.

Os parâmetros usados por este operador são os seguintes:

Atribute_name	**Function expression**
id_user	document
similaridade	distance

Após a transformação anterior, os dados ficam com um aspeto semelhante ao da figura 3.9.

Row No.	id_user	similaridade
1	4	0.40825
2	2	0.40825
3	6	0.35355

Figura 3.9 - Aspeto dos dados após a renomeação dos atributos

Em C.8.4, segue-se o operador 'Sort' que permite ordenar as linhas através da coluna 'similaridade' por ordem decrescente do seu valor.
O operador 'Set Role' (C.8.5) permite atribuir uma *role*[10] a um ou mais atributos. Neste caso, interessa que a coluna 'id_user' tenha a *role* id:

3.2.4 - Calcular o *score* dos itens

O subprocesso C.9 tem como objetivo calcular o *score* dos itens tendo em conta os *k* vizinhos mais próximos calculados anteriormente no subprocesso C.8.

O subprocesso tem o seguinte aspeto gráfico:

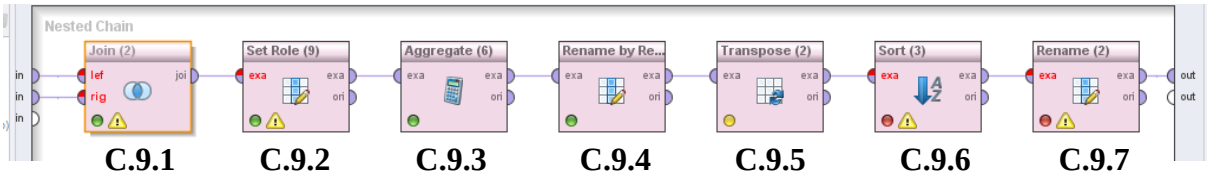

Figura 3.10 - Subprocesso que permite calcular o *score* dos itens

Este subprocesso recebe dois *inputs*: os dados originais e o conjunto das *k* linhas com os atributos: 'id_user' e 'similaridade' que resultaram do passo C.8.
Em C.9.1 é usado o operador 'Join' que tem como objetivo juntar vários conjuntos de dados num só, tendo em comum um atributo com *role* id. Neste caso, é o atributo 'id_user' que permite fazer a união dos dois conjuntos.

Um exemplo de um *output* resultante do passo C.9.1 está representado na figura 3.11:

Row No.	id_user	i1	i2	i3	i4	i5	i6	i7	i8	i9	similaridade
1	2	0	1	0	0	0	1	1	0	0	0.40825
2	4	0	0	0	0	0	0	1	1	1	0.40825
3	6	1	1	1	0	1	0	0	0	0	0.35355

Figura 3.11 - Exemplo de um *output* resultante do operador 'Join'

[10] É um estatuto especial que um atributo pode ter. Existem vários tipos: 'weight' que indica o peso da linha para possíveis cálculos; 'id' indica que o atributo identifica univocamente as linhas, etc.

Esta operação tem como objetivo criar uma estrutura de dados adequada para o cálculo dos *scores* de todos os itens.

O passo C.9.2 permite atribuir à coluna 'similaridade' a *role weight*, como sendo o atributo que representa o "peso" da linha para o cálculo dos *scores* dos itens.

Em C.9.3, o operador 'Aggregate' permite calcular os *scores* dos itens tendo em conta o peso de cada linha (passo C.9.2) e os valores de cada atributo.

São usadas as seguintes opções:

use default aggregation	true
atribute filter type	all
default aggregation function	sum

Um possível *output* desta operação está apresentado na figura 3.12:

Row No.	sum(i1)	sum(i2)	sum(i3)	sum(i4)	sum(i5)	sum(i6)	sum(i7)	sum(i8)	sum(i9)
1	0.35355	0.76180	0.35355	0	0.35355	0.40825	0.81650	0.40825	0.40825

Figura 3.12 - Exemplo de um *output* após determinação do *scores* de cada item

O passo seguinte (C.9.4), usa o operador 'Rename by Replacing' para renomear os atributos para o nome inicial com base numa expressão regular.

Nas expressões regulares do RM, todo o texto que se encontra entre os caracteres especiais '(' e ')' são chamados de grupos. Uma expressão regular pode ter vários grupos. Automaticamente, o texto dentro de cada um desses grupos é armazenados nas variáveis $1, $2, etc., respetivamente.

Os parâmetros usados por este operador são os seguintes:

replace what	sum\((.*)\)
replace by	$1

Após a renomeação, os nomes dos atributos ficam com o seguinte aspeto:

Row No.	i1	i2	i3	i4	i5	i6	i7	i8	i9
1	0.35355	0.76180	0.35355	0	0.35355	0.40825	0.81650	0.40825	0.40825

Figura 3.13 - Exemplo de um *output* após a renomeação das colunas

Em C.9.5 é feita a transposta da linha anterior para que os nomes e os *scores* dos itens fiquem dispostos em várias linhas tal como representado na figura 3.14:

Row No.	id	att_1
1	i1	0.35355
2	i2	0.76180
3	i3	0.35355
4	i4	0
5	i5	0.35355
6	i6	0.40825
7	i7	0.81650
8	i8	0.40825
9	i9	0.40825

Figura 3.14 - Exemplo de um *output* após a conversão dos nomes e *scores* dos itens em linhas

O passo C.9.6 permite ordenar descendentemente as linhas pelo atributo ‘att_1’.

O último passo (C.9.7) deste subprocesso, renomeia a coluna ‘att_1’ para um nome mais apropriado: ‘*score*’, através do operador ‘Rename’.

3.2.5 - Converter os itens observados numa coluna

O subprocesso C.7 tem como objetivo transformar os itens observados numa coluna para facilitar futuras operações sobre os dados.

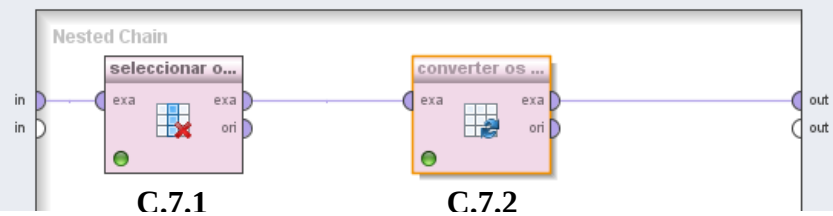

Figura 3.15 – Subprocesso que permite transformar os itens observados numa coluna

Recebe como *input* a linha do utilizador atual, proveniente de C.5 (depois de os valores dos itens escondidos já terem sido transformados em 0).

O passo C.7.1, através do operador ‘Select Attributes’ permite selecionar apenas os itens observados (os atributos em que o valor = 1). Da linha resultante de C.7.1 é feita a transposta (C.7.2) através do operador ‘Transpose’.

3.2.6 - Fazer recomendações de itens

O subprocesso C.10 representado na figura 3.16 permite fazer as recomendações de itens.

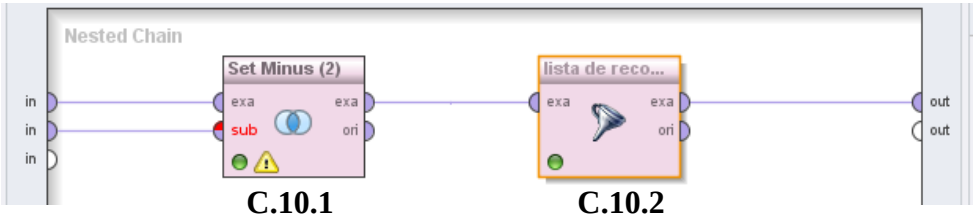

Figura 3.16 - Subprocesso que tem como objetivo fazer as recomendações de itens

Recebe como *input* dois parâmetros: o resultado obtido do passo C.9 (lista de todos os itens ordenados de forma decrescente do seu *score*) e os itens observados na linha do utilizador atual (resultante do passo C.7).

Assim, no passo C.10.1, o operador 'Set Minus' permite fazer a subtração entre o primeiro e o segundo parâmetro indicado anteriormente. Ou seja, da lista de todos os itens serão retirados os observados de modo a restarem apenas os que podem ser recomendados.

Seguidamente em C.10.2 é usado o operador 'Filter Example Range' para selecionar os itens com maior *score*. Os itens resultantes deste operador são os que vão ser recomendados ao utilizador atual. De relembrar que o número de itens a recomendar está definido no parâmetro `numero_recomendacoes`.

Tal como a etapa descrita na secção 3.2.2, as duas seguintes (secções 3.2.7 e 3.2.8) não fazem parte do algoritmo de recomendação propriamente dito. Mas, pelas suas funções não poderão ser isoladas deste algoritmo.

3.2.7 - Atualizar as variáveis de avaliação

Este subprocesso (C.12) permite atualizar as variáveis de avaliação que vão ajudar a obter os resultados finais.

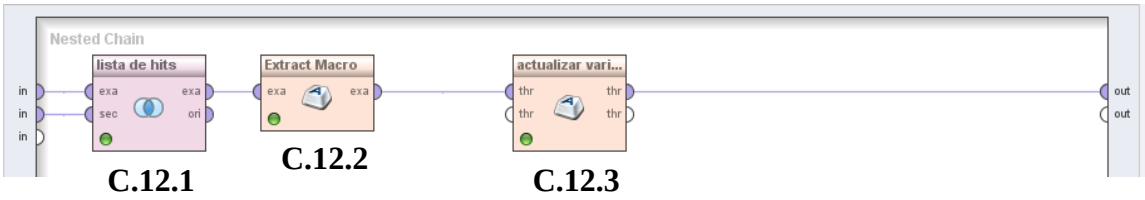

Figura 3.17 - Subprocesso que permite a atualização das variáveis de avaliação

Recebe dois parâmetros de *input*: a lista de itens recomendados (proveniente de C.11) e a lista de itens escondidos (proveniente de C.5).

O operador 'Intersect' (C.12.1) faz a intersecção dos conjuntos anteriores. Permite determinar quantos são os itens comuns entre os recomendados e os escondidos e consequentemente saber quantos foram os acertos (*hits*).

O passo seguinte (C.12.2) usa o operador 'Extract Macro' para atribuir a uma macro (com o nome `numero_hits`) o referido número de acertos.

De seguida, em C.12.3, é usado o operador 'Generate Macro' que vai atualizar variáveis usadas na avaliação para a iteração corrente:

total_hits	%{total_hits} + %{numero_hits}
number_users	%{number_users} + 1

Assim, a variável `%{total_hits}` acumula o número de acertos de cada utilizador e a variável `%{number_users}` contabiliza o número de utilizadores.

3.2.8 - Repor os valores originais, após as alterações feitas à linha do utilizador atual

O RM não possui nenhum operador específico para clonar dados (Rapid Miner – fórum oficial, 2012). Devido à inexistência de tal funcionalidade, há a necessidade de forçar a reposição dos valores da linha do utilizador atual (depois de alterados no passo C.5) para a forma que estava no passo C.3.

Assim, a matriz mantêm-se sempre com os mesmos valores que tinha no início do algoritmo para não afetar os cálculos dos utilizadores seguintes.

Como já foi referido anteriormente, o RM tem como base a linguagem Java. As cópias e as alterações aos dados são feitas sempre por referência e não por valor. Por isso, qualquer alteração que seja feita na linha do utilizador atual reflete-se na matriz original.

A operação realizada em C.13 é muito semelhante à C.5.5. A única diferença é que os valores dos itens que foram escondidos são alterados de 0 para 1.

3.3 - Implementação da metodologia experimental

Nesta secção vai ser descrita a metodologia experimental referida na secção 3.2.

São usados os seguintes parâmetros:

- `numero_minimo_itens_por_coluna`: Frequência mínima que cada item deve ter.
  Além de permitir reduzir a dimensão da matriz (no número de colunas), um valor elevado dá maior relevância a itens mais frequentes (que pertençam a mais utilizadores). Se não se pretende eliminar itens deve ser colocado o valor 1;
- `numero_minimo_itens_por_linha`: Número mínimo de itens observados que cada utilizador deve conter para a determinação das recomendações.
  Um valor elevado permite excluir utilizadores com poucos itens e dar maior relevância a aqueles com mais itens observados. O valor mínimo é 1.

O processo é constituído por quatro módulos:

- A - Leitura dos dados;
- B - Preparação dos dados;
- C - Aplicação do algoritmo de FC a cada utilizador;
- D - Geração dos resultados finais;

O aspeto gráfico do processo no nível mais alto é o seguinte:

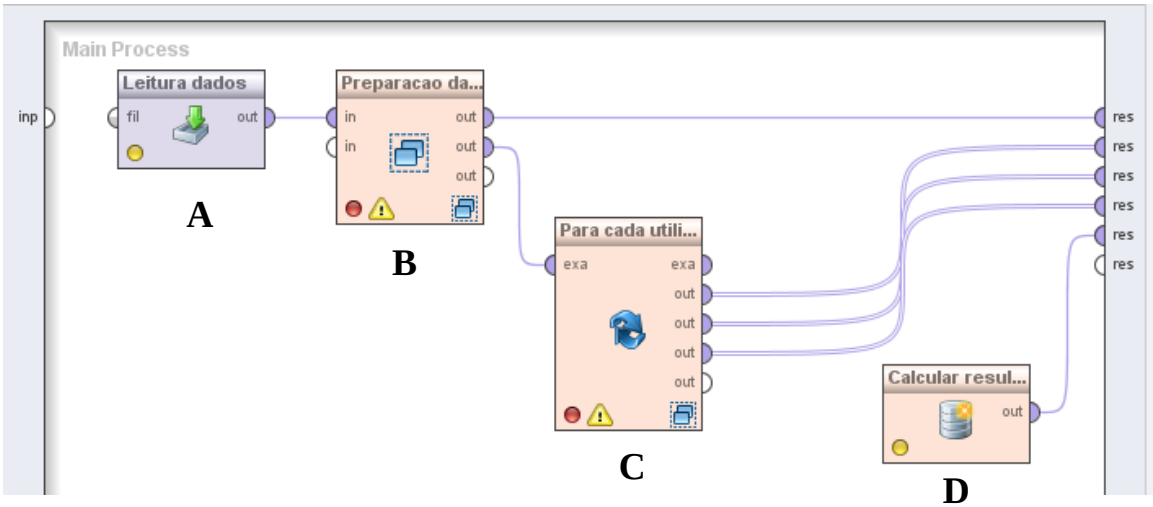

Figura 3.18 – Nível mais alto do processo

O algoritmo de recomendação (representado na figura 3.18 por C) já foi descrito na secção 3.2.

3.3.1 - Leitura dos dados

A leitura dos dados é feita através do operador 'Read CSV'. Neste operador, é indicado o caracter separador entre atributos e o caminho físico do ficheiro de dados. Os dados são automaticamente transformados numa estrutura para que possam ser trabalhados pelo RM.

O ficheiro de dados de *input* tem de estar no formato CSV[11] (*comma-separated values*) e ter obrigatoriamente pelo menos duas colunas com os seguintes nomes: 'id_user' e 'id_item' que representam respetivamente os identificadores dos utilizadores e dos itens.

A figura 3.19 apresenta um exemplo de um possível ficheiro de dados que serve de *input* ao operador 'Read CSV':

[11] É um formato de dados delimitado que possui campos (colunas) separados por vírgula e registos (linhas) separados por quebra de linha.

```
id_user,id_item
1,4
1,5
1,7
2,2
2,6
2,7
```

Figura 3.19 – Exemplo parcial da estrutura dos dados originais

3.3.2 - Preparação dos dados

Depois da leitura, segue-se a fase de preparação e limpeza dos dados.

Esta fase permite a realização das seguintes tarefas:

- Eliminação de registos duplicados;
- Eliminação de registos com valores em falta;
- Eliminação de utilizadores e de itens pouco frequentes;
- Conversão dos dados para uma matriz binária;
- Inicialização de algumas variáveis que vão auxiliar no processo de avaliação.

A figura 3.20 apresenta o aspeto do subprocesso de preparação dos dados:

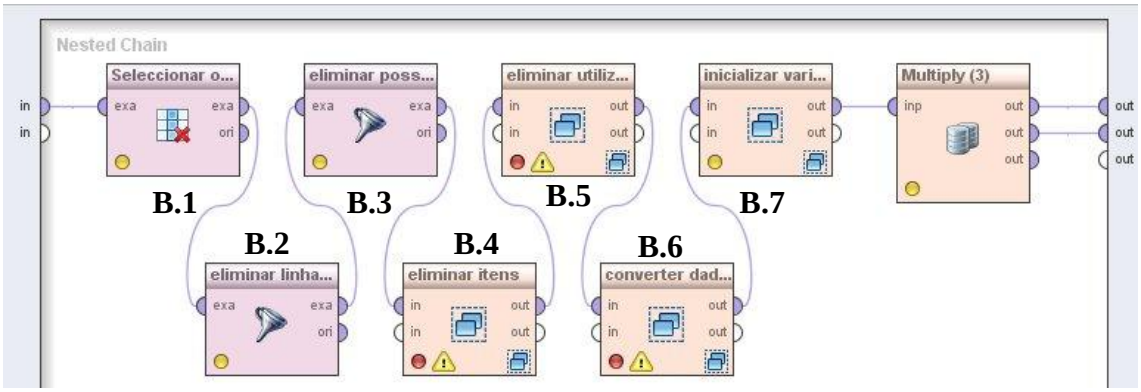

Figura 3.20 - Subprocesso de preparação dos dados

Esta fase inicia-se com a seleção dos dois atributos relevantes ao processo (‘id_user’ e ‘id_item’), através do operador ‘Select Attributes’ (B.1). Os outros atributos são descartados.

Row No.	id_user	id_item
1	1	4
2	1	5
3	1	7
4	2	2
5	2	6
6	2	7

Figura 3.21 – Exemplo do *output* parcial após a seleção dos dois atributos relevantes.

Em B.2 segue-se a filtragem dos registos que não tenham valores em falta. Pode acontecer que nos dados de *input* apareçam registos em que pelo menos uma das colunas tem um valor em falta (desconhecido).

O operador 'Filter Examples' tem a seguinte opção para realizar tal propósito:

condition class	no_missing_attributes

O passo seguinte (B.3) tem como objetivo eliminar linhas com registos duplicados. O operador 'Remove Duplicates' realiza essa operação.

Após este passo, a estrutura de dados fica com um aspeto semelhante ao da figura 3.21, mas eventualmente com menos registos.

O subprocesso indicado em B.4 permite eliminar os itens menos frequentes. Para grandes volumes de dados ou para dados demasiado esparsos, pode fazer sentido eliminar itens que estejam associados a poucos utilizadores. Este efeito reduz a complexidade computacional e o tempo dispensado para a obtenção de resultados porque são necessários menos cálculos e operações.

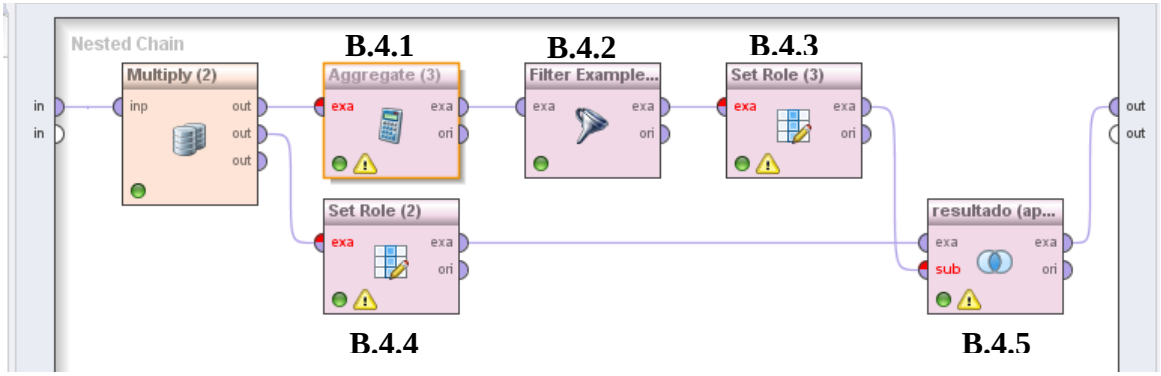

Figura 3.22 - Subprocesso que permite eliminar colunas

A forma mais simples e rápida de eliminar itens que pertençam no máximo a um determinado número de utilizadores é começar por fazer a contagem do número de utilizadores que cada item contêm. O operador 'Aggregate' (B.4.1) realiza tal tarefa, indicando qual o atributo pelo qual os dados são agrupados (neste caso, é pelo 'id_item') e a expressão de agregação (neste caso 'count').
Ou seja, para cada 'id_item' (ver figura 3.21) será contabilizado o número de vezes que este se repete nos dados originais.

A figura 3.23 apresenta um possível *output* obtido após esta operação:

Row No.	id_item	count(id_item)
1	1	3
2	2	4
3	3	5
4	4	5
5	5	2
6	6	2
7	7	3
8	8	1
9	9	1

Figura 3.23 - Exemplo de um *output* após a agregação dos dados pelo atributo 'id_item'

Em B.4.2, usando o operador 'Filter Examples', processa-se à filtragem dos registos cuja frequência absoluta é inferior ao parâmetro `numero_minimo_itens_por_coluna`. De notar que a opção 'invert_filter' está ativa:

condition class	atribute_value_filter
parameter string	count(id_item) >= %{numero_minimo_itens_por_coluna}
invert filter	true

Segue-se em B.4.3 o operador 'Set Role' para definir o atributo 'id_item' como *role* id, requisito necessário para o passo B.4.5. O mesmo acontece em B.4.4, mas para os dados de entrada neste subprocesso.
Os passos B.4.3 e B.4.4 são necessários para ser possível usar a subtração de exemplos (é requisito obrigatório que os dois exemplos tenham cada um, um atributo com a *role* id) (B.4.5). Assim, o operador 'Set Minus' permite realizar a subtração entre o conjunto

de dados de entrada neste subprocesso e os dados resultantes do passo B.4.2 (conjunto de itens menos frequentes).

Em B.5 é realizada uma operação semelhante à B.4 mas, neste caso agregado pela coluna 'id_user' (representado na figura 3.21).

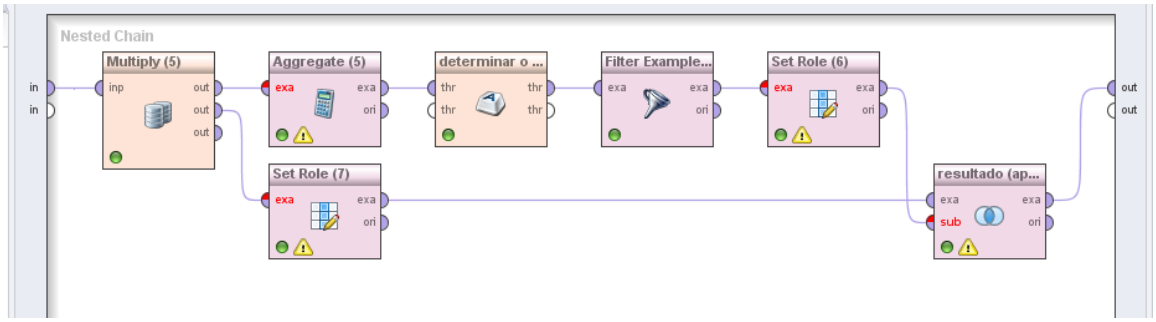

Figura 3.24 - Subprocesso que permite eliminar linhas

Uma das poucas diferenças é o uso do operador 'Extract Macro' para determinar o número mínimo de itens que cada linha deve ter. Este valor resulta da soma de dois parâmetros: `numero_minimo_itens_por_linha` e `numero_itens_escondidos`.

numero_minimo_itens	%{numero_minimo_itens_por_linha} + %{numero_itens_escondidos}

É a macro com o nome `numero_minimo_itens` que é usada na filtragem das linhas.
Em suma, esta operação permite eliminar ocorrências de 'id_user' com poucos itens.

Segue-se em B.6 um dos passos mais importantes deste processo: a conversão dos registos numa matriz binária de utilizadores *vs* itens.

Este subprocesso tem o seguinte aspeto:

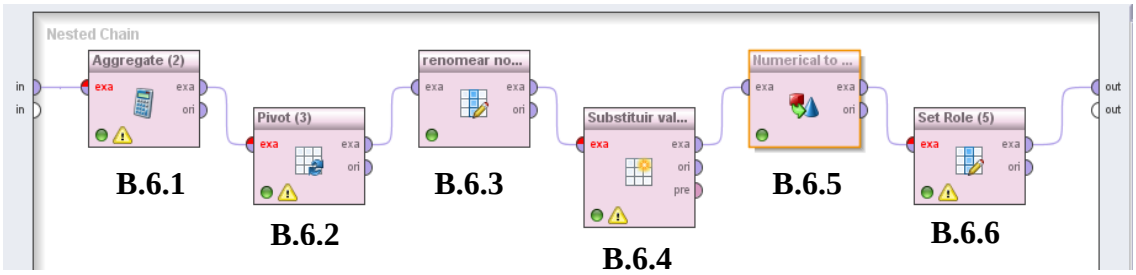

Figura 3.25 - Subprocesso que faz a conversão dos dados para uma matriz utilizadores *vs* itens

Em B.6.1 é usado o operador 'Aggregate' que tem como função contar o número de ocorrências de cada par 'id_user' 'id_item'. Como anteriormente foram eliminados os registos duplicados, então cada par terá como frequência o valor 1.

O passo B.6.2 transforma os dados numa matriz com ajuda do operador 'Pivot'. Como o próprio nome indica, este operador transforma um conjunto de exemplos noutro conjunto por agrupamento de colunas.
Neste caso, foram definidas as seguintes opções neste operador:

group atribute	id_user
index attribute	id_item
weight aggregation	sum
consider weighs	true

Desta forma, o conjunto de exemplos resultante terá como primeira coluna 'id_user' e as seguintes serão os códigos dos vários itens.

A figura 3.26 apresenta um exemplo parcial de um possível *output* resultante do passo anterior:

Row No.	id_user	count(id_item)_1.0	count(id_item)_2.0	count(id_item)_3.0
1	1	?	?	?
2	2	?	1	?
3	3	?	?	1
4	4	?	?	?
5	5	1	?	?
6	6	1	1	1
7	7	?	?	1
8	8	?	1	1
9	9	1	1	1

Figura 3.26 - Exemplo de um *output* após a transformação do conjunto de dados numa matriz

A matriz resultante (figura 3.26) contêm células com o valor '1' caso a relação utilizador/item exista e o caracter '?' caso contrário.
Com ajuda do operador 'Rename by Replacing', o passo B.6.3 permite renomear (com ajuda de uma expressão regular) as colunas para um nome mais apropriado: i1, i2,…, i_n

attribute filter type	all
include special attributes	true
replace what	count\(id_item\)_(.*).0
replace by	$1

O passo seguinte (B.6.4) permite substituir os valores em falta *(missing values*) na matriz. Com o operador 'Replace Missing Values', as células com o caracter '?' são substituídas por 0 (zero).

Se as colunas só contiverem valores numéricos, o RM assume por defeito o tipo de dados inteiro. Tendo em vista uma generalização para qualquer tipo de dados no código do utilizador (inteiros ou literais), o passo B.6.5, com a ajuda do operador 'Numerical to Polynominal' converte o atributo 'id_user' de inteiro para nominal. Este requisito é necessário porque algumas operações posteriores só funcionam se esta coluna for do tipo nominal.

As opções definidas no operador ‘Numerical to Polynominal’ são as seguintes:

atribute filter type	single
attribute	id_user

Finalmente, o passo B.6.6 permite definir o atributo ‘id_user’ como *role* id.

name	id_user
target role	id

O subprocesso que se segue no passo B.7 permite inicializar algumas variáveis que vão ajudar a obter os resultados finais da avaliação.

É constituído apenas por dois operadores ‘Set Macro’ que inicializam a 0 (zero) as macros: `total_hits` e `numero_linhas`.

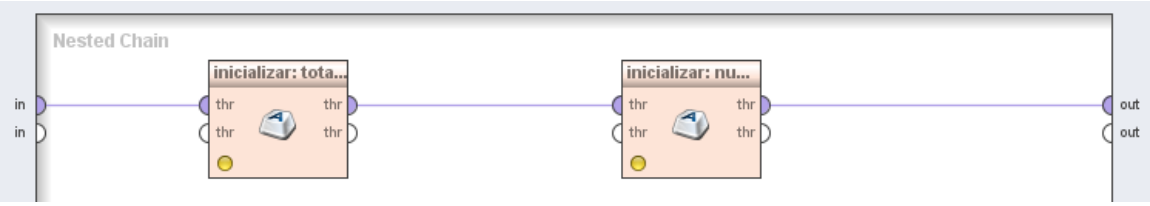

Figura 3.27 - Aspeto gráfico do subprocesso B.7

3.3.3 – Determinar os resultados finais

Nesta fase é preparado o *output* com as medidas de avaliação da performance do algoritmo.

Com o operador ‘Generate Data by User Specification’ é gerado um registo cujos valores dos atributos são definidos por expressões.

As medidas de avaliação usadas são: *precision*, *recall* e F1 (descritas na secção 2.6.2).

A tabela 3.1 apresenta os atributos resultantes deste passo (incluindo as métricas de avaliação e outras variáveis informativas):

Atributo	**Fórmula**
precision	%{total_hits} / (%{numero_recomendacoes} * %{number_users})
recall	%{total_hits} / (%{number_users} * %{numero_itens_escondidos})
F1	(2 * (%{total_hits} / (%{number_users} * %{numero_itens_escondidos})) * (%{total_hits} / (%{numero_recomendacoes} * %{number_users}))) / ((%{total_hits} / (%{number_users} * %{numero_itens_escondidos})) + (%{total_hits} / (%{numero_recomendacoes} * %{ number_users})))
total_hits	%{total_hits}
number_users	%{number_users}

Tabela 3.1 – Fórmulas usadas para o cálculo das medidas de avaliação e outras variáveis informativas

A figura 3.28 apresenta um exemplo obtido de um *output* com as métricas de avaliação (*precision*, *recall* e F1):

ExampleSet (1 example, 0 special attributes, 5 regular attributes)

Row No.	precision	recall	F1	total_hits	number_users
1	0.15556	0.77778	0.25926	7	9

Figura 3.28 - Exemplo de um *output* final.

4 - Resultados experimentais

Este capítulo corresponde à componente empírica desta dissertação. Permite validar o processo de RM implementado.

É iniciado com a descrição dos dados usados nas experiências. Segue-se uma breve análise exploratória dos mesmos e a descrição das condições experimentais. Por fim, a apresentação e discussão dos resultados obtidos.

4.1 – Descrição dos dados

O conjunto de dados provém de uma recolha recente obtida durante um determinado período de tempo na rede social Palco Principal (PP).

Como já foi referido na secção 2.7, o PP é uma rede social de música. Oferece diferentes funcionalidades aos utilizadores registados: divulgar trabalhos musicais, ouvir músicas, criar as suas próprias *playlists*, adicionar músicas a essas *playlists* de acordo com as suas preferências, descartar músicas que lhe são recomendadas pelo sistema, adicionar explicitamente sugestões de músicas, entre outras (Palco Principal – site oficial, 2012).

Pretende-se usar o algoritmo desenvolvido e descrito na secção 3.3 para recomendar músicas a *playlists* (utilizadores) e avaliar a sua performance na recomendação.

Os dados originais foram fornecidos no formato CSV. São constituídos por duas colunas (atributos): ‘id_playlist’ e ‘id_music’.

De uma forma resumida, os dados representam um conjunto de *playlists* e as músicas que os utilizadores lhes adicionaram.

A figura 4.1 apresenta a estrutura dos dados do ficheiro original.

```
id_playlist,id_music
1,82
2,1932
1,39
1,1089
3,1377
7,4,2128
```

Figura 4.1 - Pequeno extrato do ficheiro CSV que contêm os dados originais

Os dois atributos indicam respetivamente, os códigos das *playlists* e das músicas.
Tal como referido na secção 3.3.1, para o processo funcionar, os dados originais tem de ter pelo menos dois atributos com os seguintes nomes: 'id_user' e 'id_item'. Assim, o atributos 'id_playlist' e 'id_music' são renomeados à *priori* para 'id_user' e 'id_music', respetivamente.

4.2 – Análise exploratória dos dados

Os dados iniciais são constituídos por 95081 registos, que se caracterizam por 9711 *playlists* (utilizadores) e 15804 músicas (itens). Segundo Chen e Yin (2006), para avaliar o nível de esparsidade da matriz correspondente (ver secção 3.2), usa-se normalmente o fator ψ que é definido como sendo:

$$\psi = 1 - \frac{\{r \in R \mid r \neq 0\}}{|R|} \quad (4.1)$$

em que r representa um valor da matriz R (que neste caso, pode tomar valores 0 ou 1) e $|R|$ a respetiva dimensão. O numerador da equação (eq. 4.1) é o número de células cujo valor é diferente de zero e o denominador a dimensão da matriz (produto do número de linhas pelo número de colunas).
Neste caso, $\psi = 1 - \frac{95081}{9711*15804}$ que resulta aproximadamente no valor 0.9994. Pode-se considerar um valor muitíssimo alto. Significa que cerca de 99.94% das células da matriz estão preenchidas com valor 0.

1 - Relação do número de *playlists* associadas a cada música:
Das 15804 músicas, 5645 pertencem a apenas uma única *playlist*, 2617 a duas, 1543 a três e 1164 a quatro. Estes valores indicam que cerca de 69% das músicas fazem parte de no máximo quatro *playlists* o que demonstra que a maioria das músicas são pouco solicitadas pelos utilizadores.
A música mais requisitada faz parte de 325 *playlists*. Em média, cada música pertence a aproximadamente seis *playlists.*

2 - Relação do número de músicas por *playlist*:

Do total das 9711 *playlists*, 2749 são constituídas por apenas uma música (que corresponde a cerca de 28.3%), 1105 por duas e 877 três. As restantes *playlists* têm mais do que três músicas.

A *playlist* mais preenchida contêm 2237 músicas e o número médio de músicas por *playlist* é de aproximadamente: $\frac{95081}{9711} \cong 9.79$. A mediana é de quatro músicas, ou seja 50% do total das *playlists* têm no máximo quatro músicas.

Estes valores indiciam uma distribuição muito assimétrica da distribuição do número de músicas por *playlist*.

4.3 – Descrição das condições experimentais

Uma elevada dimensão da matriz (principalmente no número de atributos – músicas) tem um impacto negativo na execução do processo de RM (implementação).

Nestas condições, o tempo de processamento é muitíssimo elevado e muitas vezes não é concluído devido a problemas de falta de memória, mesmo seguindo os procedimentos sugeridos por fontes oficiais do RM para resolver este tipo de situações. O RM é muito sensível à dimensão da matriz, principalmente ao número de atributos.

Para contornar este problema houve a necessidade de fazer um tratamento prévio ao conjunto de dados iniciais (antes da eliminação dos menos frequentes). Assim, os dados foram repartidos aleatoriamente em três conjuntos disjuntos tendo como única regra: terem aproximadamente igual número de *playlists*.

A tabela 4.1 apresenta a constituição de cada um dos três conjuntos:

Conjunto	Nº *playlists*	Nº músicas	Nº total registos
1	3237	10836	32654
2	3237	9554	31717
3	3237	9520	30710
TOTAL	**9711**	*****	**95081**

Tabela 4.1 – Constituição de cada um dos três conjuntos após a divisão dos dados iniciais

Nas experiências, foram tidos em conta os seguintes pressupostos:

- O número mínimo de *playlists* que uma música tem de pertencer são cinco;
  Este critério tem como objetivo eliminar atributos (músicas) que menor influencia terão na decisão de recomendação pelo facto de pertencerem a poucas *playlists*;
- O número de músicas escondidas por *playlist* são sempre duas.

Assim sendo, temos os seguintes parâmetros fixos usados nas experiências:

Numero mínimo *playlists* por música	5
Número de músicas escondidas por *playlist*	2

As várias experiências foram baseadas na combinação dos seguintes parâmetros:

- Número de recomendações: 3, 5 e 10;
- Número de vizinhos mais próximos: 3, 5 e 7;
- Número de itens observáveis: 1, 3 e 5.

Para cada combinação foram realizadas 15 experiências (cinco para cada um dos três conjuntos) e obtido o valor médio de F1 (medida que vai ser usada na avaliação da performance do algoritmo de recomendação). O que varia entre cada uma das cinco experiências de cada conjunto são os itens escondidos.

Por exemplo, usando 3 recomendações, 7 vizinhos mais próximos e 1 item observado, os resultados obtidos foram os seguintes:

Exp.	Conjunto 1			Conjunto 2			Conjunto 3			
	Prec.	Recall	F1	Prec.	Recall	F1	Prec.	Recall	F1	
1	0,33100	0,49600	**0,39700**	0,31370	0,47055	**0,37644**	0,31400	0,47100	**0,37700**	
2	0,33233	0,49849	**0,39879**	0,30982	0,46473	**0,37178**	0,30779	0,46168	**0,36935**	
3	0,32680	0,49020	**0,39216**	0,31539	0,47309	**0,37847**	0,31324	0,46987	**0,37589**	
4	0,32152	0,48228	**0,38582**	0,30982	0,46473	**0,37178**	0,31126	0,46689	**0,37351**	
5	0,32604	0,48906	**0,39125**	0,31248	0,46873	**0,37498**	0,31052	0,46577	**0,37262**	
			0,39300			**0,37469**			**0,37367**	**0,380**

F1 médio do conjunto 1 — F1 médio do conjunto 2 — F1 médio do conjunto 3 — F1 médio dos 3 subconjuntos

Tabela 4.2 – Resultados obtidos para uma das combinações de parâmetros (3 recomendações, 7 vizinhos mais próximos e 1 item observado)

4.4 – Apresentação e discussão dos resultados

Os valores de F1 obtidos usando as várias combinações de parâmetros referidos anteriormente foram os seguintes:

Nº itens observados	3 recomendações			5 recomendações			10 recomendações		
	3 viz.	5 viz.	7 viz.	3 viz.	5 viz.	7 viz.	3 viz.	5 viz.	7 viz.
1	0,358	0,374	0,380	0,283	0,295	0,301	0,182	0,190	0,196
3	0,374	0,386	0,388	0,291	0,303	0,307	0,190	0,196	0,200
5	0,377	0,384	0,388	0,294	0,302	0,305	0,190	0,196	0,198

Tabela 4.3 – Resultados obtidos (F1) tendo em conta o número de itens observados, de vizinhos mais próximos e de recomendações

Pela análise dos resultados podem-se tirar as seguintes conclusões:

- Em todos os cenários, para o mesmo número de itens observados e de recomendações, o aumento do número de vizinhos mais próximos melhora sempre (embora ligeiramente) o valor de F1.

  Ao incluir mais vizinhos, as recomendações são baseadas em mais exemplos (mais evidência).

  Quando os vizinhos têm preferências semelhantes às do utilizador que se pretende recomendar, normalmente origina uma melhoria dos resultados até um certo número de vizinhos. A partir desse número, irão existir cada vez mais vizinhos com diferentes preferências, afetando os resultados (piores recomendações);
- Não é possível estabelecer uma relação entre o número de itens observados e as restantes variáveis. De um até três itens há sempre uma ligeira melhoria de F1 em todos os cenários. De três a cinco itens os resultados variam de cenário para cenário;
- Para o mesmo número de vizinhos mais próximos e de itens observados, os valores de F1 pioram sempre à medida que o número de recomendações aumenta.

  Ao fazer mais recomendações, é esperado que o *recall* aumente (são selecionados mais itens relevantes) e que a *precision* diminua (são selecionados mais itens não relevantes). A redução de F1 é causada pela *precision* baixar muito mais do que o aumento do *recall*. Isto pode dever-se ao facto de a matriz ser muito esparsa e pelo número médio de músicas por *playlist* ser baixo. Também pode indicar que não há grupos muito homogéneos de listas (isto é, com grande sobreposição nos itens selecionados).

No gráfico da figura 4.2, são analisadas isoladamente as componentes: *precision* e *recall* para um cenário com 3 vizinhos mais próximos; 1 item observado; 3, 5 e 10 recomendações:

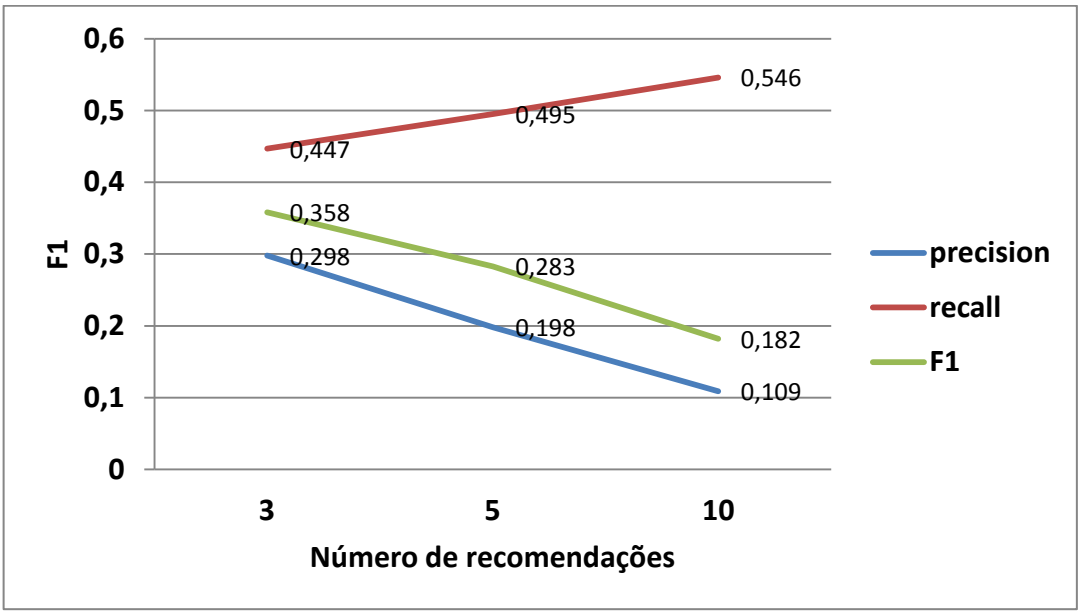

Figura 4.2 - Variação de *precision*, *recall* e F1 em função do número de recomendações

A figura 4.2 confirma que o valor de F1 diminui à medida que o número de recomendações aumenta.

Tal como foi previsto anteriormente, a parcela responsável pela diminuição de F1 é a *precision* (a azul na figura 4.2) porque diminui significativamente de 0.298 (3 recomendações) para 0.109 (10 recomendações). O *recall* (a vermelho na figura 4.2) pelo contrário, aumenta mas com menor intensidade (de 0.447 até 0.546).

Com a realização destas experiências, verifica-se que a implementação funciona, gera recomendações e os mecanismos de avaliação funcionam corretamente.

5 - Conclusões

Hoje em dia, os Sistemas de Recomendação (SR) têm um papel fundamental tanto para o utilizador comum que pesquisa um item como para o gestor de negócio que pretende apostar em novas formas de rentabilizar o seu negócio.
A utilidade principal destes sistemas é a capacidade de prever classificações ou gostos a um utilizador do sistema baseando-se no histórico de preferências de outros utilizadores.

Na área dos SR existem essencialmente três abordagens principais: filtragem baseada no conteúdo (FBC), filtragem colaborativa (FC) e híbrida. Destas, a FC é a mais utilizada hoje em dia em grandes companhias a nível mundial como por exemplo, a Amazon, eBay, Netflix, etc.
A performance de um SR pode ser avaliada usando várias métricas: *precision*, *recall* e F1 são das mais usadas pelo menos se os dados forem binários.

Apesar da importância destes métodos, o Rapid Miner (RM) não tem de base algoritmos nem operadores específicos para SR.
O RM é uma ferramenta de *Data Mining* (DM) e é utilizada em muitas empresas a nível mundial. A sua versão gratuita (*community*) já permite realizar projetos interessantes. É uma aplicação baseada numa sequência de operadores em que cada um tem uma função específica.

O objetivo principal desta tese consistiu em avaliar as capacidades do RM na implementação de um algoritmo de FC usando somente operadores de base do RM.
O processo desenvolvido tem uma estrutura modular bem definida. A leitura dos dados (de um ficheiro CSV externo) corresponde à primeira fase. Segue-se a preparação dos dados onde podem ser realizadas diversas operações sobre dados (eliminação de utilizadores e de itens menos frequentes, conversão dos dados originais para uma matriz binária de utilizadores *vs* itens, etc.). Posteriormente, o algoritmo de recomendação propriamente dito, onde todo o mecanismo de recomendação se desenrola. Por fim, a

geração dos resultados finais, que inclui os valores das métricas de avaliação e outras informações auxiliares (número de utilizadores e número de *hits*).

A implementação funciona, gera recomendações e os mecanismos de avaliação apresentam corretamente os resultados.

De uma forma geral, o processo desenvolvido apresenta as seguintes características:

- Bem estruturado, organizado e fácil de compreender;
- Configurável para dados de qualquer dimensão (quer em número de utilizadores, quer de itens);
- Tem vários parâmetros configuráveis: número de recomendações, número de vizinhos mais próximos, número de itens escondidos, número mínimo de itens por coluna e número mínimo de itens observados por linha;
- Conversão do conjunto de dados no formato CSV para uma matriz binária de utilizadores *vs* itens;
- Implementa a componente de avaliação do SR usando as métricas mais apropriadas para dados binários (*precision*, *recall* e F1);
- Adaptado a dados de qualquer contexto de negócio (neste caso, *playlists vs* músicas, como poderia ser clientes *vs* produtos, etc.);

Na minha opinião, posso considerar como alguns dos pontos fortes do RM:

- Facilidade de utilização;
- Grande variedade de operadores existentes para executar os diversos passos de construção de um modelo;
- Grande capacidade de comunicação com outras aplicações externas;
- Rapidez e utilidade das respostas a dúvidas colocadas no fórum oficial[1];
- Disponibilidade de alguns exemplos de pequenos processos publicados no site 'MyExperiment'.

[1] http://forum.rapid-i.com

Alguns dos pontos negativos:

- Falta de alguns operadores que facilitem a manipulação de matrizes para realizar certas tarefas básicas;
- É muito sensível à dimensão da matriz, principalmente no número de atributos. Este problema torna-se mais preocupante quando o conjunto de dados é grande, que é normal num SR.

A realização desta tese permitiu-me:

- Compreender melhor a importância dos SR nos dias de hoje;
- Conhecer com um maior detalhe os casos reais de uso destes sistemas inteligentes de recomendação;
- Perceber os mecanismos de funcionamento de um SR, bem como as suas vantagens/desvantagens e utilização;
- Aprender a trabalhar com o RM: perceber as suas capacidades, funcionalidades e limitações;
- Compreender as métricas e possíveis metodologias de avaliação dos SR;

A implementação realizada pode ser útil para diversos fins. Um professor do Canadá viu nela valor pedagógico (uma forma de explicar o algoritmo aos alunos) e até solicitou autorização para utilizar para esse fim.

O objetivo principal foi atingido: a implementação de um SR com base nos operadores base do RM.

5.1 - Trabalho futuro

Como trabalho futuro seria interessante comparar os resultados obtidos deste processo com os obtidos usando a extensão recentemente criada pelo projeto *e-lico*.

6 - Referências

Adomavicius, G. e Tuzhilin, A. (2005), "*Toward the Next Generation of Recommender Systems: A Survey of State-of-the-Art and Possible Extensions*", IEEE Transactions on Knowledge and Data Engineering, v.17 n.6 p. 734-749

Anand, S. S. e Mobasher, B. (2005), "*Intelligent Techniques for Web Personalization*", http://maya.cs.depaul.edu/mobasher/papers/am-itwp-springer05.pdf (acedido a 21 Jul. 2012)

Batista, P. (2010), "*Data Mining* na identificação de atributos valorativos da habitação", Universidade de Aveiro, http://www2.mat.ua.pt/gladys/AlunosTeses/tese_PauloBatista.pdf (acedido a 5 Ago. 2012)

Breese, J., Herkerman, D. e Kadie, C. (1998), "*Empirical Analysis of Predictive Algorithms for Collaborative Filtering*", Proceedings of the Fourteenth Conference on Uncertaintly in Artificial Intelligence, EUA http://www.cs.washington.edu/education/courses/csep546/07sp/proj2/algsweb.pdf (acedido a 24 Nov. 2011)

Cazela, S., Nunes, M. e Reategui, E. (2010), "A Ciência da Opinião: Estado da arte em sistemas de recomendação" http://www.dcomp.ufs.br/~gutanunes/hp/publications/JAI4.pdf (acedido a 21 Jul. 2012)

Chen, J. e Yin, J. (2006), "*Recommendation based on Influence Sets*"; http://webmining.spd.louisville.edu/webkdd06/papers/paper-12-Recommendation%20Based%20on%20Influence%20Sets-WM_1054%5B1%5D.pdf (acedido a 19 Nov. 2011)

Gunawardana, A. e Shani, G. (2009), "*A Survey of Accuracy Evaluation Metrics of Recommendation Tasks*", Journal of Machine Learning Research; http://research.microsoft.com/pubs/118124/gunawardana09a.pdf (acedido a 1 Mar. 2012)

Hashsler, M. (2011), "*Developing and Testing Top-N Recomendation Algorithms for 0-1 Data using recommenderlab*"

Jorge, Alípio; Domingues, Marcos e Soares, Carlos (2011), "*Dimensions as Virtual Items: Improving the Predictive Ability of Top-N Recommender Systems*", INESC-Porto

Karypis, G. (2000), "*Evaluation of the Item-Based Top-N Recommendation Algorithms*", Department of Computer Science and Engineering, University of Minnesota

Kim, Y. S., Yum, B., Song, J. e Kim, S. M. (2005), "*Development of a recommender system based on navigational and behavioral patterns of customers in e-commerce sites*", Journal Expert Systems with Applications; http://nclab.kaist.ac.kr/papers/Journal/Development_of_a_recommender_system.pdf (acedido a 19 Jul. 2012)

McCarey, F., Cinneide, M. e Kushmerick, N. (2003), "*A Case Study on Recommending Reusable Software Components using Collaborative Filtering*", Department of Computer Science, University College Dublin

Melville, P. e Sindhwani, V. (2010), "*Recommender Systems*", "*Encyclopedia of Machine Learning*"; http://www.prem-melville.com/publications/recommender-systems-eml2010.pdf (acedido em 22 Nov. 2011)

Mobasher, B. (2007), "*Recommender Systems*", Kunstliche Intelligenz, Special Issue on Web Mining, No. 3, PP. 41-43, 2007. BottcherIT Verlag, Bremen, Germany http://maya.cs.depaul.edu/~mobasher/papers/mobasher-ki2007.pdf (acedido a 22 Set. 2012)

Nichols, D. M. (1997), "*Implicit Rating and Filtering*", Computing Department, Lancaster University, Lancaster, UK http://www.comp.lancs.ac.uk/computing/research/cseg/projects/ariadne/docs/delos5.pdf (acedido a 1 Nov. 2011)

Oliveira, L. G. (2007), "*Proposal of the methodological structure to implement recommender systems*" http://www.intelog.net/ArtigosNoticias/Arquivos/artigo_leonardo_oliveira.pdf (acedido a 15 Out. 2011)

Palco Principal, Blog (2010), http://palco3.palcoprincipal.org/?cat=21 (acedido em 20 Out. 2011)

Palco principal, Site oficial (2012), http://palcoprincipal.sapo.pt/main/sistemaSugestao (acedido a 11 Mai. 2012)

Pereira, D. (2007), "Uma aplicação em Sistemas de Recomendação: Sistema de Recomendação para pacotes GNU/Linux", Universidade Federal do Rio Grande do Sul; http://www.inf.ufrgs.br/bdi/administrator/components/com_jresearch/files/publications/Monografia-DiegoPereira.pdf (acedido a 25 Ago. 2012)

Ramos, J. (2010), "Algoritmos Colaborativos para Sistemas de Recomendação"; Faculdade de Engenharia da Universidade do Porto (FEUP)

Rapid Miner, brochura oficial, http://rapid-i.com/content/view/102/118/lang,en (acedido a 14 Jul. 2012)

Rapid Miner, fórum oficial (2012), http://forum.rapid-i.com

Rapid-I GmbH (2010), "Manual oficial do Rapid Miner 5.0", http://sourceforge.net/projects/rapidminer/files/1.%20RapidMiner/5.0/rapidminer-5.0-manual-english_v1.0.pdf/download; (acedido a 14 Jul. 2012)
Segaran, T. (2007), "*Programming Collective Intelligence*", O'Reilly

Soboroff, I. (1999), "*Combining Content and Collaboration in Text Filtering*", Department of Computer Science and Electrical Engineering; University of Maryland, Baltimore County
http://www.csee.umbc.edu/csee/research/cadip/1999Symposium/mlif.pdf (acedido a 9 Jul. 2012)

Valdemir (2008), "Sistema de Recomendação para venda de computadores na *Web*"; http://pt.scribd.com/doc/64869945/TCC2-Valdemir-16122008 (acedido a 9 Nov. 2011)

Venâncio, N. e Nóbrega, S. (2011), "Sistemas de Recomendação em Redes Sociais", Ciclo de Seminários Técnicos 2011

Vozalis, E. e Margaritis, K. (2003), "*Analysis of Recommender Systems Algorithms*" In *Proceedings of the Sixth Hellenic-European Conference on Computer Mathematics and its Applications* - HERCMA 2003.
http://users.uom.gr/~mans/papiria/hercma2003.pdf (acedido a 15 Out. 2011)

Sistemas de Recomendação em Rapid Miner:

um caso de estudo

Sérgio Morais

2012