Departamento de Ciências e Tecnologias de Informação

Recomendação de música: comparação entre Collaborative Filtering e Context Filtering

Vicente Canhoto

Dissertação submetida como requisito parcial para obtenção do grau de

Mestre em Gestão de Sistemas de Informação

Orientador:

Doutor Carlos Serrão, Professor Auxiliar, ISCTE-IUL

Co-Orientadora:

Doutora Elsa Cardoso, Professora Auxiliar, ISCTE-IUL

Outubro, 2013

Resumo

A massificação de serviços de música *online* democratizou o acesso a milhões de músicas. No entanto, é impossível para os utilizadores ouvirem e conhecerem todas essas músicas. De modo a auxiliar na sugestão sobre o que ouvir num dado momento, foram desenvolvidos sistemas que recomendam músicas ao utilizador. A técnica de *Collaborative Filtering* gera recomendações com base nas músicas ouvidas por utilizadores com gostos semelhantes. Apesar de apresentar um bom desempenho, vários investigadores propuseram melhoramentos aos mesmos. Um dos mais referidos é a utilização de informação contextual sobre o utilizador. A relação entre a utilização desta informação em sistemas de recomendação e o aumento da satisfação dos utilizadores foi provada por diversos investigadores.

O trabalho desenvolvido nesta dissertação focou-se na comparação entre um algoritmo de recomendação por *Collaborative Filtering* tradicional e outro baseado em determinados elementos do contexto. Para isso foi proposto e implementado um sistema de recomendação *online* que integra estas duas abordagens, apoiado numa revisão da literatura.

Por fim, este sistema foi utilizado numa experiência de campo *online* em que qualquer utilizador pôde fazer pedidos de recomendação. Estes pedidos foram servidos alternadamente por cada um dos algoritmos de recomendação, e foram registadas as avaliações dos utilizadores às músicas recomendadas de modo a aferir a sua satisfação com ambas as abordagens. Os resultados obtidos demonstram que a recomendação baseada no contexto foi superior ao *Collaborative Filtering*, exceção apenas para a fase inicial do funcionamento do sistema em que existiam poucos dados acerca das interações dos utilizadores com as músicas disponibilizadas.

Palavras chave: sistema de recomendação, recomendação de música, *collaborative filtering*, recomendação baseada no contexto, *context pre-filtering*, comparação de sistemas de recomendação.

Abstract

The massification of online music services democratized the access to millions of songs. Nevertheless, it is impossible for the users to enjoy and know all those songs. In order to assist in the suggestion about what to listen in a given moment, there have been developed systems which recommend music to the user. The well-known *Collaborative Filtering* technique generates recommendations based on the interests of users with similar tastes. Despite presenting a good performance, several investigators proposed improvements to it. One of the most mentioned is the use of contextual information about the user. The relationship between the use of this information in recommendation systems and increased user satisfaction has been proven by several investigators.

The work developed in this thesis was focused on the comparison between a *Collaborative Filtering* recommendation algorithm and a *Context-based* one. In order to achieve that, an online recommendation system that integrates these two approaches was proposed and implemented, supported by a literature review.

Finally, this system was used in an online study in which any user could make music recommendation requests. These requests were served alternately by each one of the implemented algorithms and the user's ratings to the recommendations were recorded in order to assess their satisfaction with both approaches. The results showed that the *Context-based* recommendation approach was superior to the *Collaborative Filtering* algorithm, except only for the initial phase of the system operation where there was little music ratings data.

Keywords: recommendation system, music recommendation, collaborative filtering, context-based recommendation, context pre-filtering, recommendation systems comparison.

Agradecimentos

Embora o resultado do trabalho desenvolvido no último ano, e que é espelhado nesta dissertação, seja fruto do meu empenho e dedicação, a sua conclusão não seria possível sem o apoio prestado por um conjunto importante de pessoas.

Agradeço ao Professor Carlos Serrão, pela sua orientação ao longo de todo o trabalho e pela sua disponibilidade para me esclarecer dúvidas, sugerir caminhos alternativos e criticar opções menos acertadas.

À Professora Elsa Cardoso, agradeço o interesse demonstrado no meu trabalho e o seu *feedback* em relação ao seu desenvolvimento e às opções tomadas.

À minha família, agradeço o apoio e motivação a vários níveis ao longo de todo o meu percurso académico. Estiveram sempre disponíveis, tanto nos sucessos como nos momentos de stress.

Aos meus amigos, pela compreensão em relação aos momentos de convívio a que faltei e pela motivação que me deram nos momentos em que estive presente.

À Ana Pereira, por ter partilhado comigo os altos e baixos ao longo desta dissertação, pelo apoio e motivação sempre presentes.

A todos o meu muito obrigado.

Índice

Índice de Figuras

Índice de Tabelas

1. Introdução

Este capítulo apresenta um enquadramento geral para esta dissertação e a Motivação para o seu desenvolvimento. São também definidos o Problema a que se pretende responder, a Hipótese de resposta a esse problema, os Objectivos que se pretendem atingir e a Metodologia seguida.

1.1. Enquadramento

Nos dias de hoje, os utilizadores de serviços na *web* são bombardeados com imensa informação potencialmente relevante. De facto, a massificação da Internet e de dispositivos digitais em todo o Mundo possibilita não só a criação de conteúdos com grande facilidade mas também a divulgação e disseminação dos mesmos. No entanto, embora uma vasta oferta de conteúdos tenha os seus benefícios, é impraticável para um utilizador consumir todos esses recursos dados os constrangimentos temporais óbvios. Coloca-se então o problema da sobrecarga de informação e seleção dos conteúdos de maior interesse para o utilizador. Devido à emergência deste problema, a recomendação de informação tornou-se uma importante área de investigação nomeadamente a partir dos anos 90 (Shardanand 1994). Devido ao aumento de utilizações práticas desde então, continua a ser uma área bastante ativa. Alguns exemplos vão desde a personalização de lojas online para cada utilizador de acordo com os seus artigos de interesse (Linden, Smith, York 2003), passando pela recomendação de programas de televisão com interesse para o utilizador (Lucas 2010) e até recomendação de aplicações para dispositivos móveis (Sousa 2012).

Outra das aplicações práticas desta tecnologia é a recomendação de música. À medida que a Internet se torna no meio privilegiado de distribuição de música digital, uma maior quantidade de música (comercial e não comercial) torna-se acessível para os utilizadores. Nesta situação, são úteis sistemas que recomendem música para cada pessoa de forma personalizada, porque se torna difícil e demorado procurar música nova de interesse para o utilizador. Para dificultar esta tarefa, apenas uma pequena percentagem de artistas tem destaque nos meios de comunicação, *web* sites e serviços de música online.

Apesar do tema da recomendação ter sido já bastante estudado, um dos problemas atuais é a adequação da recomendação não apenas para a pessoa mas tendo igualmente em consideração a situação e o contexto em que a pessoa se encontra aquando da recomendação.

Por exemplo, na recomendação de viagens faz sentido incorporar o contexto temporal de modo a distinguir ofertas de viagens de verão e viagens de inverno. Ou na recomendação de restaurantes, em que pode ser útil ter sugestões diferenciadas caso a pessoa queira jantar sozinha ou acompanhada. Deste modo, a experiência de audição de música pode ser igualmente influenciada por um conjunto de factores da envolvente do utilizador, pelo que não devem ser ignorados caso se queiram obter recomendações o mais eficazes possível e com um elevado nível de satisfação.

1.2. Motivação

Atualmente o acesso a música digital está muito facilitado. A quantidade de serviços de *streaming* de música online continua a crescer, sendo que muitos deles apresentam funcionalidades semelhantes. Uma das características predominantes destes serviços é a recomendação de música, sendo que um dos métodos mais eficazes e mais utilizados é o *Collaborative Filtering* (Ricci, Rokach, Shapira 2011; Su, Khoshgoftaar 2009; Linden, Smith, York 2003; Bernhardsson 2009). Embora a adopção deste método proporcione à partida uma capacidade comprovada de recomendação de música, as recomendações feitas aos utilizadores serão muito semelhantes entre os vários serviços que utilizem este método. Isto acontece pois geralmente os dados de partida utilizados no cálculo das recomendações são semelhantes: as músicas são provenientes das mesmas editoras, através de acordos ou licenciamento, e de fontes públicas de música gratuita; os *ratings* atribuídos às músicas pelos utilizadores terão tendência para se aproximar de um valor igual nos vários serviços, assumindo que a música tenha sido "avaliada" por um número de utilizadores considerável.

A dificuldade de diferenciação destes serviços justifica e motiva a melhoria deste método de recomendação. Através de uma análise da literatura foi possível identificar um conjunto de alterações ao algoritmo de *Collaborative Filtering* tradicional que resultaram em aumentos de eficácia e de satisfação dos utilizadores (Su, Yeh, Yu, Tseng 2010; Lee, Lee 2007; Ono, Kurokawa, Motomura, Asoh 2007).

Estas alterações consistem na introdução no algoritmo de elementos que identifiquem o contexto e a situação em que o utilizador se encontra (localização geográfica e estado do tempo, por exemplo), no momento em que pretende que lhe seja recomendado um determinado conteúdo.

Deste modo, a recomendação é adequada não só à pessoa e aos seus gostos mas também à influência que a sua envolvente tem nos seus padrões de consumo de informação e de conteúdos.

Uma das hipóteses para a recolha de dados sobre a envolvente do utilizador é a introdução explícita por parte do mesmo. Esta tarefa pode revelar-se desmotivadora e, por conseguinte, resultar numa barreira para a aquisição dos dados necessários para o algoritmo que se pretende implementar. No entanto, a massificação de *smartphones* e de outros dispositivos móveis com acesso à internet e com sensores inteligentes (p. ex. receptor de GPS) possibilita a recolha de dados em plano de fundo (background) acerca do contexto do utilizador.

1.3. Definição do Problema

Como foi referido anteriormente, a utilização de elementos de contexto trouxe ganhos de satisfação do utilizador em diversos domínios de recomendação. Através de diversos estudos (Su, Yeh, Yu, Tseng 2010; Lee, Lee 2007) foi identificado um conjunto de categorias de elementos de contexto que consistentemente demonstram ter influência no consumo e recomendação de música, nomeadamente elementos meteorológicos e temporais. No entanto, estes estudos utilizam algoritmos de recomendação muito diferentes dos algoritmos de *Collaborative Filtering* tradicionais, não sendo possível aferir o impacto direto dos referidos elementos de contexto nestes algoritmos já implementados em larga escala.

Este problema assume especial relevância nos dias que correm, em que o consumo de música através da *web* recorrendo a serviços de *streaming* tem cada vez maior peso na indústria. A crescente emergência de novos serviços de música com funcionalidades de recomendação (frequentemente utilizando o método de *Collaborative Filtering*) dificulta a sua diferenciação.

Assim, a questão central a que esta dissertação pretende responder é:

"Será que a utilização dos elementos de contexto *hora, dia da semana* e *estado do tempo* num sistema de recomendação de música por *Collaborative Filtering* melhora a satisfação dos utilizadores em relação a um sistema de *Collaborative Filtering* tradicional?"

1.4. Hipótese

A hipótese que esta dissertação pretende verificar é a seguinte:

"A utilização dos elementos de contexto *hora, dia da semana* e *estado do tempo* num sistema de recomendação de música por *Collaborative Filtering* melhora a satisfação dos utilizadores relativamente a um sistema de *Collaborative Filtering* tradicional".

1.5. Objectivos

Os objectivos desta dissertação são os seguintes:

- Comparação da eficácia (em termos de satisfação do utilizador) de recomendações de música entre um sistema de *Collaborative Filtering* tradicional e um de *Collaborative Filtering* baseado no estado do tempo, hora do dia e dia da semana (sistema *Context-based).*
- Desenvolver um sistema de recomendação de música *online* baseado no estado do tempo, hora do dia e dia da semana.

1.6 Metodologia

A questão de investigação levantada na secção anterior caracteriza-se pela sua natureza prática. Com vista a serem alcançadas soluções rigorosas foi adoptada a metodologia *Design Science Research* (Peffers, Tuunanen, Rothenberger, Chatterjee 2007), que visa resolver problemas práticos e teóricos com recurso a artefactos de TI destinados a resolver problemas organizacionais bem identificados (Peffers, Tuunanen, Rothenberger, Chatterjee 2007; Hevner, March, Park, Ram 2004; March, Smith 1995).

Assim, as fases descritas por esta metodologia foram aplicadas da seguinte forma:

- **Motivação e Identificação do Problema –** Na Motivação são descritos alguns problemas atuais dos serviços de música na *web* com recomendação (e mais especificamente do método de recomendação *Collaborative Filtering*) e é identificada uma oportunidade para o melhoramento do método de recomendação indicado. Estes argumentos servem de base para a definição do Problema que guia esta dissertação e da Hipótese que vai ser testada.

- **Definição de Objectivos para a Solução –** No capítulo 1.5 são inferidos os objectivos a partir da definição do problema e do conhecimento sobre o estado da arte. São analisados os diferentes métodos de recomendação, incluindo os seus problemas e limitações. O método *Collaborative Filtering* é descrito com maior detalhe, mencionando e comparando as técnicas matemáticas mais utilizadas. De seguida é apresentada a teoria que suporta a utilização de elementos de contexto em sistema de recomendação por *Collaborative Filtering*, como é feito em termos práticos e casos de utilização noutros domínios.

- **Desenho e Desenvolvimento –** Nos capítulos 3. e 4. é proposto o desenho do sistema a ser desenvolvido e explicado o seu desenvolvimento. São definidos os elementos de contexto a ser utilizados (e justificada a sua utilização, com casos de uso e referências de utilização), o modo como se integram com o método de recomendação *Collaborative Filtering* e é detalhada a implementação técnica do sistema.

- **Demonstração –** No capítulo 5. será demonstrado o sistema, na forma de um teste com utilizadores reais divididos em dois grupos. Um grupo de utilizadores utilizará o sistema *context-aware* desenvolvido nesta dissertação, enquanto o outro utilizará um sistema de recomendações *Collaborative Filtering* tradicional. Todos os utilizadores usarão o sistema para receberem e ouvirem recomendações de música, atribuindo um *rating* a cada recomendação de modo a poderem ser extraídas conclusões quanto à satisfação dos utilizadores e à eficácia do sistema.

Ambos os sistemas terão por base as mesmas músicas e os mesmos metadados, de modo a que as diferenças entre as recomendações as suas recomendações possam ser justificadas com a utilização dos elementos de contexto.

- **Avaliação –** Ainda no capítulo 5. será descrita a avaliação do sistema implementado, através de uma comparação da sua eficácia com um sistema de *Collaborative Filtering* tradicional. Serão comparadas as médias dos *ratings* atribuídos às recomendações dos sistemas pelos dois grupos e será extraída uma conclusão. Pretende-se saber essencialmente se a utilização de elementos de contexto na recomendação de músicas melhora a satisfação dos utilizadores comparativamente a sistemas de recomendação por *Collaborative Filtering* tradicionais.

  No processo de dedução de conclusões podem surgir propostas de alteração ou melhoramentos do sistema. Essas propostas serão indicadas como trabalho futuro.

2. Conceitos e Trabalho Relacionado

Neste capítulo é feita uma revisão da literatura, começando pelos conceitos que serão abordados e apresentando-se de seguida o trabalho relacionado já desenvolvido sobre o tópico que se pretende abordar.

2.1. Conceitos

Neste capítulo são expostos os principais conceitos teóricos e trabalho relacionado relativamente ao tópico dos sistemas de recomendação. São descritos os principais métodos de recomendação com base na literatura existente, dando maior ênfase aos métodos que se pretende estudar. Para além disso, é descrito o modo como estes sistemas são avaliados e como podem ser comparados.

Os sistemas de recomendação são técnicas e ferramentas de *software* que fornecem sugestões que possam ter utilidade para um utilizador. As sugestões referem-se a vários processos de tomada de decisão, como que livro comprar, que música ouvir ou que hotel escolher. Os sistemas de recomendação são principalmente direccionados para pessoas que não têm experiência pessoal suficiente ou competência para avaliar o potencialmente esmagador número de *items* que, por exemplo, um *website* tem para oferecer e seleccionar os que mais lhes convêm. Um caso recorrente é um sistema de recomendação de música que assiste os utilizadores na seleção de música para ouvir. O popular site de música Last.fm[1] utiliza um sistema de recomendação para personalizar deste modo a experiência do serviço que disponibiliza a cada utilizador (Mahmood, Ricci 2009).

O termo *item* é geralmente usado para designar o que o sistema recomenda aos utilizadores. Habitualmente, um sistema de recomendação foca-se num tipo de *item* específico (p. ex. livros, faixas de música) e, consequentemente, o seu design, a sua interface gráfica e a técnica utilizada para gerar recomendações são personalizadas para fornecer sugestões úteis para o tipo específico de *item*.

Na sua forma mais simples, as recomendações personalizadas são fornecidas como listas ordenadas de *items*. Para construir este *ranking*, o sistema tenta prever quais os produtos ou serviços mais adequados baseado nas preferências e restrições do utilizador.

[1] http://www.lastfm.pt/

De forma a completar esta tarefa computacional, os sistemas de recomendação recolhem as preferências dos utilizadores, que são expressas explicitamente (p.ex. através do *rating* atribuído a um produto) ou são inferidas implicitamente através das acções do utilizador. Por exemplo, o sistema pode interpretar a permanência demorada num determinado artigo noticioso como um sinal implícito de preferência pelo tipo de notícia em questão (Oard, Kim 1998).

Na sua formulação mais comum, o problema da recomendação é reduzido ao problema de estimação de *ratings* para *items* que o utilizador ainda não viu/avaliou. Esta estimação é normalmente baseada nos *ratings* atribuídos pelo utilizador a outros *items* e *ratings* atribuídos a este *item* por outros utilizadores (Adomavicius, Tuzhilin 2005).

Na sua forma geral, estes sistemas são designados de bi-dimensionais (2D) pois utilizam apenas as dimensões *Utilizador* e *Item* no processo de recomendação, e podem ser descritos como uma função em que o *input* são dados com as preferências do utilizador e que produz uma lista de recomendações como *output*. A figura 1 apresenta uma visão geral do processo de recomendação 2D tradicional composto por estes três componentes.

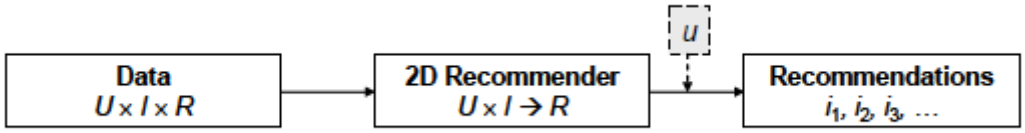

Figura 1- Componentes principais de um processo tradicional de recomendação (Adomavicius, Tuzhilin 2011)

Como indicado na figura, a função de recomendação é definida (ou construída) a partir dos dados disponíveis. Depois, a função é calculada para o Utilizador *u* e sobre todos os *items* candidatos para se obter um *rating* previsto para cada *item*. No fim, os *items* são ordenados pelo seu *rating* previsto e são recomendados os top-*N* (Adomavicius, Tuzhilin 2011).

2.2. Algoritmos de Recomendação

De acordo com a síntese efectuada por (Celma 2010), a classificação de sistemas de recomendação pode ser feita de acordo com as categorias que se apresentam de seguida.

2.2.1. Demographic Filtering

O *Demographic Filtering* pode ser utilizado para identificar que tipo de utilizadores gostam de um determinado item. Por exemplo, (Pazzani 1999) demonstra a influência que os dados demográficos têm na recomendação de restaurantes. A sua aplicação no domínio da música passa por saber, por exemplo, o estereótipo de utilizador que gosta de uma determinada banda ou artista.

Com esta técnica os perfis dos utilizadores são agrupados de acordo com determinados dados pessoais (idade, sexo, etc), dados geográficos (cidade, país) ou outros dados demográficos. Um dos primeiros exemplos de um sistema deste género foi o Grundy (Rich 1979), que recomendava livros baseando-se em dados pessoais recolhidos através de um diálogo interativo.

A natureza pouco personalizada deste método traduz-se em taxas de satisfação das recomendações relativamente baixas (quando usado exclusivamente), comparativamente com outros métodos de recomendação. O estudo de (Pazzani 1999) revelou que apenas 57.5% das recomendações de restaurantes baseadas em perfis demográficos foram do agrado dos utilizadores. No entanto, a utilização deste método supera a recomendação de items aleatoriamente quando não existem dados ou quando não é possível recorrer a outro método de recomendação. Outra possibilidade é a conjugação dos dados demográficos com outros métodos com vista ao aumento da precisão das recomendações.

2.2.1.1. Limitações

O principal problema deste método é a elevada generalização das recomendações, pois os mesmos *items* são recomendados a pessoas com dados demográficos semelhantes. Esse problema acentua-se caso os utilizadores forneçam poucos dados, levando a uma maior generalização dos perfis. No entanto, a massificação da utilização de redes sociais com API's abertas a outras aplicações podem minorar este problema sem grande esforço por parte do utilizador.

2.2.2. Content-Based Filtering

No método *content-based*, o sistema de recomendação recolhe informação descritiva dos *items* e depois, baseado num perfil do utilizador e nos *items* que manifestou interesse, prevê que *items* é que este poderá gostar usando uma função de semelhança. A informação de perfil pode ser pedida aos utilizadores explicitamente (p. ex. através de questionários) ou implicitamente (aprendendo os seus gostos à medida que este utiliza o sistema). Esta abordagem não depende dos *ratings* dados pelos utilizadores mas sim na descrição dos *items*.

Por exemplo, numa aplicação de recomendação de música o sistema iria compilar as preferências definidas no perfil do utilizador e as características das músicas que ouviu no passado. Depois, apenas as músicas que tivessem maior correspondência com essas características e preferências seriam recomendadas. No MusicSurfer (Cano, Koppenberger, Wack 2005), o sistema extrai descrições relacionadas com a instrumentação, ritmo e harmonia do sinal áudio de modo a providenciar recomendações.

O processo de caracterização dos *items* é feito geralmente de uma de duas formas:

- **Automático** – através de processos de extração de características e análise do conteúdo (p. ex. *audio fingerprinting*). Um exemplo de sistema utilizado para esta função é o jAudio (McKay, Fujinaga, Depalle 2005).
- **Anotações Manuais** - é um processo lento e complexo, em que especialistas em música e áudio ouvem cada música e categorizam-na de acordo com uma série de parâmetros definidos previamente (melodia, ritmo, harmonia, etc). O exemplo mais conhecido é o do sistema Pandora[2], em que cada música é avaliada segundo cerca de 400 parâmetros[3].

[2] http://www.pandora.com
[3] http://www.cs.washington.edu/education/courses/csep521/07wi/prj/michael.pdf

2.2.2.1. Limitações

Como todos os métodos de recomendação, a abordagem *conten-based* apresenta algumas desvantagens. O autor (Celma 2010) define os seguintes problemas em sistemas de recomendação de música *content-based*:

- **Problema do "arranque a frio"** – este problema ocorre quando o utilizador entra pela primeira vez no sistema. Caso não seja possível fazer um mapeamento entre as preferências especificadas pelo utilizador e as características dos itens (ou o sistema esteja desenhado para deduzir as preferências do utilizador através da interação do mesmo com os itens) as primeiras recomendações serão pouco eficazes.
- **Problema da "ovelha cinzenta"** – podem surgir utilizadores com gostos atípicos, pelo que um sistema com uma coleção de *items* pequena ou mais virados para determinados géneros poderão não conseguir fazer recomendações significativas nalguns casos.
- ***"Novelty Problem"* –** assumindo que a função de semelhança é precisa, os items recomendados serão sempre muito semelhantes aos que o utilizador já ouviu, dando pouco espaço para a descoberta de novas músicas por parte do utilizador ou para a criação de *playlists* ecléticas.
- **Dificuldade de extração de características –** dependendo da complexidade do domínio de *items*, pode haver dificuldade na extração automática de características de alto nível com significado para os utilizadores. No caso da música, as características mais facilmente analisadas são p. ex. a harmonia ou o ritmo, mas estes elementos raramente são especificados pelos utilizadores nas suas preferências. Um exemplo de característica de alto nível com significado poderia ser o tipo de emoção ou sentimento associado à música, mas os processos de extracção automática ainda não conseguem avaliar esse tipo de conteúdo das músicas.

2.2.3. Collaborative Filtering

O desenvolvimento de sistemas de recomendação partiu de uma simples observação: as pessoas confiam frequentemente nas recomendações fornecidas por outros em tomadas de decisão rotineiras. Por exemplo, é comum para um indivíduo confiar nas recomendações de familiares e amigos ao escolher um livro para ler, ou decidir se pretende assistir à próxima estreia do cinema ao ler a respectiva crítica escrita por um crítico de cinema com gostos semelhantes aos seus. Na tentativa de simular este comportamento, os primeiros sistemas de recomendação empregavam algoritmos que potenciavam as recomendações produzidas por uma comunidade de utilizadores para sugerir recomendações a um utilizador em particular. Estas recomendações partiam de *items* que utilizadores similares (com gostos semelhantes) tivessem gostado. Esta abordagem ficou denominada de *collaborative filtering* e a sua lógica é a de que se o utilizador que pretende recomendações (utilizador ativo) concordou no passado com outros utilizadores, então recomendações oriundas desses utilizadores semelhantes serão relevantes e de interesse para o utilizador ativo (Ricci, Rokach, Shapira 2011).

Concretamente, a abordagem *collaborative filtering* é utilizada para sugerir novos *items* a um determinado utilizador com base nos *ratings* que atribuiu a outros *items* e nos utilizadores com gostos parecidos. Ou seja, o utilizador atribui *feedback* ao sistema de modo a que este possa formular previsões acertadas baseadas no seu próprio *feedback* e de outros utilizadores. Este feedback pode ser dado explicitamente pelos utilizadores através de um *rating* ou *score* (geralmente dentro de uma determinada escala numérica), ou derivado implicitamente através da interação dos utilizadores com o sistema. No caso de um sistema de recomendação de música, essa interação poderia corresponder à quantidade de vezes que o utilizador ouve determinada música, que músicas faz "*skip*", entre outros.

Como foi referido, ideia principal por trás desta abordagem é a de que utilizadores semelhantes atribuem *feedback* semelhante aos mesmos *items* (Goldberg, Roeder, Gupta, Perkins 2001).

O funcionamento base dos métodos de *collaborative filtering* passa pela construção de uma matriz ***M*** com ***n*** *items* e ***m*** utilizadores, que contêm o *feedback* dos utilizadores dado aos *items*. Cada linha representa um utilizador, enquanto que as colunas representam os *items*.

A figura 2 retracta a matriz de *ratings* utilizador-item, em que o valor $\mathbf{M_{ua,ij}}$ corresponde ao *rating* atribuído pelo utilizador $\boldsymbol{u_a}$ ao *item* $\boldsymbol{i_j}$:

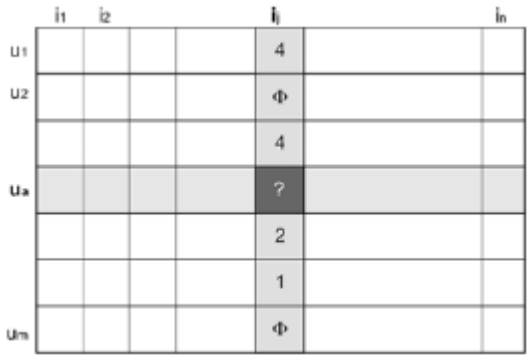

Figura 2 - Matriz utilizador-item (Celma 2010)

Um dos primeiros sistemas que implementaram este método foi o projeto *Tapestry* na Xerox PARC (Goldberg, Nichols, Oki, Terry 1992), projeto esse que cunhou a expressão "*collaborative filtering*". Outro dos primeiros sistemas a surgir inclui o GroupLens (Resnick, Iacovou, Suchak, Bergstrom, Riedl 1994), para avaliação e recomendação de artigos Usenet, e um sistema de recomendação de filmes desenvolvido por (Hill, Stead, Rosenstein, Furnas 1995). O primeiro sistema de recomendação na área da música foi o Ringo, da autoria de (Shardanand 1994).

Na tabela seguinte, (Su, Khoshgoftaar 2009) apresentam uma síntese das técnicas de *collaborative filtering* mais utilizadas:

CF categories	Representative techniques	Main advantages	Main shortcomings
Memory-based CF	*Neighbor-based CF (item-based/user-based CF algorithms with Pearson/vector cosine correlation) *Item-based/user-based top-*N* recommendations	*easy implementation *new data can be added easily and incrementally *need not consider the content of the items being recommended *scale well with co-rated items	*are dependent on human ratings *performance decrease when data are sparse *cannot recommend for new users and items *have limited scalability for large datasets
Model-based CF	*Bayesian belief nets CF *clustering CF *MDP-based CF *latent semantic CF *sparse factor analysis *CF using dimensionality reduction techniques, for example, *SVD*, *PCA*	*better address the sparsity, scalability and other problems *improve prediction performance *give an intuitive rationale for recommendations	*expensive model-building *have trade-off between prediction performance and scalability *lose useful information for dimensionality reduction techniques
Hybrid recommenders	*content-based CF recommender, for example, *Fab* *content-boosted CF *hybrid CF combining memory-based and model-based *CF* algorithms, for example, Personality Diagnosis	*overcome limitations of CF and content-based or other recommenders *improve prediction performance *overcome CF problems such as sparsity and gray sheep	*have increased complexity and expense for implementation *need external information that usually not available

Tabela 1 - Principais técnicas de recomendação Collaborative Filtering

De acordo com (Breese, Heckerman, Kadie 1998), os algoritmos de *collaborative filtering* podem ser agrupados em dois grupos: *Memory-based* e *Model-based*.

2.2.3.1. Memory-based

Os algoritmos *memory-based* utilizam a totalidade da base de dados utilizador-item para gerar uma previsão. Estes sistemas empregam técnicas estatísticas para encontrar um conjunto de utilizadores, conhecidos como "vizinhos", cujo histórico de avaliações de *items* esteja próximo do histórico do utilizador-alvo. Estes utilizadores são encontrados através de métricas de semelhança ou de correlação, sendo o coeficiente de correlação de Pearson um dos mais utilizados (Celma 2010).

Assim que um conjunto de utilizadores fica formado são utilizados diferentes algoritmos que combinam as preferências dos vizinhos para produzir uma previsão de *rating* a um determinado item ou uma recomendação dos **top-N** *items* para o utilizador-alvo. Esta técnica, mais conhecida como *nearest-neighbor,* é uma das mais populares e mais utilizadas na prática.

2.2.3.2. Model-based

Ao invés dos algoritmos *memory-based,* este tipo de algoritmos de recomendação funciona através da criação de modelos aproximados dos *ratings* atribuídos pelos utilizadores, com recurso a técnicas como *machine learning* e algoritmos de *data mining*. Deste modo, o sistema aprende a reconhecer padrões complexos baseados em *training data,* podendo depois fazer previsões inteligentes para as tarefas do sistema de *collaborative filtering* sobre os dados dos utilizadores reais.

Segundo (Breese, Heckerman, Kadie 1998; Ungar, Foster 1998; Chien, George 1999), o processo de construção dos modelos envolve técnicas de *machine learning,* sendo que as mais utilizadas são as redes Bayesianas e *clustering*. As abordagens por redes Bayesianas formulam um modelo probabilístico, enquanto que os modelos por *clustering* tratam o *collaborative filtering* como um problema de classificação que funciona pelo agrupamento de utilizadores com gostos semelhantes em *clusters* e através da computação de probabilidades condicionais.

2.2.3.3. Limitações

Tal como os outros métodos de recomendação, o Collaborative filtering apresenta algumas limitações e problemas, de maior ou menor relevância. Autores como (Celma 2010), (Adomavicius, Tuzhilin 2005) e (Su, Khoshgoftaar 2009) definem os principais:

- **Escassez dos dados –** A escassez dos dados é uma característica própria da matriz de dados utilizador-item. Com um número relativamente alto de utilizadores e *items*, o problema principal é a baixa cobertura dos *ratings* atribuídos pelos utilizadores aos *items*. É comum ter-se uma matriz esparsa com apenas 1% de cobertura, o que dificulta a computação de vizinhanças e correlações fortes entre os elementos.

- **"Gray Sheep" –** Outro problema relacionado com o anterior é a existência de utilizadores com gostos atípicos ou que sejam muito diferentes dos outros utilizadores do sistema. Nesses casos é difícil encontrar vizinhanças e correlações significativas, o que origina recomendações pouco eficazes.

- **Sinonimidade dos items –** A sinonimidade refere-se à tendência para que um certo número de *items* iguais ou muito semelhantes tenham nomes e descrições diferentes. Por exemplo, um sistema de *Collaborative filtering* não consegue saber que dois *items* aparentemente diferentes com os nomes "*children movie*" e "*children film*" são na realidade o mesmo *item*, pelo que seriam tratados separadamente. É relatado que esta variabilidade na descrição dos *items* é maior do que se suspeita habitualmente, pelo que a sua prevalência diminui a qualidade das recomendações destes sistemas.

- **"Shilling attacks" –** Nos casos em que qualquer pessoa pode atribuir *ratings* livremente e a todos os *items*, podem haver situações de pessoas que atribuem *ratings* positivos aos seus próprios conteúdos e *ratings* negativos aos conteúdos de concorrentes, de maneira a condicionar as recomendações feitas a outros utilizadores para seu próprio proveito. É aconselhável que estes sistemas introduzam precauções de modo a desencorajar este tipo de fenómeno.

- **"Cold start" –** Uma vez que o *Collaborative filtering* é baseado nos *ratings* dos utilizadores, os utilizadores novos no sistema com poucos ou nenhuns *ratings* são mais difíceis de categorizar (e por conseguinte, fazer recomendações). Este problema ocorre também quando se adicionam novos *items*, pois quando são inseridos na coleção não têm nenhum *rating* associado. Por isso, nunca são recomendados até que os utilizadores comecem a atribuir-lhes *ratings*. No entanto este problema pode ser mitigado com a utilização de métodos de recomendação híbridos.

- **"Popularity bias" –** O *Popularity Bias* é outro problema que surge frequentemente nos sistemas de *Collaborative filtering*. É análogo ao paradigma *rich get richer*, em que *items* populares são similares ou relacionados com muitos outros *items*. Portanto, é muito mais provável que o sistema recomende os *items* populares. Esta limitação pode ser relevante pois a tendência de recomendação dos *items* mais populares sobrepõe-se muitas vezes a *items* menos populares mas que poderiam ser mais interessantes para os utilizadores.

2.2.4. Context-Based Filtering

O autor (Dey 2001) definiu *contexto* como qualquer informação que possa ser utilizada para caracterizar a situação de uma entidade como uma pessoa, lugar ou objecto que seja considerado relevante para a interação entre um utilizador e uma aplicação, incluindo o próprio utilizador e a aplicação. O contexto é um factor importante em serviços como a recomendação de música, pois é importante incorporar informação contextual sobre o cenário de decisão do utilizador no processo de recomendação (Adomavicius, Sankaranarayanan, Sen, Tuzhilin 2005).

O contexto do utilizador também se aplica noutros domínios de recomendação. No caso da recomendação de filmes, um sistema pode recomendar filmes diferentes consoante o utilizador o queira ver num sábado à noite com o/a namorado/a ou num dia de semana com os pais. Ou no caso de um sistema de recomendação de restaurantes, que toma em consideração a localização do utilizador (Horozov, Narasimhan, Vasudevan 2006).

Outros exemplos de sistemas desenvolvidos recorrendo a este método são o CoCo (Si, Kawahara, Kurasawa, Morikawa, Aoyama 2005), um sistema de recomendação de conteúdos físicos num mercado; um sistema de recomendação de filmes utilizando redes Bayesianas (Ono, Kurokawa, Motomura, Asoh 2007); e o CA-MRS, um sistema de recomendação de música (Park, Yoo, Cho 2006).

Em Marketing, a investigação comportamental sobre a tomada de decisões dos consumidores descobriu que esse processo depende do contexto em que essa decisão é tomada. O mesmo consumidor pode preferir diferentes produtos ou marcas sob diferentes condições ou situações (Klein, Yadav 1989). Portanto, é possível afirmar que a previsão precisa das preferências de um consumidor/utilizador depende invariavelmente do grau em que a informação contextual relevante é incorporada no processo de recomendação (Adomavicius, Sankaranarayanan, Sen, Tuzhilin 2005).

Como referido anteriormente, os sistemas de recomendação tradicionais baseiam-se no conhecimento de preferências parciais do utilizador, ou seja, as preferências do utilizador para um conjunto (muitas vezes limitado) de *items*, e os dados de *input* são tipicamente registos com a forma <*utilizador, item, rating*>. Em contraste, os sistemas de recomendação baseados em contexto são construídos com base no conhecimento de preferências contextuais parciais do utilizador e geralmente com a forma <*utilizador, item, contexto, rating*>, onde cada registo inclui não apenas o quanto o utilizador gosta

de um dado *item* mas também o contexto em que esse *item* foi "consumido" (p.ex: *contexto* = Domingo) (Adomavicius, Tuzhilin 2011).

Se tomarmos como exemplo de representação de um sistema de recomendação 2D como na figura 2, um sistema de recomendação baseado no contexto pode ser representado do mesmo modo mas com mais dimensões, correspondendo cada dimensão adicional a um elemento de contexto diferente. Por exemplo, se considerarmos Contexto = *Time*, podemos representar o espaço de recomendação *Utilizador x Item x Time* através de um cubo.

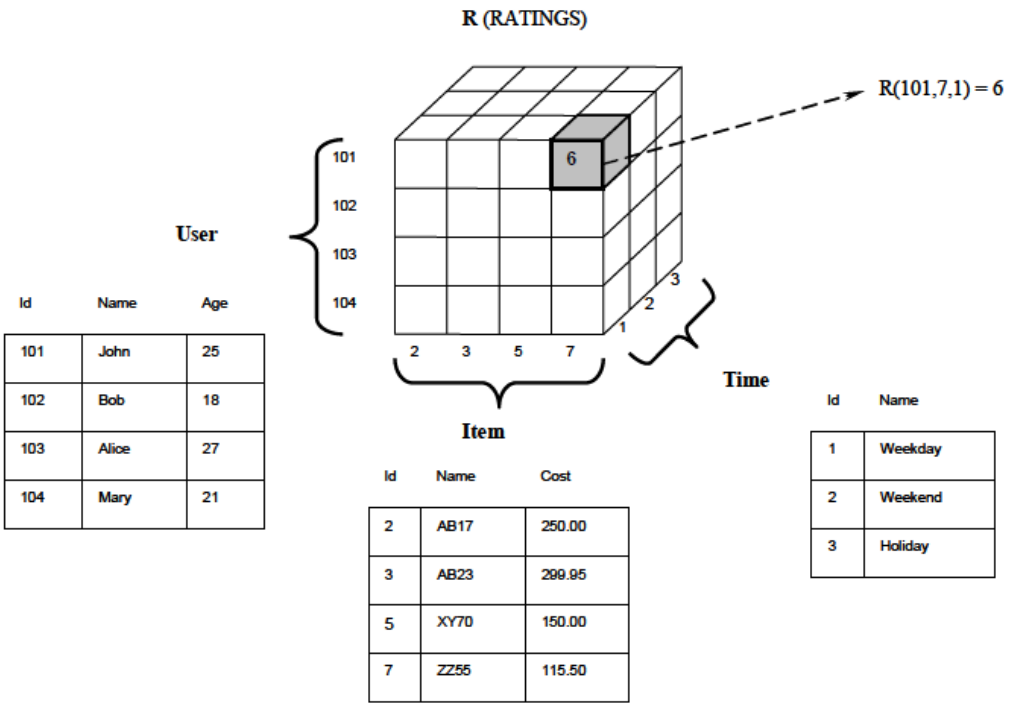

Figura 3 - Modelo multidimensional para o espaço de recomendação *User x Item x Time* (Adomavicius, Tuzhilin 2011)

Este cubo armazena *ratings* para a função R(*utilizador*, *item*, *time*), em que as três tabelas definem o conjunto de utilizadores, *items* e hipóteses temporais para cada uma das respectivas dimensões. Por exemplo, na figura 3 o *rating* R(101, 7, 1) = 6 significa que o utilizador com o ID 101 atribuiu o *rating* 6 ao *item* 7 num dia de semana.

2.2.4.1 Métodos de Incorporação de Elementos de Contexto

Ao contrário do processo de recomendação dos sistemas tradicionais, ilustrado na figura 1, que não tem em consideração a informação contextual, os sistemas de recomendação

baseados no contexto utilizam essa informação ao longo das várias fases do processo. Mais concretamente, o processo de recomendação dos sistemas baseados no contexto pode tomar uma de três formas, dependendo da fase em que essa informação é aplicada. De acordo com (Ricci, Rokach, Shapira 2011), os métodos de utilização de elementos de contexto em sistemas de recomendação podem ser categorizados de três modos:

- ***Contextual Pre-Filtering*** (figura 4a) – Neste paradigma, o contexto do utilizador define ele próprio os dados a serem utilizados no processo de recomendação.

  Por outras palavras, a informação do contexto atual é utilizada para selecionar e construir (filtrar) o conjunto de dados relevantes que servirá de *input*, a partir do conjunto de todos os dados. Por exemplo, se uma pessoa quiser ouvir música num sábado, apenas seriam utilizados os *ratings* atribuídos a músicas nesse dia da semana para o cálculo das músicas a recomendar.

- ***Contextual Post-Filtering*** (figura 4b) **–** Ao contrário do *Contextual Pre-Filtering*, neste paradigma os elementos de contexto são aplicados ao *output* do processo de recomendação. As previsões de *ratings* do utilizador são calculadas sobre o conjunto de todos os dados, recorrendo a um tradicional sistema de recomendação bi-dimensional (User x Item) e ignorando a informação contextual. Depois, o conjunto de recomendações resultante é ajustado à pessoa utilizando a informação de contexto.

- ***Contextual Modeling*** (figura 4c) – No *Contextual Modeling*, ao invés do que acontece nos outros dois paradigmas, não são utilizadas funções tradicionais de sistemas de recomendação bidimensionais. A informação contextual é usada diretamente na função de recomendação e no cálculo dos *ratings* previstos.

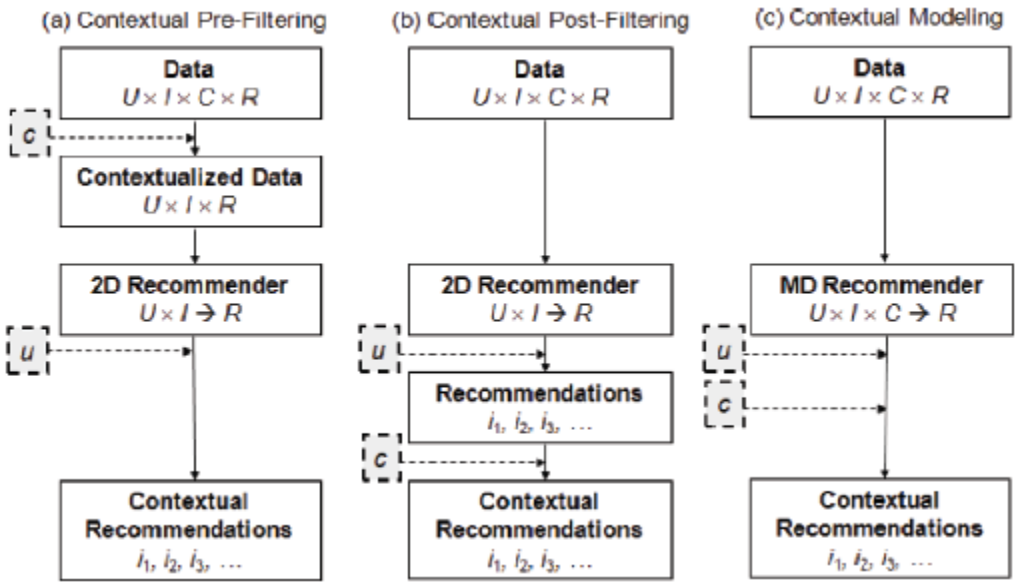

Figura 4 - Paradigmas de introdução de contexto em sistemas de recomendação

2.2.4.1.1. Contextual Pre-Filtering

Como é possível ver na figura 4a, o método de *contextual pre-filtering* utiliza a informação contextual para selecionar os dados 2D (Utilizador x Item) para a geração de recomendações. Uma das principais vantagens desta abordagem é a possibilidade de ser utilizado qualquer algoritmo de recomendação tradicional previamente referido na revisão da literatura (Adomavicius, Tuzhilin 2005). Mais especificamente, ao ser utilizado este método, o contexto ***c*** serve essencialmente como uma *query* para selecção dos *ratings* relevantes que servirão de *input* para um qualquer algoritmo de recomendação. Um exemplo de filtragem deste tipo pode ser num sistema de recomendação de filmes, em que se uma pessoa pretende assistir a um filme num sábado, apenas os dados com *ratings* atribuídos em sábados serão utilizados no cálculo.

No entanto, o contexto exato pode muitas vezes ser demasiado restrito. Consideremos por exemplo o contexto de assistir a um filme em Lisboa com uma namorada num domingo formalmente, *c* = (Lisboa, namorada, domingo). Utilizar este contexto exacto como *query* de filtragem dos dados de entrada pode ser problemático. Primeiro, as preferências do utilizador podem ser as mesmas quer seja num sábado ou num domingo, mas diferentes numa terça-feira. Segundo, para o contexto exato podem não haver dados suficientes que possibilitem a geração de recomendações precisas, que é conhecido como o problema da escassez dos dados.

2.2.4.1.2. Contextual Post-Filtering

De acordo como está ilustrado na figura 4b, a abordagem *contextual post-filtering* ignora a informação de contexto no *input* de dados do processo de recomendação. A lista de recomendações para o utilizador é gerada normalmente de acordo com o algoritmo escolhido. Depois, esta lista é ajustada de acordo com o contexto. Este ajuste pode ser feito de duas maneiras:

- Filtragem de recomendações que sejam irrelevantes num dado contexto
- Ajuste da ordem das recomendações na lista, baseado num dado contexto

Continuando com o exemplo do sistema de recomendação de filmes, se uma pessoa pretende assistir a um filme num fim de semana e se essa pessoa aos fins de semana só assiste a filmes de ação, o sistema pode remover todos os filmes que não sejam de ação da lista de recomendações. Mais genericamente, a ideia básica para abordagem *contextual post-filtering* é a de analisar as preferências contextuais de um dado utilizador num dado contexto para identificar padrões de utilização/consumo dos *items*, e posteriormente utilizar esses padrões para ajustar a lista de recomendações, resultando numa lista mais contextual como é possível ver na figura 5.

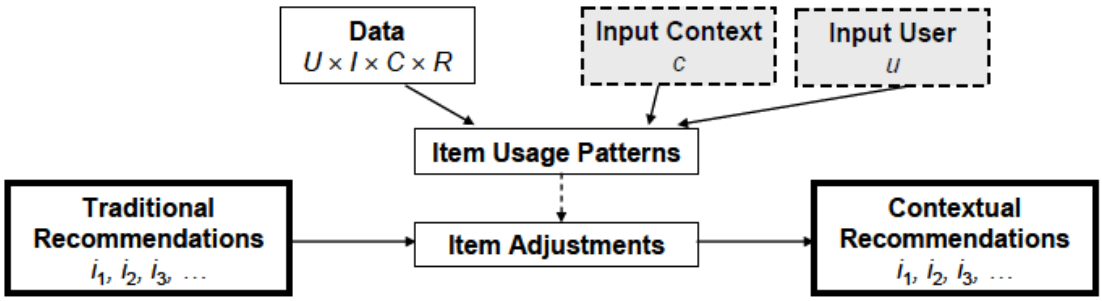

Figura 5 - Abordagem Contextual Post-Filtering: Ajustamento da lista de recomendações (Adomavicius, Tuzhilin 2011)

Tal como na abordagem *contextual pre-filtering*, a maior vantagem do *contextual post-filtering* é a possibilidade de utilização de um qualquer algoritmo de recomendação tradicional.

2.2.4.1.3. Contextual Modeling

A abordagem *contextual modeling* (figura 4c) utiliza a informação contextual diretamente na função de recomendação. Enquanto as abordagens *contextual pre-filtering* e *post-filtering* usam funções de recomendação 2D, o *contextual modeling* dá origem a funções verdadeiramente multidimensionais, que são essencialmente modelos preditivos (construídos usando árvores de decisão, regressão e modelos probabilísticos ou outras técnicas) que incorporam informação contextual em adição aos dados *utilizador* e *item*. Um número significativo de algoritmos tem sido desenvolvido nos últimos 10-15 anos, e algumas dessas técnicas são extensões de modelos 2D para modelos multidimensionais (Adomavicius, Sankaranarayanan, Sen, Tuzhilin 2005).

2.2.4.2. Obtenção de informação de contexto

As informações de contexto podem ser obtidas de diferentes formas, nomeadamente:

- **Explicitamente** – através da abordagem direta ao utilizador (ou a outras fontes de informação contextual) e recolha explícita desta informação através de questões ou pedidos diretos. Por exemplo, um *website* sobre restaurantes pode pedir ao utilizador para preencher um questionário ou para responder a questões específicas antes de ter acesso à funcionalidade de recomendações.
- **Implicitamente** – através da recolha de dados da envolvente do utilizador sem a intervenção do mesmo, como a mudança de localização do utilizador detectada pelo sensor de GPS do telemóvel. Outro exemplo é o contexto temporal, que pode ser implícito através do *timestamp* da operação. Nestes casos não é necessária nenhuma interação com o utilizador, visto que as fontes da informação contextual são acessíveis directamente.
- **Por inferência** – utilizando métodos estatísticos ou de *data mining*, por exemplo para identificar o perfil de um determinado espectador de televisão por cabo. Embora não seja explicitamente conhecido, pode ser inferido com uma precisão razoável a partir dos programas a que assistiu e que canais visitou.(Palmisano, Tuzhilin, Gorgoglione 2008) demonstraram que em certos domínios podem ser inferidos vários tipos de informação contextual utilizando certos métodos de *data mining.* Para isso, é necessário construir um modelo preditivo antecipadamente. O sucesso da inferência de informação contextual depende significativamente da qualidade deste modelo.

2.2.4.3. Limitações

As principais limitações e problemas dos sistemas de recomendação baseados no contexto têm a ver com a filtragem dos dados e/ou recomendações em resultado de um contexto demasiado específico. Em situações em que sejam utilizados diversos elementos de contexto simultaneamente ou contextos com uma granularidade muito elevada (p. ex. um contexto *Tempo* registado ao nível dos segundos), a eficácia e o valor destes sistemas para os utilizadores pode descer consideravelmente, principalmente se a base de *ratings* tiver uma dimensão reduzida. Posto isto, é necessário encontrar um equilíbrio entre a especificidade do contexto introduzido no sistema de recomendação e o valor que produz para o utilizador final.

2.2.4.4. Sistema de Recomendação de Música baseado no contexto – Exemplo

Um dos exemplos de sistemas de recomendação de música baseados no contexto é o C^2_Music, desenvolvido por (Lee, Lee 2007). Em termos gerais, é um sistema que se caracteriza por utilizar o método de *contextual pre-filtering* recorrendo ao algoritmo de inteligência artificial *Case-based Reasoning*.

Para avaliar a performance do C^2_Music, (Lee, Lee 2007) implementaram um sistema semelhante com a única diferença de não utilizar informação contextual nas suas recomendações, de modo a poderem comparar a eficácia de ambos. De facto, de acordo com este estudo concluíram quea introdução de informação contextual em sistemas de recomendação, no domínio da música, aumenta a precisão das recomendações.

2.2.5. Abordagens Híbridas

Muitos sistemas de recomendação utilizam uma abordagem híbrida, através da combinação de dois ou mais métodos de recomendação. A ideia desta abordagem é a de aproveitar as vantagens de um dos métodos para colmatar as limitações do outro (Balabanović, Shoham 1997; Claypool, Gokhale, Miranda, Murnikov, Netes, Sartin 1999; Pazzani 1999; Schein, Popescul, Ungar, Pennock 2002; Burke 2007). A combinação mais utilizada é entre *Collaborative filtering* e *Content-based Filtering*, em que o problema *cold-start* do *collaborative filtering* é resolvido através da capacidade que o método *Content-based* tem de recomendar *items* com base nas suas características. Ou seja, deste modo é possível colmatar algumas das limitações que cada uma destas abordagens apresentam separadamente.

Existem diferentes maneiras de combinar estas duas abordagens, sendo que (Adomavicius, Tuzhilin 2005) identificam as seguintes:

- Implementação de ambos os métodos separadamente e combinação dos seus *outputs*/previsões.
- Incorporação de algumas características de recomendação por *collaborative filtering* num sistema *Content-based.*
- Incorporação de algumas características de recomendação *Content-based* num sistema de *collaborative filtering*.
- Construção de um modelo unificado que incorpore características de ambas as abordagens.

Existe investigação diversa feita nesta área (Balabanović, Shoham 1997; Melville, Mooney, Nagarajan 2002; Pazzani 1999) que comprova a maior eficácia das recomendações efectuadas por esta abordagens quando comparada com métodos *collaborative filtering* e *content-based* puros.

2.2.6. Avaliação de Sistemas de Recomendação

Segundo (Konstan, Riedl 1999; Herlocker, Konstan, Terveen, Riedl 2004), a avaliação de sistemas de recomendação pode ser feita de dois modos distintos:

- ***Offline*** – a avaliação do sistema é realizada a partir de um *dataset* com as preferências dos utilizadores, previamente adquiridas. Estes *datasets* são geralmente oriundos de terceiros. Alguns exemplos são o EachMovie[4], MovieLens[5] , ou o Million Song Dataset[6].

- ***Online*** – a avaliação é realizada a partir de experimentação e/ou observação de um sistema de recomendação em produção. Por exemplo, consideremos um investigador que tente determinar se a satisfação dos utilizadores aumenta com um sistema de recomendação que use um sistema de *ratings* de sete pontos em vez de cinco. Neste caso pode preferir realizar uma experiência em que os utilizadores sejam confrontados com diferentes interfaces, o seu comportamento seja observado e sejam inclusivamente entrevistados. De facto, a maioria da investigação referente à utilização das tecnologias de recomendação em novas aplicações, em novas interações ou novas interfaces, requerem experimentação *online* (Konstan, Riedl 1999).

A maioria do trabalho desenvolvido na avaliação de algoritmos de recomendação tem-se focado na análise *offline* da precisão de predições, em que o algoritmo em estudo é utilizado para prever determinados valores no *dataset*, geralmente *ratings* atribuídos a *items*. Posteriormente os resultados são analisados recorrendo a uma ou mais métricas criadas para o efeito. Este tipo de avaliações tem a vantagem de ser rápida e económica, permitindo realizar avaliações em grande escala e em diferentes *datasets* ou com diferentes algoritmos, simultaneamente e com pouco trabalho acrescido.

No entanto, a avaliação *offline* tem duas desvantagens importantes. Primeiro, a escassez natural dos *ratings* nos *datasets* limitam o número de *items* que podem ser utilizados na avaliação, pois não é possível avaliar a adequação de um *item* recomendado a um utilizador se não existir um *rating* do mesmo utilizador para esse *item*. Segundo, a análise é limitada à avaliação da predição de resultados.

[4] http://grouplens.org/datasets/eachmovie/
[5] http://grouplens.org/datasets/movielens/
[6] http://labrosa.ee.columbia.edu/millionsong/

Nenhuma análise *offline* consegue avaliar a satisfação dos utilizadores com um determinado sistema, e se essa satisfação se deve às suas recomendações ou a factores mais subjetivos como a interface. Nestes e noutros casos semelhantes, a abordagem alternativa é realizar uma experiência com utilizadores. Esta experiência pode ser controlada (p. ex. escolhendo cada um dos utilizadores e submetendo-os a determinados cenários) ou um estudo de campo em que o sistema é disponibilizado a uma comunidade de utilizadores e os efeitos do sistema são observados (Herlocker, Konstan, Terveen, Riedl 2004).

Também (Hayes, Cunningham 2002) argumentam que as avaliações *offline* são muito limitadas: ao serem utilizados *datasets* externos (como o Million Song Dataset) o contexto em que as recomendações são feitas é ignorado, sendo para isso necessário realizar uma avaliação *online*. Afirmam também que o propósito dos sistemas de recomendação é ajudar os utilizadores a explorar os recursos disponíveis no domínio do sistema, de maneira que a satisfação do utilizador se traduza num uso continuado desses recursos por parte deste último. E que, portanto, é esta satisfação que deve ser medida aquando da avaliação de um sistema de recomendação. As avaliações *offline* , ao recorrerem aos referidos *datasets*, não permitem a medição de tais parâmetros.

O processo de validação e avaliação dos sistemas de recomendação depende de diversos factores, nomeadamente dos objectivos do sistema, do domínio das recomendações, do contexto em que o sistema vai ser utilizado, das conclusões que se pretendem retirar, entre outros. Isto porque para além do poder de predição do algoritmo de recomendação, os utilizadores podem estar interessados em descobrir novos *items* que desconheciam, ter uma resposta rápida do sistema, ter a sua privacidade salvaguardada, entre outras propriedades da interação com o sistema de recomendação. É por isso necessário estabelecer um conjunto de propriedades que possam influenciar o sucesso de um sistema de recomendação no contexto de uma aplicação específica. Depois, o sistema pode então ser avaliado com base nas propriedades relevantes que se pretende analisar.

Os autores (Shani, Gunawardana 2011) apresentaram uma lista de propriedades destes sistemas que servem para este propósito, nomeadamente:

- Preferências dos utilizadores
- Precisão das predições
- Cobertura do espaço de *items*
- Cobertura do espaço de utilizadores
- Performance de recomendações para novos utilizadores
- Performance de recomendações para novos *items*
- Confiança do sistema nas suas recomendações
- Confiança do utilizador nas recomendações
- Novidade nos *items* recomendados
- *Serendipity* (surpresa) do utilizador com os *items* recomendados
- Diversidade nas recomendações
- Utilidade das recomendações
- Risco associado a más recomendações
- Robustez das recomendações
- Privacidade do utilizador
- Capacidade de adaptação
- Escalabilidade

Cada uma destas propriedades pode ter maior ou menor relevância, dependendo dos objectivos do sistema avaliado. Podem inclusivamente ocorrer *trade-offs*, ou seja, abdicar de parte ou totalidade de uma ou várias propriedades em benefício de outras.

2.2.6.1. Avaliação Online

Ao contrário da avaliação *offline* de sistemas de recomendação, a avaliação *online* padece da falta de métricas estandardizadas que permitam uma comparação directa de algoritmos e sistemas, inclusivamente em diferentes domínios. Este facto justifica-se com as próprias características deste tipo de análise, que assenta na observação e registo do comportamento dos utilizadores e da sua interação com o sistema. As suas conclusões são, portanto, subjetivas e restritas ao domínio em estudo, ao contexto da utilização do sistema, entre outros factores.

No entanto, (Herlocker, Konstan, Terveen, Riedl 2004) propuseram um conjunto de dimensões de avaliação que auxiliam nesta análise:

- **Explícito (perguntar) vs Implícito (observar) –** distinção básica entre avaliações que pedem explicitamente reações aos utilizadores sobre o sistema e avaliações que observam o seu comportamento implicitamente. Enquanto o primeiro tipo utiliza questionários e entrevistas, no segundo o comportamento do utilizador é registado e posteriormente sujeito a vários tipos de análises.

- **Estudos laboratoriais vs estudos de campo –** estudos laboratoriais permitem focar a investigação em aspectos específicos e testar hipóteses bem definidas sob condições controladas. Estudos de campo podem revelar como se comportam realmente os utilizadores no seu contexto real, demonstrando usos comuns e padrões de utilização ou questões que os investigadores pudessem não identificar num estudo laboratorial.

- **Curto-prazo vs longo-prazo –** algumas questões podem não ser aparentes num estudo de curto-prazo, particularmente em estudos laboratoriais. (Turpin, Hersh 2001) verificaram que, apesar dos sujeitos de um estudo realizarem com sucesso as tarefas propostas de utilização de um sistema, ao longo do tempo tinham tendência para ficar insatisfeitos, desencorajados, e eventualmente paravam de utilizar o sistema.

2.2.7. Comparação de Sistemas de Recomendação

No caso particular do estudo comparativo de sistemas de recomendação, um método utilizado é a medição do comportamento dos utilizadores na sua interação com os sistemas comparados. Segundo (Shani, Gunawardana 2011), se os utilizadores seguem as recomendações de um dos sistemas mais frequentemente, então é possível concluir que esse sistema é superior aos restantes no estudo (assumindo que todos os outros elementos do sistema que possam influenciar a satisfação do utilizador se mantêm constantes).

Geralmente nestes casos, o tráfego do sistema é direcionado para cada um dos motores de recomendação em estudo e são guardadas as interações dos utilizadores com os diferentes algoritmos. No entanto, são necessárias algumas condições para assegurar o sucesso destes testes. Por exemplo, é importante assegurar que os utilizadores são direccionados para cada um dos motores de recomendação aleatoriamente, de modo a que a comparação entre as alternativas seja justa. É também importante uniformizar todos os aspectos dos vários sistemas, excepto o componente que se pretende comparar. Por exemplo, se se pretende comparar algoritmos, a interface deve manter-se constante. Por outro lado, se o objectivo é encontrar a melhor interface, o algoritmo de recomendação deve ser igual.

2.2.7.1. Frameworks de Recomendação

O componente principal dos sistemas de recomendação é o algoritmo que gera as recomendações. Nesta secção é analisado um conjunto de *frameworks open* source existentes que facilitam o desenvolvimento de sistemas de recomendação ao disponibilizarem alguns dos algoritmos e funcionalidades mais comuns e populares, permitindo ao programador focar-se no desenvolvimento dos componentes específicos do seu sistema. Estas *frameworks* foram alvo de uma análise por parte de (Sousa 2012) e (Lucas 2010).

2.2.7.1.1 Mahout

O Mahout Taste[7] é uma *framework* escrita em Java que disponibiliza um conjunto de componentes para a construção de um sistema de recomendação personalizável, com especial destaque para os algoritmos de recomendação por *collaborative filtering*. Foi desenhado para ser flexível, escalável e de elevada performance. Embora seja escrito em Java, pode ser utilizado como servidor externo que expõe o seu motor de recomendações a outras aplicações através de *web services* ou por HTTP.

É parte integrante do projeto Apache Mahout, que tem por objectivo desenvolver algoritmos *open source* escaláveis de aprendizagem automática e *data mining*, com a possibilidade de serem utilizados em sistemas distribuídos com recurso ao paradigma Map/Reduce. O Mahout Taste é um dos componentes mais antigos, mais estáveis e mais utilizados deste projeto.

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

Disponibiliza vários tipos de algoritmos de recomendação, como o *collaborative filtering* (*user-to-user* e *item-to-item*) e o *slope one*, algoritmos de *clustering* e de classificação. Disponibiliza também ferramentas para avaliação dos sistemas de recomendação implementados.

2.2.7.1.2. RACOFI

O RACOFI[89] (acrónimo para "Rule-Applying Collaborative Filtering") é um sistema de recomendação composto por dois componentes principais: o COFI, que produz recomendações de *items* com um algoritmo de *collaborative filtering*, e o RALOCA (Rule-Applying Learning Object Comparison Agent), que filtra as recomendações produzidas pelo COFI de acordo com um conjunto de regras e informações do conteúdo dos *items* (Anderson, Ball, Boley, Greene, Howse, McGrath, Lemire 2003). A figura 6 apresenta a arquitectura deste sistema.

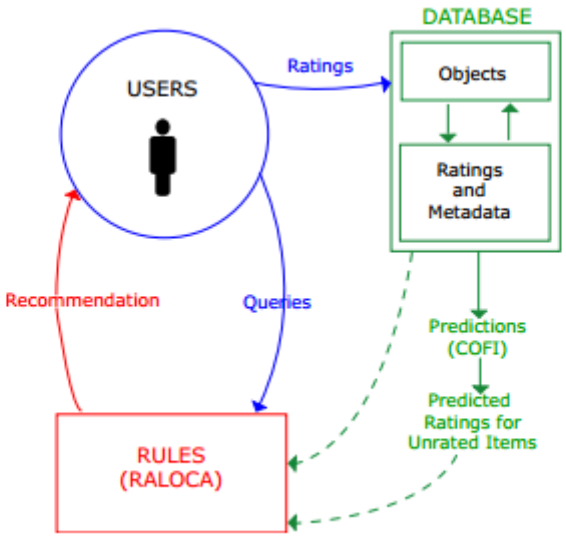

Figura 6 - Arquitectura do sistema RACOFI

2.2.7.1.3. ColFi

Esta *framework* (figura 7) implementada em Java reúne um conjunto de algoritmos de *collaborative filtering* (*user-to-user* e *item-to-item*), tendo sido desenvolvido na Universidade de Charles na República Checa. O ColFi[10] (acrónimo para Collaborative Filtering) é bastante flexível, visto que é possível acrescentar novos algoritmos colaborativos facilmente (Brozovsky, Petricek 2007).

[8] http://cofi.elg.ca/racofi.html
[9] http://lemire.me/fr/abstracts/COLA2003.html
[10] http://colfi.wz.cz/

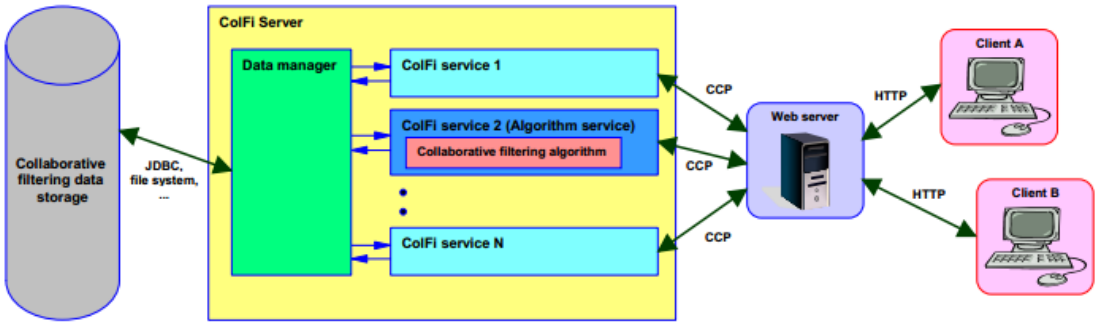

Figura 7 - Arquitectura do sistema ColFi

2.2.7.1.4. Duine Framework

O Duine Framework[11] (figura 8) é um projecto composto por um conjunto de bibliotecas de *software open source* escritas em Java, e que possibilitam o desenvolvimento de sistemas de predição de itens com base em diferentes técnicas. Para além das técnicas básicas de recomendação baseadas na relação quantificada entre um utilizador e um *item* (*rating*), o Duine também processa e armazena metadados sobre cada item avaliado (p. ex. o artista ou género de uma música) de modo a mais facilmente determinar os interesses do utilizador. Ou seja, integra também técnicas de recomendação baseadas no conteúdo.

Atualmente, o Duine oferece sete técnicas de previsão/recomendação: *Collaborative Filtering*, *Information Filtering* (recomendação baseada no conteúdo), *Case-based Reasoning*, *GenreLMS*, *TopNDeviation*, *AlreadyKnown* e *UserAverage*.

Três das suas vantagens são:

- Extensibilidade – pode ser estendido com novas técnicas de predição, modelação de perfis de utilizador, entre outros.
- Validação – disponibiliza ferramentas de validação e medição da precisão das recomendações.
- Explicação de resultados – possui ferramentas de explicação *user-friendly* que podem ser diretamente integradas nas aplicações *end-user*.

[11] http://www.duineframework.org/

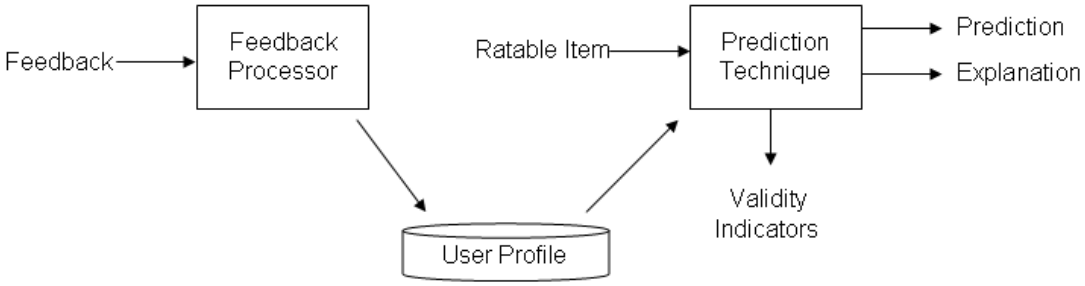

Figura 8 - Arquitectura do sistema Duine

3. Proposta Conceptual

Neste capítulo irão ser descritas as funcionalidades do sistema proposto, a sua arquitetura técnica e os seus componentes. A este sistema iremos chamar Music Discovery.

O propósito principal deste sistema é o de verificar a veracidade da hipótese desta dissertação, que implica a comparação de dois algoritmos de recomendação. Para isso, e de acordo com os métodos de avaliação referidos no capítulo anterior, decidiu-se fazer um estudo de campo *online*. Esta decisão tem por base a métrica que vai ser analisada – a satisfação do utilizador. Para isso contribui a disponibilização do sistema via *web*, pois é possível apreender o comportamento dos utilizadores num contexto de utilização real e de sua livre e espontânea vontade.

Posto isto, foi desenhado um sistema não só capaz de recolher os dados necessários à obtenção de uma resposta ao problema da dissertação mas também de fácil utilização e atractivo para os utilizadores. De modo a que os casos de uso deste sistema sejam mais perceptíveis, iremos considerar o seguinte cenário:

> "O utilizador A é um estudante universitário com 20 anos que gosta de ouvir música com muita regularidade. São nove horas da manhã de uma Sexta-feira e está um dia de chuva. O utilizador acede ao seu serviço de *streaming* de música favorito, que lhe recomenda a música '*Times Like These*' da banda Foo Fighters, ouvida habitualmente por outros utilizadores com gostos semelhantes ao utilizador A em dias como o de hoje."

Nesta situação a música recomendada seria escolhida pelo Music Discovery, sem nenhum *input* por parte do utilizador para além dos *ratings* atribuídos previamente a outras músicas do seu interesse.

A partir deste cenário podemos inferir as funcionalidades essenciais do sistema proposto:

- **Disponibilização de uma base de dados de música -** possibilidade de navegar pelas diferentes categorias/géneros e ouvir todas as músicas do sistema.

- **Atribuição de *ratings* –** *input* quantificável que reflete o interesse do utilizador numa determinada música. Estes dados são mais tarde utilizados no cálculo de recomendações.
- **Recolha passiva dos elementos do contexto do utilizador –** utilização de processos e dispositivos/sensores de recolha de elementos do contexto do utilizador sem a intervenção do mesmo (p. ex. sensor GPS).
- **Recomendação de músicas –** recomendação de acordo com os gostos do utilizador e com o contexto em que o mesmo se encontra.

No entanto, e como já foi referido, para responder ao problema desta dissertação não é suficiente construir um sistema de recomendação baseado no contexto do utilizador, visto que se pretende fazer uma comparação com um sistema de *collaborative filtering* tradicional. Para isso é necessário que a solução proposta integre ambos os métodos e a recolha de dados da sua utilização seja feita de forma invisível para o utilizador. Nomeadamente, sempre que for pedida uma recomendação a lista de músicas a recomendar será gerada alternadamente entre os dois métodos. A metodologia de (Hayes, Cunningham 2002) para avaliação *online* de sistemas de recomendação incorpora uma *framework* conceptual para comparação de dois algoritmos distintos. De modo a ser possível desenvolver uma solução capaz de responder convenientemente ao problema desta dissertação, decidiu-se adoptar os princípios presentes nesta *framework.*

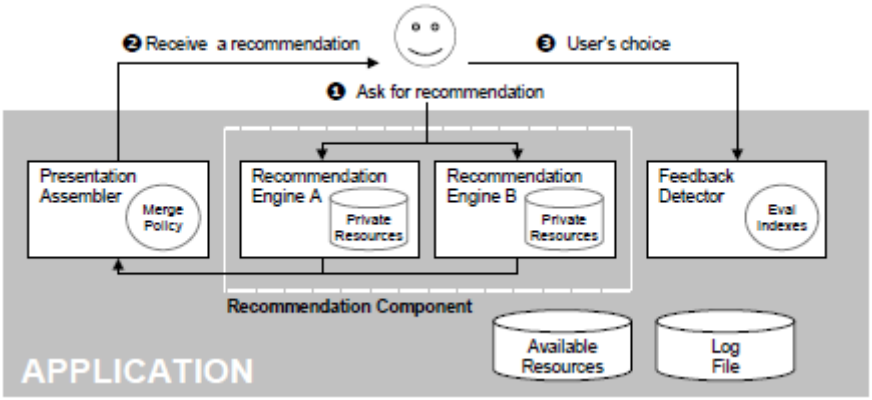

Figura 9 - Framework de comparação de sistemas de recomendação (Hayes, Cunningham 2002)

Na figura 9 é possível observar os diferentes componentes da *framework*, como se interligam e como interagem com o utilizador. Os principais são:

- **Recommendation Component**

  Este componente recebe o pedido de recomendação por parte do utilizador e retorna uma ou mais recomendações. Internamente é composto por dois algoritmos de recomendação distintos. As recomendações podem ser geradas por um dos algoritmos ou por ambos separadamente, dependendo das configurações do sistema.

- **Presentation Assembler**

  Este componente é responsável pela apresentação das recomendações ao utilizador. Caso se opte pela geração de recomendações por ambos os algoritmos, é aqui que se define como é que esse par de resultados se conjuga num *output* único para o utilizador (*Merge Policy*). Os modos possíveis são o *Merged Set* (unificação das recomendações dos dois algoritmos numa única lista), *Contrasting Set* (separação clara das recomendações de ambos os algoritmos) e o *Cascading Set* (apresentação de recomendações de apenas um dos algoritmos alternadamente para cada pedido de recomendações).

- **Feedback Detector**

  O *Feedback Detector* é responsável por registar o comportamento do utilizador e determinar a sua satisfação relativamente a cada um dos algoritmos. Esta determinação pode ser feita através de indicações expressas do utilizador (atribuição de um *rating* ao *item* recomendado) ou inferida a partir de outros factores, como o tempo de permanência no *website* ou o tempo que despende a experienciar o *item* que lhe foi recomendado.

Para além destes três componentes principais, existem também duas bases de dados com os recursos disponíveis (*items* do domínio que o sistema pretende recomendar, p. ex. faixas de música) e o registo do comportamento e ações dos utilizadores.

3.1. Elementos de Contexto

Como foi referido anteriormente, o contexto do utilizador será tido em consideração na geração de recomendações. No caso em estudo, os elementos de contexto que serão introduzidos no cálculo das recomendações serão o estado do tempo, a hora do dia e o dia da semana. A recolha destes dados será feita sem intervenção do utilizador, de modo a que não haja perturbação da sua experiência de utilização do sistema.

A recolha da hora e do dia da semana são simples, através do relógio interno da máquina onde corre o sistema. No entanto, a recolha da informação sobre o estado do tempo no local onde se encontra o utilizador requer mais passos:

- São recolhidas as coordenadas geográficas do utilizador, através da API de geolocalização da linguagem HTML5 ou através do sensor de GPS caso se trate de um dispositivo móvel.
- As coordenadas são passadas a um serviço *web* externo que devolve o estado do tempo nessa localização (*OpenWeather*).

Deste modo, é possível associar a cada *rating* a hora, o dia e o estado do tempo que se faz sentir sempre que o utilizador oiça uma música ou solicite uma recomendação. Juntamente com a indicação do utilizador que atribuiu o *rating* e a que música foi atribuído, ficam completos os dados necessários para a geração de recomendações.

Como foi referido no capítulo Conceitos e Trabalho Relacionado na secção *Context-based Filtering*, existem três modos de integração da informação contextual num sistema de recomendação bi-dimensional (*Context Pre-Filtering*, *Context Post-Filtering* e *Contextual Modeling*). De acordo com as características e vantagens de cada um, foi escolhido o *Context Pre-Filtering* para ser utilizado na solução desta dissertação. Esta técnica implica a filtragem prévia da base global de *ratings*, de maneira a que apenas os *ratings* atribuídos por todos os utilizadores num contexto igual ou semelhante (dependendo do grau de precisão e relevância dos dados pretendidos) sejam utilizados como *input* para a geração de recomendações.

3.2. Arquitectura do Sistema

Na figura 10 é apresentada a arquitetura proposta por (Lee, Lee 2007) para um problema semelhante de recomendação de música baseada no contexto. A solução proposta para esta dissertação irá basear-se nesta arquitetura e nos princípios da *framework* de (Hayes, Cunningham 2002).

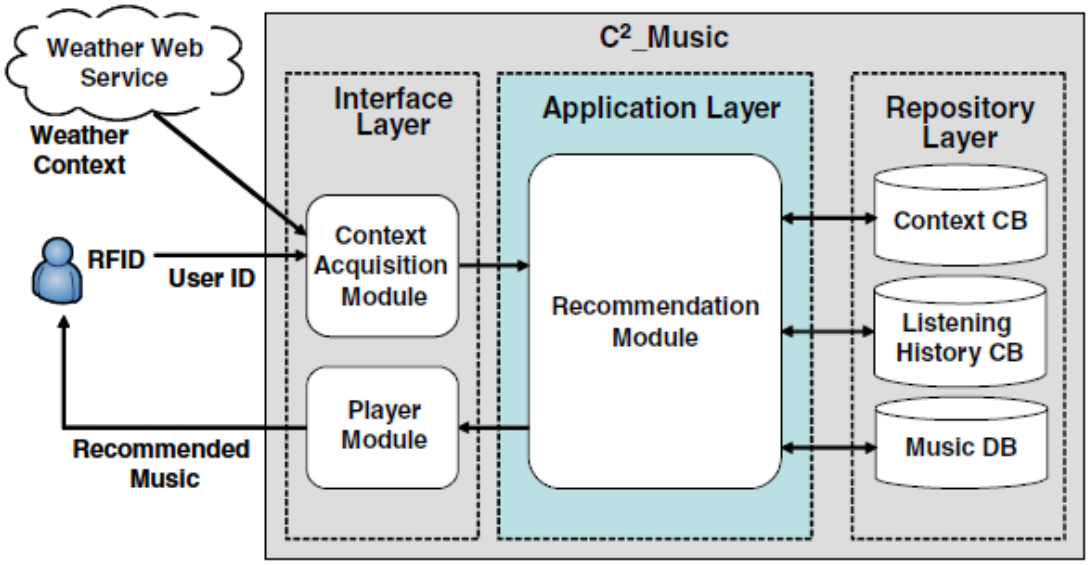

Figura 10 - Arquitectura do sistema de recomendação C^2_Music (Lee, Lee 2007)

De acordo com as especificidades desta dissertação, nomeadamente a necessidade de comparação de dois algoritmos de recomendação e os elementos de contexto em questão, a arquitetura proposta para a solução que se pretende implementar é apresentada na figura seguinte.

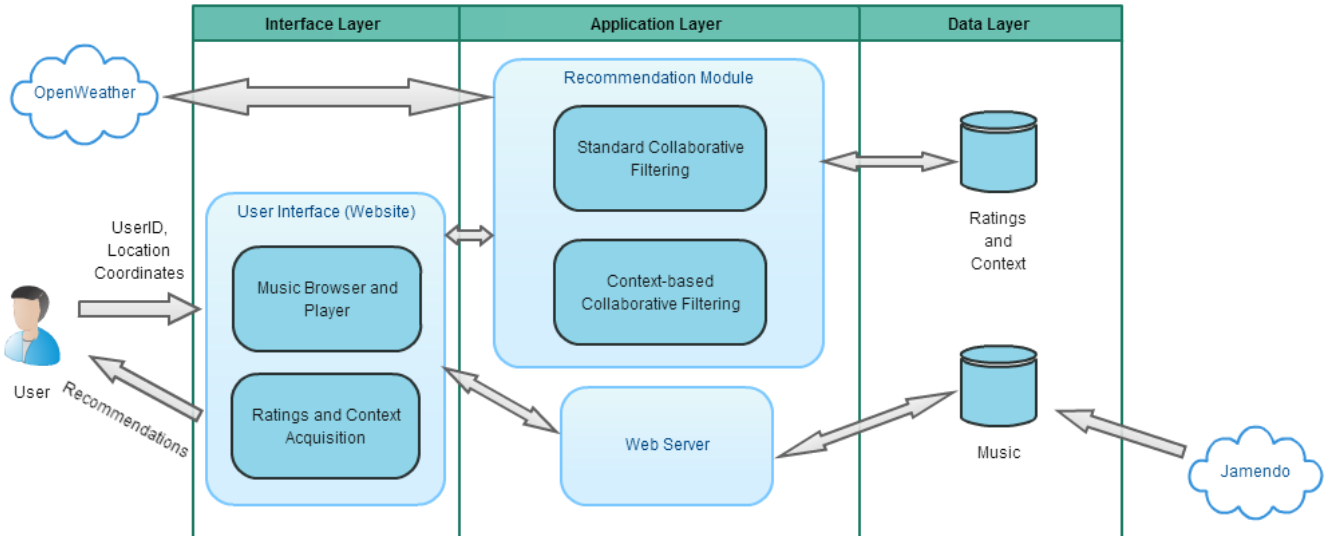

Figura 11 - Arquitectura do sistema proposto

A *Interface Layer* é responsável por toda a interação com o utilizador, nomeadamente através de um *website*, em que o utilizador pode navegar por toda a música, ouvir, avaliar e receber recomendações. É também aqui que são recolhidas as coordenadas geográficas do utilizador, através da API de geolocalização da tecnologia HTML5.

A lógica do sistema encontra-se na *Application Layer*. É aqui que as recomendações são geradas e onde a base de *ratings* é alimentada, no módulo de recomendação. Este módulo comunica com a API *OpenWeather*, de modo a obter as condições meteorológicas na localização geográfica obtida na *Interface Layer*. É também aqui que a interface com o utilizador é suportada, através de um *Web Server*.

A *Data Layer* é composta por duas bases de dados, nomeadamente a base de dados composta por *ratings*, elementos do contexto do utilizador e outros dados necessários ao cálculo de recomendações, e a base de músicas. Esta última é alimentada pelo serviço de música gratuita Jamendo.

3.3. Comparação de *frameworks* de recomendação

Como foi referido na proposta conceptual de solução, um dos componentes necessários é um sistema de recomendação com algoritmos de *collaborative filtering* capaz de gerar recomendações de música. O quadro seguinte compara as *frameworks* referidas na revisão da literatura de acordo com algumas das características relevantes para a escolha da *framework* a utilizar. Esta comparação foi realizada por (Sousa 2012) e (Lucas 2010).

	Mahout Taste	RACOFI	ColFi	Duine Toolkit
Última actualização	Julho 2013	2003	Abril 2013	Fevereiro 2009
Documentação	Sim (site e livro publicado)	Não	Não	Sim
Comunidade activa	Sim	Não	Não	Sim
Avaliação de Recomendações	Sim	Não	Não	Sim
Abstracção do modelo de dados	Sim	Não especificado	Não especificado	Sim
Filtragem Colaborativa	X	X	X	X
Filtragem baseada no conteúdo		X		X
Filtragem Híbrida		X	X	X

Tabela 2 - Comparação de Frameworks

3.3.1. Conclusão

De acordo com os requisitos necessários para a solução proposta e através da comparação que foi feita, a *framework* escolhida foi o Mahout Taste. As principais razões para a escolha desta ferramenta em detrimento de outras foram a sua comunidade ativa e o seu desenvolvimento contínuo, a quantidade e qualidade da documentação disponível, e a possibilidade de ser utilizado como servidor externo via *webservices* ou HTTP.

4. Implementação

Neste capítulo é apresentada a implementação do sistema proposto, com as decisões de implementação justificadas. Com a excepção da secção sobre as tecnologias utilizadas no desenvolvimento, este capítulo está dividido em secções que espelham a as camadas da arquitetura proposta.

4.1. Tecnologias utilizadas

Para o desenvolvimento do sistema proposto foram utilizadas um conjunto de tecnologias e linguagens de programação, cada uma com a sua função específica. Para o armazenamento e gestão dos dados foi utilizado o Sistema de Gestão de Bases de Dados (SGBD) MySQL[12], um dos sistemas deste género mais utilizados no mundo. A sua robustez, documentação disponível e facilidade de utilização/integração foram os factores-chave para a sua escolha.

Na camada aplicacional foram utilizados dois servidores distintos: O Apache HTTP Server[13] e o Apache Tomcat[14]. O Apache HTTP Server, um dos servidores *web* mais utilizados no mundo, foi escolhido pela sua facilidade de utilização e integração com outras tecnologias como a linguagem de programação PHP[15]. O seu propósito é o de suportar o funcionamento online de todo o sistema à exceção do módulo de recomendação, que é executado pelo servidor de programas escrito em Java[16] Apache Tomcat. Embora não seja o único servidor de aplicações Java, o Apache Tomcat foi escolhido pois é uma dependência do Apache Mahout. Este último, como já foi referido anteriormente, é o motor de recomendações utilizado no sistema.

[12] http://www.mysql.com/
[13] http://httpd.apache.org/
[14] http://tomcat.apache.org/
[15] http://php.net/
[16] http://www.java.com

4.2. Implementação – Camada de dados

É na camada de dados que são armazenados todos os recursos utilizados pelo sistema para gerar recomendações, nomeadamente os registos de todos os utilizadores, todas as músicas e os *ratings* atribuídos pelos utilizadores a essas mesmas músicas. Estes registos encontram-se num SGBD MySQL em três tabelas distintas: *Users*, *Tracks* e *RatingsContext*.

4.2.1. Users

A tabela *Users* é relativa aos utilizadores presentes no sistema, de modo a ser possível identificar cada utilizador e gerar recomendações personalizadas. Neste caso a tabela tem apenas dois campos: *id* e *idFacebook*.

#	Coluna	Tipo	Collation	Atributos	Nulo	Padrão	Extra
1	id	int(11)			Não	*None*	AUTO_INCREMENT
2	idFacebook	varchar(45)	latin1_swedish_ci		Não	*None*	

Tabela 3 - Tabela Users

Esta abordagem é diferente das abordagens tradicionais, em que são guardados diversos dados identificativos de cada utilizador. Visto que o utilizador faz *login* no sistema através de *Facebook Login*[17], resolveu-se utilizar apenas o seu *id* da rede social Facebook[18] para o identificar e evitar a recolha e armazenamento de outros dados pessoais que seriam desnecessários.

4.2.2. Tracks

A informação relativa às músicas presentes no sistema, e que podem ser ouvidas e recomendadas, encontra-se na tabela *Tracks*. É de salientar que os ficheiros de música não se encontram no sistema, são sim carregados a partir do serviço de música gratuita Jamendo[19]. Esse processo será explicado mais à frente.

[17] https://developers.facebook.com/docs/reference/plugins/login/
[18] https://www.facebook.com/
[19] http://www.jamendo.com

#	Coluna	Tipo	Collation	Atributos	Nulo	Padrão	Extra
1	id	int(11)			Não	*None*	AUTO_INCREMENT
2	idJamendo	int(11)			Não	*None*	
3	url	varchar(100)	latin1_swedish_ci		Não	*None*	
4	title	varchar(80)	latin1_swedish_ci		Não	*None*	
5	artistName	varchar(80)	latin1_swedish_ci		Não	*None*	
6	albumName	varchar(80)	latin1_swedish_ci		Não	*None*	
7	albumImage	varchar(150)	latin1_swedish_ci		Não	*None*	

Tabela 4 - Tabela Tracks

Esta tabela é composta pelos seguintes campos:

- **id** – identificador da música no sistema
- **idJamendo** – identificador da música no serviço Jamendo
- **url** – *url* do ficheiro de música no serviço Jamendo
- **title** – título da música
- **artistName** – nome do artista
- **albumName** – nome do album
- **albumImage** – *url* com a imagem do album

À exceção do campo *id*, todos os outros campos são preenchidos com dados oriundos do serviço de música Jamendo. Este serviço foi escolhido para fornecer os *items* disponibilizados aos utilizadores deste sistema pois as músicas são gratuitas e de distribuição livre, possui uma API para acesso aos metadados de cada música e permite *streaming* por aplicações, sistema e serviços de terceiros.

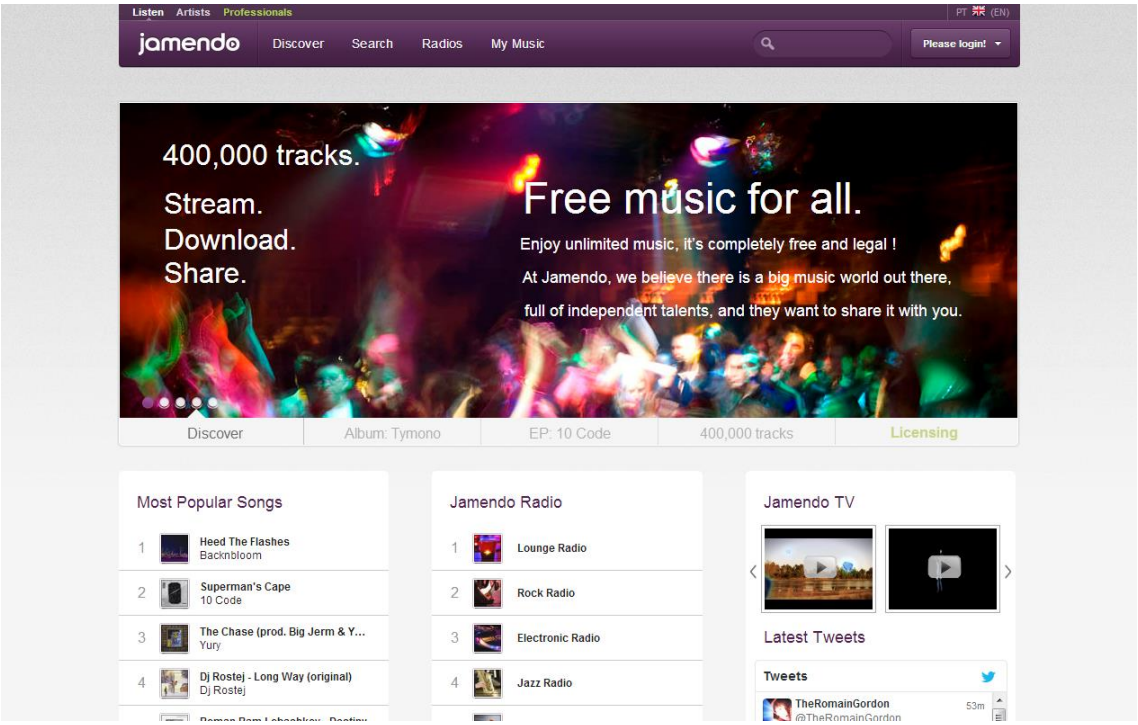

Figura 12- Página inicial do Jamendo

O acesso à API do Jamendo é feito através de pedidos HTTP GET, e podem ser consultados metadados de Álbuns, Artistas, Faixas, entre outros (http://developer.jamendo.com/v3.0#readmethods). Em cada pedido pode ser passada uma extensa série de parâmetros, que definem ou condicionam os dados a serem retornados. Por exemplo, alguns desses parâmetros para um pedido de metadados de uma determinada faixa são:

- **format** – o formato da resposta (xml ou json)
- **order** – ordena os resultados por número de *downloads*, popularidade, nome de artista, entre outros
- **namesearch** – nome da faixa a pesquisar
- **artist_name** – nome do artista
- **imagesize** – tamanho da imagem do álbum a retornar
- **audioformat** – formato do ficheiro de áudio na *url* retornada (mp3, ogg ou flac)
- **vocalinstrumental** – define se a faixa pretendida é cantada ou se é instrumental
- **speed** – define a velocidade da faixa pretendida (verylow, low, medium, high, veryhigh)
- **limit** – número de resultados a devolver

Por exemplo, se quisermos pesquisar uma música que tenha no nome a expressão "rock" e que tenha um ritmo muito acelerado, poderia ser feito um pedido semelhante ao pedido HTTP GET à url http://api.jamendo.com/v3.0/tracks/?client_id=b6747d04&format=json&limit=1&namesearch=rock&speed=veryhigh. O resultado deste pedido é apresentado na figura 13.


```
{
  - headers: {
        status: "success",
        code: 0,
        error_message: "",
        warnings: "",
        results_count: 1
    },
  - results: [
      - {
            album_id: "21357",
            duration: 244,
            id: "148954",
            artist_id: "338761",
            artist_name: "Houdini Roadshow",
            position: 2,
            album_name: "Down in the City",
            name: "Rocksteady",
            license_ccurl: "http://creativecommons.org/licenses/by-nc-nd/3.0/",
            releasedate: "2008-03-17",
            album_image: "http://imgjam.com/albums/s21/21357/covers/1.200.jpg",
            audio: "http://storage-new.newjamendo.com/download/track/148954/mp31/",
            prourl: "pro.jamendo.com/royalty-free-music-library#search=track-148954"
        }
    ]
}
```


Figura 13 - Exemplo de resposta da API do Jamendo

Dentro do *array results*, podemos ver um objecto ***json*** composto pelos diferentes metadados disponibilizados. São os campos ***id***, ***artist_name***, ***album_name***, ***name***, ***album_image*** e ***audio*** que vão popular a tabela de Faixas do sistema e que vão ser apresentados aos utilizadores. Na secção sobre a implementação da camada de interface é explicado como é que estes dados são apresentados e que outras interações existem com o serviço Jamendo.

4.2.3. RatingsContext

A tabela *RatingsContext* é responsável por armazenar todos os registos referentes aos *ratings* atribuídos pelos utilizadores às músicas e as informações de contexto em que estes se encontravam no momento dessa avaliação. São os dados desta tabela (nomeadamente os campos *user*, *item* e *rating*) que vão alimentar diretamente o motor de recomendações.

#	Coluna	Tipo	Collation	Atributos	Nulo	Padrão	Extra
1	id	int(11)			Não	*None*	AUTO_INCREMENT
2	time	time			Não	*None*	
3	day_week	int(11)			Não	*None*	
4	weather	int(11)			Sim	*NULL*	
5	latitude	double			Não	38.6580282	
6	longitude	double			Não	-9.0646414	
7	user	int(11)			Não	*None*	
8	item	int(11)			Não	*None*	
9	rating	decimal(11,1)			Não	*None*	
10	isRecommendation	int(11)			Sim	0	

Tabela 5 - RatingsContext

Esta tabela é composta pelos seguintes campos:

- **id** – identificador da avaliação
- **time** – hora do dia em que o *rating* foi atribuído, no formato HH:MM:SS
- **day_week** – dia da semana em que a avaliação foi feita
- **weather** - estado do tempo no momento da avaliação (dos macro-estados definidos anteriormente)
- **latitude** – latitude da posição do utilizador
- **longitude** – longitude da posição do utilizador
- **user** – identificador do utilizador no sistema
- **item** – identificador do *item* no sistema
- **rating** – *rating* numérico atribuído pelo utilizador ao *item*
- **isRecommendation** – indica se o *rating* foi atribuído a um *item* recomendado pelo sistema. Os valores possíveis são: 0 – *item* do catálogo; 1 – *item* recomendado pelo algoritmo de *collaborative filtering* tradicional; 2 – *item* recomendado pelo algoritmo baseado no contexto

Como é possível verificar, tanto a latitude como a longitude têm um valor pré definido, pois por vezes os utilizadores não aceitavam o acesso às coordenadas da sua posição geográfica. Decidiu-se atribuir um valor por defeito a estes campos de modo a que estes registos pudessem ser utilizados pelo algoritmo de recomendação baseado no contexto. A sua ausência poderia significar uma redução no número de registos que serve de base ao funcionamento deste algoritmo, podendo inclusivamente não ser possível recomendar qualquer música em certas situações.

4.3. Implementação – Módulo de Recomendação

O módulo de recomendação do sistema recomenda música ao utilizador em três ou quatro passos, dependendo do algoritmo utilizado. Como foi referido no capítulo anterior, as recomendações serão feitas alternadamente por cada algoritmo a cada pedido. Este processo está detalhado na figura seguinte.

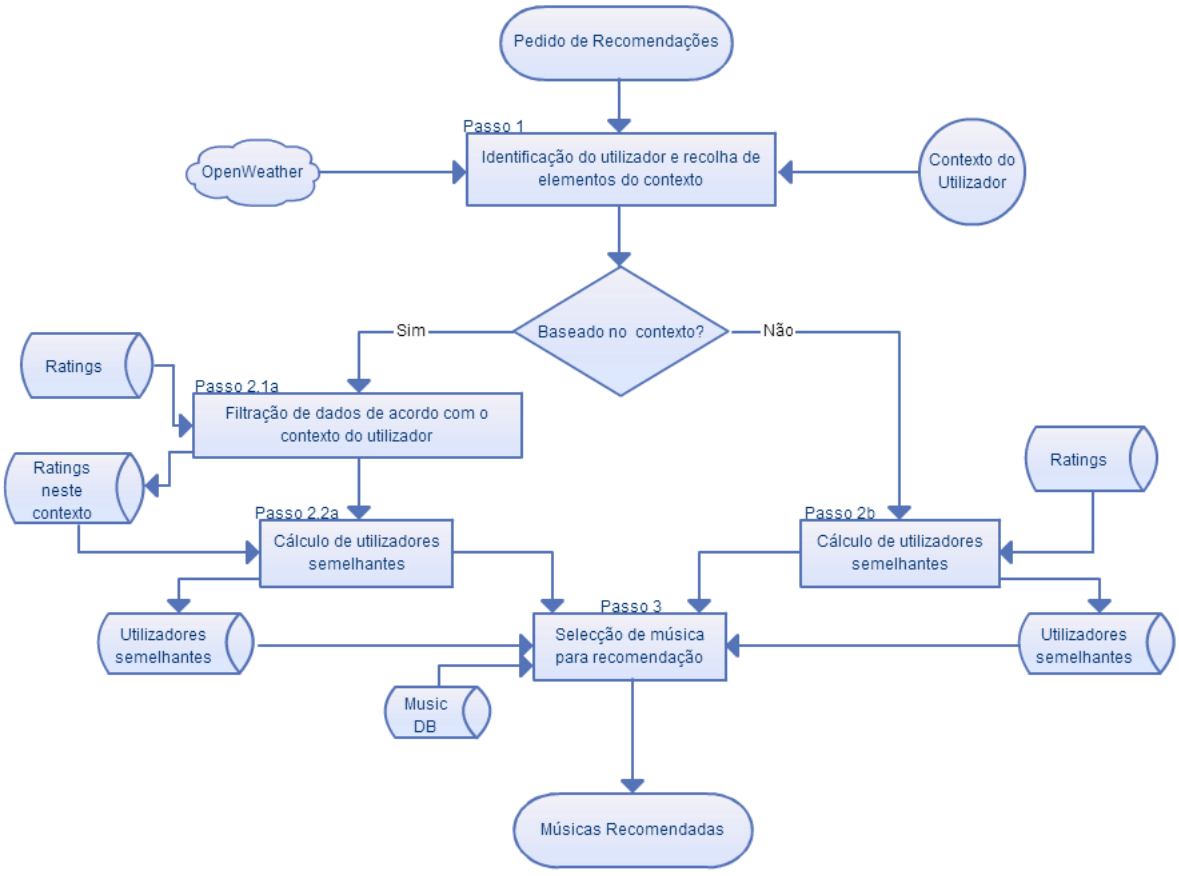

Figura 14 – Processo de Recomendação do sistema proposto

4.3.1. Passo 1

O processo inicia-se com um pedido de recomendações despoletado pelo utilizador na *Interface Layer*. É também aí que o utilizador é identificado (através de *login* na página com as suas credenciais do Facebook) e é recolhida a informação do seu contexto, nomeadamente a sua localização em termos de coordenadas geográficas, a hora e o dia da semana em que o pedido é feito (este processo vai ser detalhado na secção relativa à implementação da camada de interface).

4.3.1.1. OpenWeather

Seguidamente, as coordenadas geográficas são utilizadas para obter o estado do tempo que se faz sentir nessa localização. Para isso recorreu-se ao serviço *OpenWeather*[20]. O *OpenWeather* disponibiliza gratuitamente um *website* e API com dados e previsões meteorológicas adequadas para qualquer serviço *web* ou aplicação móvel. Fornece uma ampla gama de dados meteorológicos, como o clima atual numa dada localização, a previsão para toda a semana, precipitação, vento, nuvens, dados de estações meteorológicas, entre outros. Estes dados são recolhidos através de uma rede global de mais de 40.000 estações meteorológicas.

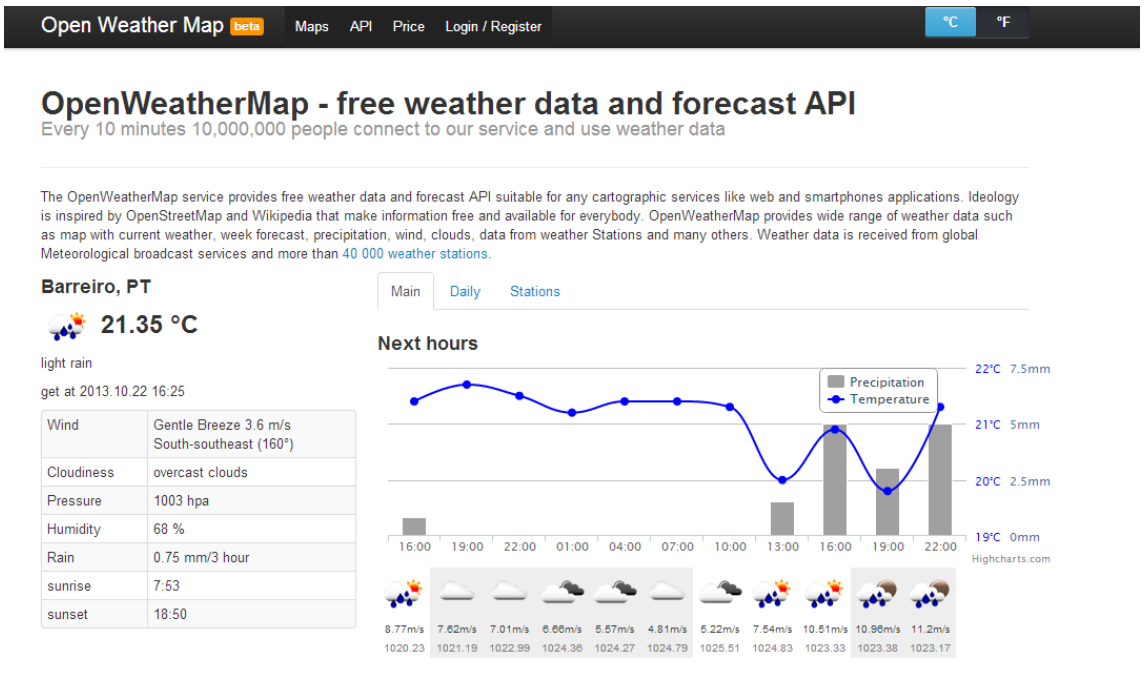

Figura 15 - Página inicial do OpenWeather

[20] www.openweathermap.org

A API do *OpenWeather* proporciona um grande conjunto de operações neste domínio, que podem ser agrupadas em quatro categorias: Dados meteorológicos actuais, Previsões a 5 e 14 dias, Pesquisa de locais e Mapas. Os pedidos são feitos via HTTP GET e os dados são devolvidos num dos formatos JSON, XML ou HTML.

4.3.1.1.1. Dados meteorológicos atuais

Com este tipo de operações podem ser obtidos dados meteorológicos para qualquer localização no planeta, e podem ser obtidos de três maneiras diferentes:

- Por nome de cidade
- Por coordenadas geográficas
- Pelo ID da cidade

Por exemplo, um pedido de dados meteorológicos para a localização com a latitude 38.75 e longitude -9.15 (Lisboa) pode ser feito com um pedido HTTP GET à URL http://api.openweathermap.org/data/2.5/weather?lat=38.75&lon=-9.15&units=metric.
Sãode destacar os parâmetros *lat* e *lon*, que se referem aos valores da latitude e longitude do local.

O parâmetro *units* com o valor *metric* significa que os dados são devolvidos no sistema métrico (por defeito são devolvidos no sistema imperial).

O *output* do pedido é apresentado na figura 16.

```
{
  - coord: {
        lon: -9.15,
        lat: 38.75
    },
  - sys: {
        country: "PT",
        sunrise: 1382424821,
        sunset: 1382464095
    },
  - weather: [
      - {
            id: 802,
            main: "Clouds",
            description: "scattered clouds",
            icon: "03d"
        }
    ],
    base: "global stations",
  - main: {
        temp: 20.36,
        pressure: 1004,
        humidity: 68,
        temp_min: 20,
        temp_max: 21
    },
  - wind: {
        speed: 3.1,
        deg: 180
    },
  - rain: {
        3h: 0
    },
  - clouds: {
        all: 36
    },
    dt: 1382459100,
    id: 2272005,
    name: "Alfornelos",
    cod: 200
}
```


Figura 16 - Output de um pedido de dados meteorológicos

Os dados obtidos dividem-se entre dados quantitativos, como a temperatura (*temp)*, temperatura máxima e mínima (*temp_min* e *temp_max*), velocidade do vento (*speed*), e dados qualitativos como os presentes no campo *weather*. Estes últimos são de especial interesse pois dão uma caracterização sobre o estado do tempo no local pedido. Neste caso é possível saber que o céu está enevoado nas coordenadas indicadas, através dos campos *main* e *description*. O campo *id* é referente ao identificador deste estado do tempo em particular na base de dados do *OpenWeather*. Na página http://bugs.openweathermap.org/projects/api/wiki/Weather_Condition_Codes são apresentados todos os estados do tempo possíveis e os seus respectivos identificadores.

Por defeito os dados são devolvidos no formato JSON (para outros formatos é necessário acrescentar o parâmetro *mode=xml* ou *mode=html*, para os formatos XML e HTML respectivamente).

4.3.1.1.2. Previsões a 5 e 14 dias

É possível receber previsões do estado do tempo para qualquer localização geográfica e não apenas para cidades. As previsões podem ser dadas para intervalos de três horas num período de cinco dias ou previsões diárias para os próximos catorze dias.

Tal como para os dados meteorológicos atuais, as previsões podem ser obtidas através do nome do uma cidade, de coordenadas geográficas ou do ID de uma cidade.

Pegando no exemplo anterior com as coordenadas 38.75 e -9.15, a obtenção de previsões é feita através de um pedido HTTP GET à URL http://api.openweathermap.org/data/2.5/forecast?lat=38.75&lon=-9.15.


```
      name: "Damaia",
    - coord: {
        lon: -9.21227,
        lat: 38.751411
      },
      country: "PT",
      population: 0
  },
  cnt: 41,
- list: [
    - {
        dt: 1382454000,
      - main: {
            temp: 292.62,
            temp_min: 292.62,
            temp_max: 293.287,
            pressure: 1020.23,
            sea_level: 1020.56,
            grnd_level: 1020.23,
            humidity: 98,
            temp_kf: -0.67
        },
      - weather: [
          - {
              id: 500,
              main: "Rain",
              description: "light rain",
              icon: "10d"
            }
        ],
      - clouds: {
            all: 92
        },
      - wind: {
            speed: 8.77,
            deg: 231.501
        },
      - rain: {
            3h: 0.75
        },
      - sys: {
            pod: "d"
        },
        dt_txt: "2013-10-22 15:00:00"
    },
    - {
```


Figura 17 - Output de um pedido de previsões meteorológicas

O *output* deste pedido é uma lista de respostas muito semelhante ao pedido de dados meteorológicos actuais, com a diferença de cada um dos elementos desta lista ter um campo *dt_txt* com a data e hora a que esse dados meteorológicos se aplicam.

4.3.1.1.3. Pesquisa de Locais

O *OpenWeather* pode também ser utilizado para procurar cidades por nome, país ou coordenadas geográficas.

A resposta ao pedido HTTP GET http://api.openweathermap.org/data/2.5/find?lat=57&lon=-2.15 devolve uma lista de cidades perto das coordenadas indicadas e os seus respectivos dados meteorológicos atuais.

4.3.1.1.4. Mapas

O último conjunto de funcionalidades do *OpenWeather* é referente a mapas meteorológicos com mapas de precipitação, nuvens, pressão, temperatura, vento e outros. Estes mapas podem ser aplicados livremente em *websites* ou aplicações móveis, e estão disponíveis em http://openweathermap.org/hugemaps.

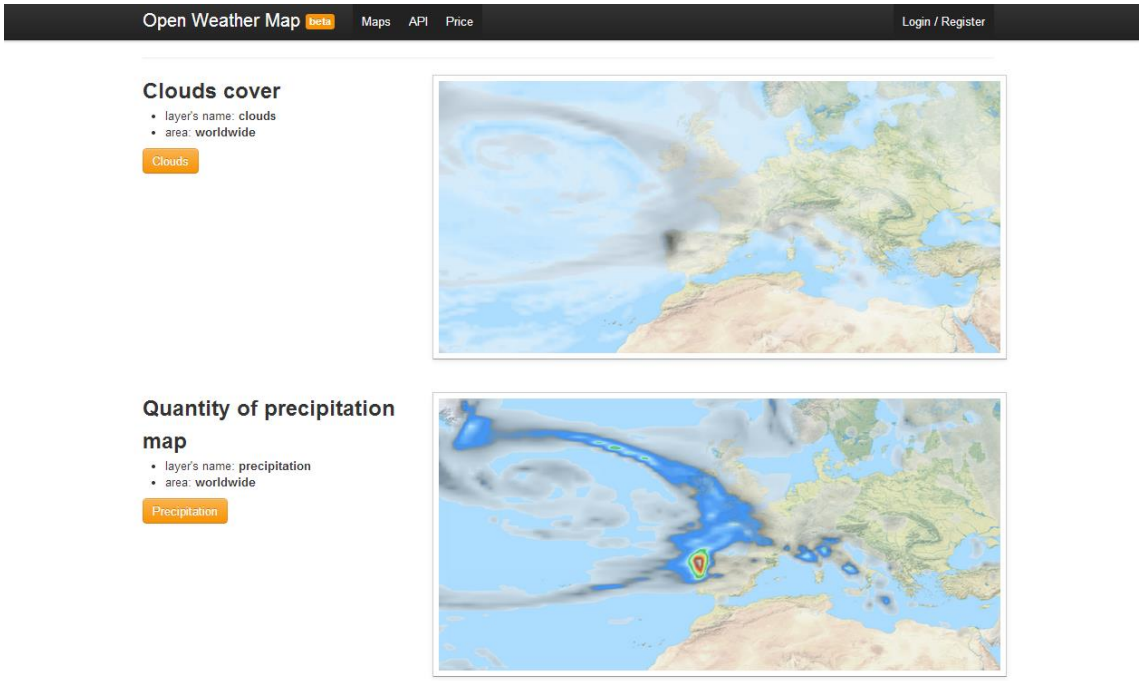

Figura 18 - Exemplos de mapas meteorológicos do OpenWeather

4.3.1.2. Utilização do OpenWeather

Como foi referido anteriormente, o sistema recorre à API do OpenWeather de modo a obter o estado do tempo para as coordenadas geográficas relativas à localização do utilizador. Para isso é feito um pedido HTTP GET de dados meteorológicos atuais similar ao exemplo, mas com as coordenadas do utilizador em causa. Este processo é executado na função *getWeather* do *servlet* do módulo de recomendação, e é apresentado na figura seguinte.

```
private int getWeather(double latitude, double longitude) {
    String app_id = "c1ed5750ce733fc72ba7894b9af583bb";
    String url = "http://api.openweathermap.org/data/2.5/weather?lat=" + latitude + "&lon="
                    + longitude + "&units=metric&APPID=" + app_id;
    int openWeatherMapCode = 0;
    try {
        URI uri = new URI(url);
        JSONTokener tokener = new JSONTokener(uri.toURL().openStream());
        JSONObject root = new JSONObject(tokener);
        JSONArray array = root.getJSONArray("weather");
        openWeatherMapCode = array.getJSONObject(0).getInt("id");
    } catch (Exception e){
        e.printStackTrace();
    }
    switch(openWeatherMapCode){
        case 800:
            return 1;
        case 801:
            return 2;
        case 802: case 803:
            return 3;
        case 804: case 701: case 711: case 721: case 731: case 741:
            return 4;
        case 500: case 501: case 300: case 301: case 302: case 310: case 321:
            return 5;
        case 502: case 503: case 504: case 520: case 521: case 522: case 311: case 312:
            return 6;
        case 200: case 201: case 202: case 210: case 211: case 212: case 221: case 230: case 231: case 232:
            return 7;
        case 600: case 601: case 602: case 611: case 621:
            return 8;
        default:
            return 1; // If code unknown, default to clear sky.
    }
```

Figura 19 - Método getWeather

Como é possível ver, é recolhido o valor do campo *id* do *array* JSON *weather* (de acordo com o formato da resposta da API na figura 16). É este valor que identifica o estado do tempo em que o utilizador se encontra.

No entanto este valor não pode ser usado diretamente na geração de recomendações. O número de valores possíveis que o *id* retornado pela API pode tomar (40) é muito elevado, de modo a refletir caracterizações do estado do tempo muito específicas.

Por exemplo, só para estados do tempo com precipitação (não incluindo queda de neve) existem 19 *ids* diferentes. Este facto iria dificultar a pré-filtragem de dados para o cálculo de recomendações e a qualidade das mesmas. Um exemplo disto seria a não inclusão dos *ratings* de outro utilizador no cálculo de recomendações mesmo que fossem muito semelhantes aos do utilizador alvo, se o primeiro os tivesse atribuído sob chuviscos (*id* 301) e o último sob chuva leve (*id* 500).

Para resolver este problema foi aplicada a técnica sugerida por (Adomavicius, Tuzhilin 2001) de agregação de valores em categorias que juntassem vários tipos de estados do tempo e assim formassem macro-estados claramente distintos. As categorias criadas são apresentadas na tabela seguinte.

1 – Céu Limpo	**2 – Algumas núvens**	**3 – Céu muito nublado**	**4 - Nevoeiro**	**5 - Chuva**	**6 – Chuva intensa**	**7 - Trovoada**	**8 - Neve**
800	801	802	804	500	502	200	600
		803	701	501	503	201	601
			711	300	504	202	602
			721	301	520	210	611
			731	302	521	211	621
			741	310	522	212	
				321	311	221	
					312	230	
						231	
						232	

Tabela 6 - Categorias de estados de tempo ou Macro-estados

Deste modo, os valores possíveis para o estado do tempo do utilizador ficam reduzidos a 8, assegurando uma menor escassez dos dados quando comparado com a alternativa de 40 valores. Este modelo foi organizado de maneira que as categorias sejam tão mais semelhantes quanto mais próximas estiverem numericamente. Por exemplo, as categorias 1 e 2 são mais semelhantes do que as 1 e 3.

4.3.1.3. Recolha de data e hora

A obtenção dos restantes dados do contexto do utilizador é feita na *Interface Layer* através de uma chamada ao relógio do sistema operativo do utilizador (via Javascript), e são enviados juntamente com o pedido de recomendações.

4.3.2. Passo 2

Recebido o pedido de recomendações com a informação contextual do utilizador e obtido o estado do tempo na sua localização, o sistema decide aleatoriamente se o algoritmo a utilizar inclui estes elementos contextuais ou se é executado o algoritmo de *collaborative filtering* tradicional.

```
if(random.nextBoolean()){
    standardRecommendation(request, response);
} else {
    contextRecommendation(request, response);
}
```

Figura 20 - Selecção aleatória do algoritmo de recomendação

4.3.2.1. Passo 2.1a

Caso seja selecionado o algoritmo de recomendação baseado no contexto, antes da geração de recomendações propriamente dita ocorre uma fase de filtragem dos *ratings* e preferências de todos os utilizadores no sistema de modo a que as recomendações sejam geradas apenas com base em *ratings* atribuídos em contextos semelhantes aos do utilizador (*context pre-filtering*). Tendo em conta que os dados se encontram numa base de dados MySQL, esse filtragem será feita através de uma *query* que devolva apenas os registos associados a contextos similares ao do utilizador.

Vamos assumir o seguinte contexto do utilizador:

- Hora – 15h32
- Dia da semana – Sexta-feira
- Estado do tempo – Céu limpo (*id* 1)

Uma abordagem possível seria utilizar estes valores diretamente na *query* SQL, ou seja, pedir só os *ratings* que tivessem sido atribuídos às 15h32 de uma Sexta-feira com céu limpo. No entanto esta abordagem é de uma especificidade desnecessária para o propósito deste sistema, principalmente em termos temporais, pois não se pretende que as recomendações sejam apenas para contextos exatamente iguais.

Acresce ainda o fato que, deste modo seria introduzido um grande nível de dispersão dos dados que dificultaria imenso a possibilidade de gerar recomendações.

A abordagem adoptada foi a criação de intervalos temporais para os elementos de contexto *Hora* e *Dia da semana* e agregações de categorias adjacentes para o *Estado do tempo*. Para a *Hora* da recomendação foi definido um intervalo de 4 horas centrado na hora do pedido de recomendação, ou seja, seriam pedidos todos os *ratings* que tivessem sido atribuídos entre as 13h32 e as 17h32.

```
int hour_min;
int hour_max;
// Defines the min and max hour limits
if(hour == 0) {
    hour_min = 22;
    hour_max = hour+2;
} else if(hour == 23) {
    hour_max = 1;
    hour_min = hour - 2;
} else {
    hour_min = hour - 2;
    hour_max = hour + 2;
}
```

Figura 21 - Determinação do intervalo temporal - horas

Para o *Dia da semana* foi definido um intervalo de 3 dias, resultando num intervalo de Quinta-feira a Sábado para o exemplo em questão.

```
int dayOfWeek_min;
int dayOfWeek_max;
if(dayOfWeek == 1){
    dayOfWeek_min = 7;
    dayOfWeek_max = 2;
} else if(dayOfWeek == 7){
    dayOfWeek_min = 6;
    dayOfWeek_max = 2;
} else {
    dayOfWeek_min = dayOfWeek - 1;
    dayOfWeek_max = dayOfWeek + 1;
}
```

Figura 22 - Determinação do intervalo temporal - dias da semana

Quanto ao *Estado do tempo*, foi criada uma agregação com as duas categorias numericamente adjacentes (excepto para as categorias 1 e 8, que se agregam apenas com as categorias 2 e 7, respectivamente).

```
int weather_code = getWeather(latitude, longitude);
int weather_min;
int weather_max;
if(weather_code == 1){
    weather_min = weather_code;
    weather_max = weather_code + 1;
} else if(weather_code == 8){
    weather_min = weather_code - 1;
    weather_max = weather_code;
} else {
    weather_min = weather_code - 1;
    weather_max = weather_code + 1;
}
```

Figura 23 - Determinação do intervalo de Macro-estados meteorológicos

Assim, é possível criar uma *query* de SELECT condicional que devolva um conjunto de registos restritos às condições definidas para serem utilizados na recomendação de músicas para o contexto do utilizador.

4.3.3. Passos 2.2a, 2b e 3

Embora os passos 2.2a e 2b sejam referentes ao algoritmo baseado no contexto e ao algoritmo de *collaborative filtering* tradicional respectivamente, o algoritmo de ambos nesta fase é o mesmo. A única diferença é a utilização de uma base de *ratings* filtrada para o algoritmo baseado no contexto versus a utilização da base de todos os *ratings* no algoritmo de *collaborative filtering* tradicional. Como foi referido no capítulo dos Conceitos e Trabalho Relacionado, é possível desenvolver um sistema de recomendação baseado no contexto utilizando um qualquer algoritmo de recomendação com uma base de *ratings* contextualizada. Neste caso é utilizado um algoritmo de *collaborative filtering*.

Os passos 2.2a-2b e 3 estão separados conceptualmente pois são duas fases distintas de um algoritmo de *collaborative filtering*. No entanto, na implementação do sistema são ambos executados internamente pela *framework* Mahout. Esta *framework* abstrai o cálculo das recomendações de modo a não ser necessário implementar de raiz um algoritmo já largamente estudado como o *collaborative filtering*, sendo apenas necessário passar-lhe a base de *ratings* que vai ser utilizada e configurar alguns parâmetros.

4.3.3.1. Implementação interna do Mahout

O Mahout é uma biblioteca *open source* de *Machine Learning* da Apache Foundation que implementa algoritmos de recomendação (*collaborative filtering*), *clustering* e de classificação. É uma biblioteca Java com o objectivo de ser utilizada e adaptada por programadores, e por isso não disponibiliza uma interface de utilizador ou um programa de instalação. No entanto, o módulo de recomendações do Mahout não está restrito a programas escritos em Java – pode ser executado como servidor externo que expõe a sua lógica de recomendação via *web services* ou HTTP. Nesta dissertação só será utilizado este módulo.

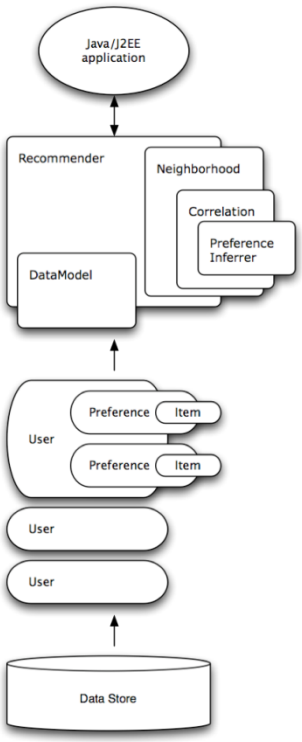

Figura 24 - Arquitectura do Mahout

Os componentes principais do módulo de recomendação do Mahout são apresentados em seguida.

4.3.3.1.1. DataModel

O DataModel é o objecto que compõe a informação relativa às preferências dos utilizadores e que é utilizado como *input* do Recommender. Cada preferência consiste num ID de utilizador, ID de *item* e um valor que expresse a força da preferência desse utilizador pelo *item* (*rating*). No Mahout os ID's são sempre valores inteiros.

O valor do *rating* pode ser qualquer número desde que valores maiores signifiquem preferências positivas mais fortes. Estes valores podem por exemplo estar na escala de 1 a 5.

O DataModel tem implementações diferentes para métodos de aquisição das preferências diferentes. Pode ser utilizado, por exemplo, o MySQLJDBCDataModel para construção da base de *ratings* a partir de uma base de dados MySQL ou o FileDataModel para carregar os dados a partir de um ficheiro. Na figura 25 é apresentado o conteúdo de um ficheiro de texto com uma lista de preferências. Para poder ser lido pelo Mahout, o ficheiro deve estar no formato CSV e com a estrutura *userID, itemID, rating*.

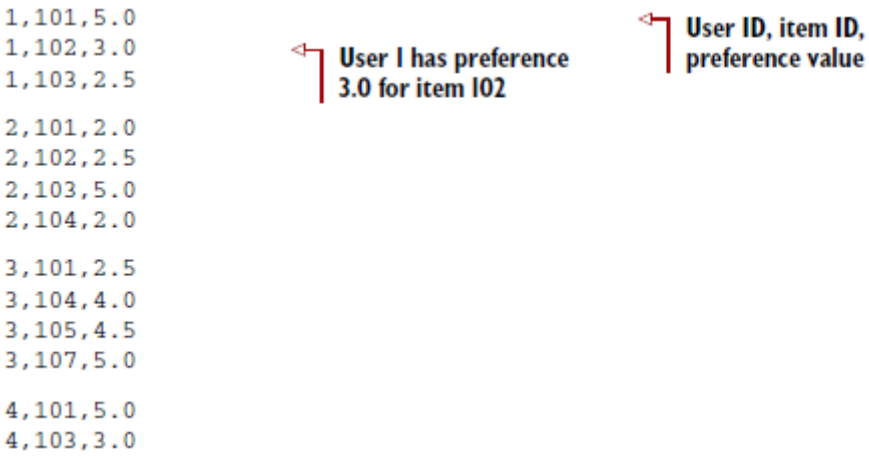

Figura 25 - Exemplo de base de preferências (Owen 2012)

4.3.3.1.2. UserSimilarity

A classe UserSimilarity é essencial para um motor de recomendação, pois define uma noção de semelhança entre dois utilizadores. Isto pode ser feito com recurso a várias métricas, também abstraídas pelo Mahout. As principais são a *PearsonCorrelationSimilarity* e a *EuclideanDistanceSimilarity*, cada uma com as suas vantagens e limitações.

Esta classe está intimamente ligada ao componente *UserNeighbourhood*, pois para se definir uma "vizinhança" de utilizadores semelhantes é necessário saber como é que essa semelhança é medida.

4.3.3.1.3. UserNeighbourhood

Num sistema de recomendação por *collaborative filtering*, as recomendações são produzidas através da descoberta de uma "vizinhança" de utilizadores com gostos semelhantes a um dado utilizador. É este componente que define essa noção de grupo – por exemplo, os 10 utilizadores mais semelhantes. O Mahout oferece dois modos de definir esse grupo: com um tamanho fixo ou baseado num determinado limiar de semelhança.

A abordagem de grupo de tamanho fixo (*NearestNUserNeighbourhood*) define um número de utilizadores mais semelhantes que serão utilizados para gerar recomendações. Este número é geralmente arbitrário e deve ser determinado caso a caso por experimentação, pois depende muito do número de utilizadores presentes no sistema, da métrica de semelhança e de outras configurações próprias de cada sistema. Normalmente um número pequeno significa que as recomendações vão ser geradas a partir de utilizadores muito semelhantes ao utilizador-alvo, o que pode resultar em recomendações de *items* que o utilizador já conhece (embora não tenha ainda avaliado).

Um número muito grande para o tamanho deste grupo geralmente significa que vão ser incluídos no cálculo das recomendações gostos de utilizadores pouco semelhantes ao utilizador-alvo. Isto também pode ser prejudicial pois podem-lhe ser recomendados *items* com pouco interesse (Owen 2012). É portanto necessário encontrar um equilíbrio que seja adequado ao sistema em causa.

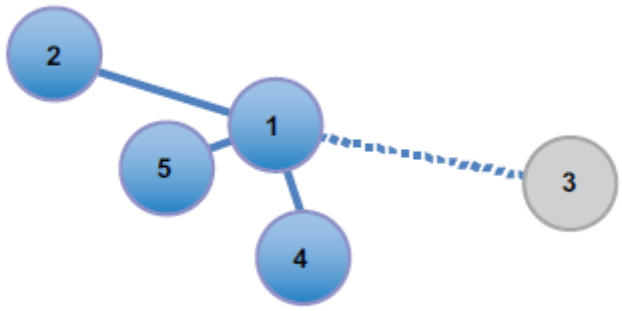

Figura 26 - Exemplo conceptual de Fixed-Sized Neighbourhood (Mahout in Action.pdf)

A abordagem baseada num limiar de semelhança (*threshold*) não define o número de utilizadores semelhantes que vão ser utilizados para calcular as recomendações, mas sim o valor mínimo da métrica de semelhança entre cada utilizador e o utilizador-alvo necessário para pertencer à "vizinhança".

Deste modo é possível assegurar que todos os utilizadores considerados para o cálculo da recomendação têm um mínimo de semelhança. A desvantagem desta abordagem é que com limiares demasiado altos pode não ser possível agrupar um número de utilizadores suficientes para gerar boas recomendações, sendo necessário baixar esse valor. Este facto é especialmente problemático em sistemas com poucos utilizadores. Tal como na abordagem de grupo de tamanho fixo, é necessário optimizar este valor iterativamente através de experimentação em cada sistema.

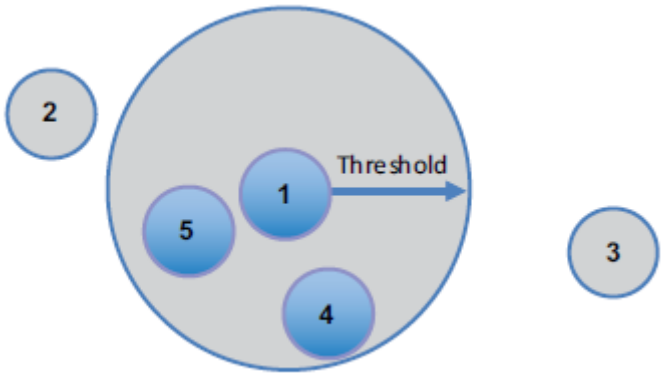

Figura 27 - Exemplo conceptual de Threshold-Based Neighbourhood (Owen 2012)

4.3.3.1.4. Recommender

O *Recommender* é o componente principal do Mahout e do motor das recomendações, onde são reunidos os outros três componentes principais e são produzidas recomendações de *items*. As suas principais implementações são *GenericUserBasedRecommender* e *GenericItemBasedRecommender*, respectivamente para recomendações *user-to-user* ou *item-to-item*.

A lógica do cálculo de recomendações efectuado pelo *Recommender* é a seguinte (Owen 2012):

> *Para cada item **i** que o utilizador **u** não atribuiu rating*
>
> *Para cada outro utilizador **v** que atribuiu rating a **i***
>
> *Calcular a semelhança **s** entre os utilizadores **u** e **v***
>
> *Incluir o rating de **v** atribuído a **i**, ponderado por **s**, numa média móvel*
>
> *Devolver os top items, ordenados por média ponderada*

Um exemplo da utilização concreta do Mahout com todos os componentes analisados é apresentado na figura 28.

```
DataModel model = new FileDataModel(new File("intro.csv"));
UserSimilarity similarity = new PearsonCorrelationSimilarity (model);
UserNeighborhood neighborhood =
  new NearestNUserNeighborhood (2, similarity, model);
Recommender recommender =
  new GenericUserBasedRecommender(model, neighborhood, similarity);
```

Figura 28 - Exemplo de utilização do Mahout (Owen 2012)

Na primeira linha podemos ver a criação do DataModel, em que a base de preferências é carregada a partir do ficheiro **intro.csv**. Seguidamente é definida a métrica de semelhança (correlação de Pearson) para o DataModel e é definida uma vizinhança de tamanho fixo com os 2 utilizadores mais semelhantes. Por fim, é construído um motor de recomendação *user-to-user* a partir de todos os componentes definidos anteriormente.

4.3.3.2. Implementação do módulo de Recomendação

Como foi referido anteriormente, o módulo de Recomendação do sistema foi implementado com recurso ao Mahout. Na figura 29 é apresentada a implementação da geração de recomendações, dado um DataModel composto com preferências de utilizadores. É neste DataModel que está a diferença entre a recomendação *collaborative filtering* tradicional e a baseada no contexto. O primeiro é composto pelas preferências de todos os utilizadores enquanto que o segundo é composto apenas pelas preferências registadas em contextos semelhantes ao do utilizador, como foi referido na secção do Passo 2.1a.

```
JSONArray items = new JSONArray();
try {
    UserSimilarity similarity = new EuclideanDistanceSimilarity(model);
    UserNeighborhood neighborhood = new NearestNUserNeighborhood (4, similarity, model);
    Recommender recommender = new GenericUserBasedRecommender (model, neighborhood, similarity);
    List<RecommendedItem> recommendations = recommender.recommend(user_to_recommend, 3);
    for (RecommendedItem recommendation : recommendations) {
        items.put(recommendation.getItemID());
    }
    if(items.length() < 1){
        standardRecommendation(request, response);
    }
} catch(Exception e){
    printWriter.println(e.getStackTrace());
}
```

Figura 29 - Geração de Recomendações

Para a métrica de semelhança entre utilizadores foi utilizada a distância Euclideana em detrimento da correlação da Pearson. Foi tomada esta decisão pois embora ambas as métricas tenham uma *performance* semelhante nas recomendações, a distância Euclideana tem uma melhor *performance* em sistemas com poucos utilizadores (Owen 2012).

Para a criação dos grupos de utilizadores semelhantes foi utilizada a abordagem de tamanho de grupo fixo (*NearestNUserNeighbourhood*), também devido à necessidade de obter recomendações mesmo que existam poucos utilizadores com uma semelhança elevada. Caso fosse utilizada a abordagem baseada num limiar de semelhança haveria o risco de não serem encontrados utilizadores com um grau de semelhança igual ou superior ao limite definido e, consequentemente, não ser possível gerar recomendações. Neste caso foi definido o tamanho máximo de 4 utilizadores semelhantes pois foi o que revelou melhor *performance*.

Seguidamente é criado o motor de recomendação e são pedidas três recomendações através do método *recommend* do motor de recomendação, passando-lhe dois valores como argumentos: o *id* do utilizador para o qual se pretende obter recomendações e o número de recomendações pedidas.

O excerto de código da figura 29 em particular é executado para gerar recomendações baseadas no contexto. Visto que o sistema foi utilizado por poucos utilizadores (comparativamente com um sistema de larga escala com dezenas ou centenas de milhar), em algumas situações poderia não ser possível recomendar músicas a utilizadores em contextos muito específicos ou particulares. Nos casos em que isso acontece é executado o algoritmo de *collaborative filtering* tradicional, de modo a aumentar significativamente a possibilidade de serem recomendadas músicas e a experiência de utilização do sistema não ser afectada.

4.4. Implementação – Camada de Interface

A camada de interface é a face do sistema visível para o utilizador. É aqui que o utilizador realiza as tarefas de navegação pelo catálogo de músicas e audição das mesmas, avaliação de músicas e pedidos de recomendação. Esta interface é um *website* alimentado pelo servidor *web* e pelo módulo de recomendação, e está disponível em http://musicdiscovery.pt. Nesta página os utilizadores podem ouvir e avaliar músicas provenientes do catálogo de música gratuita Jamendo, numa escala de um a cinco. É também disponibilizada a funcionalidade de recomendação de música, que utiliza os *ratings* atribuídos por todos os utilizadores da página e recorre ao sistema de recomendação implementado para a geração de recomendações de música individualizadas.

4.4.1. Website – musicdiscovery.pt

O *website* foi desenhado tendo em conta o design e funcionalidades de outros *websites* de *streaming* de música, de modo a reduzir a barreira à utilização da página por parte dos utilizadores.

Para o seu desenvolvimento foram utilizadas várias tecnologias. O *backend* foi construído com a linguagem de programação PHP com recurso ao SGBD MySQL, como referido anteriormente. Para o *frontend*, foram utilizadas as tecnologias *web* HTML5+CSS+Javascript e a *framework* Bootstrap.

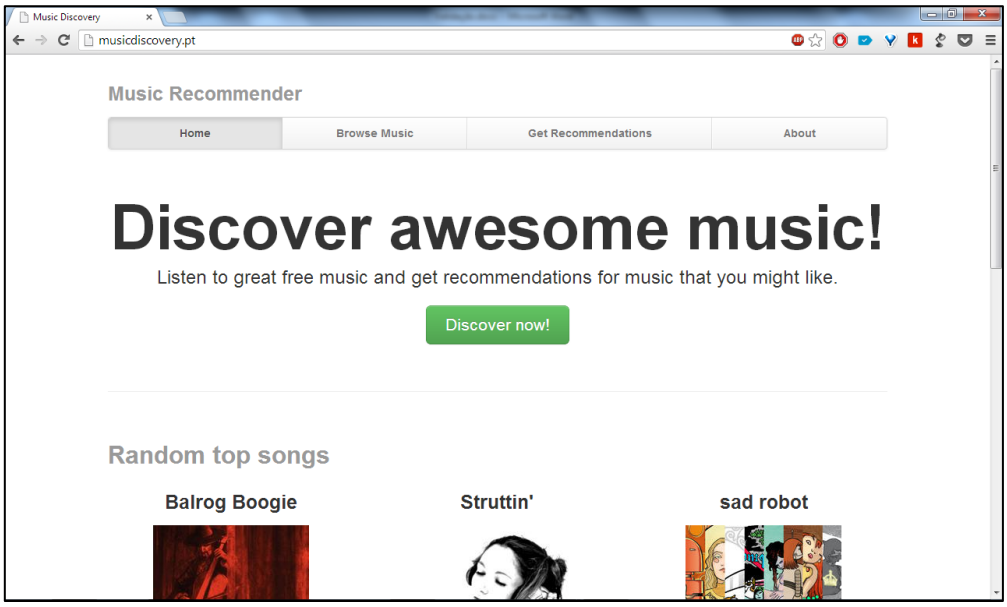

Figura 30 – Página Inicial

Na página de entrada é descrito o propósito do web*site* e são apresentadas algumas músicas ao utilizador, de modo a gerar interesse sobre as suas potencialidades.

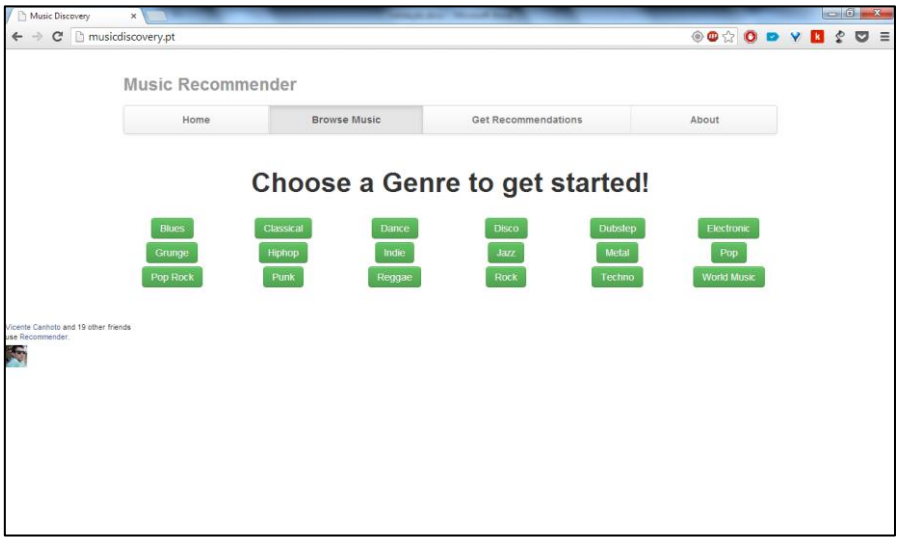

Figura 31 – Escolha de géneros musicais

Na página "*Browse Genres*" são apresentados todos os géneros de música presentes na plataforma ao utilizador. É-lhe pedido que escolha o género de música que pretende explorar e que faça *login* com o Facebook (foi utilizado o *plugin* oficial do Facebook disponível em https://developers.facebook.com/docs/facebook-login/).

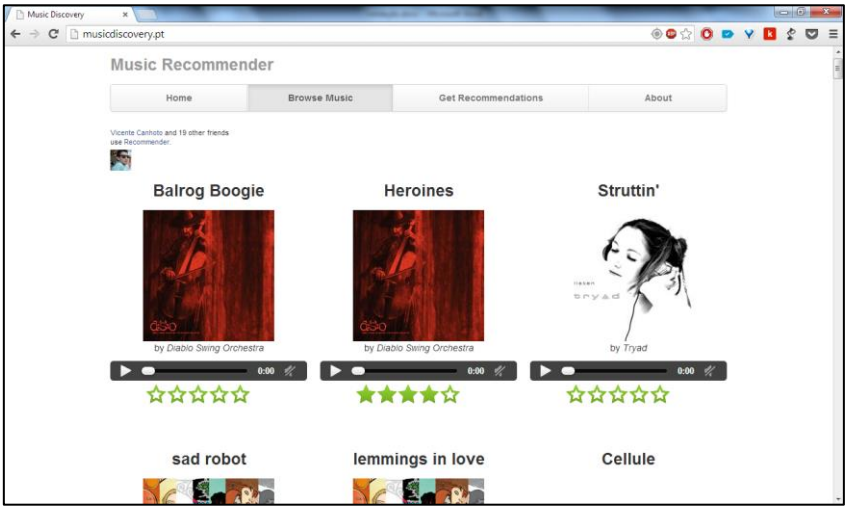

Figura 32 - Página de Género

Após escolhido o género, são listadas todas as músicas do género escolhido e é possível ouvir cada uma delas individualmente através do *player* embutido na página e avaliar cada uma através de um sistema de estrelas. É também pedido ao utilizador que aceite partilhar a sua localização geográfica com o *website*. Isso é feito com o código Javascript da figura 33.

```
<script type="text/javascript">
var latitude;
var longitude;
function getLocation()
  {
  if (navigator.geolocation)
    {
    navigator.geolocation.getCurrentPosition(showPosition);
    }
  else{alert("Geolocation is not supported by this browser.");}
  }
function showPosition(position){
  // We have to check if the user accepted to share the location!!!!!
  latitude = position.coords.latitude;
  longitude = position.coords.longitude;
  }
</script>
```

Figura 33 - Algoritmo de recolha das coordenadas geográficas

Na página "*Get Recommendations*" são listadas as recomendações individualizadas de música geradas pelo sistema (figura 34). Tal como acontece com o catálogo de músicas, é dada a possibilidade ao utilizador de avaliar as músicas recomendadas.

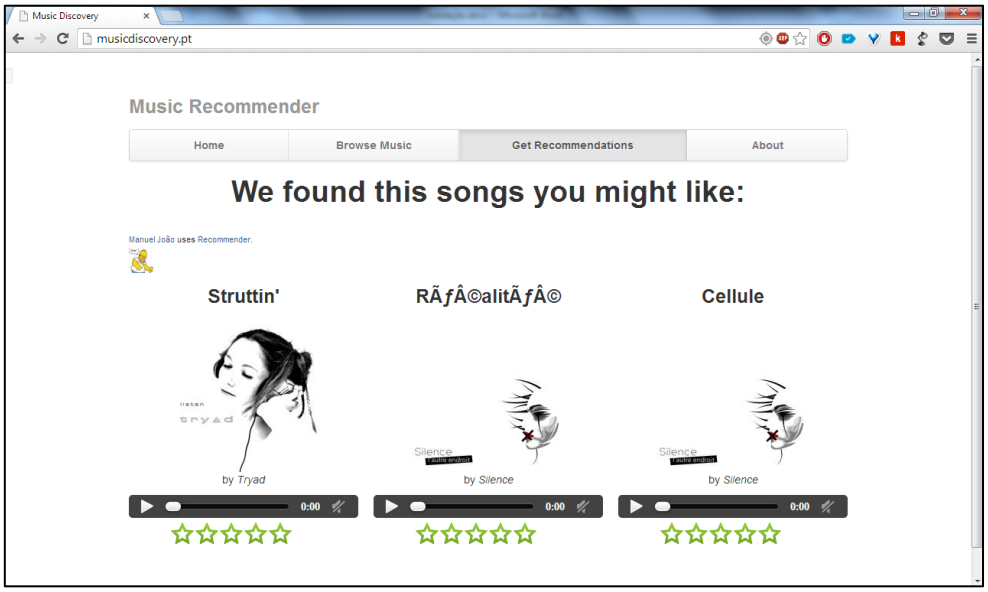

Figura 34 – Página de Recomendações

Caso não seja possível fazer nenhuma recomendação é pedido ao utilizador que avalie um determinado conjunto de músicas de modo a aumentar a possibilidade de gerar recomendações futuras (figura 35).

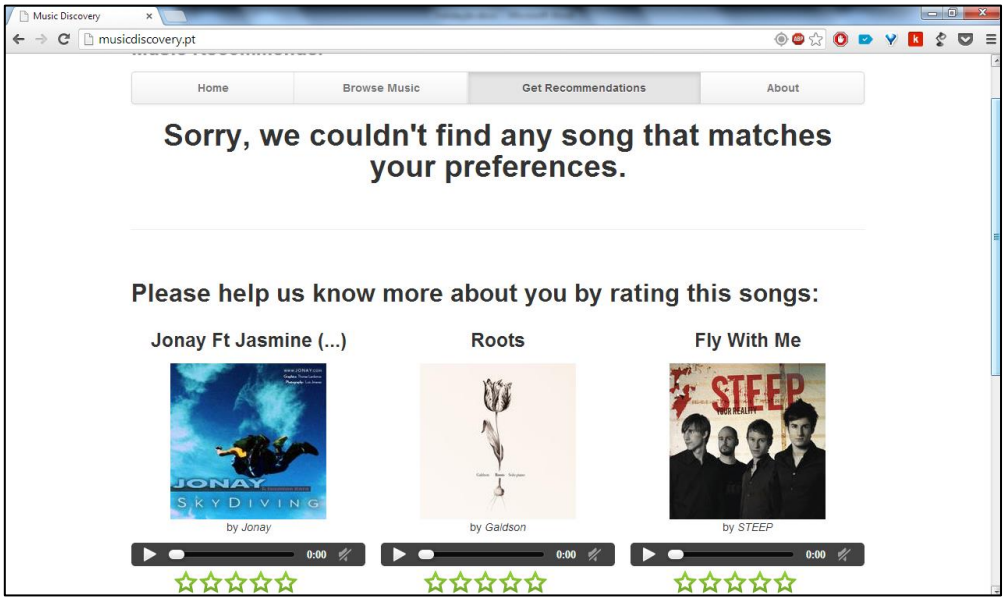

Figura 35 – Página de Recomendações(2)

5. Validação e Resultados obtidos

Para validar o sistema desenvolvido e verificar a veracidade da hipótese desta dissertação foi efectuada uma experiência de utilização real do sistema, com o objectivo de comparar a satisfação dos utilizadores que receberam recomendações de cada um dos algoritmos. Deste modo foi possível aferir se a introdução de determinada informação contextual melhorou a satisfação dos utilizadores em relação à música recomendada. Para isso foi desenvolvida uma metodologia de avaliação que teve por base os métodos apresentados na secção de avaliação de sistemas de recomendação no capítulo "Conceitos e Trabalho Relacionado".

5.1. Metodologia de validação

O ambiente de avaliação consiste numa aplicação *online* real utilizada por uma comunidade de utilizadores, em que qualquer utilizador poderia pedir recomendações de música e atribuir-lhes *ratings* de acordo com as suas preferências. Decidiu-se realizar um estudo de campo *online* devido às características da análise em causa, em que se pretende apreender a satisfação do utilizador. Também porque num estudo laboratorial os resultados da satisfação poderiam não ser os reais, visto que os utilizadores não estariam a utilizar o sistema para realizar uma tarefa de livre e espontânea vontade. Para além disso, para realizar um estudo controlado seria necessário arranjar um número significativo de pessoas para um espectro alargado de diferentes configurações reais de contexto (várias horas do dia, todos os dias da semana, diferentes condições meteorológicas). Num estudo de campo deixamos o sistema aberto 24h durante vários dias, com o intuito de simular mais fielmente o comportamento dos utilizadores.

A aplicação é servida por dois algoritmos de recomendação concorrentes. De modo a ser possível medir uma medida relativa de satisfação do utilizador é necessário registar as interações com cada um dos motores de recomendação (sem que os utilizadores saibam com qual motor estão a interagir). Através da comparação dessas interações, traduzidas em *ratings* atribuídos, será possível afirmar qual estratégia teve um melhor desempenho. Essas interações consistem em *ratings*, e são atribuídos numa escala de 1.0 (*Não gosto*) a 5.0 (*Gosto muito*).

Para ser considerado nesta avaliação, cada utilizador teve de realizar as seguintes tarefas:

a) Efectuar *login* na aplicação
b) Ouvir e avaliar algumas músicas do catálogo disponibilizado
c) Solicitar recomendações de músicas
d) Avaliar as músicas recomendadas

Caso tenha passado à tarefa c) sem ter realizado a b), o sistema solicita que avalie um conjunto de músicas antes de poder prosseguir.

No final do período em que a aplicação esteve disponível, os dados foram compilados e foram calculados os valores médios dos *ratings* atribuídos às músicas recomendadas por cada um dos algoritmos. De acordo com (Sinha, Swearingen 2001), estes *ratings* foram considerados como a satisfação dos utilizadores em relação às músicas recomendadas. Para além disso foram também recolhidos alguns dados demográficos de cada utilizador a partir do seu *id* do Facebook e da sua informação pública nesse *website*.

A metodologia proposta pode ser vista como uma competição entre duas abordagens diferentes à resolução de um mesmo problema (neste caso, a melhor satisfação do utilizador), em que o vencedor é definido pela forma como os utilizadores fazem uso das recomendações. Para assegurar uma distribuição equitativa de oportunidades de recomendação para ambas as abordagens, foi realizado um *A/B testing* (Kohavi, Longbotham, Sommerfield, Henne 2008) em que 50% dos pedidos de recomendação foram servidos pelo motor de *collaborative filtering* tradicional e os outros 50% pelo motor de recomendação baseado no contexto.

Esta metodologia está de acordo com a metodologia proposta por (Hayes, Cunningham 2002) para a avaliação comparativa de dois algoritmos de recomendação distintos.

5.2. Resultados obtidos

A aplicação esteve disponível desde a primeira semana de Outubro de 2013. Desde então, foram atribuídos 386 *ratings* a 132 músicas por 82 participantes (com uma média de 5 *ratings* atribuídos por participante). Desses 82 participantes, 57 eram do sexo masculino (70%), 20 eram do sexo feminino (24%) e não foi possível determinar o sexo de 5 deles (6%). Em relação às idades, a maioria dos participantes tinha entre os 19 e 35 anos de idade.

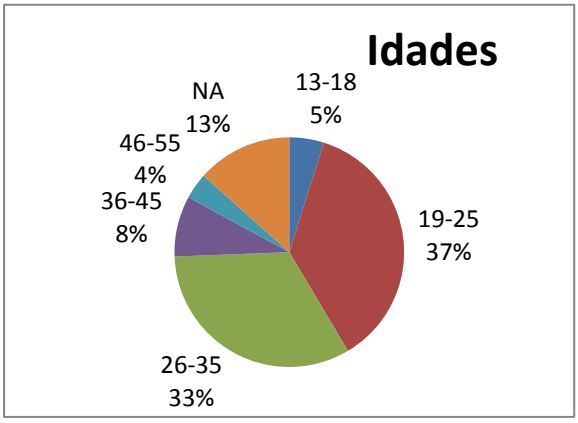

Figura 37 - Distribuição das idades dos participantes

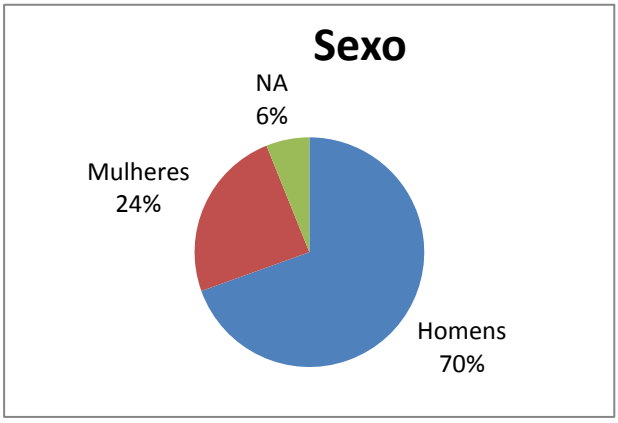

Figura 36 - Distribuição do sexo dos participantes

Dos 386 *ratings*, 62 foram atribuídos a músicas recomendadas pelo sistema, com uma distribuição de 50% para as recomendações de cada algoritmo. Na figura 38 é apresentado um gráfico de frequências destes *ratings*.

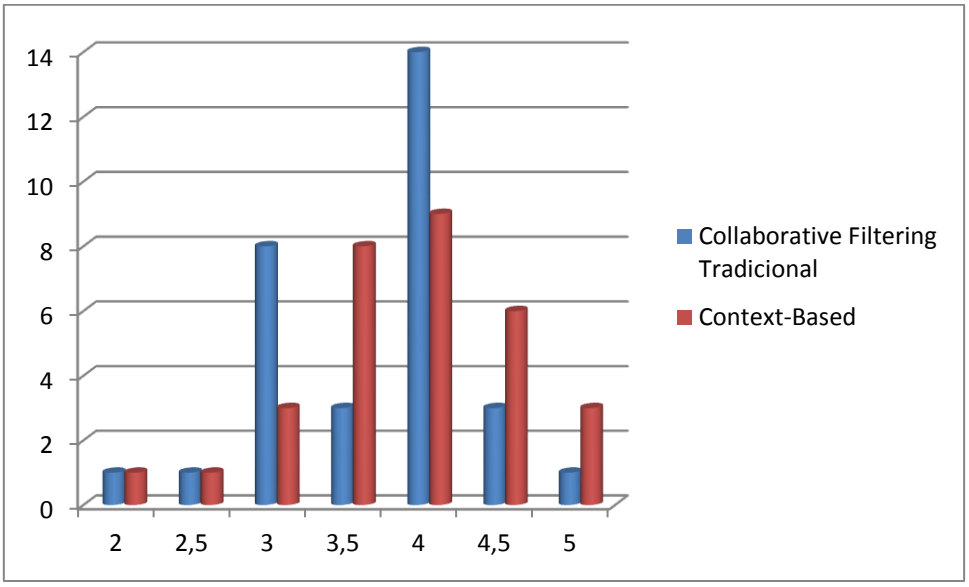

Figura 38 - Gráfico de frequências dos ratings atribuídos

Neste gráfico assistimos a uma predominância de músicas com *rating* de 3 ou de 4 para o algoritmo de *collaborative filtering* tradicional, enquanto que o algoritmo baseado no contexto assume a forma de uma distribuição Normal de mediana 4. É também possível observar que a maioria das músicas com *rating* 4,5 ou 5 foram recomendadas pelo algoritmo baseado no contexto.

Os dados recolhidos referem-se a interações com o sistema a partir do momento em que este passou a dispor das preferências de utilizadores suficientes para gerar recomendações. Uma vez que os sistemas de recomendação por *collaborative filtering* são mais eficientes quanto mais utilizadores e mais interações com os *items* existem, achou-se relevante a progressão dos *ratings* atribuídos às recomendações de ambos os algoritmos ao longo do tempo (figura 39).

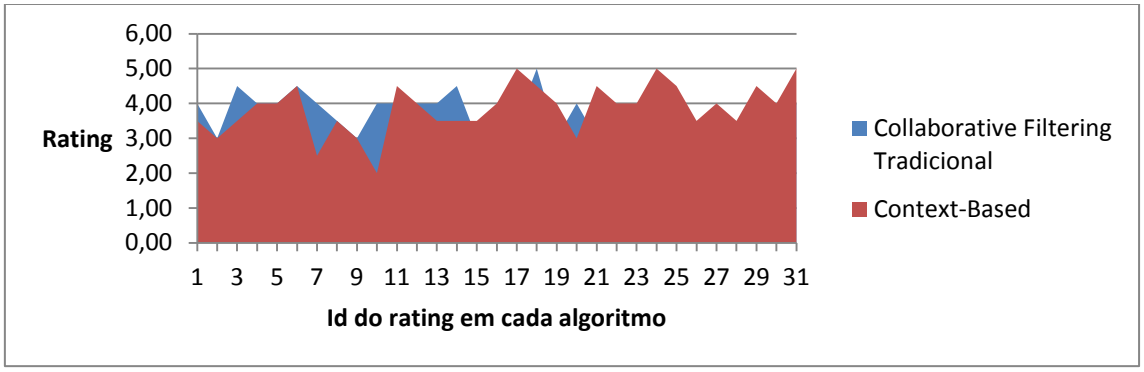

Figura 39 - Evolução dos *ratings* atribuídos

É possível observar que no início da utilização do sistema, o algoritmo de recomendação por *collaborative filtering* tradicional apresentou consistentemente recomendações que resultaram numa maior satisfação do utilizador. No entanto, à medida que o sistema foi sendo mais utilizado, a abordagem baseada no contexto superou as recomendações da abordagem anterior.

Para melhor aferir a satisfação média dos utilizadores com cada um dos algoritmos em causa, foi calculado o valor médio dos *ratings* para ambos (tabela 7).

Método	Média	Desvio-padrão
Recomendação Collaborative Filtering Tradicional	3,66	0,66
Recomendação Context-Based	3,85	0,68

Tabela 7 - Rating médio e Desvio-padrão

5.3. Discussão

Através dos resultados apresentados é possível concluir primeiramente que ambas as abordagens geraram recomendações consistentemente positivas. Tanto no gráfico de frequências, como no gráfico de progressão dos *ratings* e nos valores médios para ambos os algoritmos, constata-se que os *ratings* atribuídos às recomendações situaram-se maioritariamente no valor 3, o que configura um sentimento positivo numa escala de 1 a 5.

Quanto ao desempenho de ambos os algoritmos ao longo do tempo, a inferioridade do algoritmo baseado no contexto na fase inicial da experiência explica-se com o funcionamento do próprio algoritmo. O método *context pre-filtering* diminui a quantidade de preferências que alimentam o motor de recomendação, o que no início da utilização do sistema (com poucos utilizadores e poucos *ratings* atribuídos) se reflete num *input* pobre que prejudica a geração de recomendações satisfatórias.

No entanto, com o aumento dos dados disponíveis no sistema, a abordagem baseada no contexto é claramente superior à abordagem de *collaborative filtering* tradicional. O valor médio e o desvio-padrão dos *ratings* atribuídos a cada algoritmo confirma esta afirmação, com um aumento superior a 5% do valor médio atribuído.

Assim, é possível considerar que o algoritmo baseado no contexto teve um desempenho superior ao *collaborative filtering* tradicional nesta experiência. Confirma-se portanto que a introdução dos elementos de contexto hora, *dia da* semana e *estado do tempo* melhoraram a satisfação dos utilizadores neste sistema de recomendação de música, e que a hipótese colocada nesta dissertação é verdadeira. No entanto não é possível generalizar esta conclusão para todos os sistemas de recomendação de música e muito menos para os sistemas de recomendação em geral, sendo o reduzido número de utilizadores e *ratings* atribuídos nesta experiência uma das razões principais.

6. Conclusões

Esta dissertação centrou-se principalmente no estudo dos sistemas de recomendação, com ênfase na recomendação de música, e particularmente na comparação da satisfação obtida pelos utilizadores entre sistemas de recomendação por *collaborative filtering* e sistemas baseados no contexto. No entanto, o trabalho desenvolvido faz parte de um projeto de maior dimensão, com origem no trabalho realizado por (Ricardo, 2010) na indexação de música gratuita disponível na *web*, e em que está igualmente a ser avaliada a influência dos amigos das redes sociais na recomendação de música.

Foi realizada uma revisão da literatura que se focou nos três grandes métodos de recomendação atuais, tendo sido descrito o seu funcionamento, vantagens, desvantagens e problemas associados. Foram também revistos alguns métodos de avaliação de sistemas de recomendação, com especial ênfase para a avaliação de sistemas *online*. Para além disso foram também analisadas diversas *frameworks* de recomendação existentes que permitem implementar sistemas de recomendação sem ser necessário desenvolver de raiz alguns dos algoritmos mais estudados, tendo sido escolhida a *framework* Mahout para auxiliar o desenvolvimento da solução proposta.

Para responder ao problema identificado por esta dissertação e aferir a veracidade da hipótese que lhe está associada, foi proposto e implementado um sistema de recomendação de música *online* que integra os dois algoritmos de recomendação a ser comparados em regime de concorrência. Foi descrita a sua arquitectura e implementação, as tecnologias utilizadas e os serviços externos que contribuíram para o desenvolvimento.

Por fim foi realizada uma experiência de utilização do sistema num cenário real, por utilizadores reais que pretendiam espontaneamente obter recomendações de música de acordo com os seus gostos. Essa experiência permitiu obter dados sobre o desempenho de cada uma das abordagens de recomendação e, consequentemente, da satisfação dos utilizadores. Esses dados confirmaram a veracidade da Hipótese colocada, principalmente devido a um aumento superior a 5% do valor dos *ratings* atribuídos, o que demonstra que a utilização dos elementos de contexto *hora, dia da semana* e *estado do tempo* melhoraram a satisfação dos utilizadores. O sistema desenvolvido irá continuar disponível *online* com o objectivo de recolher uma maior quantidade de dados sobre a sua utilização, de modo a continuar a avaliação do mesmo.

Os resultados deste trabalho serão alvo de comunicação científica em conferências sobre este tópico de investigação.

6.1. Limitações

A principal limitação encontrada neste trabalho foi a utilização de um sistema de raiz sem dados prévios sobre utilizadores e as suas preferências. O reduzido número de utilizadores (82) e a reduzida utilização que deram ao sistema (em média menos de 5 *ratings* atribuídos por utilizador) limitou a obtenção de resultados mais fidedignos e que pudessem ser extrapolados para o universo de sistemas de recomendação de música.

Para a realização de uma experiência mais completa seria necessário a utilização do sistema por uma maior comunidade de utilizadores já estabelecida.

Outra limitação foi a dificuldade de instalação e configuração inicial da *framework* Mahout, apesar de ter sido considerada através da análise feita como a melhor para utilização neste sistema.

6.2. Trabalho futuro

Uma das formas de continuar o trabalho realizado nesta dissertação é a de colmatar as limitações encontradas, nomeadamente a criação de um estudo mais alargado que inclua um maior número de utilizadores e que esteja disponível para utilização durante um maior período de tempo, de modo a possibilitar a criação de uma comunidade mais ativa no sistema.

Outra maneira de contribuir para esta área com base no trabalho já desenvolvido é a experimentação com outros elementos do contexto do utilizador. Embora esta dissertação se tenha focado apenas em três elementos de contexto, na literatura desta área em particular dos sistemas de informação são referidos inúmeros elementos de contexto em diferentes dimensões. Deste a temperatura do ar à presença de amigos, a lista é extensa. A aferição individual da contribuição de cada um desses elementos para a satisfação dos utilizadores com os sistemas de recomendação seria também um contributo bastante importante, de modo a perceber que factores melhoram realmente essa satisfação. Essas conclusões irão potenciar os sistemas de recomendação futuros.

7. Bibliografia

ADOMAVICIUS, Gediminas, SANKARANARAYANAN, Ramesh, SEN, Shahana and TUZHILIN, Alexander, 2005. Incorporating contextual information in recommender systems using a multidimensional approach. In: *ACM Transactions on Information Systems (TOIS)*. 2005. Vol. 23, no. 1, pp. 103–145.

ADOMAVICIUS, Gediminas and TUZHILIN, Alexander, 2001. Multidimensional recommender systems: a data warehousing approach. In: Electronic commerce. S.l.: Springer. pp. 180–192.

ADOMAVICIUS, Gediminas and TUZHILIN, Alexander, 2005. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. In: *Knowledge and Data Engineering, IEEE Transactions on*. 2005. Vol. 17, no. 6, pp. 734–749.

ADOMAVICIUS, Gediminas and TUZHILIN, Alexander, 2011. Context-Aware Recommender Systems. In: RICCI, Francesco, ROKACH, Lior, SHAPIRA, Bracha and KANTOR, Paul B. (eds.), *Recommender Systems Handbook* [online]. S.l.: Springer US. pp. 217–253.

ANDERSON, Michelle, BALL, Marcel, BOLEY, Harold, GREENE, Stephen, HOWSE, Nancy, MCGRATH, S. and LEMIRE, Daniel, 2003. Racofi: A rule-applying collaborative filtering system. 2003.

BALABANOVIĆ, Marko and SHOHAM, Yoav, 1997. Fab: content-based, collaborative recommendation. In: *Communications of the ACM*. 1997. Vol. 40, no. 3, pp. 66–72.

BERNHARDSSON, Erik, 2009. *Implementing a scalable music recommender system*. S.l.: Skolan för datavetenskap och kommunikation, Kungliga Tekniska högskolan.

BREESE, John S., HECKERMAN, David and KADIE, Carl, 1998. Empirical analysis of predictive algorithms for collaborative filtering. In: *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*. S.l.: s.n. 1998. pp. 43–52.

BROZOVSKY, Lukas and PETRICEK, Vaclav, 2007. Recommender system for online dating service. 2007.

BURKE, Robin, 2007. Hybrid web recommender systems. In: *The adaptive web*. S.l.: Springer. pp. 377–408.

CANO, Pedro, KOPPENBERGER, Markus and WACK, Nicolas, 2005. Content-based music audio recommendation. In: *Proceedings of the 13th annual ACM international conference on Multimedia*. S.l.: s.n. 2005. pp. 211–212.

CELMA, Oscar, 2010. *Music Recommendation and Discovery*. S.l.: Springer.

CHIEN, Yung-Hsin and GEORGE, Edward I., 1999. A bayesian model for collaborative filtering. In: *Proceedings of the 7th International Workshop on Artificial Intelligence and Statistics*. S.l.: s.n. 1999.

CLAYPOOL, Mark, GOKHALE, Anuja, MIRANDA, Tim, MURNIKOV, Pavel, NETES, Dmitry and SARTIN, Matthew, 1999. Combining content-based and collaborative filters in an online newspaper. In: *Proceedings of ACM SIGIR workshop on recommender systems*. S.l.: s.n. 1999.

DEY, Anind K., 2001. Understanding and using context. In: *Personal and ubiquitous computing*. 2001. Vol. 5, no. 1, pp. 4–7.

GOLDBERG, David, NICHOLS, David, OKI, Brian M. and TERRY, Douglas, 1992. Using collaborative filtering to weave an information tapestry. In: *Communications of the ACM*. 1992. Vol. 35, no. 12, pp. 61–70.

GOLDBERG, Ken, ROEDER, Theresa, GUPTA, Dhruv and PERKINS, Chris, 2001. Eigentaste: A constant time collaborative filtering algorithm. In: *Information Retrieval*. 2001. Vol. 4, no. 2, pp. 133–151.

HAYES, Conor and CUNNINGHAM, Pádraig, 2002. An on-line evaluation framework for recommender systems. 2002.

HERLOCKER, Jonathan L., KONSTAN, Joseph A., TERVEEN, Loren G. and RIEDL, John T., 2004. Evaluating collaborative filtering recommender systems. In: *ACM Transactions on Information Systems (TOIS)*. 2004. Vol. 22, no. 1, pp. 5–53.

HEVNER, Alan R., MARCH, Salvatore T., PARK, Jinsoo and RAM, Sudha, 2004. Design science in information systems research. In: *MIS quarterly*. 2004. Vol. 28, no. 1, pp. 75–105.

HOROZOV, Tzvetan, NARASIMHAN, Nitya and VASUDEVAN, Venu, 2006. Using location for personalized POI recommendations in mobile environments. In: *Applications and the Internet, 2006. SAINT 2006. International Symposium on*. S.l.: s.n. 2006. pp. 6–pp.

KLEIN, Noreen M. and YADAV, Manjit S., 1989. Context effects on effort and accuracy in choice: An enquiry into adaptive decision making. In: *Journal of Consumer Research*. 1989. Vol. 15, no. 4, pp. 411–421. DOI 10.1086/209181.

KOHAVI, Ron, LONGBOTHAM, Roger, SOMMERFIELD, Dan and HENNE, Randal M., 2008. Controlled experiments on the web: survey and practical guide. In: *Data Mining and Knowledge Discovery*. 30 July 2008. Vol. 18, no. 1, pp. 140–181. DOI 10.1007/s10618-008-0114-1.

KONSTAN, Joseph A. and RIEDL, John, 1999. Research resources for recommender systems. In: *CHI'99 Workshop Interacting with Recommender Systems*. S.l.: s.n. 1999.

LEE, Jae Sik and LEE, Jin Chun, 2007. Context awareness by case-based reasoning in a music recommendation system. In: *Ubiquitous Computing Systems*. S.l.: Springer. pp. 45–58.

LINDEN, Greg, SMITH, Brent and YORK, Jeremy, 2003. Amazon. com recommendations: Item-to-item collaborative filtering. In: *Internet Computing, IEEE*. 2003. Vol. 7, no. 1, pp. 76–80.

LUCAS, André, 2010. Recomendação de programas de televisão. Lisboa: Universidade Técnica de Lisboa - Instituto Superior Técnico. Tese de Mestrado.

MAHMOOD, Tariq and RICCI, Francesco, 2009. Improving recommender systems with adaptive conversational strategies. In: *Proceedings of the 20th ACM conference on Hypertext and hypermedia*. S.l.: s.n. 2009. pp. 73–82.

MARCH, Salvatore T. and SMITH, Gerald F., 1995. Design and natural science research on information technology. In: *Decision Support Systems*. December 1995. Vol. 15, no. 4, pp. 251–266. DOI 10.1016/0167-9236(94)00041-2.

MCKAY, Cory, FUJINAGA, Ichiro and DEPALLE, Philippe, 2005. jAudio: A feature extraction library. In: *Proceedings of the International Conference on Music Information Retrieval*. S.l.: s.n. 2005. pp. 600–3.

MELVILLE, Prem, MOONEY, Raymod J. and NAGARAJAN, Ramadass, 2002. Content-boosted collaborative filtering for improved recommendations. In: *Proceedings of the National Conference on Artificial Intelligence*. S.l.: s.n. 2002. pp. 187–192.

OARD, Douglas W. and KIM, Jinmook, 1998. Implicit feedback for recommender systems. In: *Proceedings of the AAAI workshop on recommender systems*. S.l.: s.n. 1998. pp. 81–83.

ONO, Chihiro, KUROKAWA, Mori, MOTOMURA, Yoichi and ASOH, Hideki, 2007. A context-aware movie preference model using a Bayesian network for recommendation and promotion. In: *User Modeling 2007*. S.l.: Springer. pp. 247–257.

OWEN, Sean, 2012. *Mahout in action*. Shelter Island, N.Y.: Manning Publications Co. ISBN 9781935182689 1935182684.

PALMISANO, C., TUZHILIN, A. and GORGOGLIONE, M., 2008. Using Context to Improve Predictive Modeling of Customers in Personalization Applications. In: *IEEE Transactions on Knowledge and Data Engineering*. November 2008. Vol. 20, no. 11, pp. 1535–1549. DOI 10.1109/TKDE.2008.110.

PARK, Han-Saem, YOO, Ji-Oh and CHO, Sung-Bae, 2006. A context-aware music recommendation system using fuzzy bayesian networks with utility theory. In: *Fuzzy systems and knowledge discovery*. S.l.: Springer. pp. 970–979.

PAZZANI, Michael J., 1999. A framework for collaborative, content-based and demographic filtering. In: *Artificial Intelligence Review*. 1999. Vol. 13, no. 5-6, pp. 393–408.

PEFFERS, Ken, TUUNANEN, Tuure, ROTHENBERGER, Marcus A. and CHATTERJEE, Samir, 2007. A Design Science Research Methodology for Information Systems Research. In: *Journal of Management Information Systems*. 1 December 2007. Vol. 24, no. 3, pp. 45–77. DOI 10.2753/MIS0742-

1222240302.

RESNICK, Paul, IACOVOU, Neophytos, SUCHAK, Mitesh, BERGSTROM, Peter and RIEDL, John, 1994. GroupLens: an open architecture for collaborative filtering of netnews. In: *Proceedings of the 1994 ACM conference on Computer supported cooperative work*. S.l.: s.n. 1994. pp. 175–186.

RICARDO, André, 2010. Building a Scalable Index and a Web Search Engine for Music on the Internet using Open Source software. Lisboa: ISCTE - Instituto Universitário de Lisboa. Tese de Mestrado.

RICCI, Francesco, ROKACH, Lior and SHAPIRA, Bracha, 2011. Introduction to Recommender Systems Handbook. In: RICCI, Francesco, ROKACH, Lior, SHAPIRA, Bracha and KANTOR, Paul B. (eds.), *Recommender Systems Handbook*. Boston, MA: Springer US. pp. 1–35. ISBN 978-0-387-85819-7, 978-0-387-85820-3.

RICH, Elaine, 1979. User modeling via stereotypes. In: *Cognitive science*. 1979. Vol. 3, no. 4, pp. 329–354.

SCHEIN, Andrew I., POPESCUL, Alexandrin, UNGAR, Lyle H. and PENNOCK, David M., 2002. Methods and metrics for cold-start recommendations. S.l.: ACM Press. 2002. pp. 253.

SHANI, Guy and GUNAWARDANA, Asela, 2011. Evaluating recommendation systems. In: *Recommender systems handbook*. S.l.: Springer. pp. 257–297.

SHARDANAND, Upendra, 1994. *Social information filtering for music recommendation*. S.l.: Massachusetts Institute of Technology.

SI, Hua, KAWAHARA, Yoshihiro, KURASAWA, Hisashi, MORIKAWA, Hiroyuki and AOYAMA, Tomonory, 2005. A context-aware collaborative filtering algorithm for real world oriented content delivery service. In: *Proc. of ubiPCMM*. 2005.

SINHA, Rashmi R. and SWEARINGEN, Kirsten, 2001. Comparing Recommendations Made by Online Systems and Friends. In: *DELOS workshop: personalisation and recommender systems in digital libraries*. S.l.: s.n. 2001.

SOUSA, Nuno, 2012. Sistema de Recomendação de Aplicações Android. Lisboa: ISCTE - Instituto Universitário de Lisboa. Tese de Mestrado.

SU, Ja-Hwung, YEH, Hsin-Ho, YU, Philip S. and TSENG, Vincent S., 2010. Music recommendation using content and context information mining. In: *Intelligent Systems, IEEE*. 2010. Vol. 25, no. 1, pp. 16–26.

SU, Xiaoyuan and KHOSHGOFTAAR, Taghi M., 2009. A Survey of Collaborative Filtering Techniques. In: *Advances in Artificial Intelligence*. 2009. Vol. 2009, pp. 1–19. DOI 10.1155/2009/421425.

TURPIN, Andrew H. and HERSH, William, 2001. Why batch and user evaluations do not give the same results. In: *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*. S.l.: s.n. 2001. pp. 225–231.

UNGAR, Lyle H. and FOSTER, Dean P., 1998. Clustering methods for collaborative filtering. In: *AAAI Workshop on Recommendation Systems*. S.l.: s.n. 1998.