TÉCNICO LISBOA

Sistema de Recomendação

Filtragem Colaborativa

Vasco Ferreira Marcelino

Dissertação para obtenção do Grau de Mestre em

Engenharia Electrotécnica e de Computadores

Orientadores: Professor Doutor Mário Alexandre Teles de Figueiredo

Engenheiro Miguel Sousa Borges de Almeida

Júri

Presidente: Professor Doutor Fernando Duarte Nunes

Orientador: Engenheiro Miguel Sousa Borges de Almeida

Vogal: Professor Doutor Bruno Emanuel da Graça Martins

Outubro de 2014

“Forecasting the future hinges on understanding the present”

Noah Smith

Aos meus pais, irmão e avó,

Agradecimentos

Em primeiro lugar gostaria de agradecer ao Prof. Mário Figueiredo por ter aceitado a minha candidatura.

Gostaria de agradecer também aos meus co-orientadores, André Martins e Miguel Almeida pela orientação e supervisão que me foram dando ao longo de todo este trabalho. Por todas as sugestões e críticas que permitiram melhorar a qualidade de todo o trabalho produzido. E por último, pela forma como se demonstraram sempre disponíveis para me ajudar.

Sem a vossa ajuda este trabalho teria sido ainda mais difícil de realizar.

Queria agradecer também aos meus pais e irmão que me deram sempre muito apoio, ânimo e que se mostraram sempre disponíveis para me ajudar e ouvir nos momentos mais difíceis.

Gostaria também de agradecer em particular:

Ao meu irmão, que demonstrou ser uma peça essencial na minha formação pessoal e académica, pela coragem e incentivo que sempre me transmitiu ao longo do curso.

À minha namorada, Diana, que provou ser o meu suporte nos momentos mais difíceis. Por toda a sua compreensão, incentivo e apoio neste longo percurso.

Aos meus primos, Vítor, Tomé e André por todos os momentos especiais proporcionados e por todo o bom humor transmitido.

Aos meus amigos Pedro Neves, Lúcia, Margarida, Silva, Miguel Neto, e Sónia pela amizade e por todos os bons momentos.

Aos companheiros e amigos de faculdade pela amizade e entreajuda.

A todos aqueles que não foram aqui referidos mas que também participaram neste processo de formação pessoal e académica.

Obrigado.

Abstract

With the growth in the number of users and products on the internet, it became useful to create recommendation systems that allow users an easier access to products of their interest.

In this work, a brief analysis of the main recommendation systems was done, giving special emphasis to collaborative filtering systems. It was performed a comparative study of algorithms based on memory and on model, which were implemented in Python language, using the dataset from MovieLens and Netflix.

The results obtained were compared with each other and with those presented in the articles [20] and [24], by applying two evaluation metrics (root mean square error and mean absolute error).

It was confirmed the importance of selecting the number of nearest neighbors in algorithms based on memory and, comparing the obtained results for each of the algorithms used, it can be seen that the model-based algorithm computes predictions with higher quality.

Hereafter, it would be important to reduce training time on model-based algorithm and raise the number of recommendations per second, besides testing and comparing other model-based algorithms namely probabilistic ones.

Keywords:

Collaborative filtering, recommendation system, k-nearest neighbor, scalability, single value decomposition, sparsity.

Resumo

Com o crescimento acentuado do número de utilizadores e produtos na *internet*, tornou-se útil a criação de sistemas de recomendação que permitem aos utilizadores o acesso facilitado a produtos do seu interesse.

Nesta dissertação, foi feita uma breve análise dos principais sistemas de recomendação, com destaque para os sistemas de filtragem colaborativa. Dentro destes, realizou-se um estudo comparativo entre algoritmos baseados em memória e em modelo, cuja implementação foi produzida em linguagem Python, utilizando o conjunto de dados do *MovieLens* e da *Netflix*.

Os resultados obtidos foram comparados entre si e com os apresentados nos artigos [20] e [24], através da aplicação de duas métricas de avaliação (raiz quadrada do erro quadrático médio e erro médio absoluto).

Verificou-se a importância da escolha do número de vizinhos próximos nos algoritmos baseados em memória e, comparando os resultados obtidos por cada um dos algoritmos utilizados, verifica-se que o algoritmo baseado em modelo calcula predições com maior qualidade.

Futuramente seria importante diminuir o tempo de treino do algoritmo baseado em modelo e aumentar o número de recomendações calculadas por segundo, além de testar e comparar outros algoritmos baseados em modelo, nomeadamente os algoritmos probabilísticos.

Palavras-Chave

Sistemas de recomendação, Filtragem Colaborativa, Esparsidade, Escalabilidade, *k*-vizinhos mais próximos, Decomposição em Valores Singulares.

Conteúdo

Lista de Figuras

Lista de Tabelas

Abreviaturas

RS – *Recommender Systems*
CFS – *Collaborative Filtering Systems*
CBS – *Content-based Systems*
HRS – *Hybrid Recommender Systems*
RMSE – *Root Mean Square Error*
MAE – *Mean Absolute Error*
IMDB – *Internet Movie Database*
CF – *Collaborative Filtering*
NN – *Nearest Neighbor*
SVD – *Singular Value Decomposition*

1. Introdução

1.1. Motivação

Com a evolução da internet o acesso à mesma tornou-se mais fácil tendo levado os seus utilizadores a interagir cada vez mais com novos produtos/conteúdos como a pesquisa de um artigo, a leitura de um jornal, a aquisição de um produto ou a procura de um amigo nas redes sociais. Com o crescimento das tecnologias de informação o volume de dados disponível na internet intensificou-se, tornando o processo de pesquisa de informação mais difícil aos utilizadores uma vez que nem sempre o que lhes era apresentado seria o que estes pretendiam.

A necessidade de recolha e organização da informação revelou ser cada vez maior face à evolução no mundo da comunicação digital e foi essa necessidade que levou à criação dos sistemas de recomendação ou *Recommender Systems* (RS). Estes sistemas vieram facilitar o tratamento da informação permitindo a procura e sugestão de produtos que estejam dentro dos interesses de cada utilizador, evitando expor o utilizador a um demoroso processo de selecção de informação supérflua [1].

Como um dos objectivos primordiais das empresas são as vendas, com a introdução dos sistemas de recomendação nos seus serviços de venda *online* a sugestão de novos produtos era vital pois não só o utilizador tinha acesso ao produto que pretendia adquirir como também ficaria interessado noutros produtos que lhe eram recomendados. A introdução destes sistemas nos serviços *web* registou um aumento do número de vendas, e consequentemente o interesse das empresas em adquirir novos sistemas de recomendação [1]. A qualidade das recomendações revelou ser um factor determinante nestes sistemas uma vez que a confiança dos utilizadores nas sugestões apresentadas só se verificaria se estas se aproximassem o mais possível dos seus interesses.

Os sistemas de recomendação possuem diferentes técnicas que diferem entre si pela forma como cada uma reúne e trata a informação relativa às preferências dos utilizadores, como é o caso de produtos anteriormente vistos ou adquiridos. Assim como existem diferentes técnicas de recomendação existem também diferentes formas de recolha de informação que poderão ou não envolver a intervenção do utilizador.

Dentro dos sistemas de recomendação um dos mais utilizados é a filtragem colaborativa ou *collaborative filtering systems* (CFS) [1] que têm por base o seguinte pressuposto: se dois utilizadores classificaram/adquiriram produtos semelhantes no passado, então estes irão classificar/adquirir produtos semelhantes no futuro.

Com a crescente implementação dos SR nos serviços de vendas *online* surge, em 2006, uma iniciativa por parte da *Netflix*, cujo objectivo passaria pela implementação de um algoritmo de filtragem colaborativa capaz de obter melhores resultados que os obtidos pelo algoritmo utilizado pela empresa, ou seja, o novo algoritmo teria que apresentar uma métrica de erro 10% inferior ao do

sistema utilizado pelo próprio *Netflix*, o *Cinematch* [3]. Foram mais de cinco mil o número de equipas que participaram na competição, e foi em 2009 que foi atribuído o prémio à equipa *BellKor's Pragmatic Chaos* [2].Esta competição levou à criação e aperfeiçoamento dos sistemas de filtragem colaborativa, e é vista como tendo sido um impulsionador no desenvolvimento dos sistemas colaborativos.

1.2. Objectivos

Neste trabalho de dissertação pretende-se fazer um estudo e apresentação do estado da arte dos sistemas de recomendação, focando-se essencialmente nos algoritmos de filtragem colaborativa ou *collaborative filtering algorithms* (CFA).

Estes algoritmos servem-se de diversas técnicas de análise de dados, que aqui serão implementadas utilizando dois conjuntos de dados: o do *MovieLens* e o do *Netflix*, ambos disponíveis *online*.

Os resultados obtidos serão analisados e comparados entre si e em relação a resultados apresentados em trabalhos desta área [20] [24], com base em duas métricas de avaliação de erro, a RMSE e o MAE[1].

1.3. Contribuições Originais

Neste trabalho de dissertação foi realizada a reprodução e comparação de duas abordagens utilizando dois conjuntos de dados, o *MovieLens* e a *Netflix*

Esta dissertação foi feita no âmbito de uma parceria entre o Instituto Superior Técnico e a Priberam[2]. No entanto nenhuma das técnicas apresentadas tinha sido implementada no âmbito desta parceria, pelo que todo o trabalho aqui desenvolvido é trabalho individual.

A linguagem de programação utilizada na realização de todos os algoritmos foi exclusivamente a linguagem Python. A escolha desta linguagem deve-se sobretudo à influência que tem vindo a ter no mundo da programação e pelo facto de ser uma linguagem de fácil aprendizagem [12], sendo considerada uma das linguagens mais importantes actualmente [5] [6] [7].

1 Estas métricas de erro encontram-se detalhadas na secção 4.2.

[2] http://www.priberam.pt/

1.4. Organização

Este documento encontra-se organizado da seguinte forma:

Na secção 2 abordam-se temas como o estado da arte dos sistemas de recomendação, a classificação da informação e o condicionamento dos sistemas face a essa classificação e algumas limitações dos sistemas de recomendação. Nesta secção abordam-se ainda duas técnicas baseadas em sistemas de filtragem colaborativa.

Na secção 3 realiza-se um estudo de dois tipos de algoritmos colaborativos, os baseados em memória e baseado em modelo. Dentro destes algoritmos serão analisadas algumas técnicas implementadas neste trabalho.

Na secção 4 começa-se por introduzir os conjuntos de dados utilizados neste trabalho, juntamente com as métricas de avaliação utilizadas. Por último apresentam-se e discutem-se os resultados obtidos na implementação dos algoritmos propostos.

Na secção 5 apresentam-se as conclusões do trabalho e possíveis abordagens futuras.

2. Conceitos e Revisão dos Sistemas de Recomendação

2.1. Estado da Arte

2.1.1. Contextualização histórica

Com o rápido crescimento da internet, tornou-se importante o desenvolvimento de mecanismos capazes de facilitar a filtragem de informação consoante as preferências dos utilizadores.

As empresas de comércio *online* começaram por introduzir estes sistemas nas suas plataformas de venda de produtos por permitirem a aprendizagem das preferências e gostos dos consumidores, podendo com base nessa informação, gerar recomendações de novos produtos aos utilizadores.

Os sistemas de recomendação têm vindo a ganhar destaque também nas áreas de investigação com o objectivo de melhorar e criar novos algoritmos, mais eficientes e mais eficazes nas suas recomendações.

A competição da *Netflix* foi importante e serviu de motivação aos participantes que para além de poderem vir a ver os seus algoritmos implementados também ganhariam o prémio no valor de 1 milhão de dólares.

De seguida serão apresentados alguns serviços que utilizam sistemas de recomendação, e que de alguma forma fazem parte do estado da arte dos sistemas de recomendação.

Tapestry

Os sistemas de recomendação surgiram como área de pesquisa independente em meados de 1990 com a criação do sistema *Tapestry* [43].

Tapestry foi o primeiro sistema de recomendação comercial que utilizava algoritmos colaborativos e de conteúdo na filtragem de *emails* recebidos por um conjunto de utilizadores que pertenciam a uma *mailing list*.

O modo de funcionamento do sistema *Tapestry* consistia num conjunto de anotações feitas pelos utilizadores às mensagens recebidas (classificando-as com “bom” ou “mau” ou através de um texto de opinião) [51]. Essas mensagens eram armazenadas numa base de dados e podiam ser consultadas a partir do seu conteúdo ou a partir da opinião de outros utilizadores. Por exemplo, o

sistema era capaz de sugerir um conjunto de documentos com classificações positivas mais elevadas ou através da existência de palavras-chave no documento [51].

Uma das desvantagens apresentadas por este sistema devia-se à inexistência de um mecanismo capaz de agrupar os utilizadores por interesses similares, razão pela qual este sistema necessitava da interacção do utilizador através das suas anotações às mensagens.

GroupLens

Em 1994 foi criado um novo sistema de recomendação, *GroupLens*, baseado em filtragem colaborativa e desenvolvido por um grupo de investigação da Universidade de Minnesota também denominado *GroupLens* [45].

Este sistema tinha como objectivo a procura e recomendação de novos artigos presentes no serviço *Usenet* [46]. O algoritmo utilizado pelo *GroupLens* baseava-se na relação entre utilizadores similares permitindo a formação de uma vizinhança, isto é, de um conjunto de utilizadores que possuíssem gostos semelhantes entre si.

O projecto *GroupLens* contribuiu de diferentes formas para a evolução dos sistemas de recomendação. Por exemplo, demonstraram com base em alguns estudos iniciais que era possível obter-se um maior número de classificações ou *ratings*[3] de forma implícita (com base no tempo de leitura do utilizador) e que essas predições eram comparáveis quanto às obtidas a partir de *ratings* explícitos[4] [25]. Isto deve-se ao facto de nem todos os utilizadores tenderem a classificar os documentos lidos e neste caso a única alternativa passa pela interpretação da informação obtida implicitamente.

Esta equipa de investigação possui outro projecto na área da recomendação como é o caso do *MovieLens*.

MovieLens

O *MovieLens* é um sistema de recomendação de filmes que utiliza sistemas de filtragem colaborativa como técnica de recomendação de filmes. Neste sistema são feitas recomendações de filmes baseadas nos *ratings* atribuídos pelos utilizadores ao filmes. O sistema realiza várias sugestões à medida que vai tendo maior conhecimento dos gostos dos utilizadores, no entanto, para o caso do utilizador ser novo no sistema, o serviço faz uma sugestã de filmes baseada na lista de filmes com melhores classificações.

Na Figura 1 apresenta-se resumidamente um exemplo de recomendações efectuadas pelo sistema *MovieLens*.

[3] Será esta a terminologia adoptada ao longo do trabalho como referência a uma classificação/cotação atribuída por um utilizador a um item.
[4] A classificação de *ratings* como sendo implícita ou explícita encontra-se detalhada na secção 2.2.

Figura 1 – Exemplo de recomendações de filmes feitas pelo sistema de recomendação *MovieLens*.

Da Figura 1, o rectângulo correspondente ao número 1 representa o intervalo de classificação possível dos filmes, contida entre 1 e 5, sendo que 1 representa a classificação mais baixa e 5 a mais elevada, ou seja, que não gostou ou gostou muito, respectivamente.

O rectângulo 2 corresponde à classificação que o sistema considera que o utilizador classificaria, ou seja, corresponde à predição feita pelo sistema.

O rectângulo 3 corresponde ao processo de votação dos filmes, é com base nesta informação que o sistema analisa a similaridade com outros filmes e posteriormente executa o cálculo de uma predição ou conjunto de predições para o utilizador em questão.

O rectângulo 4 representa uma lista de interesses, esta lista é importante para o sistema uma vez que representa os gostos do utilizador quer pelas sugestões apresentadas pelo sistema quer pelos filmes anteriormente vistos e que foram da sua preferência.

À semelhança do *MovieLens*, existem outros serviços *online* de recomendação de filmes como é o caso do *IMDB*[5] ou da *Netflix*[6].

[5] http://imdb.com/
[6] https://www.*Netflix*.com/

Netflix

A empresa norte-americana, *Netflix*, é uma das empresas líder na transmissão de conteúdos cinematográficos, venda de DVDs e aluguer de filmes. Está representada em mais de 40 países, sendo que em Setembro de 2013 o número de utilizadores registados já ultrapassava os 30 milhões só nos Estados Unidos [38].

Este serviço começou por criar um perfil para que cada utilizador pudesse classificar cada filme visto com base na atribuição de uma classificação de 1 a 5. Com base nas classificações dos utilizadores o sistema de recomendação, *Cinematch*, faz recomendações de outros filmes que ainda não tenham sido vistos ou classificados.

Em outubro de 2006 a *Netflix* lança uma competição, *Netflix Prize*, que oferecia um prémio de 1 milhão de dólares à equipa que conseguisse desenvolver um algoritmo capaz de calcular predições para um conjunto de utilizadores com uma redução de 10% da RMSE relativamente ao valor anterior obtido pelo *Cinematch*, isto é, RMSE=0.9514 [15] [3].

A competição e o valor monetário associado levaram muitas equipas de investigação a desenvolver novos algoritmos de recomendação [50], e em seis dias desde o início da competição já existia uma equipa, *WXYZConsulting*, capaz de fazer recomendações com maior qualidade que o sistema de recomendação *Cinematch* [39] e durante o período da competição estiveram inscritas mais de 20 mil equipas [3].

A competição terminou ao fim de praticamente 3 anos e o prémio foi entregue à equipa *BellKor's Pragmatic Chaos* que obteve uma melhoria de 10,05% relativamente ao *Cinematch*.

Existem ainda inúmeros artigos científicos que utilizam o conjunto de dados do *Netflix* no desenvolvimento e optimização dos seus algoritmos, sendo considerado um dos maiores conjuntos de dados para análise de sistemas baseados em filtragem colaborativa.

Algumas das equipas que participaram na competição desenvolveram posteriormente algoritmos que viriam a ser utilizados por empresas como a 4-Tell [14].

De seguida, apresenta-se na Figura 2 algumas recomendações feitas pelo *Cinematch*.

Figura 2 - Exemplo de recomendações feitas pelo *Cinematch*.

Da Figura 2 é possível verificar-se que o algoritmo de recomendação do *Netflix* recolhe a informação relativa às preferências dos utilizadores através da classificação de itens anteriormente vistos por eles (na figura esta escala de classificação é representada por uma escala de 1 a 5 estrelas consoante o grau de satisfação). Repare-se que o sistema calculou 1279 sugestões de filmes com base em 208 *ratings* atribuídos pelo utilizador.

As recomendações propostas aos utilizadores provêm da análise da informação dos itens classificados, tais como o género do filme, os itens vistos recentemente, ou a relação entre o item visto e outros itens classificados por diferentes utilizadores que possuam gostos similares.

Ringo

Ringo foi um serviço de recomendação de música desenvolvido por um grupo de investigação do MIT [47]. Este serviço baseava-se na criação de um perfil de utilizador a partir das suas preferências, a partir do qual cada utilizador criava o seu perfil especificando as músicas ou álbuns que gostava ou não. A partir dessa informação o sistema *RINGO* agrupava os utilizadores com preferências musicais próximas, e gerava recomendações com base nos *ratings* atribuídos.

Para o caso dos novos utilizadores ou daqueles que ainda não tinham classificado nenhuma música o sistema recomendava o top de músicas/álbuns mais votados.

Actualmente existem alguns serviços semelhantes a este, como é o caso da *Last*.fm[7] e *Pandora*[8].

[7] http://www.lastfm.pt/
[8] http://www.pandora.com/

PHOAKS

PHOAKS (*People Helping One Another Know Stuff*) é um serviço que utiliza algoritmos colaborativos na localização de informação relevante nas páginas *web* para os utilizadores, ou seja, este sistema filtra e analisa as mensagens postadas pelos utilizadores no serviço *Usenet* comparando-as com as preferências dos diferentes grupos de notícias. Esta comparação é feita a partir de um processo de contagem de páginas *web* mais importantes para os leitores daquele grupo de notícias [44].

Este sistema utiliza um mecanismo de recolha de informação baseada em *feedback* implícito.

Fab

Fab é um sistema de recomendação que combina métodos colaborativos com métodos baseados em conteúdo como forma de diminuir os aspectos negativos apresentados por estes sistemas quando implementados individualmente [16].

Este sistema foi criado com o objectivo de ajudar os utilizadores a lidar com o aumento de informação disponível na *internet* [16] gerando recomendações de documentos através da análise do seu conteúdo com o de documentos previamente classificados com cotação elevada.

A estrutura híbrida deste sistema permite uma actualização corrente do perfil de preferências dos utilizadores.

Redes Sociais

Com a crescente popularização das redes sociais, serviços como o *Facebook*, o *Twitter* ou o *Linkedin* recorrem à utilização de sistemas de recomendação com o objectivo de divulgar a existência de novas aplicações, novas amizades ou actualizações de notícias que estejam a ocorrer na sociedade, tudo consoante os objectivos definidos pelo serviço em questão.

No caso do *Twitter* ou do *Facebook*, os sistemas de recomendação são utilizados como forma de sugestão de novos amigos ou de novas aplicações que possam ser do interesse dos utilizadores.

Na Figura 3 encontra-se representado um exemplo de recomendações feitas pelo *Facebook*.

Figura 3 – Exemplo de recomendações feitas na página *Facebook.com*.

Na Figura 3 são sugeridas páginas web, novos amigos ou grupos onde se encontram utilizadores próximos ao utilizador corrente.

Dentro dos diferentes serviços de comércio *online*, como é o caso do *ebay.com* ou da *amazon.com*, os sistemas de recomendação são utilizados na recomendação de artigos/produtos que possam interessar aos utilizadores.

Na Figura 4 apresenta-se um exemplo em que o sistema de recomendação da *amazon.com* sugere ao utilizador uma série de livros ainda não lidos com base nos itens anteriormente vistos ou adquiridos. Na mesma figura, é possível observar que o sistema realiza uma análise dos temas mais vistos pelo utilizador, e para cada um deles sugere uma coletânea de livros que considera serem do seu interesse.

Figura 4 - Exemplo de aplicação de recomendações feitas no *site* amazon.com.

Na Figura 5 encontram-se representadas algumas recomendações de vídeos feitas ao utilizador pelo serviço *YouTube.com*.

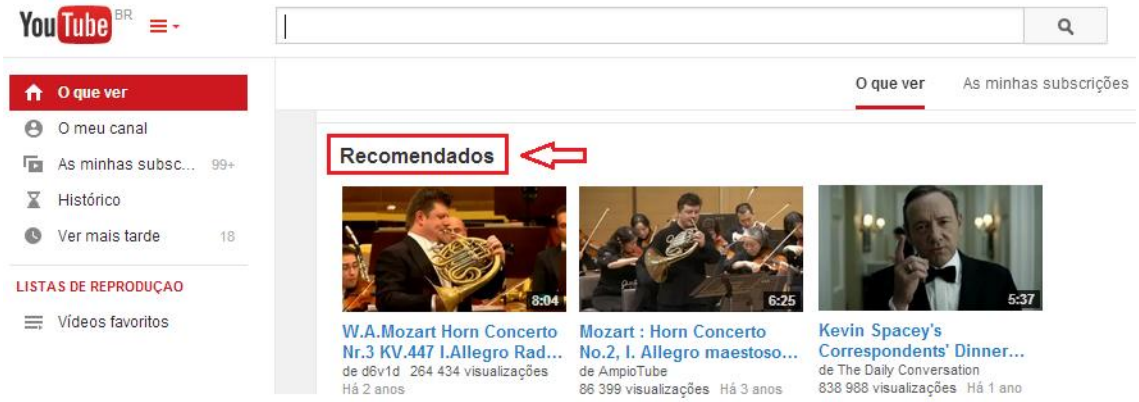

Figura 5 - Exemplo de recomendação efectuada pelo serviço *YouTube.com*.

2.1.2. Definição do problema

Considere-se $\mathbf{U}$ o conjunto de todos os utilizadores de um determinado sistema, e $\mathbf{I}$ o conjunto de todos os itens que podem ser recomendados, tais como livros, artigos, músicas ou filmes.

Seja r a função utilidade que mede a utilidade/relevância de um item s para um determinado utilizador activo[9] (ou *active user*) representada por: $r: \boldsymbol{U} \times \boldsymbol{I} \rightarrow \boldsymbol{R}$, em que $\boldsymbol{R}$ é o conjunto real. Muitas vezes é utilizado um subconjunto de $\boldsymbol{R}$ como por exemplo: a classificação de um livro através da atribuição de inteiros de 1 a 5.

Um sistema de recomendação pode apresentar as recomendações de duas formas: ou sugere um único um item ao utilizador ou um conjunto de itens (*top-N recommendations*). Neste trabalho será abordado apenas o primeiro caso, cuja função de optimização é dada por [13]:

$$\forall\, u \in \mathbf{U}, \quad i'_u = arg\, max_{i \in I}\, r(u, i) \qquad (1).$$

Da Equação 1, o sistema de recomendação pretende escolher para um utilizador $u \in \mathbf{U}$ um item $i' \in \mathbf{I}$ que maximiza a utilidade do mesmo para o utilizador.

Do conjunto de utilizadores, $\mathbf{U}$, normalmente representado através da atribuição de um ID a cada utilizador pode também conhecer-se informação adicional como a idade ou o sexo. O mesmo se pode aplicar ao conjunto dos itens, $\mathbf{I}$.

Quanto à representação das classificações dos itens por parte dos utilizadores, a metodologia adoptada será a atribuição de um *rating* num intervalo entre 1 e 5, sendo que 1 será a cotação mais baixa (sinónimo de que o utilizador não gostou ou não lhe interessa o item) e 5 a mais elevada (isto é, que o utilizador gostou muito do item ou que lhe foi muito útil).

Além da representação das classificações considerar-se-á a representação das características de um utilizador pelo seu identificador, o mesmo se aplica ao conjunto de itens.

[9] Utilizador para o qual o sistema pretende fazer uma recomendação.

A representação da informação relativa a um par utilizador-item pode ser representada por um triplet da forma <<ID do utilizador, ID do item, Classificação>> [13].

2.2. Formas de Classificação da Informação

Na Secção 2.1 foi referido que existem métodos distintos de recolha de informação dos utilizadores dependendo do tipo de sistema utilizado. A recolha de informação pode ser efectuada por dois modos [21]:

- Explicitamente
- Implicitamente

2.2.1. Classificação Explícita

A classificação explícita providenciada pelos utilizadores oferece uma descrição completa das preferências de um utilizador por um item a partir de uma pequena quantidade de dados (exemplo: número inteiro) [32].

A aquisição dos dados, realizada explicitamente, pode ser feita a partir de questionários ou da atribuição de classificações num intervalo fixo a um item (exemplo: de 1 a 5). Existem também sistemas que optam por utilizar métodos de classificação binária como é o caso do serviço TiVo através do botão *tumbs-up* ou *tumbs-down* [30]. Na Figura 6 apresentam-se alguns exemplos de sistemas que utilizam *ratings* explícitos.

Figura 6 – Exemplos de avaliação de *ratings* explícitos. A primeira figura exemplifica um sistema que avalia a informação através de números contidos num intervalo entre 1 e 10. A segunda figura retrata o sistema TiVo cuja avaliação dos conteúdos é feita através da opção gosto/não gosto.

Os *ratings* explícitos são os que melhor servem os interesses dos sistemas de filtragem colaborativa [21], no entanto requerem a participação dos utilizadores no processo de classificação dos itens, tarefa que nem sempre é vista de forma positiva pelos utilizadores [32].

Por forma a incentivar os utilizadores a classificarem os produtos adquiridos, existem já alguns serviços *online* que recorrem a um esquema de atribuição de pontos que poderão servir posteriormente para adquirir produtos grátis, participar em concursos de oferta de produtos, ou até terem acesso a conteúdo privilegiado [32].

Neste trabalho de dissertação serão utilizados apenas *ratings* explícitos como medida de avaliação das preferências dos utilizadores.

2.2.2. Classificação Implícita

Como nem todos os utilizadores classificam os itens vistos, o sistema pode não conseguir fazer uma recomendação visto não possuir informação necessária sobre as preferências do utilizador, nesse caso, alguns sistemas recolhem a informação sobre os interesses dos utilizadores de forma implícita [21].

Para o caso em que a avaliação é feita implicitamente, não é feita uma sugestão ao utilizador para que este classifique ou dê a sua opinião sobre um item, como acontecia no caso em que a classificação era feita explicitamente. O sistema infere os gostos dos utilizadores com base no histórico de compras, o histórico de itens pesquisados, o movimento ou número de cliques do rato, o tempo de permanência num *website* ou padrões de pesquisa (ex: a aquisição de filmes de um realizador específico pode significar que o utilizador gosta daquele realizador) [23].

Podem ocorrer, no entanto, algumas situações em que a análise implícita da informação pode ser imprecisa; por exemplo, o facto de um utilizador clicar numa notícia não significa que a vá ler. Logo, se o sistema realizar sugestões com base nos cliques das notícias poderá recomendar de forma errada outras notícias que não fazem parte dos interesses desse utilizador.

2.3. Classificação dos Sistemas de Recomendação

Os sistemas de recomendação são normalmente classificados de acordo com a abordagem que utilizam na estimação dos *ratings*, isto é, na forma como interpretam a informação de cada utilizador [13] [26] [17].

- Sistemas baseadas em conteúdo ou *content-based systems* (CBS). Utilizam a informação sobre o conteúdo dos itens vistos no passado para recomendar novos itens.
- Sistemas baseadas em filtragem colaborativa ou *collaborative filtering systems* (CFS). É utilizada a informação da matriz utilizador-item.
- Sistemas baseadas em filtragem híbrida ou *hybrid recommender systems* (HRS). Estes sistemas combinam essencialmente técnicas baseadas em filtragem colaborativa e em conteúdo com o objectivo de colmatar as falhas apresentadas por cada método quando implementado individualmente.

2.3.1. Sistemas baseados em Conteúdo

Os sistemas baseados em conteúdo focam-se sobretudo nas propriedades ou características dos itens e no perfil do utilizador partindo do pressuposto que itens com características similares são classificados similarmente [17] [19]. Exemplo: se um utilizador pesquisar numa página *web* a palavra "Federer", então o sistema pressupõe que este utilizador irá gostar de uma outra página *web* que contenha a palavra "Federer", mas no caso de um sistema inteligente este poderia inferir que o utilizador possui interesse pela modalidade de ténis. E neste caso o sistema recomendaria outros artigos relacionados com o tema, ainda que no seu conteúdo não estivesse presente a palavra "Federer".

Numa primeira análise o objectivo destes sistemas passa pela análise de características/propriedades dos itens que são mais importantes para, numa fase posterior, criarem uma lista com as características mais relevantes a cada perfil de utilizador. Para o caso de uma notícia de jornal, o sistema poderia utilizar apenas as palavras contidas nessa notícia e utilizá-las na comparação com outras notícias, e por último recomendar uma nova notícia. Isto é, o sistema compara as preferências de perfil do utilizador com as características do novo item (neste caso uma nova notícia), apresentando ao utilizador possíveis sugestões [13].

A subjectividade de alguns itens dificulta esta tarefa pois nem sempre é fácil a sua categorização, isto é, uma descrição objectiva das características de um item [17]. Por exemplo, a escolha de livros que não possuam ainda críticas de utilizadores (*reviews*), ou a dificuldade em obter uma representação/descrição de um conteúdo, como é o caso de filmes ou músicas.

Um dos aspectos que distingue os sistemas de filtragem de conteúdos dos restantes é o facto de estes fazerem recomendações de itens semelhantes, uma vez que a recomendação de um item é feita com base na semelhança das características desse item com as do perfil do utilizador [18].

2.3.2. Sistemas baseados em Filtragem Colaborativa

Ao contrário dos sistemas baseados em conteúdo, itens pelo qual o conteúdo não está disponível ou é difícil obtê-lo podem ser ainda recomendados a utilizadores com base na classificação desses itens por outros utilizadores [13] [26]. Os sistemas colaborativos ou sistemas de filtragem colaborativa fazem recomendações de itens desconhecidos a um utilizador com base nos itens classificados anteriormente por outros utilizadores cujo perfil ou gostos sejam similares ao do utilizador activo [20].

Uma das vantagens dos sistemas colaborativos é a possibilidade de recomendar diferentes produtos de diferentes categorias, como a recomendação de um filme a um utilizador mesmo que este só tenha visto livros. Isto é algo que se tornava complicado para os sistemas baseados em conteúdo uma vez que estes teriam que mapear o conteúdo de um domínio para o outro, ou seja, assim como existem livros categorizados como livros de acção também existem filmes, com significado próximo [26].

O perfil dos utilizadores é representado por uma matriz de *ratings* de dimensão n por m, em que n é o número de utilizadores e m o número de itens [28]. A cada entrada da matriz de *ratings* (u, i) está associada uma classificação atribuída pelo utilizador u ao item i como se encontra representado na Tabela 1 [20]. Para a matriz apresentada, os valores dos *ratings* são inteiros entre 1 a 5.

O *site Amazon.com* ou *CDnow.com* são alguns exemplos de serviços que utilizam esta forma de representação da informação [9] [11].

		Itens					
		1	2	….	*i*	….	*m*
Utilizadores	1	3				5	
	2	1	1	2			
	…			4	3	5	
	u		4		5		
	…	1	2			1	
	n			3	1	2	

Tabela 1 – Representação do conjunto de dados do sistema a partir de uma matriz de *ratings.*

Em sistemas reais verifica-se que a matriz de *ratings* é esparsa [20], ou seja, a quantidade de *ratings* atribuídos é pequena quando comparada com o conjunto total de entradas da matriz.

Existem ainda outros factores que condicionam o desempenho dos sistemas colaborativos [16] [18] [22] [26].

- Quanto maior o número de pessoas que classificam os produtos mais fácil será encontrar um grupo de utilizadores com gostos similares ao utilizador activo.
- O sistema não consegue fazer uma predição para novos produtos introduzidos no conjunto de dados até que algum utilizador classifique o item ou informe o sistema da sua similaridade com outros itens. Este fenómeno é denominado *cold-start* e será abordado na Secção 2.4.
- As preferências de um utilizador podem ser incomuns quando comparados com as de outros utilizadores. Deste modo não haverá nenhum que seja semelhante, pelo que a qualidade das predições pode ser posta em causa.

Para além dos factores acima apresentados existem outros que se encontram detalhados na Secção 2.4.

Os sistemas de recomendação baseados em filtragem colaborativa podem dividir-se em duas categorias de algoritmos [20]:

- *A*lgoritmos baseados em memória ou *Memory-Based Algorithms;*
- Algoritmos baseados em modelo ou *Model-Based Algorithms.*

Estes dois algoritmos apresentam-se detalhados na Secção 3.

2.3.3. Sistemas de recomendação Híbridos

Os sistemas de recomendação por filtragem híbrida resultam da combinação de duas ou mais técnicas com o objectivo de melhorar o seu desempenho, evitando simultaneamente, algumas das debilidades que um só método apresentaria se fosse implementado individualmente [19].

Segundo Burke, os métodos híbridos resultam da combinação de técnicas baseadas em filtragem colaborativa com técnicas de filtragem baseada em conteúdo [19].

De seguida apresentam-se alguns métodos híbridos que resultam da combinação de diferentes técnicas [19].

Método *Weighted:*

Cada componente do sistema híbrido pontua um dado item, sendo que essas pontuações são combinadas utilizando uma combinação linear. Por exemplo: o score final do item para o utilizador activo seria calculado através da soma de scores obtidos por diferentes técnicas do sistema [19].

Método *Switching*:

O sistema alterna entre diferentes técnicas de recomendação consoante a interpretação dos dados do sistema. Por exemplo: se os sistemas de filtragem colaborativa não conseguirem produzir

recomendações com elevada confiança então o sistema tenta calcular uma recomendação a partir de outra técnica como a filtragem baseada em conteúdo, e assim sucessivamente [19].

Este método supõe que a comutação entre as diferentes técnicas é viável [51].

Método *Mixed*:

Este método híbrido apresenta as recomendações dos diferentes componentes lado a lado numa lista combinada [19], sem tentativas de combinar a informação obtida por métodos de recomendação diferentes.

Método *Feature Combination*:

Neste método, o *rating* produzido por um dos sistemas de recomendação entra no próximo sistema como uma característica [19].

Um possível exemplo seria a entrada do *rating* produzido pelo sistema baseado em filtragem colaborativa num sistema baseado em conteúdo, uma vez que os sistemas baseados em conteúdo analisam as características representativas do item.

Método *Cascade*:

Neste método as recomendações obtidas por um sistema são aperfeiçoadas por outro, ou seja, este método de hibridizar possui um conjunto de fases a seguir.

A primeira técnica de recomendação aplicada produz um *ranking* grosseiro dos candidatos, enquanto a segunda técnica aperfeiçoa os scores dos itens [19]. Exemplo: itens que tenham sido classificados com scores idênticos pela primeira técnica poderão vir a ser reclassificados pela segunda técnica do sistema [51].

A vantagem deste método passa por evitar que a segunda técnica do sistema seja aplicada a itens bem diferenciados pela primeira técnica, ou suficientemente mal avaliados de tal modo que o sistema nunca os irá recomendar [51].

Método *Feature Augmentation*:

A saída de um sistema é utilizada como uma característica de entrada noutro sistema.

Este método, da mesma forma que o método *cascade*, requer fases de processamento [51]. A primeira técnica de recomendação produz ratings para cada item. Depois a segunda técnica de recomendação recolhe a informação obtida pela primeira técnica e integra-a no processo de recomendação.

Note-se no entanto que este método é diferente do método *cascade* uma vez que no último método as recomendações obtidas pela aplicação da primeira técnica não influenciam a segunda [51].

Método *Meta-level*:

A aprendizagem de um modelo é utilizada como um parâmetro de entrada de outro sistema [19]. Este método difere do *feature augmentations* uma vez que neste último é feita uma aprendizagem do modelo que gera características que serão usadas como argumentos de entrada do segundo algoritmo, enquanto no método híbrido *meta-level* todo o modelo se torna a entrada do segundo algoritmo [19].

A vantagem deste método, sobretudo se forem utilizadas técnicas de filtragem colaborativas e baseadas em conteúdo, é que a aprendizagem do primeiro mecanismo de recomendação (baseado em conteúdo) é feita a partir de uma representação compacta das preferências dos utilizadores, enquanto o segundo mecanismo (baseado em filtragem colaborativa) é capaz de trabalhar com uma estrutura de dados densa.

O primeiro sistema híbrido criado que utilizava este método foi o sistema *Fab* [19].

2.4. Desafios e Limitações dos Sistemas Colaborativos

Embora os sistemas de recomendação tenham vindo a melhorar ainda existem alguns desafios, tais como: esparsidade de dados (*Data Sparsity*), escalabilidade (*Scalability*), problema de inicialização (*Cold Start)*, fraude (*Fraud)*, *Black sheep, Gray sheep* e sinonímia (*Synonymy)* [31] [11] [13].

2.4.1. Esparsidade

É comum referir-se na área dos sistemas de recomendação que a representação da informação a partir de uma matriz de *ratings* é esparsa, uma vez que existe um grande número de entradas não classificadas/conhecidas no sistema (habitualmente representadas em figuras com um espaço vazio na matriz como se apresentou na Tabela 1).

Este problema ocorre porque nem todos os utilizadores classificaram a grande maioria dos itens que viram ou adquiriram.

A esparsidade dos dados no sistema pode influenciar a qualidade da recomendação. Uma das formas de colmatar este problema é através da utilização da informação de perfil do utilizador ou item no cálculo da similaridade[10]. Esta informação pode ser, por exemplo, o sexo da pessoa, a idade ou a sua profissão.

2.4.2. Escalabilidade

Ocorre quando o número de utilizadores e itens aumenta drasticamente exigindo um maior esforço computacional no tratamento da informação relativa a cada utilizador [11].

Para um grande número de utilizadores e itens, o simples armazenamento da matriz de *ratings* pode ser difícil.

Uma possível solução para o problema da escalabilidade dos dados seria a implementação de um algoritmo capaz de dividir o conjunto de utilizadores de um conjunto de dados em grupos de utilizadores com gostos/preferências semelhantes. Deste modo, sempre que o sistema pretender calcular uma recomendação, ao invés de comparar o utilizador com todos os utilizadores do sistema compara apenas com o grupo de utilizadores que possuam maiores semelhanças com as do utilizador que se pretende recomendar.

2.4.3. Problema de Inicialização (*Cold Start Problem*)

Este problema descreve uma situação em que o sistema de recomendação é incapaz de fazer uma recomendação face à falta de informação inicial [11]. Este problema pode ocorrer de duas formas: quando um novo utilizador ou item dá entrada no sistema.

Quando um novo item é recente no sistema (*early rater*), até que este item seja classificado por um grupo razoável de utilizadores os sistemas não serão capazes de fazer recomendações.

No caso de ser um novo utilizador, o sistema em primeiro lugar deve extrair as preferências deste novo utilizador para posteriormente poder selecionar o grupo de utilizadores que classificaram itens similares a ele.

2.4.4. Fraude

Muitas vezes é sugerido um item pela quantidade de comentários ou pelo número de *ratings* positivos, no entanto essa informação pode não ser totalmente fiável.

Por exemplo, um vendedor pode classificar o seu item inúmeras vezes como sendo um artigo de “grande qualidade” sem que o sistema detecte que a avaliação do item foi fraudulenta. Este

[10] A similaridade entre utilizadores ou itens encontra-se definida na Secção 3.1.

problema ocorre em sistemas que não consigam detectar que o mesmo utilizador possui diferentes contas.

Ao contrário do exemplo anteriormente apresentado pode ocorrer o caso em que um produto seja de boa qualidade mas que possua uma classificação baixa, uma vez que vendedores concorrentes classificaram o produto como mau.

Estes actos podem ser denominados *push attacks* (em seu próprio benefício) ou *nuke attack* (reduzindo os *ratings* dos seus concorrentes) [11].

2.4.5. Gray Sheep e Black Sheep

Gray sheep ocorre quando a opinião de um utilizador não está nem de acordo nem em desacordo com o restante grupo de utilizadores do sistema. Este grupo de pessoas dificilmente receberá recomendações com alguma qualidade mesmo depois de feita uma fase inicial de conhecimento das suas preferências [11].

Um utilizador *white sheep* é aquele que classifica os seus produtos de forma semelhante à maioria dos utilizadores [11].

O utilizador *black sheep* é aquele que ou classifica os produtos como sendo de categoria “muito mau” ou “muito bom”, acabando por não se poder relacionar com o restante grupo de utilizadores [11]. Este problema pode ser diminuído com o aparecimento de utilizadores com gostos ou formas de classificar semelhantes.

2.4.6. Sinonímia

Ocorre quando produtos com nomes diferentes se referem a produtos similares. Os sistemas baseados em semelhanças, como os de filtragem colaborativa, não conseguem distinguir esta associação latente pelo que tratam estes produtos de forma diferente.

Considere-se o exemplo de dois utilizadores, onde um classifica um bloco de notas com classificação máxima de 5, enquanto outro classifica um caderno de argolas com classificação máxima também. Para este caso estes sistemas baseados na correlação não conseguem detectar a relação entre estes dois produtos como sendo dois produtos pertencentes à classe escritório/papelaria, e com a mesma finalidade [15].

3. Técnicas de Recomendação Implementadas

3.1. Algoritmos baseados em Memória

Esta classe de algoritmos é conhecida por armazenar todos os dados em memória e, a partir dessa informação calcular medidas de similaridade entre utilizadores ou itens para um utilizador activo cada vez que se pretende fazer uma recomendação. Este passo é realizado sempre que o sistema necessite de calcular a predição de um item para um utilizador, pelo que estes algoritmos não possuem uma fase de treino.

3.1.1. Filtragem Colaborativa baseada em Vizinhança (*Neighborhood-based* CF)

Uma das técnicas mais utilizadas nos sistemas de recomendação baseados em memória é o $k-NN$ (k- *Nearest Neighbor*), isto é, k – vizinhos mais próximos.

A definição de vizinhos mais próximos pode ser aplicada tanto à relação entre utilizadores como itens. Pela simples razão de se tornar mais intuitivo, todos os exemplos abaixo apresentados são dados relativamente à relação entre utilizadores, porém todos os exemplos e definições se aplicam de igual forma aos itens.

Os algoritmos de filtragem colaborativa baseada na relação entre utilizadores ou *user-based* CF têm como objectivo a procura dos utilizadores que melhor se aproximam das preferências do utilizador activo. Por exemplo, pode ser útil ao sistema uma atribuição de pesos a todos os utilizadores consoante a sua semelhança com o utilizador activo.

Esta análise de preferências entre utilizadores é feita a partir dos *ratings* dos itens atribuídos por ambos, sendo que se se verificar uma similaridade entre os utilizadores o sistema é capaz de recomendar um item que ainda não tenha sido visto por um dos utilizadores [26].

Por último, o sistema obtém uma lista com o número de utilizadores que mais se aproximam do utilizador activo, e com base nessa semelhança calcula predições para itens ainda não classificados.

De uma forma mais compacta estes algoritmos podem dividir-se em três fases [26]:

- Cálculo das similaridades entre todos os utilizadores do sistema e o utilizador activo,
- Selecção dos k utilizadores com maior grau de similaridade com o utilizador activo.

- Cálculo de um *score* (ou pontuação) ponderado para cada item com base no conjunto dos k utilizadores com maior grau de semelhança com o utilizador activo.

Às três fases do algoritmo apresentadas anteriormente podia acrescentar-se ainda uma quarta fase em que o sistema seleciona os itens ainda não classificados que obtiveram maior *score* na fase anterior do algoritmo, e com base nesse conjunto de itens recomendar um, ou um conjunto de N itens ($top - N$ *items*).

Na segunda fase do algoritmo é importante a escolha do valor de k no sistema, sendo que, para valores de k muito pequenos, corremos o risco do classificador se tornar demasiado sensível uma vez que dependerá sobretudo de um conjunto de dados reduzido [26]. Por outro lado, se o valor de k for demasiado elevado, a vizinhança poderá incluir demasiados vizinhos que podem diminuir a qualidade do sistema uma vez que nem todos os utilizadores dessa vizinhança são os melhores utilizadores a considerar, para além do sistema ficar mais lento.

É recorrente nos sistemas com implementação ingénua que o custo computacional seja elevado uma vez que o sistema recalcula o conjunto de utilizadores com gostos similares ao utilizador activo sempre que se pretende recomendar um item. Num sistema mais inteligente bastaria recalcular esse conjunto de utilizadores a cada T horas ou dias [26].

Podem ocorrer casos em que o utilizador pode alterar sempre que desejar a classificação que atribuiu a um produto, porém quando o sistema necessitar de calcular novamente o grupo de vizinhos mais próximos do utilizador activo, este pode deixar de pertencer à vizinhança uma vez que os *ratings* dos seus produtos sofreram alterações. O mesmo pode ocorrer em relação aos itens do sistema.

O esforço computacional é variável consoante o número de avaliações feitas pelo sistema. Quanto maior o número de utilizadores mais exigente será realizar as operações acima descritas.

Das limitações apresentadas na Secção 2.4, para os algoritmos baseados em memória destacam-se essencialmente os problemas de escalabilidade e esparsidade dos dados.

De seguida apresentam-se algumas medidas possíveis para o cálculo das similaridades entre utilizadores.

Similaridade do co-seno (ou *Cosine Similarity*):

No cálculo das similaridades apresentadas neste trabalho, os utilizadores ou itens são representados por vectores. Cada vector do utilizador possui os *ratings* correspondentes aos itens que classificou. No caso dos itens, cada vector representa o conjunto de utilizadores que classificaram esse item.

Partindo da matriz de *ratings* representada na Tabela 1, o vector de dimensão m [11]correspondente ao utilizador U_1 poderia ser representado por:

U_1	1	3				5	

Tabela 2 - Representação de um utilizador através de um vector.

A representação de um item corresponderia a uma coluna de dimensão n, em que n corresponde ao número de utilizadores do sistema.

A medida de similaridade do co-seno entre os *ratings* do utilizador activo e os dos utilizadores que fazem parte da sua vizinhança é representada pela medida do ângulo entre eles.

Assim, o cálculo da similaridade do co-seno entre o utilizador activo u e um utilizador pertencente à vizinhança, v, é dado pela Equação (2) [32].

$$sim_{cos}(\overrightarrow{r_u},\overrightarrow{r_v}) = \frac{\overrightarrow{r_u} \cdot \overrightarrow{r_v}}{\|\overrightarrow{r_u}\| \times \|\overrightarrow{r_v}\|} = \frac{\sum_{i=1}^{m} r_{u,i} r_{v,i}}{\sqrt{\sum_{i=1}^{m} {r_{u,i}}^2}\sqrt{\sum_{i=1}^{m} {r_{v,i}}^2}} \qquad (2).$$

Da Equação (2), $\overrightarrow{\boldsymbol{r_u}} \cdot \overrightarrow{\boldsymbol{r_v}}$ representa o produto interno entre os dois vectores de ratings $\overrightarrow{\boldsymbol{r_u}}$ e $\overrightarrow{\boldsymbol{r_v}}$, e $\|\overrightarrow{\boldsymbol{r_u}}\|$ e $\|\overrightarrow{\boldsymbol{r_v}}\|$ representam a norma Euclidiana de $\overrightarrow{\boldsymbol{r_u}}$ e $\overrightarrow{\boldsymbol{r_v}}$, respectivamente.

O cálculo da similaridade de co-seno é sempre um número não negativo uma vez que os *ratings* são sempre números não negativos. Portanto a medida de similaridade do co-seno varia entre 0 (fraca correlação) e 1 (correlação forte).

Para o caso dos itens não classificados assume-se que têm *rating* zero.

Partindo da Equação (2), é possível ainda centrar os *scores* (ver Equação (3)) através da subtracção do valor 3 aos *ratings* do utilizador u e v, por ser o valor central do intervalo entre 1 e 5.

$$sim_{cent\_scores}(\overrightarrow{r_u},\overrightarrow{r_v}) = \frac{\sum_{i=1}^{m}(r_{u,i}-3)(r_{v,i}-3)}{\sqrt{\sum_{i=1}^{m}(r_{u,i}-3)^2}\sqrt{\sum_{i=1}^{m}(r_{v,i}-3)^2}} \qquad (3).$$

Da Equação (3) verifica-se que os *ratings* centrados estão contidos entre -2 e 2, e que é possível obter valores negativos para o cálculo da similaridade ao contrário do que acontece na Equação (2). Deste modo o valor da similaridade de co-seno pode variar entre -1 e 1 [18] ao contrário do que acontecia no caso anterior em que os *ratings* estavam contidos entre 1 e 5, onde a similaridade variava entre 0 e 1.

Como alguns utilizadores tendem a classificar alguns itens com *ratings* mais elevados que outros, e como alguns itens tendem a ter uma média de classificações superior a outros, para que se possa diminuir os efeitos globais enunciados é necessário aplicar correcções tais como a normalização dos *ratings* [52].

[11] Recorde-se que a matriz de *ratings* é definida como uma matriz de dimensões $n \times m$.

A normalização dos *ratings* dos utilizadores a partir da subtracção ao *rating* original de uma combinação pesada das médias dos ratings total, do item e do utilizador em causa, diminui o efeito das preferências individuais de cada utilizador e os efeitos de popularização dos itens.

Um possível exemplo de normalização dos *ratings* encontra-se apresentado na Equação (4):

$$sim_{norm}(\overrightarrow{r_u}, \overrightarrow{r_v}) = \frac{\sum_{i=1}^{m}(\frac{r_{u,i} - mR_u}{MR_u - mR_u})(\frac{r_{v,i} - mR_v}{MR_v - mR_v})}{\sqrt{\sum_{i=1}^{m}(\frac{r_{u,i} - mR_u}{MR_u - mR_u})^2}\sqrt{\sum_{i=1}^{m}(\frac{r_{v,i} - mR_v}{MR_v - mR_v})^2}} \quad (4).$$

Da Equação (4), mR_u e mR_v correspondem ao *rating* mínimo dos utilizadores u e v, respectivamente, e MR_u e MR_v ao *rating* máximo dos utilizadores u e v.

Aplicando a similaridade de co-seno a estes *ratings* normalizados tem-se que utilizadores com gostos/preferências muito díspares possuem vectores praticamente dispostos em direcções opostas [18] pelo que o valor da similaridade de Co-seno se encontra próximo de -1.

Coeficiente de correlação de Pearson (ou *Pearson Correlation Coefficient Similarity*):

Partindo da Equação (2), subtraindo ao *rating* dos utilizadores u e v a sua média, o cálculo da similaridade é dado pela Equação (5) [10]:

$$sim_{pears\_corr}(r_u, r_v) = \frac{\sum_{i \in \mathrm{I}_u \cap \mathrm{I}_v}(r_{u,i} - \overline{r_u})(r_{v,i} - \overline{r_v})}{\sqrt{\sum_{i \in \mathrm{I}_u \cap \mathrm{I}_v}(r_{u,i} - \overline{r_u})^2}\sqrt{\sum_{i \in \mathrm{I}_u \cap \mathrm{I}_v}(r_{v,i} - \overline{r_v})^2}} \quad (5).$$

Este cálculo da similaridade é conhecido por coeficiente de correlação de Pearson.

Da Equação (5) tem-se que $\mathbf{I} = \mathbf{I_u} \cap \mathbf{I_v}$ é o conjunto de itens avaliados por ambos os utilizadores, $\boldsymbol{r_{u,i}}$ é o *rating* atribuído ao item i pelo utilizador u e $\overline{\boldsymbol{r_u}}$ é a média de todos os *ratings* classificados pelo utilizador u.

O cálculo da similaridade só é possível se forem isolados todos os *co-ratings*, ou seja, todos os casos em que os utilizadores classificaram simultaneamente ambos os itens i e j.

Os coeficientes de correlação de Pearson podem variar entre -1 e 1 sendo que -1 corresponde a uma correlação fortemente negativa e 1 a uma correlação fortemente positiva [10] [18].

Cálculo da predição:

Foram apresentadas algumas formas de calcular a similaridade entre o utilizador activo e os utilizadores que pertencem à sua vizinhança.

Conhecidas as similaridades, apresenta-se na Equação (6) uma possível forma de calcular as predições dos itens [32].

$$pred(u, i) = \overline{r_u} + \frac{\sum_{v \in V} sim(r_u, r_v) \times (r_{v,i} - \overline{r_v})}{\sum_{v \in V} sim(r_u, r_v)} \quad (6)$$

Da Equação (6), $pred(u,i)$ corresponde à predição do item i para o utilizador u sendo que $\mathbf{V}$ representa a vizinhança de u.

Filtragem colaborativa baseada na relação entre itens (*Item-based CF*)

À semelhança da filtragem colaborativa baseada na relação entre utilizadores, neste método as predições para os utilizadores do sistema são calculadas com base na semelhança entre itens [20] [32]. Assim, se dois ou mais itens tiverem sido classificados de forma semelhante por diferentes utilizadores então o sistema infere esses utilizadores possuem gostos similares pelos mesmos itens [20].

Para uma melhor clarificação deste processo de avaliação entre itens considere-se a análise dos itens I_1e I_2 da Tabela 1 representado de uma forma resumida na Tabela 3.

		Itens	
		I_1	I_2
Utilizadores	U_1	3	
	U_2	1	1
	U_u	?	4
		1	2
	U_n		

Tabela 3 - Comparação dos itens 1 e 2 da matriz de *ratings.*

Se o sistema quiser predizer se o utilizador U_u irá ou não gostar do item I_1 com base na relação com I_2, então para os mesmos itens, será feito um cálculo da medida de similaridade com base na classificação atribuída pelos utilizadores representados pela cor cinza da Tabela 3.

De um modo geral, pode inferir-se que que a similaridade baseada nos utilizadores é calculada utilizando as linhas da matriz de *ratings*, enquanto nos sistemas colaborativos baseados na relação entre itens são utilizadas as colunas [20].

Analogamente ao cálculo da similaridade entre utilizadores, a similaridade entre itens pode ser calculada a partir das seguintes medidas de similaridade [20]:

Similaridade de co-seno:

Esta medida de similaridade já foi apresentada na filtragem colaborativa baseada na relação entre utilizadores (ver Equação (2).). Para o caso dos sistemas colaborativos baseados na relação entre itens, o cálculo da similaridade de co-seno entre os itens i e j é dado pela Equação (7) [20]:

$$sim_{\cos\_item}(\overrightarrow{r_i}, \overrightarrow{r_j}) = \frac{\overrightarrow{r_i} \cdot \overrightarrow{r_j}}{\|\overrightarrow{r_i}\| \times \|\overrightarrow{r_j}\|} \quad (7).$$

Da Equação (7), $\overrightarrow{r_i} \cdot \overrightarrow{r_j}$ representa o produto interno entre os dois vectores de ratings $\overrightarrow{r_i}$ e $\overrightarrow{r_j}$, e $\|\overrightarrow{r_i}\|$ e $\|\overrightarrow{r_j}\|$ representam a norma Euclidiana de $\overrightarrow{r_i}$ e $\overrightarrow{r_j}$, respectivamente.

Comparativamente com a Equação (2)., a Equação (7) utiliza a mesma expressão mas transpondo a matriz.

Similaridade baseada na correlação:

Para este caso a similaridade entre os itens i e j é medida a partir do cálculo da correlação de *Pearson-r* [20]. À semelhança da relação entre utilizadores o cálculo da similaridade só é possível se forem isolarem todos os *co-ratings*, ou seja, todos os casos em que os utilizadores classificaram simultaneamente ambos os itens i e j.

Seja U o conjunto formado por todos os utilizadores que classificaram ambos os itens i e j, $r_{u,i}$ e $r_{u,j}$ a classificação atribuída pelo utilizador u aos itens i e j, e $\overline{r_i}$ e $\overline{r_j}$ os *ratings* médios dos itens i e j, então o cálculo da similaridade é dado por [20]:

$$sim(i,j) = \frac{\sum_{u \in U}(r_{u,i} - \overline{r_i})(r_{u,j} - \overline{r_j})}{\sqrt{\sum_{u \in U}(r_{u,i} - \overline{r_i})^2}\sqrt{\sum_{u \in U}(r_{u,j} - \overline{r_j})^2}} \quad (8).$$

Repare-se que a Equação (8) parte do cálculo da similaridade de co-seno subtraindo aos *ratings* originais a média do item, enquanto na Equação (3) se subtrai a média dos *ratings* do utilizador.

Similaridade de co-seno ajustada:

O cálculo desta similaridade é muito semelhante ao da Equação (8), sendo que a única diferença está na subtracção feita aos itens, isto é, enquanto no caso anterior se subtraia ao *rating* do item a média desse item, na similaridade de co-seno ajustada é subtraída a média do utilizador.

Esta medida surgiu como uma melhoria à similaridade de co-seno, uma vez que a última não tem em conta o peso da diferença nas classificações dos *ratings* entre os utilizadores [20]. Problema que pode ser colmatado quando aplicada a similaridade de co-seno ajustada através da subtracção da média do utilizador a cada par de *co-ratings*.

Este cálculo é feito utilizando a Equação (9) [20]:

$$sim(i,j) = \frac{\sum_{u \in U}(r_{u,i} - \overline{r_u})(r_{u,j} - \overline{r_u})}{\sqrt{\sum_{u \in U}(r_{u,i} - \overline{r_u})^2}\sqrt{\sum_{u \in U}(r_{u,j} - \overline{r_u})^2}} \quad (9).$$

Da Equação (9) $\overline{r_u}$ corresponde à média do *rating* do utilizador u.

Cálculo da predição:

Obtidas as similaridades para os itens, o próximo passo será o cálculo do valor da predição para um item ou conjunto de itens. Esta recomendação pode ser calculada utilizando a Equação (10).

$$pred(u,i) = \frac{\sum_{j \in \mathrm{K}} sim(i,j) \times r_{\mathrm{u},j}}{\sum_{j \in \mathrm{K}} |sim(i,j)|} \quad 10).$$

Da Equação (10), i corresponde ao item para o qual se pretende calcular a predição para para o utilizador u, $r_{\mathrm{u},j}$ é a classificação atribuída pelo utilizador u ao item j e K representa a vizinhança de itens similares a i classificados pelo utilizador u.

Note-se que o cálculo da predição apresentada na Equação (10) pode ser afectada por outras condicionantes, como por exemplo, utilizadores que tendem a atribuir classificações muito altas ou muito baixas aos itens, enquanto outros tendem a nunca atribuir classificações muito altas.

O efeito dessa condicionante pode ser diminuído utilizando a Equação (11):

$$pred(u,i) = \overline{r_u} + \frac{\sum_{j \in \mathrm{K}} sim(i,j) \times (r_{u,j} - \overline{r_u})}{\sum_{j \in \mathrm{K}} sim(i,j)} \quad 11)$$

3.2. Algoritmos baseados em Modelo

Ao contrário dos algoritmos baseados em memória os algoritmos baseados em modelo utilizam a informação correspondente às classificações (ou *ratings*) do conjunto de dados de treino para criar um modelo estimado.

Estes algoritmos fazem recomendações baseadas na estimação de parâmetros de modelos estatísticos/probabilísticos para os *ratings* dos utilizadores [20] [32].

Neste trabalho será feito um estudo e análise de algumas técnicas de factorização de matrizes. A factorização matricial é uma das técnicas mais importantes dos sistemas de recomendação baseados em modelo, uma vez que este tipo de abordagem se torna mais vantajosa quando existe uma grande esparsidade dos dados [33].

Esta técnica baseia-se na representação de utilizadores e itens como vectores de preferências/características desconhecidas. Após uma fase de treino destes vectores de preferências

(para o caso dos utilizadores) e de características (para o caso dos itens) é fácil obter uma predição com base no produto interno desses dois vectores.

De seguida apresentam-se duas técnicas: a SVD completa (ou *Full* SVD) e a SVD reduzida (ou *Low-rank* SVD).

3.2.1. SVD Completa

O método de Decomposição em Valores Singulares ou *Singular Value Decomposition* (SVD), é uma técnica de factorização de matrizes que decompõe uma matriz $\mathbf{X}$ de dimensão $m \times n$ de $\text{rank}\, r \leq \min(m, n)$ como um produto de três matrizes, representado pela Equação (12):

$$\mathbf{X} = \mathbf{USV^T} \quad (12).$$

É sempre possível obter-se a decomposição apresentada na Equação (12) qualquer que seja a matriz $\mathbf{X}$. As matrizes $\mathbf{U}$ e $\mathbf{V}$, de dimensão $m \times r$ e $n \times r$, são ortogonais, isto é, $\mathbf{U^T U} = \mathbf{V^T V} = \mathbf{I}$, sendo que $\mathbf{I}$ representa a matriz identidade de dimensão r.

A matriz $\mathbf{S}$ representa uma matriz diagonal $r \times r$ em que todas as entradas são números reais não negativos [34]. As r entradas da diagonal de $\mathbf{S}$ (s_1, s_2, …, s_r) correspondem aos valores singulares. Repare-se que na SVD a ordem dos valores singulares não está determinada porque se se permutarem as entradas da diagonal de $\mathbf{S}$ e se aplicar a mesma permutação às colunas de $\mathbf{U}$ e de $\mathbf{V}$ obtém-se exactamente o mesmo $\mathbf{X}$. Neste trabalho convencionou-se que os valores singulares estão ordenados da seguinte forma:

$$\begin{gathered} s_i > 0 \\ s_1 \geq s_2 \geq \cdots \geq s_r > 0 \end{gathered} \quad (13).$$

As primeiras r colunas de $\mathbf{U}$ e $\mathbf{V}$ representam os vectores próprios associados aos r valores próprios, não nulos, de $\mathbf{XX^T}$ e $\mathbf{X^T X}$, respectivamente [36].

Na Figura 7 encontra-se representado um esquema da SVD completa de uma matriz.

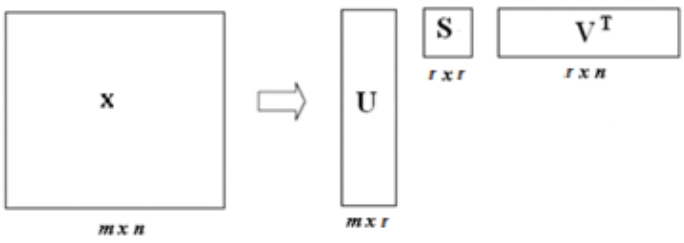

Figura 7 – SVD completa de uma matriz $\mathbf{X}$

De seguida apresenta-se um exemplo de aplicação da SVD completa a uma matriz de *ratings* de utilizadores $\mathbf{X}$:

$$\mathbf{A} = \begin{bmatrix} 5 & 1 & 3 & 4 \\ 2 & 4 & 2 & 3 \\ 5 & 5 & 3 & 4 \\ 2 & 3 & 1 & 2 \end{bmatrix}$$

Às linhas da matriz $\mathbf{X}$ correspondem os utilizadores e às colunas os itens.

A SVD da matriz $\mathbf{X}$[12] é dada pela seguinte decomposição:

$$\mathbf{A} = \begin{bmatrix} -0.51 & 0.8 & 0.2 & 0.2 \\ -0.43 & -0.5 & 0.8 & -0.1 \\ -0.7 & -0.2 & -0.6 & -0.4 \\ -0.3 & -0.3 & -0.2 & 0.8 \end{bmatrix} \begin{bmatrix} 12.8 & 0 & 0 & 0 \\ 0 & 3.4 & 0 & 0 \\ 0 & 0 & 0.9 & 0 \\ 0 & 0 & 0 & 0.3 \end{bmatrix} \begin{bmatrix} -0.6 & 0.5 & -0.7 & 0.1 \\ -0.5 & -0.8 & -0.2 & 0.1 \\ -0.4 & 0.2 & 0.4 & -0.8 \\ -0.5 & 0.2 & 0.6 & 0.6 \end{bmatrix}$$

Repare-se que os valores singulares da matriz central da decomposição de $\mathbf{X}$ estão ordenados por ordem decrescente, verificando-se que o peso das duas primeiras entradas dessa matriz são mais significativas do que as restantes.

A SVD completa não pode ser aplicada em recomendação porque num sistema real não é possível conhecer todas as entradas da matriz de *ratings* dos utilizadores. Não conhecendo todas as entradas da matriz tem-se que a SVD completa não é definida.

3.2.2. SVD Reduzida

A decomposição em valores singulares pode ser usada para obter aproximações de característica ou $rank$ inferior à matriz original [35]. Esta propriedade torna possível a redução do $rank$ da matriz através da selecção dos k valores singulares mais elevados.

As matrizes obtidas pelo cálculo da SVD são extremamente úteis uma vez que existe uma propriedade da SVD que nos garante a melhor aproximação de $rank$ inferior de uma matriz segundo a norma de Frobenius [35] [36]. Esta aproximação é possível mantendo os $k \ll r$ valores singulares e descartando as restantes entradas, resultando numa matriz reduzida $\mathbf{S}_k$ [36]. Desde que as entradas de $\mathbf{S}$ estejam ordenadas, como apresentado na Equação (13), o processo de redução ocorre retendo os primeiros k valores singulares, obtendo-se então $\mathbf{S}_k$ como apresentado na Figura 8 [36].

[12] Utilizou-se o software MATLAB no cálculo da SVD da matriz $\mathbf{X}$.

A norma de Frobenius é definida por:

$$\|\mathbf{X}\|_{\mathrm{F}} = \sqrt{\sum_{i=1}^{\min\{m,n\}} {s_i}^2} = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}\left|x_{ij}\right|^2} \qquad (14).$$

Da Equação (14), s_i corresponde aos valores singulares de $\mathbf{X}$, e x_{ij} corresponde aos elementos da matriz.

Se as matrizes $\mathbf{U}$ e $\mathbf{V}$ forem reduzidas da mesma forma que $\mathbf{S}$ obtém-se $\mathbf{U_k}$ e $\mathbf{V_k}$, ou seja, $\mathbf{U_k}$ resulta da remoção das últimas $r-k$ colunas de $\mathbf{U}$, enquanto $\mathbf{V_k}$ resulta da remoção das últimas $r-k$ linhas da matriz $\mathbf{V^T}$ [36]. Multiplicando as três matrizes reduzidas $\mathbf{U_k}$, $\mathbf{S_k}$ e $\mathbf{V_k}$ obtém-se a matriz $\mathbf{X_k}$ dada pela Equação (15).

$$\mathbf{X_k} = \mathbf{U_k S_k V^T}_{\boldsymbol{k}} \qquad (15).$$

A SVD de $rank\ k$ minimiza a distância entre $\mathbf{X}$ e $\mathbf{X_k}$, isto é, $\mathbf{X_k}$ é a matriz reduzida aproximada de $rank\ k$ mais próxima da matriz original $\mathbf{X}$, dada pela Equação (16):

$$\|\mathbf{X} - \mathbf{X_k}\|_{\mathrm{F}} = \sqrt{\sum_{i=k+1}^{\mathrm{r}} {s_i}^2} = \sqrt{{s_{k+1}}^2 + \cdots + {s_r}^2} \qquad (16).$$

De seguida apresenta-se graficamente o processo de redução de uma matriz:

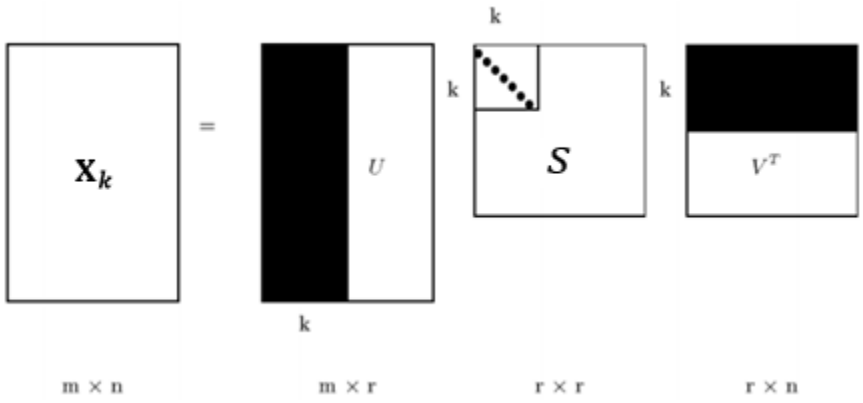

Figura 8 – SVD reduzida da matriz $\mathbf{X}$

Para a matriz $\mathbf{A}$ apresentada anteriormente, o processo de redução de $\mathbf{A}$ para um valor de $k = 2$, ou seja, $\mathbf{A_{k=2}}$ é dado por:

$$\mathbf{A} \approx \mathbf{A_{k=2}} = \begin{bmatrix} -0.5 & 0.8 \\ -0.4 & -0.5 \\ -0.7 & -0.2 \\ -0.3 & -0.3 \end{bmatrix} \begin{bmatrix} 12.8 & 0 \\ 0 & 3.4 \end{bmatrix} \begin{bmatrix} -0.6 & 0.5 & -0.7 & 0.1 \\ -0.5 & -0.8 & -0.2 & 0.1 \end{bmatrix}$$

$$\mathbf{A_{k=2}} = \begin{bmatrix} -0.5 & 0.8 \\ -0.4 & -0.5 \\ -0.7 & -0.2 \\ -0.3 & -0.3 \end{bmatrix} \begin{bmatrix} -7.6 & 6.4 & -8.9 & 1.3 \\ -1.7 & -2.7 & -0.7 & 0.3 \end{bmatrix}$$

A matriz que se obtém pelo processo de redução de $\mathbf{A}$, é aproximadamente idêntica à matriz original [35], no entanto, alguns autores [36] afirmam que a matriz reduzida possui mais informação qualitativa que a original uma vez que foram removidos os valores singulares mais baixos.

Seja $\mathbf{R}$ a matriz que contém as classificações atribuídas pelos utilizadores aos itens, onde as linhas da matriz representam os utilizadores e as colunas da matriz $\mathbf{R}$ representam os itens. Ao aplicar a SVD a esta matriz tem-se que:

$$\mathbf{R}_{\mathbf{m}\times\mathbf{n}} \approx \mathbf{U}_{\mathbf{m}\times\mathbf{r}}\mathbf{V'}^{\mathbf{T}}_{\mathbf{r}\times\mathbf{n}} \quad 17).$$

Repare-se que se passou a ter uma matriz $\mathbf{V'}^{\mathbf{T}}$ em virtude da multiplicação de $\mathbf{S}_{\boldsymbol{k}}$ por $\mathbf{V}^{\mathbf{T}}$. Neste caso optou-se por multiplicar $\mathbf{S}$ por $\mathbf{V}^{\mathbf{T}}$, no entanto alguns autores [36] recorrem a uma decomposição de $\mathbf{S}$ como o produto de duas matrizes, pelo que:

$$\boldsymbol{S_k} = \sqrt{\boldsymbol{S_k}}^{\boldsymbol{T}}\sqrt{\boldsymbol{S_k}} \quad 18).$$

Logo, a matriz $\mathbf{R}_{\mathbf{m}\times\mathbf{n}}$ será dada por:

$$\mathbf{R}_{\mathbf{m}\times\mathbf{n}} \approx \mathbf{U}_{\boldsymbol{mk}}\sqrt{\boldsymbol{S_{kk}}}^{\boldsymbol{T}}\sqrt{\boldsymbol{S_{kk}}}\mathbf{V}^{\mathbf{T}}{}_{\boldsymbol{kn}} \quad 19).$$

3.2.3. Técnica Implementada baseada em Factorização de Matrizes

Motivação:

Considere-se como exemplo o conjunto de dados do *Netflix* constituído por 17770 filmes e aproximadamente 480 mil utilizadores.

Representando toda esta informação numa matriz, e admitindo que os utilizadores são representados através das linhas da matriz e os filmes das colunas, cada elemento da matriz representa uma classificação desse utilizador ao item.

Sabendo que são conhecidas 100 mil entradas da matriz num total de 8.5 mil milhões, aproximadamente, significa que se tem conhecimento de apenas 1% da informação global, ou seja, a matriz é 99% esparsa[13].

Como foi referido na Secção 3.2.1 a implementação de uma SVD não é exequível pois não são conhecidas todas as entradas da matriz de *ratings*, factor necessário para que se possa calcular a SVD de uma matriz [37]. Supondo que eram conhecidas todas as entradas da matriz de *ratings*, o cálculo da SVD completa para um conjunto de dados com uma dimensão tão elevada (como o da *Netflix*) seria penosa, tornando a decomposição dessas matrizes quase impraticável [49].

Como a complexidade do cálculo da SVD é $O(mn^2)$, em que m é a dimensão menor da matriz e n a dimensão maior, e porque não se conhecem todas as entradas da matriz optou-se por

[13] Ao contrário da terminologia usual, neste trabalho esparsidade refere-se a um grande número de entradas não conhecidas da matriz.

implementar uma técnica que não é uma SVD mas que consiste na factorização de uma matriz no produto de duas matrizes como se pode constatar nos parágrafos seguintes. Esta técnica permite colmatar os problemas apresentados anteriormente para o caso em que se aplicasse a SVD à matriz.

Esta técnica baseada em factorização de matrizes parte de uma solução apresentada por um ex-concorrente da competição da *Netflix* (*Netflix Prize*), Simon Funk [24].

Definição do Problema:

Partindo do pressuposto que não se conhece nada sobre factorização matricial, mais especificamente sobre SVD, seria vantajoso conseguir representar um item (exemplo: um filme) a partir de um conjunto de características que variasse consoante os gostos dos utilizadores. Deste modo, cada utilizador atribuía um peso superior às características que considerava mais importantes num filme (exemplo: se o filme é de acção, comédia, romance, entre outros), e da mesma forma para as preferências de um utilizador (exemplo: se um utilizador prefere filmes de acção ou comédia).

Partindo desta ideia, é possível obter um valor para a predição dos *ratings* dos itens ainda não classificados através de uma combinação entre as preferências dos utilizadores e as características dos itens.

Neste trabalho, cada filme é representado por um vector $\mathbf{v}_j \in \mathbb{R}^f$, que possui um conjunto de f características que o definem, . Cada utilizador é representado por um vector $\mathbf{u}_i \in \mathbb{R}^f$ que possui um conjunto de preferências f a ele associadas [24].

Para um *triplet* da forma $(utilizador, item, r)$ a predição calculada pelo sistema para o *rating* r é dada pela Equação (20):

$$\hat{r}_{ij} = \mathbf{u}_i{}^{\mathrm{T}}\, \mathbf{v}_j \qquad (20).$$

Da Equação (20), $\hat{r}_{ij}$ representa uma estimativa do *rating* calculada através do produto interno dos vectores de preferências dos utilizadores, $\mathbf{u}_i{}^{\mathrm{T}}$, com os vectores de características dos itens, $\boldsymbol{v}_j$.

Portanto o desafio apresentado neste trabalho passa por calcular as características e preferências dos vectores $\mathbf{v}_j$ e $\mathbf{u}_i$ para cada item/utilizador, respectivamente, através da minimização da Equação (21) [53]:

$$\frac{1}{2}\|\mathbf{R} - \mathbf{U}^{\mathbf{T}}\mathbf{V}\|_{\mathrm{F}}^2 \qquad (21).$$

Da Equação (21), $\|\cdot\|_{\mathrm{F}}^2$ representa a norma de Frobenius, $\mathbf{R}$ a matriz de *ratings* com toda a informação do conjunto de dados, $\mathbf{U}^{\mathbf{T}}$ a matriz com todos os vectores de preferências de utilizadores, $\mathbf{u}^{\mathbf{T}}$, e $\mathbf{V}$ a matriz com todos os vectores de características dos itens, $\mathbf{v}$.

Como a matriz de *ratings* $\mathbf{R}$ é esparsa, a minimização só pode ser efectuada para os elementos conhecidos da matriz, pelo que a Equação (21) se traduz na forma [53]:

$$\mathrm{E} = \min_{\mathrm{U,V}} \frac{1}{2} \sum_{(i,j)\in\psi} (\mathrm{R}_{ij} - \mathrm{U}_i{}^{\mathrm{T}}\, V_j)^2 \qquad (22).$$

Da Equação (22), ψ representa o conjunto de pares (i, j) para o qual r_{ij} é conhecido.

Por forma a prevenir/evitar o efeito de *overfitting*[14] foram introduzidos à Equação (22), dois coeficientes de regularização, λ_{u}, $\lambda_{\mathrm{v}} > 0$, dando origem à Equação (23) [53].

$$\mathrm{E} = \min_{\mathrm{U,V}} \frac{1}{2} \sum_{(i,j)\in\psi} (\mathrm{R}_{ij} - \mathrm{U}_i{}^{\mathrm{T}} V_j)^2 + \frac{\lambda_{\mathrm{u}}}{2} \sum_{i=1}^{n} \|\mathrm{U}_i\|^2 + \frac{\lambda_{\mathrm{v}}}{2} \sum_{j=1}^{m} \|\mathrm{V}_j\|^2 \qquad (23).$$

A minimização das Equações (22) e (23) é realizada através do cálculo do gradiente da função de custo, a partir do qual o algoritmo dá passos na direcção oposta à do gradiente. A optimização de **U** e **V** encontra-se representada pelas Equações (24) e (25).

$$-\frac{\partial \mathrm{E}}{\partial U_i} = \sum_{j=1}^{m} (\mathrm{R}_{ij} - \mathrm{U}_i{}^{\mathrm{T}} \mathrm{V}_j)\mathrm{V}_j - \lambda_{\mathrm{u}} \mathrm{U}_i \,, i = 1, \dots, n, \qquad (24).$$

$$-\frac{\partial \mathrm{E}}{\partial V_j} = \sum_{i=1}^{n} (\mathrm{R}_{ij} - \mathrm{U}_i{}^{\mathrm{T}} \mathrm{V}_j)\mathrm{U}_i - \lambda_{\mathrm{v}} \mathrm{V}_j \,, j = 1, \dots, m, \qquad (25).$$

Para o caso em que não há regularização, os termos λ_{u} e λ_{v} são nulos.

Até agora foi apresentada a forma como é feita a optimização de **U** e **V**, no entanto não foi feita nenhuma referência à forma como é feita a actualização destas duas matrizes pelo algoritmo. As Equações (26) e (27) representam as funções que caracterizam a actualização de **U** e **V** durante a fase de treino do algoritmo, dada por:

$$\mathrm{U}_{ij}^{\mathrm{t+1}} = \mathrm{U}_{ij}^{\mathrm{t}} + \gamma * \left[(\mathrm{R}_{ij} - \mathrm{U}_i{}^{\mathrm{T}} \mathrm{V}_j)\mathrm{V}_j - \lambda_{\mathrm{u}} \mathrm{U}_i \right] \qquad (28).$$

$$\mathrm{V}_{ij}^{\mathrm{t+1}} = \mathrm{V}_{ij}^{\mathrm{t}} + \gamma * \left[(\mathrm{R}_{ij} - \mathrm{U}_i{}^{\mathrm{T}} \mathrm{V}_j)\mathrm{U}_i - \lambda_{\mathrm{v}} \mathrm{V}_j \right] \qquad (29).$$

Das Equações (30) e (31), γ representa a taxa de aprendizagem (*learning rate*).

De uma forma compacta o algoritmo implementado neste trabalho pode ser representado por:

Algoritmo 1 – Algoritmo baseado em SVD

1. Selecção dos valores de γ, λ_{u}, λ_{v}, max_epoch, f_{max}
2. $\mathrm{U} \leftarrow inicializa\_matriz(\mathrm{U})$, $\mathrm{V} \leftarrow inicializa\_matriz(\mathrm{V})$
3. **Para** $f < f_{max}$
4. **Para** $epoch < max\_epoch$
5. (a) Calcula os gradientes ∇_{U}, ∇_{V} através de (24) e (25)
6. (b) Define $\mathrm{U} \leftarrow \mathrm{U} - \gamma\nabla_{\mathrm{U}}$, $\mathrm{V} \leftarrow \mathrm{V} - \gamma\nabla_{\mathrm{V}}$
7. (c) Até que a RMSE comece a crescer
8. **fim do 1º ciclo**
9. **fim do 2º ciclo**

[14] Overfitting – Ocorre quando um sistema obtém bons resultados utilizando o conjunto de treino e maus resultados com o conjunto de teste.

Do Algoritmo 1 a variável max_epoch representa o número máximo de épocas que o algoritmo tem que executar durante o treino de cada preferência/característica f (caso se trate de $\mathbf{U}$ ou $\mathbf{V}$, respectivamente). A variável f_{max} representa o número máximo de características de $\mathbf{U}$ e $\mathbf{V}$.

O ciclo de épocas executado pelo algoritmo pode ser inferior ao número de épocas máximo se a métrica de erro, RMSE, for relativamente superior à obtida anteriormente (a diferença entre as RMSE pode ser controlada a partir de uma constante inicialmente definida, por exemplo, sempre que a diferença entre as métricas for de 0.0001 então o algoritmo passa para a próxima característica).

A função $inicializa\_matriz$ corresponde ao preenchimento inicial dos valores das matrizes $\mathbf{U}$ e $\mathbf{V}$, que pode ser dado por um valor fixo, tal como sugerido por Simon Funk [24], ou por um valor aleatório entre 1 e 5.

Após o algoritmo 1 ter realizado todos os passos, a predição final dos itens é feita através do produto interno de $\mathbf{U}$ com $\mathbf{V}$, Equação (20). Como o resultado do produto interno pode ser um valor inferior a 1 ou superior a 5, o sistema estipula um valor de limiar dado por 1 ou 5, respectivamente. Isto permite que o sistema obtenha melhores valores de RMSE uma vez que o valor da diferença entre o *rating* original e o estimado é inferior relativamente aos casos em que o valor estimado se encontra fora do intervalo de classificação considerando.

Repare-se ainda que os parâmetros γ, λ_{u}, λ_{v}, f_{max}, são ajustáveis e podem condicionar a performance do sistema assim como o valor dos *ratings* estimados. A influência de cada um destes parâmetros será apresentada na Secção 4.3.4.

4. Resultados Experimentais

4.1. Descrição dos dados

Neste trabalho foram utilizados dois conjuntos de dados distintos para a aplicação dos métodos enunciados nas Secções 3.1.1 e 3.2.3.

Plataforma Experimental – Toda a análise e interpretação de dados, assim como a implementação das técnicas aqui apresentadas foram realizadas num computador intel i5 com 6GB de RAM. Os programas foram executados utilizando exclusivamente a linguagem Python.

O sistema operativo utilizado foi o Linux.

4.1.1. Netflix

De acordo com a informação disponibilizada no *site* da *Netflix*, o conjunto de dados é constituído por todos os *ratings* recolhidos entre Outubro de 1998 e Dezembro de 2005.

O conjunto de dados da *Netflix* é composto por 4 ficheiros:

Training set – Corresponde ao conjunto de dados de treino e é constituído por 100,480,507 *ratings* escolhidos aleatoriamente a partir de um conjunto de 480,189 utilizadores e de 17,770 filmes classificados entre 1 e 5. Os dados de treino são apresentados por q*uadruplets* da forma <utilizador, nome do filme, data da cotação, cotação atribuída>[15].

Probe set – Este conjunto de dados é constituído por 1,408,395 *ratings* e tem a função de conjunto de teste ou *test set*. É com base neste conjunto de dados que é feita a comparação dos *ratings* estimados pelo algoritmo e os *ratings* reais, sendo que é com essa informação que é avaliada a qualidade do sistema utilizando métricas de avaliação como a RMSE ou MAE[16].

Qualifying set - O conjunto de dados de validação contém 2,817,131 *triplets* na forma <utilizador, filme, data da cotação>. Os *ratings* deste conjunto não eram conhecidos pelos concorrente da competição[17]. Este conjunto de dados era utilizado pelos júris da competição para medir a qualidade do algoritmo de cada equipa a partir das predições calculadas.

Movie titles – Ficheiro que contém o nome e a data de lançamento de cada filme.

Como o conjunto de dados de treino tem aproximadamente 2 Gbytes de tamanho, decidiu fazer-se uma partição dos dados num conjunto menor.

Este novo subconjunto de dados da *Netflix* utilizado na implementação dos algoritmos neste trabalho é constituído por 3000 itens e aproximadamente 6 mil utilizadores escolhidos aleatoriamente do conjunto de dados original.

[15] Neste trabalho não será utilizada a data da cotação.
[16] Estas métricas de avaliação encontram-se definidas na Secção 4.2.
[17] Como as cotações não são conhecidas este conjunto de dados não foi utilizado neste trabalho.

Como se encontra representado na Figura 9, utilizou-se um pequeno subconjunto de validação/desenvolvimento ou *development set.* Este subconjunto foi criado com o objectivo de estudar os parâmetros do sistema para o qual se podem obter os melhores resultados, isto é, para que parâmetros o algoritmo calcula melhores predições.

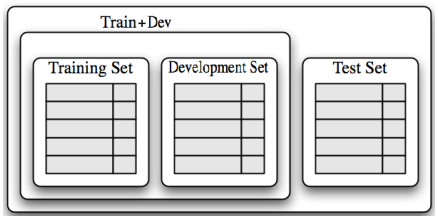

Figura 9 - Representação da divisão do conjunto de dados original num conjunto de treino, validação e teste.

4.1.2. MovieLens

O conjunto de dados utilizado foi recolhido entre Setembro de 1997 e Abril de 1998 e é composto por:

- 100,000 *ratings* contidos num intervalo de 1 a 5
- 943 utilizadores e 1682 filmes
- Cada utilizador classificou pelo menos 20 filmes. Todos os utilizadores que tenham classificado menos que este número não foram considerados.
- Informação dos utilizadores (idade, sexo e ocupação)[18].

Aplicando a separação dos *ratings* totais em três conjuntos de dados como representado na Figura 9, obtiveram-se 64000 *ratings* para o conjunto de treino, 16000 para validação e 20000 para teste. Como a divisão do conjunto de dados realizada não é a mesma que a efectuada em [20], os resultados obtidos não são completamente generalizáveis.

Uma alternativa à solução apresentada na Figura 9, e utilizada neste trabalho, seria a utilização de um método de estimação de parâmetros (*k-fold cross validation ou validação cruzada*) como o realizado num dos artigos seguidos como referência para a realização deste trabalho [20].

Este método consiste na divisão de um conjunto de dados em *k* subconjuntos mutuamente exclusivos que podem ou não ser do mesmo tamanho. Desses subconjuntos, um deles é utilizado para teste e os restantes *k-1* para treino (ou estimação de parâmetros), e vão sendo calculadas medidas de avaliação das predições face aos *ratings* reais conhecidos. E o processo repete-se *k* vezes como apresentado na Figura 10. Feitas as *k* validações é escolhido o parâmetro que oferece ao sistema resultado com melhor qualidade, ou seja, quando as predições calculadas se aproximam do valor real do *rating*.

[18] À semelhança do conjunto de dados da *Netflix*, neste trabalho utilizou-se apenas o ID do utilizador, o ID do item e o respectivo *rating*.

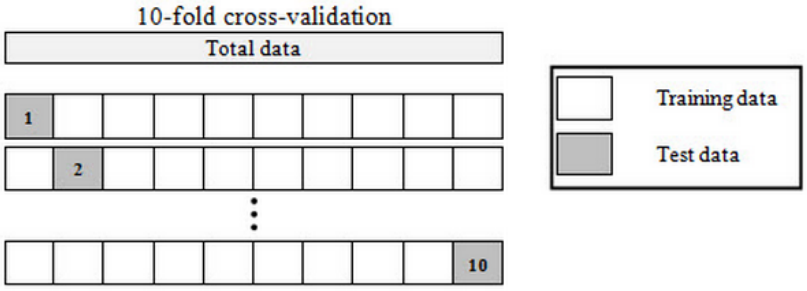

Figura 10 - Exemplo de validação cruzada de 10 ou *10-fold cross validation*

4.2. Métricas de Avaliação

A qualidade das recomendações varia com a interpretação do mesmo face às similaridades calculadas pelos algoritmos de recomendação, mas de que forma é o sistema capaz de interpretar as similaridades entre utilizadores ou itens e concluir que possivelmente o item recomendado será do interesse do utilizador?

A resposta passa pela existência de métricas de avaliação que servem para verificar quão bem as previsões calculadas pelo algoritmo se ajustam aos *ratings* reais do conjunto de desenvolvimento. Por outras palavras, o cálculo das métricas de erro pode ser feito a partir da comparação do *rating* atribuído por um utilizador a um item com o *rating* que o sistema prevê que seja a classificação atribuída por esse utilizador ao item que se pretende recomendar.

Neste trabalho foram utilizadas duas das métricas mais utilizadas na avaliação dos sistemas de recomendação [22].

- Erro médio absoluto ou *mean absolute error* (MAE)
- Raiz quadrada do erro quadrático médio ou *root mean square error* (RMSE)

4.2.1. RMSE

A RMSE é talvez a métrica de desempenho mais utilizada na avaliação dos sistemas de recomendação [22].

O sistema de recomendação gera uma predição de *rating* dado por $\hat{r}_{ui}$ para um par utilizador-item (u, i), que pertence ao conjunto de teste τ, em que $|\tau|$ é o número de *ratings*. Como se conhece o verdadeiro *rating,* r_{ui}, atribuído pelo utilizador u ao item i, tem-se que o cálculo da RMSE é dado por:

$$RMSE = \sqrt{\frac{1}{|\tau|} \sum_{(u,i)\in\tau} (\hat{r}_{ui} - r_{ui})^2} \quad 32).$$

Quanto menor for o valor da RMSE maior será a qualidade dos *ratings* previstos pelo algoritmo.

4.2.2. MAE

Como alternativa à RMSE temos o MAE, dado por:

$$MAE = \frac{\sum_{(u,i)\in\tau} |\hat{r}_{ui} - r_{ui}|}{|\tau|} \quad 33).$$

À semelhança da RMSE, quanto menor a MAE melhor é a qualidade do algoritmo na predição de *ratings*.

De seguida apresenta-se na Tabela 4 um exemplo que destaca a diferença entre as duas métricas, RMSE e MAE.

Comparativamente com o MAE, a RMSE penaliza desproporcionalmente erros grandes, de modo que, dado um conjunto de teste com quatro itens desconhecidos, a RMSE prefere um sistema que comete um erro de dois em três *ratings* e zero no quarto do que um erro de quatro num dos *ratings* e zero nos restantes, enquanto a MAE prefere o segundo sistema como se pode verificar pelo exemplo da Tabela 4 [22].

	Caso 1	Caso 2
Erro 1	2	4
Erro 2	2	0
Erro 3	2	0
Erro 4	0	0
RMSE	$\sqrt{3} = 1.73$	2
MAE	1.50	1

Tabela 4 – Exemplo de comparação da penalização do erro utilizando a RMSE e o MAE.

4.3. Procedimento Experimental

4.3.1. Comportamento dos dados

Antes de se apresentarem os resultados para as técnicas propostas na Secção 3, é importante estudar a distribuição dos dados de um sistema para posteriormente se perceber de que forma essa distribuição influencia os resultados obtidos.

Para o conjunto de dados do *MovieLens* e da *Netflix*, admitindo que existe um grupo de *ratings* conhecidos por $\mathbf{R} = \{r_{ij}\}$ atribuídos por um conjunto de utilizadores $\mathbf{C}$ a um conjunto de itens $\mathbf{I}$, ordenados de acordo com o número de *ratings* atribuídos pelos utilizadores, tem-se que os dados

seguem uma distribuição semelhante a uma cauda longa, ou *Long Tail* [40] representado na Figura 11.

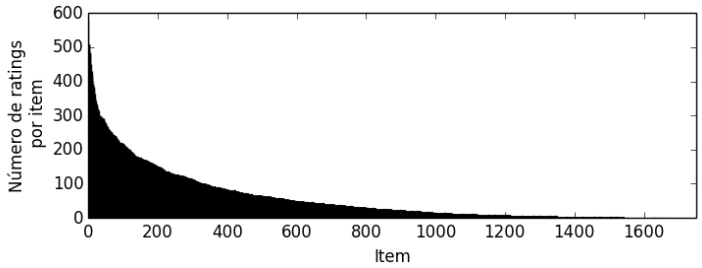

Figura 11 – Representação do número de *ratings* por item para o conjunto de dados do *MovieLens*.

Para a amostra de dados da *Netflix* a distribuição do número de *ratings* por item é dada pela Figura 12.

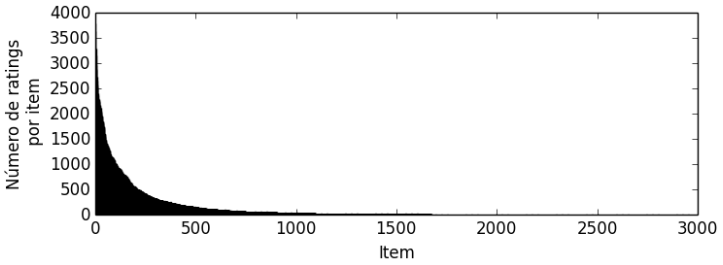

Figura 12 – Representação do número de *ratings* por item para a amostra da *Netflix*.

Como seria de esperar o comportamento dos dados do *MovieLens* é semelhante ao da amostra da *Netflix*.

Long Tail – Definição

O termo *long tail* ou cauda longa foi popularizado por Chris Anderson numa publicação de um artigo em 2004 na revista Wired [42] e posteriormente com o lançamento do livro "The Long Tail: Why the future of business is selling less of more" em 2006 [41].

O fenómeno cauda longa é medido através de uma distribuição de frequência (ou valores de uma amostra), isto é, para cada item existe uma contagem do número de *ratings* atribuídos por uma população ou grupo de utilizadores (exemplo, compras, downloads, etc).

Segundo Chris Anderson, produtos com pouca procura ou com um reduzido volume de vendas podem, colectivamente, atingir uma quota de mercado que rivaliza ou ultrapassa a dos produtos mais vendidos em menores quantidades. É por isso que o autor considera que existem duas partes distintas no gráfico: a cabeça e a cauda.

A cabeça da distribuição corresponde ao mercado de produtos mais populares enquanto a cauda corresponde ao pequeno nicho de mercado, que emergiu com o aparecimento da internet [41]. Um estudo mais completo desta distribuição pode ser encontrado em [41].

Embora o conceito de cauda longa esteja associado maioritariamente ao estudo dos mercados económicos, também é possível extrair alguma informação relativa aos sistemas de recomendação.

Observando as Figuras 11 e 12, é possível verificar que a distribuição de *ratings* é quase sempre muito enviesada/distorcida, ou seja, enquanto uma pequena parte dos itens tendem a obter a maioria dos votos dos utilizadores, uma outra parte, pertencente à cauda longa, tende a receber poucas classificações por item.

Um outro dado estatístico relevante para o estudo da predição e recomendação de *ratings* é o cálculo das médias do número de *ratings* atribuídos pelos utilizadores e a média de classificação dos itens. O gráfico da Figura 13 ilustra a distribuição dos *ratings* médios para cada item.

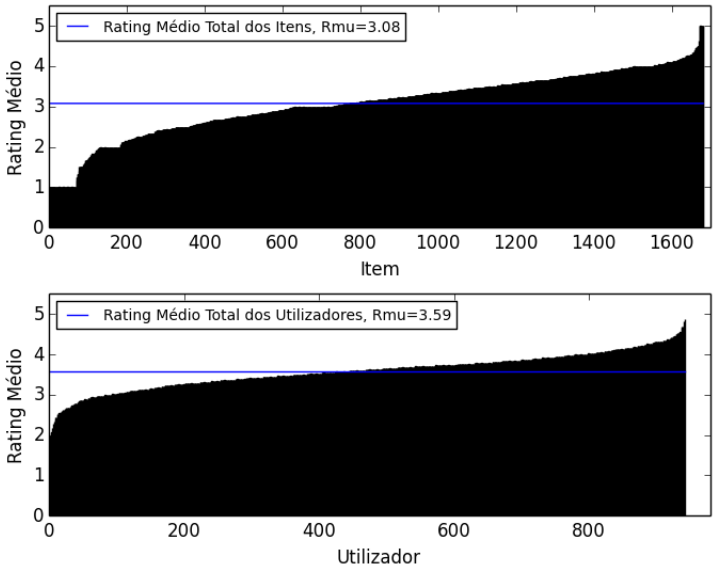

Figura 13 – Distribuição dos *ratings* médios dos itens e utilizadores para o *MovieLens*.

Na Figura 13 está representada a distribuição dos *ratings* médios dos itens e dos utilizadores para o subconjunto de dados da *Netflix*.

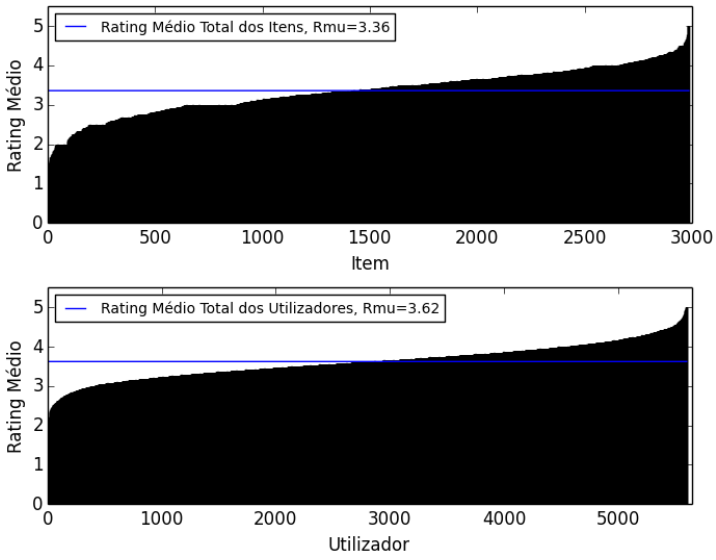

Figura 14 – Distribuição dos *ratings* médios dos itens e utilizadores para a *Netflix*.

Repare-se que na Figura 13 existe um grupo de itens classificados com *rating* médio de 1, porém não são conhecidas as razões que levem a apontar algum erro na recolha de dados uma vez que o conjunto de dados utilizado é o conjunto de dados total fornecido pelo *GroupLens* [4].

4.3.2. Pré-processamento dos dados

Antes de se apresentarem os resultados obtidos na implementação dos algoritmos referidos na Secção 3, é importante referir que os tempos de processamento mencionados para as diferentes técnicas correspondem apenas ao tempo de processamento do algoritmo implementado. Ou seja, existe uma fase de pré-processamento a partir do qual é feita uma leitura e análise inicial dos dados, cujo tempo não é contabilizado.

O pré-processamento dos dados serve essencialmente para alocar os dados em estruturas que permitam uma leitura mais rápida sempre que seja necessário executar o mesmo algoritmo múltiplas vezes, porém é mantida a estrutura de dados da matriz de *ratings*. Como neste trabalho a linguagem de programação foi exclusivamente Python, optou-se por utilizar o pacote Cpickle na medida em que este permite gravar estruturas de dados (como dicionários, tabelas, vectores, entre outras) num único ficheiro poupando tempo de processamento ao sistema.

Na tabela abaixo apresentam-se os tempos de pré-processamento dos dados antes e depois da criação das estruturas acima referidas utilizando o pacote Cpickle.

	MovieLens	*Netflix*
	Tempo de Processamento (s)	
Leitura dos dados não optimizada	3.54	7.15
Leitura dos dados usando o pacote Cpickle	0.23	0.78

Tabela 5 - Tempos de leitura dos dados com e sem optimização

É de salientar que os tempos

4.3.3. Implementação de Algoritmos Colaborativos baseados em Memória

Foram apresentadas na Secção 3.2 algumas das técnicas mais utilizadas de filtragem colaborativa baseada em memória. Uma delas foi o $k-\mathrm{NN}$ (k- Nearest Neighbor), ou k vizinhos mais próximos. Dentro desta classe de algoritmos apresentaram-se os algoritmos baseados na relação entre utilizadores e na relação entre itens.

De seguida, apresenta-se para cada uma das técnicas as principais diferenças assim como uma análise e comparação dos resultados obtidos com os apresentados pelos autores em [20].

Técnica baseada na relação entre utilizadores:

Como foi referido na Secção 3.1.1 estes algoritmos baseiam-se na relação entre utilizadores que tenham classificados itens em comum, e com base no cálculo das similaridades entre utilizadores o sistema calcula a vizinhança do utilizador activo.

Tal como referido na Secção 3.1.1, o cálculo do tamanho da vizinhança, *k*, condiciona a qualidade das predições calculadas como se pode verificar pela Figura 15. A qualidade das predições, medida a partir dos valores do MAE e da RMSE[19], foram obtidas através da variação do valor de *k* no conjunto de desenvolvimento do *MovieLens* como se descreveu na Secção 4.1.

Para cada experiência abaixo realizada, assim que se obtiver o melhor valor de *k* serão reportados os valores obtidos usando esse valor de *k* no conjunto de teste.

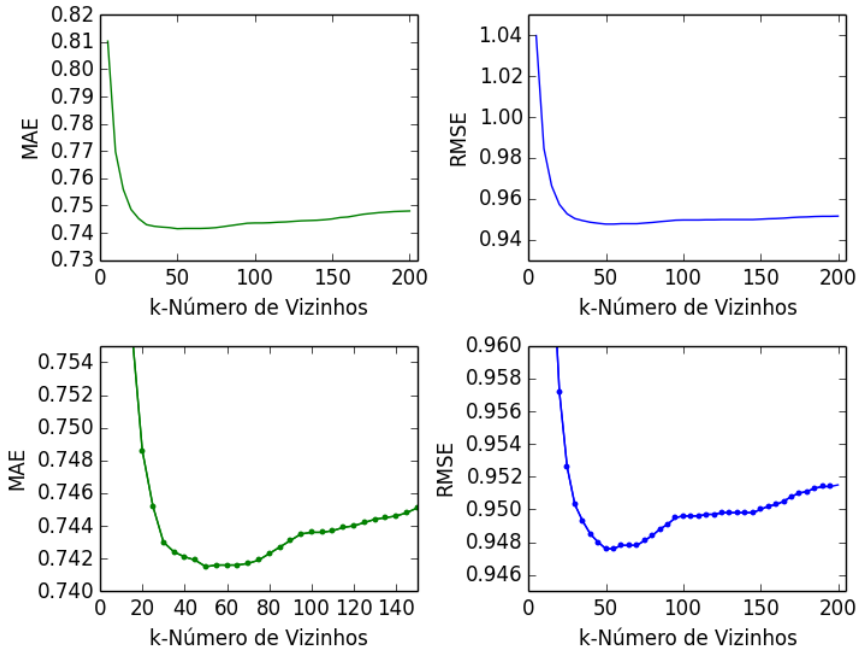

Figura 15 - Sensibilidade das predições ao tamanho da vizinhança, no conjunto de dados do *MovieLens*. O primeiro gráfico representa os valores do MAE e da RMSE até *k* =200. Os gráficos da linha de baixo são uma ampliação do primeiro.

Para o conjunto de dados da *Netflix* a qualidade das predições com a variação de *k* encontra-se representado na Figura 16.

[19] Neste trabalho o valor de *k* correspondente à melhor métrica do MAE será o valor a considerar, e não o menor valor da RMSE uma vez que o artigo seguido como referência [20] para a Secção 3.1 faz a análise dos seus resultados baseada unicamente no MAE.

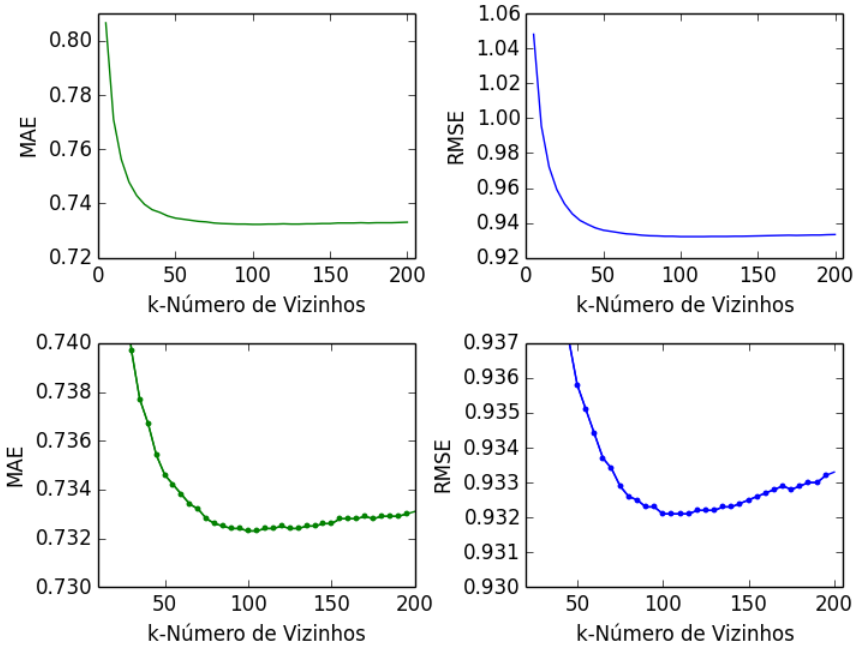

Figura 16 - Sensibilidade das predições ao tamanho da vizinhança, no conjunto de dados da *Netflix*. O primeiro gráfico representa os valores do MAE e da RMSE até *k* =200. Os gráficos da linha de baixo são uma ampliação do primeiro.

Discussão dos resultados:

Como se pode observar pela Figura 15, o número de vizinhos tem um impacto significativo na qualidade da predição [20], isto é, a qualidade da predição aumenta com *k* para valores de k contidos entre 10 e 50. A partir de *k*=50 a qualidade da predição volta a diminuir, ou seja, ter muitos vizinhos nem sempre é um bom indicador de que as predições resultantes sejam melhores, porém ter poucos vizinhos pode causar limitações ao sistema, como foi referido na Secção 2.4, como é o caso do *cold start*. O melhor valor de MAE ocorre para *k=50*.

Para o caso da *Netflix*, a qualidade das predições vai aumentando até *k*=100, sendo que a partir desse valor de *k* o MAE e a RMSE aumentam ligeiramente acabado por estabilizar a partir de um certo ponto.

Conhecido o melhor valor de *k*, obtido no conjunto de desenvolvimento (*dev*), esse será o *k* a utilizar pelo algoritmo no cálculo das predições para o conjunto de teste.

Na Figura 17 apresentam-se os valores do MAE e da RMSE obtidos no conjunto de desenvolvimento e teste para o melhor valor de *k,* para o *MovieLens*.

No cálculo do MAE e da RMSE, para ambos os conjuntos de dados, utilizou-se a correlação de *Pearson* como cálculo da similaridade entre utilizadores. Das medidas de similaridade apresentadas na Secção 3.1.1 o comportamento do MAE e da RMSE face à variação de *k* é semelhante ao verificado nas Figuras 15 e 16, pelo que as restantes medidas de similaridade abordadas na Secção 3.1.1 se encontram representadas nas Figuras 17 e 18.

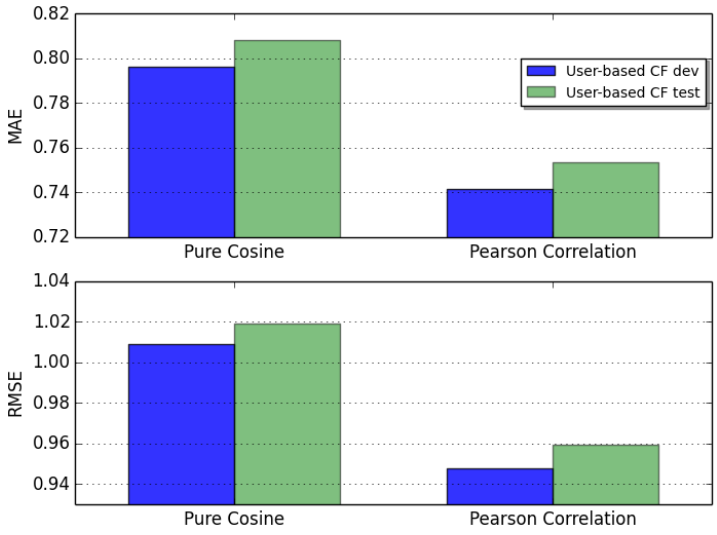

Figura 17 – Comparação dos resultados obtidos para o MAE e RMSE para diferentes medidas de similaridade, para o *MovieLens*.

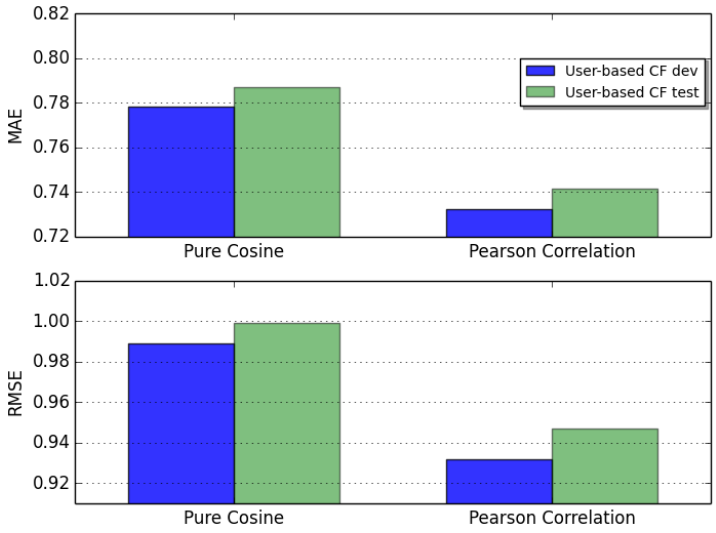

Figura 18 - Comparação dos resultados obtidos para o MAE e RMSE para diferentes medidas de similaridade, para a *Netflix*.

Na Tabela 6 apresentam-se alguns dados adicionais das medidas de similaridade apresentadas nas Figuras 17 e 18.

		Cosine Similarity	Pearson Correlation
MovieLens	Número de experiências	200	
	Tempo médio de processamento em segundos	5,4	6,1
	Melhor valor de *k* obtido no conjunto *dev*	65	50
Netflix	Número de experiências	200	
	Tempo médio de processamento em segundos	103	77
	Melhor valor de *k* obtido no conjunto *dev*	20	100

Tabela 6 – Dados adicionais obtidos para as diferentes medidas de similaridade para o conjunto de dados do *MovieLens* e da *Netflix*.

Técnica baseada na relação entre itens:

Nesta parte do trabalho será feito um estudo e análise semelhante ao realizado para as Figuras 15 e 16, utilizando uma medida de similaridade diferente. A utilização de um cálculo de similaridade diferente do utilizado na relação entre utilizadores comprova que o comportamento do MAE e da RMSE face à variação do valor de *k* é idêntico ao já verificado nas Figuras 15 e 16.

A medida de similaridade utilizada nas Figuras 19 e 20 foi o co-seno ou *cosine similarity*.

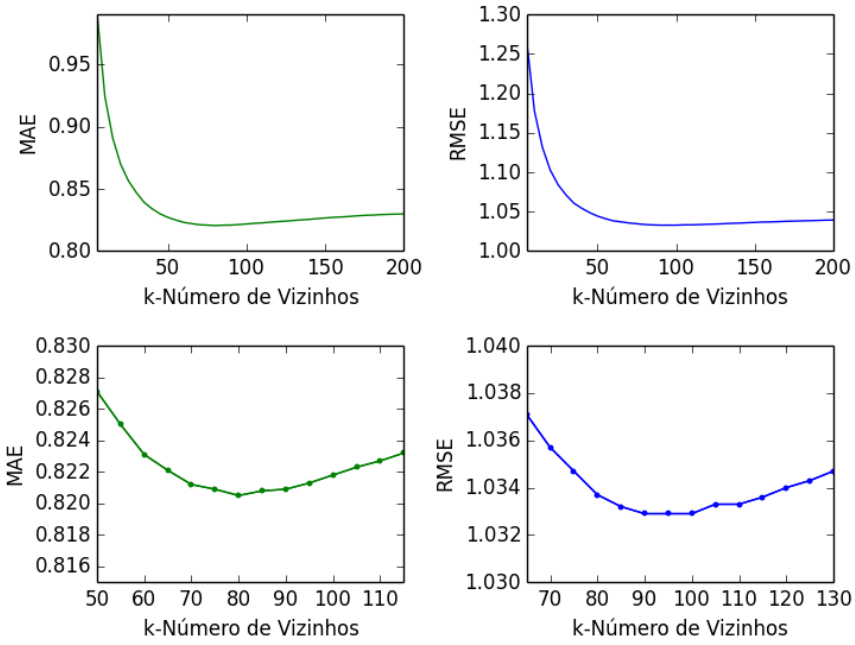

Figura 19 - Sensibilidade das predições ao tamanho da vizinhança, no conjunto de dados do *MovieLens*. O primeiro gráfico representa os valores do MAE e da RMSE até *k* =200. Os gráficos da linha de baixo são uma ampliação do primeiro.

Como se pode observar pelos gráficos inferiores da Figura 19, o melhor valor do MAE é atingido para *k*=80, porém não se verifica igualmente o menor valor da RMSE para esse mesmo valor de *k*. Este fenómeno vai ao encontro do que se referiu na Secção 4.2, ou seja, a RMSE pode penalizar mais a diferença entre os *ratings* preditos e os reais para um dado valor de *k* mas penalizar menos para outro,

Da Figura 19 tem-se que o menor valor da RMSE é obtido para *k*=100.

Encontrados os valores de *k* para o qual a qualidade das predições é maior tem-se pela Figura 20 a representação das métricas MAE e RMSE para as três medidas de similaridade estudadas na Secção 3.1.1.

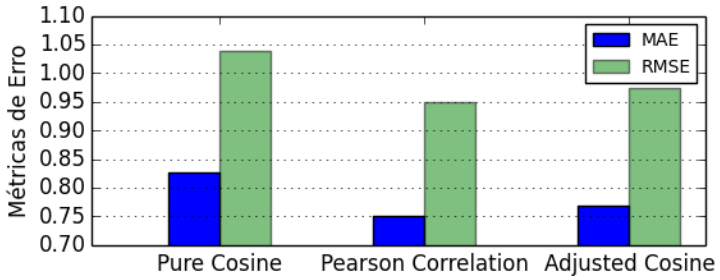

Figura 20 – Impacto das medidas de similaridade nos sistemas colaborativos baseados na relação entre itens para o conjunto de teste do *MovieLens*.

Discussão dos resultados:

Comparando os resultados obtidos na Figura 20 com os do artigo [20] (representados na Figura 21) tem-se que os valores do MAE para a medida de similaridade do co-seno são muito próximos.

Para o cálculo da correlação de Pearson a diferença entre o MAE obtido neste trabalho e no artigo [20] é significativa, pelo que é possível concluir que as predições calculadas neste trabalho são superiores às obtidas no artigo [20].

Não existe uma explicação válida para este fenómeno, no entanto pode especular-se que a diferença do MAE se deve à forma como os parâmetros foram ajustados, ou seja, enquanto neste trabalho se utilizou um conjunto de desenvolvimento para se analisar o melhor valor de *k*, no artigo [20] utilizou-se o ajuste do parâmetro *k* utilizando validação cruzada, e tal como se observou pela Figura 19, a escolha do valor de *k* pode produzir um valor do MAE final diferente.

Para os valores da similaridade do co-seno ajustada verifica-se que as métricas de erro implementadas são superiores às apresentadas na Figura 21.

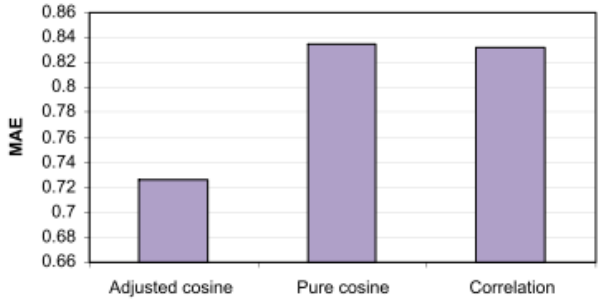

Figura 21 - Impacto das medidas de similaridade nos sistemas colaborativos baseados na relação entre itens para o *MovieLens* obtidas em [20].

Uma medida de similaridade que não foi ainda analisada é a similaridade de co-seno com os *ratings* centrados em 3. A Figura 22 apresenta a variação da qualidade das predições com a variação de *k*.

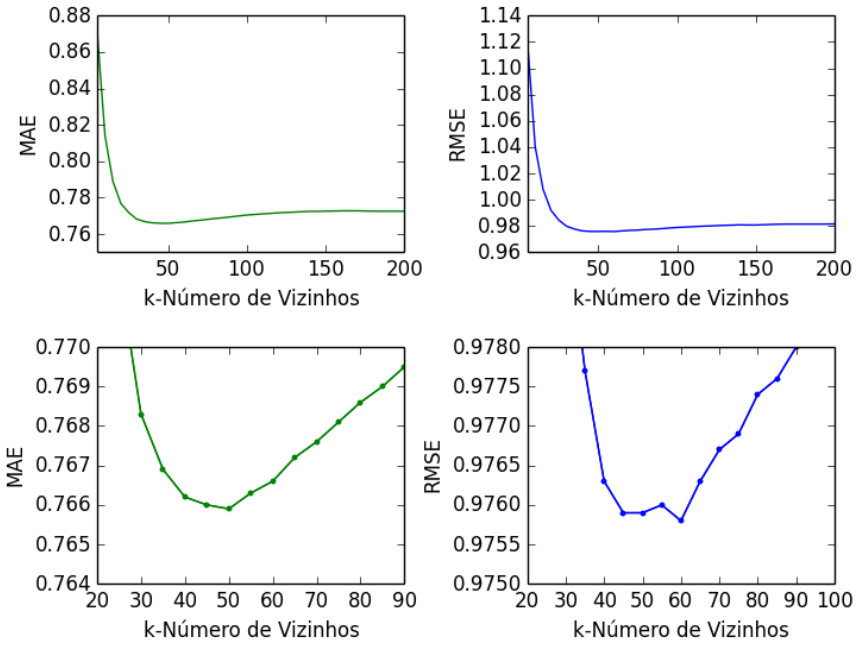

Figura 22 - Sensibilidade das predições com o tamanho da vizinhança, no conjunto de dados do *MovieLens*. O primeiro gráfico representa os valores do MAE e da RMSE até *k* =200. Os gráficos da linha de baixo são uma ampliação do primeiro.

Comparando os resultados obtidos na Figura 22 com os das outras similaridades apresentadas na Figura 20 tem-se que a similaridade de co-seno centrada em 3 atinge valores de MAE tão próximos como os apresentados para a similaridade de co-seno ajustada. Um outro facto interessante é a superioridade na qualidade das predições quando comparada com a similaridade de co-seno.

Analisando os dados da Figura 22, o melhor valor do MAE é atingido para *k*=50, enquanto a RMSE é menor para *k*=60.

De seguida apresentam-se na Tabela 22 alguns dados adicionais para as medidas de similaridade apresentadas nas Figuras 20 e 22

		Cosine Similarity	Pearson Correlation	Adjusted Cosine	Centered Cosine
MovieLens	Número de experiências	200			
	Tempo médio de processamento em segundos	6661	5817	5512	8364
	Melhor valor de *k* obtido no conjunto *dev*	80	60	95	50

Figura 23 - Dados adicionais obtidos para as técnicas colaborativas baseadas na relação entre itens para o conjunto de dados do *MovieLens*.

Uma análise de resultados que não é feita no artigo [20] e que poderá ser feita neste trabalho é a comparação das técnicas colaborativas baseadas na relação entre utilizadores e na relação entre itens.

Comparando as métricas MAE das duas técnicas apresentadas nas Figuras 18 e 20 tem-se que a medida de co-seno é muito próxima para ambos os casos, ainda que seja ligeiramente inferior para a relação entre utilizadores. Para a medida de correlação de Pearson, o menor valor de MAE corresponde aos sistemas baseados na relação entre utilizadores, sendo que este valor se aproxima bastante do apresentado no artigo [20].

Ainda relativamente às técnicas baseadas na relação entre itens, para o conjunto de dados da *Netflix* a qualidade das predições com a variação de *k* encontra-se representada na Figura 24.

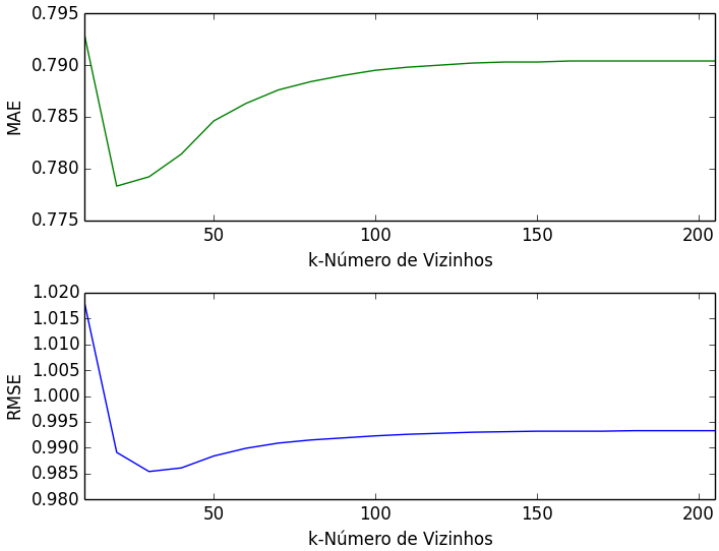

Figura 24 - Sensibilidade das predições ao tamanho da vizinhança, no conjunto de dados da *Netflix*.

Observando a Figura 24 o melhor valor de *k* para o conjunto de dados da *Netflix* ocorre quando *k*=20 para o caso do MAE, e *k*=30 para a RMSE. Repare-se que a qualidade das predições a partir dos valores de *k* apresentados vai diminuindo, acabando por estabilizar a partir de *k*=140, aproximadamente, para ambos os casos.

De seguida apesentam-se na Tabela 7 os tempos médios de processamento para cada medida de similaridade e o melhor valor de *k* obtido no conjunto *dev*.

		Cosine Similarity	Pearson Correlation	Adjusted Cosine
Netflix	Número de experiências	400		
	Tempo médio de processamento em segundos	42038	44712	54788
	Melhor valor de *k* obtido no conjunto *dev*	20	70	50

Tabela 7 - Dados adicionais obtidos para as diferentes medidas de similaridade baseadas na relação entre itens para o conjunto de dados da *Netflix*.

É de notar que não são feitas comparações entre os resultados obtidos para a *Netflix* e o *MovieLens* pelo facto de se tratarem de conjuntos de dados distintos.

4.3.4. Implementação de Algoritmos Colaborativos baseados em Modelo

Como foi referido na Secção 3.3.3 esta parte do trabalho baseia-se na implementação de um algoritmo baseado em factorização de matrizes apresenntada por Simon Funk durante a competição da *Netflix* [24].

Foram realizadas algumas experiências iniciais com o objectivo de encontrar os parâmetros que produzem uma maior variação no comportamento global do sistema, isto é, na proximidade entre o *rating* real e o calculado pelo algoritmo (*rating* estimado).

Na Secção 3.3.3 foram referidos os parâmetros mais importantes para esta técnica, tais como: o número de épocas, o número de características (ou features), a taxa de aprendizagem e o termo de regularização, que podem ser observados nas Figuras 25, 26, 27 e 28, respectivamente.

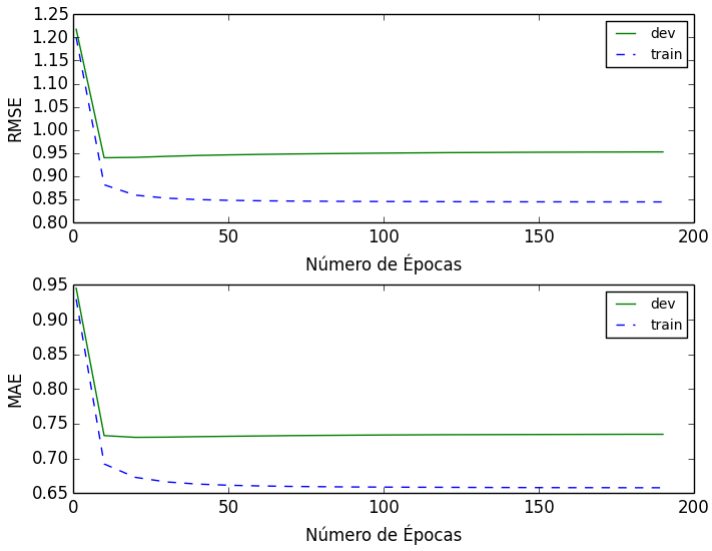

Figura 25 – Variação do MAE e da RMSE com o número de épocas para o *MovieLens*.

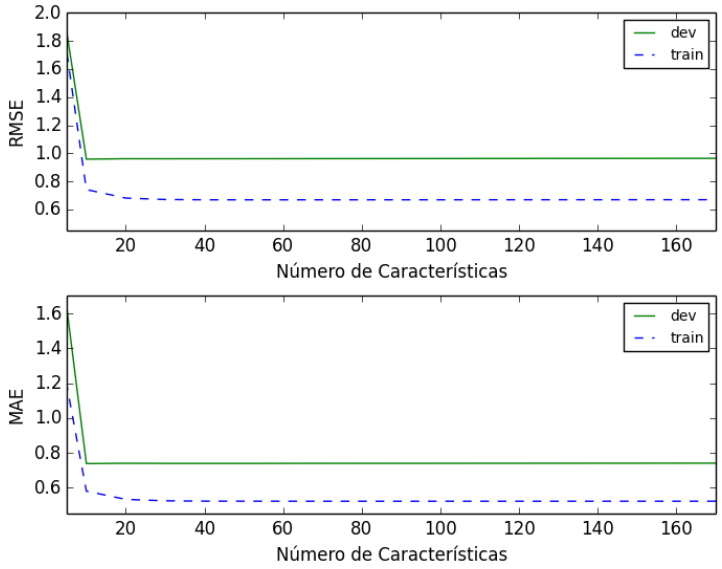

Figura 26 - Variação do MAE e da RMSE com o número de características para o *MovieLens*.

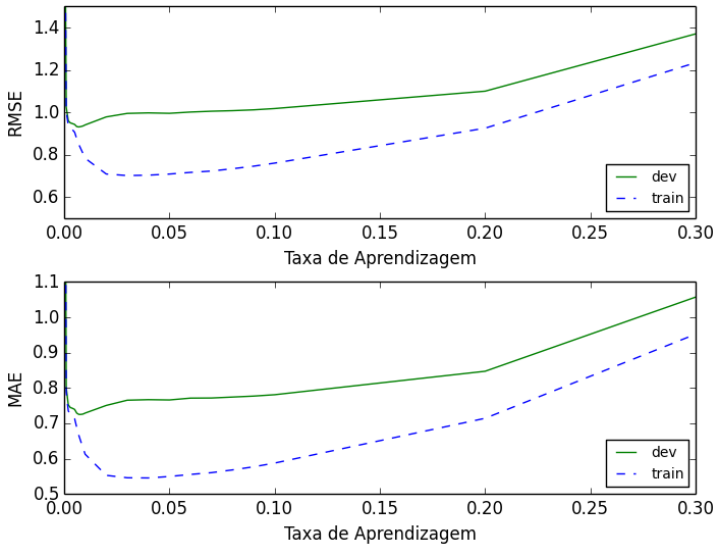

Figura 27 - Variação do MAE e da RMSE com a taxa de aprendizagem para o *MovieLens*.

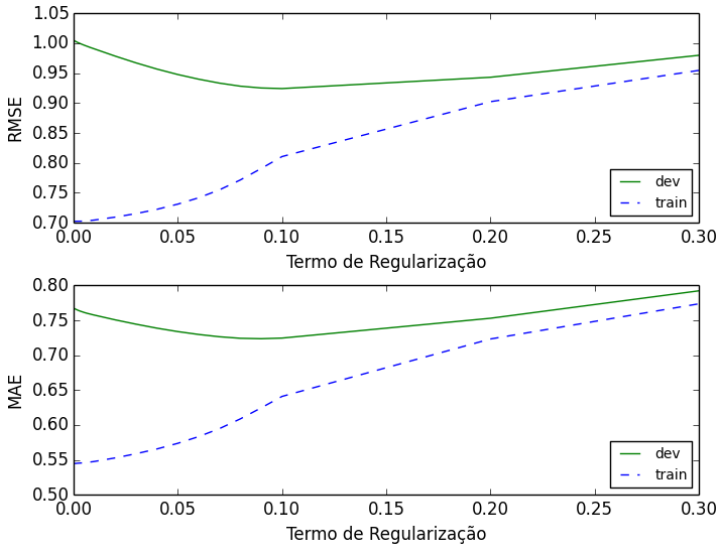

Figura 28 - Variação do MAE e da RMSE com a taxa de regularização para o *MovieLens*.

Discussão de Resultados:

Analisando a Figura 25, quando o número de épocas é 100, o algoritmo obtém o melhor valor para a RMSE, embora isso não seja visível no gráfico.

Considerando o melhor número de épocas acima apresentado, calculou-se qual seria o número de características ideais a considerar, lembrando que cada para cada utilizador e item existe um vector composto por *f* características. Da Figura 26 tem-se que o menor valor da RMSE é atingido quando *f*=10.

Analisando o gráfico da Figura 27 verifica-se uma diminuição no erro das predições quando se aumenta a taxa de aprendizagem até 0.007. A partir de Lrate=0.007[20] o valor do erro no conjunto de desenvolvimento ou *dev* regista um aumento, no entanto, no conjunto de treino o erro continua a descer até 0.02, ou seja, verifica-se o fenómeno de *overfitting*, porém a partir de 0.02 à medida que a RMSE do conjunto dev aumenta, a RMSE de treino acompanha esse aumento gradualmente começando a seguir um comportamento semelhante. Este fenómeno verifica-se sobretudo para valores menores da taxa de aprendizagem.

Os resultados apresentados na Figura 28 confirmam que utilizando valores muito baixos para a regularização se traduz num aumento do *overfitting* durante a fase de treino dos dados, porém aumentando a regularização tem-se que o valor da RMSE no conjunto *dev* diminui e que a RMSE de treino se aproxima mais do obtido na fase de desenvolvimento, tal como seria esperado que acontecesse.

Portanto, o aumento da regularização traduz um efeito positivo no algoritmo, embora este aumento deva ser controlado, pois se for demasiado elevado a RMSE aumenta, diminuindo o desempenho do sistema, ou seja, obtém-se piores predições para os *ratings* calculados. Por último,

[20] Abreviatura de *Learning rate* ou taxa de aprendizagem

conclui-se que a utilização da taxa de regularização previne o *overfitting* e neste caso o menor valor da RMSE é atingido quando a taxa de regularização é 0.1.

Ainda que não tenham sido reportados os tempos de execução do algoritmo, quanto menor for o número de épocas mais rápido será o tempo de execução, o mesmo se aplica para o caso do número de *features* escolhidas.

Conhecidos os melhores valores para o número de épocas, *features*, taxa de aprendizagem e regularização, obtidos no conjunto de desenvolvimento, tem-se que os valores finais para a RMSE e para o MAE obtidos no conjunto de teste são:

MAE	0.7208
RMSE	0.9209
Tempo de processamento [segundos]	6887

Tabela 8 - Métricas de erro obtidas para o conjunto de teste utilizando o melhor valor obtido para cada parâmetro do sistema.

Como se pode observar, as métricas de erro obtidas no conjunto de teste são inferiores, e portanto melhores, às obtidas pelos algoritmos baseados em memória. Comparando estes valores aos obtidos no artigo [20], para o melhor caso, ainda que muito próximos os valores apresentados neste trabalho atingem predições com maior qualidade.

Para o conjunto de dados da *Netflix* começou por se estudar a influência da regularização face às métricas de erro, Figura 29.

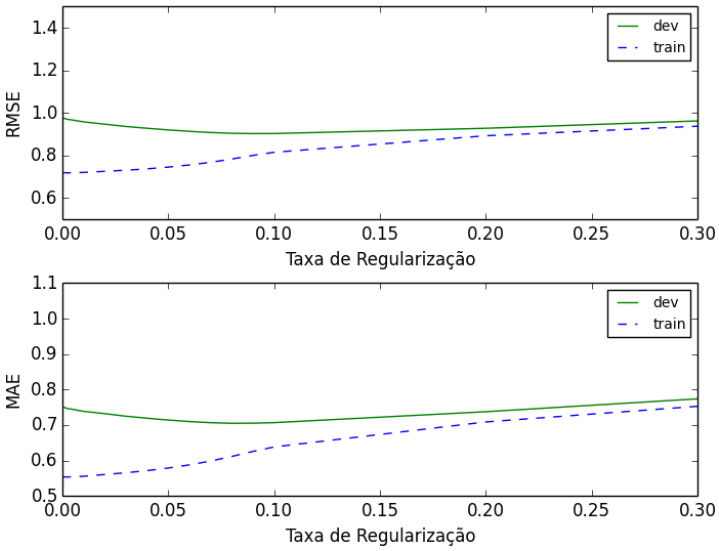

Figura 29 - Variação do MAE e da RMSE com a taxa de regularização para a *Netflix*.

Tal como se verificou para o *MovieLens*, à medida que a regularização aumenta maior é a proximidade entre as métricas de erro de teste e *dev*, sendo que a partir de 0.2 as duas métricas começam a ter valores muito próximos.

Fixado o melhor valor para a taxa de regularização de 0.09, calculou-se o comportamento das métricas de erro face à variação da taxa de aprendizagem, Figura 30.

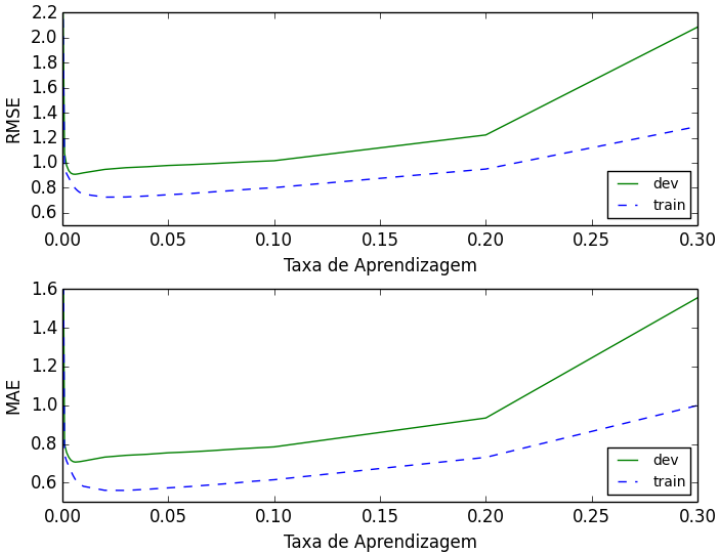

Figura 30 - Variação do MAE e da RMSE com a taxa de aprendizagem para a *Netflix*.

Observando a Figura 30 tem-se que os valores da RMSE e do MAE são bons até 0.009, sendo que a partir desse valor a qualidade das predições diminui.

É possível verificar ainda pela Figura 30 que a partir de 0.2 a diferença entre os valores obtidos no conjunto de treino e *dev* começa a ser significativa. Isto deve-se ao facto de a taxa de aprendizagem corresponder ao passo do algoritmo, ou seja, para valores muito pequenos da taxa de aprendizagem o algoritmo poderá demorar muito tempo a convergir para uma solução que correspondente a um mínimo local, porém se o passo for muito grande o algoritmo pode nunca conseguir convergir para uma solução que deveria ser a mínima.

A Figura 31 representa a variação das métricas de erro com a variação do número de épocas utilizadas pelo algoritmo.

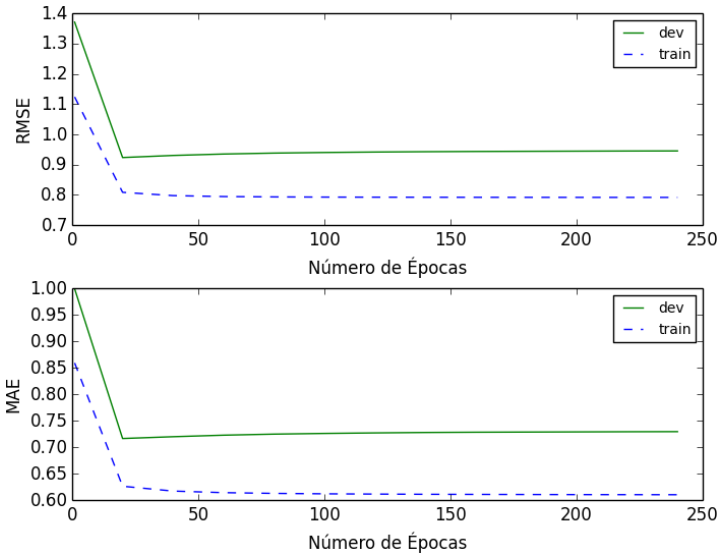

Figura 31 - Variação do MAE e da RMSE com o número de épocas para a *Netflix*.

Observando a Figura 31 retira-se que o valor mínimo da RMSE ocorre quando o número de épocas é igual a 20. Este número afasta-se do número apontado em [24] que seria de 120 épocas, no entanto o autor considerou a avaliação de todo o conjunto de dados da *Netflix* enquanto neste trabalho é feito uma análise de um subconjunto desses dados.

Resta analisar a qualidade das predições calculadas pelo algoritmo face ao número de características utilizadas, apresentada na Figura 32.

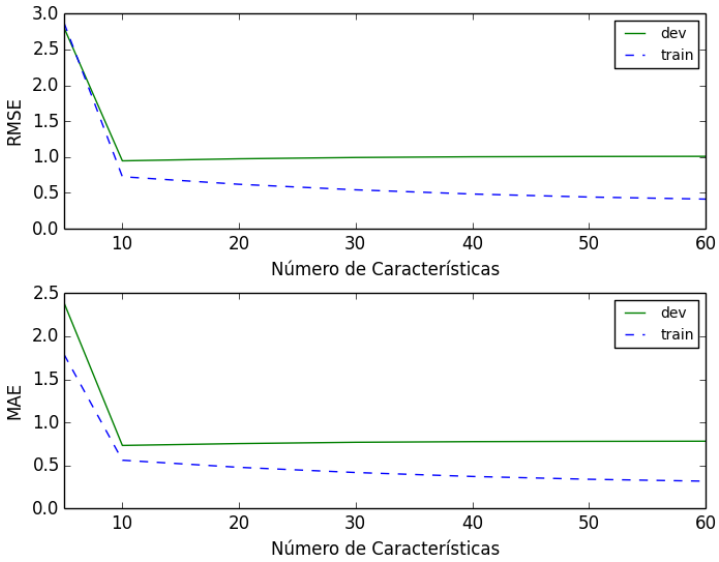

Figura 32 - Variação do MAE e da RMSE com o número de características para o *Netflix*.

Observando a Figura 32, quando o número de características é igual a 10, o sistema atinge valores para a RMSE de 0.947 e para o MAE de 0.732, na fase de desenvolvimento (*dev*).

Conhecendo todos os melhores valores para os parâmetros do sistema, e utilizando o algoritmo no conjunto de teste obtêm-se os seguintes valores:

MAE	0.9260
RMSE	0.7298
Tempo de processamento [segundos]	965

Tabela 9 - Métricas de erro obtidas para o conjunto de teste utilizando o melhor valor obtido para cada parâmetro do sistema.

O valor obtido para a RMSE no conjunto de teste é inferior ao do *Cinematch*, ou seja, este sistema é capaz de calcular predições com maior qualidade que as obtidas pelo algoritmo utilizado pelo *Netflix* durante o período da competição, como se apresentou na Secção 2.1.1.

Repare-se também na diferença de tempo, que é considerável quando comparada com a implementação do *k*-NN, porém o tempo de treino do algoritmo é bastante elevado visto que é necessário verificar quais os parâmetros que melhor desempenho oferecem ao sistema.

5. Conclusões e Trabalho Futuro

5.1. Conclusões

Foi apresentado neste trabalho o papel fundamental dos sistemas de recomendação como mecanismos de predição e recomendação de itens a utilizadores, partindo do seu histórico de compras ou de classificações anteriormente atribuídas. Reforçou-se também a importância da criação e desenvolvimento destes sistemas que se tornaram imprescindíveis no mundo do comércio *online*.

De seguida fez-se uma revisão temporal de alguns serviços que utilizam ou utilizaram sistemas de recomendação como principais técnicas de recomendação de produtos que os vendedores queriam ver escoados ao mesmo tempo que pretendiam ver os seus potenciais compradores interessados no produto sugerido ou noutro semelhante (também ele sugerido pelos sistemas de recomendação).

Dos três tipos de sistemas de recomendação apresentados neste trabalho, destacaram-se essencialmente os sistemas de recomendação por filtragem colaborativa, e para esta classe de sistemas foram implementados dois algoritmos, um deles relativo aos sistemas colaborativos baseados em memória e o restante ao conjunto de algoritmos baseados em modelo.

Dos algoritmos baseados em memória, os resultados obtidos foram comparados com os apresentados no artigo utilizado como guia nesta fase do trabalho [20]. Para o último algoritmo implementado a referência utilizada foi uma página *web* realizada por um dos concorrentes da competição do *Netflix*, vista como um dos acontecimentos que levou à criação e desenvolvimento de novos algoritmos na área dos sistemas de recomendação [24].

Os resultados obtidos foram os esperados para ambos os algoritmos, no entanto existem algumas considerações a ter em conta. Para o caso do algoritmo baseado na relação entre utilizadores as métricas de erro, MAE e RMSE, são inferiores para o caso em que a similaridade utilizada é a similaridade do co-seno, além disso, se for utilizada uma similaridade como a correlação de Pearson os resultados melhoram bastante aproximando-se dos valores obtidos pelos autores do artigo [20].

Comparando os resultados obtidos para os algoritmos colaborativos baseados na relação entre utilizadores e os baseados na relação entre itens tem-se que os valores das métricas de erro não são tão díspares como seria esperado, uma vez que os autores [20] referem que os sistemas que utilizam algoritmos baseados na relação entre itens calculam predições de itens com maior qualidade que os baseados na relação entre utilizadores.

Comparando os resultados do algoritmo baseado em factorização de matrizes com os obtidos pelos algoritmos anteriormente referidos é perceptível o melhoramento da qualidade de predição dos *ratings.*

Por último, foi possível ainda verificar a forte dependência da qualidade das predições face ao ajustamento dos parâmetros do algoritmo.

5.2. Trabalho Futuro

Numa fase futura deste trabalho seria interessante estudar outros algoritmos baseados em modelo como é o caso dos modelos probabilísticos e posteriormente comparar os seus resultados com os obtidos neste trabalho.

Seria interessante também aperfeiçoar o algoritmo baseado em factorização de matrizes aqui implementado juntamente com técnicas já conhecidas e referidas neste trabalho, como é o caso dos sistemas que utilizam a relação entre itens no cálculo das similaridades entre os utilizadores. Já existem alguns trabalhos que combinam uma análise baseada em factorização de matrizes com uma análise baseada na relação entre itens mais próximos que obtiveram bons resultados [34].

Tentar melhorar o tempo de treino do algoritmo baseado em factorização de matrizes poderia ser um desafio visto que o tempo despendido pelo algoritmo é elevado caso se pretendesse fazer recomendações em tempo real.

Referências

[1] Schafer, J. B.; Konstan, J. and Riedl, J. Recommender systems in e-commerce. New York.

[2] http://www.*Netflix*prize.com/leaderboard

[3] http://www.*Netflix*prize.com/

[4] www.grouplens.org

[5] http://www.gonow.com.br/blog/2011/11/16/python-a-arma-secreta-do-google/

[6] Daniel G. Shafer, Python Streamlines Space Shuttle Mission Design, 2003.

[7] Darryl K. Taft, Python Slithers into Systems eWEEK, 2005.

[8] http://www.businessinsider.com/amazon-prime-10-million-members-morningstar-2013-3

[9] Sarwar, B.M., Konstan, J.A., Borchers, A., Herlocker, J., Miller, B., Riedl, J.: Using filtering agents to improve prediction quality in the grouplens research collaborative filtering system. New York, 1998.

[10] Desrosiers, C., & Karypis, G. - A comprehensive survey of neighborhood-based recommendation methods. 2011.

[11] Zuva, T., Ojo, S. O., Ngwira, S. M., & Zuva, K. A Survey of Recommender Systems Techniques , 2012.

[12] www.python.org

[13] G. Adomavicius and A. Tuzhilin, “Towards the next generation of recommender systems: a survey of the state-of-the-art and possible extensions”, 2005.

[14] Gemmis, M. De, Iaquinta, L., Lops, P., Musto, C., Narducci, F., & Semeraro, G. Preference Learning in Recommender Systems, 2009.

[15] M. J. Pazzani and D. Billsus, "Content-based Recommendation Systems", 2007.

[16] M. Balabanovic and Y. Shoham, “Fab: Content-Based, Collaborative Recommendation”, 1997.

[17] T. Bogers and A. Bosch, "Collaborative and Content-based Filtering for item Recommendation on Social Bookmarking Websites", 2009.

[18] U. Jeffrey, “Recommendation Systems”. Fonte: http://infolab.stanford.edu/~ullman/

[19] Burke, R. Hybrid Recommender Systems : Survey and Experiments, 2002.

[20] Sarwar, B., Karypis, G., Konstan, J., & Riedl, J. Item-Based Collaborative Filtering Recommendation Algorithms, 2001.

[21] Papagelis, M., & Plexousakis, D. Qualitative analysis of user-based and item-based prediction algorithms for recommendation agents, 2005.

[22] Shani, G., & Gunawardana, A., Evaluating Recommendation Systems, 2011.

[23] D.W. Oard and J.Kim, “Implicit Feedback for a Recommender Systems”, 1998.

[24] http://sifter.org/~simon/journal/20061211.html

[25] Konstan, J. A., Miller, B. N., Maltz, D., Herlocker, J. L., Gordon, L. R., & Riedl, J. Recom. Syst., 1997.

[26] Melville, P., Sindhwani, V., & Heights, Y., Recommender Systems, 2010.

[27] G. Linden, B. Smith, and J. York. Amazon.com Recommendations: Item-to-Item Collaborative Filtering, 2003.

[28] M. Montaner, B. Lopez, and J. L. De La Rosa. A Taxonomy of Recommender Agents on the Internet, 2003.

[29] Breese, J. S., D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering, 1998.

[30] http://store.virginmedia.com

[31] Gong, S. A Collaborative Filtering Recommendation Algorithm Based on User Clustering and Item Clustering, 2010.

[32] Ekstrand, M. D. Collaborative Filtering Recommender Systems, 2010.

[33] Sharifi, Z., Rezghi, M., & Nasiri, M. New algorithm for recommender systems based on singular value decomposition method, 2013.

[34] Vozalis, M. G., Margaritis, K. Applying SVD on Generalized Item-based Filtering, 2006.

[35] Sarwar, B. M., Karypis, G., Konstan, J. A., & Riedl, J. T. Application of Dimensionality Reduction in Recommender System -- A Case Study, 1998.

[36] Sarwar, B., Karypis, G., Konstan, J., & Riedl, J. Incremental Singular Value Decomposition Algorithms for Highly Scalable Recommender Systems, 2002.

[37] Chris, V., Robert, B., Matrix Factorization Techniques for recommender systems. Yahoo Research, 2009.

[38] http://www.newsmax.com/SciTech/*Netflix*-reporting-web-grown/2014/01/22/id/548377/

[39] Bennett, J., Lanning, S. The *Netflix* Prize. In KDD Cup and Workshop in conjunction with KDD, 2007

[40] C. Anderson, *The Long Tail. Why the Future of Business Is Selling Less of More*.NewYork, NY: Hyperion, 2006.

[41] Anderson, Chris. The Long Tail: Why the Future of Business is Selling Less of More, 2006.

[42] Quinion, Michael., "Turns of Phrase: Long Tail". World Wide Words, 2011.

[43] D. Goldberg, D. Nichols, B. M. Oki, and D. Terry, “Using Collaborative Filtering to Weave an Information Tapestry”, 1992.

[44] Terveen, L., Hill, W., Amento, B., Mcdonald, D., & Creter, J. PHOAKS : a system for sharing recommendations, 1997.

[45] http://grouplens.org/

[46] https://usenet-news.net/index1.php?url=home

[47] http://jolomo.net/ringo.html

[48] Sarwar, B., Karypis, G., Konstan, J., & Riedl, J. Analysis of recommendation algorithms for e-commerce, 2000.

[49] Brand, M. Fast Low-Rank Modifications of the Think Singular Value Decomposition, 2006.

[50] G. Takács, I. Pilászy, B. Németh e D. Tikk, “Scalable Collaborative Filtering Approaches for Large Recommender Systems”, 2009.

[51] Cantador, I., Bellogín, A., & Castells, P. A Multilayer Ontology-based Hybrid Recommendation Model, 2008.

[52] Uluyagmur, M., Cataltepe, Z., & Tayfur, E. Content-Based Movie Recommendation Using Different Feature Sets, 2012.

[53] Lyu, M. R. Recommender Systems with Social Regularization Categories and Subject Descriptors, 2011.

[54] Terveen, L., & Hill, W. Beyond Recommender Systems : Helping People Help Each Other Recommendation : Examples and Concepts, 2001.