risti
Revista Ibérica de Sistemas e Tecnologias de Informação
Revista Ibérica de Sistemas y Tecnologías de Información

Recebido / Recibido: 08/10/2011
Aceitação / Aceptación: 09/12/2011

Algoritmo de Recomendação Baseado em Passeios Aleatórios num Grafo Bipartido

Ricardo Gama [1], Nuno André, César Pereira, Luís Almeida, Pedro Pinto

[1]rgama@estgl.ipv.pt

Centro de Estudos em Educação, Tecnologias e Saúde, ESTGL, Instituto Politécnico de Viseu

Av. Coronel José Maria V. de Andrade, Campus Politécnico, 3504-510 VISEU, Portugal

Resumo: Neste trabalho apresentamos um novo algoritmo de recomendação para aplicações de *E-commerce*, baseado em passeios aleatórios num grafo bipartido. O desempenho do mesmo é testado com dados provenientes de transações comerciais reais e os resultados comparados com os de cinco outros algoritmos. Por fim, apresentamos possíveis generalizações do algoritmo, tornando-o muito versátil em aplicações práticas.

Palavras-chave: Algoritmos de Recomendação, Filtragem Colaborativa, Passeios Aleatórios.

Abstract: In this paper we present a new recommendation algorithm for E-commerce applications, based in random walks in a bipartite graph. Its performance is tested using real commercial transaction data and the results are compared with five other algorithms. Finally, we present possible generalizations for our algorithm making it very versatile for practical applications.

Keywords: Recommendation algorithms, Collaborative Filtering, Random Walks.

1. Introdução

Os algoritmos de recomendação representam, atualmente, um esforço acrescido, por parte das empresas de venda online, em fornecer aos seus clientes uma recomendação de produtos que vão ao encontro das características do consumidor, potenciando não só a venda dos seus produtos, mas também fornecendo um serviço de valor acrescentado ao consumidor que apenas deseja adquirir um produto.

Apesar do enorme sucesso de alguns algoritmos de recomendação (Zanker et al, 2007) grande parte apresenta alguns problemas, nomeadamente, na iniciação e esparsidade dos dados (Huang et al., 2004) contribuindo assim, para uma redução da qualidade das recomendações. Numa fase inicial, a informação que relaciona os consumidores com os produtos é escassa, o que dificulta a aprendizagem dos algoritmos e o bom

desempenho dos mesmos. Deste modo, é necessário criar métodos para reduzir ao máximo o tempo de aprendizagem dos algoritmos e dotá-los de mecanismos de previsão, baseados na pouca informação inicial disponível. Motivados pelo grande êxito de vários algoritmos aplicados a motores de busca na Internet, como os algoritmos *PageRank*, HITS e SALSA (Langville et al., 2006) surgiram, nos últimos anos, vários trabalhos que sugerem a utilização de modelos semelhantes em sistemas de recomendação (Huang et al., 2004,2007; Meyer et al., 2008; Pucci et al, 2007, Li et al., 2009).

2. Descrição do Algoritmo

Neste artigo, apresentamos um novo algoritmo que efetua recomendações baseadas em passeios aleatórios no grafo bipartido de consumidores/produtos.

À semelhança dos trabalhos apresentados por (Huang et al., 2004, 2007) no nosso modelo, interpretamos os dados das transações como um grafo bipartido, onde num dos grupos temos os consumidores e no outro os produtos possíveis de adquirir (2.1). Apesar de ter uma origem idêntica, o algoritmo apresentado neste artigo destaca-se dos trabalhos de Huang et al., pelo facto de assentar num modelo de passeios aleatórios diferente. Consequentemente, para além de ter um desempenho diferente, o algoritmo proposto utiliza apenas matrizes estocásticas (matrizes normalizadas por linhas) no seu funcionamento, evitando assim a necessidade de efetuar normalizações intermédias no decorrer do mesmo, que aumentam os tempos de execução e criam instabilidades numéricas desnecessárias.

De modo a preparar os dados para o inicio do algoritmo, a informação contida no grafo é armazenada numa matriz consumidores/produtos, A, cujas entradas a_{ij} indicam o número de vezes que o consumidor i comprou o produto j. Considerando que temos nc consumidores e np produtos, a matriz A vai ter dimensões $nc \times np$.

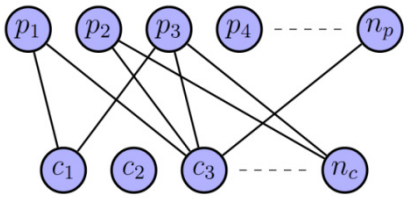


Figura 2.1: Grafo Consumidores/Produtos

Partindo deste modelo, pudemos criar as matrizes Nc e Np normalizando por linhas e por colunas a matriz A. Para contornar os problemas que surgem com clientes que não efetuaram compras, ou com produtos que nunca foram comprados, criámos dois vetores ac e ap de dimensões $nc \times 1$ e $np \times 1$ respetivamente, cujas entradas vão ser 1, caso o cliente (produto) tenha efetuado compras (sido comprado) e zero no caso contrário. Construímos as matrizes seguintes:

$$Mc = Nc + (1/np)\, ac \times ep^T, \qquad (1)$$

$$Mp = Np + (1/nc) ep \times ac^T, \qquad (2)$$

onde os vetores ec e ep de dimensões $nc \times 1$ e $np \times 1$ respetivamente, são vetores com todas as entradas iguais a um. Assim, contornámos o problema da inicialização, garantindo que todos os clientes possam ter recomendações. Um outro problema que pode surgir é a existência de grupos de clientes sem ligações entre eles. Para contornar esta situação, definimos a matriz:

$$Mm = (1/nc)\, ec \times ep^T, \qquad (3)$$

e criamos uma matriz Mc generalizada:

$$M_c^g = \beta_0 Mc + (1 - \beta_0) Mm. \qquad (4)$$

O nosso algoritmo consiste em simular nc passeios aleatórios, com recomeço, pelo grafo de consumidores/produtos, tendo como matrizes de transição, as matrizes M_c^g e Mp construídas anteriormente. Inicialmente, cada cliente inicia o seu passeio no vértice do grafo que lhe corresponde, avançando para o conjunto dos produtos, tendo em conta as probabilidades da matriz de transição M_c^g. Após este passo, cada cliente pode regressar para o grupo dos consumidores utilizando a matriz de transição Mp, com uma probabilidade α, ou retomar o seu ponto de partida, com uma probabilidade 1 - α. Este procedimento é repetido até se obter convergência ou até atingirmos um determinado número de iterações.

O resultado final do procedimento será uma matriz Cp, de dimensão $nc \times np$, que contém os vetores estacionários dos passeios de todos os clientes. Ou seja, obtemos uma matriz cujas entradas c_{ij} contêm a representatividade que o produto j tem para o cliente i.

Este procedimento pode ser resumido no seguinte algoritmo:

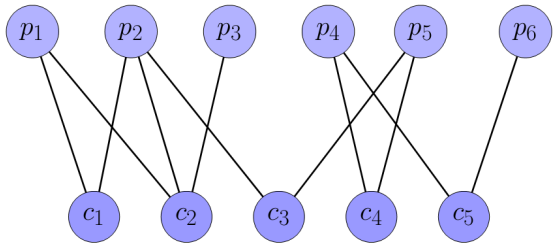
Entrada: M_c^g, M_p, α, erro máximo tolerado ϵ_m, número máximo de iterações k_m

C_0 =matriz identidade $n_c \times n_c$
$C^{(0)} = C_0$, $k = 1$
enquanto $k < k_m \wedge \epsilon > \epsilon_m$ **faça**
  $C_p^{(k)} = C^{(k-1)} \cdot M_c^g$
  $C^{(k)} = \alpha C_p^{(k)} \cdot M_p^T + (1 - \alpha) C^0$
  $k = k + 1$
  $\epsilon = |C_p^{(k)} - C_p^{(k-1)}|$
fim
retorna C_p

Figura 2.2: Algoritmo de Recomendação

A matriz Cp pode então ser utilizada para efetuar recomendações, escolhendo para cada cliente os produtos ainda não comprados com maior representatividade.

Vejamos um exemplo ilustrativo do algoritmo. Suponhamos que temos os seguintes registos de transações:



Produtos

Clientes	p_1	p_2	p_3	p_4	p_5	p_6
c_1	1	1	0	0	0	0
c_2	1	1	1	0	0	0
c_3	0	1	0	0	1	0
c_4	0	0	0	1	1	0
c_5	0	0	0	1	0	1

Figura 2.3: Exemplo de um grafo de transações e respectiva matriz

A partir desta informação podemos construir, tal como explicado anteriormente, as seguintes matrizes de transição:

$$M_c = \begin{pmatrix} 0.5000 & 0.5000 & 0 & 0 & 0 & 0 \\ 0.3333 & 0.3333 & 0.3333 & 0 & 0 & 0 \\ 0 & 0.5000 & 0 & 0 & 0.5000 & 0 \\ 0 & 0 & 0 & 0.5000 & 0.5000 & 0 \\ 0 & 0 & 0 & 0.5000 & 0 & 0.5000 \end{pmatrix} \qquad M_p = \begin{pmatrix} 0.5000 & 0.3333 & 0 & 0 & 0 & 0 \\ 0.5000 & 0.3333 & 1 & 0 & 0 & 0 \\ 0 & 0.3333 & 0 & 0 & 0.5000 & 0 \\ 0 & 0 & 0 & 0.5000 & 0.5000 & 0 \\ 0 & 0 & 0 & 0.5000 & 0 & 1 \end{pmatrix}$$

Inicializando $Cp^{(0)} = Mc$, aplicamos o algoritmo anterior até ser atingido o critério de paragem, obtendo a seguinte matriz:

$$C_p^{(\infty)} = \begin{pmatrix} 0.3057 & 0.3850 & 0.1112 & 0.0594 & 0.1168 & 0.0219 \\ 0.2843 & 0.3582 & 0.1732 & 0.0552 & 0.1087 & 0.0204 \\ 0.1532 & 0.3304 & 0.0738 & 0.1326 & 0.2609 & 0.0490 \\ 0.0723 & 0.1559 & 0.0348 & 0.3267 & 0.2896 & 0.1207 \\ 0.0424 & 0.0914 & 0.0204 & 0.3984 & 0.1698 & 0.2777 \end{pmatrix}.$$

Após a eliminação das compras previamente efetuadas, obtemos a matriz de recomendações:

$$\begin{pmatrix} - & - & 0.1112 & 0.0594 & 0.1168 & 0.0219 \\ - & - & - & 0.0552 & 0.1087 & 0.0204 \\ 0.1532 & - & 0.0738 & 0.1326 & - & 0.0490 \\ 0.0723 & 0.1559 & 0.0348 & - & - & 0.1207 \\ 0.0424 & 0.0914 & 0.0204 & - & 0.1698 & - \end{pmatrix}$$

A título de exemplo, podemos observar que se desejarmos efetuar duas recomendações ao cliente c_3, podemos ver que recomendaríamos os produtos $\{p_1; p_4\}$.

3. Avaliação Experimental e Resultados

Com o objetivo de avaliar o desempenho do algoritmo proposto neste trabalho, utilizámos dois conjuntos de dados cedidos em http://isl.ifit.uni- klu.ac.at, por (Zanker et al, 2007), *Buys Advisor Data* e *Buys All Data*, contendo informação sobre transações comerciais de uma loja online de cigarros cubanos.

Efetuámos a comparação dos resultados do nosso algoritmo com os resultados obtidos por cinco algoritmos de recomendação descritos na literatura:

- Recomendação aleatória, onde para cada cliente é escolhida, aleatoriamente, para recomendação uma lista de n produtos ainda não comprados pelo mesmo, (Aleat.);
- Recomendação dos produtos mais vendidos, onde para cada cliente é escolhida para recomendação, a lista dos *n* produtos mais comprados, que ainda não foram adquiridos pelo mesmo, (*Top n*);
- Algoritmos clássicos de filtragem colaborativa baseados em produtos e em clientes, usando o coeficiente de Pearson na criação das matrizes de semelhança [8, 2], (*CFUB* e *CFIB*);
- Algoritmo de recomendação proposto em (Huang et al., 2007), (L.A.ALg:).

A metodologia usada consistiu na utilização de dois métodos: dado n e todos menos n.

O método dado *n* consiste em utilizar n produtos selecionados aleatoriamente da lista de compras de um dado consumidor, que tenha pelo menos *n+1* compras, como dados de treino, considerando as restantes compras como conjunto de teste, *CT*.

No método todos menos *n*, por outro lado, usámos todos os produtos comprados pelos utilizadores que tenham pelo menos *n+2* compras, como dados de treino, exceto uma seleção aleatória de *n* deles, considerada como conjunto de teste.

Nos nossos testes, repetimos ambos os métodos 250 vezes para todos os utilizadores que verificavam as condições e medimos o desempenho dos vários algoritmos, calculando as medidas clássicas de precisão (P) e sensibilidade (S) (Huang et al., 2004; Zanker et al, 2007).

Após ter um conjunto de *nr* de artigos a recomendar para cada utilizador, *CR*, confrontámos o mesmo com o conjunto de teste e determinámos o conjunto de acertos do algoritmo de recomendação para esse cliente: $CA = CR \cap CT$, calculando posteriormente a precisão e a sensibilidade:

$$P = \frac{\#CA}{\#CR}\ ; \quad S = \frac{\#CA}{\#CT}\ .$$

Como o algoritmo apresentado neste trabalho depende do parâmetro x, usámos o mesmo, para valores de α de 0.01 a 0.99 por passos de 0.01, escolhendo no final o que apresentou valores mais elevados das medidas.

Os resultados obtidos estão sintetizados nas tabelas seguintes:

Buys Advisor Data

Dados n							
$n = 2$ & $n_r = 5$	α	proposto	Aleat.	topN	CFUB	CFIB	L.A.ALg.
S	0.82	16.08 ± 6	3.55 ± 4	15.50 ± 6	9.25 ± 3	9.02 ± 5	11.36 ± 4
P	0.93	7.16 ± 2	2.44 ± 2	8.24 ± 2	5.44 ± 2	4.37 ± 2	5.21 ± 2
$n = 3$ & $n_r = 5$							
S	0.95	14.71 ± 7	3.61 ± 5	13.62 ± 5	13.31 ± 5	6.64 ± 5	13.81 ± 7
P	0.95	8.59 ± 3	2.44 ± 2	9.79 ± 3	7.44 ± 3	5.24 ± 3	6.85 ± 3
Todos menos n							
$n = 1$ & $n_r = 5$	α	proposto	Aleat.	topN	CFUB	CFIB	L.A.ALg.
S	0.62	5.95 ± 3	1.26 ± 2	6.48 ± 3	3.70 ± 2	3.50 ± 2	4.82 ± 2
P	0.90	3.22 ± 1	0.66 ± 1	3.33 ± 1	2.22 ± 1	1.80 ± 1	2.23 ± 1
$n = 2$ & $n_r = 5$							
S	0.94	10.55 ± 4	2.27 ± 3	8.18 ± 4	7.25 ± 3	3.69 ± 3	8.70 ± 3
P	0.94	6.32 ± 3	1.47 ± 2	5.26 ± 2	4.15 ± 2	2.76 ± 2	4.97 ± 2

Tabela 1: Desempenho dos algoritmos em 250 repetições para o conjunto de dados *Buys Advisor Data*, contendo informação sobre as compras efetuadas por 37 clientes tendo à disposição 143 produtos, (média±desvio padrão).

Buys All Data

Dados n							
n=2 & nr=5	α	proposto	Aleat.	topN	CFUB	CFIB	L.A.ALg.
S	0.82	17.50 ± 2	3.64 ± 1	13.78 ± 2	14.68 ± 2	5.71 ± 1	16.5 ± 2
P	0.82	7.47 ± 1	1.64 ± 1	5.54 ± 1	6.32 ± 1	2.62 ± 1	6.92 ± 1
$n = 3$ & $n_r = 5$							
S	0.63	19.38 ± 3	3.70 ± 2	12.04 ± 3	14.72 ± 3	6.26 ± 2	16.52 ± 3
P	0.64	7.96 ± 1	1.69 ± 1	5.25 ± 1	6.27 ± 1	2.24 ± 1	6.82 ± 1
Todos menos n							
$n = 1$ & $n_r = 5$	α	proposto	Aleat.	topN	CFUB	CFIB	L.A.ALg.
S	0.81	6.89 ± 1	1.30 ± 1	5.41 ± 1	5.77 ± 1	2.59 ± 1	6.95 ± 1
P	0.47	3.80 ± 1	0.74 ± 1	2.81 ± 1	3.22 ± 1	1.42 ± 1	3.74 ± 1
n=2 & nr=5							
S	0.65	13.64 ± 2	2.65 ± 1	8.51 ± 2	11.82 ± 2	4.77 ± 1	11.56 ± 2
P	0.73	7.38 ± 1	1.44 ± 1	4.69 ± 1	6.31 ± 1	2.43 ± 1	6.21 ± 1

Tabela 2: Desempenho dos algoritmos em 250 repetições para o conjunto de dados *Buys All Data*, contendo informação sobre as compras efetuadas por 508 clientes tendo à disposição 143 produtos, (média±desvio padrão).

Os resultados apresentados nas tabelas anteriores, demonstram que, escolhendo o parâmetro α no intervalo 0.6 a 0.95, os resultados do algoritmo proposto neste trabalho superam os outros algoritmos, em alguns casos com melhorias significativas, evidenciando a vantagem da utilização deste método em sistemas de recomendação.

Observámos também que, os valores de desempenho em função do parâmetro α não variavam significativamente para $\alpha \in (0.6; 0.95)$, indicando que, em aplicações práticas se possa fixar um α escolhido deste intervalo.

4. Generalização e Aplicação em Contexto Real

Uma mais-valia do algoritmo proposto, face aos descritos em (Huang et al., 2004, 2007) é o facto de poder ser facilmente generalizado para adaptação a situações práticas. Concretamente, ao contrário dos trabalhos de Huang et al., o algoritmo deste artigo possibilita que se inclua, na determinação das recomendações, mais informação do que apenas a disponível nas transações comerciais, permitindo uma utilização mais alargada do mesmo, como se passa a descrever.

Seguindo uma generalização do algoritmo de *PageRank* inicialmente proposta em (Meyer et al., 2008), podemos alterar o passo iterador 4 do algoritmo da seguinte forma:

$$Cp^{(k)} = C^{(k-1)} \times (\beta_0 Mc + \beta_1 M1 + \beta_2 M2 + \cdots + \beta_m Mm) \qquad (5)$$

com $0 \leq \beta_i \leq 1$ para todo i e $\sum_{i=0}^{m} \beta_i = 1$.

As matrizes Mi serão matrizes $nc \times np$, normalizadas por linhas, construídas de forma a conterem informação útil para o sistema de recomendação. Em particular, podem conter informação relativa às preferências dos clientes, personalizando as recomendações ou serem construídas de modo a realçar determinados produtos em campanhas promocionais, melhorando assim a qualidade das recomendações. Os valores dos pesos β_i podem ficar acessíveis numa eventual aplicação, permitindo ao gestor da mesma adaptar o comportamento do algoritmo consoante os seus interesses.

O teste em contexto real do algoritmo foi realizado numa empresa de comércio online de t-shirts Marvelclick (http://dezpeme.com). Inicialmente esta empresa contava com uma base de dados em MySQL e uma página realizada em PHP. A empresa não dispunha de nenhum sistema de recomendação, embora fosse apresentado aos visitantes uma lista de produtos escolhidos aleatoriamente da base de dados. Para a implementação do algoritmo optamos pela linguagem Python, especialmente pela grande rapidez da sua biblioteca numérica Numpy e também pela simplicidade das suas bibliotecas de acesso a bases de dados, em particular a MySQLdb.

Antes de proceder a implementação do algoritmo, foram efetuadas simulações de desempenho de modo a testar exaustivamente a escalabilidade do algoritmo. Estes testes tiveram como objetivo a previsão do comportamento do mesmo em contexto real, onde se espera que as bases de dados tenham um crescimento contínuo, como é o caso das páginas web de comércio eletrónico. Visto que a implementação final do algoritmo iria utilizar uma base de dados em MySQL, optámos então, por realizar os testes também em MySQL usando a biblioteca MySQLdb, tendo assim a possibilidade de medir o desempenho do algoritmo, e também o desempenho do mesmo quando combinado com a base de dados. Assim, foi possível ter uma noção real do futuro desempenho do conjunto prevendo possíveis problemas.

Os testes foram realizados, simulando bases de dados com um determinado número de clientes e produtos onde o número de transações era escolhido aleatoriamente. Foram repetidas cinco vezes as simulações, para cada par de números clientes/produtos, e registado o tempo de consulta na base de dados – processamento do algoritmo – preenchimento da base de dados com as recomendações. Os testes foram realizados num computador com um processador Intel Core Duo CPU P8400 2.26GHz, com 4Gbyte de RAM e correndo num sistema operativo Ubuntu. Os tempos de execução registados foram compilados na figura seguinte:

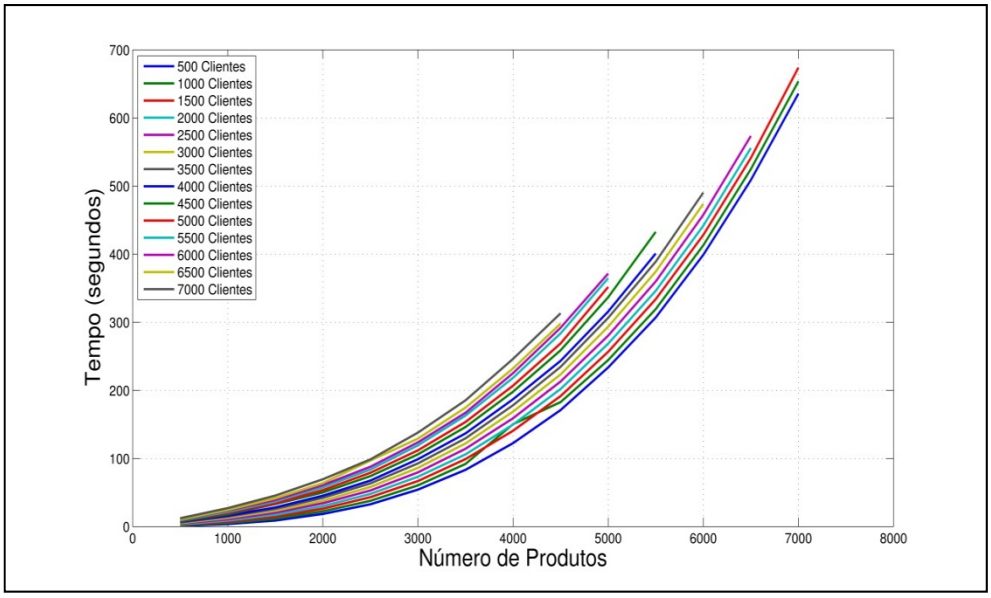


Figura 4.1: Gráfico dos Resultados dos Testes de Desempenho

Como se observa na figura anterior, podemos concluir que, numa implementação prática real, o algoritmo não poderá ser executado em tempo real, visto que o tempo de espera previsto seria da ordem das décimas de segundo, sendo um tempo de execução demasiado elevado para uma navegação rápida entre páginas. De facto, para empresas de dimensão pequena e média, como é a Marvelclick, a dinâmica de transações não é suficientemente rápida que exija uma atualização das recomendações em tempo real. Assim, é suficiente efectuar uma implementação que contemple uma atualização dos rankings de produtos guardados na base de dados com uma periodicidade da ordem das horas ou mesmo de dias. Desta forma, o algoritmo poderá correr em "back office" com uma frequência programável ou sempre que o administrador do *Website* o desejar, não decorrendo deste facto nenhuma desvantagem real do algoritmo proposto.

A implementação final consistiu num sistema que incluía o algoritmo generalizado proposto neste trabalho, assim como o interface que permitia a iteração do administrador com o mesmo. Neste interface foram definidas três modalidades de recomendações, no sentido de possibilitar um controlo da execução do algoritmo através de um interface intuitivo, permitindo adaptar a execução do mesmo a diversas

situações, tais como promoções ou alterações ao tipo de recomendação. A primeira modalidade é baseada nas compras efetuadas no website, a segunda tem em conta as preferências dos utilizadores e a última modalidade tem como base as escolhas do administrador. O controlo por parte do administrador advém do facto de que nem sempre a recomendação mais precisa é a economicamente mais viável. Por tal facto, foi dada a possibilidade ao administrador de definir pesos para todos os produtos que deseje, fazendo com que estes tenham maior probabilidade de recomendação. No interface, o administrador pode fazer pesquisas detalhadas de produtos, alterar o peso que os mesmos terão nas recomendações e influência que estes têm no algoritmo, alterando os valores dos pesos β_i, ou seja, escolhendo se as recomendações vão ser baseadas nas compras, nas preferências do administrador ou nas preferências dos clientes.

Com isto, foi possível obter um sistema que, para além de efetuar recomendações personalizadas potenciando as vendas, pode ter um papel fundamental em possíveis campanhas de marketing, assim como, no escoamento do stock residual.

5. Conclusões

Neste artigo foi proposto um algoritmo de recomendação para funcionar em aplicações de *E-commerce*. A qualidade das recomendações do mesmo foi testada, comparando o seu desempenho com outros algoritmos, usando uma base de dados comum. Os resultados dos testes realizados são muito promissores, indicando que o mesmo supera o desempenho dos algoritmos de recomendação clássicos.

Após provada a qualidade do algoritmo, foram desenvolvidas e testadas algumas funcionalidades mais vocacionadas para uma implementação em contexto empresarial de *E-commerce*. Devido à sua estrutura, o algoritmo pode ser facilmente generalizado para incluir vários tipos e níveis de informação, permitindo assim a sua utilização num leque alargado de cenários. Esta facilidade de generalização foi comprovada com uma implementação em ambiente real, no qual o algoritmo pode correr utilizando informação de transações anteriores, preferências dos clientes/utilizadores e preferências do gestor da empresa. Assim, neste trabalho foi apresentado um novo algoritmo de recomendação que, para além de registar muito bons resultados em testes controlados, é de fácil adaptação e implementação para funcionamento em lojas *on-line* reais.

Agradecimentos

Os autores agradecem os comentários e sugestões feitas pelo nosso colega Paulo Lousado e por dois revisores anónimos, que muito ajudaram a enriquecer a versão final do artigo.

Referências

Huang Z., Zeng D., Chen, H., (2005). A Link Analysis Approach to Recommendation under Sparse Data. Americas Conference on Information Systems.

Huang Z., Zeng D., Chen H., (2007). A Comparative Study of Recommendation Algorithms in E-commerce Applications, IEEE Intelligent Systems, 22(5), pp. 68-78.

Langville A., Meyer C., (2006). Google's PageRank and Beyond: The Science of Search Engine Rankings, Princeton University Press.

Li M. et al., (2009). Grocery Shopping Recommendations Based on Basket Sensitive Random Walk, KDD09, June 28 July 1, Paris, France.

Meyer C., Albright R., Govan A., (2008). Generalizing Google's PageRank to Rank National Football League Teams, Proceedings of SAS Global Forum 2008, San Antonio, TX, March 16-19, Article 151-2008.

Pucci A., Gori M., Maggini M., (2007). A Random-Walk Based Scoring Algorithm Applied to Recommender Engines , Lecture Notes in Computer Science - Springer.

Yildirim H., Krishnamoorthy M., (2008). A random walk method for alleviating the sparsity problem in collaborative filtering, RecSys 08: Proceedings of the 2008, ACM conference on Recommender systems, pp 131-138.

Zanker M., Jessenitschnig M., Jannach D., Gordea S., (2007). Comparing recommendation strategies in a commercial context, IEEE Intelligent Systems, Special issue on Recommender Systems, Vol. 22(3), pp. 69-73.