OERecommender: um Sistema de Recomendação de REA para MOOC

Ariel Gustavo Zuquello, Itana Maria de Souza Gimenes

Departamento de Informática – Universidade Estadual de Maringá (UEM)
Av. Colombo, 5.790 Jd. Universitário Maringá – PR – Brasil CEP 87020-900
arielzuquello@gmail.com, itana@din.uem.br

Abstract. *One of the key issues of Massive Open Online Courses (MOOC) is to provide mechanisms to support the learning process of their participants. Their population of participants is culturally diverse and the dropout rate is considered high, around 90%. One of the drawbacks of MOOC is the lack of open materials recommendation to support students in their learning process. In order to contribute to the improvement of this scenario, this paper proposes the OERecommender, a recommendation system for MOOC. The OERecommender aims to support students in searching for Open Educational Resources (OER) that can help in their learning process.*

Resumo. *Um pontos chave dos Massive Open Online Courses (MOOC) é oferecer mecanismos de apoio ao processo de aprendizagem de seus participantes. Sua população de participantes é culturalmente diversa e a taxa de abandono é considerada alta, em torno de 90%. Um das deficiências reconhecidas é a falta de indicação de materiais abertos que possam enriquecer a base de apoio aos alunos. Para contribuir com a melhoria deste cenário, este trabalho propõe o OERecommender, um Sistema de Recomendação para MOOC. O OERecommender visa a apoiar os alunos a encontrar Recursos Educacionais Abertos (REAs) que possam ajudar em seu processo de aprendizagem.*

1. Introdução

Os *Massive Open Online Courses* (MOOCs) são cursos online abertos, ofertados para uma grande quantidade de pessoas que podem estar em qualquer lugar do mundo em que se tenha acesso à Internet (Downes, 2008). Os MOOCs, diferente da EAD tradicional promovem o estabelecimento de redes de aprendizagem sem limite de participantes, em que todos estão ao mesmo tempo ensinando e aprendendo de forma ativa. Pesquisas são necessárias para explorar vários aspectos dos MOOCs, tanto do ponto de vista educacional e seus impactos políticos e de negócios, quanto do ponto de vista de recursos computacionais de apoio, que é o foco deste artigo.

Apesar do crescimento dos MOOCs, pouco ainda se sabe sobre a estrutura interna de suas plataformas e sobre a articulação entre as Tecnologias de Informação e Comunicação (TICs) utilizadas e os processos de aprendizagem, do ponto vista pedagógico. Um dos problemas relacionados aos MOOCs é a taxa de evasão (Chung, 2013) que está em torno de 90%. Isto deve-se a diversos fatores como: objetivos diversos dos alunos que realizam os cursos, idioma, cultura, bem como o fato deles não estarem preparados para receber e assimilar os conteúdos. Uma potencial melhoria já

aplicada em domínios semelhantes, como por exemplo, em Ambientes Virtuais de Aprendizagem (AVAs), são os Sistemas de Recomendação (SRs) (Barcellos et al. 2007). Importante ressaltar que este trabalho aborda sistemas de recomendação e não sistemas de reputação.

Um SR tem por objetivo desenvolver modelos para reduzir a sobrecarga de informações que um usuário recebe ao procurar algum produto, por intermédio da recomendação personalizada, com base no perfil do usuário (Adomavicius e Tuzhilin, 2005). As recomendações geradas sugerem novos recursos de aprendizagem para auxiliar no aprendizado e motivação do aluno a continuar o curso (Hilen, 2006). Esses recursos podem ser internos ou externos às plataformas utilizadas pelo usuário.

Os REAs são importantes recursos adicionais que podem ser utilizados por participantes de MOOCs para ampliar conhecimento ou fornecer conceitos que constituem pré-requisitos ao curso. Alguns MOOCs possuem conteúdo aberto e podem ser considerados REAs, mas muitos possuem conteúdo fechado de propriedade dos fornecedores. Assim, mecanismos que apoiem a utilização de REAs em MOOCs como os SRs, são importantes para proporcionar aos aprendizes uma diversidade de materiais de apoio no processo de aprendizagem.

Este artigo apresenta o *OERecommender*, um SR que pode ser integrado em uma arquitetura de MOOC, para apoiar a recuperação de REAs. O *OERecommender* foi avaliado por meio de uma prototipação de cenário fictício. A Seção 2 apresenta o projeto do *OERecommender*; a Seção 3 resume a avaliação realizada, na Seção 4 descreve os trabalhos relacionados. Por fim, apresenta as Conclusões.

2. Projeto do *OERecommender*

OERecommender tem sua arquitetura baseada em Govaerts et al. (2009) que utiliza um sistema de pesquisa federada juntamente com um serviço de recomendação em um contexto similar. O *OERecommender* é um componente que pode ser incorporado a qualquer Ambiente MOOC, conforme mostra a Figura 1. Ele é composto de um `Serviço de Pesquisa Federada` que por sua vez invoca um Serviço de busca de REA especializado denominado `SeeOER` (Gazzola et al. 2014); e, de duas bases de armazenamento para `Eventos - CAM` e `Estatísticas de Aprendizagem`. Os Eventos de *Contextualized Attention Metadata* (CAM) são metadados de atenção contextualizada que permitem monitorar as interações do usuário com os ambientes de aprendizagem (Wolpers e Najjar, 2007). As `Estatísticas de Aprendizagem` constituem o processo de medição, coleta, análise e divulgação de dados sobre alunos e seus contextos para fins de compreensão e evolução da aprendizagem e dos ambientes em que ela ocorre (Siemens et al. 2011).

O ***S**earch **E**ngin**E** for **O**pen **E**ducation **R**esources (SeeOER*) é um mecanismo de busca especializado em encontrar REAs considerando suas características específicas e metadados. Ele visa a resolução de conflitos em nível de esquema e em nível de instância oriundos do uso de diferentes padrões de metadados, repositórios e plataformas de REAs (Gazzola et al. 2014). *Widget* é um componente que pode ser utilizado em computadores, celulares, tablets e outros aparelhos para simplificar o acesso a um outro programa ou sistema (Helou e Salzmann, 2010).

O processo de recomendação é composto de sete etapas, conforme está numerado na Figura 1, descritas na seção 2.1.

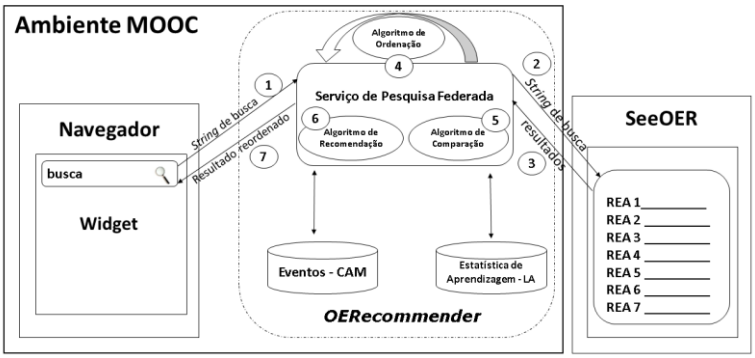

Figura 1. Arquitetura do *OERecommender*

2.1 Processo de Recomendação do *OERecommender*

O *OERecommender* utiliza filtragem colaborativa para encontrar similaridade entre os itens, como nos tradicionais SRs (Resnick e Varian, 1997; Adomavicius e Tuzhilin, 2003; Barcellos et. al 2007; Burke 2007; Jannach e Zanker, 2011) que seguem os métodos baseados em filtragem de conteúdo, colaborativa ou híbrida. A opção por esse método de filtragem deve-se ao fato de ser importante analisar as experiências obtidas pelos usuários no decorrer do MOOC, para posteriormente utilizá-las como parâmetro para sugerir bons REAs a novos usuários.

O maior diferencial do *OERecommender* é usar informações contextuais do MOOC em execução para encontrar a similaridade entre os usuários. Essas informações são posteriormente utilizadas para prever a recomendação de REA relevante ao usuário. As informações contextuais são identificadas pela quantidade de atributos semelhantes entre instâncias de CAM do usuário que já utilizou e do usuário que fará uso do REA. Essas instâncias de CAM são resultantes das interações do usuário com o MOOC e são armazenadas em uma base de dados.

O processo de recomendação consiste de sete etapas que são sumariamente descritas a seguir. Essas etapas envolvem quatro algoritmos que são descritos nas seções 2.1.1 a 2.1.4 respectivamente, a saber: análise de similaridade entre os usuários; ordenação de REA; comparação de atributos de registros de CAM; e, o algoritmo de recomendação *Slope One*. Neste artigo, chama-se de usuário alvo aquele que receberá as recomendações.

Etapa 1 e 2 – Geração da *string* e realização de busca automática

Na Etapa 1, as *strings* de busca por REAs são gerados por meio de palavras-chave, encontradas nas instâncias de CAM com as quais o usuário está interagindo. Primeiramente, o *OERecommender* captura o valor do atributo *group.title* identificando unicamente este usuário e, em seguida, captura o *group.feed.title* identificando o título da atividade que ele está realizando. Esses atributos são oriundos de eventos de CAM e armazenados em uma base de dados XML (Wolpers e Najjar, 2007). O valor desse artibuto *group.feed.title* é utilizado como termo para a busca na API do *SeeOER*.

Etapa 3 – Busca de REAs por meio do *SeeOER*

Na Etapa 3, o SeeOER (Gazzola et. al 2014) realiza a busca na Web por REAs, tendo como parâmetro de busca a *string* coletada na etapa anterior; em seguida, indexa os resultados da busca e os devolve ao serviço de pesquisa federada.

Etapa 4 – Ordenação de Resultados

Na Etapa 4 ocorre a primeira ordenação dos REAs obtidos do *SeeOER*, pois a ordem retornada pelo *SeeOER* pode não ser a melhor a ser apresentada ao usuário alvo. É nesta etapa que se encontram as similaridades entre usuários que já utilizaram alguns dos REAs retornados e sua respectiva avaliação, com o usuário alvo. Assim, é importante ter um algoritmo que os ordene da melhor forma. Os algoritmos de análise de similaridade e ordenação de REAs são descritos nas seções 2.1.1 e 2.1.2.

Etapa 5 – Comparação de Instâncias

A Etapa 5 aplica um algoritmo de comparação dos atributos das instâncias de CAM, encontrando a similaridade contextual entre usuários. O algoritmo de comparação de instâncias está detalhado na seção 2.1.3.

Etapa 6 – Recomendação de REAs

Na Etapa 6 é executado um algoritmo de recomendação. Neste projeto, optou-se pelo algoritmo *Slope One* (Lemire e Maclachlan, 2005), pois é considerado um método de recomendação fácil de implementar, com teoria simples e apresentando bons resultados práticos sendo altamente escalável. Suas predições são calculadas a partir da comparação entre avaliações de usuários a certos itens.

Etapa 7 – Reordenação dos Resultados

Após a realização das reordenações dos REAs, o *OERecommender* finaliza seu processo predizendo ao usuário alvo, por meio de *widget* os cinco REAs mais relevantes e interessantes a ele.

2.1.1 Análise de similaridade entre os usuários

O algoritmo de análise de similaridade encontra as ações similares executadas entre o usuário alvo e os demais usuários que já utilizaram os REAs. Para isso executa um passeio aleatório aplicado em um grafo bipartido. Em uma partição de nós do grafo estão os usuários que baixaram ou avaliaram os REAs retornados na busca, e na outra, estão os respectivos REAs, criando assim a correlação entre os usuários por meio da dobradura de grafos(Ochoa e Duval, 2006).

Para encontrar essa correlação, os parâmetros avaliados nas instâncias de CAM são: primeiramente o parâmetro *group.feed.item.guid* que identifica unicamente este REA e posteriormente o parâmetro *group.title* que identifica o usuário que utilizou o REA. A correlação entre as partições representa a quantidade de vezes que o REA foi baixado ou avaliado pelos usuários. Para realizar este cálculo, o *OERecommender* usa duas métricas de classificação que são (Ochoa e Duval, 2006): `Rank de Popularidade(PR)` e o `Rank Manual (MR)`.

A primeira métrica, apresentada na *Equação 1*, é para se obter o número de vezes que um REA foi baixado. Para calcular, conta-se apenas o número de ligações

incidentes que cada nó da partição dos REAs tem nos nós da partição dos usuários. Esta métrica é usada para colocar os REAs mais baixados em primeiro lugar na lista de resultados.

$$\mathbf{PR(REA) = {}_{Numero}Downloads\ (REA)} \qquad (1)$$

O `Rank Manual (MR)`, apresentado na *Equação* 2, calcula o número de vezes que o REA foi avaliado. Esta avaliação segue a escala: "5 - Ótimo"; "4– Bom"; "3 – Regular"; "2– Ruim" e; "1 – Péssimo". Esses valores são representados para o usuário por uma escala de cinco estrelas, em que a quantidade de estrelas marcadas representa a avaliação do REA. Uma avaliação de valor zero para o nível de métrica, representa a ausência de avaliação daquele REA. A métrica MR calcula os valores como pesos para encontrar relações entre usuários. O valor real para fim de cálculo da métrica é implícito ao usuário e é usado apenas para determinar se a avaliação é um "voto" ótimo (peso 3), bom (2), regular(1), ruim(-1) ou péssimo(-2), pois diferentes usuários têm diferentes avaliações.

$$\mathbf{MR\ (REA) = Avaliação_{(positiva)}\ (REA) - Avaliação_{(negativa)}\ (REA)} \qquad (2)$$

Essas métricas calculam a relevância dos REAs. A soma ponderada das duas métricas pode ser obtida com a soma ponderada dos valores das métricas individuais. A ponderação adotada nesse contexto é peso de 20% (α) sobre a métrica que envolve o número de vezes que o REA foi baixado e 80% (β) sobre o peso da métrica dos pesos atribuídos pelos usuários para aquele REA. Isso se deve ao fato de que nem sempre que o usuário baixa um REA, ele utiliza-o. Já nos casos de avaliação da qualidade, normalmente ele fez uso do REA para posteriormente avaliá-lo. A combinação dessas duas métricas é chamada de **Métrica de Popularidade Composta (MPC),** apresentada na *Equação 3* (Ochoa e Duval, 2006).

$$\mathbf{MPC = (\alpha PR) + (\beta MR)} \qquad (3)$$

2.1.2 Algoritmo de Ordenação de REA

O algoritmo de ordenação de REA é executado após a recuperação dos REAs pelo *SeeOER*, posteriormente calcula o valor dos pesos de relevância dos REAs, e encontra os usuários mais similares ao usuário alvo. Assim, devem-se ordenar os REAs mais relevantes de acordo com o aproveitamento dos usuários semelhantes e os resultados da **MPC**. Para isso, o sistema utiliza uma pós-filtragem, em que primeiramente os REAs são ordenados de forma decrescente. Utiliza-se para tal um algoritmo simples de ordenação de vetores como mostra o Quadro 1. Essa ordenação será utilizada como vetor de entrada nas próximas etapas do *OERecommender*. O *vetor_REA* contém os índices dos REAs encontrados na busca; *i, j,* e *eleito* são variáveis utilizadas no algoritmo; *N* é um conjunto de REAs retornados da busca; *B* é o conjunto de registros de CAM desses REAs armazenados no banco de dados; *k* é o percentual de aproveitamento do alvo; *l* é o percentual de aproveitamento dos usuários que baixaram ou avaliaram esses mesmos REAs em outro momento e estão armazenados em registros de CAM na base de dados.

Quadro 1 – Algoritmo de Ordenação

```
1: inicialização
2:   Se N ⊂ B e k >= l faça
3:      vetor_REA[N]
4:      para I ← 1 até N faça
5:        eleito ← vetor_REA[N];
6:        j ← n-1;
7:        enquanto (( j >=0 ) e (eleito > vetor_REA [j] )) faça
8:           vetor_REA[j + 1] := vetor_REA[j];
9:           j ← j – 1;
10:       Fim_enquanto;
11:       Vetor_REA [j + 1] ← eleito;
12:     fim_para
13:   Senão B ← B ∩ N
14: fim
```

O algoritmo de ordenação verifica se existem eventos de CAM desses índices já baixados e/ou avaliados em outro momento pelo alvo ou outros usuários na base de dados. Verifica se os mesmos tiveram um aproveitamento igual ou superior ao alvo (linha 2). Se **SIM**, o vetor é ordenado de forma decrescente calculando a métrica **MPC**. Sabendo-se quantas vezes o mesmo foi baixado e/ou avaliado e assim é possível reordenar a primeira sequência retornada do motor de busca, colocando em ordem decrescente os REAs que tiveram maior valor no **MPC** no início do vetor (linha 3 a 10). Caso os REAs retornados da busca **NUNCA** tiverem sido baixados e/ou avaliados, ou o percentual de aproveitamento desses usuários encontrados na base de dados são inferiores ao do alvo, o algoritmo não altera a sequência original dos índices retornados pelo motor de busca do repositório (linha 11).

2.1.3 Algoritmo de Comparação

O algoritmo de comparação de atributos de registros de CAM é apresentado no Quadro 2. O *vetor_REA* contém o vetor retornado com o número de REAs encontrados com a *string* de busca, *Base_A* é o conjunto de todos os valores dos atributos dos registros de CAM de outros usuários na base de dados; Base_*C* é o conjunto de todos os valores dos atributos de CAM do usuário alvo; *P* é quantidade de atributos iguais entre os registros da base com os REAs da busca. Este algoritmo reordena o vetor, após fazer uma interseção de todos os atributos dos registros de CAM dos usuários que já baixara ou avaliaram esses REAs com os atributos dos registros de CAM do alvo (linha2) e retorna o tamanho dessa interseção (linha 3).

Quadro 2 – Algoritmo de Comparação

```
1: Tamanho_intersecao (vetor_REA, Base_C)
2:       P← Base_A ∩ Base_C
3:       retorna P.tamanho
```

Para cada similaridade encontrada entre os atributos da instância do alvo e dos demais usuários, o algoritmo de comparação incrementa no peso da correlação um valor "+1" entre os usuários. A técnica utilizada para a métrica é a de dobradura de grafos (Ochoa e Duval, 2006), O cálculo da métrica de similaridade entre os usuários é feito conforme a *Equação 4*.

$$Métrica\ de\ Similaridade\ (MS) = \sum \frac{(Peso)}{(Distância)} \qquad (4)$$

2.1.4 Algoritmo Slope One

O algoritmo *Slope One* opera nas notas que os usuários deram aos itens. Essas notas são colocadas em uma matriz de *Usuários X Itens*, de tal forma que cada linha corresponda às notas de um usuário *j* a *N* itens. Se um usuário *j* não tiver dado notas a um item *i*, o elemento $x_{i,j}$ fica igual a 0. A Figura 2 representa a matriz de um conjunto de notas:

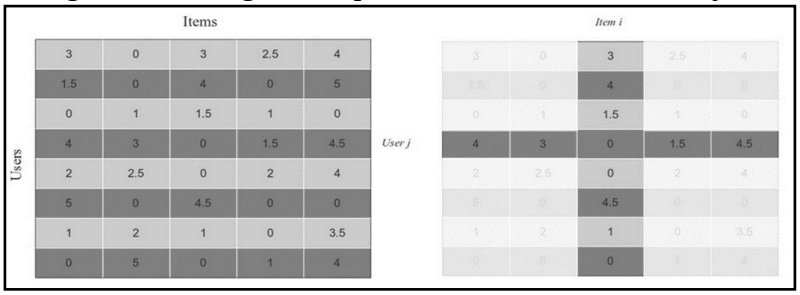

Figura 2 - Matriz dos Itens x Usuários (Lemire e Maclachlan, 2005)

Diferentemente de outras abordagens colaborativas, o *Slope One* cria uma relação linear entre os dados, usando a seguinte fórmula $f(x) = x + b$, onde a variável x representa as avaliações feitas pelo utilizador, enquanto b representa o desvio. É importante salientar que o tamanho da vizinhança considerada é tipicamente limitado a um tamanho específico, portanto, nem todos os vizinhos são levados em consideração para a previsão. Em nosso contexto específico, serão os 4 vizinhos com maior peso de similaridade e contextos similares com nosso usuário alvo.

3. Avaliação

Para avaliação foi criado um protótipo de cenário fictício a partir de um MOOC sobre "Introdução a Programação" disponibilizado no ambiente Google Course Builder (GCB). Foi implementado o algoritmo de recomendação *Slope One* enquanto as saídas dos demais algoritmos foram assumidas do cenário idealizado. A implementação do algoritmo *Slope One* foi feita em linguagem *Python*. O mecanismo de busca *SeeOER* foi disponibilizado para teste por Gazzola et al. (2014).

Inicialmente, a *string* `programming` foi submetida ao *SeeOER*. Como resultado obtivemos 1.113 REAs, dos quais os cinco mais bem ranqueados foram selecionados para nossa avaliação, sendo eles: *SI/Coursera – Programming for Everbody*, *Everything maths and Science*, *Introduction to programming*, *Introduction to programming in Java* e *J2EE Tutorial*. Posteriormente, tabulou-se o resultado da busca, como uma matriz de Usuários x REAs, resultado do grafo bipartido do algoritmo similaridade e comparação e também foi gerada uma avaliação randômica para cada REA, simulando as avaliações que os usuários fizeram aos REAs.

Esta simulação substitui o algoritmo de similaridade e comparação, que encontra os vizinhos similares, compara as instâncias de CAM e ranqueia os REAs por relevância. Assim, foi criado um dicionário de dados simples para ser submetido ao algoritmo de recomendação *Slope One*, conforme mostra a Tabela 1. A Tabela 1 e composta por colunas, onde temos os REAs e as linhas onde temos os usuários fictícios,

dentre eles o alvo que é **Aragorn,** e os vizinhos com maior peso de similaridade e contextos similares ao alvo que são: Frodo, Gandalf, Bilbo e Faramir. Neste dicionário a única nota que não foi disponibilizada foi a que queremos obter e que será calculada pelo *Slope One*, a nota do REA: *Everything maths e Science* para um usuário alvo, **Aragorn.**

Tabela 1 – Matriz com classificação dos REAs já avaliados pelos usuários.

	SI / Coursera - Programming for Everybody	***Everything maths and Science***	***Introduction to Programming***	***Introduction to programming in Java***	***J2EE Tutorial***
Frodo	3	2	2	-2	-1
Gandalf	3	1	3	1	1
Aragorn	3	***(1.44)***	-1	1	-2
Bilbo	2	3	-2	1	-1
Faramir	3	2	2	-1	-1

Após a execução do algoritmo *Slope* One, a previsão para a avaliação de **Aragorn** ao REA foi **1,44**, portanto, é um REA que foi recomendado a ele. Em nossa prototipação, todo REA com avaliação superior a zero deve ser ranqueado e consequentemente recomendado. Para precisar o ranqueamento dos REAs mais relevantes, é necessário a submissão de uma quantidade maior de dados ao algoritmo e realização de experimentos de todo o *OERecommender* no GCB, tarefa que será realizada em trabalhos futuros.

4. Trabalhos Relacionados

Alguns trabalhos foram usados como base, embora esses não tenham sido concebidos para serem diretamente utilizados em MOOC. Hsu (2008) descreve um SR para um curso online de inglês capaz de recomendar materiais que auxiliam no processo de aprendizagem. Essas recomendações são geradas por meio de filtragem baseada em conteúdo, filtragem colaborativa e técnicas de mineração de dados. Experimentos realizados durante um ano mostraram que a maioria dos estudantes aceitaram as recomendações de materiais indicadas e lições sugeridas pelo SR, provando sua viabilidade e eficiência.

Nosso trabalho possui características similares ao trabalho desenvolvido por Hsu (2008), principalmente no que tange aos métodos de filtragem dos materiais para posterior recomendação e também na forma de sugerir materiais que realmente sejam de interesse dos usuários alvo.

Wolpers e Najjar *(*2007) retratam o desenvolvimento do modelo CAM que permite monitorar o comportamento do usuário com o computador, em que, como e quanto tempo ele utilizou determinado ambiente e armazenar essas informações em uma base de dados para posterior utilização. Este trabalho foi importante para o desenvolvimento do *OERecommender*, pois gerencia as informações contextuais das atividades que o usuário realiza no MOOC servindo assim como artefato de entrada para o processo de busca por REAs realizadas pelo *SeeOER* (Gazzola et al. 2014) em diversos repositórios na Web.

Ochoa e Duval (2006) utilizaram métodos de similaridade entre usuários por meio de dobradura de grafos, método esse que foi replicado no *OERecommender*, pois já havia sido testado pelos autores e com bons resultados em contextos similares, por isso foram utilizados na concepção desse SR.

O trabalho aqui proposto se difere dos acima referenciados, pois recomenda REAs em ambientes MOOC. Para tal o *OERecommender* considera informações contextuais, gerenciando os metadados de CAM necessários para realizar a busca por REAs na Web, sendo auxiliado pelo *SeeOER* (Gazzola et. al, 2014) um buscador específico para encontra-los. Esses fatores somados, que tornam o *OERecommender* um SR diferenciado dos demais e credenciam a sua implementação em ambiente real.

5. Conclusão

Atualmente SR já estão presentes em diversos ambientes virtuais de aprendizagem. No entanto, sistema de recomendação de REAs em ambientes MOOC não são comuns, fazendo-se necessário o desenvolvimento de um projeto para sua implementação em quaisquer plataformas. Este artigo enfatiza informações contextuais (Wolpers e Najjar, 2007), algoritmos (Lemire e Maclachlan, 2005) e métodos matemáticos (Ochoa e Duval 2006) consolidados para realizar a busca e ordenação dos REAs, similaridade entre perfis e também para fazer a predição dos REAs aos usuários.

Como principal contribuição do *OERecommender* frente ao estado da arte no campo de SR em ambientes MOOC é justamente a forma com que o mesmo trata o processo de recomendação, utilizando informações contextuais como filtragem dos REAs para o processo de predição e não apenas utilizando um algoritmo de aprendizagem de máquina como nos demais métodos de recomendação utilizam, como por exemplo o trabalho de Barcellos et al. (2007). Com isso, a predição se torna muito mais eficaz desde as primeiras recomendações, minimizando o tempo de aprendizagem de máquina que ocorre nos demais métodos e também tendo uma maior satisfação pelo usuário alvo, que será evidenciada por trabalhos futuros, fruto deste trabalho.

Referências Bibliográficas

Adomavicius, G. and Tuzhilin, A. (2005) “Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions”. IEEE Transactions on Knowledge and Data Engineering, p734-749. http://www.computer.org/portal/web/csdl/doi/10.1109/TKDE.2005.99

Barcellos, M. C. D. and Brandão, L. D. and Warpechowski, A. L. (2007) “Sistema de Recomendação Acadêmico para Apoio a Aprendizagem”, http://www.cinted.ufrgs.br/ciclo10/artigos/3fDaniela.pdf

Burke, R. (2007), “Hybrid web recommender systems”. Springer, Berlin/Heidelberg. Berlin / Heidelberg.

Chung, C. (2013) “Beyond Dropouts and Dabblers: A Broader View of Auditing MOOCs”, MOOC News and Reviews, http://moocnewsandreviews.com/beyond - dropouts-and-dabblers-taking-a-broader-view-of-auditing-moocs/

Downes, S. (2008) “Places to go: Connectivism e Connective Knowledge”. Innovate, 5(1). http://www.innovateonline.info/index.php?view=articleeid=668

Gazzola, M. and Ciferri, C. and Giemenes, I. (2014) "SeeOER: Uma Arquitetura para Mecanismo de Busca na Web por Recursos Educacionais Abertos", III Congresso Brasileiro de Informática na Educação (CBIE 2014) p. 1013-1022. http://www.br-ie.org/pub/index.php/sbie/article/view/3042/2553

Govaerts, S. and Verbert, K. and Dahrendorf, D. and Ullrich, C. and Schmidt, M. and Werkle, M. and Chatterjee, A. and Nussbaumer, A. and Renzel, D. and Scheffel, M. and Friedrich, M. and Santos, J. and Duval, E. and Law, E. (2009) "Towards Responsive Open Learning Environments: The ROLE Interoperability Framework", https://lirias.kuleuven.be/bitstream/123456789/319048/1/role-ectel.pdf

Helou, S. and Salzmann, C. (2010) "The 3a personalized, contextual and relation-based recommender system". Journal of Universal Computer Science (J.UCS), v. 16, n. 16, p. 2179–2195.

Hilen, J. (2006**). "**Open Educational Resources: Opportunities and Challenges". OECD's Centre for Educational Research and Innovation. http://www.oecd.org/dataoecd/5/47/37351085.pdf

Hsu, M. H. (2008) "A personalized English learning recommender system for ESL students". Elsevier, p 683-688. http://cgit.nutn.edu.tw:8080/cgit/PaperDL/JLL_121119095754.PDF

Jannach, D. and Zanker, M. (2011) "Recommender Systems an Introduction". [S.l.]: Cambridge University.

Lemire, D. and Maclachlan, A. (2005) "Slope One Predictors for Online Rating-Based Collaborative Filtering". SIAM Data Mining 2005.

Ochoa, X. and Duval, E. (2006) Use of contextualized attention metadata for ranking and recommending learning objects. CAMA'06, v.11. Arlington, Virginia, USA.

Resnick, P. and Varian, H. (1997), "Recommender systems". Communications of the ACM, v. 40, n. 3, p. 56–58.

Siemens, G. and Gasevic, D. and Haythornthwaite, C. and Dawson, S. and Shum, S. and Duval, E. and Verbert, K. and Baker, A, (2011): **"**Open learning analytics: an integrated and modularized platform: Proposal to design, implement and evaluate an open platform to integrate heterogeneous learning analytics techniques". http://solaresearch.org/OpenLearningAnalytics.pdf

Wang, P and Ye, H. (2009) ", A Personalized Recommendation Algorithm Combining Slope One Scheme and User Based Collaborative Filtering", International Conference on Industrial and Information Systems (IIS 2009), IEEE.

Wolpers, M. and Najjar, J. (2007). Tracking actual usage: the attention metadata approach. Educational Technology e Society, v. 10, n. 3, p. 106–121., 2007