DESTAQUES ACADÊMICOS
UNIVATES | CENTRO DE CIÊNCIAS EXATAS E TECNOLÓGICAS

Revista Destaques Acadêmicos, Lajeado, v. 12, n. 4, 2020. ISSN 2176-3070
DOI: http://dx.doi.org/10.22410/issn.2176-3070.v12i4a2020.2735
http://www.univates.br/revistas

SISTEMA DE RECOMENDAÇÃO DE EVENTOS UTILIZANDO O FRAMEWORK APACHE MAHOUT

Willian Valmorbida[1], Christian Rodolpho Bugs[2]

Resumo: Atualmente, as instituições de ensino oferecem aos estudantes uma infinidade de eventos. Entretanto, devido a essa grande variedade, os alunos acabam tendo dificuldades para decidir no que se inscrever. Uma alternativa a este problema é a utilização de um Sistema de recomendação (SR), possibilitando a instituição de ensino ofertar aos alunos os melhores eventos de acordo com seu perfil e interesses. Posto isto, o objetivo principal deste trabalho foi desenvolver um sistema de recomendações de eventos, para a Univates. No processo de desenvolvimento do SR, foi utilizada a técnica de aprendizagem por máquina, sendo que os dados filtrados foram processados com o auxílio da *framework* Apache Mahout, através de filtragem colaborativa, que é a filtragem que busca perfis similares ao do usuário para criar recomendações. A fim de verificar a qualidade das recomendações geradas, foi realizada uma avaliação com usuários reais, sendo eles alunos e funcionários da instituição, a fim de gerar 3 recomendações. Com relação às avaliações feitas para as 3 recomendações geradas, o resultado de pontuação média de 7,78 demonstra a grande aceitação dos usuários ao sistema proposto. Os resultados demonstram que o objetivo geral de criar um sistema de recomendações de eventos utilizando a *framework* Apache Mahout foi atingido.

Palavras-chave: Sistema de recomendação. Eventos. Filtragem colaborativa. Aprendizagem por máquina. Apache Mahout.

1. Introdução

Atualmente as instituições de ensino oferecem aos estudantes muitas opções de eventos, como palestras, shows, oficinas, cursos e congressos. Apesar disso ser benéfico, os alunos acabam enfrentando dificuldade para decidir no que se inscrever ou até de encontrar eventos que realmente interessam a eles.

A partir desta problemática, este trabalho apresenta a criação de um sistema para realizar a recomendação dos eventos, por meio de uma filtragem

1 Mestre em Computação Aplicada – Unisinos-RS e professor do Centro de Ciências Exatas e Tecnológicas da Universidade do Vale do Taquari – UNIVATES.

2 Bacharel em Engenharia da Computação – Univates-RS.

colaborativa, baseada no usuário. Com a disponibilização de recomendações automáticas, busca-se personalizar a experiência do usuário, direcionando para ele itens dos quais haja provável interesse, melhorando assim a divulgação desses eventos. A solução apresentada neste trabalho envolve a modelagem e implementação de um protótipo de Sistema de Recomendação de serviços oferecidos pela Universidade do Vale do Taquari - Univates, tais como cursos, palestras e shows. Pretende-se, por meio da filtragem colaborativa, recomendar os eventos mais adequados para cada usuário.

Para o desenvolvimento deste projeto, foi utilizada a filtragem colaborativa, tendo em vista que essa estrutura de recomendação não analisa o tipo do item e nem seus atributos. Considera-se a filtragem colaborativa adequada, já que permite que os eventos sejam recomendados de acordo com os gostos e preferências de alunos com usuários similares. Como o protótipo do sistema de recomendação requer uma programação sofisticada, foi utilizada técnica de aprendizagem por máquina e os dados filtrados foram processados com o auxílio da biblioteca Apache Mahout

Neste contexto, o Apache Mahout apresenta-se como uma biblioteca de software livre, com algoritmos para mineração de dados distribuída, desenvolvida em Java e gerenciada pela Apache Software Foundation, cujo objetivo é criar algoritmos de aprendizagem por máquina (INGERSOLL, 2009). Embora o Apache Mahout seja um projeto aberto a implementações de todos os tipos de técnicas de aprendizado de máquina, na prática, se concentra em três áreas principais do aprendizado de máquina: mecanismos de recomendação (filtragem colaborativa), armazenamento em cluster e classificação (OWEN, 2012).

2. Referencial Teórico

Na presente seção é apresentado um breve referencial teórico sobre os Sistemas de Recomendação, abordando conceitos, técnicas de implementação, vantagens e limitações.

2.1 Sistemas de Recomendação

Com o grande número de eventos, palestras, cursos e congressos oferecidos pelas instituições de ensino, os estudantes acabam tendo dificuldade em tomar conhecimento e decidir no que se inscrever e participar. Neste sentido, os SR podem ser utilizados como uma ferramenta para recomendar conteúdos com base nas necessidades ou gosto do estudante (ROLIM *et al*., 2017). Atualmente os SR são amplamente utilizados na web, tornando possível recomendar ao usuário: filmes, produtos, restaurantes, viagens e uma infinidade de itens com base nos sites em que navegaram (AGGARWAL, 2016).

Segundo Schafer, Konstan e Riedl (2000) a estrutura de um SR é dividida em quatro processos: identificação do usuário, coleta de informações,

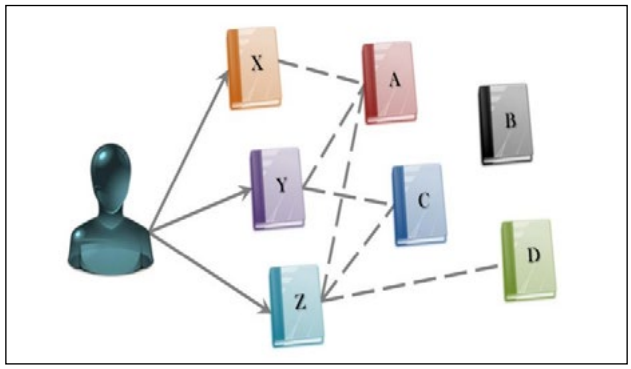
estratégias de recomendação e visualização das recomendações. A coleta dos dados do usuário pode ser feita de forma explícita, implícita ou por inferência. Na forma explícita, os dados são informados diretamente pelo usuário, como por exemplo, em um formulário onde ele informa suas preferências. Na forma implícita, as informações podem ser coletadas através de monitoramento da navegação do usuário pelo site, por um histórico de compras, conteúdos vistos pelos usuários, etc. A coleta de dados por inferência consiste em descobrir o perfil de um usuário através da comparação entre padrões de comportamento de usuários similares. Após esta coleta de dados, são criadas as recomendações, com base na estratégia de recomendação definida pelo desenvolvedor do SR (RICCI; ROKACH; SHAPIRA, 2015).

Com o usuário identificado e seus dados armazenados se faz necessário um bom sistema de filtragem de informações que permita definir os itens a serem recomendados (produtos, serviços, viagens, etc.). Basicamente existem três tipos de filtragem de informações que podem ser aplicadas em um SR, a baseada em conteúdo, a colaborativa e a híbrida (GUY, 2015; MELVILLE; SHINDHWANI, 2017).

2.2 Filtragem Baseada em Conteúdo

A técnica de filtragem baseada em conteúdo (FBC) pode analisar os produtos e serviços já pesquisados ou adquiridos pelo usuário e então fazer a recomendação de conteúdo similar (AGUIAR; FECHINE; COSTA, 2018). A Figura 1, ilustra a técnica de Filtragem Baseada em Conteúdo, na ilustração o usuário leu os livros X, Y e Z e o livro B não possui nenhuma semelhança com os demais livros, sendo assim o livro B não será recomendado para este usuário; porém o livro A tem características semelhantes aos três livros lidos, e acabará sendo recomendado; os livros C e D também possuem semelhança com ao menos um dos livros lidos, sendo assim, dependendo do nível de similaridade utilizado no sistema, eles também podem ser recomendados.

Figura 1 – Filtragem baseada em conteúdo



Fonte: Costa, Aguiar e Magalhães (2019).

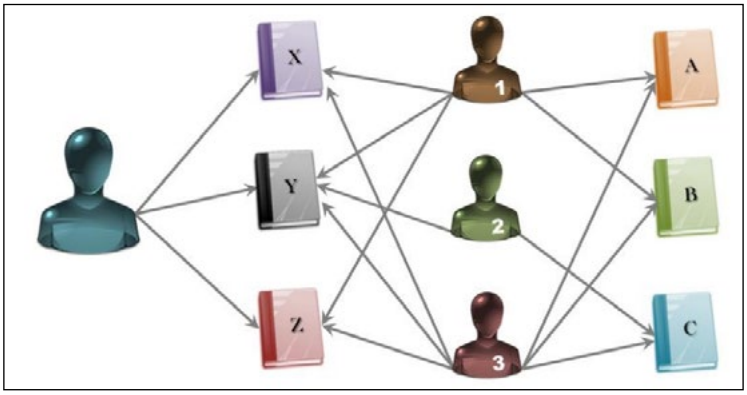
Dentre as vantagens da FBC pode-se destacar a possibilidade de encontrar bons resultados sem que haja necessidade dos usuários compartilharem informações sobre preferências, além das recomendações que independem do número de usuários e que melhoram com o tempo. Quanto às desvantagens, tem-se o baixo desempenho quanto há poucos dados sobre o usuário e a falta de relacionamento entre os usuários (AGUIAR; FECHINE; COSTA, 2018; CONCEIÇÃO, *et al*., 2016).

2.3 Filtragem Colaborativa

Segundo Barros (2014) a técnica de filtragem colaborativa (FC) baseia-se na similaridade entre perfis de usuários, ou seja, como é visto na Figura 2, o usuário alvo tem seu perfil similar ao dos usuários 1 e 3, pois ambos gostaram dos mesmos livros, suas preferências são comparadas e os mesmos conteúdos serão recomendados para ambos e para quem mais for compatível com seus perfis, gerando assim um grupo de usuários. O usuário alvo, que receberá as recomendações, gostou dos livros X, Y e Z, assim como os usuários 1 e 3, que por isso são os usuários mais similares ao usuário alvo. Além de terem gostado dos mesmos livros do usuário alvo, os usuários 1 e 3 gostaram do livro A e B, sendo assim o sistema vai recomendar esses dois livros ao usuário alvo.

Outra característica dessa técnica de filtragem é que os usuários podem avaliar os itens adquiridos através de pontuação. Essas pontuações são armazenadas na base de dados e grupos de pessoas com perfis similares são formados, assim outros usuários podem se beneficiar dessas pontuações (BARROS, 2014).

Figura 2 – Filtragem colaborativa



Fonte: Costa, Aguiar e Magalhães (2019).

O processo de filtragem colaborativa acontece em três passos: representação dos dados de entrada, formação de vizinhança e geração da recomendação. Na etapa inicial de representação dos dados o usuário indica

seus interesses e preferência, geralmente através de avaliações. Na etapa seguinte de formação de vizinhança, o sistema compara perfis de usuários para encontrar a similaridade e formar grupos de pessoas com interesses em comum. A etapa final é a geração da recomendação que filtra as avaliações feitas pelos componentes da vizinhança, o sistema gera então recomendações que visem agradar todos os usuários daquele grupo, conforme Mendes, Santos e Picoli (2018). A maior vantagem da utilização da FC é a recomendação de itens com base no histórico de outros usuários relacionados. Já entre as desvantagens estão o baixo desempenho se o usuário não tiver uma quantidade considerável de relacionamentos e a impossibilidade de recomendar itens recém adicionados ao sistema que ainda tenham sido classificados por nenhum usuário (BARROS, 2014; MENDES; SANTOS; PICOLI, 2018).

2.4 Filtragem Híbrida

A filtragem híbrida (FH) faz a combinação de duas ou mais técnicas, com o objetivo de aproveitar suas vantagens, de modo a desenvolver um sistema otimizado e que recomende o conteúdo mais adequado para o usuário (BARROS, 2014). A hibridização dessas técnicas possibilita a criação de um sistema capaz de ter bons resultados para usuários incomuns, recomendações precisas independentemente do número dos usuários, descoberta de similaridades entre os usuários e também a recomendação relacionada com o histórico do usuário (BARROS, 2014).

2.5 Aprendizagem por Máquina

Devido crescente demanda de sistemas deste tipo, a aprendizagem por máquina vem ganhando espaço nas pesquisas relacionadas à computação (AGGARWAL, 2016). A aprendizagem por máquina é um campo da inteligência artificial que torna os computadores capazes de aprender sem serem explicitamente programados. O processo de aprendizagem por máquina proporciona uma melhora em resultados gerados por computadores através da utilização de suas experiências anteriores como base. Esse tipo de ferramenta é utilizada para facilitar a construção de aplicativos, sites e sistemas inteligentes (QUILICI-GONZALEZ; ZAMPIROLLI, 2015; AMARAL, 2016; FERRARI; SILVA, 2016). Apesar de a aprendizagem por máquina ser uma tecnologia relativamente nova, grandes empresas líderes do setor tecnológico já vêm explorando seus recursos.

O que torna a aprendizagem por máquina tão relevante é que ela abrange desde a análise da bolsa de valores até a construção de sistemas de recomendação de grandes sites de vendas, que recomendam produtos aos usuários com base em compras passadas e artigos similares. As duas técnicas de aprendizagem por máquina mais utilizadas são a aprendizagem supervisionada e não supervisionada (FERRARI; SILVA, 2016).

2.5.1 Aprendizagem Supervisionada

A aprendizagem supervisionada tem como base a tarefa de aprender uma função a partir de dados de treinamento rotulados para prever o valor de qualquer entrada válida. Nela existe uma espécie de professor externo, que possuí um conhecimento sobre o assunto. Sendo assim, o algoritmo é treinado se baseando nos conhecimentos passados por seu professor (LORENA; CARVALHO, 2003; AMARAL, 2016; FERRARI; SILVA, 2016).

Em outras palavras a tarefa de atribuir sentido a dados é feita com base em exemplos do que é correto ou incorreto. Alguns exemplos comuns de aprendizagem supervisionada já foram apresentados na seção anterior, são eles: classificação de mensagens de e-mail, classificação de páginas de Web de acordo com seu conteúdo, entre outros.

2.5.2 Aprendizagem Não Supervisionada

Na aprendizagem não supervisionada, a tarefa de atribuir sentido a dados é feita sem quaisquer exemplos do que é correto ou incorreto, sendo mais utilizada para agrupar entrada similares em grupos lógicos. Neste caso não existe a presença de um professor, ou seja, não existe um especialista que passará seus conhecimentos, o algoritmo aprende a interpretar as entradas segundo uma medida de qualidade (LORENA; CARVALHO, 2003; AMARAL, 2016; FERRARI; SILVA, 2016).

3. Procedimentos Metodológicos

O presente trabalho apresenta o desenvolvimento de um sistema de recomendações de eventos, utilizando o *framework* Apache Mahout. Visando cumprir este objetivo foi feita uma pesquisa exploratória dos temas relacionados. Uma pesquisa de caráter exploratório desenvolve-se através de três etapas: o desenvolvimento de uma hipótese, a construção de um referencial teórico consistente para aumentar o conhecimento sobre o assunto a ser estudado e por último a identificação do fato ou fenômeno (MARCONI; LAKATOS, 2010).

De modo a enriquecer o trabalho foi realizada uma busca na literatura acerca do tema, o que levou a necessidade do uso do método de pesquisas bibliográficas. Segundo Gil (2002), este método de pesquisa faz o uso de referenciadas em bibliografias a respeito do tema da pesquisa, para poder melhor fundamentar o trabalho científico.

Quando aos procedimentos metodológicos, a pesquisa se caracteriza como experimental, visto que, para validar o sistema proposto, foi desenvolvido um protótipo. O mesmo foi validado por estudantes e funcionários da instituição através de uma pesquisa quantitativa, que fornece informações numéricas, possibilitando a medição da percepção dos participantes quanto as recomendações geradas pelo protótipo (PRODANOV; FREITAS, 2013).

4. Desenvolvimento

Esta seção apresenta o contexto de modelagem e desenvolvimento da solução proposta, de modo a detalhar sua especificação, modelagem, tecnologias empregadas e implementação.

4.1 Especificação do Projeto

Para a comunicação entre a interface que exibirá as recomendações e a base de dados com as informações dos usuários, foi desenvolvida uma Application Programming Interface (API), que será a responsável por tratar os dados dos usuários, formatá-los e enviar para o Apache Mahout gerar as recomendações, que por sua vez, deve encaminhar as recomendações de volta a API, culminando na exibição das mesmas na interface do usuário.

A API será chamada através de um *webservice* Simple Object Access Protocol (SOAP), recebendo o identificador do usuário que receberá as recomendações e o tipo de serviço que será recomendado. Cada tipo de serviço terá uma classe dentro da API, assim, caso seja disponibilizado um novo serviço pela instituição, basta criar uma nova classe para permitir que o mesmo também possa obter recomendações. Estas classes serão as responsáveis por buscar os dados na base e formatá-los para o modelo de dados do Apache Mahout, além de devolver as recomendações geradas para o sistema que fez a solicitação.

Na Figura 3, está representado o fluxo de dados do projeto, onde usuários que utilizam os sistemas geram informações, que são gravados no banco de dados. Ao mesmo tempo em que diferentes usuários estão abastecendo esse banco, o Apache Mahout recebe esses dados e busca encontrar perfis semelhantes ao da pessoa que está acessando o sistema, para assim fazer as recomendações.

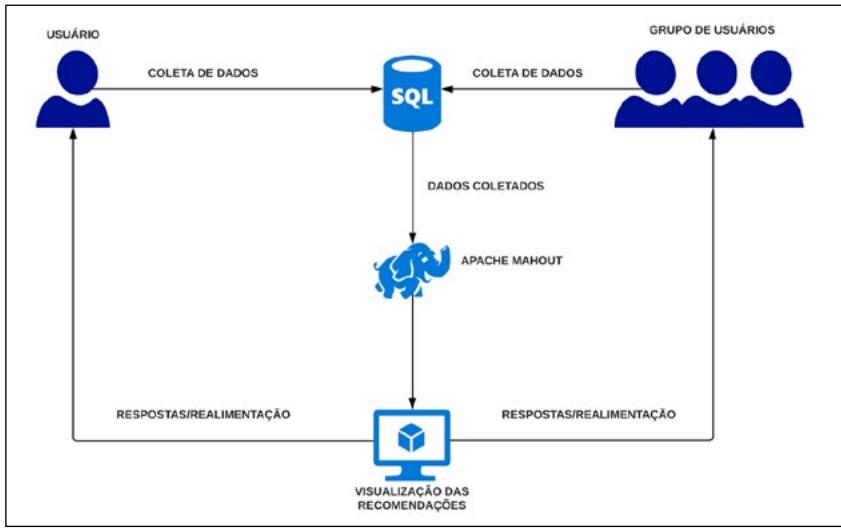
Para melhor entendimento, o fluxo da aplicação está dividido em três partes:

a) Usuário: Usuário que acessa os ambientes virtuais da Univates; possível interessado em algum serviço;

a) Banco de dados: Banco de dados onde estão os registros dos outros usuários, ou seja, sua vizinhança;

b) Apache Mahout: Com base na vizinhança e no usuário, faz o cruzamento dos dados, buscando similaridades, para assim fazer as recomendações.

Figura 3 – Fluxograma do projeto



Fonte: Do Autor (2019).

4.2 Requisitos

Existem basicamente dois tipos de requisitos, os funcionais, que são funcionalidades ou serviços que o sistema deve ter, e os não funcionais, que são aqueles que definem como as funcionalidades serão implementadas.

Para o desenvolver o sistema proposto, alguns requisitos funcionais e não funcionais foram identificados, conforme apresentado na Tabela 1:

Tabela 1 – Requisitos funcionais

Requisito	Prioridade
RF01 - Gerar Recomendações aos usuários	Alta
RF02 - Exibir as Recomendações	Alta
RF03 - Desenvolver API para integração entre sistemas	Alta
RF04 - Alertas de erros em serviços	Média
RF05 - Cadastro de usuários	Alta
RF06 - Controle de Acessos	Alta
RNF01 - Utilizar um servidor Apache	Alta
RNF02 - Utilizar o *Framework* Apache Mahout	Alta
RNF03 - Utilizar HTML5, CSS3	Alta
RNF04 - Utilizar PHP 7.2	Alta

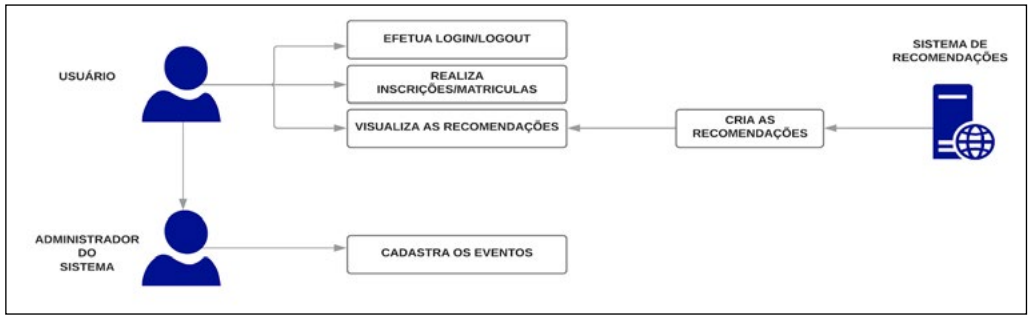
Requisito	Prioridade
RNF05 - Utilizar Adianti *Framework*	Alta
RNF06 - Utilizar PostgreSQL	Alta
RNF07 - Compatível com navegador Google Chrome	Alta

Fonte: Do Autor (2019).

4.3 Modelo de Casos de Uso

Para demonstrar os principais papéis dentro do projeto proposto, foi utilizado um diagrama de casos de uso, como visto na Figura 4.

Figura 4 – Modelo de casos de uso



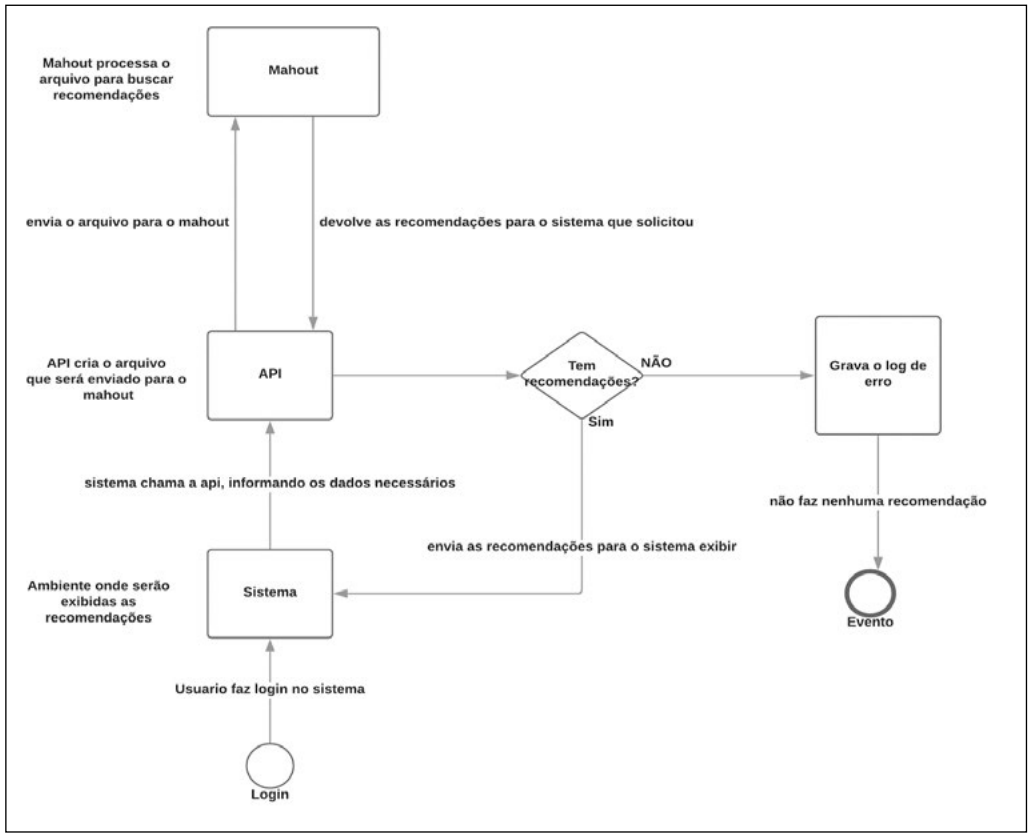
Fonte: Do Autor (2019).

São dois grupos de usuários, os que cadastram os eventos a serem oferecidos, e o público-alvo, que vai fazer a sua inscrição. Os administradores são responsáveis por cadastrar os eventos nos sistemas, informando os dados necessários para disponibilizá-los para o público, tais como o tipo do evento, as datas de inscrição e valores. Já o usuário trata-se do público, é quem utiliza o sistema buscando eventos para se inscrever e recebe as recomendações.

4.4 Diagrama de Estados

A Figura 5 representa um diagrama de estados, buscando reproduzir o cenário, onde um usuário faz o login em algum dos sistemas onde serão exibidas recomendações.

Figura 5 – Diagrama de estados



Fonte: Do Autor (2019).

Este sistema chama a API, passando para ela os dados necessários para a API criar o arquivo que será enviado para o Apache Mahout realizar as recomendações. A API recebe o retorno do Apache Mahout e o processa, caso existam recomendações, ele as retorna para o sistema que fez a solicitação.

4.5 Implementação

Nesta seção é apresentada a implementação do projeto, especificando tecnologias e algoritmos utilizados. Visando o desenvolvimento do sistema proposto, as seguintes ferramentas e tecnologias foram utilizadas: Servidor Apache, Apache Mahout, PHP, Adianti *Framework* e PostgreSQL.

4.5.1 Estrutura do Banco de Dados

O banco de dados do protótipo foi estruturado para contemplar a estrutura obrigatória para utilização do controle de permissões de acesso, disponibilizada pelo *framework* Adianti, contendo cadastros completos de

usuários, grupos, unidades, programas e permissões, assim como os recursos necessários para gerar as recomendações, esta parte sendo composta por duas tabelas.

A tabela "conexao" armazena dados de acesso aos sistemas de onde são buscados os dados para a formação do DataModel que será utilizado para gerar as recomendações. Já na tabela "fonte_dado", são inseridas as consultas que buscam os dados no banco de dados das conexões cadastradas anteriormente. O campo "sql", armazena a consulta em linguagem SQL que será executada para retornar os dados que alimentarão o DataModel.

4.5.2 Implementação do Sistema de Recomendação

O sistema de recomendações foi desenvolvido fazendo uso do *framework* Apache Mahout. O sistema de recomendações recebe como parâmetro o código que identifica unicamente a pessoa que receberá as recomendações, e o nome do arquivo utilizado pelo DataModel. O DataModel é onde estão as informações das preferências do usuário. Para aplicações que vão buscar os dados diretamente na fonte de dados, o *framework* fornece a classe FileDataModel, que permite a recuperação dos dados de um arquivo CSV.

No DataModel, usuários e itens são identificados apenas por um valor numérico. Um terceiro campo, com a força da preferência do item pode ser informado, que pode ser qualquer número, desde que valores maiores representem preferências positivas mais fortes. Caso não tenha valores de preferência, o Apache Mahout suporta o modelo de dados "booleano", no qual os usuários não expressam preferências, ou seja, existe apenas uma noção de associação ou não, entre um usuário e os itens (OWEN, 2012).

Na Univates os usuários podem atribuir uma nota para expressar sua satisfação em relação a um evento após sua participação, no entanto, estas informações ficam registradas em um formulário a parte do sistema de isncrições, ficando dificultado o relacionamento das notas atribuídas pelos usuários à base que registra os dados sobre os eventos. Além disso, a obtenção da opinião do usuário é algo mais recente que o sistema de inscrições em si, portanto, para o experimento em questão, considerou-se não haver a noção da força das associações entre usuários e eventos. No Apache Mahout, essas associações sem valores de preferência são chamadas de preferências booleanas porque uma associação pode ter um de dois valores: ela existe ou não existe (OWEN, 2012).

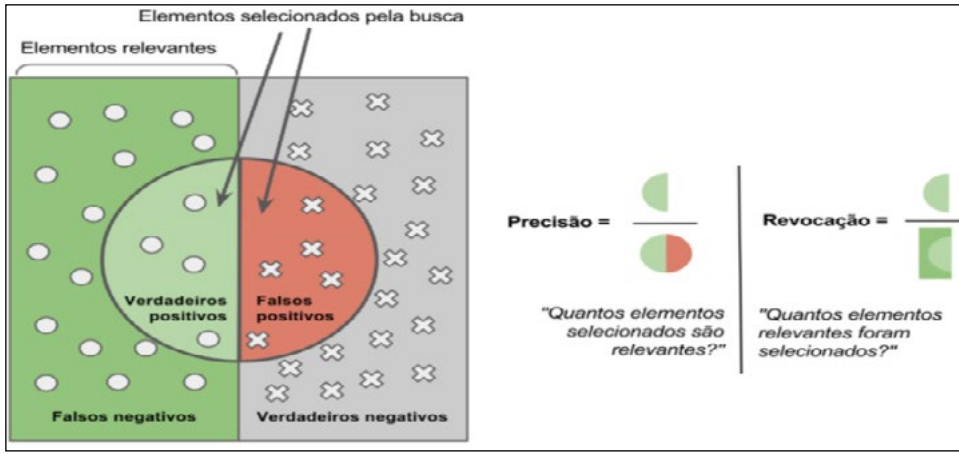
A interface UserSimilarity é onde são disponibilizadas classes que fazem os cálculos para a noção de similaridade entre dois usuários, sendo assim parte crucial de um mecanismo de recomendação (OWEN, 2012). As principais implementações da interface UserSimilarity, quando não se tem o valor da força de preferência são as classes LogLikelihoodSimilarity e TanimotoCoefficientSimilarity (OWEN, 2012).

O TanimotoCoefficientSimilarity é uma implementação com base no coeficiente de Tanimoto; também é conhecido como coeficiente de Jaccard. Ele representa o número de itens que dois usuários escolheram em comum, dividido pelo número de itens que estes mesmos usuários escolheram. Já o LogLikelihoodSimilarity faz um cálculo similar ao coeficiente de Tanimoto, porém ele avalia também se não foi uma casualidade a preferência do usuário por determinado item (OWEN, 2012).

Esta interface está ligada diretamente a classe UserNeighborhood, pois através dos usuários semelhantes que será definida a "vizinhança" para o usuário (MAHOUT, 2019).

O Apache Mahout fornece formas de avaliar o desempenho do recomendador criado. Dentre as métricas disponíveis, podemos citar a possibilidade de obter os valores *precision* e *recall*, utilizando a classe GenericRecommenderIRStatsEvaluator. *Precision* representa a relação entre o que o sistema recomendou e o que o usuário realmente teve interesse. Já o *recall* é a proporção de boas recomendações que aparecem no topo da lista de recomendações (OWEN, 2012). A Figura 6 ilustra a diferença entre *precision* e *recall*.

Figura 6 – Diferença entre precision e recall



Fonte: Wikipedia (2019, texto digital).

O teste de *precision* e *recall* foi utilizado para auxiliar a encontrar o melhor algoritmo para métrica de semelhança entre LogLikelihoodSimilarity e TanimotoCoefficientSimilarity, além de encontrar o melhor número para o tamanho da vizinhança que será utilizada, foram feitos testes com vizinhanças de dez, vinte, trinta e quarenta usuários.

Neste teste, para cada usuário o Apache Mahout remove as N principais preferências da DataModel, refazendo o cálculo de recomendações para as

novas preferências, comparando assim quanto dos novos itens recomendados estão dentro das reais preferências do usuário; cada teste durou cerca de 2 horas.

Para calcular as novas preferências de item para um usuário, precisamos considerar as preferências de usuários semelhantes. Um conjunto de usuários semelhantes ao usuário atual é chamado de vizinhança. No Apache Mahout, a noção de vizinhança é definida pela interface UserNeighborhood, que possui as implementações NearestNUserNeighborhood e ThresholdUserNeighborhood. Basicamente, na classe NearestNUserNeighborhood é definido um número de usuários mais semelhantes que serão utilizados para gerar recomendações, já no ThresholdUserNeighborhood não é definido o número usuários semelhantes, mas sim o valor mínimo de semelhança com usuário-alvo para poder pertencer a "vizinhança" (OWEN, 2012; MAHOUT, 2019).

Para a criação da vizinhança no protótipo, utilizou-se a classe NearestNUserNeighborhood, pois assim é possível obter recomendações mesmo que a vizinhança não tenha uma semelhança muito elevada. Caso fosse utilizada a ThresholdUserNeighborhood, que se baseia em um limite de semelhança, haveria o risco de não encontrar uma vizinhança adequada, com isso não gerando recomendações.

Para os testes realizados, o número N foi definido em 10, e o limite (threshold), para determinar quando a recomendação é de fato boa, foi utilizada a propriedade GenericRecommenderIRStatsEvaluator.CHOOSE_ THRESHOLD que define automaticamente esse valor.

O melhor resultado obtido foi utilizando o algoritmo de similaridade "LogLikelihoodSimilarity" com uma vizinhança de vinte usuários, que alcançou precisão de 0,4311 (43,1%) e o recall de 0,4311 (43,1%). A Tabela 2 ilustra os resultados dos testes de *precision* e *recall*, onde ambos podem variar entre 0 e 1.

Tabela 2 – Teste de *precision* e *recall*

Algoritmo	Precision	Recall	Vizinhança (10)
LogLikelihood	0,4263	0,4260	10
LogLikelihood	0,4311	0,4311	20
LogLikelihood	0,4283	0,4282	30
LogLikelihood	0,4221	0,4220	40
Tanimoto	0,4144	0,4141	10
Tanimoto	0,4263	0,4262	20
Tanimoto	0,4288	0,4285	30
Tanimoto	0,4256	0,4254	40

Fonte: Do Autor (2019).

Após identificação do algoritmo de similaridade e parametrização mais adequada ao conjunto de dados utilizado, foi criado o motor de recomendação, onde o método recommend, da interface Recommender recebe dois parâmetros, o código do usuário que se destinam as recomendações e o número máximo de recomendações a serem retornadas.

Na classe "Recomendador" foi criada a função "buscaRecomendacoes" que é responsável por fazer a chamada do sistema de recomendações, informando qual a pessoa que receberá as recomendações e a fonte de dados que será utilizada para carregar o DataModel do sistema de recomendação. Esta chamada executa o recomendador desenvolvido no Apache Mahout para gerar recomendações baseadas no arquivo que será carregado pela DataModel para um usuário, ambos passados como parâmetros para o Apache Mahout.

4.5.3 Implementação da Interface de Gestão de Fontes de Dados

A Interface de gestão de fontes de dados é a aplicação que faz o intermédio entre o sistema que exibirá as recomendações e o sistema de onde são buscadas as preferências dos usuários e os eventos que serão recomendados. Neste protótipo são cadastrados as conexões e as fontes de dados para gerar as recomendações.

O protótipo implementado apresenta uma tela de login, possibilitando que somente pessoas com permissão de acesso possam logar nele. Através da tela de fonte de dados é cadastrada a consulta SQL para criar os arquivos contendo as preferências dos usuários, o DataModel.

4.5.4 Implementação da Interface de Avaliação

Para a validação do protótipo, foi solicitada autorização para ter acesso a um *backup* do banco de dados do sistema de inscrições da Univates. O sistema de inscrições da Univates é onde são cadastrados os processos e realizada a inscrição de pessoas nesses processos, tais como inscrição de vestibular, aluguel da sede da Univates, shows, palestras e semanas acadêmicas.

Na interface de gestão de fontes de dados, foi cadastrada uma conexão para a base de dados que contêm as inscrições; também foi feito o cadastro da fonte de dados com a consulta SQL para buscar as inscrições do sistema.

A fonte de dados é a responsável por criar o arquivo para a DataModel, trazendo o código do usuário inscrito e o código do evento, filtrando somente inscrições que não foram canceladas. Além disso a consulta busca somente processos que podem ser divulgados. O arquivo gerado por esta fonte de dados ficou com 181.849 registros, contendo 42.657 usuários distintos e 2.118 eventos distintos.

Ainda, para validação do protótipo, foi desenvolvida uma tela para exibição das recomendações para o usuário, permitindo que ele visualize

e avalie recomendações, respondendo um breve questionário. Essa tela faz a chamada da Interface de gestão de fontes de dados via SOAP chamando a função "buscaRecomendacoes", da classe "Recomendador", passando os parâmetros o código da pessoa, e qual é a conexão que será utilizada, neste caso, a conexão de inscrições.

Na tela de recomendações, o usuário informa o seu código de aluno, e a partir dele são apresentadas duas questões com o objetivo de validar o quanto o usuário percebe as recomendações feitas para ele, e o quanto ele acha válido um sistema de recomendações dentro da instituição Univates. Em seguida o usuário avalia três recomendações geradas para ele, atribuindo uma nota de 0 (irrelevante) até 10 (muito relevante).

5. Validação e Resultados

Foram realizados testes com usuários reais com o objetivo de validar o SR e avaliar a satisfação dos usuários com o conteúdo recomendado. Nos testes, realizados no segundo semestre de 2019, participaram estudantes e funcionários da Univates, compreendendo a 12 pessoas, sendo 9 homens e 3 mulheres, cujo intervalo de idades foi desde os 19 até aos 32 anos (Média = 26; Desvio Padrão = 5,36).

Os testes foram realizados a partir de uma página *web* desenvolvida junto ao protótipo, na qual os participantes identificam-se com as credenciais da Univates e são direcionados a um formulário contendo algumas questões de cunho mais geral, relacionadas a sistemas de recomendações, objetivando identificar a conhecimento dos participantes em relação ao tema, e em seguida são submetidos a avaliar 3 recomendações geradas automaticamente pelo sistema.

Na primeira questão do questionário foi perguntado: "No seu dia a dia você já observou o recebimento de recomendações de sistemas, sites ou aplicativos? Se sim, quais?". Analisando os resultados é possível perceber que todos usuários já receberam recomendações vindas de um SR (100%). Dentre os SR que os usuários citaram conhecer, estavam redes sociais, sites de vendas, dentre outros.

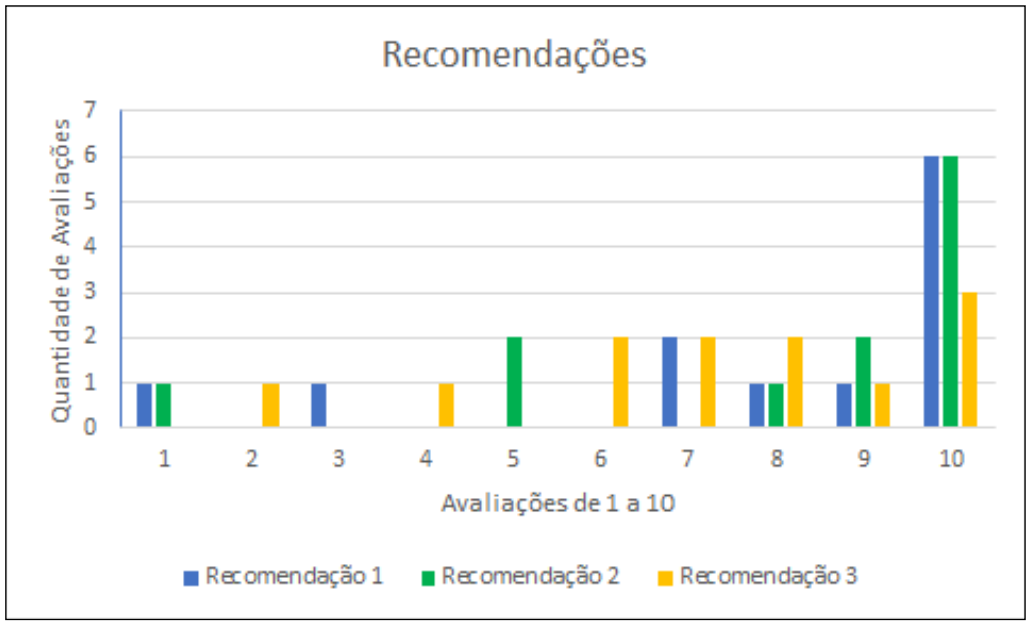
Na sequência do questionário perguntou-se: "Você considera que um sistema de recomendações para eventos de uma Universidade possa manter-lhe informado sobre potenciais interesses, de modo a contribuir com o seu desenvolvimento acadêmico/profissional?" O objetivo desta pergunta foi o de saber o grau de interesse dos usuários em um SR específico para eventos universitários. Como resultado, 91,67% dos entrevistados respondeu afirmativamente.

Três recomendações de eventos foram geradas para cada participante do teste, sendo que para cada recomendação sugerida foi permitido atribuir uma pontuação que de 1 a 10. A título de possibilitar uma validação rápida

da ferramenta, foram incluídos tanto eventos com inscrições abertas quanto eventos passados. A escolha por exibir eventos que já passaram, dá-se pelo fato de a base utilizada para os testes não ser atualizada com as novas inscrições que são feitas, assim os eventos que estão abertos para inscrição e não contém muitos inscritos podem ter um baixo valor de recomendação, pois este valor tende a crescer quanto mais pessoas estiverem inscritas no evento.

O Gráfico 1, apresenta os resultados das avaliações que os usuários fizeram para as recomendações 1, 2 e 3, que lhes foi sugerida.

Gráfico 1 – Resultados da recomendação 1



Fonte: Do Autor (2019).

Para a recomendação 1 é possível observar que 50,02% dos usuários avaliaram com a pontuação máxima, 16,66 % dos usuários avaliaram a recomendação com a pontuação 7, e apenas 8,33% atribuíram a pontuação mínima. A pontuação média para a recomendação 1 foi de 7,92 com desvio padrão de 3,02. Esse resultado demonstra que no geral as recomendações agradaram os usuários.

Para a recomendação 2 é possível observar que 50,02% dos usuários avaliaram com a pontuação máxima, 16,66 % dos usuários avaliaram a recomendação com a pontuação 9. E nenhum usuário atribuiu a pontuação mínima. A pontuação média para a recomendação 2 foi de 8,17 com desvio padrão de 2,69. Os resultados para a recomendação 2 demonstraram-se

altamente satisfatórios, levando em conta que 75% dos usuários avaliou com pontuação de 8 a 10 e que a metade avaliou com pontuação mais alta.

Para a recomendação 3 é possível observar que 25,02% dos usuários avaliaram com a pontuação máxima, 8,33% dos usuários avaliaram a recomendação com a pontuação de 9 e que para as pontuações 8, 7 e 6 houve a mesma porcentagem de avaliações (16,66%). A pontuação média para a recomendação 3 foi de 7,25 com desvio padrão de 2,49. Os resultados para a recomendação 3 foram inferiores aos encontrados para as Recomendações 1 e 2, no entanto ainda são considerados satisfatórios. Um ponto a ser destacado é que 50% dos usuários atribuíram notas entre 8 a 10 para a recomendação 3.

Obteve-se 7,78 como pontuação média considerando os resultados para as três recomendações, sendo o desvio padrão de 0,19. Levando em consideração a escala de 1 a 10 pode-se dizer que o grau de avaliação das recomendações ficou em 77,8%.

6. Conclusões

Durante o desenvolvimento deste trabalho foram estudados diversos tipos de filtragem de dados, algoritmos do Apache Mahout e abordagens relacionados à área de sistemas de recomendação. Através dessas pesquisas e do estudo de outros trabalhos foi possível concluir que a maneira mais eficiente de filtrar os dados referentes aos eventos da Universidade do Vale do Taquari - Univates a fim de gerar recomendações é utilizando filtragem colaborativa baseada no usuário.

Ao final dos testes com usuários reais ficou evidente que os usuários compreendem a necessidade de um SR para os eventos da Univates. Dessa maneira constata-se que se este protótipo fosse de fato implementado aumentaria o acesso à divulgação para os alunos, possivelmente gerando um maior número de inscritos nos eventos ofertados.

Com relação às avaliações feitas para as 3 recomendações geradas o resultado de pontuação média de 7,78, em uma escala de 1 a 10, demonstra a aceitação dos usuários ao sistema proposto. Além disso as Recomendações 1 e 2 foram avaliadas com nota máxima por metade dos usuários.

No entanto, ainda são necessárias algumas melhorias, sugerindo-se como trabalhos futuros que o Apache Mahout seja iniciado como um serviço, pré-calculando as recomendações e mantendo em memória os resultados, de modo a reduzir o tempo de resposta às chamadas feitas pela interface que está consumindo o serviço. Além disso, deve-se buscar alternativas para resolver a limitação do protótipo ao não gerar recomendações para usuários que não possuem, ou possuam histórico limitado de interação com o sistema de inscrições, incorporando ao SR a filtragem baseada em item.

Referências

AGGARWAL C. C. **Recommender systems**: The Textbook. New York: Springer International Publishing, 2016. p. 1 - 28.

AGUIAR, J.; FECHINE, J.; COSTA, E. Recomendação de Objetos de Aprendizagem utilizando Filtragem Colaborativa baseada em Tendências e em Estilos de Aprendizagem. In: **Simpósio Brasileiro de Informática na Educação - SBIE**. 28, 2018. Anais… Rio de Janeiro, 2018. 1423 p.

AMARAL, F. **Introdução à Ciência de Dados**: mineração de dados e big data. Rio de Janeiro: Alta Books Editora, 2016.

BARROS, P. A. **Sistema de filtragem colaborativa**: um estudo de caso para venda e recomendação de marmitas. 2014, 79 f. Monografia (Graduação) – Curso de Sistemas de Informação, Universidade do Planalto Catarinense, Lages, 2014.

COSTA, E.; AGUIAR, J.; MAGALHÃES, J. Sistemas de Recomendação de Recursos Educacionais: conceitos, técnicas e aplicações. In: MELO, A. M.; BORGES, M. A. F.; SILVA, C. G. (Org.). **Anais da II Jornada de Atualização em Informática na Educação**. Brasília: JAIE, 2019.

CONCEIÇÃO, F. L. A.; PÁDUA, F. L. C.; LACERDA, A.; MACHADO, A. C.; DALIP, D. H. **Multimodal data fusion framework based on autoencoders for top-N recommender systems**. Applied Intelligence, v. 49, n. 9, p. 3267-3282, 2016.

FERRARI, D. G.; SILVA, L. N. C. **Introdução a mineração de dados**. Editora Saraiva, 2016.

GIL, A. C. **Como elaborar projetos de pesquisa**. 4. ed. São Paulo: Atlas S/A, 2002.

GUY, I. Social recommender systems. In: **Recommender systems handbook**. Springer, Boston, MA, 2015. p. 511-543.

INGERSOLL, G. **Introdução ao Apache Mahout**. Disponível em: <https://www.ibm.com/developerworks/br/java/library/j-mahout/index.html#artrelatedtopics>. Acesso em: 28 de maio de 2019.

LORENA, A. C.; CARVALHO, A. C. P. L. F. **Uma introdução às support vector machines**. Revista de Informática Teórica e Aplicada, v. 14, n. 2, p. 43-67, 2007.

MARCONI, M. A.; LAKATOS, E. M. **Fundamentos de Metodologia Científica**. 3. ed. São Paulo: Atlas, 2000.

MELVILLE, P.; SHINDHWANI V. "Recommender Systems". In: SAMMUT, C.; WEBB, G. (Eds.) **Encyclopedia of Machine Learning**. Berlin: Springer, 2010, pp. 829-838.

MENDES, T. M.; SANTOS, R. V. M.; PICOLI, J. G. **Algorithm of collaborative recommendation applied to the educational context**. Brazilian Applied Science Review, 2018. p. 703-711.

OWEN, S. **Mahout in action**. Shelter Island: Manning Publications Co. 2012.

PRODANOV, C. C; FREITAS, E. C. **Metodologia do trabalho científico**: métodos e técnicas da pesquisa e do trabalho acadêmico. 2. ed. Novo Hamburgo: Feevale, 2013.

QUILICI-GONZALEZ, J. A.; ZAMPIROLLI, F. de A. **Sistemas inteligentes e mineração de dados**. Santo André: Triunfa Gráfica e Editora, 2015.

RICCI, Francesco; ROKACH, Lior; SHAPIRA, Bracha. **Recommender systems: introduction and challenges**. Boston: Springer, 2015.

ROLIM, V.; FERREIRA, R.; COSTA, E.; CAVALCANTI, A.; DIONÍSIO, M. Um Estudo Sobre Sistemas de Recomendação de Recursos Educacionais. In: Congresso Brasileiro de Informática na Educação. 6, 2017. **Anais dos Workshops do Congresso Brasileiro de Informática na Educação**. Recife, 2017. 724 p.

SCHAFER, J. B.; KONSTAN, J.; RIEDL, J. Recommender Systems. In: **Proceedings of the 1st ACM conference on Electronic commerce**, p. 158-166, 2000.