3
Sistema de recomendação proposto

3.1.
Introdução

A maior parte dos sistemas atuais armazena suas informações em bancos de dados relacionais. Apesar de apresentar inúmeros benefícios quando comparado com outras soluções, o modelo relacional não permite uma representação direta de determinadas estruturas de dados, relevantes em alguns contextos. No caso das redes sociais, os usuários e suas relações podem ser intuitivamente modelados em um grafo, como na Figura 1, onde os nós representam os usuários e os arcos as relações entre os usuários (um arco entre dois usuários indica que ambos estão conectados – são amigos). Embora seja possível armazenar essas informações utilizando algumas abstrações em um banco de dados relacional, elas não estarão dispostas de forma direta como quando armazenadas em um banco de dados em grafo.

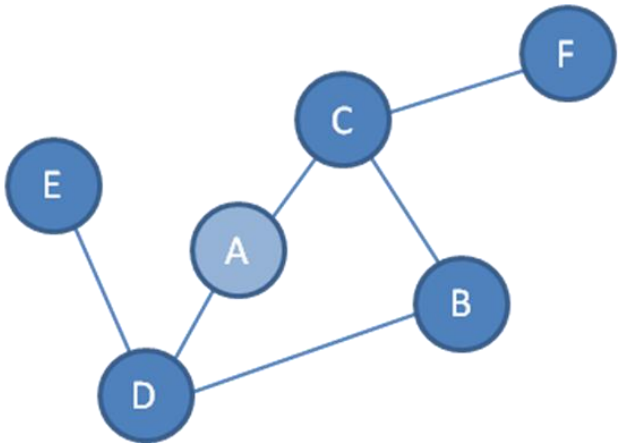

Figura 1 - Exemplo de grafo de uma rede social

A abordagem de recomendar pessoas baseando-se na rede de conexões (amizades) entre os usuários (algoritmo "friend-of-a-friend") é aplicada facilmente quando os dados estão modelados em um grafo, como o descrito anteriormente. Neste caso, basta empregar uma busca em largura partindo do nó

do usuário consumidor das recomendações: o primeiro nível visitado são os seus amigos e o segundo nível os amigos de seus amigos.

Nos bancos de dados relacionais, as informações estão dispostas na forma de linhas e colunas em tabelas e são extraídas com o uso da linguagem SQL. Uma maneira comum de armazenar os usuários e suas relações é através do uso de uma tabela para os usuários e outra para as relações. Na tabela das relações, os identificadores de dois usuários indicam que eles são amigos entre si. Assim, o acesso aos dados a fim de implementar o mesmo algoritmo "friend-of-a-friend" é complexo: é necessário listar todos os amigos do usuário, depois repetir a operação para cada amigo retornado e, por fim, juntar todas as listas de amigos retornadas.

Os bancos de dados em grafo são uma alternativa aos bancos de dados relacionais tradicionais. Eles são capazes de armazenar os dados na forma de um grafo. Com isso, além de ganhos de desempenho em determinadas tarefas, é possível acessar e manipular o grafo facilmente, algo útil dependendo da aplicação. No contexto das redes sociais, um banco de dados em grafo permite a criação de algoritmos que acessam o grafo com um código menor e menos complexo, o que contribui para sua manutenção e aprimoramento.

A transferência dos dados relacionados às amizades armazenadas em tabelas de um banco de dados relacional modeladas como explicado anteriormente para um banco de dados em grafo é direta. Cada usuário da tabela de usuários é representado como um nó no banco de dados em grafo. E, para cada relação entre dois usuários, da tabela de relações, os dois nós que os representam são ligados por um arco. Visando facilitar a implementação do algoritmo do sistema de recomendação e verificar na prática possíveis benefícios de sistemas NoSQL, o banco de dados em grafo Neo4j foi considerado neste trabalho.

Como demonstrado ao longo do texto, é possível empregar diferentes abordagens no desenvolvimento de um algoritmo de um sistema de recomendação de pessoas em uma rede social. A escolha do algoritmo depende da rede social na qual ele será aplicado. Redes sociais que possuem grandes quantidades de informações demográficas podem utilizar esses dados. Além disso, em algumas redes sociais os usuários criam e consomem muito conteúdo. Nestes casos, a natureza do conteúdo que o usuário mais se interessa pode também ser utilizada.

Apesar da diversidade das redes sociais, todas tem em comum a possibilidade dos usuários se conectarem entre si. Portanto, as informações das conexões entre os usuários podem ser sempre empregadas em um sistema de recomendação de pessoas.

É importante ressaltar que, como discutido em CHEN et al. (2009), abordagens que focam mais no conteúdo são melhores na recomendação de pessoas novas enquanto aquelas que são baseadas nas conexões entre os usuários costumam recomendar pessoas que o usuário já conhece na vida real. Ou seja, a escolha da estratégia de recomendação (e consequentemente do algoritmo utilizado) depende dos dados que estão disponíveis e também de seu objetivo (recomendar pessoas conhecidas ou não).

No texto, empregaremos os dados das conexões entre os usuários, que estão disponíveis em todas as redes sociais, para desenvolver o algoritmo do sistema de recomendação de amigos. O algoritmo será baseado no conceito de "friend-of-a-friend", ou seja, quanto maior o número de amigos (pessoas conectadas a ele) em comum um usuário A tem com um usuário B, maior a chance de A e B serem amigos. Além de poder ser aplicado em todas as redes sociais, esse conceito é a base dos algoritmos utilizados no Facebook (RATIU, 2012) e MySpace (MORICZ; DOSBAYEV; BERLYANT, 2010) e obteve bons resultados nos testes realizados na rede social Beehive (CHEN et al., 2009). Vale ressaltar que, caso disponíveis, dados demográficos e de conteúdo também podem ser utilizados em conjunto para criar um algoritmo híbrido que gere melhores recomendações.

Mas, considerando que o foco do trabalho são *sites* com poucos dados, o algoritmo desenvolvido foi baseado apenas no "friend-of-a-friend", que só utiliza as informações de conexões entre usuários, ou seja, as amizades já formadas. O algoritmo gera como saída uma lista (decrescente por uma nota) de pessoas recomendadas como potenciais amigos para um dado usuário.

Para cada pessoa da lista é atribuída uma nota correspondente ao número de amigos que aquela pessoa tem em comum com o usuário consumidor das recomendações. Dado um usuário A que é amigo de C e de D; C é amigo de B e de F; D é amigo de B e de E, como na Figura 1. Neste exemplo, o usuário B terá uma nota maior que E e F, visto que ele é amigo de C e de D e ambos (C e D) são amigos de A. Ou seja, o usuário B tem mais chances de ser amigo de A e deve aparecer antes na lista de recomendações.

As recomendações são exibidas para o usuário em sua tela, na forma de uma lista com N itens, onde cada item é uma pessoa recomendada como um potencial amigo para aquele usuário. A lista é formada pelos N primeiros usuários da lista de potenciais amigos gerada pelo algoritmo.

É fundamental exibir com clareza a recomendação e o seu motivo (SINHA; SWEARINGEN, 2002). Então, cada item da lista de pessoas recomendadas exibida para o usuário contém o nome da pessoa, sua foto, um link para seu perfil e o motivo da recomendação, que é o número de amigos em comum. Juntamente com essas informações, para cada item da lista também é exibida uma opção de adicionar aquele usuário como um novo amigo e outra de esconder aquela recomendação (caso discorde dela e não queira adicionar aquela pessoa como amiga), como é feito atualmente no *site* Facebook (vide Figura 2).

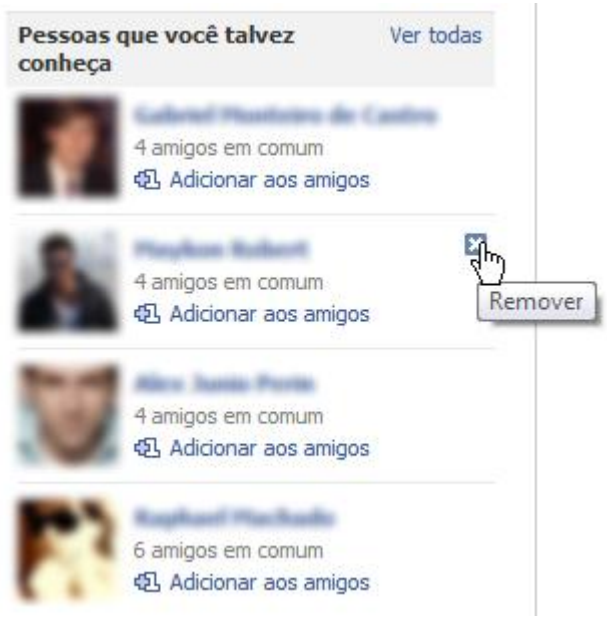

Figura 2 - Lista de recomendações de pessoas do Facebook

Se um item da lista de recomendações for escondido ou caso o usuário recomendado seja adicionado como amigo, aquele item é substituído pelo próximo item da lista gerada pelo algoritmo.

Para que o sistema de recomendação seja bem aceito por um usuário, é importante que fique claro para ele o que está sendo recomendado. No contexto de recomendação de pessoas, um nome, uma foto e um link para o perfil da pessoa recomendada são de grande valia. Porém, alguns *sites* possuem nos perfis apenas nome e sobrenome, uma quantidade de informação insuficiente. Assim, é normal

que usuários fiquem receosos em adicionar outro usuário como amigo, já que não conseguem identificar adequadamente aquela pessoa que está sendo recomendada. Este problema será abordado mais adiante no texto.

Após sua implementação, a solução foi testada com usuários reais no *site* Peladeiro. A fim de medir a qualidade das recomendações e sua importância, foram contabilizados os acertos do sistema de recomendação. Um acerto corresponde a uma adição de uma pessoa recomendada. Uma recomendação ruim é aquela que o usuário escolhe esconder por discordar dela. As recomendações que os usuários simplesmente ignoram (não adicionam a pessoa recomendada nem escondem a recomendação) também foram contabilizadas e analisadas. Todos os resultados dos testes reais, bem como dos outros testes realizados estão disponíveis na sessão correspondente do presente trabalho.

3.2. Arquitetura e implementação

O sistema de recomendação criado está dividido em uma parte no cliente e outra no servidor. A parte no cliente, desenvolvida em HTML e JavaScript utilizando a biblioteca jQuery[10], é a responsável por exibir no navegador as recomendações para o usuário do *site*. A parte no servidor foi implementada na linguagem PHP, escolhida por ser a linguagem do *site* Peladeiro, que será utilizado em nosso estudo de caso. O banco de dados utilizado é o PostgreSQL, também o mesmo do *site* Peladeiro. As duas partes, cliente e servidor, se comunicam através de requisições HTTP, conforme a Figura 3. O cliente invoca um serviço no servidor através de uma requisição HTTP GET, com parâmetros específicos dependendo do serviço, e recebe uma resposta em formato Json. Ao receber a resposta, o cliente é o responsável pela exibição para o usuário.

[10] http://www.jquery.com

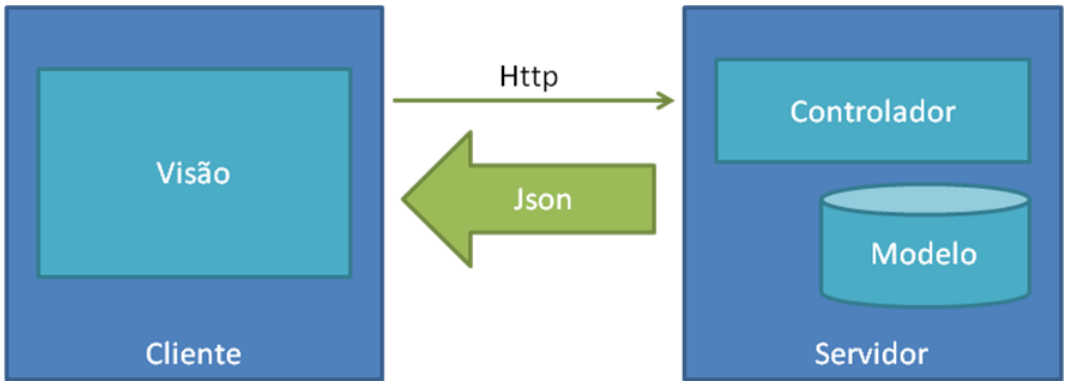

Figura 3 - Arquitetura: cliente e servidor

A parte cliente foi desenvolvida visando uma fácil integração com qualquer *site*. Para inserir o componente que exibe as recomendações para o usuário, é necessário apenas identificar uma DIV no código HTML da página, como na Figura 4.

```
<html>
  <head>
    <script type="text/javascript" src="jquery.min.js"></script>
    <script type="text/javascript" src="recomendacao.js"></script>
    <link rel="stylesheet" type="text/css" href="rec_estilo.css" />
  </head>
  <body>
    <div id="rec-titulo">Pessoas que talvez conheça:</div>
    <div class="recomendacoes"></div>
  </body>
</html>
```

Figura 4 - Inserção do sistema de recomendação na página HTML

Em seguida, basta executar o método JavaScript de geração de recomendações. Tal método recebe como parâmetros a quantidade de recomendações que deverá ser criada, quantas delas deverão ser exibidas simultaneamente e o identificador do usuário consumidor das recomendações. O método invoca o serviço de geração de recomendações no servidor e recebe como resposta recomendações em formato Json como na Figura 5.

```
[{"nome": "Homer Simpson", "foto": "img/homer.jpg", "amigos": 5, "id": 21},
 {"nome": "Lisa Simpson", "foto": "img/lisa.jpg", "amigos": 4, "id": 23},
 {"nome": "Bart Simpson", "foto": "img/bart.jpg", "amigos": 3, "id": 31}]
```

Figura 5 - Exemplo de Json retornado pelo servidor

O cliente lê o Json recebido pelo servidor e exibe uma lista (vide Figura 6).

Figura 6 - Lista de recomendados exibida para o usuário

Cada item da lista é formado pelo nome, foto e quantidade de amigos em comum com o usuário consumidor. Além disso, para cada item há uma opção para fechar aquela recomendação e outra para adicionar o usuário recomendado como amigo. Caso o usuário deseje rejeitar aquela recomendação, pode fechá-la, o que faz com que o item seja substituído por outra recomendação. Caso o usuário deseje aceitar a recomendação, pode adicionar o usuário recomendado selecionando a opção de adicionar amigo. O item correspondente a um usuário adicionado deixa de ser exibido e é substituído por outra recomendação.

O usuário interage com o sistema segundo o diagrama de casos de uso exibido na Figura 7. Todas as interações do usuário com a lista de recomendações são monitoradas e armazenadas no servidor em uma tabela específica no banco de dados (vide Figura 8). Para isso, no momento em que uma recomendação é exibida, ou quando o usuário rejeita ou aceita uma recomendação, serviços são chamados no servidor. Eles são responsáveis por armazenar essas informações na tabela no banco de dados. Nela estão presentes as datas de criação, atualização e exibição de cada recomendação. Estão presentes também a quantidade de exibições, de aceitações e de rejeições.

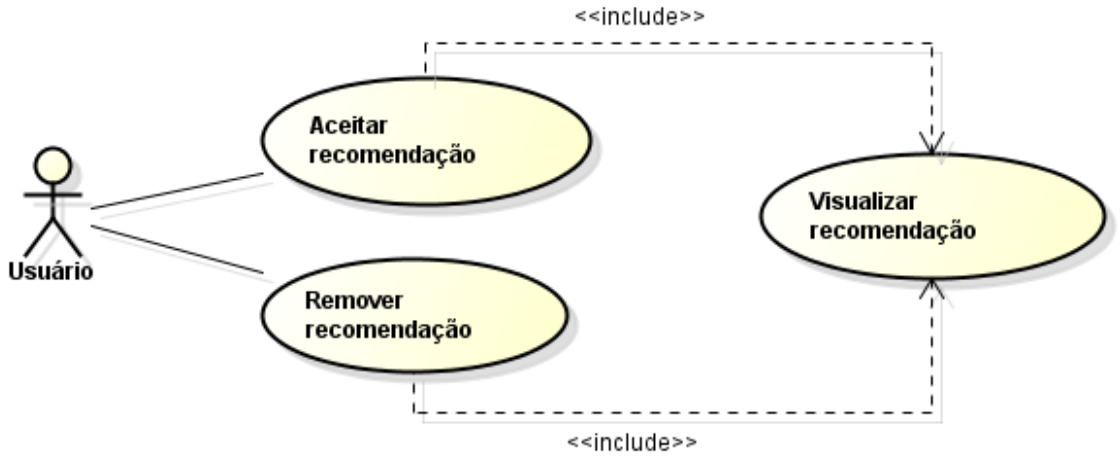

Figura 7 - Casos de uso do sistema de recomendação

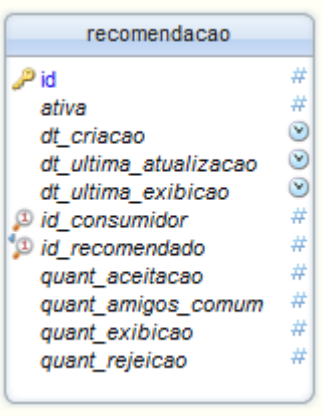

Figura 8 - Tabela utilizada pelo sistema de recomendação

O componente faz chamadas ao servidor de forma assíncrona, de modo a manter o restante da página carregando e funcionando de forma independente. Além disso, a cada chamada é exibido para o usuário alguma forma de feedback para deixar claro que algo está sendo carregado. Por exemplo, enquanto o serviço de geração de recomendações ainda não retornou é exibida uma barra de progresso (vide Figura 9).

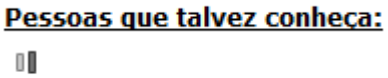

Figura 9 - Feedback exibido para usuário

Mais detalhadamente, para gerar, exibir e interagir com as recomendações são necessários os seguintes passos:

1) Cliente executa a função em JavaScript *gerarRecomendacoes* que faz uma requisição assíncrona para o servidor (*gerar.php*). Enquanto o servidor não retornar, é exibida na tela uma barra de progresso.
2) O servidor executa uma consulta no banco de dados cuja saída é uma lista de recomendações.
3) A tabela *recomendação* no banco de dados é atualizada: caso uma recomendação retornada pela consulta executada no segundo passo já esteja nela, seu registro é atualizado e, caso contrário, um novo registro é adicionado.
4) O servidor retorna um arquivo em formato Json para o cliente contendo as recomendações ativas da tabela *recomendação*.
5) O cliente lê o arquivo e exibe as recomendações na DIV identificada para este fim na página HTML, no lugar da barra de progresso.
6) Caso o usuário adicione um usuário que foi recomendado para ele, selecionando a opção de adicionar amigo, ou feche uma recomendação, o cliente chama o servidor (*adicionar.php* ou *remover.php*, respectivamente) e a tabela *recomendação* é atualizada.
7) Quando o servidor retorna, a lista de recomendações exibida é atualizada, ou seja, o cliente para de exibir uma recomendação que foi aceita ou rejeitada.

Como visto anteriormente, o sistema foi inteiramente concebido visando a fácil integração em diferentes *sites*. Além de uma clara divisão entre as partes cliente e servidor, o algoritmo de recomendação também está bem separado. Por exemplo, caso seja aplicado em algum *site* onde estejam disponíveis mais dados, o algoritmo de recomendação pode ser facilmente alterado (em *gerar.php*) a fim de utilizá-los sem que nenhuma outra modificação adicional no sistema seja necessária.

Outra possibilidade é executar o algoritmo de recomendação de forma periódica em segundo plano, ou seja, não executá-lo no momento em que o cliente necessita exibir as recomendações. Apesar das recomendações passarem a não ser completamente atualizadas, tal abordagem é útil para *sites* cuja quantidade de usuários é muito grande, onde a geração de recomendações em tempo real é inviável. Para tanto, basta retirar do arquivo *gerar.php* a responsabilidade de atualizar a tabela *recomendação* e chamar o algoritmo de recomendação em um

serviço executado periodicamente de forma independente (utilizando ferramentas como o Quartz[11], por exemplo). Este serviço passaria a ser o responsável por gerar as recomendações e atualizar a tabela *recomendação*. O restante do sistema permaneceria inalterado.

O detalhamento dos casos de uso do sistema bem como a descrição de todos os arquivos que o compõe estão no apêndice ao final do trabalho.

3.3. Limitações

O sistema desenvolvido conta com certas limitações. Algumas delas podem ser mitigadas através da adoção de medidas propostas como trabalhos futuros, descritas na conclusão do presente trabalho.

3.3.1. Recomendações para usuários sem conexões

Ao se cadastrar em um *site* de rede social, um usuário possuirá provavelmente nenhum usuário em sua lista de amigos daquele *site*. Como o algoritmo do sistema desenvolvido utiliza como entrada apenas as informações das conexões do usuário com os outros usuários do *site*, o sistema não é capaz de gerar recomendações para os novos usuários.

3.3.2. Performance e escalabilidade

Os resultados dos testes do sistema desenvolvido, explicitados na próxima sessão, mostram que sua performance é adequada para ser utilizado em *sites* de redes sociais com cerca de 500 mil usuários. Porém, o aumento do número de usuários bem como da quantidade de conexões entre eles geram um acréscimo significativo no tempo de execução do algoritmo.

[11] http://quartz-scheduler.org

3.3.3. Identificação da recomendação

Como já foi discutido ao longo do texto, é importante que o usuário consumidor da recomendação identifique bem o usuário que está sendo recomendado. Isso nem sempre é possível em um cenário como este, onde os perfis dos usuários não contam com uma boa quantidade de dados confiáveis para ser utilizada.