7
Referências bibliográficas

ADOMAVICIUS, G.; TUZHILIN, A. **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, v. 17, p. 734-749, jun. 2005.

BOYD, D. M.; ELLISON, N. B. **Social network sites: Definition, History, and Scholarship.** Journal of Computer-Mediated Communication, v. 13, n. 1, p. 210-230, out. 2007.

CHEN, J.; GEYER, W.; DUGAN, C.; MULLER, M.; GUY, I. **Make new friends, but keep the old: recommending people on social networking sites.** Proceedings of the 27th international conference on Human factors in computing systems, Boston, abr. 2009.

DAVIDSON, J.; LIEBALD, B.; LIU, J.; NANDY, P.; VAN VLEET, T. **The YouTube video recommendation system.** Proceedings of the 4th ACM conference on recommender systems, p. 293-296, 2010.

ESSLIMANI, I.; BRUN, A.; BOYER, A. **From social networks to behavioral networks in recommender systems.** Proceedings of The 2009 International Conference on Advances in Social Networks Analysis and Mining (ASONAM), IEEE Computer society, p. 143-148, 2009.

FACEBOOK. **API Reference: Graph API.** Disponível em: <http://developers.facebook.com/docs/reference/api/>. Acesso em: 12 set. 2012.

_____. **Newsroom: Key Facts.** Disponível em: <http://newsroom.fb.com/content/default.aspx?NewsAreaId=22>. Acesso em: 7 jun. 2012.

GEYER, W.; DUNGAN, C.; MILLEN, D. R.; MULLER, M.; FREYNE, J. **Recommending topics for self-descriptions in online user profiles.** Proceedings of the 2008 ACM conference on Recommender systems, Lausanne, out. 2008.

KARKADA, U. H. **Friend recommender system for social network.** Disponível em: <http://www.umanka.com/files/recommender.pdf>. Acesso em: 3 jan. 2012.

LEAVITT, N. **Will NoSQL Databases Live Up to Their Promise?**, Computer, v. 43 n. 2, p. 12-14, fev. 2010.

LINDEN, G.; SMITH B.; YORK J. **Amazon.com Recommendations: Item-to-Item Collaborative Filtering.** IEEE Internet Computing, v. 7, n. 1, p. 76-80, jan. 2003.

LOPES, G. R. **Avaliação e Recomendação de Colaborações em Redes Sociais Acadêmicas.** Tese (doutorado) - Universidade Federal do Rio Grande do Sul, Porto Alegre, mai. 2012.

MORICZ, M.; DOSBAYEV, Y.; BERLYANT, M. **PYMK: Friend Recommendation at MySpace.** Proceedings of the 2010 international conference on Management of data, p. 999-1002, Indianapolis, 2010.

NAWROTH, A. **Social networks in the database: using a graph database.** Disponível em: <http://blog.neo4j.org/2009/09/social-networks-in-database-using-graph.html>. Acesso em: 5 jan. 2012.

NEO4J. **The Neo4j Manual.** Disponível em: <http://docs.neo4j.org/pdf/neo4j-manual-stable.pdf>. Acesso em: 10 jan. 2012.

PELADEIRO. **Institucional.** Disponível em: <http://www.peladeiro.com.br/institucional.php>. Acesso em: 11 out. 2012.

QUERCIA, D.; CAPRA, L. **FriendSensing: Recommending Friends Using Mobile Phones.** Proceedings of the third ACM conference on Recommender systems, Nova Iorque, USA, out. 2009.

RATIU, F. **The Facebook Blog: People You May Know.** Disponível em: <http://blog.facebook.com/blog.php?post=15610312130>. Acesso em: 4 jan. 2012.

RESNICK, P.; VARIAN, H. R. **Recommender Systems.** Communications of the ACM, v. 40, n.3, p. 56-58, mar. 1997.

RICCI, F.; ROKACH, L.; SHAPIRA, B. **Introduction to Recommender Systems Handbook.** Springer, 2010.

SINHA, R.; SWEARINGEN, K. **The Role of Transparency in Recommender Systems.** CHI '02 extended abstracts on Human factors in computing systems, Minneapolis, abr. 2002.

SPERTUS, E.; SAHAMI, M.; BUYUKKOKTEN, O. **Evaluating similarity measures: a large-scale study in the orkut social network.** Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining, Chicago, agos. 2005.

TERVEEN, L.; MCDONALD, D. W. **Social matching: A framework and research agenda.** ACM Transactions on Computer-Human Interaction (TOCHI), v. 12, n. 3, p. 401-434, set. 2005.

VICKNAIR, C.; MACIAS, M.; ZHAO, Z.; NAN, X.; CHEN, Y.; WILKINS, D. **A comparison of a Graph Database and a Relational Database.** Proceedings of the 48th annual Southeast regional conference, ACMSE-SE 48, 2010.

XIE J.; LI X. **Make Best Use of Social Networks via More Valuable Friend Recommendations.** 2nd International Conference on Consumer Electronics, Communications and Networks (CECNet), p. 1112 -1115, 2012.

8
Apêndices

8.1.Descrição dos casos de uso do sistema de recomendação

Caso de uso: Visualizar recomendação

- Descrição sucinta

O Sistema exibe para o Usuário uma lista de recomendações.

- Atores

1. Usuário

- Fluxo básico

1. O Sistema exibe para o Usuário uma lista de recomendações de usuários candidatos a novos amigos onde cada item segue ED1.
2. O Sistema incrementa o contador de exibição para cada recomendação exibida na lista de recomendações.

- Estruturas de dados

1. Informações de uma recomendação (ED1)
    a. nome e sobrenome do usuário recomendado
    b. foto do usuário recomendado
    c. número de amigos em comum entre o usuário recomendado e o Usuário
    d. opcação de adicionar o usuário recomendado como amigo
    e. opção de fechar a recomendação

Caso de uso: Aceitar recomendação

- Descrição sucinta

O Usuário aceita um usuário recomendado.

- Atores

1. Usuário

- Pré-condições

1. Ter executado o caso de uso “Visualizar recomendação”.

- Fluxo básico

1. O Usuário seleciona o link “Adicionar aos amigos” de um item.
2. O Sistema incrementa o contador de aceitação para a recomendação correspondente a aquele item.
3. O Sistema adiciona o usuário recomendado ao conjunto de amigos do Usuário.
4. O Sistema deixa de exibir aquela recomendação e, caso exista outra recomendação para exibir, exibe a outra recomendação.

Caso de uso: Remover recomendação

- Descrição sucinta

O Usuário rejeita um usuário recomendado.

- Atores

1. Usuário

- Pré-condições

1. Ter executado o caso de uso “Visualizar recomendação”.

- Fluxo básico

1. O Usuário seleciona o link de remoção de um item.
2. O Sistema incrementa o contador de rejeição para a recomendação correspondente a aquele item.
3. O Sistema deixa de exibir aquela recomendação e, caso exista outra recomendação para exibir, exibe a outra recomendação.

8.2.Descrição dos arquivos do sistema de recomendação

Arquivo	Descrição
index.htm	Página HTML com o componente de recomendação.
rec_estilo.css	CSS utilizado no *index.htm* para formatar a apresentação do componente de recomendação.
recomendacao.js	Código JavaScript responsável por fazer chamadas no servidor e controlar o componente de recomendação no cliente. Referenciado no *index.htm*. A biblioteca jQuery é necessária.
img/carregando.gif	Imagem exibida para o usuário enquanto as recomendações estão sendo geradas no servidor.
img/fechar.gif	Imagem para botão de fechar de cada recomendação.
img/padrao.gif	Imagem exibida no lugar onde deveria estar a imagem do usuário recomendado, caso ela não tenha sido carregada com sucesso.
gerar.php	Gera recomendações para um dado usuário e as retorna em formato Json. Recebe como parâmetros a quantidade máxima de recomendações e o identificador do usuário consumidor.
exibir.php	Incrementa o contador de vezes que uma recomendação foi exibida e atualiza a data de sua última exibição. Chamado sempre que alguma recomendação é exibida para o usuário. Recebe como parâmetro o identificador da recomendação.
adicionar.php	Incrementa o contador de vezes que uma recomendação foi adicionada e adiciona o usuário recomendado como amigo. Chamado sempre que o usuário recomendado é adicionado como amigo pelo usuário. Recebe como parâmetro o identificador da recomendação.
remover.php	Incrementa o contador de vezes que uma recomendação foi removida. Chamado sempre que alguma recomendação é fechada pelo usuário. Recebe como parâmetro o identificador da recomendação.
conexao.php	Estabelece uma conexão com o banco de dados. Utilizado pelos ourtros arquivos que necessitam uma conexão com o banco de dados.
consulta.php	Consulta SQL utilizada para gerar a recomendação. Utilizada pelo gerar.php. Desde que retorne as mesmas colunas, pode ser modificada livremente.

8.3.Consulta SQL da recomendação

```
SELECT
   recomendados.idAmigoDeAmigo AS id_recomendado,
   count(DISTINCT recomendados.idAmigo) AS quant amigos comum
FROM
   ustb usuario usuario,
   (  SELECT
      CASE
         WHEN id_perfil = amigos.id_usuario THEN amigo.id_usuario
         ELSE CASE
            WHEN amigo.id usuario = amigos.id usuario THEN amigo.id perfil
         END
      END AS idAmigoDeAmigo,
      amigos.id_usuario AS idAmigo,
      amigos.ds nome,
      amigos.ds sobrenome
   FROM
      ustb_amigo amigo,
      (  SELECT
         usuario.id_usuario,
         usuario.ds nome,
         usuario.ds sobrenome
      FROM
         (  SELECT
            CASE
               WHEN id_perfil = #id# THEN id_usuario
               ELSE CASE
                  WHEN id usuario = #id# THEN id perfil
               END
            END AS id
         FROM
            ustb_amigo
         WHERE
            id_perfil <> id_usuario
            AND STATUS = 1
            AND (
               id perfil = #id#
               OR id usuario = #id#
            )  ) AS temp,
         ustb_usuario usuario
      WHERE
         usuario.id usuario = temp.id
         AND usuario.id status = 'A'
      ) AS amigos
   WHERE
      amigo.id_perfil <> amigo.id_usuario
      AND amigo.STATUS = 1
      AND (
         amigo.id perfil = amigos.id usuario
         OR amigo.id usuario = amigos.id usuario
      )  ) AS recomendados
WHERE
   usuario.id_usuario <> #id#
   AND usuario.id status = 'A'
   AND recomendados.idAmigoDeAmigo = usuario.id usuario
   AND NOT EXISTS (
      SELECT
         usuario.id_usuario,
         usuario.ds nome,
         usuario.ds sobrenome
      FROM
         (  SELECT
            CASE
               WHEN id_perfil = #id# THEN id_usuario
               ELSE CASE
                  WHEN id usuario = #id# THEN id perfil
               END
            END AS id
         FROM
            ustb_amigo
         WHERE
            id perfil <> id usuario
            AND STATUS = 1
            AND (
```

```
            id_perfil = #id#
            OR id_usuario = #id#
         )  ) AS temp,
      ustb_usuario usuario
   WHERE
      usuario.id_usuario = temp.id
      AND usuario.id_status = 'A'
      AND usuario.id_usuario = recomendados.idAmigoDeAmigo
   )
GROUP BY
   recomendados.idAmigoDeAmigo,
   usuario.ds_nome,
   usuario.ds_sobrenome
ORDER BY
   quant_amigos_comum DESC
```

8.4.Implementação Neo4j

```
/**
 * Gera uma lista com recomendacoes de usuarios considerados potenciais amigos
 * para um dado usuario. A lista e formada por usuarios que tem muitos amigos
 * em comum com um dado usuario. O algoritmo utiliza a facilidade de acesso ao
 * grafo proporcionada pelo banco de dados Neo4j para acessar diretamente
 * aqueles usuários cuja distancia para o dado usuario e 2. O algoritmo no
 * entanto nao preenche a lista de amigos em comum de cada usuário
 * recomendado.
 *
 * @param id
 *            Id do usuario para o qual se deseja obter recomendacoes de
 *            amigos.
 * @return Lista de recomendacao.
 */
public List<Recomendacao> recomendarNeo4j3(int id) {
    StopEvaluator twoSteps = new StopEvaluator() {
        @Override
        public boolean isStopNode(TraversalPosition position) {
            return position.depth() == 2;
        }
    };
    ReturnableEvaluator nodesAtDepthTwo = new ReturnableEvaluator() {
        @Override
        public boolean isReturnableNode(TraversalPosition position) {
            return position.depth() == 2;
        }
    };

    BancoNeo4j peladeiroNeo4j = new BancoNeo4j(caminhoNeo4j);
    GraphDatabaseService bd = peladeiroNeo4j.getGraphDatabase(false);
    Index<Node> nodeIndex = bd.index().forNodes(BancoNeo4j.INDICE_USUARIOS);
    Node usuario = nodeIndex.get(BancoNeo4j.PROPRIEDADE ID, id).getSingle();

    Traverser traverser = usuario.traverse(Order.BREADTH_FIRST, twoSteps,
            nodesAtDepthTwo, BancoNeo4j.RelTypes.AMIZADE, Direction.BOTH);

    ArrayList<Recomendacao> recomendacoes = new ArrayList<Recomendacao>();
    for (Node amigo : traverser) {
        Usuario usuarioRecomendado = new Usuario();
        usuarioRecomendado.setId(new Integer("" +
                amigo.getProperty(BancoNeo4j.PROPRIEDADE ID)));
        usuarioRecomendado.setNome("" +
                amigo.getProperty(BancoNeo4j.PROPRIEDADE NOME));
        usuarioRecomendado.setSobrenome("" +
                amigo.getProperty(BancoNeo4j.PROPRIEDADE_SOBRENOME));
        Recomendacao recomendacao = new Recomendacao();
        recomendacao.setUsuario(usuarioRecomendado);
        recomendacoes.add(recomendacao);
    }
    bd.shutdown();
    return recomendacoes;
}
```