ULBRA CENTRO UNIVERSITÁRIO LUTERANO DE PALMAS
COMUNIDADE EVANGÉLICA LUTERANA "SÃO PAULO"
Recredenciado pela Portaria Ministerial nº 3.607 - D.O.U. nº 202 de 20/10/2005

Charles Alex Rockenbach

Recomendação de especialistas na plataforma de gestão de conhecimento konnen

Palmas
2012

ULBRA CENTRO UNIVERSITÁRIO LUTERANO DE PALMAS
COMUNIDADE EVANGÉLICA LUTERANA "SÃO PAULO"
Recredenciado pela Portaria Ministerial nº 3.607 - D.O.U. nº 202 de 20/10/2005

Charles Alex Rockenbach

Recomendação de especialistas na plataforma de gestão de conhecimento konnen

Trabalho apresentado como requisito parcial da disciplina Trabalho de Conclusão de Curso (TCC) do curso de Sistemas de Informação, orientado pelo Professor Mestre Edeilson Milhomem da Silva.

Palmas
2012

ULBRA CENTRO UNIVERSITÁRIO LUTERANO DE PALMAS
COMUNIDADE EVANGÉLICA LUTERANA "SÃO PAULO"
Recredenciado pela Portaria Ministerial nº 3.607 - D.O.U. nº 202 de 20/10/2005

Charles Alex Rockenbach

Recomendação de especialistas na plataforma de gestão de conhecimento konnen

Trabalho apresentado como requisito parcial da disciplina Trabalho de Conclusão de Curso (TCC) do curso de Sistemas de Informação, orientado pelo Professor Mestre Edeilson Milhomem da Silva

Aprovada em _____ de 2012.

BANCA EXAMINADORA

_____

Prof. M.Sc. Edeilson Milhomem Silva

Centro Universitário Luterano de Palmas

_____

Prof. M.Sc. Jackson Gomes de Souza

Centro Universitário Luterano de Palmas

_____

Prof. M.Sc. Parcilene Fernandes de Brito

Centro Universitário Luterano de Palmas

Palmas
2012

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

1. INTRODUÇÃO

A informação é fundamental para execução de uma tarefa e representa o diferencial para os indivíduos ou organizações que almejam se destacar e manter-se competitivos no mercado. Com a informação correta no momento oportuno, é possível realizar tarefas simples, como chegar ao destino de uma viagem, ou complexas, como desenvolver um *software* de edição de imagens. Ao longo da história, muitas técnicas foram utilizadas para armazenamento e recuperação das informações, sendo uma delas, a representação através de um meio físico, como um livro.

A Tecnologia da Informação oferece métodos para aperfeiçoar o processo de manipulação da informação através de inúmeras tecnologias, como a computação em nuvem. Essa informação armazenada pode ser consumida por uma pessoa, e, assim, ser transformada em conhecimento. O processo inverso, que transforma o conhecimento em informação para ser armazenada, e assim, disponibilizada como fonte de consulta, não pode ser realizado corretamente, pois há uma dificuldade de formalizá-la, sistematizá-la e quantificá-la. Esse tipo de conhecimento é denominado conhecimento tácito, que é a experiência individual obtida através do processamento de informações durante a realização de tarefas ou observações do ambiente no dia-a-dia.

Sendo assim, para solucionar este problema de representação de conhecimento tácito, surgiram os sistemas de recomendação de especialistas, que, com base em informações explícitas sobre os usuários, inferem quais são suas especialidades e qual o seu nível de domínio sobre elas. Além da definição das especialidades, os sistemas de recomendação de especialistas dispõem de um meio para se encontrar usuários que são especialistas em determinadas áreas, para que eles possam ser contatados, e, dessa forma, compartilhar o seu conhecimento tácito.

Alguns dos sistemas de recomendação de especialistas funcionam com base em conceitos de redes sociais, como o SmallBlue (LIN *et. al.*, 2008) e ArnetMiner (TANG *et. al.*, 2008). Este cenário das redes sociais é utilizado por prover, entre outras, informações sobre os relacionamentos entre os usuários. Através das informações dos relacionamentos de um

usuário, pode-se, por exemplo, definir um caminho de amigos que o conecta a um especialista de seu interesse, facilitando assim o contato inicial entre ambos.

Considerando esse cenário, esse trabalho tem o objetivo apresentar o formalismo de um mecanismo para a inferência de especialidades dos usuários, baseando-se no seu conhecimento (produções científicas e conhecimentos produzidos pelos usuários no ambiente em que o referido modelo foi implantado). Além disso, foi realizada a implantação deste mecanismo na plataforma educacional baseada na teoria de redes sociais, intitulada konnen.

O texto está organizado da seguinte maneira: na seção 2, são apresentados os conceitos de Redes Sociais e Sistema de Recomendação de especialistas e alguns exemplos de trabalhos relacionados, envolvidos no trabalho; posteriormente, na seção 3, são apresentados os materiais e métodos; logo em seguida, na seção 4, são apresentados os formalismos para as inferências das habilidades dos usuários e a implementação realizada no ambiente da konnen; a seção 5 apresenta as considerações finais, e, por último, na seção 6, são apresentadas as referências bibliográficas.

2. REFERENCIAL TEÓRICO

Esta seção apresenta conceitos referentes a redes sociais, como o processo de análise das informações de seus usuários, e sistemas de recomendação de especialistas, abordando as etapas de identificação e recomendação de especialistas. Este referencial é o que fornece a base para a modelagem e desenvolvimento do aplicativo proposto.

2.1 Redes Sociais

Pimentel e Fulks (2011, p. 54) definem que as redes sociais, estruturas básicas de uma sociedade, são formadas pelas pessoas e seus relacionamentos. Esses relacionamentos podem ser diretos ou indiretos. Relacionamentos diretos são os relacionamentos mantidos com parentes, amigos, colegas de faculdade ou trabalho, ou seja, pessoas com quem realmente se possui algum vínculo. Os relacionamentos indiretos são relacionamentos obtidos através dos relacionamentos que têm como intermediários contatos diretos. Por exemplo, João possui um relacionamento direto com Ana e Ana possui um relacionamento direto com Claudia, sendo assim, como João não está relacionado diretamente com Claudia, ele possui um relacionamento indireto com ela, já que existe uma pessoa que intermedia essa relação. Além disso, a rede social pessoal de cada indivíduo é uma estrutura mutável em relação ao tempo, ou seja, podem-se agregar novos elementos, como novos contatos profissionais ou vizinhos, ou ainda, remover outros, como antigos conhecidos.

No meio digital, o conceito de redes sociais também está presente, recebendo o nome de redes sociais virtuais ou redes sociais na web. Para Pimentel e Fulks (2011, p. 54), “redes sociais na web são ambientes virtuais onde os participantes interagem com outras pessoas e criam redes baseadas em algum tipo de relacionamento”. As redes sociais virtuais, assim como as redes sociais, baseiam-se em elementos e seus relacionamentos, que também podem ser diretos ou indiretos.

A principal diferença entre as redes sociais virtuais (RSV) e as redes sociais (RS) é o fato de que nas RSV o conhecimento e as informações elaboradas nesse ambiente ficam armazenadas em meio digital, podendo ser recuperadas facilmente, enquanto que nas RS o conhecimento fica armazenado em sua maioria em meio físico, o que dificulta o seu acesso. As informações das RSV, por estarem no formato digital, podem ser transmitidas por meio da internet, sendo acessíveis em larga escala. Já as informações das RS são registradas em meios físicos, como livros, porém, para que sejam acessíveis em larga escala, deve-se realizar a

reprodução e envio do material para os interessados, o que resulta em um maior esforço, se comparada com a difusão da informação em meio digital.

Golbeck (2005, p. 13-14) aponta que uma RSV deve seguir as seguintes definições:

- o ambiente deve ser acessível através de um navegador web, ou seja, não é preciso instalar *softwares* específicos para ter acesso a RSV;
- os usuários devem estabelecer explicitamente seus relacionamentos, logo, todas as relações devem ser criadas pelos próprios usuários a partir da utilização da RSV;
- o sistema deve prover mecanismos nativos para a construção dos relacionamentos, ou seja, deve haver uma funcionalidade padrão no sistema para formalizar um relacionamento entre os usuários;
- os relacionamentos devem ser visíveis para as partes envolvidas no relacionamento e negáveis, logo, um usuário deve poder visualizar quais são suas conexões e rompê-las, se assim desejar.

Um exemplo de RSV é apresentado na Figura 1.

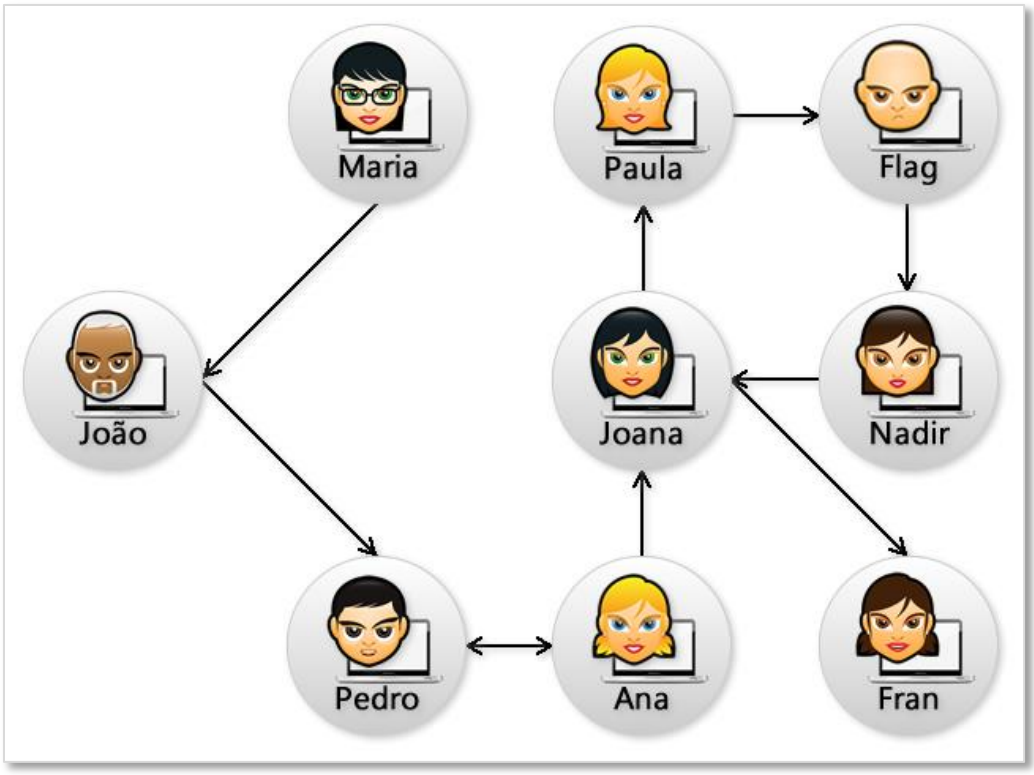


Figura 1: Exemplo de rede social virtual

A Figura 1 apresenta um exemplo de uma rede social que contém algumas pessoas e seus relacionamentos. Ao analisar a Figura 1, pode-se perceber os relacionamentos diretos entre os usuários, que estão nas extremidades das arestas, como o relacionamento entre Pedro

e Ana, e relacionamentos indiretos, como o relacionamento entre Ana e Paula através de Joana.

De acordo com Pimentel e Fulks (2011, p. 59), a conexão entre os usuários de uma RS, representada por um grafo, pode ser direcional ou não direcional. Na Figura 1 pode-se observar que as arestas que ligam as pessoas possuem uma direção, indicando quem possui relação com quem. Por exemplo, Maria se relaciona com João, porém, João não se relaciona com Maria. Outro exemplo, ainda na Figura 1, é o relacionamento bidirecional entre Pedro e Ana, o que significa que Pedro possui um relacionamento com Ana e Ana possui um relacionamento com Pedro. As possibilidades de relacionamentos em um grafo são apresentadas pela Figura 2.

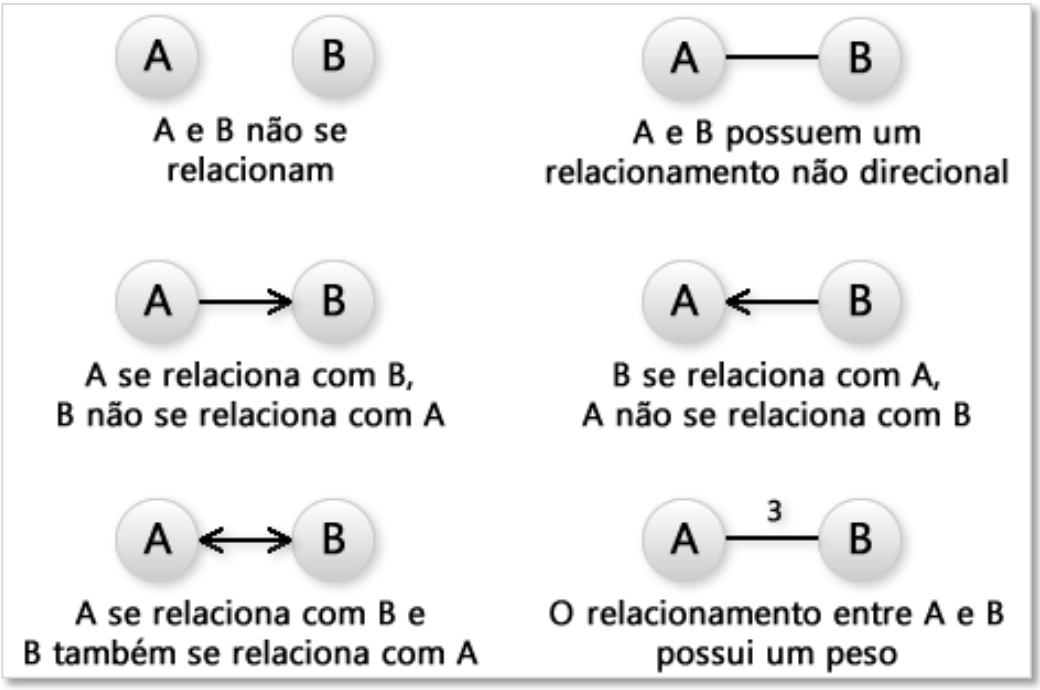


Figura 2: Relacionamentos entre nós de uma rede social (MEIRA *et. al.*, 2011, p. 59, adaptada)

Como mostrado na Figura 2, existem cinco tipos de relacionamento entre elementos de uma RS: a) os nós não se relacionam, são independentes; b) os nós possuem um relacionamento não direcional, onde A e B se relacionam de alguma forma, só que não há uma definição clara dos relacionamentos que um possui em relação ao outro; c) os nós possuem um relacionamento com sentido unilateral, uma extremidade A se relaciona com a outra extremidade B, mas o contrário não é verdade; d) os nós possuem um relacionamento bidirecional, onde os dois se relacionam; e) o relacionamento entre A e B possui um peso que indica o quanto esse relacionamento é importante. A associação de um peso ao

relacionamento pode acontecer em todos os casos apresentados, exceto o no que não há relacionamento entre os nós.

Hoje em dia existem várias RSV, com foco, principalmente, em entretenimento, como o Facebook, Badoo e Orkut, compartilhamento, como o Twitter e YouTube, e profissional, como o LinkedIn. A maioria das RSV existentes, entre elas o Facebook e LinkedIn, são sistemas de criação e aquisição de experiências, pois as experiências e informações geradas por cada pessoa são mais ricas (não baseiam-se quase que unicamente em texto, como no ICQ e MSN, base das primeiras RSV que surgiram) e podem ficar registradas. Como essas informações são armazenadas, elas podem ser exploradas de diversas maneiras, entre elas, no auxílio à resolução de problemas identificados no cotidiano, como, por exemplo, descobrir qual o ponto ou os pontos de deficiência de conhecimento da organização, e, a partir disso, ministrar treinamentos nessa área para nivelar o conhecimento dos demais membros da organização.

Dentro da área das RSV, surge o conceito de análise de redes sociais, que tem por objetivo identificar informações relevantes em meio aos relacionamentos e outros dados armazenados na estrutura da rede. A subseção 2.1.1 apresentará esse tema e algumas das suas possíveis utilidades.

2.1.1 Análise de redes sociais

As redes sociais se relacionam com uma área de pesquisa em sociologia que lida com a maneira como os elementos da rede se conectam uns com os outros e como a informação flui entre eles (SHAPIRA e ZABAR, 2011, p. 147). Através da análise de redes sociais são obtidas informações sobre os relacionamentos e conexões dos nós, o que pode indicar, por exemplo, quais são os nós com mais influência, quem são os líderes reais, como a comunicação flui dentro da organização, quais são os pontos de perda de informação, entre outros (COSTA, 2012, p. 33).

Para realizar essa análise, matemáticos e estatísticos aplicam elementos da teoria dos grafos, como a centralidade de um elemento, que mede a sua importância dentre os demais, com o objetivo de encontrar informações úteis que possam solucionar problemas. De acordo a Pimentel e Fulks (2011, p.60), as principais métricas para a análise de redes sociais são:

- **caminho**: sequência de nós que ligam dois nós específicos. Por exemplo, na rede social apresentada na Figura 1, o caminho entre os usuários Ana e Flag é formado pelos nós Ana, Joana, Paula e Flag. Entre dois nós de uma rede, vários podem ser os

caminhos que os conectam, no entanto, quando se utiliza essa métrica, o caminho que possui mais importância é o menor caminho entre dois nós;

- **distância**: medida que indica qual a proximidade entre dois nós através das arestas que compõem o menor caminho. Na Figura 1, a distância entre Ana e Flag é 3, pois o menor caminho entre esses nós possui três arestas;
- **centralidade**: medida que representa o quanto um nó está bem conectado, indicando o seu poder social, valor relacionado ao nível de influência que o nó desempenha na rede;
- **popularidade** ou ***in-degree***: quantidade de conexões que se ligam a um nó. Por exemplo, na Figura 1, o maior *in-degree* é o referente ao nó da Joana, pois ele possui os nós Ana e Nadir ligados a ele, enquanto que os demais nós possuem um *in-degree* igual a 0 ou 1;
- ***out-degree***: quantidade de conexões que partem de um nó. Neste caso, na Figura 1, os nós com maior *out-degree* são os nós de Ana e Joana, ambos com *out-degree* igual a 2;
- **densidade**: medida para calcular a proporção dos relacionamentos de um nó com o total de relacionamentos possíveis. Essa medida é obtida através da divisão da quantidade de ligações de um nó pela quantidade total de ligações possíveis. Por exemplo, se um nó possui ¼ das conexões possíveis, sua densidade é igual a 0,25.

Outro indicador importante na análise de rede social é a força da ligação entre os nós. Granovetter (1983, apud SHAPIRA e ZABAR, 2011, p. 147) aponta dois níveis de intensidade para as ligações entre os elementos da rede: laços fortes e laços fracos. Os laços fortes são as ligações formadas entre pessoas mais próximas, como família, amigos de faculdade, colegas de trabalho etc. Este tipo de ligação produz pouco conhecimento novo, no entanto, esse conhecimento gerado é relevante para ambas as partes. Laços fracos são as ligações entre pessoas que não possuem contato frequente, como antigos colegas do colégio, parentes distantes, conhecidos etc. Este tipo de laço, ao contrário dos laços fortes, fornece muita informação nova, porém, menos relevante. Para a representação da intensidade de um relacionamento, pode-se atribuir pesos às arestas do grafo, como apresentado na Figura 2. Por exemplo, quanto maior for o peso indicado no relacionamento entre dois nós, mais forte ou confiável ele será.

Sendo assim, através dos estudos das redes sociais, é possível identificar quais são os elementos centrais e que exercem maior influência na rede, podendo utilizá-los para alcançar

um maior número de nós ao enviar uma mensagem, por exemplo. Além disso, ainda é possível identificar quais são os elementos que mantém a rede conectada, que fornecem um *link* de acesso entre uma parte da rede e outra. Por exemplo, na Figura 1, o nó Joana é um nó que realiza a ligação entre os usuários Maria, João e Ana aos usuários Paula, Flag, Nadir e Fran, mantendo todos conectados e acessíveis indiretamente.

A seção 2.2 apresenta conceitos referentes à recomendação de especialistas em domínio e sua relação com elementos de redes sociais.

2.2 Sistemas de Recomendação de Especialistas

Sistemas de Recomendação de Especialistas (SREs, ou *Expertise-Locator Systems* - *ELS*) são sistemas que têm o objetivo de ajudar os usuários a encontrar uma pessoa que possui conhecimento em determinado assunto, para que ela possa fornecer ajuda para solucionar algum problema. Estes sistemas também são conhecidos como sistemas localizadores de pessoas (*People-Finder Systems*) (BECERRA-FERNANDEZ, 2006, p. 335). Lin *et. al.* (2008, p. 82) alegam que o papel dos SREs é prover aos usuários informações para ajudá-los a definir se a experiência de um candidato a especialista se adequa com as necessidades exigidas para se resolver o problema e qual a possibilidade de se obter uma resposta deste especialista, caso a sua ajuda seja solicitada.

Sendo assim, os SREs apenas devem fornecer informações, como as qualificações profissionais dos usuários e seus respectivos níveis de atividade. O nível de atividade indica se um usuário participa da rede, interagindo com os demais. Esta informação, que pode ser mensurada pela frequência de atualizações do usuário na rede, é importante, pois não há valor nenhum em saber que um candidato a especialista é o usuário que detém os conhecimentos necessários se o mesmo não participa da rede, e, consequentemente, não fornecerá uma resposta para as necessidades de outro usuário.

Por exemplo, uma determinada empresa de desenvolvimento *web* decide contratar estagiários para as novas vagas que surgiram em sua equipe. O pré-requisito para concorrer às vagas é o conhecimento na linguagem de programação PHP. Inicialmente, a empresa não possui um contato direto com nenhuma universidade, o que pode tornar difícil o processo de obtenção de candidatos para as entrevistas. Porém, ao visitar o *site* institucional de uma universidade, os gestores da empresa descobrem que esta possui um SRE. Este sistema desenvolvido pela universidade possui informações sobre as habilidades de cada aluno e possibilita uma busca, que, através de palavras-chave indicam alguma área de conhecimento,

informa quais os alunos que possuem os conhecimentos correspondentes aos termos utilizados na busca. Dessa forma, ao pesquisar por “PHP”, o sistema apresenta uma lista dos alunos que possuem conhecimento nesta linguagem, e, desta forma, os gestores podem contatá-los e entrevistá-los para os cargos disponíveis em sua empresa.

Uma das formas comumente utilizadas quando procura-se por um especialista é buscar em nossa própria rede pessoal alguém que tenha o conhecimento desejado. Porém, Lin *et. al.* (2008, p. 78) afirmam que apesar das redes pessoais serem muito importantes para se obter respostas rápidas, elas nem sempre são grandes e diversificadas o suficiente para atingir diretamente quem possui a informação ou conhecimento necessário.

Para contornar este problema de alcance da rede de contatos pessoal e atingir um número maior de possíveis especialistas, Ehrlich e Shami (2008, p. 1093) mostram algumas formas que as pessoas também costumam utilizar durante a procura de um especialista. Segundo eles, pode-se enviar mensagens em *broadcast* para comunidades, listas de e-mail, fóruns ou outros grupos de pessoas, e, a partir disso, esperar que alguém com as habilidades requeridas entre em contato. Esta forma de procurar especialistas apresentada por Ehrlich e Shami permite que o usuário atinja um número maior de pessoas, porém, isto não é garantia de que ele obterá uma resposta, pois os especialistas podem não se interessar em ajudar um estranho em suas dificuldades.

Becera-Fernandes (2006, p. 334) aponta que SREs são um tipo de Sistema de Gerenciamento de Conhecimento (*Knowledge Management System* - KMS). Alavi e Leidner (1999, *apud* Becera-Fernandes, 2006, p. 334) definem sistemas de gerenciamento de conhecimento como sendo uma linha de sistemas que enfocam nas atividades profissionais e de gestão, centrando-se na criação, coleta, organização e disseminação de conhecimento de uma organização.

Becera-Fernandes *et. al.* (2004, *apud* Becera-Fernandes, 2006, p. 334-335) desenvolveram uma *framework* para trabalhar com KMS, em que os sistemas são classificados em:

- sistemas de captura de conhecimento (*knowledge capture systems*), que capturam e formalizam o conhecimento através de estruturas (como, por exemplo, mapas conceituais) que tornem fácil o entendimento sobre determinado assunto;
- sistemas de aplicação de conhecimento (*knowledge application systems*), que auxiliam na resolução de problemas com base na utilização de um conhecimento gerado anteriormente;

- sistemas de descoberta de conhecimento (*knowledge discovery systems*), que obtêm novo conhecimento a partir da aplicação de algoritmos em informações existentes, como algoritmos de *Data Mining*;
- sistemas de compartilhamento de conhecimento (*knowledge sharing systems*), que organizam e distribuem conhecimento em repositórios. Tais repositórios se diferem de acordo com o propósito das informações que armazenam, como, por exemplo, se um repositório armazena apenas informações sobre os processos adotados por uma organização, ele é utilizado por um sistema que compartilha o conhecimento relacionado aos modelos de processo da organização. Os sistemas de recomendação de especialistas se encaixam nesta modalidade de sistema, pois armazenam informações sobre as aptidões de um usuário e as tornam acessíveis através de um sistema de busca.

Seid e Kobsa (2003, *apud* Ehrlich e Shami, 2008, p. 2) identificaram dois dos principais motivos para a procura de especialistas. O primeiro motivo diz respeito à procura de pessoas como fontes de informação, por exemplo, pessoas com conhecimento avançado em algoritmos genéticos para auxiliar na definição de parâmetros da função de *fitness* para um problema específico. Já o segundo motivo refere-se a localizar pessoas que podem exercer uma função organizacional ou social dentro de uma empresa, como pessoas com habilidades em gerência de projetos para serem líderes em um projeto especial.

Ehrlich e Shami (2008), por outro lado, através de sua pesquisa empírica que analisou o questionamento "o que as pessoas procuram quando necessitam de um especialista?", definiram quatro categorias para classificar as necessidades de especialistas, sendo:

- busca por respostas (*answers*): busca-se por uma resposta para uma pergunta específica, em que a resposta é mais importante do que quem respondeu. Um exemplo deste tipo de situação pode ser a resposta para a pergunta "como definir um sumário automático no Microsoft Word 2010?", em que a resposta que contém o passo-a-passo das ações corretas é mais importante do que saber se foi o João ou a Maria que forneceu a resposta;
- busca por pessoas (*person*): visa-se encontrar uma pessoa com habilidades específicas que são úteis na resolução de um problema, por exemplo, encontrar um especialista na linguagem de programação PHP para dar suporte na migração de plataforma de um sistema de folha de pagamento;

- busca por conhecimento (*awareness*): pretende-se obter conhecimento sobre um assunto ou situação, em que nem o tópico abordado e nem o seu autor são importantes. Um exemplo desta situação pode ser um profissional de engenharia de *software* que deseja manter-se informado sobre os trabalhos que estão sendo desenvolvidos pela comunidade acadêmica, ou seja, ele não tem em mente um tópico ou autor específico, ele deseja apenas verificar quais são os trabalhos que estão sendo publicados;
- busca por interessados em informações (*provide information*): neste caso, não procura-se obter respostas, pessoas com habilidades específicas ou conhecimento, o objetivo é encontrar alguém que se interesse por determinada informação. Por exemplo, uma pessoa descobre como burlar a segurança de um sistema e deseja encontrar outras pessoas que podem se interessar por esta informação. Os autores destacam que dentre todas as necessidades analisadas, poucas se encaixaram nesta categoria e que a maioria das ferramentas não são projetadas levando em consideração este objetivo.

O desenvolvimento de um SRE é uma tarefa não trivial, uma vez que o conceito de "especialidade" não possui um padrão formalizado para sua representação. A especialidade é um tipo de conhecimento tácito, ou seja, é composta por informações carregadas nas mentes das pessoas e, por isso, de difícil acesso. Sendo assim, os SREs fazem uso de informações explícitas, como artigos publicados ou outros documentos, para identificar os conhecimentos dos usuários, e, através disso, possibilitar o acesso ao seu conhecimento tácito (SERDYUKOV *et. al.*, 2009, p. 39).

Fazel-Zarandi *et. al.* (2011, p. 41) também notaram alguns problemas enfrentados no desenvolvimento de um SRE, que são causados pelo fato do conhecimento de um usuário ser dinâmico, difícil de qualificar e variável em relação ao tempo. Esta dificuldade é observada com o passar do tempo, pois uma pessoa aprende novos conceitos, ampliando seu campo de conhecimento, e, para o sistema, é difícil criar parâmetros para identificar e mensurar o nível de deste conhecimento adquirido pelo usuário.

Reichling e Wulf (2009, p. 59) citam que um dos maiores desafios para o desenvolvimento de um SRE é construir perfis de conhecimento de um usuário de forma rápida e confiável com o mínimo esforço manual possível, e, ao mesmo tempo, respeitando a sua privacidade. Esse detalhe se torna importante porque as formas comumente utilizadas para a obtenção automatizada de informações baseiam-se na extração automática de informações presentes em documentos de autoria do usuário, ou ainda, em bases de dados de outros sistemas de uma organização. Sendo assim, ao desenvolver um SRE, deve-se evitar processar

informações as quais o usuário não esteja ciente e tenha fornecido uma permissão para tal ação.

SERDYUKOV *et. al.* (2009, p. 39) apontam que comumente a tarefa de encontrar um especialista é dividida em duas etapas de igual importância: identificação das especialidades dos usuários e recomendação do melhor especialista para uma determinada necessidade descrita pela busca realizada. As seções 2.2.1 e 2.2.2 apresentam, de forma mais detalhada, cada uma destas etapas.

2.2.1 Identificação das especialidades

A identificação das especialidades de um determinado usuário pode ser feita de algumas maneiras, que dependem do contexto e das informações disponibilizadas por ele. Serdyukov *et. al.* (2009, p. 40) citam duas formas de se obter as especialidades de um usuário: análise do perfil e análise de documentos.

Inicialmente, quando os primeiros SREs foram desenvolvidos, as informações eram extraídas através da análise textual dos perfis dos usuários, como, por exemplo, o CONNEX (BECERA-FERNANDES, 2006, p. 335), um localizador de peritos KMS desenvolvido pela empresa HP. O CONNEX tinha por objetivo construir uma rede de peritos que estivesse disponível *on-line*, e funcionasse como um guia do conhecimento humano dentro da empresa. Os perfis dos especialistas para essa abordagem de SREs continham informações resumidas sobre formação, interesses, idiomas, meios de contato etc. O problema relacionado a esse tipo de abordagem é que a identificação das especialidades depende da atualização constante das informações, o que exige tempo do usuário, que, muitas vezes, não o possui, o que acaba resultando em perfis incompletos.

Para sanar as dificuldades encontradas na identificação das especialidades utilizando a análise de perfil, os SREs sucessores, como o *ExpertFinding* (REICHLING e WULF, 2009), consideram a análise dos documentos relacionados com o usuário como indicadores de suas especialidades. A ideia por trás dessa metodologia é que quanto maior for o número de documentos relacionados a um mesmo usuário e que tratam de um mesmo assunto, maior será a probabilidade de esse usuário ser um especialista nesse assunto. Por exemplo, um usuário "A" possui, relacionados a ele, 4 documentos que abordam o assunto "PMBOK", enquanto que um outro usuário "B" possui apenas 2 documentos associados a ele para esse mesmo assunto. Sendo assim, de acordo com esta metodologia de identificação e mensuração de especialidades, o usuário "A" teria um nível de especialidade 4 para o assunto em questão,

enquanto que o usuário “B” possuiria um nível 2, o que indica que ele possui menos experiência que o usuário “A” neste assunto.

Reichling e Wulf (2009, p. 62), ao desenvolverem um SRE denominado *ExpertFinding*, combinam duas técnicas para a criação do perfil de conhecimentos do usuário. A primeira técnica cria uma lista com palavras-chave a partir de documentos de texto produzidos pelo próprio usuário. Como para desenvolver um documento é necessário conhecimento sobre o assunto abordado, assunto esse representado pelas palavras-chave extraídas dos documentos, associa-se, então, tais palavras-chave ao autor do documento como um indicativo de seus conhecimentos. A segunda técnica utilizada consiste na obtenção direta de informações sobre o usuário através do preenchimento de formulários, onde eles podem registrar informações sobre educação, idiomas, qualificações específicas, conhecimento de idiomas, entre outras.

Além de identificar as especialidades de um usuário, é necessário definir o nível de proficiência correspondente a ela, ou seja, deve-se saber a quantidade de conhecimento que o usuário possui para cada uma de suas habilidades. Segundo Fazel-Zarandi e Fox (2011, p. 2), proficiência é uma habilidade que depende de vários fatores, como a familiaridade e experiência com o assunto, tempo de estudo e etc.

Fazel-Zarandi e Fox (2011, p. 3) definiram, para o sistema desenvolvido por eles, quatro estados que uma habilidade pode possuir, sendo:

- **habilidade demonstrada**: quando o usuário prova, por meio da prática, que ele possui uma determinada habilidade;
- **habilidade sugerida**: quando um outro usuário indica que outro possui conhecimento em determinado tópico;
- **habilidade desconhecida**: quando o usuário autodeclara uma habilidade, como, por exemplo, através do preenchimento de formulários;
- **habilidade refutada**: quando o usuário não demonstra, na prática, o conhecimento atribuído à ele.

Apesar de existirem algumas abordagens de níveis ou estados em que o conhecimento do usuário é classificado, é importante lembrar que essa classificação pode sofrer mudanças ao longo do tempo. Por exemplo, considerando os estados definidos por Fazel-Zarandi e Fox (2011) para a classificação de uma habilidade/conhecimento, Thor, que é um novato em determinada empresa, ao ser contratado, informa que possui experiência em CSS 3. Neste momento, essa habilidade do usuário é uma habilidade “desconhecida”, pois não há nenhuma

prova que demonstre que ele realmente possui o conhecimento informado. Em outro momento, Loki, um de seus colegas de trabalho que já trabalhou com Thor em outro projeto, informa que ele realmente possui a habilidade especificada anteriormente, habilidade essa, que, a partir de agora, assume o estado "sugerida". Porém, quando Fury, gerente de projetos, repassa uma atividade de implementação de um *layout* para Thor, o mesmo não desenvolve o trabalho esperado utilizando os elementos corretos do CSS 3, e, dessa forma, esta sua habilidade assume o estado "refutada". Com o passar do tempo, Thor adquire mais conhecimento e habilidades em CSS 3, e, finalmente, consegue desenvolver qualquer *layout* repassado para ele utilizando os elementos corretos do CSS 3. Neste novo cenário, a habilidade em CSS 3, possuída por Thor, passa a ostentar o estado "demonstrada", pois há provas de que ele possui tal conhecimento.

Fazel-Zarandi e Fox (2011, p. 3) defendem que uma forma eficiente e confiável para reunir e validar as habilidades de um usuário são as recomendações e avaliações em pares. Uma recomendação, neste contexto, é a sugestão realizada por uma pessoa P_1 de uma pessoa P_2 como especialista em determinada área. Neste caso, a importância da recomendação varia de acordo com o nível de confiabilidade atribuído à P_1. A avaliação em par, como o próprio nome sugere, é a reunião de duas pessoas para avaliar uma terceira. Estima-se que quanto maior a quantidade de pares que avaliaram uma pessoa, maior é a confiabilidade da avaliação, sendo que, no mínimo, são necessárias de 3 a 5 avaliações para se ter um valor aceitável.

A seção 2.2.1.1 apresenta algumas das possíveis formas de se identificar e mensurar o nível de proficiência das especialidades de uma pessoa.

2.2.1.1 Algumas abordagens para detectar especialidades

Dentro da literatura, uma das técnicas utilizadas para identificar especialidades é a de análise de texto plano, que se baseia em técnicas de *text mining*. No entanto, existem outras abordagens que podem tornar o mecanismo de identificação de especialidades mais preciso, como dicionários de dados, taxonomias e ontologias. A presente seção apresentará cada um desses elementos que podem estar presentes na etapa de identificação de especialidades de um usuário.

Análise de texto plano

O processamento de documentos textuais, segundo Baeza-Yates e Ribeiro-Neto (1999, p. 165), é um procedimento que pode ser dividido em cinco principais operações sequenciais:

análise léxica, eliminação de *stopwords*, *stemming*, seleção de termos índices e construção de estruturas de categorização de termos. Estas etapas são abordadas nos próximos parágrafos.

A análise léxica tem como objetivo converter uma sequência de caracteres do documento em uma sequência de palavras, que serão candidatas a termos índices (termos que descrevem o documento). Sendo assim, esta etapa recebe como entrada o conteúdo do documento e, durante o processamento, deve retirar do texto todos os caracteres especiais como pontuações, símbolos, excesso de espaços, normalizar as palavras para uma forma padrão (deixar todas maiúsculas ou minúsculas), etc., resultando, como saída, apenas uma sequência de palavras.

A eliminação de *stopwords*, que recebe como entrada a saída da etapa de análise léxica, realiza a remoção de palavras que são muito comuns em um idioma, e, por isso, não agregam valor semântico, pois são muito genéricas. Exemplos de *stopwords* são artigos (como “a”, “o”, “as”, “os”), preposições (como “de”, “com”), entre outras.

A próxima etapa, que recebe como entrada a saída do processo de eliminação de *stopwords*, é conhecida como *stemming*. Esta etapa tem o propósito de reduzir as palavras ao seu radical (*stem*), removendo partes que indicam o gerúndio (“ando”, “endo”, “indo” e etc.), plural (“s”), gênero (masculino e feminino) etc. A vantagem de se reduzir uma palavra ao seu radical é que um mesmo radical representa todas as variações da palavra. Por exemplo, para a palavra “cachorro”, o radical é “cachorr”, que representa as variações, “cachorros”, “cachorrinho”, “cachorrinhos”, “cachorrão” e “cachorrões”.

A etapa de seleção de índices tem o propósito de identificar quais serão os radicais que representarão o documento. Há a possibilidade de se utilizar todos os radicais gerados na etapa de *stemming*, ou ainda, selecionar alguns dos termos mais relevantes. A seleção de termos relevantes pode ser realizada por especialistas humanos ou através de abordagens que realizam essa tarefa de forma automática, como algoritmos que definem um peso para cada termo, e, assim, pode-se escolher os termos que possuem os maiores pesos como representantes do conteúdo do documento.

A fase de construção de estruturas de categorização de termos tem como objetivo otimizar a recuperação da informação no momento em que o usuário realizar uma busca. Isso é feito através da relação dos termos selecionados em um documento com outros termos de significado semelhante.

A definição de pesos para os termos encontrados em um documento, como meio de filtrar os principais, que melhor representam um documento, segundo Baeza-Yates e Ribeiro-Neto (1999, p. 38), pode ser computada através dos fatores de normalização *tf-idf* (*term*

frequency - inverse document frequency). A equação para o cálculo do tf-idf é baseada no produto entre a frequência do termo no documento (tf) e a frequência inversa de documento para o termo (idf), ambos apresentados nos próximos parágrafos.

O cálculo referente à frequência de um termo em um documento é baseado no cálculo da repetição do termo no documento. A equação para o cálculo da tf é apresentado na Equação 1Erro! Fonte de referência não encontrada. (BAEZA-YATES e RIBEIRO-NETO, 1999, p. 29), abaixo.

$$tf_{i,j} = \frac{freq_{i,j}}{max_l\ freq_{l,j}} \quad (1)$$

Na Equação 1, temos que a frequência do termo i em um documento j é obtida pela divisão da quantidade de ocorrências de i em j ($freq_{i,j}$) pela maior frequência de termo do documento ($max_l\ freq_{l,j}$). Esta equação mensura a importância de um termo em um documento através da sua relação com o termo que possui a maior frequência para o mesmo documento. A Figura 3 apresenta um exemplo de documento, que terá a frequência dos seus termos calculada.

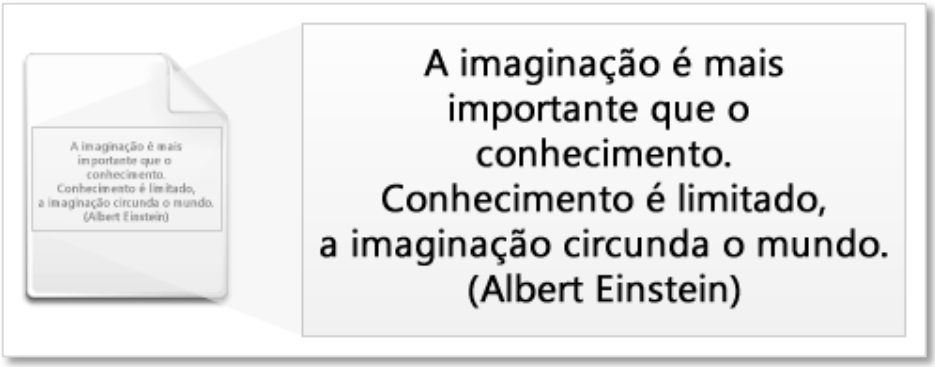


Figura 3: Exemplo de documento

A Figura 3 apresenta um exemplo do conteúdo de um documento, que, ao ter os pesos de seus termos calculados, resulta na Tabela 1:

Tabela 1: Resultado do cálculo exemplo da frequência dos termos

Termo	Radical	*freq*	*tf*
imaginação	imagin	2	1
importante	import	1	0,5
conhecimento	conhec	2	1

limitado	limit	1	0,5
circunda	circund	1	0,5
mundo	mund	1	0,5
albert	albert	1	0,5
einstein	einstein	1	0,5

Ao observar a Tabela 1, nota-se que os termos de maior frequência são os termos "imaginação" e "conhecimento", ambos com $freq$ = 2 e tf = 1. Os demais termos do documento possuem $freq$ = 1 e tf = 0,5, resultado da divisão da sua frequência pela maior frequência de termos do documento.

Porém, de acordo à Baeza-Yates e Ribeiro-Neto (*Modern Information Retrivelval*, 2ª edição, 2011, apud BAEZA-YATES e RIBEIRO-NETO, 2011, p. 34), a equação utilizada na literatura para o cálculo da tf é uma variante da Equação 1. Esta variante é apresentada na Equação 2**Erro! Fonte de referência não encontrada.** (BAEZA-YATES e RIBEIRO-NETO, 2011, p. 34, adaptada).

$$tf_{i,j} = \begin{cases} 1 + \log f_{i,j} \ , se\ f_{i,j} > 0 \\ 0 \qquad\quad , caso\ contrário \end{cases} \tag{2}$$

A **Erro! Fonte de referência não encontrada.** apresenta a adição de alguns elementos em relação a **Erro! Fonte de referência não encontrada.**, onde a $tf_{i,j}$ recebe o valor do log da $f_{i,j}$ somado ao número "1", caso a $f_{i,j}$ seja maior que zero, e, caso contrário, $f_{i,j}$ recebe o valor "0". A inclusão do logaritmo e a adição em um, segundo os autores, se justifica por proporcionar uma comparação direta entre os pesos da $tf_{i,j}$ e da idf_i. Através da utilização dessa nova equação, os termos mostrados na Tabela 1, referentes ao documento apresentado na Figura 3 tem seus valores alterados, conforme apresentado na Tabela 2.

Tabela 2: Alteração dos pesos dos termos através da utilização da **Erro! Fonte de referência não encontrada.**

Radical	***freq***	***tf***
imagin	2	1,30
import	1	1
conhec	2	1,30
limit	1	1

circund	1	1
mund	1	1
albert	1	1
einstein	1	1

Como apresentado na Tabela 2, todos os termos do documento tiveram seus valores da tf alterados. Após realizar o cálculo de $tf_{i,j}$, o próximo passo é obter o valor da frequência inversa de documento para o termo i. O cálculo da idf de um termo é realizado através da **Erro! Fonte de referência não encontrada.** (BAEZA-YATES e RIBEIRO-NETO, 1999, p. 40), apresentada abaixo.

$$idf_i = \log \frac{N}{n_i} \quad (3)$$

Na Equação 3, a idf de um termo i é calculada pelo log da divisão de N, que é o número total de documentos existentes na base, por n_i, que é o número de documentos da base relacionados ao termo i. A Tabela 3 apresenta um exemplo do cálculo da idf, sendo que a base possui um total de 10 documentos.

Tabela 3: Resultado do cálculo exemplo da frequência inversa dos termos

Radical	Documento	freq	tf	idf
imagin	A	2	1,30	0,30
import	A	1	1	0,30
conhec	A	2	1,30	0,30
limit	A	1	1	1
circund	A	1	1	1
mund	A	1	1	1
albert	A	1	1	1
einstein	A	1	1	1
conhec	B	3	1,48	0,30
import	B	1	1,30	0,30
imagin	B	4	1,60	0,30
fertil	B	2	1,30	1

O objetivo do cálculo da idf é mensurar a importância da relação termo/documento com os demais termos de outros documentos, dando importância maior para os termos que aparecem em um número menor de documentos. Como observado na Tabela 3, os termos "imaginação", "importante" e "conhecimento", por aparecerem em mais documentos, possuem um menor valor de idf, pois eles não são tão específicos e descritivos apenas para um documento, ou seja, o assunto representado por eles, dentro de toda a base de documentos, é mais comum do que os assuntos representados pelos outros termos.

Por fim, quando já se sabe qual a tf e idf de cada termo, deve-se estipular o peso que tal termo possui em um determinado documento. Esse cálculo do peso, que demonstra o quanto o termo é descritivo para o documento, é realizado usando a Equação 4**Erro! Fonte de referência não encontrada.** (BAEZA-YATES e RIBEIRO-NETO, 2011, p. 42).

$$w_{i,j} = \begin{cases} tf_{i,j} \ X \ idf_i \ , se \ f_{i,j} > 0 \\ 0 \qquad\qquad , caso \ contrário \end{cases} \tag{4}$$

Na Equação 4, tem-se que o peso w de um termo i em um documento j é igual ao produto da $tf_{i,j}$ pela idf_i, caso a $f_{i,j}$ seja maior que 0, e, caso contrário, o peso do termo i no documento j é igual a 0. Sendo assim, os pesos dos termos apresentados anteriormente são os mostrados na Tabela 4.

Tabela 4: Resultado do cálculo dos pesos dos termos de cada documento

Termo	Documento	freq	tf	idf	w
imagin	A	2	1,30	0,30	0,39
import	A	1	1	0,30	0,30
conhec	A	2	1,30	0,30	0,39
limit	A	1	1	1	1
circund	A	1	1	1	1
mund	A	1	1	1	1
albert	A	1	1	1	1
einstein	A	1	1	1	1
conhec	B	3	1,48	0,30	0,44
import	B	1	1,30	0,30	0,39
imagin	B	4	1,60	0,30	0,48
fertil	B	2	1,30	1	1,30

Como mostrado na Tabela 4, os termos que aparecem menos vezes, ou seja, que representam um assunto de um documento mais específico, recebem um peso maior, para que possam se destacar em meio aos que possuem um maior número de ocorrência em outros documentos. Após esse processo de identificação e mensuração dos pesos dos termos, nos SREs, além de associar tais termos aos documentos, eles também são associados aos autores dos documentos como um indicador de especialidade. Por exemplo, considerando que João seja o autor do documento B, ele será associado aos termos “conhecimento”, “importante”, “imaginação” e “fértil” e com os seus respectivos pesos.

Análise semântica

Para que a identificação de especialidades posa ser mais precisa, pode-se adicionar análise semântica no processo de identificação das experiências das pessoas. Para isso, podem ser adotadas diferentes abordagens, por exemplo, dicionário de dados (taxonomia/*thesaurus*) ou ontologias.

Segundo Becera-Fernandes (2006, p. 339), taxonomia é o estudo dos princípios gerais da classificação científica, enquanto que a ontologia é uma especificação formal de representação de objetos, conceitos e outras entidades existentes em um domínio e as relações entre esses elementos.

A taxonomia permite organizar (através de palavras-chave) o conhecimento ou as competências/áreas de conhecimento de uma organização, identificando quais são as suas áreas fundamentais. Esse processo é geralmente caro, pois a etapa de identificação dos termos consome muito tempo, uma vez que se deve evitar desenvolver uma taxonomia pequena, e, consequentemente, limitada e que ocasionaria o risco de eliminar áreas importantes dessa descrição de conhecimento, ou uma taxonomia muito abrangente, com mais elementos que o necessário, o que tornaria o seu uso muito complicado.

Por exemplo, na Figura 4, observa-se os termos de uma taxonomia para um contexto específico denominado “Matemática”.

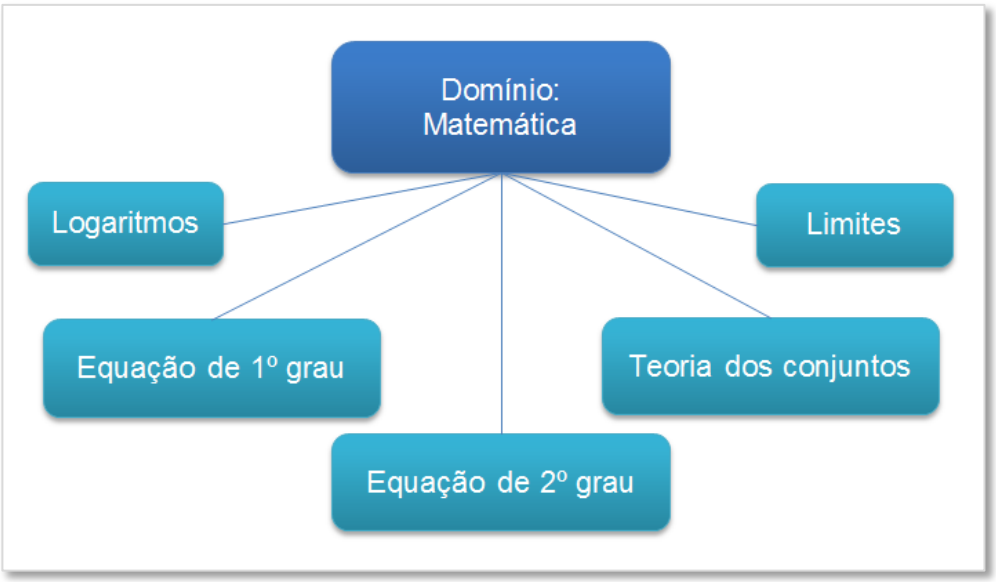


Figura 4: Exemplo da taxonomia do domínio "Matemática"

Como mostrado na Figura 4, existem termos taxonômicos simples que possuem relação apenas com o conceito “Matemática”, não há nenhum tipo de relacionamento direto entre eles. Neste caso, os termos têm o objetivo de ajudar na organização e categorização do conteúdo presente no contexto. Por exemplo, se o contexto for uma biblioteca de artigos matemáticos, todos os materiais podem ser agrupados de acordo ao seu assunto, facilitando sua posterior recuperação.

Porém, utilizando ontologia, pode-se definir axiomas que exprimam a relação entre os termos, como, por exemplo, uma relação de precedência entre o termo “Equação de 1º grau” e “Equação de 2º grau”. Sendo assim, se for identificado que o usuário João foi associado ao termo “equação de 2º grau”, ele também será associado ao termo “Equação de 1º grau”, pois o axioma desenvolvido formaliza essa relação, onde, quem conhece equações de 2º grau, obrigatoriamente deve conhecer equações de 1º grau.

Além dos termos taxonômicos que descrevem um domínio e da ontologia que pode ser aplicada a eles, existe o conceito de *thesaurus*, que, de forma simples, consiste em um conjunto pré-compilado de palavras para determinado domínio de conhecimento (que, no caso, são os elementos presentes na taxonomia), e, para cada uma destas palavras, outro conjunto de palavras relacionadas, que geralmente são sinônimos. Por ser um conjunto de termos, um *thesaurus* pode ser entendido como um dicionário de dados do domínio, ou seja, uma taxonomia. Por outro lado, um *thesaurus* também pode ser referenciado como uma ontologia, já que existe uma hierarquia entre um termo e seus termos relacionados. Porém, ao referir-se a um *thesaurus* como uma ontologia, entende-se que essa é uma ontologia limitada,

ou seja, não fornece suporte para todos os elementos ontológicos, como axiomas, apenas a relação hierárquica entre os termos que descrevem o domínio.

Por exemplo, no contexto de trabalhos científicos sobre inteligência artificial, existe uma taxonomia com vários termos, sendo, entre eles, o termo ‘algoritmos genéticos’, que possibilita o agrupamento de todo material relacionado a esse tema. Porém, este conceito também pode ser referenciado como ‘*genetic algorithms’*, o que acarretaria na criação de um novo grupo de trabalhos, embora ‘algoritmos genéticos’ e ‘*genetic algotithms*’ representem a mesma coisa. Sendo assim, a solução para esse impasse é o desenvolvimento de um *thesaurus*, que teria um termo principal chamado ‘algoritmos genéticos’ e todas as suas variações, como ‘*genetic algotithms*’, possibilitando o agrupamento de trabalhos relacionados a ambos os termos em um único conjunto. A vantagem deste tipo de organização é que em casos onde o usuário realiza a recuperação das informações com base em uma consulta, será apresentado para ele um conjunto maior de resultados, o que não aconteceria se os termos ‘algoritmos genéticos’ e ‘*genetic algotithms*’ fossem independentes.

Após a identificação das especialidades de um usuário (primeira etapa do processo geral de recomendação de especialistas), surge a etapa da recomendação dos especialistas, que apresenta ao usuário do sistema quais são os especialistas que possuem o conhecimento por ele requerido. Esta etapa de recomendação de especialistas é apresentada na seção 2.2.2.

2.2.2 Recomendação de especialistas

Após a identificação e mensuração das especialidades dos usuários, deve-se prover um meio para a sua recuperação. Essa recuperação geralmente é realizada através de uma interface de busca, onde o usuário informa qual é a área de especialidade que ele requer e o sistema se encarrega de exibir uma lista, geralmente em ordem decrescente de experiência, dos especialistas localizados.

Como mencionado no final da seção 2.1, mecanismos de recomendação que levam em consideração o contexto do usuário através das suas informações sociais, tendem a apresentar um conjunto de resultados mais precisos. Sendo assim, Fazel-Zarandi *et. al.* (2011, p. 43-44), propuseram algumas teorias para a recomendação de especialistas baseando-se teorias das ciências sociais, sendo elas:

- **teoria do auto-interesse** (*self-interest theory*): determina a seleção do especialista mais qualificado dentre todos. Neste caso, a teoria baseia-se em medidas de conhecimento que são derivadas de atributos pessoais, como a quantidade de trabalhos

publicados em uma determinada área. Por exemplo, se um usuário "A" possui 10 trabalhos sobre o assunto "usabilidade", ele será apresentado como mais qualificado que um usuário que possui apenas 2 trabalhos sobre o mesmo tema;

- **teoria da homofilia** (*homophily theory*): baseia-se na recomendação de especialistas que possuem alguma característica semelhante ao usuário que realiza a busca. Os atributos que definem semelhança entre os usuários podem ser a idade, sexo, área de atuação, cargo etc.;
- **teoria da troca social** (*social exchange*): considera a troca de materiais ou informações entre os usuários da rede. Por exemplo, em uma rede que possui apenas usuários que realizam pesquisas e publicam seus trabalhos, a lista de especialistas apresentadas após uma busca realizada por um usuário "u" pode dar uma prioridade maior a especialistas os quais referenciaram "u" em algum de seus trabalhos e possuem o conhecimento requerido. Este exemplo busca simular a relação de troca, onde, em determinado momento, o trabalho do usuário "u" foi citado por outro usuário "a", e, agora que "u" precisa de ajuda em alguma área onde "a" é especialista, ele recebe um destaque maior na lista de resultados, para que de certa forma, ele possa retribuir a ajuda, tenha ela ocorrido de forma direta ou indireta em um momento anterior;
- **teoria do contágio** (*contagion theory*): fundamenta-se na ideia de "seguir a multidão", ou seja, os especialistas mais populares são apresentados primeiro. Dentre algumas formas de se estimar qual é o especialista mais popular, pode-se utilizar a quantidade de referências em outros trabalhos que ele possui, ou seja, quantas vezes ele foi citado em todos os demais trabalhos.

Dadas essas teorias, Fazel-Zarandi *et. al.* (2011, p. 46) recomendam que os SREs que levam em consideração aspectos sociais não se baseiem apenas em uma das teorias isoladamente. Sendo assim, o ideal é uma junção de duas ou mais teorias, de acordo com o contexto, para uma melhor resposta ao usuário.

De forma geral, os sistemas de recomendação de especialista baseiam-se nas duas etapas apresentadas, a etapa de identificação e a etapa de recomendação de especialistas. Porém, apesar dessas etapas em comum, existem algumas diferenças que podem existir entre implementações de sistemas do tipo. Essas diferenças são abordadas na próxima seção.

2.2.3 Diferenças entre SREs

Apesar de os SREs apresentarem propósitos semelhantes, que se baseiam na localização de pessoas que possuem determinada habilidade ou conhecimento, existem fatores que os diferenciam. Becera-Fernandes (2006, p. 336-337) citam alguns exemplos que podem diferir um SRE dos demais, sendo: finalidade do sistema, método de acesso, autoavaliação, participação, taxonomia, níveis de competência, além de diferenças relacionadas às tecnologias empregadas no desenvolvimento, como, por exemplo, a plataforma de banco de dados ou linguagem de programação utilizada. Essas diferenças serão detalhadas a seguir:

- **finalidade**: cada SRE pode ter um propósito específico, como encontrar especialistas para resolver um determinado problema ou para analisar as deficiências de capital intelectual da empresa. De acordo com a necessidade da empresa, pode-se modelar um SRE que apenas encontre especialistas, ou ainda, modelá-lo de forma a saber se os empregados de um setor crítico possuem as qualificações necessárias para executar suas tarefas, e, em caso negativo, fornecer treinamento para estas pessoas.
- **método de acesso**: se refere à plataforma pela qual os SREs são acessados. Por exemplo, um SRE pode ser acessado apenas pela intranet da empresa ou pela internet, em qualquer local.
- **autoavaliação**: é a forma como as especialidades dos usuários são obtidas. Uma das possibilidades é a de obtenção de especialidades através do preenchimento de formulários pelo próprio usuário, em que ele indica quais são suas especialidades, o que pode, dependendo do caso, resultar em níveis de experiência distorcidos. Por exemplo, se a empresa está em uma época de corte de funcionários, há uma grande chance das pessoas informarem que possuem mais especialidades do que realmente possuem para tentar manter o seu cargo dentro da organização.
- **participação**: define o nível abrangido pelo SRE dentro da organização. Por exemplo, o SRE pode ser desenvolvido apenas para um setor gerencial da empresa, sem levar em consideração os demais, como, o setor de pesquisa de campo. Outra possibilidade é abranger apenas uma filial da organização, para simular o funcionamento do sistema e analisar a sua utilidade na realização das tarefas.
- **taxonomia**: refere-se aos termos utilizados para indexar o conhecimento dentro da organização, indicando quais são os conhecimentos mais relevantes para a empresa. Esta disposição taxonômica é importante para mapear o conhecimento do usuário com base nestes conceitos julgados como elementos-chave. Por exemplo, se um dos termos

da taxonomia da empresa for “redes neurais”, pode-se estipular, para cada usuário, um valor para este atributo, independentemente da análise de conhecimentos ter inicialmente identificado este tipo de conhecimento para o usuário.

- **níveis de competência**: definem a capacidade de um usuário sobre um determinado assunto. Wiig (1993, *apud* Becera-Fernandes, 2006) definiu oito níveis de competência que podem ser utilizados para a classificação do nível de conhecimento dos usuários sobre determinado tema, sendo:
    - ignorante: totalmente inconsciente, ou seja, nunca teve contato ou ouviu algo sobre o tema;
    - iniciante: vagamente consciente, porém, sem nenhuma experiência;
    - iniciante avançado: consciente, porém, possui pouca qualificação, ou seja, o usuário sabe do que se trata o tema, apenas não possui qualificação na área;
    - competente: estreitamente qualificado;
    - proficiente: conhecimento em áreas selecionadas;
    - especialista: altamente proficiente em determinada área, sendo reconhecido pelos demais;
    - mestre: especialista em várias áreas e amplamente reconhecido;
    - grande mestre: especialista em nível mundial em suas áreas de domínio.

  A definição desses níveis de especialidade podem ser úteis de diversas formas. Por exemplo, no contexto de uma empresa de desenvolvimento de *software*, que trabalha com a linguagem PHP, identifica-se que alguns usuários se encaixam nos níveis iniciais de competência nessa linguagem. Já que essa linguagem é a base para o desenvolvimento dos projetos, e que existem desenvolvedores que não possuem um conhecimento avançado nesta área, os usuários com nível de especialização mais elevado podem desenvolver e ministrar cursos para que haja uma nivelação de conhecimentos.

A seção 2.2.4 apresenta alguns exemplos de sistemas de recomendação de especialistas.

2.2.4 Exemplos de SREs

Na literatura, constantemente surgem novos trabalhos com a temática abordada. Dentre eles, podem ser mencionados o ICARE (PETRY, 2007), SWEETS (SILVA, 2009), SmallBlue (LIN *et. al.*, 2008), ArnetMiner (TANG *et. al.*, 2008) etc. Dentre esses, tendo como critério

utilizado a recomendação de especialistas em ambientes virtuais baseados em redes sociais, foram selecionados dois dos trabalhos para serem apresentados: o SmallBlue, um projeto desenvolvido pela IBM para ajudar os usuários a gerenciarem sua rede social para encontrar e acessar conhecimento e informação, e o ArnetMiner, que possibilita a extração e mineração de redes sociais acadêmicas.

2.2.4.1 SmallBlue

Lin *et. al.* (2008) desenvolveram uma suíte de aplicativos conhecidos por SmallBlue para desbloquear e utilizar o conhecimento de negócio armazenado na mente dos funcionários das organizações, sem que esses funcionários se envolvam explicitamente para fornecer qualquer tipo de informação sobre suas especialidades e experiências. Para atingir esse objetivo, a aplicação foi dividida em alguns módulos, sendo:

- *SmallBlue Client*: um *software* instalado na máquina de cada usuário e que realiza a captação de dados para inferir em quais áreas o usuário é um perito;
- *SmallBlue Ego*: ferramenta de gerenciamento de capital social, que permite a visualização da rede do usuário e mostra valores sociais dos amigos, demonstrando quais são os tipos de pessoa com as quais ele pode se conectar;
- *SmallBlue Find*: ferramenta que proporciona a recuperação dos especialistas de acordo com o conhecimento desejado;
- *SmallBlue Reach*: mecanismo de análise de rede social que mostra para um usuário qual é o menor caminho entre ele e o especialista para a sua necessidade. Essa ferramenta também apresenta outras informações sobre um especialista, como suas atividades em blogs externos, fóruns, perfis e etc.;
- *SmallBlue Net*: ferramenta para a visualização e análise de informações sociais em larga escala. Por exemplo, para um tópico específico, ela identifica quais são os principais especialistas, fornece a visualização de especialistas de um mesmo grupo de acordo com seus interesses em comum e etc.

Todo esse processo de descoberta de conhecimento foi dividido em duas etapas. A primeira etapa realizada por esse SRE é a aquisição de dados referentes às especialidades dos usuários. Este processo é o maior desafio do sistema devido às questões existentes em relação à privacidade dos usuários, já que as fontes de dados são os e-mails e mensagens instantâneas trocadas entre os usuários. Para que vários usuários se convencessem que os seus dados seriam usados de forma profissional, a equipe desenvolveu rígidas e detalhadas normas de

uso, explicando quais e como cada tipo de informação seria utilizada, o que facilitou a adesão de várias pessoas que possuíam certo receio de suas informações serem acessadas por outras. A segunda etapa referente a esse método é processar os dados utilizando algoritmos de mineração de texto e inteligência artificial para inferir as especialidades de cada usuário e formar uma rede social entre todos os participantes.

O SmallBlue demonstra que os dados utilizados por SREs para a inferência das especialidades dos usuários podem ser obtidas de qualquer fonte, como e-mails enviados e conversas em comunicadores instantâneos, desde que a privacidade das informações seja garantida aos usuários. Outro exemplo de SRE existente na literatura é o ArnetMiner, apresentado na seção 2.2.4.2.

2.2.4.2 Arnetminer

Tang *et. al.* (2008) implementaram um sistema denominado ArnetMiner que visa a mineração e extração de redes sociais acadêmicas. Mais especificamente, o ArnetMiner concentra-se na: 1) extração automática de perfis de pesquisadores na web; 2) integração de dados de publicações provenientes de bibliotecas digitais existentes na rede; 3) modelagem da rede; e 4) fornecimento de mecanismos de pesquisa, incluindo um mecanismo para a recomendação de especialistas.

O ArnetMiner é dividido em cinco componentes principais:

- extração: componente que tem o objetivo de extrair automaticamente perfis de pesquisadores da web. Primeiramente, esse componente extrai publicações de materiais em bibliotecas digitais, e, em seguida, utiliza uma abordagem unificada para extrair as propriedades do perfil identificadas no documento. Por exemplo, considerando a Biblioteca Digital ACM como o repositor de documentos web, o componente de extração tem como objetivo processar as páginas que possuem documentos científicos e realizar a extração de dados como o nome do autor, título, ano e local de publicação do documento, entre outras;
- integração: componente que realiza a integração entre o perfil dos pesquisadores com as suas publicações extraídas, utilizando como elo de ligação o nome do pesquisador. Os dados dos documentos processados na etapa anterior são relacionados com o autor do documento, que pode ou não estar cadastrado no sistema. Posteriormente, esses dados integrados são armazenados em uma base de conhecimento de rede de pesquisadores (*researcher network knowledge base - RNKB*);

- armazenamento e acesso: componente que realiza o armazenamento dos dados extraídos e integrados no RNKB. Os dados do pesquisador são armazenados normalmente no banco de dados MySQL, já os índices dos documentos, antes de serem armazenados, passam por um processamento que utiliza o método de arquivo invertido (*idf*);
- modelagem: este componente utiliza um gerador probabilístico para modelar simultaneamente diferentes tipos de informação, estimando a distribuição dos tópicos para cada tipo de informação com o intuito de classifica-lo de acordo a sua natureza;
- serviços de pesquisa: baseados no resultado da modelagem, esse componente fornece alguns serviços de pesquisas, como a busca de especialistas. Esse componente também disponibiliza outros recursos, como a obtenção da principal área de pesquisa de um autor, ou ainda, sugestões acadêmicas de documentos, como uma espécie de sugestão de citações.

O desafio referente à implementação do ArnetMiner foi a extração de informação, pois as bases digitais não possuem um padrão, sendo necessária adaptações na estrutura desse componente, o que não deixa claro se tais métodos são adaptáveis o suficiente para toda a web. Outro problema pertinente a esse sistema é relacionado à desambiguação dos nomes dos autores, pois como a associação do documento com o autor é feito utilizando o nome do autor, e há casos de pessoas que podem ter o mesmo nome, pode haver uma associação incorreta entre um material e o seu autor, ou ainda, serem gerados diferentes perfis para um mesmo autor, caso seu nome seja representado de forma diferente em cada documento da base.

Após a realização dos estudos sobre redes sociais e sistemas de recomendação de especialistas, foi elaborada uma metodologia para o desenvolvimento do módulo proposto para a rede social konnen. Esta metodologia é apresentada na seção 2.3.

3. MATERIAIS E MÉTODOS

Nesta seção serão apresentados os materiais e métodos empregados no desenvolvimento do trabalho, quais foram os ativos de hardware e software utilizados, período, local e a forma de trabalho adotada.

3.1 Local e período

O desenvolvimento deste trabalho foi compreendido entre o primeiro e segundo semestres do ano de 2012. No primeiro semestre, buscou-se realizar pesquisas para o entendimento de todos os elementos envolvidos no domínio: Redes Sociais e Sistemas de Recomendação de Especialistas.

3.2 Materiais

Os materiais utilizados podem dividem-se em dois grupos: software e fontes bibliográficas.

3.2.1 Software

Os softwares utilizados durante o desenvolvimento do projeto foram:

- Para a confecção das imagens:
  - Fireworks CS5 11.0
- Editores de código
  - NetBeans IDE 7.1.2, versão destinada ao desenvolvimento em PHP;
  - Notepad++ v6.1.3
- Gerenciamento do banco de dados
  - MySQL Workcench 5.2.44 CE
  - dbForge Studio Express for MySQL,
- Frameworks
  - Kohana (http://kohanaframework.org/);
  - Trilado Framework (http://www.triladophp.org/);
- Para o desenvolvimento textual e de apresentações
  - Microsoft Word 2010;
  - Microsoft Power Point 2010;

3.2.3 Fontes de dados

Para o desenvolvimento do referencial teórico foram utilizadas teses, dissertações, livros e artigos científicos. As teses, dissertações e artigos foram encontrados na internet, enquanto que os livros utilizados foram obtidos com o orientador.

3.3 Metodologia

O desenvolvimento do presente trabalho foi dividido em fases, sendo:

- fase 1: nesta fase foi necessário o entendimento do contexto. Pesquisas foram realizadas em artigos, dissertações, teses e livros até que se tivesse uma ideia sobre os principais tópicos que seriam abordados na revisão de literatura;
- fase 2: a partir do entendimento do contexto, foi desenvolvido o projeto de TCC, onde foram representados os objetivos gerais e específicos, a justificativa, o problema, as hipóteses, a revisão de literatura e o cronograma;
- fase 3: desenvolvimento do modelo de identificação e mensuração das especialidades dos usuários, que foi constituído pelas etapas:
    - identificação de fontes de dados externas que seriam utilizadas como entradas no modelo de identificação e mensuração de especialistas. Nesse passo, definiu-se que seriam utilizadas as informações do LinkedIn, da Plataforma Lattes, do Bibsonomy e as informações da própria konnen;
    - definição dos estados que as habilidades assumirão. Para isso, utilizou-se como base o trabalho de Fazel-Zarandi e Fox (2011), em que uma habilidade poderia ter o estado comprovada, desconhecida, sugerida e refutada. Além disso, foi explorada também uma categoria de habilidade diferente das categorias exploradas em Fazel-Zarandi e Fox (2011): a habilidade inferida.
    - definição da importância de cada estado de uma habilidade. Essa importância foi definida com a atribuição de pesos, baseados no nível de certeza que cada estado de habilidade fornecia sobre o conhecimento real do usuário;
    - definição da importância de cada tipo de publicação utilizada no cálculo da habilidade demonstrada;
- fase 4: escolha das ferramentas de desenvolvimento. Nesta etapa, optou-se por utilizar o NetBeans como IDE, pela afinidade possuída com tal ferramenta;
- fase 5: implementação do modelo. Neste momento foram definidas e implementadas as classes que continham a lógica definida no modelo de identificação e mensuração

das habilidades. A cada novo método implementado, eram realizados testes isolados, simulando dados de entrada e comparando as saídas fornecidas com os resultados esperados, sempre observando-se o passo-a-passo da execução. Os resultados esperados que foram comparados com as saídas fornecidas pelos métodos foram obtidos através de cálculos manuais, que foram conferidos e validados;

- fase 6: escrita sobre o processo de implementação. Este momento foi destinado a descrever formalmente o modelo proposto e a sua implementação.

Durante cada uma dessas fases, ocorreram reuniões com o orientador professor Mestre Edeilson Milhomem Silva e terceiros, que faziam sugestões sobre o que estava sendo desenvolvido.

4. RESULTADOS E DISCUSSÃO

Esta seção apresentará os resultados obtidos como desenvolvimento do trabalho, mostrando o modelo desenvolvido para a identificação e mensuração de habilidades dos usuários e como ocorreu a integração com o ambiente da rede de gestão de conhecimento konnen.

4.1 Modelo de recomendação de especialistas desenvolvido

No trabalho de Fael-Zarandi e Fox (2011) é considerado que cada habilidade de um usuário possui um estado. Os estados de uma habilidade determinados por esses pesquisadores são: demonstrada, sugerida, desconhecida e refutada. Cada estado de uma habilidade se baseia no nível de evidência que se tem sobre ela e como elas evoluem com o decorrer do tempo. Dessa forma, uma habilidade associada a um usuário em um determinado momento possui um estado único, ou seja, ela é uma habilidade demonstrada ou sugerida ou desconhecida ou refutada.

Para o trabalho em questão, considerou-se, além dos estados definidos por Fael-Zarandi e Fox (2011), a existência do estado "habilidade inferida". Segundo o modelo proposto, uma determinada habilidade pode assumir o estado "demonstrada" e/ou "sugerida" e/ou "desconhecida" e/ou "refutada" ou "inferida", ou seja, ela pode, ao mesmo tempo, possuir valores para os quatro primeiros estados, mas, uma vez que ela possuir valor para qualquer um deles, ela não terá um valor para o estado "inferida". Optou-se pela ocorrência simultânea dos estados "demonstrada", "desconhecida", "sugerida" e "refutada" para valorizar e aproveitar por completo as informações presentes no ambiente e mais fatores serem considerados no momento do cálculo de uma habilidade, o que pode resultar em um maior número de valores distintos, uma vez que cada habilidade terá um nível específico para cada estado e usuário. O status de habilidade inferida é utilizado para diferenciar as habilidades do usuário que foram identificadas pelo sistema (explicada na seção 4.1.1.5) das habilidades informadas ou obtidas através da consulta de informações do usuário.

Sendo assim, foram definidos pesos para cada estado de habilidade, de acordo com o nível de certeza que cada estado de habilidade fornecia sobre o conhecimento real do usuário. Esses pesos demonstram o valor que cada estado de uma habilidade possui perante o sistema. A Tabela 5 mostra todos os estados de uma habilidade e os seus respectivos pesos.

Tabela 5: Pesos das habilidades consideradas no sistema

Habilidade (h)	Peso (w_h)
Demonstrada	5 $(w_{h_{dem}})$
Desconhecida	3 $(w_{h_{des}})$
Sugerida	1 $(w_{h_{sug}})$
Refutada	-1 $(w_{h_{ref}})$
Inferida	--

Na Tabela 5 os pesos estão compreendidos na escala de -1 a 5, onde, quanto mais perto de -1, menos importante é o estado da habilidade, e, quanto mais próximo de 5, mais importante ela será. A habilidade demonstrada é aquela que fornece evidências concretas sobre a habilidade de um usuário, por isso, ela possui o peso 5, maior peso da escala. A habilidade desconhecida é obtida do próprio usuário, que possui ciência sobre suas habilidades, evidência considerada menos importante que a anterior, e, por isso, assume o peso 3. A habilidade sugerida é aquela que é obtida através das indicações que um usuário recebe como especialista em determinada área, por isso ela assume o peso 1, por basear-se apenas em indicações de terceiros, não fornecendo evidências concretas sobre a habilidade do usuário. A habilidade refutada tem o papel de mensurar o quanto uma habilidade do usuário não é útil para ajudar os demais usuários em suas dúvidas, por isso ela assume o valor -1, que decrementará o grau de habilidade final do usuário. Por fim, a habilidade inferida não possui peso, pois é uma sugestão realizada pelo sistema, podendo assumir o estado de habilidade desconhecida ou habilidade sugerida.

O processo de inferência de cada tipo de especialidade é apresentado na seção 4.1.1, onde são apresentadas e explicadas as equações envolvidas nesse processo.

4.1.1 Formalização

Esta seção apresenta quais são os elementos considerados para a inferência de cada tipo de habilidade e como esses elementos se relacionam. Serão apresentadas as formalizações da habilidade demonstrada, habilidade desconhecida, habilidade sugerida, habilidade refutada, habilidade inferida, e, por fim, como elas se relacionam para se chegar a um grau final de habilidade de um usuário em uma área.

4.1.1.1 Habilidade demonstrada

Na presente proposta, os trabalhos científicos publicados pelo usuário foram adotados como fonte da sua habilidade demonstrada. Optou-se por utilizar esse tipo de informação porque antes de ser publicado o trabalho passa pela avaliação de uma banca especializada, que julga se o usuário possui conhecimento sobre o tema abordado, o que foi considerado como evidência concreta da sua especialidade na área abordada pelos seus trabalhos.

Os trabalhos científicos publicados dos usuários devem ser agrupados por tipo, de forma que cada um dos tipos possíveis possui características próprias, que definem a sua importância. Sendo assim, para o modelo de inferência de especialistas proposto, foram definidos três tipos de trabalhos e seus respectivos pesos, que determinam a sua importância. Esses tipos de publicações são apresentados na Tabela 6.

Tabela 6: Pesos das publicações científicas consideradas no sistema

Tipo de publicação (t)	Peso (w_t)
Artigos completos publicados em periódicos/Capítulos de livros publicados	5 $(w_{t_{accl}})$
Trabalhos completos publicados em anais de congressos	2 $(w_{t_{tc}})$
Resumos expandidos publicados em anais de congressos	1 $(w_{t_{re}})$

Os pesos apresentados na Tabela 6 estão compreendidos no intervalo de 1 a 5, onde, quanto mais próximo de 1, menos importante é o tipo de publicação, e, quanto mais próximo de 5, mais importante ele é. O nível de importância de um tipo de publicação foi determinado empiricamente considerando a abrangência e reconhecimento da entidade que o publicou, sendo:

- Artigos completos publicados em periódico e capítulos de livros publicados exigem que o autor possua um sólido conhecimento, a fim de fornecer um maior detalhamento sobre o tema abordado. Além disso, os trabalhos publicados em forma de periódicos e livros possuem maior reconhecimento perante a comunidade acadêmica. Sendo assim, a associação desses fatores justifica a atribuição do peso 5, maior dentre todos, para esses tipos de trabalho;
- Os trabalhos completos publicados em anais de congresso também exigem um conhecimento aprofundado do autor sobre o tema abordado, porém, não possuem tanta abrangência como os trabalhos citados anteriormente. Sendo assim, o peso associado a esse tipo de publicação recebe o valor 3;

- Por fim, resumos expandidos publicados em anais de congresso não apresentam os conceitos de forma tão detalhada como os outros tipos de trabalho, o que exige menos conhecimento do autor. Dessa forma, esse tipo de publicação recebe o peso 1, que é o menor dos valores.

A partir desses pesos, foram definidos dois passos para o cálculo da habilidade demonstrada de um usuário em uma determinada área. No primeiro passo, representado pela Equação 5, é calculada a habilidade demonstrada do usuário em uma área através de suas publicações, e, no segundo passo, demonstrado na Equação 6, além da habilidade demonstrada de um usuário em uma área, é considerada a maior habilidade demonstrada dentro do ambiente para a mesma área.

A Equação 5, responsável pelo processamento inicial da habilidade demonstrada, é:

$$hdem(u,h) = \frac{1}{w_{t_{accl}} + w_{t_{tc}} + w_{t_{re}}} \times \sum_{\forall t_{u,h} \in T_{u,h}} \frac{w_t \times \sum_{\forall p_t \in t_{u,h}} |a_{p_t}|^{-1}}{|t_h|} \quad (5)$$

Onde:

- $hdem(u,h)$: valor da habilidade demonstrada de um usuário *u* em uma área *h*;
- $w_{t_{accl}}$: peso atribuído às publicações do tipo "artigos completos publicados em periódicos/Capítulos de livros publicados";
- $w_{t_{tc}}$: peso atribuído às publicações do tipo "Trabalhos completos publicados em anais de congressos";
- $w_{t_{re}}$: peso atribuído às publicações do tipo "Resumos expandidos publicados em anais de congressos";
- $T_{u,h}$: conjunto de publicações de um usuário *u* em uma área *h*, separadas por tipo. Os tipos existentes são os apresentados na Tabela 6;
- $t_{u,h}$: subconjunto de $T_{u,h}$, que agrupa os trabalhos por tipo;
- w_t: peso associado à publicação do tipo $t_{u,h}$, na Tabela 6;
- p_t: uma publicação do subconjunto $t_{u,h}$;
- $|a_{p_t}|$: quantidade de autores da publicação p_t;
- $|t_h|$: quantidade geral de publicações do tipo t na área *h*.

A Figura 5 apresenta o fluxograma do algoritmo da Equação 5.

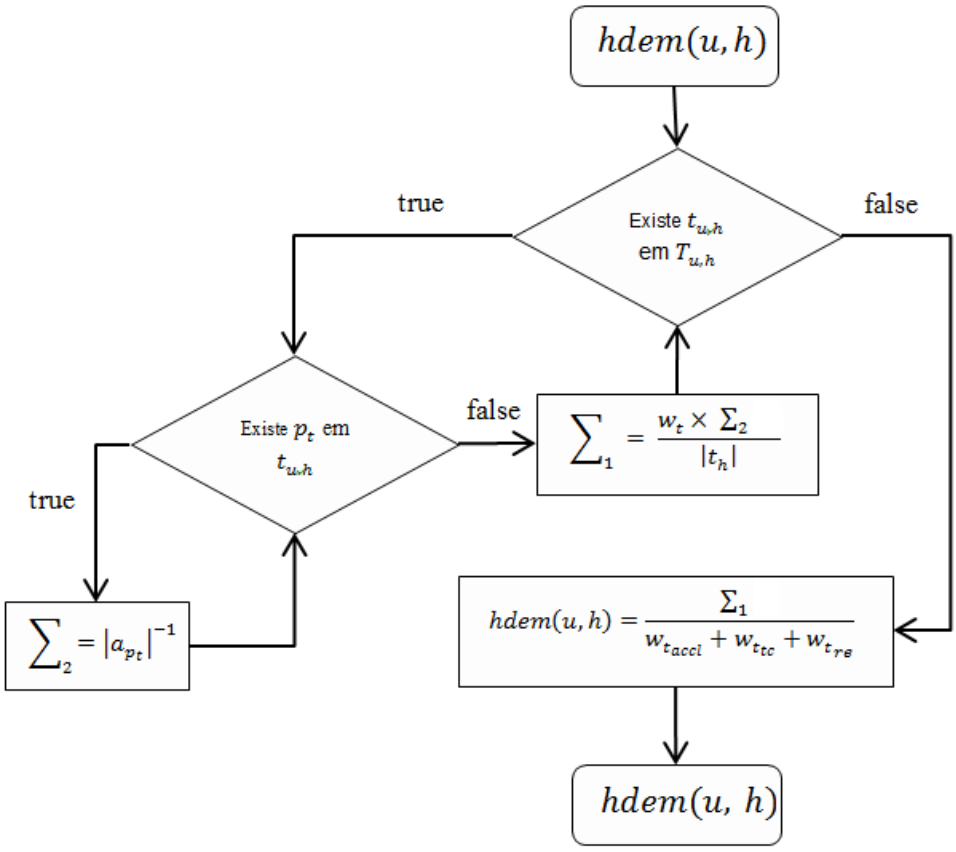


Figura 5: Fluxograma de processamento da habilidade demonstrada

Como mostrado na Figura 5, o processo para calcular a habilidade demonstrada parcial de um usuário é dividido em algumas etapas, sendo:

- Recuperação de todas as publicações do usuário u que possuam a palavra-chave h, que representa uma área de habilidade. Essas publicações são agrupadas em quatro listas $t_{u,h}$, de acordo com o seu tipo, apresentado na Tabela 6. Posteriormente, essas quatro listas são agrupadas em uma única lista, chamada $T_{u,h}$;
- Para cada publicação p_t de uma lista $t_{u,h}$ de $T_{u,h}$, é necessário estimar o quanto o usuário possui domínio sobre o assunto, pois geralmente um trabalho possui mais que um autor, onde cada um é responsável por uma parte abordada no trabalho, o que resulta em um conhecimento não uniforme do todo entre os autores. Como o presente modelo não tem por objetivo identificar qual a parte do trabalho que um usuário é responsável, para estimar o quanto ele sabe sobre o tema abordado, realiza-se o cálculo da média ponderada considerando a quantidade de autores, armazenando o resultado em um acumulador $\sum_2$;
- Ao processar todas as publicações p_t de uma lista $t_{u,h}$, é calculada a habilidade demonstrada parcial do usuário através da multiplicação do acumulador $\sum_2$ pela pelo

peso w_t associado as publicações do tipo $t_{u,h}$, dividindo-se esse resultado pela quantidade de elementos da lista t_h e o armazenando no acumulador $\sum_1$;

- Ao processar todas as listas $t_{u,h}$, realiza-se a divisão do acumulador $\sum_1$ pela soma dos pesos de cada tipo de publicação, apresentados na Tabela 6.

Quanto mais usuários e publicações existirem no ambiente onde o SRE proposto for implantado, mais próximos de 0 os resultados da Equação 5 serão. Foi constatado que essa proximidade de 0 poderia ser evitada para que a habilidade demonstrada não perca a sua representatividade em relação aos demais tipos de habilidades. Esse passo adicional definido para o cálculo da habilidade demonstrada é representado pela Equação 6.

$$hdem(u,h) = \left(\frac{hdem(u,h)}{max(hdem(h))}\right) \times w_{h_{dem}} \quad (6)$$

Onde:

- $hdem(u,h)$: valor da habilidade demonstrada de um usuário *u* em uma área *h*;
- $max(hdem(h))$: maior valor existente de uma habilidade demonstrada na área *h*;
- $w_{h_{dem}}$: peso da habilidade demonstrada, apresentado na Tabela 5.

A aplicação da Equação 6 ocorre somente após todas as habilidades de todos os usuários serem calculadas através da Equação 5, pois é necessário conhecer, antes de realizar o cálculo final da habilidade demonstrada de um usuário em uma área, qual o valor da maior habilidade demostrada na área em questão. Dessa forma, a aplicação da Equação 6 ocorre através da divisão da habilidade demonstrada do usuário u na área h pelo maior valor da habilidade demonstrada na área, o que resulta em um valor compreendido no intervalo de "0" a "1", que deve ser multiplicado pelo peso da habilidade demonstrada, apresentada na Tabela 5.

Através da aplicação das Equações 5 e 6, se tem, para cada usuário, o valor de sua habilidade demonstrada em uma área. Quanto menor for esse resultado, menor será o grau da habilidade demonstrada do usuário, e, quanto maior for o resultado, maior será o grau da habilidade.

A seção 4.1.1.2 mostra como ocorre o cálculo da habilidade desconhecida de um usuário.

4.1.1.2 Habilidade desconhecida

A habilidade desconhecida consiste em uma habilidade que o usuário afirma ter. Dessa forma, a equação responsável por estimar o valor da habilidade desconhecida de um usuário se baseia em um condicional, apresentado na Equação 7.

$$hdes(u,h) = \begin{cases} w_{h_{des}}, & se\ h \in H_u \\ 0, & caso\ contrário \end{cases} \quad (7)$$

Onde:

- $hdes(u,h)$: valor da habilidade desconhecida de um usuário u em uma área h;
- $w_{h_{des}}$: peso da habilidade desconhecida, apresentado na Tabela 5**Erro! Fonte de referência não encontrada.**;
- H_u: conjunto de habilidades que o usuário afirma possuir.

Sendo assim, o processamento para se chegar ao grau de especialidade desconhecida de um usuário verifica se ele afirmou possuir a habilidade em evidência, atribuindo para esse tipo de habilidade do usuário, em caso verdadeiro, o peso referente à habilidade desconhecida, apresentado na Tabela 5, e, em caso falso, atribuindo o valor "0". Dessa forma, os resultados obtidos para esse tipo de habilidade serão 0 ou 3, onde 0 indica a ausência da habilidade desconhecida do usuário na área em questão e 3 indica a presença da habilidade.

A seção 4.1.1.3 mostra como é realizado o cálculo da habilidade sugerida.

4.1.1.3 Habilidade sugerida

A habilidade sugerida é uma habilidade calculada a partir das sugestões que os usuários realizam na rede. Essas sugestões são indicações de determinado usuário como especialista em uma área. Para cada recomendação de especialidade que um especialista recebe, são consideradas a maturidade e reputação do autor da indicação. Dessa forma, quanto mais indicações de uma mesma habilidade um usuário receber, maior será o seu nível de habilidade sugerida.

Para realizar o cálculo da habilidade sugerida para um usuário, são necessários dois passos. O primeiro passo, representado pela Equação 8, considera apenas a reputação e maturidade dos autores das indicações, enquanto que o segundo passo, representado pela Equação 9, considera o usuário que possui a maior habilidade sugerida na área em evidência.

A Equação 8, responsável pelo cálculo inicial da habilidade sugerida de um usuário é:

$$hsug(u,h) = \sum_{\forall u \in U} u_r + u_m \quad (8)$$

Onde se têm as variáveis:

- $hsug(u,h)$: valor da habilidade sugerida de um usuário u em uma área h;
- u_r: valor da reputação de um usuário, que é normalizado no intervalo de 0 a 1;
- u_m: valor da maturidade de um usuário, que é normalizado no intervalo de 0 a 1.

Dessa forma, o processamento da Equação 8 se dá como apresentado na Figura 8.

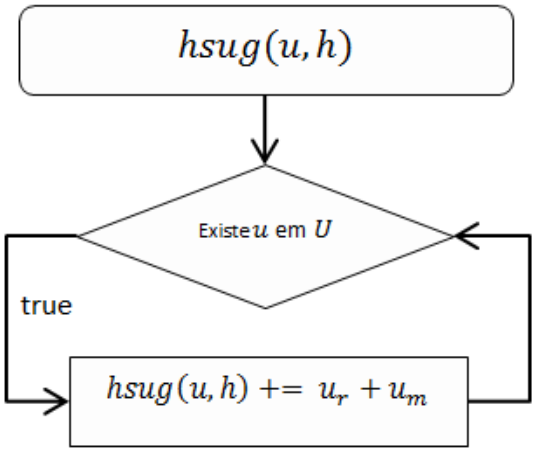


Figura 6: Fluxograma de processamento da habilidade sugerida

Segundo o fluxograma apresentado na Figura 6, o processo para o cálculo da habilidade sugerida parcial de um usuário é dividido em alguns passos, sendo:

- Inicialmente são recuperados e armazenados em uma lista todos os usuários que indicaram o u como especialista na área h;
- O valor da habilidade sugerida do usuário é incrementado pelo produto da reputação e maturidade de cada usuário da lista enquanto houver usuários na lista de usuários;

Através da aplicação da Equação 8, cada habilidade sugerida de um usuário possuirá um valor não normalizado. Para que esse valor seja normalizado no intervalo de 0 a 1, foi necessário definir um passo adicional, representado pela Equação 9.

$$hsug(u,h) = \left(\frac{hsug(u,h)}{max(hsug(h))}\right) \times w_{h_{sug}} \quad (9)$$

Onde se têm as variáveis:

- $hsug(u,h)$: valor da habilidade sugerida de um usuário u em uma área h;
- $max(hsug(h))$: maior valor existente de uma habilidade sugerida na área h;

- $w_{h_{sug}}$: peso da habilidade sugerida, apresentado na Tabela 5**Erro! Fonte de referência não encontrada.**.

A aplicação da Equação 9 ocorre somente após todas as habilidades de todos os usuários serem calculadas através da Equação 8, pois é necessário conhecer, antes de realizar o cálculo final da habilidade sugerida de um usuário em uma área, qual o valor da maior habilidade sugerida na área em questão. Dessa forma, a aplicação da Equação 9 ocorre através da divisão da habilidade sugerida do usuário u na área h pelo maior valor da habilidade sugerida na área, o que resulta em um valor compreendido no intervalo de "0" a "1", que deve ser multiplicado pelo peso da habilidade sugerida, apresentada na Tabela 5.

Através da aplicação das Equações 8 e 9, se tem, para cada usuário, o valor de sua habilidade sugerida em uma área. Quanto menor for esse resultado, menor será o grau da habilidade sugerida do usuário, e, quanto maior for o resultado, maior será o grau da habilidade.

A seção 4.1.1.4 mostra como ocorre o cálculo da habilidade refutada de um usuário.

4.1.1.4 Habilidade refutada

Como a habilidade sugerida, a habilidade refutada também é calculada a partir das sugestões que os usuários realizam na rede. Porém, essas sugestões indicam que um determinado usuário não possui conhecimento suficiente em uma área onde ele foi definido como especialista. Da mesma forma, a reputação e maturidade do autor da sugestão são considerados em uma primeira equação, apresentada na Equação 10, que resulta em um valor não normalizado. O segundo passo, representado pela Equação 11, considera o valor da maior habilidade refutada na área em evidência, com o objetivo de normalizar o resultado.

A Equação 10 é:

$$href(u,h) = \sum_{\forall u \in U} u_r + u_m \tag{6}$$

Onde se têm as variáveis:

- $href(u,h)$: valor da habilidade refutada de um usuário u em uma área h;
- u_r: valor da reputação de um usuário, que é normalizado no intervalo de 0 a 1;
- u_m: valor da maturidade de um usuário, que é normalizado no intervalo de 0 a 1.

Dessa forma, o processamento da Equação 6 se dá como apresentado na Figura 7**Erro! Fonte de referência não encontrada.**:

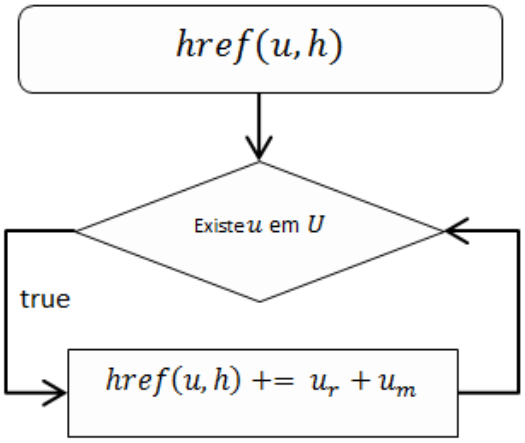


Figura 7: Fluxograma de processamento da habilidade refutada

Segundo o fluxograma apresentado na Figura 7, o processo para o cálculo da habilidade refutada parcial de um usuário é dividido em alguns passos, sendo:

- Inicialmente são recuperados e armazenados em uma lista todos os usuários que refutaram u como especialista na área h;
- O valor da habilidade refutada do usuário é incrementado pelo produto da reputação e da maturidade de cada usuário enquanto houver usuários na lista de usuários

Através da aplicação da Equação 10, cada habilidade refutada de um usuário possuirá um valor não normalizado. Para que esse valor seja normalizado no intervalo de 0 a 1, foi necessário definir um passo adicional, representado pela Equação 11.

$$href(u,h) = \left(\frac{href(u,h)}{max\big(href(h)\big)}\right) \times w_{h_{ref}} \qquad (11)$$

Onde se têm as variáveis:

- $href(u,h)$: valor da habilidade refutada de um usuário u em uma área h;
- $max(href(h))$: maior valor existente de uma habilidade refutada na área h;
- $w_{h_{ref}}$: peso da habilidade sugerida, apresentado na Tabela 5**Erro! Fonte de referência não encontrada.**.

A aplicação da Equação 11 ocorre somente após todas as habilidades de todos os usuários serem calculadas através da Equação 10, pois é necessário conhecer, antes de realizar o cálculo final da habilidade refutada de um usuário em uma área, qual o valor da maior habilidade refutada na área em questão. Dessa forma, a aplicação da Equação 11 ocorre através da divisão da habilidade refutada do usuário u na área h pelo maior valor da habilidade sugerida na área, o que resulta em um valor compreendido no intervalo de "0" a "1", que deve ser multiplicado pelo peso da habilidade refutada, apresentada na Tabela 5.

Através da aplicação das Equações 10 e 11, se tem, para cada usuário, o valor de sua habilidade refutada em uma área. Quanto menor for esse resultado, menor será o grau da habilidade refutada do usuário, e, quanto maior for o resultado, maior será o grau da habilidade.

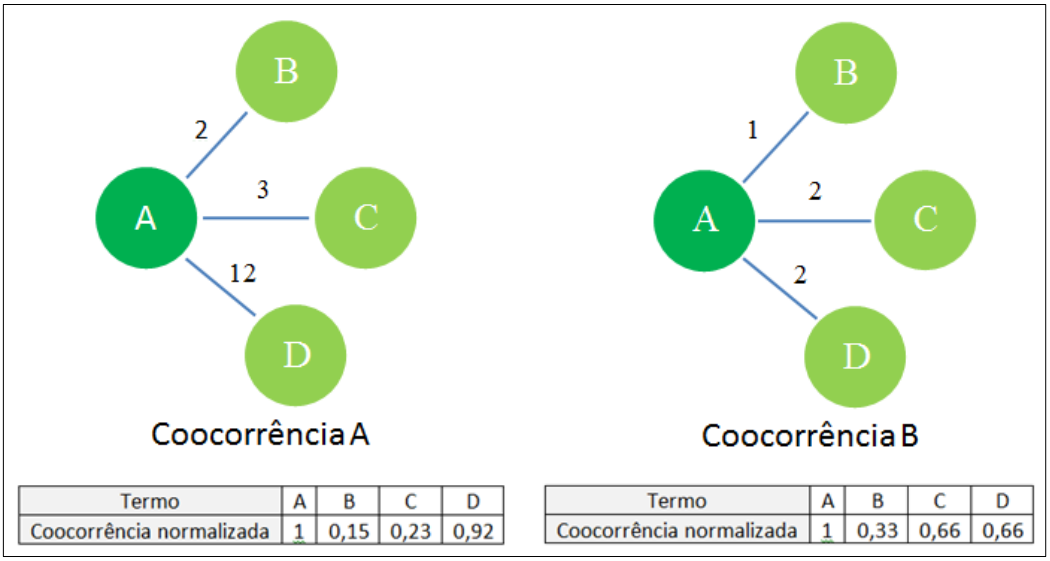
A seção 4.1.1.5 mostra como ocorre o cálculo da habilidade refutada de um usuário.

4.1.1.5 Habilidade inferida

A habilidade inferida é uma habilidade que o sistema sugere para um usuário. Esse tipo de habilidade não possui um peso associado por ser apenas uma recomendação, que, caso o usuário aceite, assume o *status* de habilidade desconhecida.

Para realizar essa inferência, são utilizados os conceitos de coocorrência e similaridade. Inicialmente, deve ser montado um grafo de coocorrência A entre os termos presentes no ambiente, e, a partir disso, deve ser montado, para cada usuário, outro grafo B, indicando a coocorrência entre os termos e o usuário. Dessa forma, para saber se um termo deve ser recomendado como uma habilidade de um usuário, deve-se evidenciar esse termo no grafo A e B. Através da evidência do termo em ambos os grafos, é possível montar um vetor com os demais termos que se relacionam com ele. Com essa representação de parte dos grafos em vetores, pode-se usar um coeficiente de similaridade, que apresenta um valor indicando o quanto esses vetores são similares. Dessa forma, ao calcular a similaridade entre os dois vetores, obtêm-se um valor que determinará se o termo que representa uma área de habilidade será sugerido para o usuário como uma de suas especialidades.

A Figura 8 mostra a coocorrência entre os termos “A”, “B”, “C” e “D”. A “coocorrência A” representa a coocorrência entre os termos no ambiente, enquanto que a “Coocorrência B” indica a coocorrência dos termos nas informações de um usuário.



Termo	A	B	C	D
Coocorrência normalizada	1	0,15	0,23	0,92

Termo	A	B	C	D
Coocorrência normalizada	1	0,33	0,66	0,66

Figura 8: Exemplo de grafos que representam a coocorrência entre os termos do ambiente e um usuário

Como mostrado na Figura 8, cada grafo de coocorrência possui um vetor com valores normalizados. Neste vetor, o termo em evidência possui o maior valor de coocorrência entre os elementos acrescido com o valor 1. A normalização da coocorrência é realizada pela divisão do valor de coocorrência do termo em relação ao termo em evidência pela divisão do valor de coocorrência do termo em evidência. Dessa forma, o resultado será um valor normalizado no intervalo de 0 a 1. A partir da normalização dos vetores, pode-se calcular o quanto eles se assemelham. Para isso, é utilizado o Coeficiente de Correlação de Pearson, apresentado na Equação 12.

$$sim(C_{t,u}, C_{t,r}) = \frac{\sum_{\forall a \in C_{t,u}} (a - \overline{C_{t,u}})(b - \overline{C_{t,r}})}{\sqrt{\sum_{\forall a \in C_{t,u}} (a - \overline{C_{t,u}})^2} \sqrt{\sum_{\forall b \in C_{t,r}} (b - \overline{C_{t,r}})^2}} \quad (12)$$

Onde se têm as variáveis:

- $C_{t,u}$: vetor de coocorrência do termo t em relação ao usuário u;
- $C_{t,r}$: vetor de coocorrência do termo t em relação à rede;
- a: elemento do conjunto $C_{t,u}$;
- $\overline{C_{t,u}}$: média dos elementos contidos no conjunto $C_{t,u}$;
- b: elemento do conjunto $C_{t,r}$;
- $\overline{C_{t,r}}$: média dos elementos contidos no conjunto $C_{t,r}$;

O resultado da Equação 12 é um número que varia no intervalo de -1 a 1, onde, quanto menor, menor é a similaridade entre os vetores, e, quanto maior, maior será a similaridade entre eles. Nesse caso, por exemplo, se o valor da similaridade for maior ou igual a 0,3 o

termo em evidência nos dois vetores deverá ser recomendado para o usuário como uma possível habilidade.

A seção 4.1.1.6 apresenta como os estados de uma habilidade se relacionam para se inferir o valor geral da habilidade de um usuário em uma área.

4.1.1.6 Habilidade geral

A partir dos cálculos das outras habilidades, é possível se estimar um valor final para a habilidade de um usuário. Esta habilidade é calculada a partir da Equação 13.

$$H(u,h) = \frac{hdem(u,h) + hdes(u,h) + hsug(u,h) + href(u,h)}{w_{h_{dem}} + w_{h_{des}} + w_{h_{sug}} + w_{h_{ref}}} \quad (13)$$

Onde se têm as variáveis:

- $H(u,h)$: valor final da habilidade de um usuário u em uma área h;
- $hdem(u,h)$: valor da habilidade demonstrada de um usuário u em uma área h;
- $hdes(u,h)$: valor da habilidade desconhecida de um usuário u em uma área h;
- $hsug(u,h)$: valor da habilidade sugerida de um usuário u em uma área h;
- $href(u,h)$: valor da habilidade refutada de um usuário u em uma área h;
- $w_{h_{dem}}$: peso da habilidade demonstrada;
- $w_{h_{des}}$: peso da habilidade desconhecida;
- $w_{h_{sug}}$: peso da habilidade sugerida;
- $w_{h_{ref}}$: peso da habilidade sugerida.

A partir desses valores, o processamento para se chegar ao grau final de especialidade desconhecida de um usuário, que é realizado somando-se os valores da habilidade demonstrada (seção 4.1.1.1), desconhecida (seção 4.1.1.2), sugerida (seção 4.1.1.3) e refutada (seção 4.1.1.4) e dividindo o resultado pela soma dos valores dos pesos de cada habilidade, apresentados na Tabela 5. O resultado desse cálculo será um valor normalizado no intervalo de 0 a 1, onde, quanto mais próximo de 0, menor é a habilidade geral do usuário, o que indica que ele poderá não ser uma boa fonte de informações, e, quanto mais próximo de 1, maior será a habilidade geral, confirmando o usuário como um especialista na área em questão.

A seção 4.1.2 apresenta um exemplo da implantação do Sistema de Recomendação de Especialistas proposto nessa seção.

4.1.2 Implementação

Para validar a proposta de sistema de recomendação de especialistas apresentado na seção 4.1.1, escolheu-se a plataforma de gestão de conhecimento konnen. Essa rede social tem por objetivo incentivar e otimizar a comunicação entre alunos, professores e demais funcionários do Centro Universitário Luterano de Palmas, CEULP/ULBRA. O fator determinante para a escolha dessa plataforma é que ela dispõe de todos os dados necessários para o cálculo de cada tipo de habilidade, apresentados abaixo. Apesar da escolha dessa plataforma, o modelo proposto pode ser aplicado a qualquer outro ambiente, desde que ele forneça as entradas requeridas para todos os cálculos realizados para a identificação e mensuração das habilidades dos usuários. A Figura 9 apresenta a arquitetura básica da rede social konnen, contendo seus principais módulos.

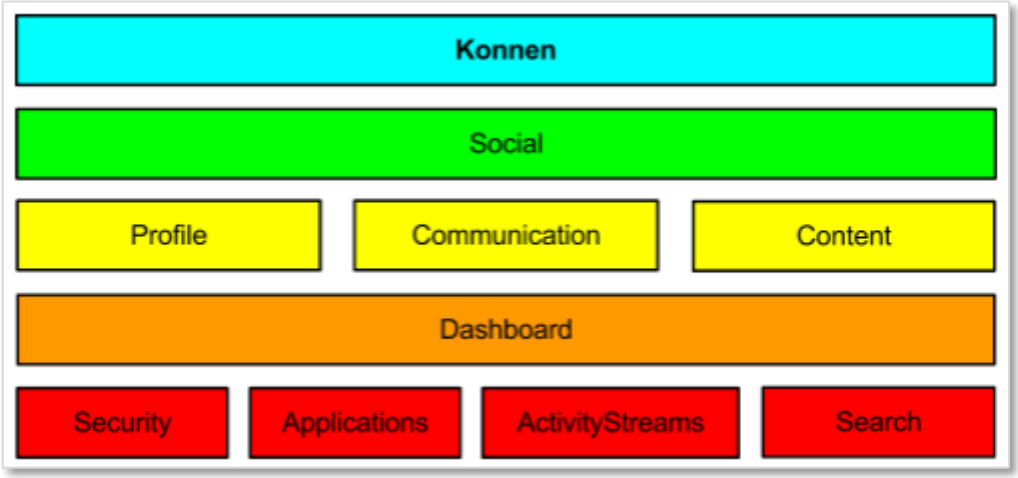


Figura 9: Arquitetura básica da rede social konnen (SOUZA, J. G. et. al.; 2012)

Na Figura 9 os aplicativos "*Security*", "*Applications*", "*ActivityStreams*" e "*Search*" são aplicativos de sistema, que fornecem informações para outros aplicativos. O aplicativo "*Dashboard*" é um aplicativo intermediário que fornece informações provenientes dos aplicativos de um usuário. "*Profile*", "*Communication*", "*Content*" e "*Social*" são aplicativos de usuário, ou seja, fornecem informações do usuário **(**SOUZA, J. G. et. al.; 2012**)**. Dentre esses aplicativos os módulos que fornecem dados para o cálculo das habilidades são:

- Habilidade demonstrada:
  - Gestão do perfil do usuário, realizada pelo módulo "*profile*", que dispõe das informações sobre as publicações científicas dos usuários;
- Habilidade desconhecida:
  - Gestão do perfil do usuário, realizada pelo módulo "*profile*", que possui informações fornecidas pelo próprio usuário sobre as suas especialidades;

- Habilidade sugerida:
    - Gestão de *tags*, realizada pelo módulo "*content*", que proporciona informações sobre as *tags* relacionadas aos usuários;
    - Gestão de discussões, realizada pelo módulo "*forum*", que fornece um ambiente onde dois usuários podem discutir sobre um assunto. Cada discussão, ao ser finalizada, é avaliada pelo usuário que a criou, onde ele pode sugerir o outro usuário participante da discussão como especialista em áreas relacionadas ao tema discutido;
    - Gestão de confiança de usuários, realizada pelo módulo "*tsweets*", que indica a confiabilidade de cada usuário presente na plataforma em relação aos outros usuários;
- Habilidade refutada:
    - Gestão de discussões, realizada pelo módulo "*fórum*", através das avaliações por parte do autor da discussão sobre a utilidade do conhecimento do outro usuário envolvido para a solução dos problemas discutidos;
    - Gestão de confiança de usuários, realizada pelo módulo "*tsweets*", que indica a confiabilidade de cada usuário presente na plataforma em relação aos outros usuários;
- Habilidade inferida:
    - Gestão de conteúdos, realizada pelo módulo "*content*", que permite a obtenção dos dados dos conteúdos compartilhados pelos usuários;
    - Gestão de *tags*, realizada pelo módulo "*content*", que proporciona informações sobre as *tags* relacionadas aos usuários;
    - Gestão do perfil do usuário, realizada pelo módulo "*profile*", que dispõe das informações sobre as publicações científicas dos usuários;
    - Gestão de mensagens, realizada pelo módulo "*message*", que comporta as mensagens enviadas pelos usuários;
    - Gestão de comentários, realizada pelo módulo "*comment*", que fornece os comentários realizados pelos usuários.

Para o processamento de todas as habilidades mencionadas, foi criado um método principal que realiza o controle das operações. Esse método é representado pelo pseudocódigo da Figura 10.

```
1  calcular_habilidades
2
3      H_DEMONSTRADA = 5
4      H_DESCONHECIDA = 3
5      H_SUGERIDA = 1
6      H_REFUTADA = -1
7
8      usuarios(): retorna todos os usuários
9
10     foreach(usuarios as u)
11         palavras_chave = get_palavras(u): retorna todas as
12                                           palavras relacionadas
13                                           ao usuário "u"
14             foreach(palavras_chave as pc)
15
16                 habilidade_demonstrada(u, pc)
17                 habilidade_desconhecida(u, pc)
18                 habilidade_sugerida(u, pc)
19                 habilidade_refutada(u, pc)
20                 habilidade_inferida(u, pc)
21
22     habilidades.all();
23
24     foreach(habilidades as h)
25
26         normalizar_habilidade_demonstada(h)
27         normalizar_habilidade_sugerida(h)
28         normalizar_habilidade_refutada(h)
29         habilidade_geral(h)
30
31     return void
```

Figura 10: Método para controlar o cálculo das habilidades dos usuários

Inicialmente (linhas 3 a 6), no pseudocódigo da Figura 10, são inicializadas as constantes que armazenam os valores de cada tipo de habilidade, apresentados na Tabela 5. Posteriormente (linha 8), todos os usuários da rede são recuperados e armazenados em uma lista. Para cada um dos usuários dessa lista é realizada (linha 11) uma busca que retorna todas as palavras relacionadas a ele. Para cada uma dessas palavras relacionadas ao usuário é calculada a habilidade demonstrada (linha 16), habilidade desconhecida (linha 17), habilidade sugerida (linha 18), habilidade refutada (linha 19) e habilidade inferida (linha 20). Após processar todas as palavras relacionadas a todos os usuários, é realizada a normalização de alguns valores. Para realizar essa normalização, recupera-se (linha 22) todas as habilidades existentes na rede, independentemente do usuário que as possui. Em seguida, para cada uma dessas habilidades é calculado o valor normalizado da habilidade demonstrada (linha 26), sugerida (linha 27), refutada (linha 28), e, em seguida, é calculada a habilidade geral

(linha29). Por fim (linha 31), o método retorna o valor `void`, indicando o fim do processamento das habilidades dos usuários.

As seções 4.1.2.1 a 4.1.2.6 detalham os métodos utilizados na FIGURA 14 para se obter o nível de especialidade dos usuários, informando as entradas, processamento e saída para cada um dos casos.

4.1.2.1 Habilidade demonstrada

O cálculo da habilidade demonstrada leva em consideração as informações sobre as publicações científicas do usuário, que são obtidas em seu currículo da Plataforma Lattes. O módulo responsável por consultar e armazenar os dados dessa plataforma é o módulo "*profile*", que tem como objetivo gerenciar as informações do perfil do usuário, fornecendo meios de integração com as redes sociais Facebook, LinkedIn e com a plataforma de currículos Lattes.

A importação das informações do Lattes é realizada através do processamento do código HTML da página do currículo do usuário, que é obtido utilizando a classe "*simple_html_dom*". Essa classe, desenvolvida utilizando PHP5+, é um analisador de documentos HTML que permite a fácil manipulação dos seus nós através do uso dos seletores jQuery (SIMPLEHTMLDOM, 2012, online).

Através da utilização dos métodos da classe "*simple_html_dom*", foi possível processar todo o conteúdo dos currículos Lattes dos usuários. O módulo "*profile*" foi implementado em um trabalho anterior de outro acadêmico, porém, quando se iniciou o desenvolvimento da proposta, verificou-se que a estrutura do código HTML dos currículos Lattes havia mudado. Como o código do módulo para a importação das informações do Lattes é baseado na interpretação da marcação HTML, a mudança realizada pelos desenvolvedores dessa plataforma resultou no não funcionamento do código desenvolvido anteriormente. Portanto, foi necessário reimplementar a parte responsável pela consulta e armazenamento das informações do Lattes, utilizando-se da mesma classe e estrutura de método e propriedades.

Ao reimplementar essa parte do módulo, verificou-se que a estrutura de armazenamento das palavras-chave das publicações deveria ser modificada para a normalização dos dados, o que facilitaria o desenvolvimento do restante do trabalho. Na Figura 11 é mostrada a estrutura dessa parte do banco de dados antes e depois das alterações.

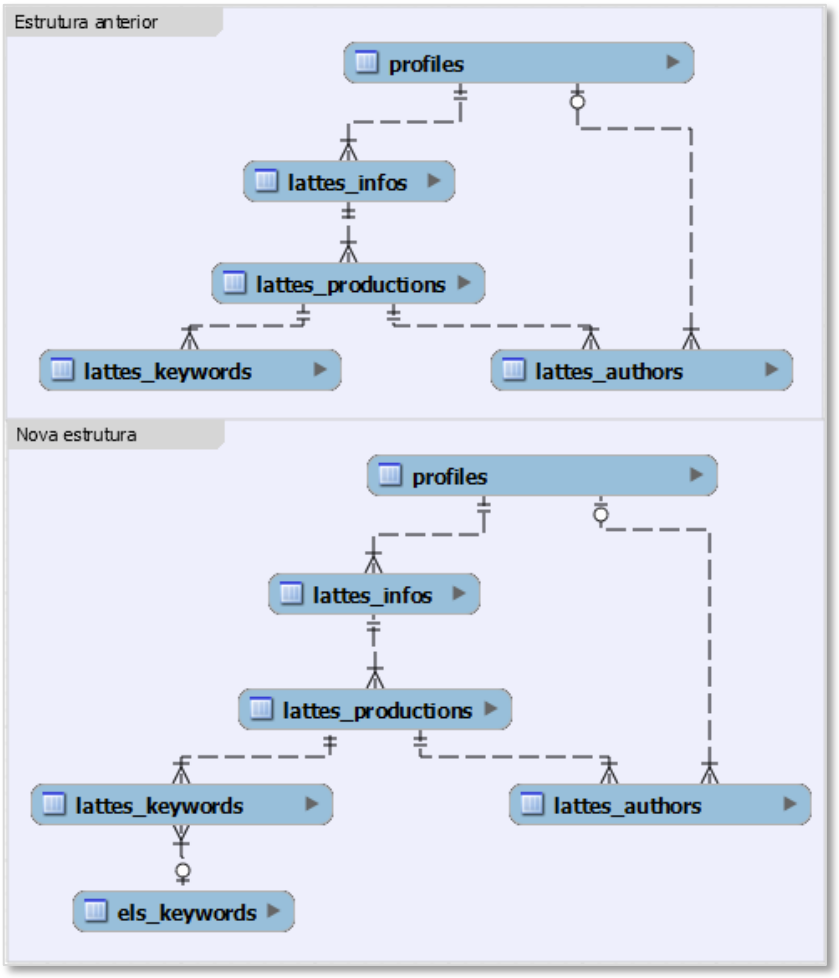


Figura 11: Comparação da estrutura de armazenamento das informações da Plataforma Lattes

Anteriormente, como mostrado na parte superior da Figura 11, a estrutura de armazenamento das publicações provenientes da Plataforma Lattes era dividida em cinco tabelas:

- `profiles`: tabela que armazena as informações do perfil do usuário;
- `lattes_infos`: tabela que armazena as informações do currículo Lattes do usuário. Essa tabela possui a propriedade para indicar o tipo da informação armazenada, que, no caso das publicações, sempre terá o valor "*production*";
- `lattes_productions`: tabela que armazena o título, ano, tipo e se a publicação é uma publicação relevante. A definição da publicação como relevante é feita pelo próprio usuário no momento que ele cadastra a publicação na Plataforma Lattes;
- `lattes_keywords`: tabela que armazena as palavras-chave da publicação;
- `lattes_authors`: tabela que armazena os dados dos autores da publicação;

Ainda na Figura 11, a parte inferior mostra a atual estrutura de armazenamento das informações do Lattes. Nesse novo modelo de representação ocorreram duas mudanças:

- a tabela `lattes_keywords` não é mais responsável por armazenar as palavras-chave da publicação, ela se tornou uma tabela de ligação entre a tabela `lattes_productions` e a tabela `els_keywords`;
- foi adicionada a tabela `els_keywords`, que é responsável por armazenar todas as palavras-chave presentes na konnen.

Essa mudança foi necessária porque as palavras-chave comuns a mais de uma publicação eram repetidas na `lattes_keywords`, visto que não havia um processo de verificação da existência dessas palavras nessa tabela, o que resultava em redundância de informação no banco de dados. Após as modificações, essa estrutura foi normalizada, e, dessa forma, as palavras-chave não são armazenadas a cada ocorrência em uma publicação, é realizado apenas o relacionamento entre a publicação e a palavra-chave existente através da `lattes_keywords`.

Após o processamento dos currículos Lattes de todos os usuários da rede, é possível realizar o cálculo da habilidade demonstrada. O código implementado para o cálculo dessa habilidade é baseado em duas etapas. A primeira etapa consiste na implementação da Equação 5 (seção 4.1.1.1). Esse código é representado através do pseudocódigo da Figura 12.

```
1  habilidade_demonstrada(usuario, habilidade):
2
3      tuh(usuario, habilidade): retorna todas as publicações do usuário
4                                na área "habilidade" separadas pelo tipo
5      usuario.hdemonstrada = 0;
6
7      foreach(tuh as t)
8          participacoes = 0;
9          if(t.count > 0)
10             foreach(t as pt)
11                 participacoes += (1 / pt.autores.count);
12             th(t.tipo, habilidade): retorna a quantidade de publicações
13                                     do tipo t na área "habilidade"
14             usuario.hdemonstrada += (t.peso * participacoes)/th.count;
15     usuario.hdemonstrada /= (wtaccl + wttc + wtre)
16
17     return usuario.hdemonstrada;
```

Figura 12: Pseudocódigo da Equação 5, responsável pela primeira etapa do cálculo da habilidade demonstrada

No pseudocódigo da Figura 12, recupera-se (linha 3) todas as publicações científicas do usuário na área informada, agrupando-as em sublistas, de acordo aos tipos apresentados na Tabela 6**Erro! Fonte de referência não encontrada.**. Para cada publicação de cada sublista, é calculado (linha 11) o quanto o usuário contribuiu para o desenvolvimento do trabalho, e, para isso, é armazenando o valor da divisão do número 1 pela quantidade de autores da

publicação, resultado armazenado na variável `participacoes`. Posteriormente (linha 14) o valor contido na variável `participacoes` é multiplicado pelo peso do tipo de publicação da sublista atual e dividido pela quantidade de elementos presentes na sublista de publicações atual, armazenando esse resultado na variável `usuario.hdemonstrada`. Ao processar todas as sublistas de tipos de publicações, a variável `usuario.hdemonstrada` terá um valor compreendido entre 0 e 8. O método apresentado precisa retornar um valor normalizado no intervalo de 0 a 1, que servirá de entrada para a segunda etapa do cálculo da habilidade demonstrada. Sendo assim, ao final (linha 15) o valor da variável `usuario.hdemonstrada` é dividido pelo somatório dos pesos dos tipos das publicações, apresentado na Tabela 6.

Após a executar a primeira etapa para todos os usuários da rede, é realizada a segunda etapa do cálculo da habilidade demonstrada, baseada na Equação 6. O pseudocódigo dessa etapa pode ser visualizado na Figura 13.

```
1  normalizar_habilidade_demonstrada(habilidade):
2
3      usuarios(habilidade): retorna todos os usuários que possuam
4                             publicações na área "habilidade"
5      maxHdem(habilidade): retorna a maior habilidade demonstrada
6                           na área "habilidade"
7
8      foreach(usuarios as u):
9          u.hdemonstrada = (u.hdemonstrada / maxHdem) * whdem;
10
11     return void;
```

Figura 13: Pseudocódigo da Equação 6, responsável pela segunda etapa do cálculo da habilidade demonstrada

Inicialmente (linha 3), no pseudocódigo da Figura 13, são recuperados todos os usuário que possuem um valor de habilidade demonstrada representado pelo parâmetro `habilidade`. Posteriormente (linha 5), é armazenado na variável `maxHdem` o maior valor da habilidade demonstrada presente entre os usuários da rede. Por fim (linha 9), o valor da habilidade demonstrada de cada usuário é dividido por `maxHdem` e multiplicado pelo peso da habilidade demonstrada, definido na Tabela 5.

Após a execução das duas etapas do cálculo da habilidade demonstrada, cada usuário que possua um relacionamento com a palavra que descreve a habilidade na área em questão terá um valor que indica o grau dessa habilidade. Esse valor é compreendido entre 0 e 5, onde, quanto mais próximo de 0, menor é a habilidade do usuário, e, quanto mais próximo de 5, maior ela será.

Na seção 4.1.2.2 será apresentada implementação do código responsável por calcular a habilidade desconhecida.

4.1.2.2 Habilidade desconhecida

O cálculo da habilidade desconhecida leva em consideração duas fontes: as informações do perfil do usuário na rede LinkedIn e as indicações de habilidade que o sistema realizou para o usuário. Em ambos os casos, as informações são obtidas através do módulo “profile”.

A importação dos dados do LinkedIn foi realizada em um trabalho anterior. No entanto, foi preciso realizar uma adaptação na estrutura de armazenamento das informações referentes à seção “Competências e especialidades”, que são as consideradas neste cálculo. Essa alteração na estrutura do banco de dados ocasionou uma mudança em parte do código responsável por essa importação. Na Figura 14 é mostrada a estrutura dessa parte do banco de dados antes e depois das alterações.

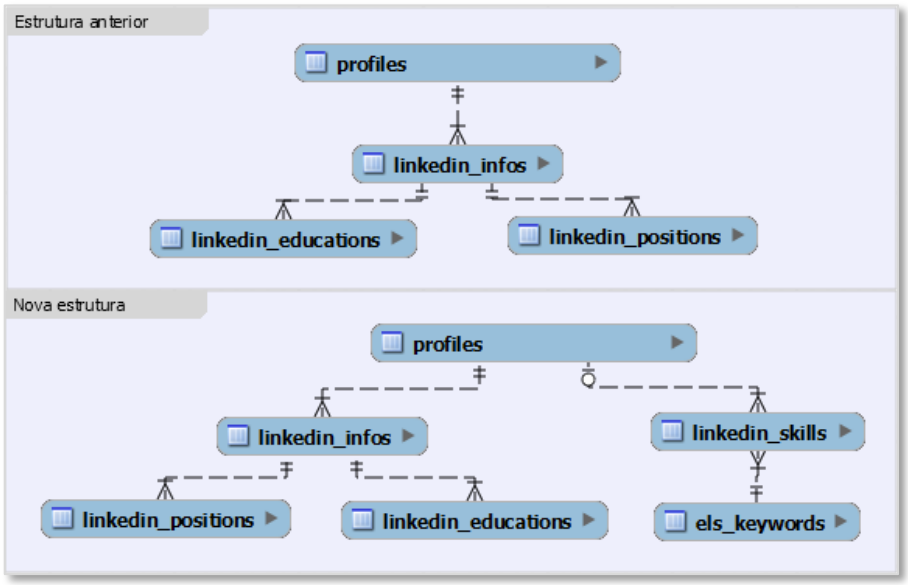


Figura 14: Comparação da estrutura de armazenamento das informações do LinkedIn

Anteriormente, como mostrado na parte superior da Figura 14, a estrutura de armazenamento das habilidades demonstradas do usuário era dividida em quatro tabelas:

- `profiles`: tabela que armazena as informações do perfil do usuário;
- `linkedin_infos`: tabela que armazena as informações do perfil do LinkedIn do usuário. Essa tabela possui uma propriedade para indicar o tipo da informação armazenada, que, no caso das “Competências e especialidades”, sempre terá o valor “*skill*”;

- `linkedin_educations`: tabela que armazena informações sobre a educação do usuário;
- `linkedin_positions`: tabela que armazena as experiências profissionais do usuário.

Ainda na Figura 14, a parte inferior mostra a atual estrutura de armazenamento das informações do Linkedin. Nesse novo modelo de representação ocorreram duas mudanças:

- a tabela `linkedin_infos` não armazena mais as informações do tipo "Competências e especialidades";
- foi adicionada a tabela `els_keywords`, que é responsável por armazenar todas as palavras-chave presentes na konnen;
- foi adicionada a tabela `linkedin_skills`, que é responsável por realizar a ligação entre as tabelas `profiles` e `els_keywords`;

Anteriormente, as informações do tipo "Competências e especialidades" do usuário eram armazenadas na tabela `linkedin_infos`, visto que não havia um processo de verificação da existência dessas palavras na referida tabela. Após as modificações, essa estrutura foi normalizada, e, dessa forma, as competências e especialidades não são armazenadas a cada ocorrência da área para um usuário, é realizado apenas o relacionamento entre o usuário e as palavras-chave que representam as suas competências.

A outra entrada para o cálculo da habilidade desconhecida são as confirmações das sugestões de habilidades que a rede sugere para cada usuário. Essa recomendação de especialidades é realizada com base no cálculo da habilidade inferida, abordado na 4.1.1.5.

O código implementado para o cálculo da habilidade desconhecida é baseado na Equação 7, representado através do pseudocódigo da Figura 15.

```
1  habilidade_desconhecida(usuario, habilidade):
2
3      if(hDesconhecida(usuario, habilidade).count > 0):
4          usuario.hdesconhecida = whdes;
5      else
6          usuario.hdesconhecida = 0
7
8      return usuario.hdesconhecida;
```

Figura 15: Pseudocódigo da Equação 2, responsável pela segunda etapa do cálculo da habilidade desconhecida

No pseudocódigo da Figura 15, verifica-se (linha 3) se o usuário possui algum vínculo com a habilidade informada pelo parâmetro `habilidade`. Caso exista um vínculo entre a

habilidade e o usuário, o valor da habilidade desconhecida recebe o valor do peso desse tipo de habilidade, apresentado na Tabela 5. Caso contrário, se não houver um vínculo entre o usuário e a habilidade, o valor desse tipo de habilidade recebe o valor "0".

Após a execução do cálculo da habilidade desconhecida, cada usuário que tenha essa habilidade terá um valor que indica o nível dessa habilidade. Esse valor será 0 ou 3, onde 0 indica a ausência dessa habilidade e 3 indica a sua presença.

Na seção 4.1.2.3 apresentará a implementação do código responsável por calcular a habilidade sugerida.

4.1.2.3 Habilidade sugerida

O cálculo da habilidade sugerida leva em consideração as sugestões de habilidade que um usuário recebe dos demais usuários da rede. Essas sugestões são realizadas através do módulo "*forum*". Esse módulo, que foi desenvolvido durante este trabalho, consiste em um ambiente onde serão realizadas discussões entre os usuários que necessitam de informações e os especialistas que detêm essas informações. Cada discussão está relacionada a uma área que representa o assunto central da discussão. Ao final dessa discussão, quando todas as dúvidas do usuário que solicita a ajuda forem sanadas, será exibida uma tela onde a ajuda do especialista será avaliada e algumas áreas relacionadas ao tema central serão exibidas como outras possíveis especialidades do especialista. Essa ação é apresentada na Figura 16.

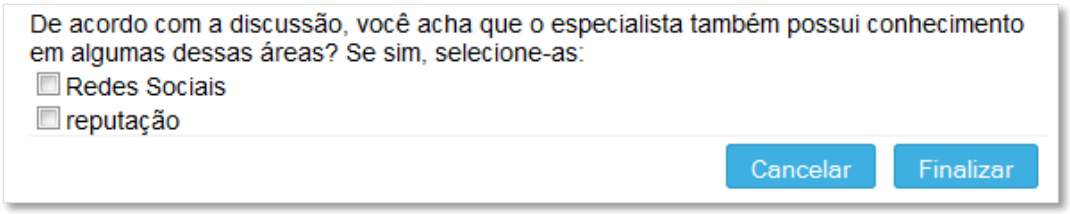


Figura 16: Tela de sugestão de habilidades

Conforme a Figura 16, o usuário que solicitou a ajuda poderá sugerir outras habilidades que o especialista poderá possuir. Essas outras habilidades relacionadas são obtidas através do processamento da coocorrência das palavras-chave presentes no Bibsonomy e da coocorrência das *tags* presentes na konnen.

O Bibsonomy é um sistema para o compartilhamento de favoritos e listas de literatura (BIBSONOMY, 2012, online). A coocorrência das tags desse ambiente é calculada utilizando

a API PHP disponibilizada na página[1] da documentação da ferramenta. Para calcular a coocorrência das *tags* de tais publicações, foi desenvolvido o método `bibsonomy()`, representado pelo pseudocódigo presente na Figura 17.

```
1  bibsonomy():
2
3      limite_requisicoes = X
4      area[] = max(coocorrencia_rede): retorna a keyword com maior
5                                       valor de coocorrencia
6                                       da rede
7      publicacoes(area): busca pelas publicações com a tag "area"
8
9      for(a = 0; limite_requisicoes >= 0; limite_requisicoes--)
10         foreach(publicacoes as p)
11             keywords = p.tags: retorna a lista de keyword da publicação
12             foreach(keywords as k)
13                 coocorrencia_temp.keyword_a = area[a];
14                 coocorrencia_temp.keyword_b = k.getName();
15                 coocorrencia_temp.bibtext_key = p.key;
16                 coocorrencia_temp.save()
17
18                 areas[] = k
19
20         a++
21         publicacoes(area[a]) : busca pelas publicações com a tag "area[a]"
22
23     coocorrencia_temp.single()
24
25     while(coocorrencia_temp)
26         peso_coocorrencia =
27             max(coocorrencia_temp.keyword_a, coocorrencia_temp.keyword_b)
28
29         coocorrencia_keyword.keyword_a = coocorrencia_temp.keyword_a
30         coocorrencia_keyword.keyword_b = coocorrencia_temp.keyword_b
31         coocorrencia_keyword.peso = peso_coocorrencia
32         coocorrencia_keyword.save()
33
34         coocorrencia_temp.delete_all(
35                 coocorrencia_temp.keyword_a, coocorrencia_temp.keyword_b)
36
37         coocorrencia_temp->single()
38
39     return void
```

Figura 17: Pseudocódigo do cálculo da coocorrência das palavras-chave das publicações do Bibsonomy

Primeiramente, no método `bibsonomy()`, apresentado na Figura 17, é definida uma variável (linha 3) para o controle da quantidade máxima de requisições que se deseja realizar para a API Bibsonomy. As requisições sempre buscam publicações que possuem todas as palavras-chave passadas como parâmetro. Para iniciar esse processo, a primeira palavra-chave utilizada é a palavra-chave que coocorre mais vezes dentro do ambiente da konnen (linha 4),

[1] Disponível em: http://www.bibsonomy.org/help/doc/download.html

porém, para as próximas requisições são utilizadas as próprias palavras-chave retornadas pelas buscas de publicações. Dessa forma, no passo seguinte (linha 7), é realizada a primeira requisição, retornando como resultado uma lista de produções bibliográficas, chamada `publications`. A partir disso, é realizado um loop enquanto a quantidade de requisições for maior ou igual a "0" (linha 9). Cada requisição retorna uma lista de publicações, que é percorrida (linha 10). Para cada publicação, são recuperadas as suas palavras-chave (linha 12), que são armazenadas no banco, realizando-se um relacionamento entre cada palavra-chave e o termo utilizado na busca de publicações (linhas 13 a 16) e adicionando-a no vetor de palavras-chave que serão utilizadas para realizar requisições (linha 18). Posteriormente (linha 20), o ponteiro que indica a palavra-chave que será utilizada na próxima requisição é somado ao valor "1". Após isso, considerando uma nova palavra-chave, é realizada uma nova requisição (linha 21), que tem o seu resultado tratado (linhas 10 a 18). Após o limite de requisições para a API Bibsonomy ser esgotado, é realizado o cálculo do peso da coocorrência entre os relacionamentos dos termos armazenados. Para isso, recupera-se um registro qualquer do relacionamento de dois termos (linha 23), e enquanto houverem registros de termos relacionados (linha 25). Para cada registro A de relacionamento entre as palavras-chave B e C, é realizada uma consulta que retorna a quantidade de vezes que eles coocorrem (linha 26 e 27), salvando-se um registro que contém o identificador das duas palavras-chave e o valor de coocorrência (linha 29 a 32). O próximo passo (linha 34 e 35) exclui da tabela que armazena os registros do tipo A todos os registros que relacionem as palavras-chave B e C. Por fim (linha 37), é recuperado mais um registro de relacionamento entre dois termos quaisquer, que terão seu valor de coocorrência calculado e salvo.

A coocorrência das tags presentes na konnen é baseada na coocorrência das *tags* dos conteúdos postados pelos usuários em seus perfis e comunidades. Esse mecanismo de *tagging*, desenvolvido durante a implementação dessa proposta, permite que cada usuário marque os conteúdos postados por ele e pelos demais usuários com as *tags* que desejar. Para calcular a coocorrência dessas *tags*, foi desenvolvido o método `network()`, apresentado no pseudocódigo da Figura 18.

```
1  network():
2
3      conteudos(): retorna todos os conteúdos postados
4                   pelos usuários da rede
5
6      foreach(conteudos as c)
7          tags_conteudo(c): retorna todas as tags
8                            de um conteúdo
9          salvar_coocorrencia(tags_conteudo)
10
11     coocorrencia_temp.single()
12
13     while(coocorrencia_temp)
14         peso_coocorrencia =
15             max(coocorrencia_temp.tag_a, coocorrencia_temp.tag_b)
16
17         coocorrencia_tag.tag_a = coocorrencia_temp.tag_a
18         coocorrencia_tag.tag_b = coocorrencia_temp.tag_b
19         coocorrencia_tag.peso = peso_coocorrencia
20         coocorrencia_tag.save()
21
22         coocorrencia_temp.delete_all(coocorrencia_temp.tag_a,
23                                      coocorrencia_temp.tag_b)
24
25         coocorrencia_temp->single()
26
27     return void
```

Figura 18: Pseudocódigo do cálculo da coocorrência das *tags* das publicações da konnen

No pseudocódigo do método `network()`, apresentado na Figura 18, primeiramente são recuperados e armazenados em uma lista (linha 3) todos os conteúdos postados nos perfis dos usuários e comunidades existentes na konnen. Para cada conteúdo da lista (linha 6), são recuperadas as suas tags (linha 7), que são passadas como parâmetro para o método `salvar_coocorrencia()`, que armazena no banco de dados o vínculo entre as *tags* passadas como parâmetro. Posteriormente, quando as *tags* de cada um dos conteúdos forem relacionadas, é realizado o cálculo do peso da coocorrência entre os relacionamentos dos termos armazenados. Para isso, recupera-se um registro qualquer do relacionamento de dois termos (linha 11), e, enquanto houverem registros de termos relacionados (linha 13), o processo continua. Para cada registro A de relacionamento entre as *tags* B e C, é realizada uma consulta que retorna a quantidade de vezes que eles coocorrem (linhas 14 e 15), salvando-se um registro que contém o identificador das duas tags e o valor da coocorrência entre elas (linhas 17 a 20). O próximo passo (linhas 22 e 23) exclui da tabela que armazena os registros do tipo A todos os registros que relacionem as tags B e C. Por fim (linha 25), é

recuperado mais um registro de relacionamento entre dois termos quaisquer, que terão seu valor de coocorrência calculado e armazenado.

A partir dos valores das coocorrências das palavras-chave das publicações do Bibsonomy e das *tags* dos conteúdos postados pelos usuários na konnen, é possível identificar quais serão as especialidades apresentadas para o usuário como sugestões de suas possíveis habilidades. Nesse caso, quando o usuário finaliza uma discussão, são consultados os cinco termos que possuem os maiores valores de coocorrência com o tema central da discussão, que são os apresentados na tela, conforme a Figura 23. Nesse passo, o usuário que pode ou não selecionar um ou mais termos como indicações de especialidades do especialista.

Essas indicações de especialidades realizadas são utilizadas como entrada para o cálculo da habilidade sugerida de um usuário, que é dividido em dois passos. O primeiro passo, representado pela Equação 8 (seção 4.1.1.3), considera apenas a reputação e maturidade dos autores das indicações. Segundo Silva et. al. (p. 3, 2012), a maturidade na rede social konnen indica o nível de maturidade do conteúdo gerado pelo usuário, onde, quanto maior, mais confiável o usuário tende a ser. Silva et. al. também definem a reputação de um usuário como o nível de confiança depositado sob ele, onde, quando maior, mais confiável o usuário será. O pseudocódigo desse algoritmo é exibido na Figura 19.

```
1  habilidade_sugerida(usuario, habilidade):
2
3      usuarios(usuario, habilidade): retorna todos os usuários que
4                                     sugeriram o "usuario" como especialista
5                                     na área "habilidade"
6      foreach(usuarios as u)
7          usuario.hsugerida += u.reputacao + u.maturidade;
8
9      return usuario.hsugerida;
```

Figura 19: Pseudocódigo da Equação 8, responsável pela primeira etapa do cálculo da habilidade sugerida

No pseudocódigo apresentado na Figura 19, são recuperados e armazenados na variável `usuarios` todos os usuários que sugeriram o usuário como especialista na área representada pelo parâmetro `habilidade` (linha 3). Após isso (linha 7), é realizado o somatório da reputação e maturidade de cada um dos usuários da lista `usuarios`. O valor retornado por esse método é o resultado do desse somatório, que não possui um valor máximo definido, pois ele depende da quantidade de usuários que realizaram a recomendação.

O segundo passo para o cálculo da habilidade sugerida é a implementação da Equação 9, responsável por normalizar o valor encontrado no primeiro passo. O pseudocódigo desse processo é apresentado na Figura 20.

```
1  habilidade_sugerida_normalizacao(habilidade):
2
3      usuarios(habilidade): retorna todos os usuários que foram
4                            sugeridos como especialistas na área "habilidade"
5      maxHsug(habilidade): retorna a maior habilidade sugerida
6                           na área "habilidade"
7
8      foreach(usuarios as u):
9          u.hsugerida = (u.hsugerida / maxHsug) * whsug;
10
11     return void;
```

Figura 20: Pseudocódigo da Equação X, responsável pela segunda etapa do cálculo da habilidade sugerida

Inicialmente (linha 3), no pseudocódigo da Figura 20, são recuperados todos os usuários que possuem um valor de habilidade sugerida representado pelo parâmetro `habilidade`. Posteriormente (linha 5), é armazenado na variável `maxHsug` o maior valor da habilidade sugerida presente entre os usuários da rede. Por fim (linha 9), o valor da habilidade sugerida de cada usuário é dividido por `maxHsug` e multiplicado pelo peso da habilidade sugerida, definido na Tabela 5.

Após a execução das duas etapas do cálculo da habilidade sugerida, cada usuário que tenha habilidade na área em questão terá um valor que indica o nível dessa habilidade. Esse valor é compreendido entre 0 e 1, onde, quanto mais próximo de 0, menor é a habilidade do usuário, e, quanto mais próximo de 1, maior é a habilidade.

Na seção 4.1.2.4 será apresentada a implementação do código responsável por calcular a habilidade refutada.

4.1.2.4 Habilidade refutada

O cálculo da habilidade refutada leva em consideração as ajudas em que o usuário teve o papel de especialista e a sua ajuda foi avaliada como não útil. Essas avaliações também são realizadas através do módulo "*forum*". Cada discussão está relacionada a uma área que representa o assunto central da discussão. Ao final dessa discussão, quando todas as dúvidas do usuário que solicita a ajuda forem sanadas, será exibida uma tela onde a ajuda do especialista será avaliada. Essa ação é apresentada na Figura 21.

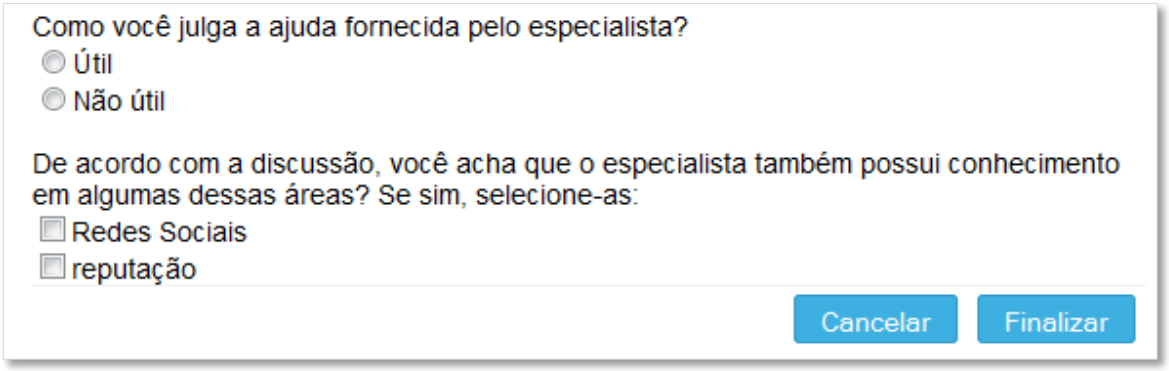


Figura 21: Tela de avaliação da ajuda prestada em uma discussão

Conforme a Figura 21, o usuário que solicitou a ajuda poderá avaliar a ajuda prestada pelo especialista como "útil" ou "não útil". As ajudas avaliadas como úteis recebem o peso 1, enquanto que as ajudas avaliadas como não úteis recebem o peso -1. As ajudas não úteis são utilizadas como entrada para o cálculo da habilidade refutada de um usuário, que é dividido em dois passos. O primeiro passo, representado pela Equação 6 (seção 4.1.1.4), considera apenas a reputação e maturidade dos autores das indicações. O pseudocódigo desse algoritmo é exibido na Figura 22.

```
1  habilidade_refutada(usuario, habilidade):
2
3      usuarios(usuario, habilidade): retorna todos os usuários que refutaram
4                                     o "usuario" como especialista
5                                     na área "habilidade"
6      foreach(usuarios as u)
7          usuario.hrefutada += u.reputacao + u.maturidade;
8
9      return usuario.hrefutada;
```

Figura 22: Pseudocódigo da Equação 6, responsável pela primeira etapa do cálculo da habilidade refutada

No pseudocódigo apresentado na Figura 22, são recuperados e armazenados na variável `usuarios` todos os usuários que refutaram o usuário como especialista na área representada pelo parâmetro `habilidade` (linha 3). Após isso (linha 7), é realizado o somatório da reputação e maturidade de cada um dos usuários da lista `usuarios`. O valor retornado por esse método é o resultado do somatório, que não possui um valor máximo definido, pois ele depende da quantidade de usuários que realizaram a avaliação.

O segundo passo para o cálculo da habilidade refutada é a implementação da EQUAÇÃO 7, responsável por normalizar o valor encontrado no primeiro passo. O pseudocódigo desse processo é apresentado na Figura 23.

```
1  habilidade_refutada_normalizacao(habilidade):
2
3      usuarios(habilidade): retorna todos os usuários que foram
4                            refutados como especialistas na área "habilidade"
5      maxHref(habilidade): retorna a maior habilidade refutada
6                           na área "habilidade"
7
8      foreach(usuarios as u):
9          u.hrefutada = (u.hrefutada / maxHref) * whref;
```

Figura 23: Pseudocódigo da Equação X, responsável pela segunda etapa do cálculo da habilidade refutada

Inicialmente (linha 3), no pseudocódigo da Figura 23, são recuperados todos os usuários que possuem um valor de habilidade refutada representado pelo parâmetro `habilidade`. Posteriormente (linha 5), é armazenado na variável `maxHref` o maior valor da habilidade refutada presente entre os usuários da rede. Por fim (linha 9), o valor da habilidade refutada de cada usuário é dividido por `maxHref` e multiplicado pelo peso da habilidade refutada, definido na Tabela 5.

Após a execução das duas etapas do cálculo da habilidade refutada, cada usuário que tenha habilidade na área em questão terá um valor que indica o nível dessa habilidade. Esse valor é compreendido entre -1 e 0, onde, quanto mais próximo de -1, maior é a refutação da habilidade, e, quanto mais próximo de 0, menor ela será.

Na seção 4.1.2.5 apresentará a implementação do código responsável por calcular a habilidade inferida.

4.1.2.5 Habilidade inferida

O cálculo da habilidade inferida leva em consideração a coocorrência das palavras-chave do Bibsonomy, coocorrência das *tags* dos conteúdos da konnen e a coocorrência das informações textuais do usuário. O método para o cálculo da coocorrência das palavras-chave do Bibisonomy e das tags da konnen foram apresentados na seção 4.1.2.3. Para calcular a coocorrência das informações textuais dos usuários foi desenvolvido o método `users()`, representado pelo pseudocódigo da Figura 24.

```
1  users():
2
3      usuarios(): retorna todos os usuários da rede
4
5      foreach(usuarios as u)
6
7          conteudos(u): retorna todos os conteúdos do usuário "u"
8          foreach(conteudos as c)
9              salvar_coocorrencia(c.conteudo)
10             tags_conteudo(c): retorna todas as tags do conteúdo
11             salvar_coocorrencia(tags_conteudo)
12
13         producoes(u): retorna todas as publicações do usuário "u"
14         foreach(producoes as p)
15             salvar_coocorrencia(p.conteudo)
16             tags_producoes(p): retorna todas as tags da publicação
17             salvar_coocorrencia(tags_producoes)
18
19         mensagens(u): retorna todas as mensagens enviadas pelo usuário "u"
20         foreach(mensagens as m)
21             salvar_coocorrencia(m.conteudo)
22
23         comentarios(u): retorna todos os comentários realizados pelo usuário
24         foreach(comentarios as c)
25             salvar_coocorrencia(c.conteudo)
26
27         coocorrencia_temp.where(u).single()
28
29         while(coocorrencia_temp)
30             peso_coocorrencia =
31                 max(coocorrencia_temp.tag_a, coocorrencia_temp.tag_b)
32
33             coocorrencia_user.keyword_a = coocorrencia_temp.tag_a
34             coocorrencia_user.keyword_b = coocorrencia_temp.tag_b
35             coocorrencia_user.peso = peso_coocorrencia
36             coocorrencia_user = u
37             coocorrencia_user.save()
38
39             coocorrencia_temp.delete_all(coocorrencia_temp.tag_a,
40                                          coocorrencia_temp.tag_b,
41                                          u)
42
43             coocorrencia_temp.where(u).single()
44
45         return void
```

Figura 24: Pseudocódigo do cálculo da coocorrência dos termos de um usuário

Primeiramente, no método `users()`, apresentado na Figura 24, todos os usuários da konnen são retornados e armazenados em uma lista (linha 3). Para cada usuário da lista, retornam-se todos os seus conteúdos (linha 7). Cada um desses conteúdos tem o seu texto e tags passados como parâmetro para o método `salvar_coocorrencia`()(linhas 9 e 11), que armazena no banco de dados o vínculo entre todas as palavras passadas como parâmetro. Seguindo a mesma lógica, são salvas no banco a coocorrência dos títulos das publicações

científicas e suas palavras-chave (linhas 13 a 17), mensagens enviadas (linhas 19 a 21) e comentários (linhas 23 a 25) do usuário. Após o registro do relacionamento das informações de cada usuário, é realizado o cálculo do peso da coocorrência entre eles. Para isso, recupera-se um registro qualquer do relacionamento de dois termos do usuário (linha 27). Para cada registro A de relacionamento entre as palavras-chave B e C, é realizada uma consulta que retorna a quantidade de vezes que eles coocorrem (linha 26 e 27) em relação ao usuário, salvando-se um registro que contém o identificador das duas palavras-chave, o valor de coocorrência e o identificador do usuário (linha 30 a 37). O próximo passo (linha 39 a 41) exclui da tabela que armazena os registros do tipo A todos os registros que relacionem as palavras-chave B e C com o usuário. Por fim (linha 43), é recuperado mais um registro de relacionamento entre dois termos quaisquer do usuário, que terão seu valor de coocorrência calculado e salvo.

Com os dados de coocorrência dos termos na rede konnen e Bibsonomy e da coocorrência dos termos em relação a um usuário, é possível realizar o cálculo da habilidade inferida do usuário. Esse cálculo, que é baseado na Equação 12, considera a similaridade entre o vetor formado com os dados da coocorrência dos termos na konnen e Bibsonomy e o vetor formado com os dados da coocorrência do usuário. O pseudocódigo do método que calcula a habilidade inferida de um usuário é apresentado no fluxograma da Figura 25.

```
1  habilidade_inferida(usuario, habilidade)
2
3      array_a(habilidade): retorna todos os registros de
4                           coocorrência que envolvem a "habilidade"
5      array_b(usuario, habilidade): retorna todos os registros de
6                           coocorrência que envolvem a "habilidade" e o "usuario"
7
8      similaridade = 0;
9
10     if(array_a.count() > 0 && array_b.count() > 0)
11
12         array_a_normalizado = normalizar(array_a, array_a_normalizado)
13
14         array_b_normalizado = normalizar(array_b, array_a_normalizado)
15
16         array_b_normalizado.fundir(array_a_normalizado)
17
18         array_b_normalizado.remover(array_a_normalizado)
19
20         numerador = 0
21
22         sum_a_pow = 0
23         media_a = sum(array_a_normalizado) / count(array_a_normalizado)
24
25         sum_b_pow = 0
26         media_b = sum(array_b_normalizado) / count(array_b_normalizado)
27
28         foreach(array_a as k -> a)
29             numerador += (array_a_normalizado[k] - media_a) *
30                          (array_b_normalizado[k] - media_b)
31             sum_a_pow += pow(array_a_normalizado[k] - media_a, 2)
32             sum_b_pow += pow(array_b_normalizado[k] - media_b, 2)
33
34         similaridade = numerador / (sqrt(sum_a_pow) * sqrt(sum_b_pow))
35
36     if(similaridade > 0.3)
37         return true
38     else
39         return false
```

Figura 25: Pseudocódigo da Equação 12, responsável pelo cálculo da habilidade inferida

No pseudocódigo da Figura 25, o método `habilidade_inferida()` recebe como parâmetro um usuário e uma habilidade. No início (linha 3), são recuperados todos os registros de coocorrência do Bibsonomy e da konnen que envolvem a habilidade informada, armazenando-os na variável `array_a`. Após isso (linha 5), são recuperados todos os registros de coocorrência do usuário que envolvem a habilidade informada, armazenando-os na variável `array_b`. Depois disso (linha 8), é criada a variável `similaridade` com o valor “0”. O passo seguinte (linha 10) verifica se existem elementos nos vetores `array_a` e `array_b`, pois, se não houverem, não há como normalizá-los para calcular a similaridade entre eles. Dessa forma, se ambos os vetores possuírem valores, o próximo passo é normalizar cada um deles. Nesse caso, a normalização de um vetor consiste em três passos: 1) criar um novo vetor onde o valor de cada posição é o identificador da palavra que coocorre com

habilidade e o conteúdo dessa posição é o valor de coocorrência entre esses termos; 2) inserir nesse vetor criado um elemento com a chave recebendo o valor do identificador da habilidade e o valor recebendo a soma do maior valor do vetor com o número um, para colocar o termo habilidade em evidência; 3) dividir o valor de cada elemento pelo maior valor do vetor, com o objetivo de normalizar os valores no intervalo de 0 a 1, e, assim, poder calcular a similaridade entre os vetores. Após a normalização dos dois vetores (linha 12 e 14), o vetor aray_a_normalizado possui os valores antes contidos no array_a e o vetor aray_b_normalizado possui os valores antes contidos no array_b. O próximo passo é inserir no aray_b_normalizado todos os valores de aray_a_normalizado que não estão no seu conjunto de valores (linha 16). Em seguida, deve-se remover do aray_b_normalizado os elementos que não ocorrem em aray_a_normalizado, para que ambos possuam os mesmo valores, pois o cálculo de similaridade considera apenas os elementos em comum entre os vetores. Após isso, o próximo passo é calcular a similaridade entre os vetores utilizando o Coeficiente de Pearson. Para isso, são criadas (linhas 20 a 26) as variáveis auxiliares numerador, sum_a_pow e sum_b_pow, ambas com valor "0", media_a, que possui o valor da média aritmética dos elementos do aray_a_normalizado, e media_b, que possui o valor da média aritmética dos elementos do aray_b_normalizado. O próximo passo é percorrer os elementos dos vetores realizando três operações: 1) o numerador acumula o valor de aray_a_normalizado na posição atual menos o valor da media_a multiplicado esse resultado pelo valor de aray_b_normalizado na posição atual menos o valor da media_b (linhas 29 e 30); 2) a variável sum_a_pow recebe o valor de aray_a_normalizado na posição atual menos o valor da media_a, elevando esse resultado a 2 (linha 31); 3) a variável sum_b_pow recebe o valor de aray_b_normalizado na posição atual menos o valor da media_b, elevando esse resultado a 2 (linha 32). Posteriormente (linha 34), a variável similaridade recebe o resultado da divisão da variável numerador pela soma da raiz quadrada de sum_a_pow multiplicada pelo valor da raiz quadrada de sum_b_pow. Por último (linhas 36 a 39), é realizada a verificação do conteúdo da variável similaridade, retornando true se esse valor for maior que "0.3", que indica uma forte correlação entre os vetores aray_a_normalizado e aray_b_normalizado, ou retornando false em caso contrário.

O retorno realizado pelo método `habilidade_inferida()` indica se uma habilidade deve ser, quando o retorno é `true`, ou não, quando o retorno é `false`, recomendada para um usuário. Essas recomendações são apresentadas para o usuário no momento que ele realiza a edição do seu perfil, onde ele pode ou não confirmar a recomendação. No momento que o usuário confirma ter conhecimento sobre determinada habilidade que o sistema sugeriu, essa habilidade deixa de ser uma habilidade inferida e passa a ser uma habilidade desconhecida.

A habilidade inferida não entra no cálculo da habilidade final de um usuário, pois ele tem como objetivo apenas auxiliar o usuário no processo de informar ao sistema quais são suas habilidades. Dessa forma, a próxima seção apresenta como se dá o cálculo da habilidade geral de um usuário, utilizando, para isso, as habilidades mencionadas nas seções anteriores.

4.1.2.6 Habilidade geral

Para o cálculo da habilidade geral de um usuário em uma determinada área, são considerados os valores dos outros tipos de habilidades que o usuário possui para essa mesma área. O código implementado para o cálculo da habilidade geral é baseado na da Equação 13 (seção 4.1.1.6), representado através do pseudocódigo da Figura 26.

```
1  habilidade_geral(usuario, habilidade):
2      usuario.habilidade = (
3                              usuario.hdemonstrada +
4                              usuario.hdesconhecida +
5                              usuario.hsugerida +
6                              usuario.hrefutada
7                            ) /
8                           (whdem + whdes + whsug + whref);
9      return usuario.habilidade;
```

Figura 26: Pseudocódigo da Equação 13, responsável pelo cálculo da habilidade geral

Como apresentado na Figura 26, o método `habilidade_final()` recebe como parâmetro um usuário e uma habilidade. A partir desses parâmetros e realizado um cálculo que considera a soma das habilidades demonstrada, desconhecida, sugerida e refutada do usuário na área representada por habilidade, dividindo-se esse valor pelos pesos da habilidade demonstrada, desconhecida, sugerida e refutada, apresentados naTabela 5. Essa divisão resulta em um valor normalizado no intervalo de 0 a 1, onde quanto mais próximo de 0,

menor é o grau final da habilidade do usuário na área mencionada, e, quanto mais próximo de 1, maior é esse grau.

A partir do cálculo da habilidade geral, é possível realizar a recomendação dos especialistas, processo esse apresentado na seção 4.1.2.7.

4.1.2.7 Recomendação

A partir do cálculo geral das habilidades dos usuários da rede, é possível fornecer uma interface de busca de especialistas. A interface de busca desenvolvida é dividida em duas páginas: uma com o campo de busca e a outra com a lista de especialistas encontrados. A página que apresenta o campo de busca é apresentada na Figura 27.

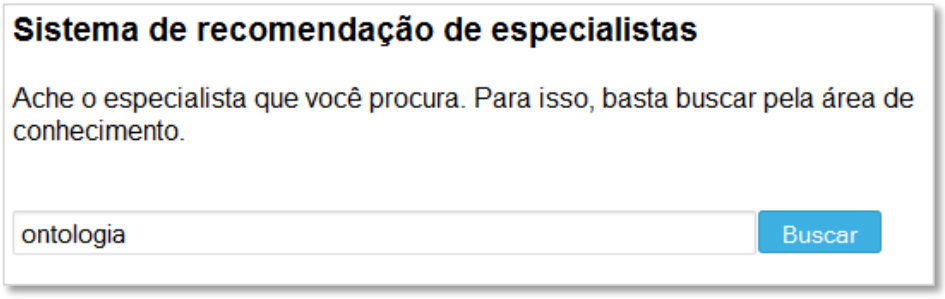


Figura 27: Página de buca

Na página exibida na Figura 27 existe um campo onde é informada a área onde se precisa de um especialista. Nesse caso, foi informado o termo “ontologia”, que resultou na lista de especialistas apresentados na Figura 28.

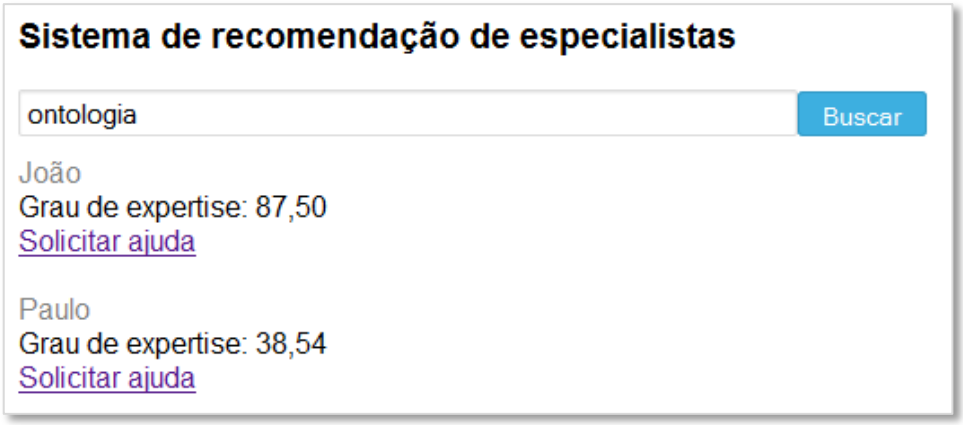


Figura 28: Página com a lista de especialistas em "ontologia"

Na Figura 28 é exibida a lista de especialistas com conhecimentos sobre “ontologia”. Para a ordenação dessa lista de especialistas, foi adotada a Teoria do auto-interesse, criada por Fazel-Zarandi *et. al.* (2011, p. 43-44), que define a ordenação dos resultados em ordem

decrescente de acordo ao grau de expertise. Essa lista apresenta dois resultados: o especialista João, com grau de especialidade 87,50, e o especialista Paulo, com grau de expertise 38,54.

Cada item da lista de especialistas apresentada possui a opção “Solicitar ajuda” para que o usuário solicite a ajuda do especialista. Ao acessar essa opção, é apresentado um formulário onde o usuário informa a sua necessidade de informação e, ao submeter esse formulário, é criada uma discussão que é vinculada ao especialista. A partir da criação dessa discussão, o usuário e o especialista debaterão o assunto até que todas as dúvidas sejam sanadas. Nesse momento, o usuário que criou a discussão avaliará a ajuda prestada pelo especialista, dizendo se ela foi útil ou não, e, opcionalmente, informando outras possíveis habilidades do especialista. Essa avaliação das discussões, juntamente com as outras informações geradas pelo usuário na rede, fornecem novos dados para uma próxima execução do algoritmo de identificação de especialistas.

A seção 5 apresenta as considerações finais sobre o trabalho desenvolvido.

5. CONSIDERAÇÕES FINAIS

Este trabalho apresentou conceitos referentes a Redes Sociais, abordando alguns aspectos relacionados aos relacionamentos existentes em uma rede, como “caminho”, “distância” e “densidade”, Sistemas de Recomendação de Especialistas, citando os principais problemas, como a construção de perfis de conhecimento de usuários de forma rápida e confiável com o mínimo de esforço manual possível do usuário.

Foi apresentado um modelo de um Sistema de Recomendação de especialistas, que identifica e mensura as habilidades dos usuários, o que fornece meios para a recomendação. O modelo apresentado no presente trabalho é genérico, ou seja, pode ser aplicado a qualquer ambiente que forneça os dados utilizados para o cálculo de cada tipo de habilidade. Nesse caso, os dados necessários para o cálculo das habilidades são:

- publicações científicas com palavras-chave e autores, para o cálculo da habilidade demonstrada;
- informações sobre as habilidades do usuário, fornecidas por ele próprio, para o cálculo da habilidade desconhecida;
- indicações de habilidades para um usuário, realizadas por outros usuários presentes no ambiente, utilizadas no cálculo da habilidade sugerida;
- avaliações sobre as ajudas prestadas por um usuário nas dúvidas dos demais usuários do ambiente, que são utilizadas no cálculo da habilidade refutada;
- dados das coocorrências entre os termos do usuário e os demais termos presentes no ambiente, para o cálculo da habilidade inferida.

Esse modelo proposto foi implantado na rede social educacional konnen, utilizando informações internas desse ambiente, que foram enriquecidas com dados externos da Plataforma Lattes, Linkedin e Bibsonomy. Ao analisar essa implantação nesse ambiente, através da avaliação dos resultados fornecidos por cada etapa implementada para a identificação e mensuração de habilidades de usuários, acredita-se que o modelo proposto confirma a hipótese de que os dados listados anteriormente fornecem parâmetros suficientes para realizar a recomendação de especialistas de forma aceitável. O conjunto de habilidades possíveis que podem ser inferidas nesse mecanismo está limitado ao conjunto de palavras-chave oriundos das publicações científicas, Bibsonomy ou marcações realizadas nos objetos do ambiente.

Acredita-se ainda que a implantação desse mecanismo na konnen estimulará os usuários a colaborarem com maior frequência com os outros. Essa hipótese deve ser testada

através da coleta de dados de utilização da rede por determinado período, que, ao serem analisados, comprovarão ou não essa suposição.

Como trabalhos futuros, pretende-se analisar os resultados provenientes da implantação deste mecanismo na plataforma konnen – que não foi realizada no presente trabalho por questões de escopo e objetivo. Ainda pode ser desenvolvida uma análise detalhada no algoritmo desenvolvido para o processo de inferência de especialidades com o objetivo de identificar possíveis otimizações que podem ser realizadas. Além disso, novas fontes de conhecimento externas dos usuários podem ser exploradas para que seus perfis contenham informações mais detalhadas sobre suas especialidades, aumentando assim a tendência de melhorias no processo de inferência das especialidades. Por fim, outra possibilidade é modificar o modelo proposto para considerar a evolução das habilidades dos usuários, ou seja, acompanhar o crescimento do grau de expertise em relação ao tempo.

6. REFERÊNCIAS BIBLIOGRÁFICAS

BAEZA-YATES, Ricardo; RIBEIRO-NETO, Berthier. **Modern information retrieval**. ACM Press, 1999, 513 p.

BAEZA-YATES, Ricardo; RIBEIRO-NETO, Berthier. **Modern information retrieval**. 2011. Disponível em: < http://grupoweb.upf.es/WRG/mir2ed/pdf/slides_chap03.pdf>. Acesso em: 15 jun. 2012.

BECERRA-FERNANDEZ, Irma. Searching for experts on the Web: A review of contemporary expertise locator systems. **ACM Trans. Internet Technol.**, v. 6, p. 333-355, nov. 2006. Disponível em: < http://dl.acm.org/citation.cfm?id=1183464>. Acesso em: 01 abr. 2012.

COSTA, Ricardo Araújo. **Uma análise do uso de Redes Sociais como ferramenta para Gestão do Conhecimento**. 2012. 166 p. Tese (Doutorado em Ciência da Computação) – Universidade Federal de Pernanbuco, PE, BR.

EHRLICH, Kate; SHAMI, N. Sadat. Searching for expertise. **Proceedings of the twenty-sixth annual SIGCHI conference on Human factors in computing systems**, Florence, IT, p. 1093-1096, 2008. Disponível em: < http://doi.acm.org/10.1145/1357054.1357224>. Acesso em: 01 abr. de 2012.

FAZEL-ZARANDI, Maryam; DEVLIN, Hugh J.; HUANG, Yun; CONTRACTOR, Noshir. Expert recommendation based on social drivers, social network analysis, and semantic data representation. **Proceedings of the 2nd International Workshop on Information Heterogeneity and Fusion in Recommender Systems**, Chicago, Illinois, USA, p. 41-48, 2011. Disponível em: < http://doi.acm.org/10.1145/2039320.2039326 >. Acesso em: 07 abr. 2012.

Fazel-Zarandi, Maryam; Fox, Mark S. Constructing expert profiles over time for skills management and expert finding. **Proceedings of the 11th International Conference on Knowledge Management and Knowledge Technologies**, Graz, AUT, p. 5:1-5:6, 2011. Disponível em: <http://dl.acm.org/citation.cfm?id=2024295>. Acessado em: 18 maio 2012.

GOLBECK, Jennifer Ann. **Computing and applying trust in web-based social networks**. 2005. 199 p. Tese (Doutorado em computação) - University of Maryland, MD, USA.

LIN, Ching-Yung; EHRLICH, Kate; GRIFFITHS-FISHER, Vicky; DESFORGES, Christopher. SmallBlue: Social Network Analysis for Expertise Search and Collective Intelligence. **IEEE MultiMedia**, v. 15, p. 78-84, jan. – mar. 2008. Disponível em: <http://dx.doi.org/10.1109/MMUL.2008.17>. Acesso em: 01 abr. de 2012.

PIMENTEL, Mariano (Org.); FULKS, Hugo (Org.). **Sistemas colaborativos**. Rio de Janeiro: Elsevier, 2011. 375 p.

REICHLING, Tim; WULF, Volker. Expert recommender systems in practice: evaluating semi-automatic profile generation. **Proceedings of the 27th international conference on Human factors in computing systems**, Boston, MA, USA, p. 59-68, 2009. Disponível em: <http://dl.acm.org/citation.cfm?id=1518712>. Acessado em: 02 abr. 2012.

SERDYUKOV, Pavel; FENG, Ling; BUNNINGEN, Arthur; EVERS, Sander; HEERDE, Harold; APERS, Peter; FOKKINGA, Maarten; HIEMSTRA, Djoerd. The Right Expert at the Right Time and Place. **Proceedings of the 7th International Conference on Practical Aspects of Knowledge Management**, Yokohama, LP, p. 38-49, 2009. Disponível em: < http://dl.acm.org/citation.cfm?id=1484842.1484851&coll=DL&dl=GUIDE>. Acessado em: 10 abr. 2012.

SILVA, E. M. ; RODRIGUES, D. O. ; Souza, J. G. ; SALGADO, A. C. ; MEIRA, S. R. L. T-SWEETS: an alternative to the Stimulus Collaboration from trust Inference in Social Networks. In: **Simpósio Brasileiro de Sistemas Colaborativos**, 2012, São Paulo. 2012 Brazilian Symposium on Collaborative Systems, 2012. p. 1.

SHAPIRA, Bracha ; ZABAR, Boaz Personalized search: Integrating collaboration and social networks. **J. Am. Soc. Inf. Sci. Technol.**, v. 62, p. 146 - 160, jan 2011. Disponível em: < http://dl.acm.org/citation.cfm?id=1943159>. Acesso em: 15 jun. 2012.

TANG, Jie; ZHANG, Jing; YAO, Limin; LI, Juanzi; ZHANG; SU, Zhong. ArnetMiner: extraction and mining of academic social networks. **Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining**, Las Vegas, NV, USA, p. 990 - 998, 2008. Disponível em: < http://dl.acm.org/citation.cfm?id=1402008>. Acesso em: 15 jun. 2012.