UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

VINÍCIUS DE OLIVEIRA ANDRADE

Sistema de Recomendação de Desenvolvedores Especialistas para Projetos de Engenharia de Software: um Estudo de Caso

Recife

2023

VINÍCIUS DE OLIVEIRA ANDRADE

Sistema de Recomendação de Desenvolvedores Especialistas para Projetos de Engenharia de Software: um Estudo de Caso

Dissertação apresentada ao Programa de Pós-graduação em Ciência da Computação do Centro de Informática (CIn) da Universidade Federal de Pernambuco – UFPE, como parte dos requisitos parciais para obtenção do título de Mestre em Ciência da Computação em 20 de agosto de 2023.

Área de Concentração: Inteligência Computacional

Orientador (a): Flávia de Almeida Barros

Recife

2023

Catalogação na fonte
Bibliotecária Monick Raquel Silvestre da S. Portes, CRB4-1217

A554s Andrade, Vinícius de Oliveira
Sistema de recomendação de desenvolvedores especialistas para projetos de engenharia de software: um estudo de caso / Vinícius de Oliveira Andrade. – 2023.
72 f.: il., fig., tab.

Orientadora: Flávia de Almeida Barros.
Dissertação (Mestrado) – Universidade Federal de Pernambuco. CIn, Ciência da Computação, Recife, 2023.
Inclui referências.

1. Inteligência computacional. 2. Engenharia de software. I. Barros, Flávia de Almeida (orientadora). II. Título.

006.31 CDD (23. ed.) UFPE - CCEN 2023-184

Vinícius de Oliveira Andrade

“Sistema de Recomendação de Desenvolvedores Especialistas para Projetos de Engenharia de Software: um Estudo de Caso”

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Pernambuco, como requisito parcial para a obtenção do título de Mestre em Ciência da Computação. Área de Concentração: Inteligência Computacional.

Aprovado em: 22 de agosto de 2023.

BANCA EXAMINADORA

_____

Prof. Dr. Alexandre Cabral Mota
Centro de Informática / UFPE

_____

Prof. Dr. André Câmara Alves do Nascimento
Departamento de Computação / UFRPE

_____

Profa. Dra. Flávia de Almeida Barros
Centro de Informática / UFPE
(**Orientadora**)

Dedico este trabalho aos meus amados pais, pois eles sempre deram o melhor de si para me apoiar em cada etapa do caminho. Sua dedicação incansável, amor incondicional e sacrifícios foram fundamentais para que eu chegasse onde estou hoje. Sou profundamente grato por tudo que fizeram por mim e por serem uma fonte constante de inspiração e força. Este trabalho é uma pequena forma de expressar minha gratidão e homenagear sua presença constante em minha vida.

AGRADECIMENTOS

Em primeiro lugar, expresso minha gratidão a Deus por Sua força e por abençoar-me com uma maravilhosa família e amigos.

Agradeço imensamente aos meus pais, Joana e Fernando, por proporcionarem as oportunidades necessárias para que eu chegasse onde estou hoje. Além disso, sou grato por todo o carinho, amor e atenção que sempre me dedicaram, incentivando-me a perseguir meus sonhos. Agradeço especialmente por estarem sempre presentes em minha vida, servindo como exemplo a ser seguido.

Quero expressar minha sincera gratidão aos amigos que conheci durante a pós-graduação. Sua ajuda foi inestimável não apenas para este projeto, mas também ao longo de nossa jornada, pois enfrentamos desafios com coragem e determinação.

Sou imensamente grato aos meus professores, que demonstraram paciência ao responder minhas dúvidas (eu sei que nem sempre foi fácil). Além de compartilharem conhecimento e experiência, eles proporcionaram momentos de reflexão e me ajudaram a enxergar novos horizontes.

Por fim, expresso minha profunda gratidão à minha orientadora, que tem sido um exemplo tanto na área acadêmica quanto na vida. Sua orientação e apoio foram fundamentais para a conclusão deste trabalho. Além disso, agradeço por orientar-me também em questões pessoais, pois sem sua presença e orientação, eu não teria alcançado esse feito.

RESUMO

Em projetos de desenvolvimento de software, a alocação eficiente de desenvolvedores a tarefas específicas é de extrema importância, uma vez que essa distribuição pode impactar diretamente o tempo de desenvolvimento e a qualidade do produto final. Objetivando auxiliar os gestores, propomos aqui um processo de recomendação de desenvolvedores para projetos e tarefas com base no conhecimento acumulado por cada especialista (modelo orientado a tópico). Nossa proposta adota uma estratégia personalizada de recomendação. Aqui, o usuário do sistema é o gerente responsável pela alocação e os desenvolvedores são vistos como os "itens" que serão recomendados. Assim, o perfil (efêmero) do usuário será representado pela descrição da tarefa a ser realizada no momento, e os desenvolvedores ("itens") são representados pela descrição dos trabalhos e atividades que cada um já realizou (dados históricos). Por fim, adotamos a técnica de recomendação baseada em filtragem de conteúdo, que analisa a similaridade entre tarefas a realizar ("perfil" do usuário) e os dados históricos dos desenvolvedores. Essa configuração de estratégia e técnica apresentou os melhores resultados nos testes exploratórios realizados com diferentes abordagens. Foi implementado um sistema protótipo para recomendação de especialistas no contexto de uma empresa de desenvolvimento de software. Esse sistema, implementado em Python, conta com três módulos: (1) Coleta de Informações, que recupera e pré-processa as informações necessárias disponíveis na ferramenta de gerenciamento de tarefas da empresa parceira; (2) Módulo de Recomendação, que analisa as informações coletadas e recomenda os desenvolvedores mais adequados para cada tarefa e projeto; e (3) Módulo de Gerenciamento de Alocação, responsável por gerenciar os cadastros do sistema, registrando e mantendo o controle das alocações de desenvolvedores a tarefas e projetos. As tecnologias utilizadas são: Django, Jira, NLTK, Pandas e Scikit-learn. A validação do protótipo foi realizada durante um período de dois meses de uso, considerando um total de 191 recomendações de atividades. Os resultados foram analisados observando-se quais desenvolvedores recomendados pelo sistema foram de fato alocados pelo gerente para as tarefas de entrada. O protótipo alcançou uma precisão de 65,13%, o que indica que os resultados não são aleatórios, e que o sistema tem potencial para melhorar o processo de alocação de desenvolvedores a tarefas, facilitando a busca pelo profissional mais adequado para cada atividade.

Palavras-chave: alocação de especialistas; sistema de recomendação; engenharia de software.

ABSTRACT

In software development projects, the efficient allocation of developers to specific tasks is of utmost importance, as this distribution can directly impact the development time and the quality of the final product. To assist managers in this endeavor, we propose a developer recommendation process for projects and tasks based on the accumulated knowledge of each specialist (topic-oriented model). Our proposal adopts a personalized recommendation strategy. Here, the system user is the manager responsible for the allocation, and the developers are seen as the "items" to be recommended. Thus, the user's (ephemeral) profile is represented by the description of the task to be performed at the moment, and the developers ("items") are represented by the description of the work and activities each of them has already performed (historical data). Finally, we adopt the content-based recommendation technique, which analyzes the similarity between the tasks to be performed ("user" profile) and the historical data of the developers. This configuration of strategy and technique yielded the best results in the exploratory tests conducted with different approaches. A prototype system for expert recommendation was implemented in the context of a software development company. This system, implemented in Python, comprises three modules: (1) Information Collection, which retrieves and preprocesses the necessary information available in the partner company's task management tool; (2) Recommendation Module, which analyzes the collected information and recommends the most suitable developers for each task and project; and (3) Allocation Management Module, responsible for managing the system's records, registering and maintaining control of developer allocations to tasks and projects. The technologies used include Django, Jira, NLTK, Pandas, and Scikit-learn. The validation of the prototype was carried out during a two-month period of usage, considering a total of 191 activity recommendations. The results were analyzed by observing which developers recommended by the system were actually allocated by the manager to the input tasks. The prototype achieved an accuracy of 65.13%, indicating that the results are not random and that the system has the potential to improve the process of allocating developers to tasks, facilitating the search for the most suitable professional for each activity.

Keywords: expert allocation; recommendation system; software engineering.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AUC	Área sob a curva ROC
SR	Sistemas de Recomendação
SVM	Support Vector Machines
TF-IDF	Term Frequency-Inverse Document Frequency

SUMÁRIO

1 INTRODUÇÃO

Em projetos de desenvolvimento de software, a alocação eficiente de desenvolvedores a tarefas específicas é de extrema importância, uma vez que essa distribuição pode impactar diretamente o tempo de desenvolvimento e a qualidade do produto final, como apontado por Filho et al. (2022). Nesse contexto, os gerentes/gestores de projeto que realizam alocação de pessoas se esforçam para identificar o melhor profissional disponível para executar as tarefas pendentes, contribuindo assim para a obtenção de resultados positivos e de alta qualidade.

Segundo Filho et al. (2022) no processo da alocação de desenvolvedores a tarefas, o mapeamento do conhecimento de um desenvolvedor é extremamente desafiador, devido às frequentes atualizações desses dados a cada nova implantação e modificação de código, especialmente em projetos ágeis. Como podemos observar, os projetos ágeis possuem o foco muitas vezes na entrega de uma quantidade maior de funcionalidades em um curto espaço de tempo, com pouca ênfase na documentação, buscando ainda manter a qualidade do projeto em níveis elevados.

Além disso, existem ainda outras questões a considerar. Por exemplo, pode haver muitos desenvolvedores na mesma empresa com conhecimento sobre diferentes tecnologias ou que estejam trabalhando em múltiplos projetos simultaneamente. Caso o gerente seja novo na empresa ou no projeto, talvez ele não conheça bem todos os desenvolvedores e suas competências. Assim como em fábricas de software, onde é comum encontrar projetos e tarefas de natureza muito distintas umas das outras, o que dificulta a identificação do profissional mais adequado para cada uma delas.

Outro problema está relacionado à falta de tempo disponível por parte do gerente para realizar a alocação, uma vez que é necessário analisar as características e conhecimentos necessários para a atividade em que o desenvolvedor será alocado. A escassez de disponibilidade de desenvolvedores também afeta a tarefa de alocação, pois pode ocorrer que o gerente tenha mapeado o conhecimento de um desenvolvedor, mas ele esteja ocupado, sendo necessário identificar outro desenvolvedor que possua conhecimento semelhante.

Outra questão levantada em Arapakis et al. (2009) é a mudança de escopo da *sprint* atual, por exemplo, onde novas tarefas são incluídas e outras removidas, exigindo uma realocação mais rápida do time. Alterações na equipe, como demissões, transferências e contratações, também podem impactar a alocação de desenvolvedores para tarefas. Esses são apenas alguns

dos problemas que os gerentes podem enfrentar ao alocar desenvolvedores para tarefas

A literatura acadêmica relacionada tem enfatizado a importância de se desenvolverem abordagens estruturadas para a alocação de tarefas, que levem em consideração não apenas as competências individuais dos profissionais, mas também a composição equilibrada do grupo como um todo.

Com o intuito de auxiliar os gestores de empresas nessa atividade complexa, propomos neste trabalho um processo de construção de Sistema de Recomendação (SR) de desenvolvedores para projetos e tarefas com base no conhecimento acumulado por cada profissional especialista. Esse processo serviu de base para a implementação de um protótipo de SR capaz de sugerir os desenvolvedores mais qualificados para alocação a tarefas específicas. O sistema se baseia na análise do histórico de tarefas já executadas por cada desenvolvedor, permitindo identificar e recomendar aqueles com maior *expertise* nas áreas relevantes.

Espera-se que esse processo geral seja um guia na criação de SR eficazes para auxiliar gestores na identificação dos desenvolvedores mais adequados para cada tarefa, levando em consideração seus conhecimentos prévios e experiências passadas. Essa abordagem tem o potencial de otimizar a alocação de recursos e contribuir para o sucesso do projeto de desenvolvimento de software como um todo.

1.1 OBJETIVOS DO TRABALHO

Nesse contexto, este trabalho teve como objetivo geral propor um processo para criação de sistemas de recomendação de especialistas para auxiliar os gerentes na alocação eficiente de especialistas a tarefas em projetos de desenvolvimento de software, com foco na melhoria da qualidade do produto final. O processo proposto foi fundamentado em conceitos e abordagens consolidadas da área de sistemas de recomendação.

1.1.1 Objetivos Específicos

1. Propor um modelo para a construção de Sistemas de Recomendação de Especialistas visando a alocação em tarefas, fundamentado em seu histórico de conhecimento.

2. Implementar um sistema de recomendação de especialista conforme o modelo, voltado para a alocação de desenvolvedores a tarefas e projetos de desenvolvimento de software,

considerando o conhecimento específico necessário para cada atividade.

3. Realizar a coleta e o pré-processamento de dados dos desenvolvedores, a fim de mapear o conhecimento e as habilidades de cada especialista.

4. Validar o modelo proposto por meio de um estudo de caso em um ambiente de desenvolvimento de software ágil, onde as atualizações de conhecimento dos desenvolvedores são frequentes e o foco está na entrega de funcionalidades em curtos períodos.

1.2 TRABALHO REALIZADO

Objetivando auxiliar os gestores de empresas nessa atividade complexa, propomos aqui um processo de recomendação de desenvolvedores para projetos e tarefas com base no conhecimento acumulado por cada especialista. Este trabalho buscou atender a demanda por abordagens estruturadas para a alocação de tarefas seguindo conceitos e abordagens consolidadas da área de sistemas de recomendação, observando em especial as abordagens da área de recomendação de especialistas.

Segundo conceitos da área de recomendação de especialistas, podemos ver que nosso trabalho optou pela técnica de recomendação baseada no *modelo orientado a tópico*, que busca identificar especialistas em um determinado tópico através dos trabalhos produzidos pelo especialista (ver Capítulo 3). Assim, nosso processo se baseia na análise do histórico de tarefas já executadas por cada desenvolvedor, permitindo identificar e recomendar aqueles com maior expertise nas áreas relevantes.

Nosso trabalho adotou uma *estratégia personalizada de recomendação* (ver Capítulo 2). Contudo, nosso contexto tem particularidades importantes a considerar. Aqui, o usuário do sistema é o *gerente* responsável pela alocação e os *desenvolvedores* são vistos como os "itens" que serão recomendados. Assim, o perfil do usuário será representado pela descrição da tarefa a ser realizada no momento, sendo assim considerado um *perfil efêmero*. Já os desenvolvedores, tratados aqui como os "itens" a recomendar, são representados pela descrição dos trabalhos e atividades que cada um já realizou (dados históricos).

Por fim, adotamos a técnica de recomendação baseada em *filtragem de conteúdo*, que analisa a similaridade entre tarefas a realizar ("perfil efêmero" do usuário) e os dados históricos dos desenvolvedores ("itens"). Essa configuração de estratégia e técnica apresentou os

melhores resultados nos testes exploratórios realizados com diferentes implementações usando abordagens variadas.

Foi implementado um sistema protótipo para recomendação de especialistas no contexto de uma empresa de desenvolvimento de software. Esse sistema, implementado em Python, conta com três módulos:

(1) Módulo de Coleta de Informações: Esse módulo é responsável por se comunicar com o sistema de gestão de tarefas utilizado pela empresa, capturando informações sobre as tarefas realizadas pelos desenvolvedores, tais como implementação de novas funcionalidades, melhorias e correções de bugs. Além disso, o módulo coleta informações sobre as tarefas que devem ser realizadas no sprint atual, juntamente com os dados necessários, e as armazena em um banco de dados que será acessado pelos demais módulos. Esse módulo também é responsável por realizar o pré-processamento das informações recuperadas.

(2) Módulo de Recomendação: Esse módulo é responsável por realizar a recomendação de desenvolvedores para tarefas e projetos, seguindo a estratégia definida e utilizando os dados do banco de dados. O módulo processa os documentos que comprovam a experiência do desenvolvedor (por exemplo, commits realizados, tarefas executadas) e calcula a similaridade entre esses documentos do histórico de experiências do desenvolvedor e a tarefa que precisa de alocação.

(3) Módulo de Gerenciamento de Alocação: Esse módulo é responsável por gerenciar os registros do sistema, registrando e mantendo o controle das alocações de desenvolvedores para tarefas e projetos. Ele atua como a interface entre o usuário e os demais módulos, interagindo e integrando-se com eles. O módulo desempenha um papel fundamental na interação com o usuário, fornecendo acesso e controle sobre as funcionalidades do sistema. As tecnologias utilizadas incluem Django, Jira, NLTK, Pandas e Scikit-learn.

O protótipo foi testado durante um período de dois meses de uso na empresa parceira, onde foram coletados e cruzados dados referentes a quais desenvolvedores foram alocados pelo gerente para as tarefas de entrada, e quais deles foram recomendados pela ferramenta. Um total de 191 recomendações de atividades foram coletadas, as quais foram utilizadas na análise da validação do protótipo.

Na avaliação quantitativa, o protótipo alcançou uma precisão de 65,13% o que indica que os resultados não são aleatórios, e que o sistema tem potencial para melhorar o processo de alocação de desenvolvedores a tarefas, facilitando a busca pelo profissional mais adequado para cada atividade.

Além da avaliação quantitativa descrita acima, foram selecionadas e analisadas aleatoriamente 105 recomendações. A análise revelou que as tarefas relacionadas a relatórios de erros (bugs) obtiveram as melhores taxas de precisão na recomendação. Isto é, a maioria das recomendações feitas pelo sistema coincide com as alocações reais feitas pelos gerentes.

Destacamos também como principais contribuições um estudo abrangente sobre sistemas de recomendação de especialistas, analisando o cenário atual dessa área e sua relevância no contexto contemporâneo. Além disso, este trabalho apresenta vários trabalhos relevantes que descrevem outros sistemas de recomendação de especialistas, discutindo suas arquiteturas e modelos subjacentes. Também é proposto um processo específico para a criação de sistemas de recomendação de especialistas, desenvolvido com base nas necessidades e particularidades identificadas em um estudo realizado na empresa parceira, fornecendo um modelo sólido para a implementação desses sistemas.

Com os resultados obtidos, consideramos que os objetivos iniciais deste trabalho foram alcançados. Destacamos ainda que a solução proposta é genérica e fácil de se aplicar em outras empresas onde existe a necessidade de alocar profissionais a tarefas.

1.3 ORGANIZAÇÃO DO DOCUMENTO

Este documento é composto por mais quatro capítulos, além da Introdução.

- Capítulo 2: Apresenta conceitos e abordagens gerais sobre Sistemas de Recomendação, com detalhamento de estratégias e modelos de recomendação.
- Capítulo 3: Traz uma apresentação mais aprofundada sobre Sistemas de Recomendação de Especialistas e suas abordagens. São apresentados também alguns trabalhos relacionados, finalizando com uma comparação com este trabalho.
- Capítulo 4: Descreve detalhadamente o trabalho realizado, abrangendo o modelo de recomendação proposto e sua implementação. Apresenta, também, um estudo de caso realizado para validar a proposta implementada.
- Capítulo 5: Traz as conclusões, dificuldades, limitações e indicações de trabalhos futuros que podem se seguir a partir do que foi apresentado neste documento.

2 SISTEMAS DE RECOMENDAÇÃO – VISÃO GERAL

Como explicado por Morais et al. (2013), a sociedade atual está fortemente conectada à Internet. Graças a isso, uma quantidade abundante de dados é disponibilizada a todo momento, sejam eles filmes, vídeos, músicas, livros, produtos, dados financeiros, entre vários outros tipos de dados. Claramente, já não é possível estar ciente de tudo que é gerado e disponibilizado na Web.

Quando necessitamos de um serviço ou produto, geralmente procuramos recomendações de pessoas em quem depositamos nossa confiança, seja sobre um serviço que já contrataram, um produto que aprovaram, ou outros itens quaisquer. Considerando a quantidade abundante de informações disponíveis, encontrar algo novo que satisfaça nossas necessidades é uma atividade custosa, que demanda muito tempo e atenção. Para resolver (ou ao menos minorar) esse problema, foram desenvolvidos os Sistemas de Recomendação (SR). Através de recomendações personalizadas, tais sistemas ajudam seus usuários a identificar produtos ou serviços de seu interesse, poupando assim tempo e esforço.

Loh, Lichtnow e Garin (2002) descreve os SRs como proativos, diferenciando-se dos Sistemas de Busca (ou engenhos de busca), que são considerados sistemas meramente responsivos. Tais sistemas apenas respondem a consultas específicas dos usuários, retornando itens de informação previamente indexados que são considerados relevantes para uma dada consulta.

Os SRs vão além ao oferecer recomendações relevantes e personalizadas aos usuários, de forma proativa, sem que estes precisem fazer consultas específicas. Conforme mencionado por Akbar, Agarwal e Obaid (2022), os SRs são sistemas que buscam e retornam resultados de maior valor para o usuário dentro de um determinado domínio, com base em suas experiências e escolhas prévias. Para fornecer uma experiência satisfatória e resultados efetivos, os SRs geralmente adotam estratégias que beneficiem os usuários. Essas estratégias são fundamentais para aprimorar a capacidade do SR em fornecer recomendações personalizadas e de qualidade, atendendo às necessidades e preferências individuais dos usuários. Portanto, é comum que um SR implemente várias estratégias, as quais serão apresentadas na Seção a seguir.

2.1 ESTRATÉGIAS DE RECOMENDAÇÃO

Diversas estratégias podem ser utilizadas para recomendar itens a um usuário. Nesta Seção, apresentaremos as estratégias mais comuns, que nem sempre são mutuamente exclusivas, ou seja, um mesmo SR pode implementar várias dessas abordagens. É importante ressaltar que cada estratégia pode apresentar um grau de personalização diferente. De fato, algumas estratégias mais simples não possuem nenhum grau de personalização, como veremos a seguir.

Em relação ao grau de personalização, as estratégias de recomendação podem ser classificadas como:

- Não personalizada, quando a aplicação oferece a mesma recomendação para todos os usuários. Por exemplo, alguns sites de lojas adotam estratégias como mostrar a lista de itens mais vendidos no momento, ou sugestões de presentes para o dia das crianças;
- Efêmera: quando o sistema utiliza informações correntes sobre o usuário, como itens visualizados no site, produtos no carrinho de compras, entre outros;
- Persistente: que utiliza informações armazenadas sobre as preferências dos usuários - seu perfil. O perfil do usuário pode ser adquirido e mantido de duas maneiras não excludentes: (a) baseado em preferências implícitas, coletadas a partir do monitoramento do uso do sistema pelo usuário; ou (b) baseado em preferências explícitas, informadas pelo próprio usuário.

Portanto, a criação do perfil do usuário normalmente não é realizada nos SRs que adotam apenas a recomendação não personalizada.

Diferentes estratégias podem ser empregadas na recomendação de itens para usuários em SR. A seguir, será apresentada uma breve descrição das estratégias de recomendação mais utilizadas em diversos sistemas de recomendação.

- Listas de recomendação
  - Consiste em manter uma lista de itens organizados por alguma característica, como "Itens mais vendidos"ou "Ideias para presentes";
  - Essa abordagem não requer uma análise profunda dos dados dos usuários para montar as listas;

  - Suas vantagens incluem uma implementação fácil, porém a recomendação não é específica para um determinado usuário.

- Avaliações de usuários

  - Essa estratégia utiliza o feedback dos usuários sobre um determinado produto. Após "consumir" um item, os usuários escrevem um comentário sobre o produto. O sistema utiliza essas avaliações para estabelecer a reputação ou pontuação do item (por exemplo, por meio de estrelinhas);
  - A vantagem é que é fácil de implementar, e as avaliações de outros clientes podem influenciar novos consumidores;
  - No entanto, a desvantagem é que depende do usuário fornecer sua opinião sobre o item adquirido, além de ser necessário verificar a veracidade das opiniões fornecidas para evitar fraudes.

- Recomendações personalizadas

  - Nessa abordagem, as sugestões são feitas especificamente para cada usuário, com base no seu perfil;
  - O sistema deve manter dados e informações sobre o usuário, representando seu perfil, a fim de oferecer recomendações personalizadas.

- Recomendação baseada em associações entre itens

  - Essa estratégia de recomendação se baseia em técnicas que criam associações entre os itens oferecidos pelo SR. Quando o usuário compra ou visualiza um determinado item, o sistema indica outros itens "associados"a ele;
  - Duas técnicas comumente utilizadas para associar itens (formando um cluster ou um grafo de relações) são:
    * Associações entre itens adquiridos ou consumidos em conjunto por usuários do SR (“Usuários que se interessaram por X também se interessaram por Y”);
    * Associação com base no conteúdo do item, como sua descrição no sistema de recomendação (“O livro XX foi escrito pelo mesmo autor do livro YY que você está visualizando”).

Essas diferentes estratégias proporcionam uma ampla gama de opções para tornar as recomendações mais adequadas, relevantes e personalizadas aos usuários. Contudo, na Academia, a área de SRs continua em desenvolvimento. Porém, já existe algum consenso em relação a um modelo base, com processos bem definidos que serão apresentados na próxima Seção.

2.2 PROCESSO GERAL DE RECOMENDAÇÃO

É possível identificar três processos principais presentes nos SR: (1) Coleta de dados e Definição do perfil do usuário; (2) Técnicas de Recomendação; e (3) Avaliação do SR

A coleta de dados tem por objetivo obter informações relevantes sobre o usuário, a fim de criar um perfil que será usado no processo de recomendação. Existem várias abordagens para realizar a coleta de dados, bem como para representar o perfil dos usuários. Esse assunto será aprofundado na Seção 2.3.

Contudo, é importante esclarecer que nem todas as estratégias de recomendação utilizam perfil de usuário - isso depende do seu grau de personalização. A estratégia de recomendação pode ser classificada como: não personalizada (oferecendo a mesma recomendação para todos os usuários), efêmera (perfil temporário) ou persistente (perfil armazenado). Assim, a criação do perfil do usuário não é realizada nos SRs que adotam apenas a recomendação não personalizada.

A Seção 2.4 irá tratar das técnicas usadas na implementação do módulo de recomendação. A Seção 2.5 trata da avaliação do SR. Essa etapa é central para ajustar o sistema, a fim de que seja possível melhorar sempre as recomendações realizadas. Por fim, a Seção 2.6 traz algumas considerações finais.

2.3 COLETA DE DADOS E DEFINIÇÃO DE PERFIL DO USUÁRIO

Para ser possível realizar recomendações personalizadas a um usuário, é necessário criar uma representação interna do seu perfil. Esse perfil será utilizado posteriormente para filtrar os itens que serão recomendados.

A criação do perfil se baseia em dados coletados sobre o usuário (suas preferências). Essa coleta de dados pode ser realizada de duas formas, explícita ou implícita. Note que essas abordagens não são excludentes, existindo sistemas que fazem coleta híbrida.

A coleta explícita demanda algum esforço por parte do usuário, pois ele deverá fornecer

informações que ele considerar relevantes. Essa coleta pode ocorrer em dois momentos distintos: no primeiro uso do sistema (quando o usuário é solicitado a informar suas preferências e outros dados importantes para ele), ou durante o uso do sistema, por exemplo, quando o usuário fornece feedback sobre produtos que tenha consumido. Note que é possível utilizar as duas opções de coleta explícita em um mesmo sistema, a fim de manter o perfil do usuário sempre atualizado.

O problema com a coleta explícita se dá por questões de usabilidade, pois não são todos os usuários que dispõem de tempo ou vontade para preencher longos formulários, ou até mesmo para fornecer feedbacks com frequência. Esse comportamento diminui a quantidade e a qualidade das informações coletadas sobre o usuário. Porém, é importante que haja este esforço, para tentar diminuir o problema da "partida fria" (do inglês "*cold-start*"). Como descrito por Herce-Zelaya et al. (2020), esse é um problema que acontece com novos usuários, quando o sistema ainda não tem muita informação sobre suas preferências, o que prejudica a precisão das recomendações iniciais para esses usuários.

Na verdade, a "partida fria" é um problema que pode afetar todos os tipos de sistemas de recomendação. Já existem vários trabalhos que tentam minimizar esse problema como os trabalhos de Wei et al. (2017), Lam et al. (2008), Fu et al. (2019) e Tahmasebi et al. (2021), porém ainda não foi encontrada uma solução definitiva. Assim, esse é um problema que acabará afetando, mesmo que pouco, todos os sistemas de recomendação.

Na coleta implícita, por sua vez, dados são coletados durante a utilização do sistema pelo usuário sem que ele precise fornecer informação explicitamente Almonte et al. (2022). Alguns dados coletados podem ser, por exemplo, o tempo que o usuário passa assistindo a um filme, quais páginas ele visualizou, quais produtos o usuário teve interesse de inspecionar, local onde o usuário se locomove, entre outros. O objetivo é identificar quais são as necessidades e preferências com base no uso do sistema pelo usuário. Normalmente SR baseado/sensível em contexto utilizam essa abordagem, como nos trabalhos de Meehan et al. (2013) e Nawara e Kashef (2021).

Essa abordagem de coleta oferece a possibilidade de obter maior quantidade de dados sobre o usuário do que na coleta explícita. Porém, esse processo deve ser conduzido com cuidado, obedecendo a regras muito bem definidas, pois pode acarretar uma imprecisão maior com relação às informações armazenadas sobre o usuário. Note que o SR precisa fazer deduções a partir do uso do sistema, e geralmente essas deduções não passam por uma avaliação do próprio usuário, para confirmar se ele concorda ou não com o que foi inferido.

Com a definição de como será feita a coleta de dados, é importante ver quais serão as entradas que farão parte do perfil do usuário. A definição do perfil do usuário é algo extremamente importante, pois é o que representará as necessidades e preferências do mesmo no sistema de recomendação.

Segundo Barth (2010), a criação de perfil é uma etapa fundamental em SR para compreender as preferências e características individuais dos usuários. O autor exemplifica várias técnicas de criação de perfil, que envolvem a coleta de dados, preferências, criação de modelos e espaços vetoriais. Após análise dos trabalhos de Barth (2010), Arapakis et al. (2009), Middleton, Shadbolt e Roure (2003) e Middleton, Shadbolt e Roure (2004), podemos listar a existência dos seguintes tipos e técnicas de criação de perfil que podem ser empregadas:

- Perfil baseado em preferências *explícitas*: nesse tipo de perfil, as preferências dos usuários são informadas diretamente por eles, por meio de avaliações, classificações ou respostas a questionários;
- Perfil baseado em preferências *implícitas*: nessa abordagem, as preferências são inferidas a partir do comportamento e das interações do usuário com o sistema. Dados como histórico de navegação, itens visualizados, tempo gasto em determinadas páginas ou cliques em itens são analisados para capturar as preferências implícitas do usuário;
- Perfil baseado em conteúdo: aqui, o perfil do usuário é criado com base nas características dos itens recomendados. A análise é feita considerando informações como descrições, tags, categorias ou atributos dos itens. O sistema identifica padrões e associações entre os itens e as preferências dos usuários;
- Perfil social ou de associação: esse tipo de perfil se baseia nas conexões sociais e nas interações entre os usuários. O sistema analisa a atividade de usuários similares ou de usuários com interesses em comum para fazer recomendações personalizadas. A ideia é aproveitar a sabedoria coletiva e as influências sociais para melhorar as recomendações.

Resumidamente, a criação de perfil em SR envolve a coleta de informações sobre as preferências dos usuários, seja por meio de respostas explícitas ou inferências implícitas. Para criar esses perfis, várias técnicas podem ser utilizadas, como filtragem colaborativa, filtragem baseada em conteúdo, análise de sentimentos, aprendizado de máquina e algoritmos de agrupamento. Cada técnica tem suas vantagens e desafios, e a escolha depende do domínio do sistema e dos recursos disponíveis, falaremos sobre elas na Seção seguinte.

2.4 TÉCNICAS DE RECOMENDAÇÃO

Atualmente não existe um sistema de recomendação que atenda a todos os gostos e necessidades, sendo necessário definir bem qual é o problema que queremos sanar com o sistema de recomendação. Dependendo de qual seja a necessidade, a técnica de recomendação irá variar, para poder atender melhor a necessidade.

Dando continuidade ao assunto visto na Seção anterior, esta Seção apresenta algumas técnicas mais populares na construção de SRs personalizados (com personalização efêmera ou persistente). Como será visto, não existe uma técnica que seja superior às outras em todos os aspectos importantes dos SR. Assim, geralmente a técnica a ser usada na implementação do sistema vai depender das necessidades e características do sistema desejado.

Destacamos aqui quatro técnicas mais utilizadas na construção de SR personalizados:

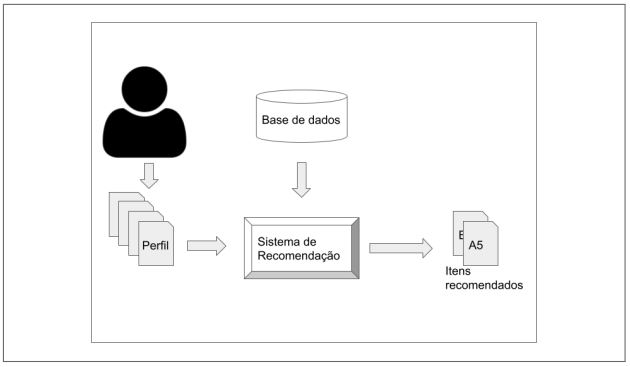
- Recomendação baseada em filtragem de conteúdo;
- Recomendação baseada em filtragem colaborativa;
- Recomendação baseada em filtragem híbrida;
- Recomendação baseada em contexto.

2.4.1 Recomendação baseada em conteúdo

Os sistemas baseados nessa técnica realizam a recomendação buscando na base de itens os que são mais similares ao perfil do usuário. Aqui, o perfil do usuário e os itens são descritos a partir de alguns atributos considerados relevantes para a aplicação – por exemplo, um SR de filmes poderá ter atributos como título do filme, gênero, atores, diretor, entre outros. Como ilustrado na Figura 1, o SR baseado em conteúdo possui apenas dois atores, o usuário (representado no sistema pelo seu perfil) e o sistema de recomendação.

Segundo Dat, Toan e Thanh (2022), essa abordagem é provavelmente a mais utilizada na construção de SRs, por ser eficaz e possuir uma alta eficiência no início do uso do SR (partida fria), já que ela se baseia apenas na descrição dos itens e no perfil do usuário. Assim, o SR será capaz de indicar todos os itens da base que sejam relevantes para o usuário. A fim de evitar recomendações repetidas, esses sistemas podem guardar a lista de itens já consumidos pelo usuário.

Figura 1 – Atores em um sistema de recomendação com filtragem baseada em conteúdo



Fonte:Elaborado pelo autor(2022)

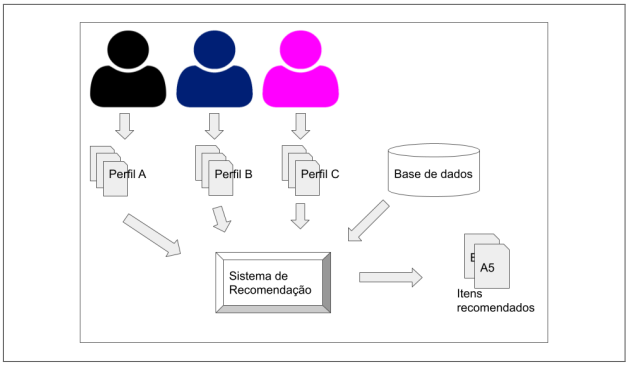
Contudo, apesar da sua simplicidade e eficácia, essa técnica apresenta algumas desvantagens. O SR baseado em conteúdo não irá recomendar itens que possam ser de interesse do usuário, porém com atributos que diferem do seu perfil atual, dificultando assim a indicação de novos itens de valor para o usuário. Considerando novamente a recomendação de filmes, e supondo que o usuário tenha indicado que gosta de filmes românticos e de comédia, um filme de super-heróis de que o usuário possa gostar nunca será recomendado pelo sistema, dado que o gênero do filme não se encontra no seu perfil. Esse comportamento é conhecido como superespecialização, ou "efeito portfólio".

2.4.2 Recomendação baseada em filtragem colaborativa

A Filtragem colaborativa é outra técnica de recomendação bastante popular. Essa abordagem parte do princípio de que pessoas próximas, ou que participam do mesmo círculo social, tendem a gostar das mesmas coisas. Diferentemente da técnica baseada em conteúdo, onde apenas o perfil do usuário é considerado na recomendação, a filtragem colaborativa considera os perfis de seus semelhantes, possibilitando assim que o usuário "descubra" novos itens de seu interesse, porém que não "casem" perfeitamente com seu perfil. Como representado na figura 2, esse tipo de SR tem vários atores: os usuários (representados pelos seus perfis) e o sistema.

Essa técnica não apenas trabalha com uma descrição explícita dos atributos de itens e usuários. A filtragem colaborativa pode ser dividida em dois tipos, de acordo com Gupta e Katarya (2019), a filtragem colaborativa baseada no usuário, exemplificada no trabalho de Wang e Ye (2009), e a filtragem colaborativa baseada em itens, demonstrada no trabalho de

Figura 2 – Atores em um sistema de recomendação com filtragem colaborativa



Fonte:Elaborado pelo autor(2022)

Zhang et al. (2009).

A filtragem colaborativa baseada no usuário é uma técnica de recomendação em que as recomendações dependem dos vizinhos do usuário. Por exemplo, se um usuário tem um vizinho que gosta de um determinado filme X, o SR irá sugerir ao usuário o mesmo filme X. Nesse tipo de SR, as recomendações são feitas exclusivamente com base nos vizinhos do usuário. O perfil de cada usuário é representado pelos itens que ele consumiu, juntamente com as notas ou avaliações que ele atribuiu a cada item consumido. Com base nessas informações, o SR pode criar grupos (clusters) de usuários semelhantes. Quando um usuário solicita uma recomendação, o sistema retorna itens que foram bem avaliados pelos usuários semelhantes (vizinhos) a ele e que ainda não foram consumidos por esse usuário. Dessa forma, essa técnica é capaz de recomendar "novidades", ou seja, itens um pouco diferentes daqueles que já foram consumidos, mas que podem ser de interesse para o usuário.

A principal desvantagem da técnica baseada no usuário é que pode ocorrer o caso em que um usuário goste de um item Y, mas nenhum de seus vizinhos tenha gostado ou consumido esse item. Portanto, o SR não recomendará o item Y, já que nenhum vizinho o consumiu. Além disso, essa técnica é conhecida por enfrentar frequentemente problemas de escassez de dados (*sparse of data*) (ZHANG et al., 2009).

Outra técnica é a filtragem colaborativa baseada em itens. Nessa abordagem, o SR dá mais importância aos itens do que aos perfis de vizinhos do usuário. A principal diferença entre essa técnica e a anterior (baseada no usuário) é que a recomendação é feita com base naquilo que um grupo de usuários considera interessante, mesmo que esses participantes do grupo não tenham alta similaridade entre si. Segundo Zhang et al. (2009), a técnica baseada em itens parte da suposição de que: se a maioria dos usuários classificar um conjunto de itens

de maneira semelhante, o usuário-alvo também classificará esses itens de maneira semelhante. Nessa técnica, são exploradas primeiramente as relações entre os itens com base nas avaliações recebidas (e não a similaridade de conteúdo), em vez das relações entre os usuários. Uma maneira de aplicar essa técnica, como demonstrado por Sarwar et al. (2001), é escolher itens que possuam notas semelhantes aos itens selecionados pelos vizinhos do usuário.

Contudo, técnicas de filtragem colaborativa, em geral, também apresentam problemas, como a já mencionada "partida fria", quando o sistema ainda não tem dados históricos sobre itens e usuários. Existem também os casos de usuários novos ou pouco frequentes no uso do sistema, além daqueles que não avaliam os itens consumidos. A ausência de informação sobre esse tipo de usuário dificulta a identificação de perfis semelhantes ao seu. Por fim, existem ainda os usuários incomuns, ou "usuários fora da curva" (do inglês, *outliers*), que não se parecem com ninguém na base do sistema.

Outro problema dessa técnica é a sua escalabilidade. À medida que novos usuários e novas interações são registradas, encontrar perfis semelhantes acaba se tornando mais custoso. Porém, há formas de diminuir esse problema de escalabilidade, como descrito por Li (2022), utilizando *clusters* diferentes para um usuário. Nesse trabalho, o autor sugere vincular o usuário a mais de um grupo diferente de usuário, esses grupos possuindo uma menor semelhança entre os participantes, para diminuir o tempo de pesquisa por usuário com perfis semelhantes.

2.4.3 Recomendação baseada em filtragem híbrida

A técnica de recomendação híbrida é uma mescla das duas técnicas anteriores, buscando aproveitar as vantagens de cada uma delas e eliminar seus problemas. Assim, essa abordagem possibilita recomendações melhores, descoberta de novos relacionamentos entre usuários e recomendações personalizadas. Esses sistemas híbridos também apresentam bons resultados para usuários incomuns (outliers) e uma precisão maior independente do número de perfis cadastrados.

Como descrito em Ko et al. (2022), essa abordagem surgiu para solucionar os problemas das abordagens anteriores. De forma geral, o principal objetivo é compensar a falta de dados (escassez de dados), ao integrar as informações dos modelos de filtragem baseada em conteúdo e filtragem colaborativa. Os autores citam que essa abordagem pode ser dividida em sete tipos, e cada um deve ser utilizado conforme o objetivo da recomendação e os dados colhidos. Os tipos de hibridização sugeridos pelos autores são:

- Hibridização Ponderada: ajusta o peso (pode ser entendido como pontuação de relevância) que é dado a um item gradualmente conforme a relação entre a avaliação do usuário sobre um item e a avaliação inferida pelo SR;
- Hibridização Comutada: altera o tipo do modelo utilizado para a recomendação, dependendo da situação;
- Hibridização em Cascata: a princípio, utiliza um dos modelos de sistema de recomendação, criando um conjunto de perfis semelhantes ao do usuário principal, e então utiliza outro modelo de sistema para ordenar o conjunto de perfis semelhantes na ordem dos itens mais semelhantes à preferência do usuário principal;
- Hibridização Mista: ao serem realizadas muitas recomendações simultaneamente, é utilizada a filtragem baseada em conteúdo para recomendar itens com base no conteúdo e não no feedback do usuário e íntegra dados do histórico coletados durante o uso do sistema;
- Combinação de Recursos: o modelo de filtragem é escolhido conforme o tipo do dado;
- Aumento de Recursos: aqui dois tipos de SR são utilizados, um para calcular a pontuação ou item, depois outro SR utiliza essa classificação para recomendar;
- Meta-nível: utiliza todos os metadados de um modelo de SR como dados de entrada para outro modelo de SR. Como as preferências do usuário são compactadas e expressas usando o Meta-Level, é mais fácil aplicar a filtragem colaborativa do que quando os dados brutos são utilizados como entrada.

Os tipos de hibridização que o autor apresenta não são necessariamente os únicos existentes, visto que qualquer sistema de recomendação que combine mais de uma técnica de recomendação será considerado híbrido. Contudo, com os tipos citados podemos ter visão global dos tipos de SR híbridos que existem atualmente.

2.4.4 Recomendação baseada no contexto - ou sensível ao contexto

A recomendação baseada em contexto, diferentemente das anteriores, que priorizam os padrões de preferências do usuário, tem como foco o contexto do usuário no momento. Assim, essa técnica possibilita recomendações mais personalizadas e próximas à realidade do usuário.

Segundo Dey, Abowd e Salber (2001), o contexto pode ser definido como qualquer informação que possa ser usada para caracterizar a situação de uma entidade.

Segundo Barbosa (2019), os sistemas de recomendação podem ser acessados sob diversos contextos diferentes e é essencial que esse contexto seja considerado, para evitar que a recomendação não seja útil para o usuário. Para ilustrar a importância do contexto, o autor apresenta a seguinte situação: um usuário necessita chegar o mais rápido possível a um hospital; ao acessar o sistema de recomendação de rotas, esse sistema deve perceber o contexto no qual o usuário está (se está andando ou em um automóvel) e realizar a recomendação da rota mais curta para chegar ao hospital conforme o modo de deslocamento do usuário.

2.5 AVALIAÇÃO DO SR

A avaliação de um SR pode ser realizada tanto durante a fase de implementação quanto após o seu lançamento, quando os usuários estão utilizando a ferramenta. Durante a avaliação na fase de implementação, um conjunto de dados de teste é utilizado para validar se as recomendações feitas pelo SR são adequadas ao perfil do usuário ou estão em conformidade com recomendações esperadas feitas por outro modelo de SR, ou separadas manualmente. Essa avaliação inicial tem como objetivo validar o modelo escolhido e realizar ajustes, se necessário. Nesse processo, medidas como acurácia, precisão e recall, conforme descrito por Vargas e Castells (2011), são comumente utilizadas. Em resumo, a acurácia mede a precisão geral das recomendações, a precisão se concentra nas recomendações positivas corretas feitas pelo SR, e o recall avalia a capacidade de recuperar corretamente os itens relevantes. A combinação dessas medidas oferece uma visão abrangente do desempenho do SR.

A avaliação após o lançamento do sistema pode ser considerada como a avaliação dos itens recomendados. Essa avaliação pode ser realizada de forma explícita, em que o usuário avalia diretamente as recomendações recebidas atribuindo uma nota ou indicando se gostou ou não do item recomendado (conhecido como feedback). Além disso, a avaliação pode ocorrer de forma implícita, por meio do uso ou consumo das entidades recomendadas. Por exemplo, se o sistema recomendar um filme e o usuário assistir até o final, ou se o sistema recomendar uma rota e o usuário seguir essa rota até chegar ao destino, o sistema pode considerar que a recomendação foi bem-sucedida.

A avaliação dos resultados da recomendação é de extrema utilidade para avaliar se a técnica utilizada e os dados do SR estão de acordo com as expectativas e se o sistema está alcançando

os objetivos propostos.

2.6 CONSIDERAÇÕES FINAIS

Este capítulo foi dedicado a explanar sobre o funcionamento e a utilidade de sistemas de recomendação. Descrevemos como é o processo de coleta e armazenamento de dados, as estratégias de recomendação, como é definido o perfil do usuário (que depende da técnica adotada na construção do sistema). Descrevemos as técnicas principais de recomendação atualmente (recomendação baseada em conteúdo, filtragem colaborativa, filtragem híbrida e a filtragem com foco no contexto), bem como suas vantagens e desvantagens. Por fim, destacamos a importância da avaliação da recomendação para o bom funcionamento do sistema.

Este capítulo nos proporcionou uma visão global sobre o que são os sistemas de recomendação. No próximo capítulo, iremos aprofundar ainda mais nos sistemas de recomendação de especialistas

3 SISTEMA DE RECOMENDAÇÃO DE ESPECIALISTAS

Como vimos, o Capítulo 2 apresentou conceitos gerais sobre Sistemas de Recomendação. Já este capítulo vai focar em uma área da recomendação específica relacionada ao tema deste trabalho: recomendação de especialistas.

A recomendação de especialistas, como descrito por Nikzad–Khasmakhi, Balafar e Reza Feizi–Derakhshi (2019), é uma área emergente, que tem como finalidade localizar e recomendar pessoas com um grande conhecimento em uma determinada área, os chamados de especialistas. De modo geral, o sistema recebe como entrada, por exemplo, um tópico relevante para o usuário ou uma query sobre um assunto que o usuário está buscando naquele momento. A partir daí, o sistema irá buscar compatibilidade entre os especialistas e suas áreas de conhecimento com a entrada do usuário, retornando uma lista de especialistas no assunto desejado, geralmente ranqueada pelo grau de relevância/conhecimento do especialista naquele assunto.

Para sistemas de recomendação de especialistas, uma atividade importante é exatamente a definição do "especialista". Porém, de acordo com Moreira e Wichert (2013), definir um especialista é algo difícil. Os autores consideram que "quantificar" áreas de especialização é uma tarefa difícil. Além disso, as experiências que os especialistas têm sobre um assunto podem variar ao longo do tempo. Por isso, muitos trabalhos seguem um caminho de definir uma pontuação para as áreas de especialização (*expertise*), para o relacionamento do especialista com os colegas e sua relevância em um dado ambiente onde compartilha seus conhecimentos. Voltaremos a esse assunto no próximo capítulo.

Sistemas de recomendação de especialistas apresentam uma arquitetura muito similar à de SRs gerais. Contudo, existem pontos específicos que devem ser considerados e bem definidos durante o seu desenvolvimento. As seções seguintes apresentam características de SR de especialistas.

3.1 PROCESSO DE CONSTRUÇÃO DE SR DE ESPECIALISTAS

Em geral, o processo de construção de um SR de especialistas conta com três etapas:

- Definição da técnica de recomendação a ser utilizada;
- Criação da base de dados do SR;

- Implementação do módulo de recomendação de especialistas.

3.1.1 Técnica de recomendação de especialistas

Em geral, os SRs de especialistas se baseiam em duas técnicas de recomendação diferentes, uma que foca no score/pontuação do especialista em uma área e outra técnica cujo foco são os trabalhos produzidos pelo especialista. Neste trabalho, adotamos a nomenclatura sugerida por Nikzad–Khasmakhi, Balafar e Reza Feizi–Derakhshi (2019), que chama a primeira técnica de "modelo orientado a especialidade (expertise)" e a segunda técnica de "modelo orientado a tópico", sendo que cada modelo possui suas vantagens e desvantagens.

Como descrito por Alarfaj et al. (2012), em geral, no modelo orientado a especialidade, os SRs criam uma representação dos especialistas e em seguida ranqueiam essas representações com base na busca do usuário. SR baseados nesse modelo realizam a escolha do especialista em uma área através do *score* do especialista. Essa medida pode ser calculada de várias formas, a depender do ambiente. De forma geral, essa pontuação pode ser calculada a partir da quantidade de trabalhos em uma área, quantas vezes uma resposta ou trabalho do autor foi citado, ou se ele recebeu alguma nota de *feedback* em seu trabalho.

Muitos trabalhos consideram apenas o *score* da reputação do especialista, que pode ser calculada com base na relação desse profissional em uma rede de especialistas e seus trabalhos, onde especialistas de maior impacto na área tendem a ter uma reputação melhor. Esse modelo é bom para encontrar especialistas em uma área, quando não há um foco específico em alguma característica da área. As principais vantagens deste modelo são: o modelo consegue recuperar vários especialistas em resposta a uma consulta, e também consegue realizar uma representação mais abrangente do conhecimento do especialista. Contudo, como já mencionado, ele não é bom para identificar profissionais em um foco específico de expertise.

Já o modelo orientado a tópico, como descrito em Lin et al. (2017), tem por objetivo encontrar um especialista em um determinado tópico através de seus trabalhos. Em geral, nesse modelo o SR busca por trabalhos que melhor "casam" (*match*) com uma consulta ou com outro documento de entrada, procurando então os especialistas citados nesses trabalhos. Esse modelo, diferentemente do anterior, tem a vantagem de ser mais específico, conseguindo retornar especialistas em uma ampla área do conhecimento que trabalhem de uma forma mais aprofundada em uma determinada área.

Este trabalho de mestrado adotou o modelo orientado a tópico, por ser o mais adequado

ao nosso contexto (ver Capítulo 4).

3.1.2 Criação da base de dados do SR

Para que o SR possa funcionar, é preciso criar uma base de dados que consiga representar o especialista e seus conhecimentos, como descrito no Capítulo 2 seção 2.3. Para isto, é necessário coletar essas informações de diversas fontes, que podem utilizar formatações diferentes, com características particulares Alarfaj et al. (2012). Em alguns casos, pode haver a necessidade de utilizar técnicas de extração de informação [1] nessa coleta como no trabalho de Sundermann (2015) .

De modo simples, podemos dizer que as informações podem estar representadas de duas formas: estruturada ou não estruturada. Quando a informação desejada já está estruturada (e.g., em tabelas), ou semiestruturada a partir de *tags* metadados, e tratada em alguma base de dados externa, basta baixar esses dados e processá-los de acordo com a necessidade do sistema atual. Porém, muitas vezes a informação desejada não está estruturada (i.e., não apresenta uma formatação regular), e está distribuída em várias páginas e bases diferentes. Nesse caso, a extração da informação desejada é mais complexa, necessitando de técnicas mais sofisticadas de extração.

Analisando agora as fontes de informação, podemos pensar em quatro aspectos importantes a considerar Alarfaj et al. (2012):

- **O local onde a informação está armazenada:** se a informação está armazenada off-line, em alguma base/repositório local acessível, é possível baixar essa informação criando assim uma nova base de dados; porém se a informação desejada está online e é atualizada periodicamente, pode haver a necessidade de desenvolver um crawler [2] (um "robô" que recolhe constantemente novas informações da fonte online).

- **A estrutura da fonte de informação:** a fonte pode ser estruturada, quando há um padrão para armazenar a informação (e.g., um BD); ou ter uma estrutura livre, como em fóruns ou em certas páginas web, onde as informações não apresentam um padrão específico que seja facilmente reconhecido.

[1] Information extraction - https://en.wikipedia.org/wiki/Information_extraction
[2] O que é um Crawler? - https://www.crawly.com.br/blog/o-que-e-crawler-robos-para-coleta-de-dados

- **A quantidade de informações da fonte:** esse é outro aspecto importante para se observar no caso dos SR de especialistas. Geralmente, quando a quantidade de dados é pequena, isso pode dificultar a categorizar o conhecimento de um especialista. Contudo, esse problema também irá depender da qualidade e especificidade da informação.
- **A formalidade de escrita da informação:** a informação pode estar escrita de modo formal, mais organizada, objetiva e sem características pessoais - como jornais, trabalhos acadêmicos e livros didáticos. Já a escrita informal é mais pessoal e difícil de processar, dificultando assim a categorização das áreas de conhecimento do especialista.

Os aspectos citados acima devem ser analisados e tratados durante a criação da base de dados do SR, a fim de não comprometer a seleção de fontes de dados apropriadas para recomendar os especialistas. Em alguns casos, fica claro que os dados obtidos diretamente das fontes ainda precisam ser tratados e pré-processados antes do seu uso na recomendação de um especialista. Esse pré-processamento é muito importante para a etapa final de recomendação, independente da técnica de recomendação adotada, pois é aí que os dados serão formatados, os dados desnecessários serão retirados e novos dados derivados da base de dados poderão ser inseridos.

O pré-processamento depende muito da necessidade de cada SR. Etapas de pré- processamento geralmente realizadas são a remoção de *stopwords*, *stemming* e a utilização de um tesauro para expansão de informação.

- *Stopwords* são palavras que não agregam valor semântico ao texto (e.g., artigos, preposições etc) Silva e Ribeiro (2003), e são geralmente removidas para diminuir o tamanho dos textos, facilitando assim seu armazenamento e seu uso.
- O processo de *stemming*, segundo Pinheiro et al. (2022), objetiva reduzir as palavras ao seu radical ("stem"), por exemplo, realização=realiza; realizado=realiza. Assim, palavras que têm a mesma raiz semântica são agrupadas, aumentando a relevância semântica dentro da coleção de dados.
- Um tesauro é basicamente um dicionário de sinônimos, e pode ser representado como lista estruturada de termos associados e suas inter-relações em um determinado ramo do conhecimento Laan e Ferreira (2000). Essa lista pode ser utilizada para expandir um documento ou uma consulta (*query*) Qiu e Frei (1993), ou pode ser usado para padronizar a terminología usada nos textos, como descrito por Carvalho (2021).

3.1.3 Recomendação de especialistas

Por se tratar de recomendações, mesmo que de especialistas, as abordagens apresentadas agora são especializações e aplicações diretas das abordagens descritas na Seção 2.4 do Capítulo 2.

Como mencionado anteriormente na Seção 3.1.1, existem geralmente duas técnicas para recomendar especialistas. A primeira é o modelo orientado à especialidade (expertise) (Alarfaj et al. (2012), Chun e Hong (2001), Davoodi, Afsharchi e Kianmehr (2012), Sousa, Dias e Pinto (2019)), que se concentra em medir ou deduzir a expertise dos especialistas através do cálculo de uma pontuação (score) em uma área específica de conhecimento (ALARFAJ et al., 2012). Há diversas formas de definir a especialidade e o nível de conhecimento do especialista nessa área. Algumas das abordagens mais comumente usadas incluem: o modelo baseado em regras, que é o mais simples, mas requer um trabalho manual extenso para definir as regras, também pode se utilizar de algoritmos de árvores para a inferência das regras; a análise de network, que calcula um score para o especialista com base nos relacionamentos entre especialistas e nas áreas de conhecimento em que os outros já possuem expertise (geralmente seguindo o modelo de filtragem colaborativa); pontos de reputação, que são scores calculados com base no feedback de trabalhos anteriores e também podem incluir o número de citações recebidas pelo especialista; algoritmos de similaridade, que busca encontrar trabalhos semelhantes ao do especialista e calcular um score de conhecimento com base nessa similaridade; o uso de algoritmos de aprendizagem profunda (*deep learning*) ou outros algoritmos de aprendizado de máquina, que têm como objetivo criar modelos que representem o especialista e seu conhecimento com base nos trabalhos que ele realizou. Vale ressaltar que nenhuma das abordagens é excludente de outra, e mais de uma pode ser utilizada simultaneamente. Além disso, existem vários outros métodos que podem ser empregados para calcular o score ou representar o conhecimento do especialista.

O modelo orientado a tópico (Matter, Kuhn e Nierstrasz (2009), Albusac et al. (2018)) tem como objetivo recomendar autores ou especialistas cujo trabalho possua similaridade com o conhecimento desejado. Nesse modelo, o foco do processamento deixa de ser o cálculo do conhecimento do especialista e passa a ser a análise de seus trabalhos (geralmente seguindo o modelo de filtragem baseada em conteúdo), o que se assemelha à indexação e recuperação de documentos. Esse tipo de recomendação permite encontrar especialistas que possuam conhecimento específico em uma ampla área. Para isso, os trabalhos do especialista são salvos

para formar uma base de dados que representa seu conhecimento. Assim, para recomendar um especialista, o sistema retornará aqueles que possuem trabalhos/documentos que melhor correspondam à consulta, sendo essa consulta uma query ou outro documento. Existem diversos algoritmos para calcular a similaridade entre documentos, como a similaridade do cosseno, que é uma medida para calcular a semelhança entre dois vetores ou documentos com base em sua orientação no espaço vetorial. Outro exemplo é o algoritmo BM25 ou cálculo de similaridade de BM25, que utiliza uma função de pontuação baseada na frequência do termo no documento, frequência do termo na query e comprimento do documento para retornar documentos relevantes para a query.

A seleção da abordagem a ser utilizada dependerá do problema e do ambiente ao qual o sistema estará integrado. A avaliação do conhecimento do especialista é um ponto importante, pois é por meio dela que serão realizadas as recomendações dos especialistas.

3.2 TRABALHOS RELACIONADOS

Esta seção apresenta quatro trabalhos de recomendação de especialistas na área da engenharia de software, sendo considerados como trabalhos relacionados à nossa pesquisa. Mais precisamente, são trabalhos que descrevem SR de desenvolvedores para projetos ou atividades em projetos de SW: Silva, Felício e Paixão (2018), Matter, Kuhn e Nierstrasz (2009), Filho et al. (2022) e Anvik, Hiew e Murphy (2006).

O trabalho de Silva, Felício e Paixão (2018) tem como objetivo verificar a possibilidade de alocação de desenvolvedores para projetos de software, com base em algoritmos de recomendação. Para isso, os autores analisaram 3 algoritmos de recomendação que consideram o histórico dos projetos em que o desenvolvedor esteve alocado em uma fábrica de software, formando assim uma matriz de "projeto-desenvolvedor". A recomendação dos desenvolvedores é realizada através da análise do preenchimento e cálculo de proximidade na matriz, técnicas de fatoração de matrizes e filtragem colaborativa. Como citado pelos autores, o projeto utiliza uma ferramenta em Java para recomendações, a LibRec [3]. Para avaliar a implementação desenvolvida, os autores utilizaram um banco de dados construído com projetos disponíveis no Github, e usaram a técnica de validação cruzada. Para verificar a precisão do SR, foi analisada a Área Sob da Curva ROC. A curva ROC (*Receiver Operating Characteristic*[4] é um gráfico

[3] LibRec - https://guoguibing.github.io/librec/
[4] Curva ROC - https://en.wikipedia.org/wiki/Receiver_operating_characteristic

que ilustra o desempenho de um algoritmo de classificação binária variando-se o limiar de discriminação entre instâncias positivas e negativas, onde valores obtidos acima de 0.5 indicam que a recomendação não é aleatória. A Área sob a curva ROC (AUC) indica se as instâncias positivas estão ranqueadas acima das negativas (quanto maior a área, melhor o ranking). Com a avaliação, os autores atentaram que os algoritmos foram eficientes nas recomendações, obtendo um valor de 0.76 na Curva ROC, indicando que as recomendações não eram aleatórias e foram assertivas. Por fim, o trabalho sofre de algumas limitações que não foram tratadas, sendo elas a recomendação de um desenvolvedor que ainda não possui nenhum histórico de alocação, ou projeto que não possua nenhum desenvolvedor alocado.

Já os autores Matter, Kuhn e Nierstrasz (2009) desenvolveram um algoritmo protótipo chamado de DEVSELECT, que sugere desenvolvedores para serem alocados para corrigirem relatórios de erros (Bugs) com base na experiência do desenvolvedor na área em que bug aconteceu. A recomendação é feita a partir da análise dos *commits* (alteração de código) realizados por cada desenvolvedor, que são usados para construir e manter um vocabulário que representa o desenvolvedor. A partir do vocabulário, é construída uma matriz de termo-autor, semelhante à matriz termo-documento comumente utilizada em sistemas de Recuperação de Informação. De fato, esse trabalho se assemelha aos sistemas de RI mais tradicionais. Ao receber um relatório de erro, o protótipo constrói uma representação vetorial do texto com os termos relevantes e suas frequências de ocorrência. A seguir, o sistema usa esses termos para filtrar da matriz termo-autor os desenvolvedores que tenham ao menos um termo referente ao relatório. Por fim, o protótipo ranqueia os desenvolvedores selecionados com base na similaridade entre o vetor do relatório de entrada e os vetores que representam os desenvolvedores, usando a medida de cosseno. Para avaliar o protótipo os autores realizaram um estudo de caso em projetos open source do Eclipse, de onde foram coletados os *commits* dos projetos e bugs relatados. A avaliação realizada objetivou medir a precisão e o recall do sistema. Os autores definiram a precisão como sendo a porcentagem dos desenvolvedores sugeridos que realmente trabalharam nos bugs, e o recall como sendo o número de desenvolvedores sugeridos dividido pelo total de desenvolvedores (sugeridos e não sugeridos) que trabalharam nos bugs. A precisão média foi de 33% e recall médio foi de 71%. É importante ressaltar que os autores não conseguiram definir qual foi a estratégia de alocação para os bugs. Contudo, é possível observar, pelo valor de recall, que grande parte dos desenvolvedores que trabalharam nos bugs estavam dentre os que foram recomendados (71%). Uma das limitações do projeto é que ele exige uma padronização da escrita de código e *commit* em relação ao que a área desenvolvida,

para ser possível coletar um vocabulário que possa expressar o conhecimento do desenvolvedor em uma área, além de que apenas desenvolvedores que tiverem realizado alguma submissão de código podem ser recomendados.

O trabalho de Filho et al. (2022) visa alocar desenvolvedores para tarefas em um projeto utilizando dados cadastrados do perfil dos desenvolvedores e as tarefas disponíveis. Vale ressaltar que, para o autor em seu trabalho, "desenvolvedor" pode ser cada profissional que faz parte do fluxo de implementação de um sistema e não apenas o programador. Portanto, o autor assume que o desenvolvedor pode ter diferentes perfis, como analista de sistemas, designer, programador ou testador, para indicar o perfil de desenvolvedor. Para a alocação, as tarefas são classificadas utilizando dois algoritmos: o *MultinomialNB* (classificador Bayesiano) e o *Random Forest*, nas classes de analistas de sistemas, designers, programadores e testadores. Esses algoritmos foram treinados com um conjunto de tarefas previamente classificadas. Já o processo de classificação dos perfis dos desenvolvedores foi baseado em questionários, com o objetivo de classificar o perfil de cada desenvolvedor nas mesmas classes das tarefas. Isso foi feito realizando cálculos matemáticos com uma fórmula criada especificamente para o problema, utilizando pesos definidos pelos gerentes de projetos que utilizavam a ferramenta. Após o treinamento dos classificadores das tarefas e perfis, o conjunto de dados é percorrido realizando-se associações dos desenvolvedores e tarefas, e no final é verificado quais são as combinações que resultam em maiores probabilidades de acontecer. A avaliação foi feita de duas formas, uma quantitativa e outra qualitativa. Para a avaliação qualitativa, foram coletados *feedbacks* dos desenvolvedores quanto a sua alocação, onde a maioria dos participantes afirmou serem relevantes as recomendações. Já a avaliação quantitativa mediu acurácia, precisão, recall e $F1$ do sistema, obtendo valores de 73% para a acurácia, para a precisão, a de menor valor obtido sendo desenvolvedor com 53% e analista de sistema com 83%. Já os valores de recall variaram entre 63% a 89%, e $F1$ entre 62% a 73%. A partir dos resultados, conclui-se que a ferramenta foi capaz de realizar recomendações adequadas para uma determinada tarefa. Contudo, um ponto que deve ser considerado é a necessidade de realização de um questionário extenso para conseguir mapear bem o perfil do desenvolvedor, algo que pode dificultar a coleta dos dados. Porém, esse questionário parece difícil de evitar para a montagem de um perfil acurado.

Os autores Anvik, Hiew e Murphy (2006) apresentam um algoritmo de aprendizagem de máquina para realizar a análise de novos *bugs* e de alocação de desenvolvedores para resolver esses bugs em projetos *open source*. Os autores utilizaram um algoritmo de *Support*

Vector Machines (SVM) para realizar a classificação dos bugs em categorias, essas categorias sendo os nomes dos desenvolvedores recomendados para a alocação. Para ser possível essa classificação, é necessário que o projeto já possua um repositório de bugs com alocações já feitas, para o treinamento e a criação do modelo do classificador. Após o treinamento, a pessoa responsável por realizar a alocação envia os bugs de cada novo relatório de erro cadastrado para o algoritmo, que classifica/recomenda qual desenvolvedor deve ser alocado. Os autores selecionaram um projeto *open source*, o gcc, para treinar e avaliar o algoritmo, obtendo uma precisão baixa, de 6% a 18%. Os autores ressaltam que esse problema pode ser devido à dominância de um único desenvolvedor na tarefa de resolução de bugs na lista de desenvolvedores, que corrigiu até 8 vezes mais bugs que o segundo colocado da lista. Um novo teste foi realizado removendo esse desenvolvedor dominante do treinamento, aumentando assim a taxa de precisão para 30%. A seguir, os autores utilizaram o mesmo algoritmo para treinar e avaliar 2 grandes projetos, Eclipse e Firefox, obtendo valores de precisão de 57% e 64% respectivamente. A melhora no resultado pode ser devida ao repositório de bugs ser melhor distribuído, ou os filtros aplicados na escolha dos relatórios de erros de treinamentos possibilitaram uma amostragem melhor. Como dito pelos autores, algo que pode impactar no processo é a necessidade de um bom repositório de bugs, pois é necessário conseguir uma boa amostra dos relatórios para o treinamento do algoritmo. Contudo, mesmo com problemas para categorizar relatórios do Firefox (foi possível categorizar apenas 49% dos relatórios), o algoritmo ainda conseguiu um bom resultado. Dado o filtro aplicado, outro problema é a necessidade de que desenvolvedores possuam uma contribuição ativa na resolução dos bugs. A falta dessa contribuição ocasionava uma menor precisão na recomendação.

A Tabela 1 traz uma síntese dos trabalhos descritos acima, destacando algumas características consideradas relevantes nessa área de trabalho. A partir dessa tabela, é possível também fazer uma breve comparação entre esses trabalhos e a solução implementada neste mestrado. A última linha da tabela antecipa algumas informações sobre este trabalho, que serão aprofundadas no Capítulo 4.

As características escolhidas para síntese e comparação entre os trabalhos são:

- Que entidade está sendo recomendada?
- Qual é a abordagem utilizada para a recomendação?
- Que informações são usadas para a recomendação?

- Qual linguagem, framework ou ferramenta foi utilizado na implementação?
- Como o sistema foi avaliado?
- A avaliação de desempenho foi qualitativa ou quantitativa?
- Qual o desempenho do sistema?

Este trabalho objetiva desenvolver uma solução para a alocação de desenvolvedores para projetos e para realização de tarefas. Para isto, são empregadas técnicas de processamento de texto e recomendação baseada em conteúdo. Para a realização da recomendação de alocação dos desenvolvedores a tarefas, são realizados cálculos que computam a similaridade entre as atividades já realizadas pelos desenvolvedores e as tarefas que necessitam de alocação, usando a medida de similaridade do cosseno. Esta proposta tem como base a utilização dos dados dos desenvolvedores e das tarefas realizadas por eles. Para isto, foi criado um cadastro de informações sobre a experiência do desenvolvedor, e foi implementado um módulo que captura e atualiza uma base de dados das tarefas realizadas, a fim de realizar o mapeamento do perfil do desenvolvedor e suas competências.

Como será detalhado no Capítulo 4, este trabalho adotou a filtragem baseada em conteúdo para realizar recomendação, sendo essa a abordagem predominante nos trabalhos relacionados, quando se deseja realizar recomendações mais específicas. A exceção é o trabalho Silva, Felício e Paixão (2018), que utiliza uma abordagem baseada em filtragem colaborativa.

Para mitigar problemas de *cold start* na nossa solução proposta, são coletadas informações mais completas sobre cada desenvolvedor para formar seu perfil: além de ser coletado o dado histórico de tarefas realizadas pelo desenvolvedor nos projetos, são utilizadas também as experiências cadastradas no seu perfil. Essa é uma pequena vantagem para a realização de recomendações, pois os dados coletados não são oriundos de apenas uma fonte de informação, como é o caso dos outros trabalhos. Porém, por falta de tempo, não implementamos a opção de *feedback* do usuário, como vemos no trabalho de Filho et al. (2022), sendo esse um recurso muito importante para a melhora contínua da recomendação.

Como dito anteriormente, o objetivo deste trabalho é recomendar desenvolvedores para projetos e tarefas, e para esta atividade utilizamos dados históricos de atividades realizadas pelo desenvolvedor em um projeto, visando sempre indicar o melhor especialista para a atividade. Já o trabalho de Silva, Felício e Paixão (2018) visa recomendar desenvolvedores para projetos e não para tarefas, por isso esse trabalho utiliza o histórico de projetos nos quais o desenvolvedor

Tabela 1 – Tabela de comparação dos trabalhos relacionados

Trabalho	Que entidade está sendo recomendada?	Qual é a abordagem para a recomendação?	Que informações são usadas para a recomendação	Qual linguagem, framework ou ferramenta foi utilizado na implementação?	Como o sistema foi avaliado?	A avaliação de desempenho foi qualitativa ou quantitativa?	Qual o desempenho do sistema?
Silva, Felício e Paixão (2018)	Alocação de desenvolvedores para projetos	3 Algoritmos de fatoração de matrizes e filtragem colaborativa	Histórico de projetos	Java com biblioteca LibRec	Validação cruzada	Quantitativo - Área Abaixo da Curva ROC	AUC de 0,76%
Matter, Kuhn e Nierstrasz (2009)	Alocação de desenvolvedores para correção de relatórios de erros	Baseado em filtragem de conteúdo	Histórico de commits	Não informado	Estudo de caso	Quantitativo - Precisão e recall	Precisão 33% e Recall 71%
Filho et al. (2022)	Alocação de desenvolvedores para tarefas	Baseado na filtragem de conteúdo, com processamento textual	Cadastro de perfil dos desenvolvedores	Python com bibliotecas Sklearn, textblob e googletrans	Estudo de caso	Quantitativo e qualitativo - Acurácia, precisão, recall e F1	Positivo - acurácia de 73% precisão 53% a 83% recall 63% a 89% F1 62% a 73%
Anvik, Hiew e Murphy (2006)	Alocação de desenvolvedores para correção de relatórios de erros	Baseado na filtragem de conteúdo e Classificação	Histórico de bugs corrigidos	Não informado	Estudo de caso	Quantitativo - Precisão e recall	Precisão 64% a 57% recall de 2% a 7%
Este trabalho - Protótipo desenvolvido	Alocação de desenvolvedores para projetos e tarefas (Implementação de novas funcionalidades ou melhoria e correção de relatórios de erros)	Baseado na filtragem de conteúdo, recomendação de documentos	Histórico de tarefa, perfil cadastrado dos desenvolvedores	Python com bibliotecas django, jira, nltk, scikit-learn	Estudo de caso	Quantitativo - Precisão	Precisão 65.13%

Fonte: Elaborada pelo autor (2022)

esteve alocado. No trabalho de Filho et al. (2022), o objetivo é alocação para tarefas, porém a sua fonte de dados difere da nossa, utilizando o perfil do desenvolvedor coletado e deduzido através de um questionário.

Vale ressaltar que, embora este trabalho compartilhe semelhanças com estudos no campo da formação de grupos, seu objetivo principal difere. Patil (2020) relata em seu trabalho a criação de um framework integrativo usando um sistema de recomendação para a formação de equipes. Segundo esse autor, o processo tradicional de formação de grupos busca criar equipes eficazes, nas quais as habilidades individuais se complementam, enquanto nosso foco reside na recomendação do especialista mais adequado para uma tarefa específica. Isso não implica na exclusão de especialistas anteriormente recomendados para outras tarefas, uma responsabilidade que recai sobre o gerente.

3.3 CONSIDERAÇÕES FINAIS

Neste capítulo, dedicamo-nos a explorar o funcionamento e a construção de sistemas de recomendação de especialistas. Discutimos o processo de coleta e armazenamento de dados, estratégias de recomendação, e apresentamos técnicas fundamentais de recomendação de especialistas, considerando suas vantagens e desvantagens. Além disso, examinamos trabalhos relevantes que desenvolveram sistemas de recomendação de especialistas, permitindo uma comparação com o sistema desenvolvido neste trabalho.

O próximo capítulo oferecerá uma descrição minuciosa da solução de recomendação concebida e implementada nesta pesquisa.

4 PROCESSO DE CRIAÇÃO DE UM SR DE ESPECIALISTAS DE ENGENHARIA DE SOFTWARE

Este capítulo apresenta o processo proposto para criação de sistemas de recomendação para auxiliar na alocação de desenvolvedores de SW a projeto e tarefas. O objetivo principal é melhorar a alocação de desenvolvedores para a realização das atividades de desenvolvimento, buscando sempre o profissional mais adequado para a execução de cada tarefa específica.

Seguimos aqui as diretrizes de construção de SR de especialistas apresentadas no Capítulo 3, contando com três etapas (Seção 4.1): Definição da técnica de recomendação a ser utilizada; Criação da base de dados do SR; e Implementação do sistema de recomendação de especialistas. Adotamos aqui a abordagem de recomendação baseada na similaridade entre trabalhos realizados e a nova tarefa a executar, que será detalhada na Seção 4.1.1.

Já o protótipo implementado conta com três módulos distintos (Seção 4.1.3): Coleta de Informações (para criação da base de dados), Recomendação e Gerenciamento de Alocação. Esse protótipo foi utilizado em um estudo de caso realizado em parceria com uma empresa de desenvolvimento de SW chamada Mesa Mobile Thinking (Seção 4.3).

Este capítulo está dividido em quatro seções. A Seção 4.1 apresenta o processo proposto para SR, juntamente com uma descrição detalhada dos módulos definidos e desenvolvidos. A Seção 4.2 detalha o protótipo em si, incluindo a interface do sistema e seu funcionamento. Na Seção 4.3, descrevemos o estudo de caso conduzido para validar o protótipo e discutimos os resultados obtidos. Por fim, a Seção 4.4 traz as considerações finais deste capítulo.

4.1 SOLUÇÃO PROPOSTA

Como visto no Capítulo 3, o processo de construção de um SR de especialistas conta com três etapas: (1) Definição da estratégia e técnica de recomendação a serem utilizadas (Seção 4.1.1); (2) Criação da base de dados do SR (Seção 4.1.2); e (3) Implementação do sistema de recomendação de especialistas (Seção 4.1.3).

Antes de iniciar a descrição dessas etapas, é importante relacioná-las aos módulos do sistema protótipo implementado. O protótipo conta com três módulos distintos: (A) Módulo de coleta de informações (Seção 4.1.3.1), que realiza a criação da base de dados a ser usada pelo sistema; (B) Módulo de Recomendação (Seção 4.1.3.2), que analisa as informações coletadas e recomenda os desenvolvedores mais adequados para cada tarefa; e (C) Módulo de

Gerenciamento de Alocação (Seção 4.1.3.3), responsável por registrar e manter o controle das alocações de desenvolvedores em projetos e tarefas.

Como podemos ver, todos os módulos do protótipo se relacionam com as etapas (2) e (3) do processo de construção do SR de especialistas, sendo que a etapa (1) corresponde a decisões de projeto que são anteriores à implementação do sistema.

4.1.1 Estratégia e técnica de recomendação adotadas

Nosso trabalho adota uma estratégia personalizada de recomendação (ver Capítulo 2, Seção 2.1). Contudo, nosso contexto tem particularidades importantes a considerar. Aqui, o *usuário* do sistema é o gerente responsável pela alocação das tarefas e os *desenvolvedores* são vistos como os "*itens*" que serão recomendados.

Assim, o perfil do usuário será representado aqui pela descrição da tarefa a ser realizada (tarefa para a qual o usuário está buscando profissionais especialistas). Como as tarefas mudam a cada rodada do sistema, trata-se de um perfil "efêmero", que não será persistido para uso na próxima consulta ao sistema, mas será vinculado ao desenvolvedor que a realizará como experiência.

Por sua vez, os desenvolvedores ("*itens*") são representados pela descrição dos trabalhos/atividades que cada um já realizou (dados históricos). Considerando agora os conceitos apresentados no Capítulo 3 (Seção 3.1.1), vemos que nosso trabalho adota *o modelo orientado a tópico* descrito por Nikzad–Khasmakhi, Balafar e Reza Feizi–Derakhshi (2019), que se baseia nos trabalhos produzidos pelo especialista.

Por fim, adotamos a técnica de recomendação baseada em *filtragem de conteúdo*, que analisa a similaridade entre tarefas a realizar ("*perfil*" do usuário) e os dados históricos dos desenvolvedores. Essa configuração de estratégias e técnicas apresentou os melhores resultados nos testes exploratórios realizados com estratégias e abordagens variadas. A Seção 4.1.3.2.2 irá detalhar como essa configuração foi implementada no protótipo desenvolvido.

4.1.2 Criação da base de dados do SR

A criação da base de dados é realizado pelo módulo de Coleta de Informações. Este módulo é responsável por conectar-se a bases de dados externas e recuperar as informações necessárias

para o funcionamento do sistema. Ele pode utilizar *crawlers*, acessar APIs[1] ou qualquer método necessário para obter os dados. Além disso, ele realiza o pré-processamento dos dados para torná-los utilizáveis pelo sistema implementado. O módulo também gerencia o banco de dados do SR, executando consultas e registrando as informações necessárias solicitadas pelos outros módulos.

Para criar a base de dados utilizada pelo SR, foram coletados dados de duas fontes diferentes. A primeira fonte é uma coleta direta realizada durante o cadastro do projeto e do desenvolvedor. Essas informações são coletadas pelo módulo de gerenciamento, conforme explicado na Seção 4.1.3.3.

A segunda fonte de dados é o sistema de gerenciamento de tarefas da empresa, conhecido como *Jira*. O módulo de coleta de informações (conforme descrito na Seção 4.1.3.1) consulta o *Jira* por meio de uma API. As informações do *Jira* são armazenadas de forma estruturada em um banco de dados online, facilitando o trabalho com os dados. Por ser amplamente utilizado pela gerência da empresa, o *Jira* concentra todo o histórico de informações necessário para realizar as recomendações. Além disso, todas as informações passam por uma revisão tanto pelo cliente quanto pelos gerentes, o que garante a padronização e formalidade dos dados.

As informações provenientes das fontes mencionadas anteriormente compõem a nossa base de dados, que inclui os projetos cadastrados, o histórico de tarefas já realizadas e os desenvolvedores que as executaram. O processo de pré-processamento das informações é detalhado na Seção 4.1.3.1.

4.1.3 Implementação do SR de especialistas

A empresa parceira possui um sistema de gerenciamento de tarefas que registra tanto as atividades a serem realizadas quanto aquelas que já foram concluídas, mantendo assim um histórico completo das atividades. Como o protótipo desenvolvido neste trabalho é independente desse sistema de gerenciamento de tarefa, foi necessário dividi-lo em três módulos distintos para melhor atender, apresentado na Figura 3:

- **Módulo de Coleta de Informações:** Este módulo é responsável por coletar os dados relevantes do sistema de gerenciamento de tarefas, como informações sobre as tarefas concluídas, os desenvolvedores envolvidos e as competências técnicas exigidas. Ele extrai

[1] O que é uma API? - https://aws.amazon.com/pt/what-is/api/

essas informações do sistema de gerenciamento existente e as prepara para análise no módulo seguinte.

- **Módulo de Recomendação:** O segundo módulo é responsável por analisar as informações coletadas e recomendar os desenvolvedores mais adequados para cada tarefa. Utilizando técnicas de recomendação, como a análise de similaridade entre tarefas e perfis de desenvolvedores, esse módulo gera sugestões baseadas em experiências passadas e na expertise dos profissionais.
- **Módulo de Gerenciamento de Alocação:** O terceiro módulo é responsável por registrar e manter o controle das alocações de desenvolvedores em projetos e tarefas. Além disso, o módulo gerencia o cadastro dos desenvolvedores, coletando informações sobre suas experiências e um registro completo das tarefas realizadas por cada desenvolvedor. Essas informações são utilizadas para selecionar os melhores desenvolvedores de acordo com as características do projeto.

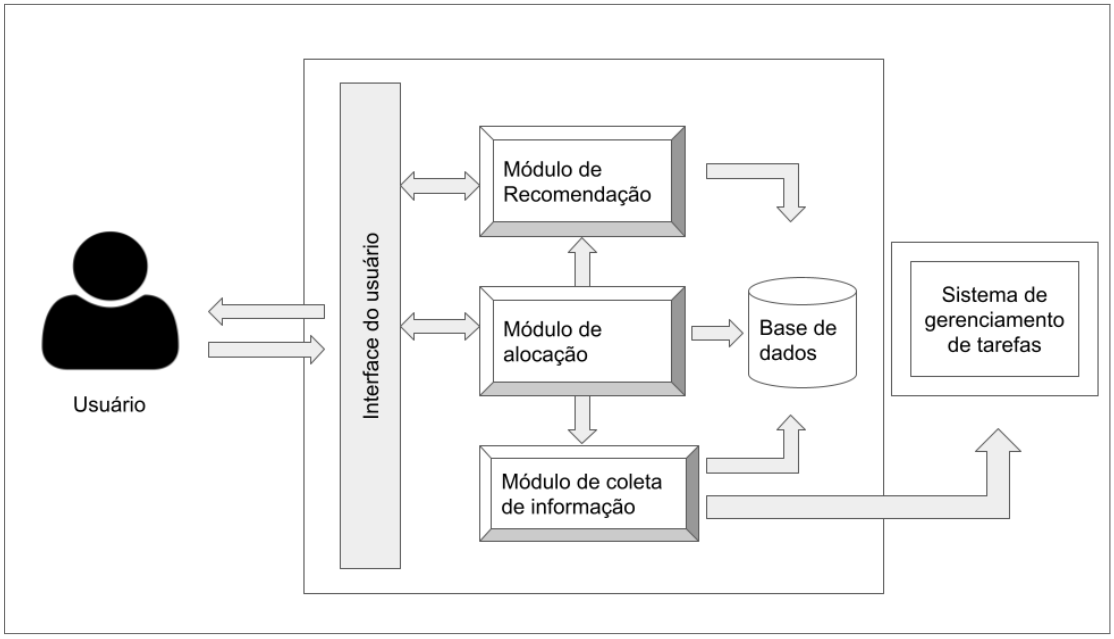
A divisão do sistema em módulos distintos permite uma abordagem modular e flexível, garantindo que cada etapa do processo de alocação seja tratada de maneira eficiente e eficaz. Essa estrutura modular também facilita futuras melhorias e adaptações do protótipo, permitindo a integração com diferentes sistemas de gerenciamento de tarefas e a incorporação de novas técnicas de recomendação, à medida que a área evolui.

4.1.3.1 Módulo de coleta de informação

Para viabilizar o mapeamento do conhecimento dos desenvolvedores, essencial para a posterior recomendação, foi implementado um módulo de coleta de informações que se conecta ao sistema de gerenciamento de tarefas da empresa. Esse módulo tem a responsabilidade de se comunicar com o sistema de gestão de tarefas utilizado pela empresa, capturando as tarefas realizadas pelos desenvolvedores, como implementação de novas funcionalidades, melhorias e correções de bugs. Além disso, ele coleta as tarefas a serem realizadas no *sprint* atual, juntamente com as informações necessárias, e as armazena em um banco de dados, que será acessado pelos demais módulos.

Antes de realizar o cadastro das informações coletadas, é necessário realizar um pré-processamento desses dados. Foram realizados testes para determinar as configurações de

Figura 3 – Arquitetura proposta de sistema de recomendação de especialistas



Fonte:Elaborado pelo autor(2023)

pré-processamento mais eficazes para a nossa tarefa de recomendação. Ao final desses testes, definiu-se a configuração que obteve os melhores resultados para o pré-processamento. Etapas de pré-processamento selecionadas são listadas a seguir:

- Remoção de links: Nas tarefas coletadas, é comum encontrar links para sites externos que auxiliam os desenvolvedores, como sites de *wireframes* ou documentos externos. Esses links são removidos, pois, além de poluírem os dados e prejudicarem a recomendação, a ferramenta atualmente não navega nesses sites para obter informações adicionais, uma vez que as informações contidas nas tarefas se mostraram suficientes para uma boa recomendação.

- Remoção de edição de texto: O sistema de gerenciamento de tarefas possui um editor de texto que permite edições de cor, fonte, entre outros, que não agregam valor à tarefa. Portanto, essas edições são eliminadas, uma vez que não contribuem para a recomendação.

- Remoção de caracteres especiais: Também são removidos caracteres especiais, como %, $, #, @ e !, que não possuem valor no texto ou no contexto da empresa. Embora a presença desses caracteres não tenha impacto nos resultados, eles são removidos para melhorar o desempenho do protótipo, evitando a validação de caracteres sem valor.

- Tokenização: após a remoção de links e caracteres especiais, além da limpeza das edições do texto da tarefa, o texto restante passa por um processo de tokenização que o divide em segmentos menores. Foram realizados testes com diferentes tipos de tokenização, como por parágrafos, por frases e por conjuntos de duas e três palavras. No entanto, a tokenização que obteve melhor desempenho nos testes foi por palavras isoladas (cada palavra se tornando um token individual).

- Remoção de *stopwords*: Como já mencionado na Seção 3.1.2 do capítulo anterior, *stopwords* são palavras que geralmente não agregam valor semântico ao texto (e.g., artigos, preposições etc.), e assim sua relevância varia de acordo com o contexto do sistema. As stopwords são removidas para diminuir o tamanho dos textos, facilitando assim seu armazenamento e seu uso. Para este trabalho, foi utilizado um conjunto de palavras disponível na biblioteca do NLTK, juntamente com um conjunto adicional construído manualmente. No total, foram consideradas 543 palavras como *stopwords*.

- Stemização: Como também foi visto na Seção 3.1.2, o processo de stemming (ou stemização) objetiva reduzir uma palavra ao seu radical, agrupando assim palavras que têm a mesma raiz semântica. Em sistemas de Recuperação de Informação, essa operação aumenta a cobertura das consultas à base de documentos, uma vez que o sistema recupera todos os documentos que contenham palavras com a mesma raiz de alguma palavra da consulta. Durante os testes realizado com e sem stemização, foi notório o impacto positivo que esse processo trouxe. Antes da stemização, eram retornadas cerca de 4 a 6 tarefas para cada consulta, e após a aplicação desse processo, o sistema passou a retornar de 9 a 16 novas tarefas que não eram retornadas antes por causa de flexão das palavras. Isso melhorou muito a relevância da resposta do sistema, possibilitando recomendações mais precisas. Ainda observamos outra consequência positiva, que foi a identificação de tarefas duplicadas de melhoria e correção de bugs durante a realização do experimento.

Esta configuração final de etapas de pré-processamento foi aplicada a todas as tarefas recuperadas e a todos os campos coletados da tarefa, com exceção do campo *feature*, que não pode ser modificado em nada, pois define diretamente uma funcionalidade conhecida nos relatórios de erros.

Para cada tarefa, foram coletados os valores do título, descrição da *feature*, descrição da melhoria, descrição e passos do relatório de erros, descrição da categoria, plataforma e responsável pela tarefa. Esses atributos foram utilizados para definir a tarefa, enquanto os demais atributos não foram considerados. As tarefas coletadas e pré-processadas foram cadastradas no banco de dados e disponibilizadas para os demais módulos do protótipo. Isso evitou um alto número de requisições ao sistema de gerenciamento de tarefas.

4.1.3.2 Módulo de recomendação

O módulo de recomendação desempenha a função de analisar um projeto ou uma tarefa de entrada e sugerir o desenvolvedor mais adequado com base em suas experiências. Esse módulo é dividido em dois submódulos: (1) recomendação de desenvolvedor para um projeto e (2) recomendação de desenvolvedor para uma tarefa específica.

4.1.3.2.1 Recomendação para Projeto

A empresa parceira possui diversos projetos em andamento e novos projetos são firmados regularmente, o que resulta na contratação e alocação frequente de novos desenvolvedores. Embora a alocação de desenvolvedores contratados a tarefas seja mais frequente, também se faz necessário contratar novos desenvolvedores para suprir novas demandas.

Para essa atividade, foi adotada uma abordagem de recomendação baseada em filtragem de conteúdo, utilizando dados do desenvolvedor capturados durante seu cadastramento no módulo de alocação. Caso o desenvolvedor já seja da empresa, usam-se também os dados de atividades já realizadas em outros projetos. Esses dados são comparados com os dados do campo características do projeto cadastrado no módulo de alocação.

A ideia inicial utilizada nesse submódulo foi de clusterizar automaticamente as atividades já realizadas pelo desenvolvedor e fazer a recomendação para projeto a partir do cluster resultante. Contudo, os resultados das recomendações baseadas em clusters não foram satisfatórios, igualando-se aos resultados das recomendações que utilizavam apenas o campo de *feature* das atividades já realizadas. Verificou-se também, ao imprimir esses clusters, que muitas dos grupos identificados já estavam mapeados e alguns grupos não faziam sentido, como a classe de "adicionar produto" e "remover produto" que, na verdade, era apenas uma "subclasse" de "Produtos". Portanto, como não foi possível definir um número k de grupos

que fizesse sentido para todos os projetos que fossem cadastrados, decidiu-se utilizar apenas o campo *feature* e as experiências cadastradas diretamente pelo gerente.

Por fim, ficou definida aplicação de alguns filtros para a recomendação, sendo eles a plataforma na qual o desenvolvedor atua e a sua alocação atual em outro projeto, evitando que um desenvolvedor já alocado seja recomendado.

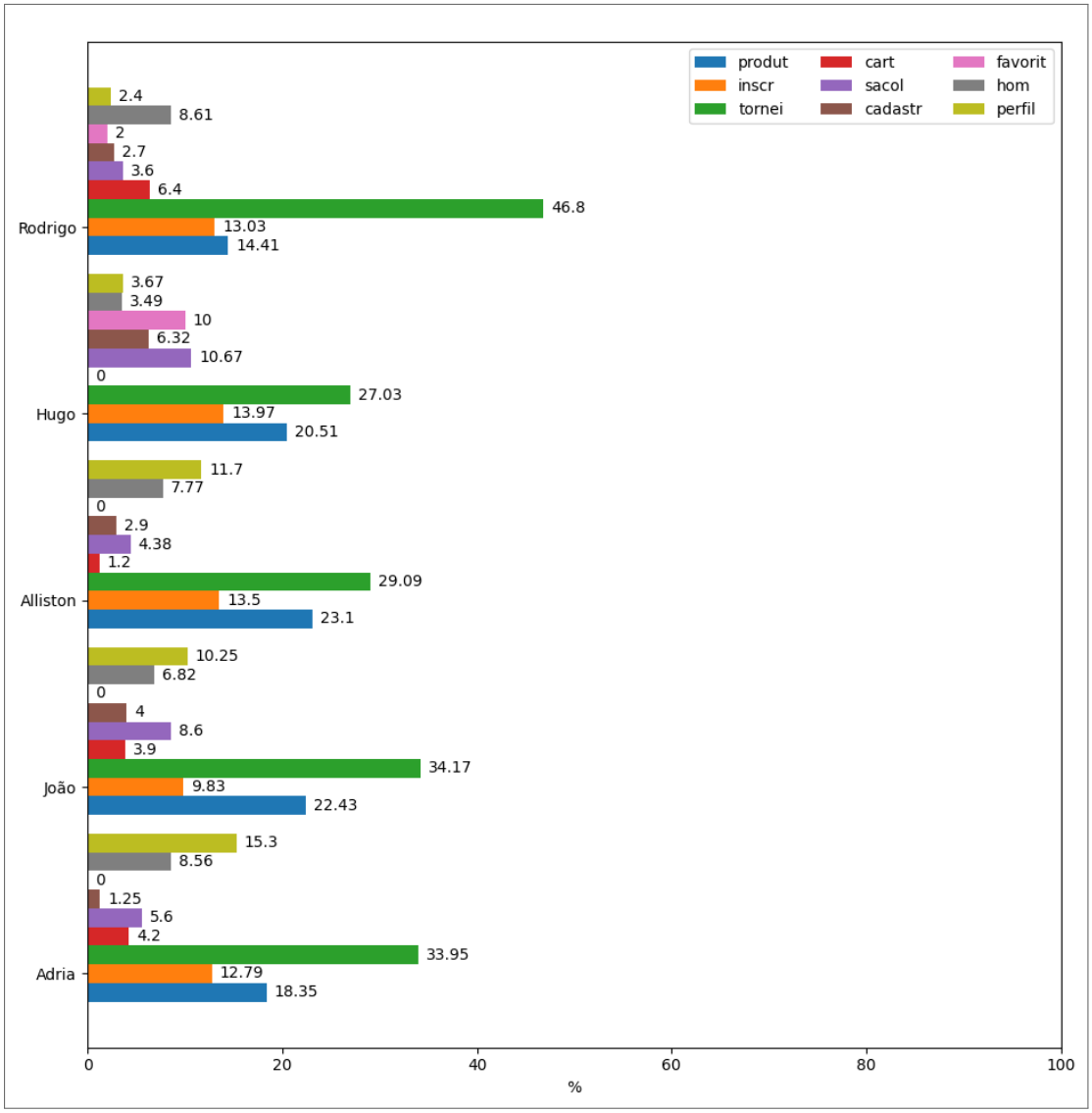
4.1.3.2.2 Recomendação para tarefa

O processo de recomendação de tarefas começou com uma tentativa de calcular a experiência dos desenvolvedores em uma determinada área. Para isso, foram capturadas todas as atividades realizadas por cada desenvolvedor durante um período de um ano. Inicialmente, foi calculado o percentual de atividades relacionadas para cada *feature* em relação ao total de atividades desenvolvidas pelo desenvolvedor. Essa abordagem visava avaliar a distribuição das áreas de atuação de cada desenvolvedor nas tarefas, levando em consideração o tempo de atuação no projeto, para evitar que o resultado fosse prejudicado por desenvolvedores mais recentes com menor quantidade de atividades.

No projeto utilizado para validação do protótipo, os desenvolvedores estão organizados em *squads*, onde cada *squad* é responsável por um conjunto específico de funcionalidades (*features*). A Figura 5 apresenta como é a distribuição das *features* por tarefas em uma das *squads* do projeto. Após análise conjunta com a Figura 4, observou-se que a distribuição de conhecimento e tarefas por *features* era equilibrada entre os desenvolvedores, conforme mostra a Figura 4, com exceção de algumas *features* que não foram modificadas por um longo período devido à sua estabilidade, já as tarefas que eram constantemente modificadas ao longo do período, tinham uma distribuição notavelmente uniforme entre os desenvolvedores. Essa distribuição uniforme do conhecimento sobre as *features* entre os desenvolvedores pode ser atribuída a uma filosofia de compartilhamento de conhecimento adotada pela empresa, na qual busca-se manter os desenvolvedores com um conhecimento mínimo de todas as *features*, de modo que, na ausência de um desenvolvedor, sempre haja outro capaz de assumir a tarefa.

Considerando a distribuição de conhecimento sobre as *features* nas *squads*, uma abordagem mais precisa foi adotada para a recomendação, em vez de calcular apenas a pontuação de um desenvolvedor em uma determinada *feature*/área. Essa abordagem consiste em recomendar um desenvolvedor que tenha realizado uma tarefa mais semelhante à tarefa que precisa de uma recomendação de alocação. Dessa forma, a abordagem inicial de calcular uma pontuação

Figura 4 – Relação entre *feature* e desenvolvedor



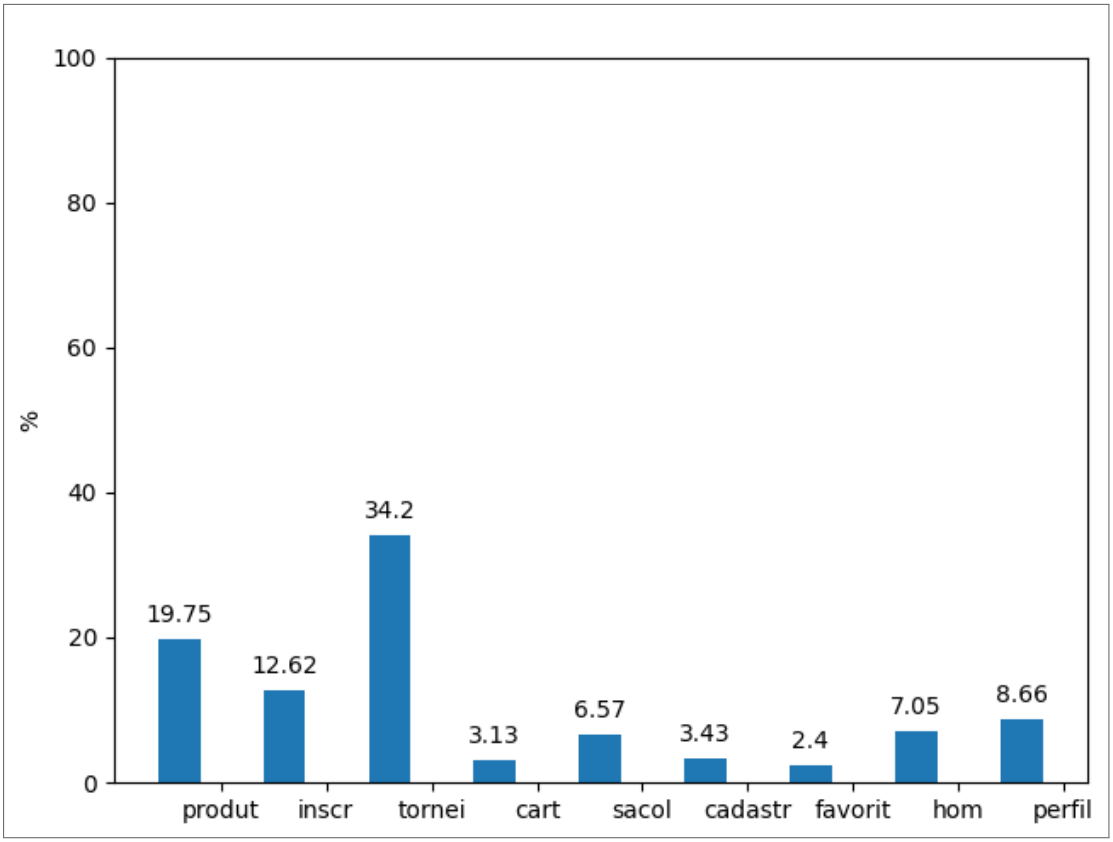
Fonte:Elaborado pelo autor(2023)

para uma determinada *feature* foi substituída por uma filtragem baseada em conteúdo que busca identificar a tarefa já realizada pelos desenvolvedores mais similar à tarefa atual.

No processo de recomendação, os textos que representam as tarefas coletadas de cada desenvolvedor passaram por diversas etapas de pré-processamento, além das já mencionadas na Seção anterior. As novas etapas realizadas são:

- **Cálculo do TF-IDF:** *Term Frequency-Inverse Document Frequency (TF-IDF)* é um cálculo estatístico que busca quantificar a importância (ou relevância) de um termo com relação ao conjunto de termos em um corpus de documentos. Em resumo, o algoritmo inicialmente quantifica a frequência de um termo (TF) em um documento - quanto maior sua frequência, maior é a relevância do termo para representar o documento.

Figura 5 – Relação entre *feature* e tarefas da *squad*



Fonte:Elaborado pelo autor(2023)

Em seguida, é calculada a frequência do termo nos documentos do corpus (IDF). Em seguida, o valor do TF é multiplicado pelo inverso do valor do IDF, obtendo-se assim o "peso" do termo para representar o documento. Caso o termo seja muito frequente no corpus (i.e., apareça em muitos documentos), o seu "peso" final será reduzido em relação ao seu TF; caso contrário, se o termo for raro no corpus, o seu "peso" final pode ser aumentado. A representação interna de cada texto/tarefa passa a ser agora um array de termos com pesos associados.

- **Criação do espaço vetorial de termos x documentos:** A seguir, é criado um espaço vetorial multidimensional no qual cada eixo representa um termo que ocorre no corpus, e cada tarefa é representada nesse espaço vetorial por um vetor cuja direção depende do peso de cada termo que a representa. Os pesos já foram calculados na etapa anterior, e estão contidos no array criado na etapa anterior. A nova tarefa submetida ao sistema para recomendação de desenvolvedores também será representada nesse espaço vetorial.

- **Cálculo da similaridade usando cosseno:** Por fim, os vetores criados na etapa anterior

são comparados com o vetor que representa a nova tarefa a ser realizada. A medida do cosseno do ângulo entre vetores é usada para calcular a distância entre dois vetores em um espaço vetorial. O valor do cosseno indica a similaridade entre vetores - quanto menor é o ângulo, maior é o valor do cosseno (e portanto, mais similares são os dois vetores). Quando dois vetores coincidem em direção, o valor do cosseno atinge seu máximo (igual a 1). Esse cálculo é realizado para todas as atividades já realizadas em comparação com cada nova tarefa a ser realizada.

Ao final do processo, é gerada uma matriz contendo as distâncias calculadas entre todas as tarefas em relação à tarefa em foco. Em seguida, as dez tarefas mais similares à nova tarefa serão selecionadas pelo sistema em um ranking ordenado pelo cosseno do ângulo entre tarefas anteriores e a nova tarefa. A partir desses resultados, os três desenvolvedores responsáveis pelas tarefas de maior similaridade em relação à tarefa foco são ranqueados, sendo então recomendados em conjunto para a tarefa em questão.

Vale ressaltar que um mesmo desenvolvedor pode ter mais de um trabalho com alta similaridade, como em um cenário em que o resultado seja *[dev10, dev5, dev10, dev6, dev1, dev1, dev1, dev8, dev7, dev3]*. Nesses casos, apenas a primeira ocorrência (a de valor mais alto) do desenvolvedor é considerada. Apesar de atualmente selecionarmos apenas uma ocorrência de cada desenvolvedor, essas informações abrem espaço para estudos futuros sobre a possibilidade de conceder algum tipo de bonificação para ocorrências repetidas.

Após análise das recomendações resultantes dentro dessa nova abordagem, bem como a investigação das razões por trás dessas recomendações (tarefas já realizadas), é evidente que, devido à distribuição do conhecimento conforme observado anteriormente, todos os desenvolvedores acabavam possuindo alguma tarefa com alta similaridade àquela em foco. No entanto, aqueles que estavam entre os 10 primeiros possuíam atividades extremamente semelhantes à tarefa desejada. Assim, avaliamos que essa nova abordagem foi mais bem sucedida que a inicial, trazendo melhores resultados para os usuários do sistema.

4.1.3.3 Módulo de gerenciamento de alocação

O módulo de Gerenciamento de Alocação desempenha um papel fundamental interagindo com os demais módulos, registrando as alocações tanto a projetos quanto a tarefas. Esse módulo é responsável por armazenar as informações do projeto, incluindo suas características espe-

cíficas, que são posteriormente repassadas ao módulo de Recomendação. Essas características do projeto são consideradas pelo módulo de Recomendação ao selecionar os desenvolvedores mais adequados com base na experiência de cada um.

No módulo de Gerenciamento de Alocação, são realizadas as tarefas de gerenciamento e manutenção dos registros das *squads* ativas no momento. Além disso, nesse módulo são realizadas as solicitações de recomendação de desenvolvedores para as tarefas ativas no *sprint* atual.

Outra funcionalidade importante desse módulo é o cadastro dos desenvolvedores. Durante o processo de cadastro, além de coletar as informações solicitadas, como experiências e habilidades, o módulo consulta o módulo de Coleta de Informações para obter um registro completo das tarefas já realizadas pelo desenvolvedor e associá-las ao seu perfil. Essas informações são posteriormente utilizadas pelo módulo de Recomendação ao realizar consultas de recomendação de desenvolvedores.

Dessa forma, o módulo de Gerenciamento de Alocação desempenha um papel crucial no registro e na manutenção das alocações de desenvolvedores nos projetos e tarefas. Ele permite a vinculação das características do projeto aos desenvolvedores mais adequados, com base em suas experiências anteriores. Além disso, o módulo realiza o cadastro e o registro das tarefas realizadas por cada desenvolvedor, fornecendo informações valiosas para o processo de recomendação.

4.2 DETALHES DE IMPLEMENTAÇÃO E USO DO PROTÓTIPO

Como já apresentado na Seção 4.2.3, o protótipo do sistema de recomendação desenvolvido conta com três módulos principais: (1) Coleta de Informações, (2) Recomendação e (3) Gerenciamento de Alocação. Como visto na Figura x, o módulo (1) coleta as informações necessárias a partir da base de dados do sistema, não existindo interface de comunicação entre esse módulo e o usuário gerente. Já o módulo (2) se comunica com o gerente através da interface, recebendo de entrada a nova tarefa ou o projeto que precisa de alocação de desenvolvedores. Por fim, o módulo (3) realiza a recomendação de desenvolvedores e devolve as sugestões através da interface do usuário.

O protótipo foi desenvolvido em Python utilizando o *framework open source* Django[2], que disponibiliza um conjunto de ferramentas de alto nível para a criação de aplicações WEB

[2] https://www.djangoproject.com/

utilizando Python. Esse *framework* foi selecionado para agilizar o desenvolvimento de um protótipo que pudesse ser utilizado para iniciar as validações e diminuir o tempo gasto com o desenvolvimento da página WEB, que não é o foco do trabalho. Utilizamos a biblioteca Jira[3] para fazer a conexão com a ferramenta de gerenciamento de tarefas da empresa. Ela é responsável por criar a conexão que permite ler e escrever dados nas tarefas.

O pré-processamento das tarefas foi feito com auxílio da biblioteca NLTK[4] (Natural Language Toolkit), que oferece várias funcionalidades para o tratamento de texto. Para a realização da recomendação, foram utilizadas as bibliotecas *open source* Pandas[5] e Scikit-learn[6]. A biblioteca Pandas foi utilizada para a geração das matrizes de tarefas por desenvolvedores, usadas como insumo para o Scikit-learn e para análise dos resultados durante experimentos. Já a biblioteca Scikit-learn, que está em uso no mundo acadêmico há muito tempo, oferece as funcionalidades necessárias para a realização da recomendação, diminuindo o tempo de implementação desse algoritmo.

Através da interface do usuário, o gerente pode realizar a gestão dos desenvolvedores (Figuras 6 e 7) e o gerenciamento dos projetos (Figuras 9 e 10).

A gestão dos desenvolvedores é realizada inicialmente pela criação do cadastro do desenvolvedor, que deverá conter informações como tecnologias conhecidas e suas experiências. Esses dados são coletados através da interface de registro do desenvolvedor (ver Figura 6). Vale ressaltar que esse cadastro só pode ser realizado depois que o desenvolvedor já possui um cadastro no Jira, pois o ID no Jira é um campo obrigatório. É a partir desse ID que são coletados todos os dados do desenvolvedor referentes ao projeto ao qual ele está alocado.

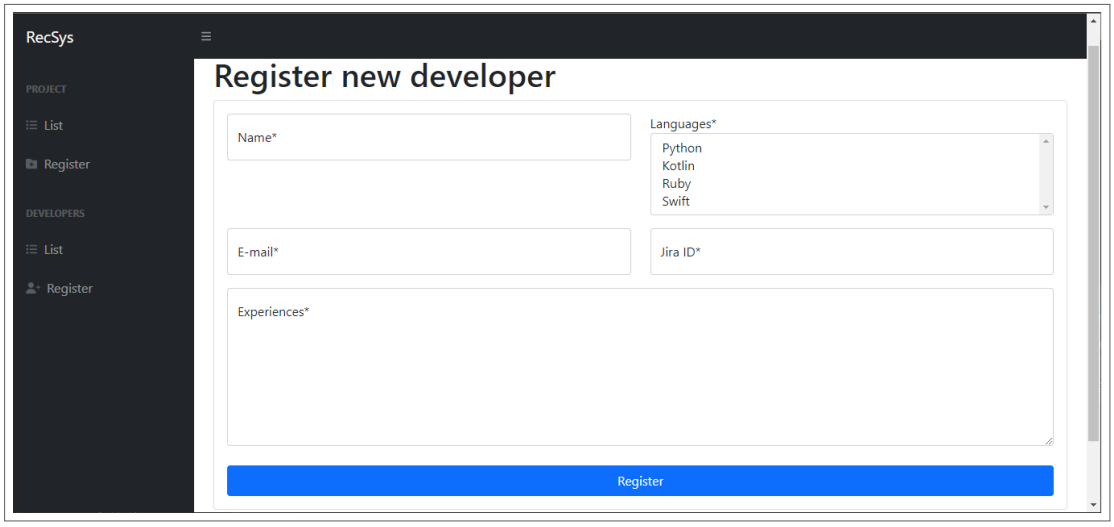
A interface mostrada na figura 7 apresenta ao gerente a lista de todos os desenvolvedores cadastrados, bem como a que projetos eles estão alocados. A partir dessa interface, o gerente pode selecionar um desenvolvedor específico e editar (dar manutenção/atualizar) seus dados através da interface de edição (Figura 8).

No protótipo atual, a interface de edição de dados do desenvolvedor está igual à interface de cadastramento, pois, por falta de tempo, não foi possível modificar seu *layout*. Contudo, por se tratar de um projeto do Django, é possível editar os dados cadastrados através da função de "gestão de dados" do painel "admin" oferecido pelo *framework*.

A tarefa de *gerenciamento de projeto* utiliza todos os módulos do sistema de forma in-

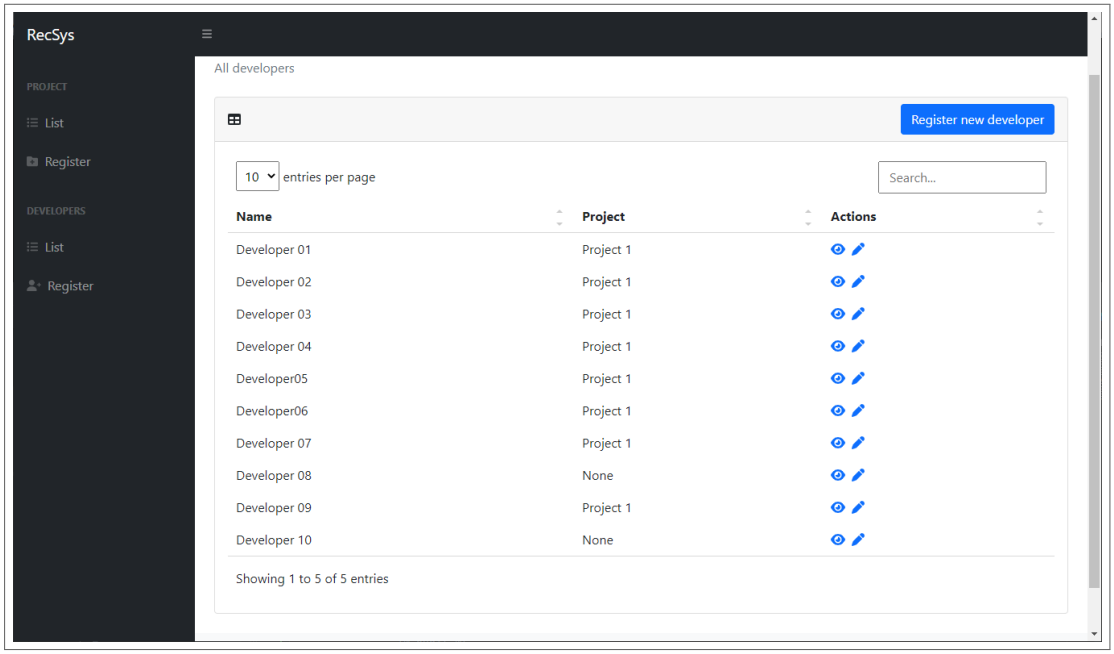
[3] https://pypi.org/project/jira/
[4] https://www.nltk.org/
[5] https://pandas.pydata.org/
[6] https://scikit-learn.org/

Figura 6 – Interface de registro de desenvolvedor



Fonte:Elaborado pelo autor(2023)

Figura 7 – Interface de listagem dos desenvolvedores registrados

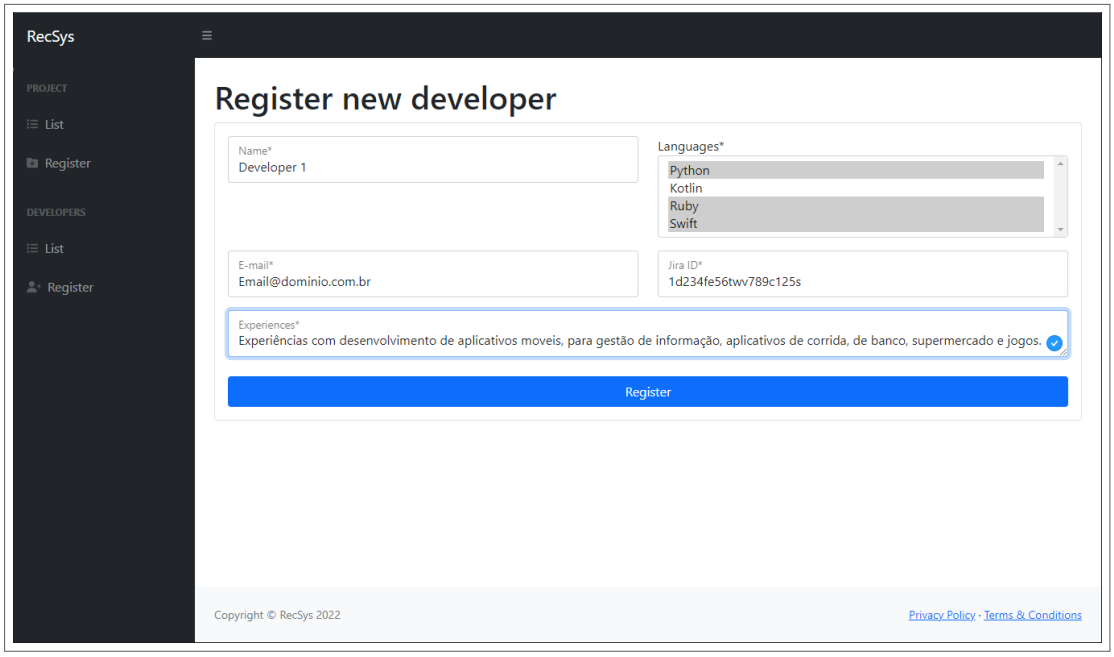


Fonte:Elaborado pelo autor(2023)

tegrada. Inicialmente é feito o cadastro básico de informações sobre o projeto, como nome do projeto e sua URL no Jira. Esse cadastramento é feito através da interface mostrada na figura 9. Esses dados são utilizados para guiar o módulo de coleta de informação a obter as informações no Jira sobre as tarefas. Outras informações, são as linguagens de programação e características do projeto, que serão utilizadas como requisitos para realização de recomendação de desenvolvedores a serem alocados no projeto.

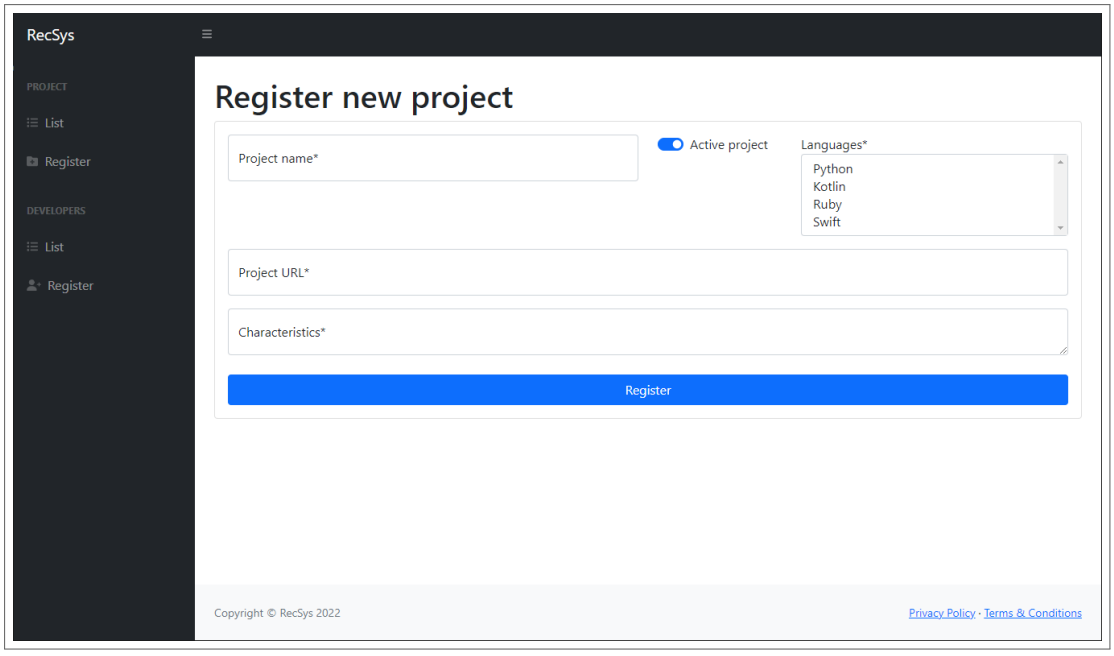
Após concluído o cadastramento inicial, os gerentes conseguem editar os dados do projeto

Figura 8 – Interface de edição dos dados do desenvolvedor



Fonte:Elaborado pelo autor(2023)

Figura 9 – Interface de registro do projeto

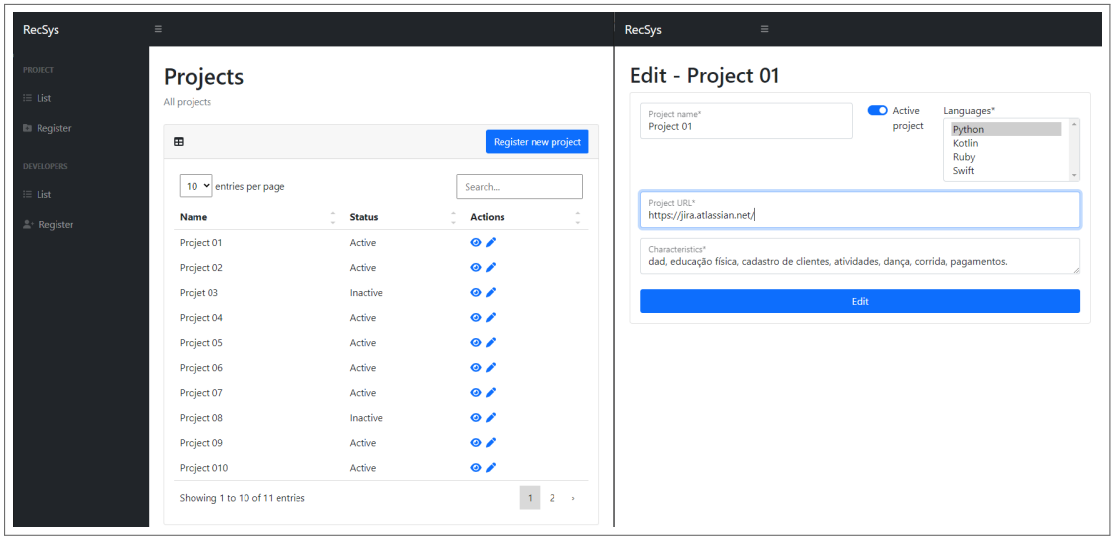


Fonte:Elaborado pelo autor(2023)

a qualquer momento através da interface mostrada na figura 10.

A interface mostrada na figura 11 dá acesso aos detalhes de um projeto específico, que foi selecionado na interface anterior. Ao abrir o detalhe de um projeto, o usuário consegue visualizar e ter acesso direto a área de alocação que contém quais desenvolvedores estão alocados no

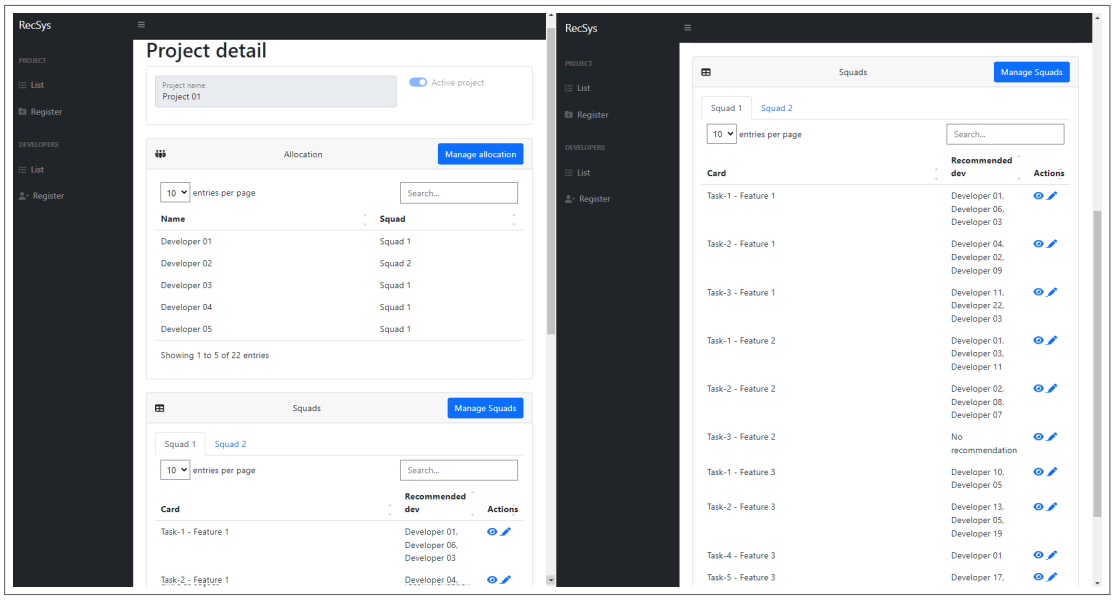
Figura 10 – Interface de listagem e edição dos dados dos projetos



Fonte:Elaborado pelo autor(2023)

projeto e a qual *squad*. A área das *Squads* que se encontra os *squads* ativos e quais tarefas ainda estão sem alocação na *sprint* atual da *squad* e que não foram iniciadas. Além disso, ao lado de cada tarefa, é exibida a recomendação sugerida pelo módulo de recomendação. Atualmente, o protótipo realiza a recomendação para a alocação, porém o usuário não consegue realizar a alocação direta no Jira através do nosso sistema.

Figura 11 – Interface de detalhe de um projeto



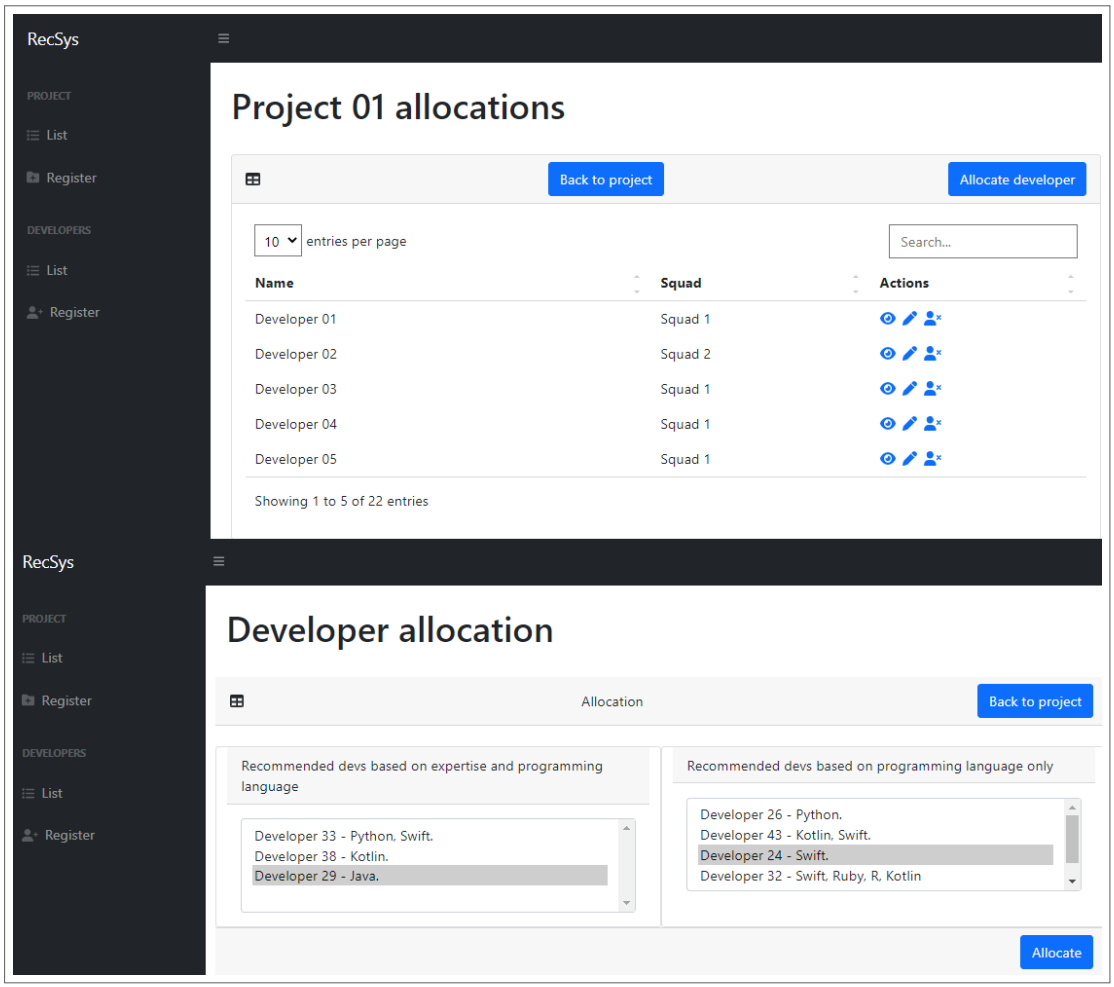
Fonte:Elaborado pelo autor(2023)

Na área de "alocação"(figura 12) , o usuário consegue ver inicialmente o desenvolvedor e em qual *squad* ele está alocado. Ao entrar no "gerenciamento de alocação", é possível alterar a

alocação do desenvolvedor, assim como alocar um novo desenvolvedor ao projeto, como visto na figura 12. A interface (figura 12) usada para realizar a alocação do desenvolvedor ao projeto possui duas opções: recomendação baseada exclusivamente nas linguagens que o desenvolvedor conhece, ou recomendação baseada nas linguagens conhecidas e nas experiências cadastradas em comparação com as características do projeto.

Uma vez concluída a alocação e definida a *squad* na qual o desenvolvedor irá atuar, automaticamente ao entrar na interface de "detalhe do projeto", o módulo de recomendação irá acessar os dados do desenvolvedor, e começará a incluir esse mesmo desenvolvedor nas recomendações.

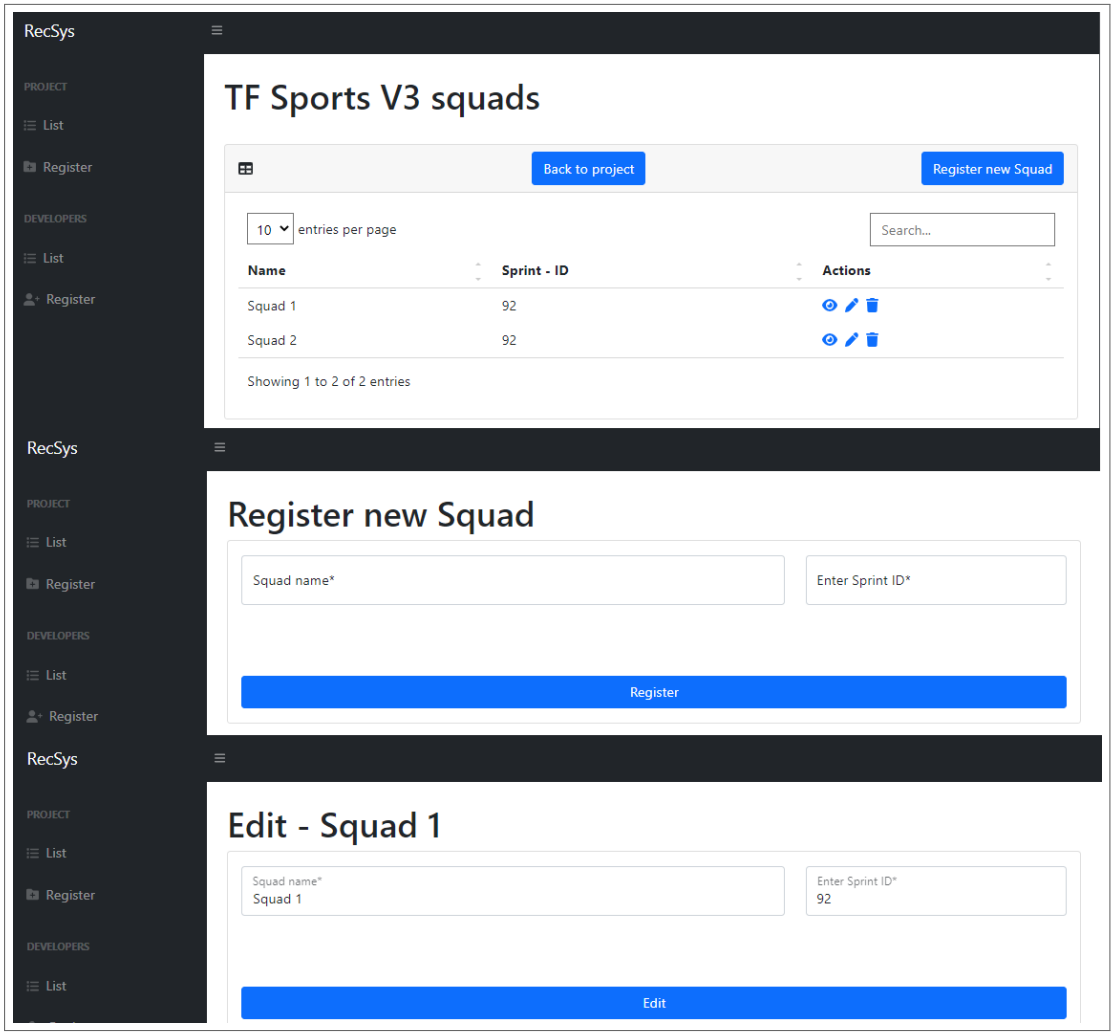
Figura 12 – Interface de detalhe da alocação do projeto



Fonte:Elaborado pelo autor(2023)

Ao adentrar na área de gerenciamento de *squad* (figura 13) na interface de detalhe do projeto, o usuário consegue atualizar o ID do Jira utilizado pelo módulo de coleta de informações, para saber para qual *sprint* ativa o usuário deseja coletar as tarefas. Nessa mesma área o usuário consegue cadastrar novas *squads* e remover as existentes.

Figura 13 – Interface de listagem, edição e registro das *Squads*



Fonte:Elaborado pelo autor(2023)

4.3 VALIDAÇÃO DO PROTÓTIPO

A validação do protótipo ocorreu ao longo de um período de dois meses, durante o qual registramos as atribuições dos desenvolvedores, feitas pelos gerentes, para as tarefas de entrada que exigiam alocação. Em seguida, essas alocações foram então comparadas com as recomendações feitas pelo protótipo desenvolvidos.

Inicialmente, o projeto e os desenvolvedores alocados no projeto foram cadastrados no protótipo. Em seguida, foi realizado o acompanhamento das *sprints*. No início de cada *sprint*, foram registradas todas as tarefas a serem realizadas e os desenvolvedores recomendados pelo protótipo. Ao término de cada sprint, realizou-se uma nova coleta das atividades que foram adicionadas ou removidas, e, a partir desses dados, uma nova rodada de recomendações foi gerada para as atividades recém-adicionadas. Além disso, registrou-se quais desenvolvedores

foram alocados pelos gerentes e executaram com sucesso as tarefas. Um total de 191 recomendações de atividades foram coletadas, as quais foram utilizadas na análise da validação do protótipo.

A coleta contínua de informações foi interrompida por dois motivos. Primeiramente, houve uma demanda significativa para entregar um grande volume de trabalho em um curto período de tempo. Isso levou a mudanças na alocação de desenvolvedores, que passou a ser baseada em disponibilidade em vez de experiência. Novos desenvolvedores foram alocados no projeto para lidar com a quantidade excessiva de tarefas. Embora essa situação representasse uma oportunidade para implantar a ferramenta e melhorar a distribuição de tarefas de forma eficiente, tanto os gerentes quanto os desenvolvedores não dispunham de tempo livre para aprender ou utilizar o protótipo.

O segundo motivo da interrupção da coleta foi uma reorganização das equipes, quando o time foi rearrumado em pequenas *squads* de 4 desenvolvedores para cada plataforma (Android, IOS, API, Front-end). A partir daí, as recomendações englobavam o time inteiro da *squad*, aumentando consideravelmente as chances de que o desenvolvedor responsável pela atividade fosse recomendado. Essa alteração influenciou as métricas de validação, resultando em uma taxa de acertos significativamente mais alta.

Antes de iniciar a avaliação quantitativa das recomendações em relação às alocações nas atividades efetivamente realizadas, foram selecionadas aleatoriamente 105 recomendações, a fim de analisar as razões pelas quais o protótipo as considerou. A análise revelou que as tarefas relacionadas a problemas (*bugs*) foram as mais relevantes para a realização das recomendações, visto que foram as atividades frequentemente com maior avaliação de similaridade. Isso pode ser atribuído a vários motivos, acredita-se que esse resultado se deve à descrição passo a passo do problema e ao resultado esperado, que utiliza critérios de aceitação ou são muito semelhantes a eles.

Outro cenário importante envolveu recomendações de tarefas semelhantes simplesmente por estarem na mesma área. Por exemplo, se uma tarefa envolvia melhorar a implementação de uma funcionalidade X em uma determinada área T, poderia ser recomendada uma tarefa relacionada a uma funcionalidade Y na mesma área em que a funcionalidade X seria implementada. Embora essa recomendação possa ser válida, considerando que a pessoa já trabalhou naquela área do aplicativo, pode não ser a melhor opção, pois pode haver outro desenvolvedor com experiência específica na funcionalidade X, mas em outra área do aplicativo. Ao analisarmos as tarefas anteriores desse desenvolvedor, descobrimos o motivo pelo qual essa recomendação

não foi feita: a descrição da tarefa relacionada à funcionalidade X não estava de acordo com o padrão da empresa e não continha nenhum detalhe sobre o que deveria ser implementado, apenas o título da tarefa. Isso resultou em uma pontuação de similaridade significativamente menor em comparação com as outras atividades, que apresentavam detalhes mais abrangentes sobre a área na qual a funcionalidade X seria implementada. Isso destaca a importância de um padrão de escrita que inclua detalhes da tarefa, como o que será implementado, em qual local será implementado e, se possível, como será a implementação.

Após uma análise detalhada das recomendações, procedeu-se ao cálculo da precisão dessas recomendações. A precisão é definida como a proporção de instâncias corretamente classificadas como positivas (verdadeiros positivos) em relação ao total de instâncias classificadas como positivas pelo modelo (verdadeiros positivos + falsos positivos). Nesse sentido, foi obtida uma precisão de 65,13%, o que indica que os resultados não são aleatórios e que o protótipo é capaz de realizar recomendações com base na experiência do desenvolvedor.

———————————

Também avaliamos o ranking dos desenvolvedores recomendados, para isso foram utilizadas as métricas de MRR[7] (*Mean Reciprocal Rank*) e o MAP[8] (*Mean Average Precision*), que são amplamente utilizadas para avaliar a eficácia de modelos de classificação, como no caso das recomendações dos desenvolvedores, onde os melhores deveriam no topo da lista de recomendações. Obtivemos os seguintes resultados, considerando os "top K"do ranking, com K variando de 1 a 3.

- K = 1: MRR@1 = 0.3228 e MAP@1 = 0.3228
- K = 2: MRR@2 = 0.5580 e MAP@2 = 0.4293
- K = 3: MRR@3 = 0.6760 e MAP@3 = 0.5185

Esses valores refletem o desempenho do protótipo na tarefa de recomendar os desenvolvedores corretos, priorizando a classificação dos mais adequados no topo da lista. Quanto mais próximo o MRR estiver de 1 e quanto maior for o valor do MAP, melhor é o desempenho do modelo. Essas métricas são indicativas de que as sugestões mais relevantes tendem a aparecer nas posições iniciais da lista de recomendação, demonstrando a capacidade do sistema em identificar e priorizar os desenvolvedores mais apropriados.

[7] Mean Reciprocal Rank (MRR) - https://machinelearning.wtf/terms/mean-reciprocal-rank-mrr/

[8] Mean Average Precision (MAP) - https://towardsdatascience.com/breaking-down-mean-average-precision-map-ae462f623a52

No entanto, devido à ausência de um feedback por parte dos gestores sobre as alocações que divergiram das recomendações, não é possível afirmar se as discrepâncias foram causadas por problemas na recomendação em si ou se as alocações foram baseadas em critérios como disponibilidade do desenvolvedor ou em sua oportunidade de aprender mais sobre a funcionalidade descrita na tarefa.

4.4 CONSIDERAÇÕES FINAIS

Este capítulo apresentou o processo proposto de recomendação de alocação de desenvolvedores a tarefa e a projetos, bem como protótipo implementado com base nesse processo. O objetivo principal do trabalho foi melhorar a realização das atividades, buscando sempre o melhor profissional para sua execução. Para alcançar esse objetivo, o sistema de recomendação desenvolvido utiliza dados históricos das atividades já realizadas.

O protótipo desenvolvido mostrou-se muito promissor na melhoria do processo de alocação de tarefas para desenvolvedores, apresentado uma taxa de precisão de 65,13% nos testes realizados. Através da análise de dados históricos e da aplicação de técnicas de recomendação, foi possível identificar os profissionais mais adequados para cada tarefa, levando em consideração suas experiências anteriores.

No próximo capítulo, serão apresentados as conclusões e indicações de trabalhos futuros a partir dos resultados desta pesquisa.

5 CONCLUSÃO E CONSIDERAÇÕES FINAIS

Este capítulo finaliza o documento de dissertação apresentando um panorama geral do trabalho realizado destacando suas principais contribuições e dificuldades encontradas no seu desenvolvimento. Por fim teremos uma seção de indicação de trabalhos futuros.

Este trabalho de mestrado explorou o tema de recomendação de especialistas para a tarefas e projetos de engenharia de software. Ao longo dos capítulos anteriores, examinamos os principais conceitos, teorias e tendências relacionados ao assunto, analisamos estudos de caso relevantes e discutimos as implicações práticas e teóricas das descobertas feitas.

Nesse contexto, nosso projeto propôs um processo de construção de sistemas de recomendação, e constrói um protótipo para auxiliar no processo de alocação de desenvolvedores a tarefas, com o objetivo de melhorar a eficiência e a qualidade das atividades realizadas. O protótipo foi desenvolvido em Python utilizando o framework Django e fez uso das bibliotecas Jira, NLTK, Pandas e Scikit-learn. O protótipo passou por período experimental, onde foram coletados dados durante dois meses, registrando quais desenvolvedores foram alocados para que tarefas, e quais deles foram também recomendados pela ferramenta.

A análise qualitativa realizada nas recomendações selecionadas aleatoriamente revelou que as tarefas relacionadas a problemas (bugs) foram as mais relevantes para as recomendações. O protótipo conseguiu alcançar uma precisão calculada de 65,13% nas recomendações, indicando que o protótipo é capaz de realizar recomendações com base na experiência dos desenvolvedores. No entanto, devido à falta de *feedback* dos gestores sobre as alocações divergentes, não foi possível determinar se as discrepâncias foram causadas por problemas nas recomendações ou por outros fatores da empresa, contudo, mesmo com a falta dos *feedback*, acreditamos que o protótipo seja um caso de sucesso.

No geral, o protótipo desenvolvido demonstrou potencial para melhorar o processo de alocação de desenvolvedores a tarefas, facilitando a busca pelo profissional mais adequado para cada atividade. Embora o protótipo tenha sido desenvolvido levando em conta as necessidades e particularidades da empresa parceira, a abordagem utilizada pode ser adaptada e aplicada em outros contextos. A alocação de recursos humanos é um desafio comum em muitas empresas, independentemente do setor de atuação, e a utilização de técnicas de recomendação pode trazer benefícios significativos em termos de eficiência e produtividade.

5.1 PRINCIPAIS CONTRIBUIÇÕES

Esta seção destaca as diversas contribuições relevantes, que incluem:

- Realização de um estudo aprofundado sobre sistemas de recomendação de especialistas. Esse estudo abrange uma análise detalhada do estado atual dessa área, considerando seu impacto e relevância no contexto atual. Essa análise fornece insights valiosos sobre os desafios e as melhores práticas relacionadas aos sistemas de recomendação de especialistas;

- Apresentação de trabalhos relevantes que apresentam outros sistemas de recomendação de especialistas, abordando suas arquiteturas e modelos subjacentes. Essa revisão da literatura fornece uma visão das pesquisas e soluções existentes, permitindo a comparação e o entendimento dos avanços alcançados nesse campo. Isso contribui para uma compreensão mais ampla do tema e para o embasamento teórico do estudo;

- Proposição de um processo específico para sistemas de recomendação de especialistas. Esse processo foi projetado com base nas necessidades e nos requisitos identificados, considerando as peculiaridades do contexto de aplicação. Essa arquitetura fornece um modelo conceitual sólido para a implementação de sistemas de recomendação de especialistas, que orientou o desenvolvimento do nosso protótipo;

- Desenvolvimento de um protótipo funcional que pode ser facilmente integrado ao Jira, com poucas alterações necessárias para se adequar ao projeto em questão. Além disso, a integração com uma ferramenta amplamente utilizada como o Jira aumentará a aplicabilidade e a relevância do protótipo, tornando-o mais prático e útil no ambiente de trabalho.

Essas contribuições fornecem uma base sólida para a dissertação, permitindo explorar e discutir a relevância dos sistemas de recomendação de especialistas, apresentar uma revisão da literatura abrangente, propor uma arquitetura específica e desenvolver um protótipo funcional. Essa abordagem abrangente contribuirá para o avanço do conhecimento nessa área e para a aplicabilidade dos sistemas de recomendação de especialistas no ambiente empresarial.

5.2 DIFICULDADES ENCONTRADAS

Durante o processo de desenvolvimento do protótipo, enfrentamos diversas dificuldades que precisaram ser superadas. Listamos abaixo os principais problemas enfrentados.

- O acesso inicial às informações necessárias para a construção do protótipo foi um desafio. Foi preciso aguardar um tempo considerável para obter acesso legal aos dados da empresa, e então conseguir modelar o sistema de dados de forma adequada. Esse processo inicial de obtenção de dados retardou o progresso do desenvolvimento;
- Outro obstáculo que enfrentamos foi a falta de um padrão fixo para a escrita das tarefas por parte da empresa, especialmente durante as transições de gestores do projeto. Isso resultou na necessidade de implementar um extrator de informação específico para cada modelo adotado. Essa falta de uniformidade atrasou a implementação do protótipo e reduziu o tempo disponível para a validação. Além disso, aumentou a complexidade do módulo de extração de informações, tornando-o mais desafiador de ser desenvolvido;
- Por fim, tivemos dificuldades em definir o modelo de recomendação para as tarefas. Optamos por não utilizar o modelo mais comum, baseado no cálculo de pontos dos desenvolvedores, optando por recomendação baseada em trabalhos similares mais realizados em outros sistemas. Essa escolha exigiu uma análise minuciosa das melhores práticas existentes e sua adaptação ao nosso contexto específico.

5.3 LIMITAÇÕES DO SISTEMA

O protótipo apresenta algumas limitações que ainda precisam ser superadas antes de sua utilização plena. Essas limitações incluem:

- Falta de integração completa com o sistema de gestão de tarefas utilizado pela empresa. Isso resulta na incapacidade do protótipo em reconhecer diretamente os projetos já cadastrados e os desenvolvedores ativos neles;
- Como já mencionado, atualmente os gerentes não utilizam um padrão único para descrever as novas tarefas. Assim, não foi possível definir regras padronizadas para a extração da informação. Para garantir uma recomendação mais precisa, é necessário estabelecer uma padronização rigorosa das tarefas;

- O protótipo não é capaz de lidar automaticamente ou via interface com novos campos de informação cadastrados posteriormente nas tarefas. Isso significa que, quando novos campos forem adicionados, o protótipo não terá a capacidade de extrair automaticamente essas informações, sendo necessário que haja sempre um esforço de desenvolvimento, o que limita sua adaptabilidade a mudanças futuras;

- O protótipo atual não consegue realizar a extração de conhecimento para tarefas que requerem a colaboração de múltiplos desenvolvedores. No momento, o protótipo não tem a capacidade de discernir as contribuições individuais dentro de uma tarefa coletiva. O protótipo baseia-se unicamente no campo "responsável"para determinar o desenvolvedor encarregado de uma tarefa específica. Contudo, situações podem surgir onde múltiplos desenvolvedores estão envolvidos na mesma tarefa, mesmo que seus nomes não constem no campo de "responsável". Nessas instâncias, o protótipo não consegue atribuir a esses desenvolvedores o conhecimento vinculado àquela tarefa, limitando, assim, a amplitude das recomendações proporcionadas pelo sistema.

Essas limitações representam desafios significativos que precisam ser abordados para garantir o pleno funcionamento e a eficácia do protótipo de recomendação no contexto empresarial.

5.4 TRABALHOS FUTUROS

Esta seção propõe trabalhos futuros que podem tratar algumas das limitações apontadas acima. São eles:

- Estender o escopo do sistema para outras áreas das empresas, além da recomendação de desenvolvedores, possibilitando a recomendação de diferentes especialistas (e.g., designers, testadores e PO) de diversas áreas para tarefas específicas, levando em consideração o tipo de cada tarefa (e.g., design, aprovação do cliente, desenvolvimento e teste). Dessa forma, o sistema poderia fornecer recomendações mais abrangentes e alinhadas com as necessidades de cada etapa da tarefa;

- Aprimorar o fluxo de recomendação, especialmente no que diz respeito à bonificação da posição dos especialistas que possuam mais de uma tarefa recomendada. Esse aperfeiçoamento foi discutido no Capítulo 4, na seção de recomendação para tarefas;

- Explorar a possibilidade de um modelo que combine a recomendação com base em tarefas anteriores e a pontuação da expertise do profissional na área relacionada à tarefa atual. Essa abordagem poderia levar em conta tanto o histórico do profissional quanto a sua competência específica para a tarefa em questão, aprimorando a precisão das recomendações e tornando-as mais personalizadas;
- Realizar uma validação mais extensa do sistema, em termos de tempo e projetos abrangidos. Isso permitiria identificar pontos falhos no processo e possíveis melhorias a serem implementadas. Ao submeter o protótipo a um período mais longo de validação e envolvendo uma variedade maior de projetos, seria possível obter feedback valioso e refinamentos adicionais para o sistema;
- Realizar uma pesquisa para aprimorar o modelo atual de coleta de informações que busque uma abordagem mais generalista. Essa abordagem deve ser capaz de extrair informações corretas, independentemente do modelo adotado na escrita das tarefas.

Esses pontos de aprimoramento representam um caminho promissor para o desenvolvimento futuro do protótipo, visando atender às necessidades da empresa de forma mais abrangente e eficiente, bem como superar as limitações e dificuldades identificadas ao longo do processo.

REFERÊNCIAS

AKBAR, A.; AGARWAL, P.; OBAID, A. Recommendation engines-neural embedding to graph-based: Techniques and evaluations. *International Journal of Nonlinear Analysis and Applications*, Semnan University, v. 13, n. 1, p. 2411–2423, 2022. ISSN 2008-6822. Disponível em: <https://ijnaa.semnan.ac.ir/article_5941.html>.

ALARFAJ, F.; KRUSCHWITZ, U.; HUNTER, D.; FOX, C. Finding the right supervisor: Expert-finding in a university domain. In: *Proceedings of the NAACL HLT 2012 Student Research Workshop*. Montréal, Canada: Association for Computational Linguistics, 2012. p. 1–6. Disponível em: <https://aclanthology.org/N12-2001>.

ALBUSAC, C.; CAMPOS, L. M. de; FERNáNDEZ-LUNA, J. M.; HUETE, J. F. Content-based recommendation for academic expert finding. In: *Proceedings of the 5th Spanish Conference on Information Retrieval*. New York, NY, USA: Association for Computing Machinery, 2018. (CERI '18). ISBN 9781450365437. Disponível em: <https://doi.org/10.1145/3230599.3230607>.

ALMONTE, L.; GUERRA, E.; CANTADOR, I.; LARA, J. de. Recommender systems in model-driven engineering. *Software and Systems Modeling*, v. 21, n. 1, p. 249–280, Feb 2022. ISSN 1619-1374. Disponível em: <https://doi.org/10.1007/s10270-021-00905-x>.

ANVIK, J.; HIEW, L.; MURPHY, G. C. Who should fix this bug? In: *Proceedings of the 28th International Conference on Software Engineering*. New York, NY, USA: Association for Computing Machinery, 2006. (ICSE '06), p. 361–370. ISBN 1595933751. Disponível em: <https://doi.org/10.1145/1134285.1134336>.

ARAPAKIS, I.; MOSHFEGHI, Y.; JOHO, H.; REN, R.; HANNAH, D.; JOSE, J. M. Enriching user profiling with affective features for the improvement of a multimodal recommender system. In: *Proceedings of the ACM International Conference on Image and Video Retrieval*. New York, NY, USA: Association for Computing Machinery, 2009. (CIVR '09). ISBN 9781605584805. Disponível em: <https://doi.org/10.1145/1646396.1646433>.

BARBOSA, J. d. M. Recsc: Um progressive webapp para recomendação sensível ao contexto. v. 2019, 2019.

BARTH, F. Modelando o perfil do usuário para a construção de sistemas de recomendação: um estudo teórico e estado da arte. *Revista de Sistemas de Informação da FSMA*, v. 6, p. 59–71, 2010. Disponível em: <http://fbarth.net.br/docs/bar2010a.pdf>.

CARVALHO, M. F. d. *Um processo para construção de tesauros de domínio específico no contexto de uma empresa de teste de software*. Dissertação (Mestrado) — Universidade Federal de Pernambuco, 2021. Disponível em: <https://repositorio.ufpe.br/handle/123456789/44252>.

CHUN, I.-G.; HONG, I.-S. The implementation of knowledge-based recommender system for electronic commerce using java expert system library. In: *ISIE 2001. 2001 IEEE International Symposium on Industrial Electronics Proceedings (Cat. No.01TH8570)*. [S.l.: s.n.], 2001. v. 3, p. 1766–1770 vol.3.

DAT, N. V.; TOAN, P. V.; THANH, T. M. Solving distribution problems in content-based recommendation system with gaussian mixture model. *Applied Intelligence*, v. 52, n. 2, p. 1602–1614, Jan 2022. ISSN 1573-7497. Disponível em: <https://doi.org/10.1007/s10489-021-02429-9>.

DAVOODI, E.; AFSHARCHI, M.; KIANMEHR, K. A social network-based approach to expert recommendation system. In: CORCHADO, E.; SNÁŠEL, V.; ABRAHAM, A.; WOŹNIAK, M.; GRAÑA, M.; CHO, S.-B. (Ed.). *Hybrid Artificial Intelligent Systems: 7th International Conference, HAIS 2012, Salamanca, Spain, March 28-30th, 2012. Proceedings, Part I 7*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012. p. 91–102. ISBN 978-3-642-28942-2.

DEY, A. K.; ABOWD, G. D.; SALBER, D. A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. *Human–Computer Interaction*, Taylor & Francis, v. 16, n. 2-4, p. 97–166, 2001. Disponível em: <https://doi.org/10.1207/S15327051HCI16234_02>.

FILHO, E. L. d. R. F. et al. Abordagem inteligente para alocação de tarefas em projetos de software. Universidade Federal de Santa Maria, 2022. Disponível em: <http://repositorio.ufsm.br/handle/1/25786>.

FU, W.; PENG, Z.; WANG, S.; XU, Y.; LI, J. Deeply fusing reviews and contents for cold start users in cross-domain recommendation systems. *Proceedings of the AAAI Conference on Artificial Intelligence*, v. 33, n. 01, p. 94–101, Jul. 2019. Disponível em: <https://ojs.aaai.org/index.php/AAAI/article/view/3773>.

GUPTA, G.; KATARYA, R. Recommendation analysis on item-based and user-based collaborative filtering. In: *2019 International Conference on Smart Systems and Inventive Technology (ICSSIT)*. [S.l.: s.n.], 2019. p. 1–4.

HERCE-ZELAYA, J.; PORCEL, C.; BERNABé-MORENO, J.; TEJEDA-LORENTE, A.; HERRERA-VIEDMA, E. New technique to alleviate the cold start problem in recommender systems using information from social media and random decision forests. *Information Sciences*, v. 536, p. 156–170, 2020. ISSN 0020-0255. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0020025520304916>.

KO, H.; LEE, S.; PARK, Y.; CHOI, A. A survey of recommendation systems: Recommendation models, techniques, and application fields. *Electronics*, v. 11, n. 1, 2022. ISSN 2079-9292. Disponível em: <https://www.mdpi.com/2079-9292/11/1/141>.

LAAN, R. H. v. d.; FERREIRA, G. I. S. Thesaurus e terminologia. In: *Congresso Brasileiro de Biblioteconomia e Documentação,(19.: 2000: Porto Alegre, RS). Anais. Porto Alegre, 2000*. [s.n.], 2000. Disponível em: <http://hdl.handle.net/10183/10208>.

LAM, X. N.; VU, T.; LE, T. D.; DUONG, A. D. Addressing cold-start problem in recommendation systems. In: *Proceedings of the 2nd International Conference on Ubiquitous Information Management and Communication*. New York, NY, USA: Association for Computing Machinery, 2008. (ICUIMC '08), p. 208–211. ISBN 9781595939937. Disponível em: <https://doi.org/10.1145/1352793.1352837>.

LI, J. A recommendation model for college english digital teaching resources using collaborative filtering and few-shot learning technology. *Computational Intelligence and Neuroscience*, Hindawi, v. 2022, p. 1233057, Jul 2022. ISSN 1687-5265. Disponível em: <https://doi.org/10.1155/2022/1233057>.

LIN, S.; HONG, W.; WANG, D.; LI, T. A survey on expert finding techniques. *Journal of Intelligent Information Systems*, v. 49, n. 2, p. 255–279, Oct 2017. ISSN 1573-7675. Disponível em: <https://doi.org/10.1007/s10844-016-0440-5>.

LOH, S.; LICHTNOW, D.; GARIN, R. S. Arquitetura de um sistema de recomendação para apoio à colaboração. In: . VIII Congreso Argentino de Ciencias de la Computación, 2002. p. 846–857. Disponível em: <http://sedici.unlp.edu.ar/handle/10915/23126>.

MATTER, D.; KUHN, A.; NIERSTRASZ, O. Assigning bug reports using a vocabulary-based expertise model of developers. In: IEEE. *2009 6th IEEE International Working Conference on Mining Software Repositories*. 2009. p. 131–140. Disponível em: <https://ieeexplore.ieee.org/abstract/document/5069491>.

MEEHAN, K.; LUNNEY, T.; CURRAN, K.; MCCAUGHEY, A. Context-aware intelligent recommendation system for tourism. In: IEEE. *2013 IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops)*. [S.l.], 2013. p. 328–331.

MIDDLETON, S. E.; SHADBOLT, N. R.; ROURE, D. C. D. Capturing interest through inference and visualization: Ontological user profiling in recommender systems. In: *Proceedings of the 2nd International Conference on Knowledge Capture*. New York, NY, USA: Association for Computing Machinery, 2003. (K-CAP '03), p. 62–69. ISBN 1581135831. Disponível em: <https://doi.org/10.1145/945645.945657>.

MIDDLETON, S. E.; SHADBOLT, N. R.; ROURE, D. C. D. Ontological user profiling in recommender systems. *ACM Trans. Inf. Syst.*, Association for Computing Machinery, New York, NY, USA, v. 22, n. 1, p. 54–88, jan 2004. ISSN 1046-8188. Disponível em: <https://doi.org/10.1145/963770.963773>.

MORAIS, A. J.; NETO, J.; OLIVEIRA, E.; JORGE, A. M. Sistema de recomendação web usando agentes. *Revista de Ciências da Computação*, Universidade Aberta, p. 23–35, 2013. Disponível em: <http://hdl.handle.net/10400.2/3092>.

MOREIRA, C.; WICHERT, A. Finding academic experts on a multisensor approach using shannon's entropy. *Expert Systems with Applications*, v. 40, n. 14, p. 5740–5754, 2013. ISSN 0957-4174. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0957417413002388>.

NAWARA, D.; KASHEF, R. Context-aware recommendation systems in the iot environment (iot-cars)–a comprehensive overview. *IEEE Access*, IEEE, v. 9, p. 144270–144284, 2021.

NIKZAD–KHASMAKHI, N.; BALAFAR, M.; Reza Feizi–Derakhshi, M. The state-of-the-art in expert recommendation systems. *Engineering Applications of Artificial Intelligence*, v. 82, p. 126–147, 2019. ISSN 0952-1976. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0952197619300703>.

PATIL, S. Team formation using recommendation systems. New Jersey Institute of Technology, 2020.

PINHEIRO, P. L. B. et al. Uma metodologia em cascata de quatro etapas para classificar códigos ncm usando técnicas de pln. Campus Universitário de Tucuruí, Universidade Federal do Pará, Tucuruí, p. 18, 2022. Disponível em: <https://bdm.ufpa.br:8443/jspui/handle/prefix/5010>.

QIU, Y.; FREI, H.-P. Concept based query expansion. In: *Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. New York, NY, USA: Association for Computing Machinery, 1993. (SIGIR '93), p. 160–169. ISBN 0897916050. Disponível em: <https://doi.org/10.1145/160688.160713>.

SARWAR, B.; KARYPIS, G.; KONSTAN, J.; RIEDL, J. Item-based collaborative filtering recommendation algorithms. In: *Proceedings of the 10th International Conference on World Wide Web*. New York, NY, USA: Association for Computing Machinery, 2001. (WWW '01), p. 285–295. ISBN 1581133480. Disponível em: <https://doi.org/10.1145/371920.372071>.

SILVA, C.; RIBEIRO, B. The importance of stop word removal on recall values in text categorization. In: IEEE. *Proceedings of the International Joint Conference on Neural Networks, 2003.* [S.l.], 2003. v. 3, p. 1661–1666 vol.3.

SILVA, P. A. de O.; FELÍCIO, C. Z.; PAIXÃO, K. Sistema de recomendação de desenvolvedores para projetos de software. *Periódico de Pesquisas e Trabalhos de conclusão de curso IFTM-Campus Uberlândia Centro*, 2018.

SOUSA, S. J. de; DIAS, T. M. R.; PINTO, A. L. *A Strategy for Expert Recommendation From Open Data Available on the Lattes Platform*. eprint arXiv:1906.06437, 2019. Disponível em: <https://arxiv.org/abs/1906.06437>.

SUNDERMANN, C. V. *Extração de informação contextual utilizando mineração de textos para sistemas de recomendação sensíveis ao contexto*. Tese (Doutorado) — Universidade de São Paulo, São Carlos : Instituto de Ciências Matemáticas e de Computação, University of São Paulo, 2015. Disponível em: <https://doi.org/10.11606/D.55.2015.tde-10082015-192318>.

TAHMASEBI, F.; MEGHDADI, M.; AHMADIAN, S.; VALIALLAHI, K. A hybrid recommendation system based on profile expansion technique to alleviate cold start problem. *Multimedia Tools and Applications*, v. 80, n. 2, p. 2339–2354, Jan 2021. ISSN 1573-7721. Disponível em: <https://doi.org/10.1007/s11042-020-09768-8>.

VARGAS, S.; CASTELLS, P. Rank and relevance in novelty and diversity metrics for recommender systems. In: *Proceedings of the Fifth ACM Conference on Recommender Systems*. New York, NY, USA: Association for Computing Machinery, 2011. (RecSys '11), p. 109–116. ISBN 9781450306836. Disponível em: <https://doi.org/10.1145/2043932.2043955>.

WANG, P.; YE, H. A personalized recommendation algorithm combining slope one scheme and user based collaborative filtering. In: IEEE. *2009 International Conference on Industrial and Information Systems*. [S.l.], 2009. p. 152–154.

WEI, J.; HE, J.; CHEN, K.; ZHOU, Y.; TANG, Z. Collaborative filtering and deep learning based recommendation system for cold start items. *Expert Systems with Applications*, v. 69, p. 29–39, 2017. ISSN 0957-4174. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0957417416305309>.

ZHANG, J.; LIN, Z.; XIAO, B.; ZHANG, C. An optimized item-based collaborative filtering recommendation algorithm. In: IEEE. *2009 IEEE International Conference on Network Infrastructure and Digital Content*. [S.l.], 2009. p. 414–418.