# Interação Humano-Dados no contexto de recomendação para desenvolvedores de projetos de software open-source do GitHub

**Kattiana Constantino[1], Raquel Prates[2], e Eduardo Figueiredo[2]**

[1]Universidade Federal dos Vales do Jequitinhonha e Mucuri

[2]Universidade Federal de Minas Gerais

{kattiana.constantino}@ufvjm.edu.br

***Abstract.*** *Active collaboration is essential for the success of software projects throughout the development lifecycle. Unfortunately, on social coding platforms such as GitHub, it is still challenging for developers to identify potential collaborators and thus improve the quality of contributions. To this end, we implemented developer recommendation strategies, supported by a visual and interactive tool, called* COOPFINDER*, to connect collaborators based on a set of files of interest to them. In addition, the tool provides metadata and links different attributes that cannot be analyzed using the GitHub interface, facilitating the decision-making of collaborators.*

***Resumo.*** *A colaboração ativa é essencial para o sucesso de projetos de software em todo o ciclo de vida do desenvolvimento. Infelizmente, em plataformas de codificação social, como o GitHub, ainda é desafiador para os desenvolvedores identificar potenciais colaboradores e por conseguinte, melhorar a qualidade das contribuições. Para esse fim, implementamos estratégias de recomendação de desenvolvedores, apoiadas por uma ferramenta visual e interativa, chamada* COOPFINDER*, para conectar colaboradores com base em um conjunto de arquivos de seus interesses. Além disso, a ferramenta fornece metadados e vincula diferentes atributos que não podem ser analisados usando a interface do GitHub, facilitando a tomada de decisão dos colaboradores.*

## 1. Introdução

Os desenvolvedores de software deveriam colaborar em todos os estágios do ciclo de vida do software para criar produtos de qualidade. No entanto, para grandes projetos, com centenas de desenvolvedores dinâmicos, como vários projetos de código aberto bem-sucedidos, pode ser muito complexo encontrar desenvolvedores com os mesmos interesses e, assim, obter colaborações adequadas e novos *insights*. Recursos e esforços podem ser desperdiçados no contexto do projeto, desencorajando desenvolvedores a permanecer. Pode ser custoso gerenciar tantas contribuições, o que é outra questão para o mantenedor que deseja aproveitar essa pequena, modesta, mas útil contribuição feita por um desenvolvedor voluntário no menor tempo possível.

Este trabalho tem como objetivo discutir a Interação Humano-Dados no contexto de recomendações para desenvolvedores de software open-source, do GitHub. O propósito é que desenvolvedores, líderes, mantenedores e pesquisadores, na tomada

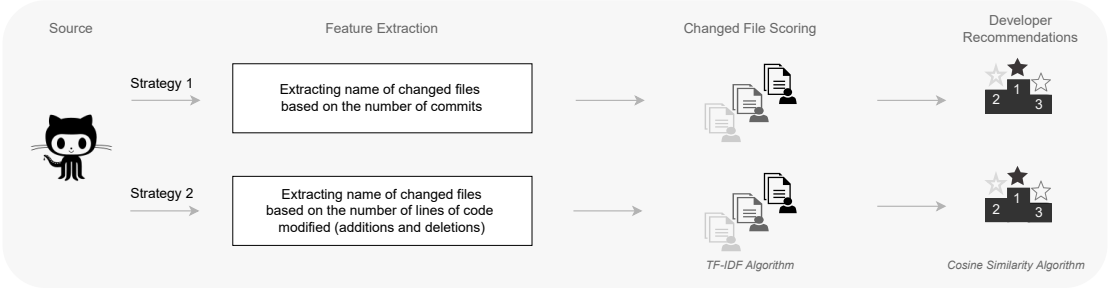


**Figura 1. Duas estratégias de recomendação de desenvolvedores [Constantino et al. 2023a].**

de decisão, tenham um maior entendimento sobre como melhorar as oportunidades de colaboração entre desenvolvedores em um projeto específico.

## 2. CoopFinder - Ferramenta de Recomendação de Desenvolvedores

As estratégias para recomendar desenvolvedores são suportadas pelo COOPFINDER, uma ferramenta de protótipo que aprimora as oportunidades de colaboração em um projeto com base em arquivos co-alterados. Esses arquivos co-alterados referem-se a arquivos que dois ou mais desenvolvedores modificaram. COOPFINDER é um aplicativo da web interativo e visualmente rico que ajuda a conectar desenvolvedores desses arquivos.

COOPFINDER implementa duas estratégias de recomendação do desenvolvedor, a saber, ESTRATÉGIAS 1 e 2, que são baseadas em arquivos co–alterados. Para ESTRATÉGIA 1, o "número de *commits*" foi extraído para determinar a frequência de modificações de arquivo por um desenvolvedor. Para ESTRATÉGIA 2, o "número de linhas de código alteradas (LoC)" foi extraído. Esta métrica calcula a soma de linhas de código adicionadas e removidas por um desenvolvedor em um arquivo específico [Constantino et al. 2023a]. A Figura 1 apresenta uma visão geral das etapas necessárias para recomendar um desenvolvedor a outro desenvolvedor no projeto de software, como a seguir.

**Etapa 1 – Extração de recursos:** O histórico de modificações feito por todos os desenvolvedores em um projeto foi extraído em relação às entradas, conforme ilustrado na Figura 1. A plataforma GitHub, uma rede social que hospeda projetos e suporta o modelo fork & pull, foi utilizada para esse propósito. Os desenvolvedores do GitHub criam cópias do repositório original e fazem alterações em suas respectivas cópias. Depois que essas alterações são finalizadas, eles têm a opção de enviá-las de volta ao repositório original por meio de uma solicitação de pull.

**Etapa 2 – Pontuação de arquivos alterados:** Realizamos mineração de arquivos para cada desenvolvedor do projeto extraindo o conjunto de arquivos co-alterados. Esse conjunto de arquivos foi então classificado usando o algoritmo Term Frequency – Inverse Document Frequency (TF–IDF) [Salton 1989]. A classificação resultante de arquivos relevantes para cada desenvolvedor é apresentada na Figura 1. Repetimos essa etapa para ambas as estratégias, produzindo classificações diferentes para cada desenvolvedor.

**Etapa 3 – Modelo de recomendação do desenvolvedor:** A classificação de arquivos relevantes para cada desenvolvedor do projeto, calculada usando o modelo de

espaço vetorial, foi utilizada para calcular sua similaridade por meio da métrica de cosseno amplamente usada [Salton 1971, Salton and Harman 2003]. Essa métrica tem sido amplamente empregada [Rahman et al. 2016, Franco et al. 2019] devido à sua capacidade de quantificar a similaridade de dois objetos [Ricci et al. 2011]. Repetimos essa etapa para cada estratégia, conforme mostrado na Figura 1.

### 2.1. Tecnologias de implementação

COOPFINDER é uma ferramenta Web, construída sobre uma arquitetura cliente-servidor, aproveitando técnicas de visualização eficazes para aprimorar a experiência do usuário. O lado do servidor é desenvolvido usando Python 3[1], com o suporte das bibliotecas scikit-learn[2], uma biblioteca de aprendizado de máquina robusta e gratuita para Python. Para as visualizações em COOPFINDER, integramos HighCharts[3], uma poderosa biblioteca JavaScript especializada em visualização de dados analíticos. Esta biblioteca facilita a manipulação de documentos com base em dados, contribuindo para uma interface de usuário atraente e informativa. Para garantir uma experiência de usuário contínua e interativa, incorporamos componentes Bootstrap Framework[4]. Esses componentes abrangem uma variedade de folhas de estilo e plugins jQuery[5], permitindo a criação de uma interface responsiva e dinâmica. Nossa seleção dessas tecnologias é proposital, motivada pelo compromisso em fornecer uma experiência dinâmica de exploração e visualização para nossos usuários.

### 2.2. Interface e interação

A Figura 2 mostra a tela do COOPFINDER relacionadas à lista de colaboradores de um projeto selecionado. Esta lista inclui todos os colaboradores que modificaram quaisquer arquivos em suas cópias (*fork*) de um projeto selecionado do GitHub, conforme descrito na Seção 2. Na Figura 2, o Quadro (A) exibe informações do projeto, como o nome do repositório, número de estrelas, número de bifurcações e número de problemas abertos, aos quais os colaboradores pertencem. O Quadro (B) apresenta uma tabela de todos os colaboradores do projeto selecionado. Para cada colaborador, a tabela fornece suas informações de desenvolvedor, incluindo seu avatar, nome, bifurcação, número de seguidores, número de seguidores, número de confirmações no upstream, número de confirmações não mescladas e a data de sua última confirmação. O quadro (C) exibe a atividade do código para commits upstream e não mesclados, juntamente com a última data de commit. Isso ajuda os usuários a avaliar o status dos colaboradores no projeto, incluindo seu nível de atividade com base em commits mesclados e na última data de commit. Commits não mesclados recentes podem sinalizar uma necessidade de assistência. Além disso, os mantenedores podem revisar os interesses dos colaboradores no projeto ou criar equipes em torno de seus arquivos co-alterados. Finalmente, o botão *"Executar"* executa os algoritmos das recomendações e os resultados são apresentados da seguinte forma.

A figura 2 descreve uma captura de tela do COOPFINDER com uma lista de colaboradores recomendados para o desenvolvedor alvo, que pode variar dependendo da

[1]https://www.python.org/
[2]https://scikit-learn.org/stable/index.html
[3]https://www.highcharts.com/
[4]https://getbootstrap.com/
[5]https://jquery.com

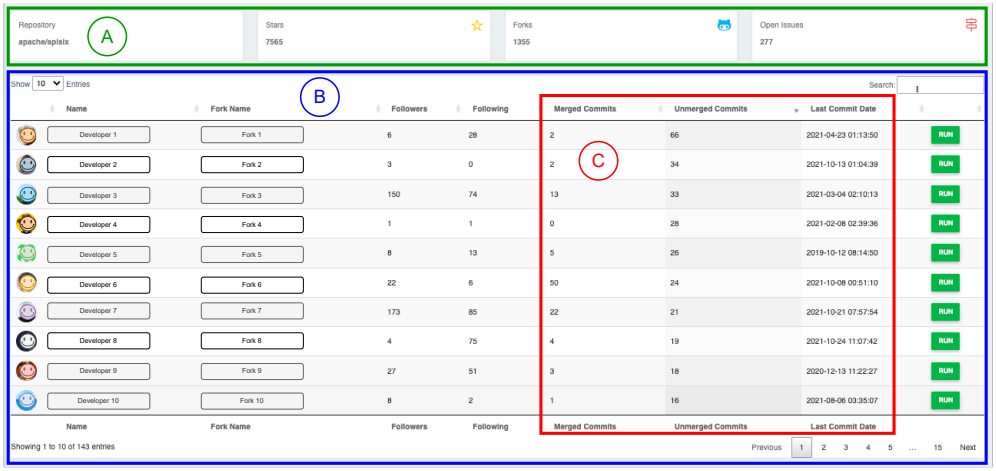


**Figura 2. Visão geral das informações dos colaboradores de um projeto específico do GitHub [Constantino and Figueiredo 2022].**

estratégia selecionada e da classificação de arquivos relevantes para cada desenvolvedor do projeto.O quadro (D) exibe informações do projeto, como o nome do repositório, número de estrelas, número de bifurcações e número de problemas abertos, aos quais os colaboradores recomendados pertencem. O quadro (E) apresenta informações sobre o desenvolvedor alvo, como seu nome, avatar, data do último commit, número total de commits, seguidores e seguidores. Por fim, o quadro (F) mostra uma lista de desenvolvedores recomendados com interesses semelhantes com base em arquivos co-alterados.

No quadro (F), os usuários podem selecionar um dos colaboradores recomendados para comparar com o desenvolvedor alvo mostrado no quadro (E). Uma vez selecionado, o quadro (G) exibe os dois colaboradores escolhidos junto com seus nomes e bifurcações, vinculados ao seu perfil do GitHub. O Frame (H) permite que os usuários analisem os arquivos comuns dos dois desenvolvedores. Por exemplo, "*t/discovery/nacos.t*" e "*api-six/discovery/nacos.lua*" são arquivos comuns que ambos os desenvolvedores. Eles estão interessados e familiarizados com esses arquivos (Figura 3). Finalmente, no Frame (I), é apresentada a expertise do desenvolvedor recomendado (linguagem de programação) relacionada ao projeto em foco. Essa expertise é calculada como uma porcentagem do número total de arquivos alterados em cada linguagem de programação. Observe que esse recurso não é o foco principal do nosso trabalho atual. No entanto, deixamos esse espaço aberto para possíveis caminhos para pesquisas futuras. Outros trabalhos, como [Oliveira et al. 2019, Oliveira et al. 2020, de Neira et al. 2018], exploraram a expertise dos desenvolvedores.

## 3. Por que recomendar desenvolvedores?

Todas as contribuições devem ser apreciadas e encorajadas [Pham et al. 2013, Gousios et al. 2014, Pinto et al. 2016]. Trabalhos anteriores mostram que os desenvolvedores geralmente pedem ajuda aos membros da equipe principal, que devem compartilhar sua motivação, conhecimento e experiência [Minto and Murphy 2007, Kononenko et al. 2016]. No entanto, isso pode não funcionar sempre, pois os membros da equipe principal podem estar muito ocupados para responder [Yu et al. 2015, Gousios et al. 2015, Steinmacher et al. 2018]. Assim, outros desenvolvedores experien-

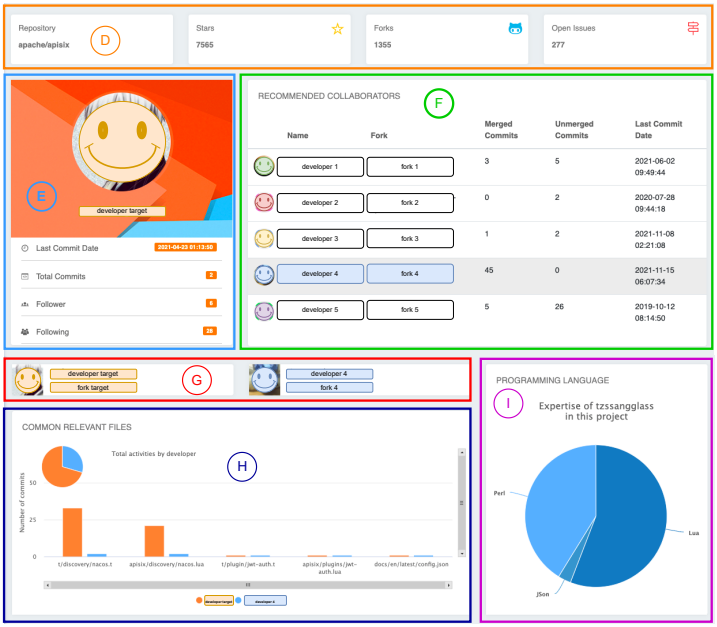


**Figura 3. Overview of developer recommendations and their aggregated information [Constantino and Figueiredo 2022].**

tes fora da equipe principal também podem ser úteis e podem estar mais disponíveis. Porém, como saber sobre esses em um projeto com milhares e milhares de desenvolvedores? Nesse contexto, enquanto um pequeno grupo de desenvolvedores podem ficar sobrecarregado [Avelino et al. 2016, Ferreira et al. 2017] outros podem ficar ficam sem oportunidades de participar de forma efetiva do projeto [Tamburri et al. 2015]. Essas situações podem levar à frustração e encorajar os desenvolvedores a deixar o projeto. Esses problemas estão relacionados a como os desenvolvedores interagem entre si e como esses relacionamentos afetam o projeto [Constantino et al. 2020, Constantino et al. 2021]. Portanto, é importante fomentar os laços entre os desenvolvedores do projetos, apresentar dados importantes para os desenvolvedores e mantenedores do projeto que viabilizem esse engajamento no projeto. Portanto, propomos as estratégias de recomendação de desenvolvedores apoiadas por uma ferramenta visual e interativa para conectar colaboradores com base em um conjunto de arquivos de seus interesses [Constantino and Figueiredo 2023].

Além disso, a ferramenta fornece metadados e vincula diferentes atributos que não podem ser analisados usando apenas a interface do GitHub. Além das recomendações de desenvolvedores, a ferramenta também oferecer suporte aos mantenedores do projeto [Constantino et al. 2023b, Constantino et al. 2024]. Por exemplo, os mantenedores poderão acompanhar a evolução das contribuições dos desenvolvedores e também descobrir novos contribuidores, além de não perder contribuições preciosas, uma vez que eles conseguem acompanhar a data das últimas contribuições dos desenvolvedores. Assim, CoopFinder pode ser para a tomanda de decisão, como por exemplo, encontrar novos desenvolvedores para colaborar e/ou gerenciar um projeto, montar equipes com base em seus interesses comuns. Além disso, CoopFinder também pode ajudar a encontrar novos desenvolvedores ou colaboradores para contribuirem no desenvolvimento, ou em outras atividades, tais como testes ou documentação.

## Referências


Avelino, G., Passos, L., Hora, A., and Valente, M. T. (2016). A novel approach for estimating truck factors. In *Proc. of the 24th International Conference on Program Comprehension (ICPC)*, pages 1–10.

Constantino, K., Belém, F., and Figueiredo, E. (2023a). Dual analysis for helping developers to find collaborators based on co-changed files: An empirical study. *Journal of Software: Practice and Experience (JSPE)*, pages 1–27.

Constantino, K. and Figueiredo, E. (2022). Coopfinder: Finding collaborators based on co–changed files. In *Proc. of the IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, pages 1–3.

Constantino, K. and Figueiredo, E. (2023). Finding collaborations based on co-changed files. In *Anais Estendidos do XVIII Simpósio Brasileiro de Sistemas Colaborativos*, pages 57–66.

Constantino, K., Prates, R., and Figueiredo, E. (2023b). Recommending collaborators based on co–changed files: A controlled experiment. In *Proc. of the 18th Brazilian Symposium on Collaborative Systems*, pages 154–168.

Constantino, K., Prates, R., and Figueiredo, E. (2024). An user evaluation of a collaborator recommender based on co-changed files. *Journal on Interactive Systems*, 15(1):157–169.

Constantino, K., Souza, M., Zhou, S., Figueiredo, E., and Kästner, C. (2021). Perceptions of open-source software developers on collaborations: An interview and survey study. *Journal of Software: Evolution and Process (JSEP)*, 33:e2393.

Constantino, K., Zhou, S., Souza, M., Figueiredo, E., and Kästner, C. (2020). Understanding collaborative software development: An interview study. In *Proc. of the 15th International Conference on Global Software Engineering (ICGSE)*, page 55–65.

de Neira, A. B., Steinmacher, I., and Wiese, I. S. (2018). Characterizing the hyperspecialists in the context of crowdsourcing software development. *Journal of the Brazilian Computer Society (JBCS)*, 24(1):1–16.

Ferreira, M., Valente, M. T., and Ferreira, K. (2017). A comparison of three algorithms for computing truck factors. In *Proc. of the 25th International Conference on Program Comprehension (ICPC)*, pages 207–217.

Franco, M. F., Rodrigues, B., and Stiller, B. (2019). Mentor: The design and evaluation of a protection services recommender system. In *Proc. of the 15th International Conference on Network and Service Management (CNSM)*, pages 1–7.

Gousios, G., Pinzger, M., and Deursen, A. v. (2014). An exploratory study of the pull-based software development model. In *Proc. of the 36th International Conference on Software Engineering (ICSE)*, pages 345–355.

Gousios, G., Zaidman, A., Storey, M.-A., and Deursen, A. v. (2015). Work practices and challenges in pull-based development: The integrator's perspective. In *Proc. of the 37th International Conference on Software Engineering (ICSE)*, volume 1, pages 358–368.

Kononenko, O., Baysal, O., and Godfrey, M. W. (2016). Code review quality: How developers see it. In *Proc. of the 38th International Conference on Software Engineering (ICSE)*, pages 1028–1038.

Minto, S. and Murphy, G. (2007). Recommending emergent teams. In *Proc. of the 4th International Conference on Mining Software Repositories (MSR)*, pages 5–5.

Oliveira, J., Pinheiro, D., and Figueiredo, E. (2020). Jexpert: A tool for library expert identification. In *Proc. of the 34th Brazilian Symposium on Software Engineering (SBES)*, pages 386–392.

Oliveira, J., Viggiato, M., and Figueiredo, E. (2019). How well do you know this library? mining experts from source code analysis. In *Proc. of the XVIII Brazilian Symposium on Software Quality (SBQS)*, pages 49–58.

Pham, R., Singer, L., Liskin, O., Figueira Filho, F., and Schneider, K. (2013). Creating a shared understanding of testing culture on a social coding site. In *Proc. of the 35th International Conference on Software Engineering (ICSE)*, pages 112–121.

Pinto, G., Steinmacher, I., and Gerosa, M. (2016). More common than you think: An in-depth study of casual contributors. In *Proc. of the 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, volume 1, pages 112–123.

Rahman, M. M., Roy, C. K., Redl, J., and Collins, J. A. (2016). Correct: Code reviewer recommendation at github for vendasta technologies. In *Proc. of the 31st International Conference on Automated Software Engineering (ASE)*, page 792–797.

Ricci, F., Rokach, L., and Shapira, B. (2011). Introduction to recommender systems handbook. In *Recommender Systems Handbook*, pages 1–35.

Salton, G. (1971). The smart retrieval system: Experiments in automatic information retrieval.

Salton, G. (1989). Automatic text processing: The transformation, analysis, and retrieval of. *Reading: Addison-Wesley*, 169.

Salton, G. and Harman, D. (2003). Information retrieval. In *Encyclopedia of Computer Science*.

Steinmacher, I., Pinto, G., Wiese, I. S., and Gerosa, M. A. (2018). Almost there: A study on quasi-contributors in open-source software projects. In *Proc. of the 40th International Conference on Software Engineering (ICSE)*, pages 256–266.

Tamburri, D. A., Kruchten, P., Lago, P., and Van Vliet, H. (2015). Social debt in software engineering: Insights from industry. *Journal of Internet Services and Applications (JISA)*, 6(1):1–17.

Yu, Y., Wang, H., Filkov, V., Devanbu, P., and Vasilescu, B. (2015). Wait for it: Determinants of pull request evaluation latency on github. In *Proc. of the 12th International Conference on Mining Software Repositories (MSR)*, pages 367–371.