Universidade Federal do Amazonas
Instituto de Computação
Programa de Pós-Graduação em Informática

Márcia Sampaio Lima

Minerando Conhecimentos de Projetos de *Software* a partir dos Registros de Comunicação de Desenvolvedores

Manaus
Outubro de 2023

Márcia Sampaio Lima

Minerando Conhecimentos de Projetos de *Software* a partir dos Registros de Comunicação de Desenvolvedores

Tese submetida ao Programa de Pós-Graduação em Informática do Instituto de Computação da Universidade Federal do Amazonas como requisito para obtenção do grau de Doutor(a) em Informática.
Área de concentração: Engenharia de Software

Orientador: Prof. Dr. Bruno Freitas Gadelha
Coorientadora: Profa. Dra. Tayana Uchoa Conte
Coorientador: Prof. Dr. Igor Steinmacher

Manaus
2023

Ficha Catalográfica

Ficha catalográfica elaborada automaticamente de acordo com os dados fornecidos pelo(a) autor(a).

Lima, Márcia Sampaio
L732m Minerando conhecimentos de projetos de software a partir dos registros de comunicação de desenvolvedores / Márcia Sampaio Lima . 2023
218 f.: il. color; 31 cm.

Orientador: Bruno Freitas Gadelha
Coorientadora: Tayana Uchoa Conte
Coorientador: Igor Steinmacher
Tese (Doutorado em Informática) - Universidade Federal do Amazonas.

1. Conhecimentos de software. 2. Ferramentas de comunicação. 3. Times de desenvolvimento. 4. Mineração de repositórios de eng. de software. I. Gadelha, Bruno Freitas. II. Universidade Federal do Amazonas III. Título

Ministério da Educação
Universidade Federal do Amazonas
Coordenação do Programa de Pós-Graduação em Informática

FOLHA DE APROVAÇÃO

"MINERANDO CONHECIMENTOS DE PROJETOS DE SOFTWARE A PARTIR DOS REGISTROS DE COMUNICAÇÃO DE DESENVOLVEDORES"

MÁRCIA SAMPAIO LIMA

Tese de Doutorado defendida e aprovada pela banca examinadora constituída pelos Professores:

Prof. Dr. Bruno Freitas Gadelha - PRESIDENTE

Prof. Dr. Alessandro Fabrício Garcia - MEMBRO EXTERNO

Profa. Dra. Bianca Trinkenreich - MEMBRO EXTERNO

Prof. Dr. Leonardo Gresta Paulino Murta - MEMBRO EXTERNO

Prof. Dr. Igor Scaliante Wiese - MEMBRO EXTERNO

Manaus, 10 de Outubro de 2023

Documento assinado eletronicamente por **Igor Scaliante Wiese**, **Usuário Externo**, em 19/10/2023, às 10:38, conforme horário oficial de Manaus, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Alessandro Fabricio Garcia**, **Usuário Externo**, em 19/10/2023, às 10:42, conforme horário oficial de Manaus, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Bianca Trinkenreich**, **Usuário Externo**, em 19/10/2023, às 11:40, conforme horário oficial de Manaus, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Leonardo Gresta Paulino Murta**, **Usuário Externo**, em 19/10/2023, às 21:31, conforme horário oficial de Manaus, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Bruno Freitas Gadelha**, **Professor do Magistério Superior**, em 30/10/2023, às 14:44, conforme horário oficial de Manaus, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site https://sei.ufam.edu.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **1722357** e o código CRC **D1E0A93D**.

Avenida General Rodrigo Octávio, 6200 - Bairro Coroado I Campus Universitário Senador Arthur Virgílio Filho, Setor Norte - Telefone: (92) 3305-1181 / Ramal 1193
CEP 69080-900, Manaus/AM, coordenadorppgi@icomp.ufam.edu.br

Referência: Processo nº 23105.042201/2023-14 SEI nº 1722357

AGRADECIMENTOS

A Deus pela minha vida, saúde, família, amigos e pelas oportunidades e desafios que me foram dados.

Ao meu marido, Leonardo Lima, que me apoia, incentiva, ajuda, respeita, acalenta, diverte e me compreende nos momentos difíceis, nos momentos de desespero e nos momentos felizes. Léo sempre fez questão de lutar, sonhar e comemorar junto comigo e com nossa amada Annabel, pois para ele a família é o seu grande tesouro.

À minha amada filha, Annabel Lima, que me encoraja a ter força, a lutar diariamente pelo meu sonho, que me incentiva a ter percepções diferentes sobre o mundo, que me desafia a cada dia e me ensina o que é o amor incondicional.

Aos meus pais, Leônidas e Silene, que construíram o alicerce sobre o qual caminho. Me ensinaram valores morais e éticos que embasam a minha vida e a relação que tenho com todos ao meu redor. Meu muito obrigado por terem me protegido, cuidado e guiado por todos esses anos da minha vida.

Aos meus irmãos, Marco e Túlio, que sempre torceram por mim.

À minha amiga Jocélia. Esta caminhada sem o seu apoio teria sido muito mais difícil. A você, meus sinceros agradecimentos.

À minha amiga Marcela Pessoa. Por coincidência do destino, traçamos objetivos comuns na vida. Obrigada por seu apoio e sua amizade sincera.

Ao meu orientador e amigo, Professor Bruno Gadelha, que confiou em mim, que me incentiva e acredita em mim. Bruno, você é um amigo e um profissional brilhante. Sou muito feliz por estar ao seu lado nessa caminhada e imensamente grata a você. A combinação de sua experiência acadêmica e seu apoio amigável criou um ambiente no qual me senti confortável para explorar novas ideias, fazer perguntas e buscar soluções criativas.

À minha coorientadora e amiga, Professora Tayana Conte, pela confiança, pela dedicação, pelas oportunidades e pelo carinho a mim oferecidos. Querida Tay, você me inspira a ser melhor, me encoraja a não ter medo e me incentiva a crescer. Sua orientação, paciência e apoio foram inestimáveis para o meu crescimento

acadêmico e pessoal ao longo deste período. Agradeço por investir seu tempo e energia em me ajudar a desenvolver minhas habilidades de pesquisa e a superar os desafios que surgiram ao longo do caminho. Agradeço demais pela amiga Tay!

Ao meu coorientador, Professor Igor Steinmacher, que após a minha qualificação, me coorientou e me propiciou oportunidades excepcionais que contribuíram para o desenvolvimento desta tese. Professor Igor me convidou para participar de reuniões e trabalhos com profissionais do GitHub e pesquisadores da Microsoft *Research*. Muito, muito, muito obrigada!

Aos professores Alessandro Garcia e Igor Steinmacher por aceitarem participar da minha banca de qualificação desta pesquisa, pela leitura atenta do meu texto e por serem referências como pesquisadores e professores. Muito obrigada!

Aos professores Alessandro Garcia, Bianca Trinkenreich, Igor Wiese e Leonardo Murta, por aceitarem participar da minha banca de defesa de doutorado. Expresso meus agradecimentos a cada um de vocês pelo tempo, atenção e valiosas contribuições durante a avaliação da minha tese de doutorado. Agradeço por dedicarem parte do seu tempo para analisar o meu trabalho e por compartilharem seus conhecimentos e *insights* durante a defesa. Suas perguntas, comentários e sugestões enriqueceram a discussão e contribuíram para aprimorar a qualidade da minha pesquisa e da pesquisadora Márcia! Admiro e respeito o amor e o profissionalismo que todos vocês possuem pela área de Engenharia de Software.

Aos membros do grupo USES que sempre me receberam e ajudaram de braços abertos. Obrigada queridos amigos pelas discussões de pesquisas, pelas dúvidas sanadas, pelos sucessos comemorados, pelos trabalhos realizados em conjunto e pelos momentos de descontração. A experiência de fazer parte deste grupo foi incrivelmente enriquecedora e inspiradora. A troca de conhecimentos, as discussões apaixonadas e a dedicação de todos à pesquisa criaram um ambiente estimulante, onde pude crescer como pesquisador e indivíduo.

Um agradecimento especial à Universidade Federal do Amazonas, ao Instituto de Computação e ao programa de pós-graduação em informática da UFAM (PPGI-UFAM) pelas oportunidades proporcionadas.

À Universidade do Estado do Amazonas (UEA), instituição na qual trabalho. Agradeço aos amigos professores que torcem por mim.

Agradeço aos participantes voluntários dos estudos aqui descritos.

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001. Este trabalho foi parcialmente financiado pela Fundação de Amparo à Pesquisa do Estado do Amazonas – FAPEAM – por meio do projeto POS-GRAD e processo número 062.00150/2020. Gostaríamos também de agradecer o apoio financeiro concedido pelo CNPq através dos processos nº 314174/2020-6 e 313067/2020-1. FAPESP sob bolsa 2020/05191-2. Esta pesquisa também foi realizada no âmbito do Projeto Samsung-UFAM de Ensino e Pesquisa (SUPER). Esta pesquisa, conforme previsto no Art. 48 do decreto n. 6.008/2006, foi parcialmente financiada pela Samsung Eletrônica da Amazônia Ltda., nos termos da Lei Federal n. 8.387/1991, através do convênio n 003/2019, firmado com o ICOMP/UFAM.

Resumo

Equipes de desenvolvimento de software recorrem a diferentes canais de comunicação para dar suporte às tarefas de desenvolvimento e gerenciamento de projetos. Dentre tais canais, pode-se citar as ferramentas de mensagens instantâneas (*chats*) e fóruns de discussões. Contudo, quando equipes usam tais canais, discussões relevantes relacionadas ao software são registradas nos arquivos de *logs* destes recursos, podendo tornar-se "perdidas", não implementadas, esquecidas, duplicadas ou difíceis de serem encontradas em meio ao excesso de mensagens trocadas. Em todos os cenários destacados, a perda e a duplicação de informações podem comprometer o compartilhamento e o reuso de conhecimentos dos projetos de software. Motivada pelo contexto mencionado e pela hipótese de que se pode determinar conhecimentos relevantes de software a partir dos registros de discussões de desenvolvedores, a seguinte questão de pesquisa (QP) guia este trabalho: **QP: Como utilizar métodos automáticos para identificar discussões relevantes de projetos de software a partir dos registros de comunicação de desenvolvedores?** Objetivando responder à questão norteadora desta pesquisa e com base na metodologia *Design Science Research*, foi desenvolvido o *framework Miner4DevTeam*. O *framework* apoia a determinação de conhecimentos de projetos de software a partir dos registros de comunicação de desenvolvedores, contribuindo para a realização das tarefas de desenvolvimento. Tais conhecimentos revelam o contexto de desenvolvimento dos projetos. Os resultados aqui obtidos mostram que os conhecimentos determinados podem apoiar a evolução do produto, auxiliar o processo de tomada de decisões estratégicas das empresas e promover o compartilhamento e a reutilização do conhecimento dos projetos. Trabalhos futuros oferecem oportunidades para o aprimoramento do *Miner4DevTeam*, para a utilização de diferentes repositórios de dados e para desenvolvimento de estratégias que apoiem a área de gestão de conhecimento de projetos de software.

Palavras-chave: Conhecimentos de *Software*, Ferramentas de Comunicação, Times de Desenvolvimento, Mineração de repositórios de ES.

Abstract

Software development teams use collaborative communication channels, such as instant messaging (IM) tools and forums, to support software development and management tasks. When teams use such channels, relevant software-related discussions stay in these tools' log files. However, such relevant software-related discussions and decisions can get "lost," unimplemented, forgotten, duplicated, or hard to find amidst the large volume of messages. Lost and duplicated information can compromise the sharing and reuse of project knowledge. Motivated by the mentioned context and the hypothesis that one can extract relevant software-related knowledge from the developer's communication logs, this work aims to answer the following research question (RQ): **RQ: How can we use automatic approaches to identify relevant software-related discussions from the developers' communication log files?** Based on the Design Science Research methodology, we developed the framework *Miner4DevTeam* to answer this question. The framework helps stakeholders determine software project knowledge from team communication log files and supports the development tasks. Such knowledge can reveal the project development context. Our findings show that the determined knowledge can support product evolution, aid companies' strategic decision-making process, and support project knowledge sharing and reuse. Future work brings opportunities to enhance the framework, explore different data repositories, and support the knowledge management field.

Keywords: Software Knowledge, Communication Tools, Software Development Teams, Mining SE repositories.

Lista de Figuras

Lista de Tabelas

Sumário

Capítulo 1

Introdução

Este capítulo apresenta a área de interesse da pesquisa de doutorado aqui descrita. A Seção 1.1 destaca o contexto e a motivação do trabalho desenvolvido. As Seções 1.2 e 1.3 apresentam a questão de pesquisa a ser respondida e os objetivos a serem alcançados, respectivamente. A metodologia de pesquisa adotada é exposta na Seção 1.4. Por fim, na Seção 1.5 é apresentada a estrutura de organização deste texto.

1.1 Contexto

No contexto atual do desenvolvimento de software, as equipes adotam diferentes canais de comunicação para apoiar o modelo de desenvolvimento colaborativo dos projetos (Storey et al., 2014, 2016; Tantisuwankul et al., 2019). Estes canais são usados para promover o aprendizado, sanar dúvidas, obter e dar *feedbacks*, mostrar resultados, monitorar e coordenar atividades (Storey et al., 2016). Dentre tais canais destacam-se aqueles que viabilizam as comunicações sociais colaborativas, como, por exemplo, *e-mail*, *chat* e fóruns (Storey et al., 2014, 2016; Pérez-Soler et al., 2018).

Por se tratar de uma tarefa colaborativa, as atividades de desenvolvimento de software exigem constantes discussões e consenso entre os *stakeholders* envolvidos

nos projetos (Pérez-Soler et al., 2018). Tanto as equipes distribuídas quanto as colocalizadas (situadas em um mesmo local) utilizam ferramentas síncronas (*chats*) ou assíncronas (*e-mails*, fóruns, ferramentas de gerenciamento de tarefas, etc.) visando intermediar a colaboração durante o processo de desenvolvimento do software. Diante desse contexto, é natural que a interação das equipes demande a troca de um grande fluxo de informações entre seus membros. Alkadhi et al. (2018) relatam que as mensagens de *chats* substituíram os *e-mails* em muitas equipes, o que vem sendo evidenciado em diversas pesquisas que analisam dados de ferramentas como o Slack[1], Telegram[2] e WhatsApp[3] (Storey et al., 2014, 2016; Pérez-Soler et al., 2018).

Em comunidades de software livre (*open source*) também observa-se uma mudança de comportamento com respeito a utilização das ferramentas de comunicação. Historicamente, as listas de discussão (*e-mails*) eram o canal preferido para comunicação de tais comunidades (Rigby and Hassan, 2007; Guzzi et al., 2013; Vasilescu et al., 2014). No entanto, com o surgimento de outros meios de comunicação, observou-se uma mudança de preferência (Guzzi et al., 2013; Vasilescu et al., 2014). Nos últimos anos, fóruns do tipo Q&A, voltados para desenvolvedores de software (*Programming Community-based Question Answering (PCQA)*) têm atraído, cada vez mais, a atenção dos membros das equipes de software (Wang et al., 2020; Pei et al., 2021). Desenvolvedores recorrem a tais ambientes a fim de sanarem, de forma rápida, suas dúvidas técnicas (Zhang et al., 2017), impactando diretamente no processo de desenvolvimento de software (Yazdaninia et al., 2021). Em fóruns do tipo PCQA a comunicação extrapola os limites das equipes. Nesses ambientes diversos usuários podem compartilhar conhecimentos técnicos e discutir tópicos de interesse de uma comunidade. Como exemplos de fóruns do tipo PCQA, pode-se citar o Stack Overflow[4].

Tanto os *chats* como os fóruns de comunicação de desenvolvedores constituem valiosas fontes de informação acerca dos projetos de software, pois podem conter registros de discussões relacionadas ao processo de desenvolvimento dos projetos de software. Contudo, pesquisas destacam que muitas decisões discutidas pelos

[1]https://slack.com/
[2]https://telegram.org/
[3]https://www.whatsapp.com/
[4]https://stackoverflow.com/

desenvolvedores são raramente transcritas para a documentação oficial do projeto e que, tampouco, constituem estímulo para a atualização de tais documentos, tornando-se perdidas do ponto de vista da equipe de desenvolvimento (Burge and Brown, 2008; Manteuffel et al., 2014; Soria and van der Hoek, 2019). Contudo, essas informações são essenciais para a compreensão do escopo de desenvolvimento de projetos de software, sendo úteis para apoiar as tarefas de desenvolvimento e embasar as tomadas de decisões dos times (Alkadhi et al., 2017, 2018), ignorá-las pode comprometer o compartilhamento e a reutilização do conhecimento dos projetos. Este problema motivou o desenvolvimento desta pesquisa de doutorado.

Diante de tal cenário, o objeto de interesse desta pesquisa compreende os registros gerados pela comunicação de desenvolvedores de software. Neles podem ser explicitados conhecimentos e experiências úteis para uma melhor compreensão e gestão dos projetos de software. Nesta pesquisa, os objetos de interesse considerados foram (1) os registros de comunicação de equipes de projetos de software armazenados nos *logs* das ferramentas de comunicação síncronas e (2) as postagens de discussão de desenvolvedores de software feitos por meio de fóruns do tipo PCQA. Pressupõe-se que a mineração de tais registros revela aspectos técnicos e gerenciais relacionados ao desenvolvimento de projetos de software relevantes para apoiar a realização das atividades de desenvolvedores e gerentes de projetos de software.

Sendo assim, constituem alvo de estudo desta pesquisa de doutorado o planejamento, a criação e a avaliação de mecanismos capazes de recuperar discussões e decisões de projetos relevantes a partir dos registros de comunicação de desenvolvedores. Tais informações são relevantes para que os *stakeholders* possam melhor gerenciar e desenvolver projetos de software, apoiar a determinação de conhecimentos de projetos de software (*software project knowledge*) e promover o compartilhamento e reuso de tais conhecimentos.

O tema desta pesquisa se enquadra na interseção das seguintes áreas:

- Gestão de Conhecimento de Projetos Software: pois foram desenvolvidos mecanismos capazes apoiar a recuperação e explicitação de conhecimentos de projeto de software armazenados nos registros de comunicação de desenvolvedores, propiciando chances para a organização, armazenamento,

compartilhamento e reutilização destes. E ainda, oportunizando possibilidades de disponibilizá-los para todos os *stakeholders* e para as futuras iterações do projeto, visando melhorar a produtividade, a qualidade do software e a inovação.

- Trabalho Colaborativo e Computação Social suportados por Computador: pois são estudados como as interações *online* de desenvolvedores, feitas em diferentes canais sociais, podem ser usadas para apoiar o desenvolvimento colaborativo de projetos de software e promover o compartilhamento e reuso de conhecimentos de software.

- Engenharia de Software: pois como resultado, objetiva-se prover alicerces para o desenvolvimento e gerenciamento de projetos de software, com foco na comunicação e colaboração de times de desenvolvimento e, consequentemente, contribuir para o desenvolvimento de projetos de qualidade. Esta pesquisa tem o propósito de enriquecer a capacidade de resgate de conhecimentos em projetos de software, viabilizando, dentre outras oportunidades, a reconstrução do histórico de decisões e raciocínios (*rationales*) empregados ao longo do ciclo de vida dos projetos.

Por fim, destaca-se que esta pesquisa foi submetida e recebeu aprovação do Comitê de Ética da Universidade Federal do Amazonas, CAAE:57361622.5.0000.5020, parecer número 5.350.142, refletindo o compromisso em conduzir toda a pesquisa conforme os padrões éticos necessários. Reconhecemos a importância de garantir a integridade e o bem-estar dos participantes envolvidos nos estudos. Todas as diretrizes e regulamentos estabelecidos pelo comitê foram seguidos, assegurando que a coleta, análise e uso dos dados foram realizados de forma ética e responsável.

1.2 Questão de Pesquisa

Motivada pelo contexto acima destacado, esta tese visa responder a seguinte questão de pesquisa (QP):

QP: Como utilizar métodos automáticos para identificar discussões relevantes de projetos de software a partir dos registros de comunicação de desenvolvedores?

No contexto desta pesquisa, discussões relevantes são aquelas que viabilizam a construção e a disseminação de conhecimentos de projetos de software entre integrantes de equipes de software. Ou seja, discussões que revelam informações acerca da análise, do planejamento, desenvolvimento, utilização, validação, lições aprendidas e gerenciamento do software (Levy and Hazzan, 2009). Tais discussões podem conter raciocínios que embasam as tomadas de decisões feitas pelas equipes, sendo essenciais para apoiar os membros das equipes na realização de suas atividades.

No transcorrer deste texto, os conhecimentos de projetos de software foram subdivididos em três categorias: "Conhecimentos Perdidos de Projetos", "Conhecimentos Frequentes de Projetos" e "Conhecimentos Relacionados de Projetos". Conhecimentos perdidos de projetos, aqui chamados de *Project Lost Knowledge* (PLK), compreendem as decisões e discussões de projetos feitas pelas equipes e que não foram formalmente registradas na documentação oficial do projeto ou que, simplesmente, caíram no esquecimento dos membros das equipes. Conhecimentos frequentes de projetos, aqui chamados de *Project Frequent Knowledge* (PFK), compreendem as decisões e discussões sobre o desenvolvimento e gerenciamento do projeto que já foram bem debatidas pela equipe. Supõe-se que tais decisões já se tornaram claras e intrínsecas ao time de desenvolvimento, porém podem não ter sido formalmente registradas na documentação oficial do projeto. O PFK é útil para apoiar a familiarização de novos membros às equipes. Por fim, "Conhecimentos relacionados de projetos", aqui chamados de *Project Related Knowledge* (PRK), compreendem as decisões e discussões feitas repetidas vezes, sendo estas duplicadas ou quase duplicadas (conforme definição apresentada na Seção 3.1), possivelmente por não serem encontradas em meio às discussões anteriormente realizadas.

1.3 Objetivos

O objetivo desta tese é **definir mecanismos para apoiar a mineração de conhecimentos de projetos de software a partir dos registros não estruturados de comunicação de desenvolvedores, gerados através do uso de ferramentas de comunicação síncronas (*chats*) e assíncronas (fórum do tipo PCQA).** O conhecimento minerado será usado para dar suporte ao compartilhamento e reuso do conhecimento dos projetos de software, revelar o escopo de desenvolvimento dos projetos e apoiar as tarefas de desenvolvimento e gerenciamento de equipes colocalizadas e distribuídas de software.

Objetivos específicos:

- Categorizar os diferentes tópicos de discussões registrados nos canais de comunicação das equipes de desenvolvimento.
- Definir estratégias para minerar decisões e discussões de projeto relevantes para apoiar as atividades de times de desenvolvimento, a partir dos registros de comunicação das equipes.
- Determinar e explicitar conhecimentos de projetos de software registrados nos canais de comunicação colaborativos utilizados por times de desenvolvimento: o *Project Lost Knowledge*, *Project Frequent Knowledge* e o *Project Related Knowledge*.
- Validar, junto à indústria de software, a viabilidade de utilização das estratégias de mineração de conhecimentos de projetos de software implementadas.

1.4 Metodologia

A metodologia de pesquisa usada no desenvolvimento desta pesquisa de doutorado é a *Design Science Research* (DSR). A DSR foca em produzir artefatos inovadores para problemas práticos através da investigação de uma ou mais instâncias de soluções para o problema (Storey et al., 2017). A utilização da metodologia apoia a aceitação e a credibilidade dos artefatos (soluções) criados.

De acordo com Hevner (2007), a DSR pode ser aplicada em um determinado contexto por meio de um processo iterativo composto por três ciclos interligados: ciclo de relevância, o ciclo de *design* e o ciclo de rigor. No ciclo de relevância é definido o domínio de aplicação da pesquisa, são explicitados o problema a ser tratado, os objetivos a serem alcançados e os critérios de aceitação para a avaliação final dos resultados da pesquisa. No ciclo de *design*, a solução para o problema descrito no ciclo de relevância é desenvolvida, refinada e avaliada. Considerado por Hevner (2007) o ciclo núcleo de qualquer pesquisa baseada em DSR, o ciclo de *design* itera mais rapidamente entre as atividades de construção, refinamento e avaliação dos artefatos que compõem a solução do problema. O ciclo de rigor gera a base de conhecimento que provê a fundamentação teórica e metodológica para o rigor científico adotado na definição, planejamento, construção e avaliação dos artefatos desenvolvidos no ciclo de *design*. A Figura 1.1, adaptação feita a partir de Hevner (2007), ilustra as etapas executadas na condução desta pesquisa.

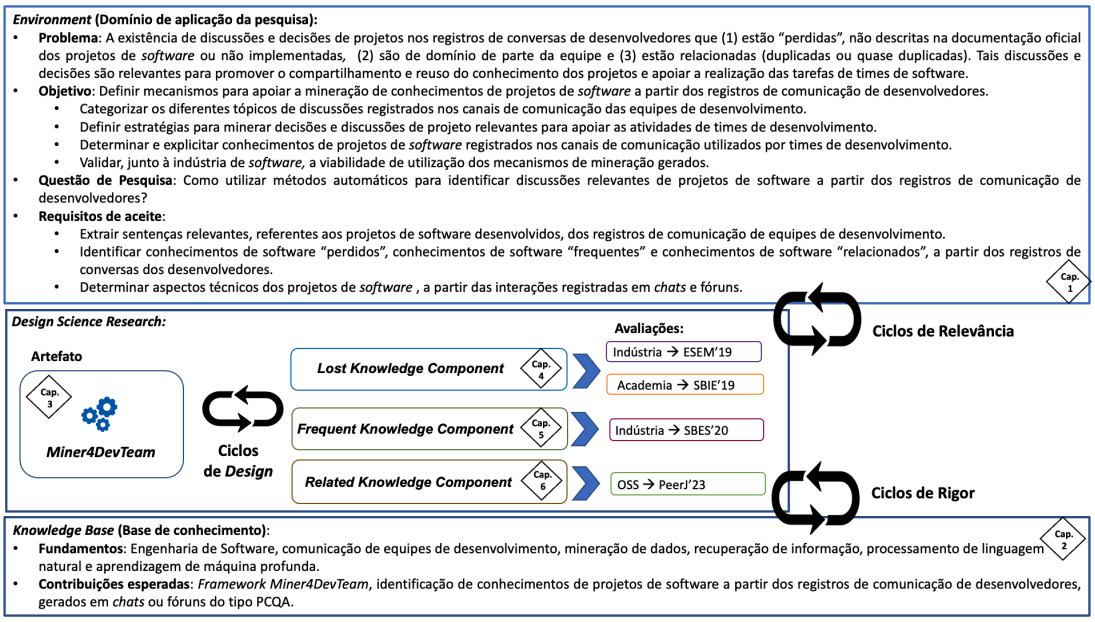

Figura 1.1: Metodologia de desenvolvimento da pesquisa.

A definição do domínio de aplicação desta pesquisa ocorreu mediante revisões da literatura feitas para delimitar o problema a ser tratado (Seções 1.1 e 1.2), os objetivos a serem alcançados (Seção 1.3) e os critérios de aceite da solução

proposta para o problema identificado (Figura 1.1).

Na etapa de *design*, ciclos de iteração foram executados visando desenvolver e avaliar o *Miner4DevTeam*, *framework* apresentado no Capítulo 3. A principal motivação em desenvolver o *Miner4DevTeam* é apoiar as equipes de desenvolvimento de software que utilizam *chats* e fóruns como ferramentas de comunicação para estabelecer a interação entre os membros das equipes de desenvolvimento, viabilizando a mineração dos conhecimentos relacionados aos projetos de software desenvolvidos.

O *framework* baseia-se em técnicas e algoritmos de Recuperação de Informação (RI), Processamento de Linguagem Natural (PLN) e *Deep Learning* (DL), para minerar discussões relevantes de projeto a partir de registros não estruturados de comunicação de desenvolvedores. Tais discussões são relevantes para apoiar o compartilhamento e reuso de conhecimento dos projetos e dar suporte à execução e ao gerenciamento das atividades de equipes de desenvolvimento de software. Em seguida, foram executados estudos que avaliam a relevância das discussões identificadas na determinação de conhecimentos dos projetos de software.

O *Miner4DevTeam* é formado por três componentes que possuem o objetivo de atender os requisitos de aceite delimitados no ciclo de relevância: o *Lost Knowledge Component* (*LK-Component*), o *Frequent Knowledge Component* (*FK-Component*) e o *Related Knowledge Component* (*RK-Component*). Os componentes visam dar suporte a determinação do *Project Lost Knowledge* (PLK), do *Project Frequent Knowledge* (PFK) e do *Project Related Knowledge* (PRK), respectivamente. A determinação do PLK e do PFK foi feita a partir de *logs* de ferramentas de mensagens instantâneas. Já a determinação do PRK foi feita a partir dos registros de discussões do fórum GitHub *Discussions*, o qual é um fórum de comunicação voltado para comunidades hospedadas na plataforma GitHub. A avaliação de adequação dos componentes foi feita com equipes atuantes na indústria de software e em comunidades de projetos *open source*. Para tanto, discussões de diferentes equipes de desenvolvimento foram submetidas ao *framework* e os resultados gerados foram avaliados primeiramente em testes iniciais realizados em laboratório, para que, em seguida, pudessem ser avaliados em estudos realizados na indústria. Também foram realizadas entrevistas com integrantes de equipes de desenvolvimento e mantenedores de projetos *open source* com o objetivo de

capturar e analisar o ponto de vista de diferentes perfis de profissionais em relação aos resultados revelados pelo *framework*. Métricas adequadas à avaliação dos algoritmos implementados foram utilizadas para prover medidas que quantifiquem a eficácia destes.

A base de conhecimento gerada deu-se com uma revisão da literatura a fim de investigar as formas de comunicação atualmente adotadas por equipes de desenvolvimento, determinar as características referentes aos repositórios de dados gerados durante o processo de criação de um software, identificar como a área de mineração de dados vem sendo usada no desenvolvimento de soluções que apoiam as atividades de desenvolvimento de software e determinar métodos, técnicas e algoritmos da área de recuperação de informação, PLN e DL que possam ser usados na construção do *Miner4DevTeam*. A base de conhecimento gerada foi atualizada a cada ciclo de *design* e de relevância executados, provendo fundamentações teóricas para a solução do problema levantado. O Capítulo 2 descreve a base de conhecimento gerada nessa tese.

Nesta tese, em paralelo à realização dos ciclos de relevância e de rigor, foram realizados três ciclos de *design*. No primeiro ciclo de *design* foi executado o planejamento, desenvolvimento e avaliação do *Lost Knowledge Component*. As avaliações foram feitas em dois contextos distintos: na indústria de software (Seção 4.2) e na academia (Seção 4.3). A fim de atingir os objetivos destacados pelo ciclo de relevância foi realizado o segundo ciclo de *design*. Como resultado foi planejado, desenvolvido e avaliado o *Frequent Knowledge Component*, descrito no Capítulo 5. No terceiro e último ciclo de *design* foi desenvolvido o *Related Knowledge Component* que foi avaliado no contexto de projetos *open source*, conforme descrição apresentada no Capítulo 6. Destaca-se que a execução dos ciclos de *design* gerou a necessidade de execução de ciclos de rigor e ciclos de relevância.

1.5 Organização

Além deste capítulo inicial, que visa contextualizar a área de atuação desta pesquisa, esta tese contém outros seis capítulos. A seguir é apresentada a estrutura de organização deste texto:

- **Capítulo 2 -** Apresenta o referencial teórico destacando as formas de comunicação atualmente adotadas por equipes de desenvolvimento, os repositórios de dados gerados durante o processo de criação de um software, os conceitos, técnicas e algoritmos usados para propor e desenvolver o *framework Miner4DevTeam* e os trabalhos relacionados ao contexto desta tese.
- **Capítulo 3 -** Apresenta o *framework Miner4DevTeam*, artefato projetado para minerar discussões relevantes relacionadas a projetos de software, a partir dos registros de comunicação de desenvolvedores, viabilizando o compartilhamento e o reuso de conhecimentos sobre tais projetos.
- **Capítulo 4 -** Apresenta o planejamento, desenvolvimento e avaliação do *Lost Knowledge Component*, componente resultante da execução do primeiro ciclo de *design* da metodologia DSR aqui adotada.
- **Capítulo 5 -** Apresenta a execução do segundo ciclo de *design*. Nele são discutidos o planejamento, o desenvolvimento e a avaliação do *Frequent Knowledge Component*.
- **Capítulo 6 -** Apresenta a execução do terceiro ciclo de *design*. Nele são discutidos o planejamento, o desenvolvimento e a avaliação do *Related Knowledge Component*.
- **Capítulo 7 -** Apresenta a conclusão desta pesquisa, onde é respondida a questão de pesquisa norteadora deste trabalho, são apresentadas as direções futuras oportunizadas e o *Visual Abstract* (Storey et al., 2017) desta tese.

Capítulo 2

Referencial Teórico

Este capítulo apresenta a base de conhecimento usada como fundamentação teórica para o desenvolvimento desta pesquisa. A Seção 2.1 apresenta as formas de comunicação atualmente adotadas por equipes de desenvolvimento. A Seção 2.2 discorre sobre repositórios de dados gerados durante o processo de criação de um software. Na Seção 2.3 são descritos conceitos, técnicas e algoritmos usados para propor e desenvolver o *framework Miner4DevTeam*. Por fim, na Seção 2.4 são apresentados os trabalhos relacionados ao contexto desta pesquisa.

2.1 Comunicação em Times de Desenvolvimento

As equipes de desenvolvimento adotam várias ferramentas e canais de comunicação durante a execução de suas tarefas (Storey et al., 2016). Tanto as equipes de software distribuídas quanto as colocalizadas fazem uso desses recursos para se comunicar, aprender e apoiar o modelo de desenvolvimento colaborativo de software e, ainda, coordenar as tarefas dos membros dos times (Storey et al., 2014). Tais canais abrangem aqueles que viabilizam as formas tradicionais de comunicação, por exemplo, telefone e interações pessoais, e aqueles que viabilizam as comunicações sociais colaborativas, por exemplo, *e-mail*, *chat* e fóruns. No entanto, estudos mostram que as equipes de desenvolvimento estão usando cada vez mais as mídias sociais para fins de comunicação em vez de usarem os canais de comunicação tradicionais (Storey et al., 2014, 2016; Pérez-Soler et al., 2018).

Tais mídias mudaram o panorama da Engenharia de Software, alterando a forma como os desenvolvedores aprendem e trabalham juntos. Os meios de comunicação, como telefone, *e-mail*, *chat*, ferramentas de rastreamento de erros e *sites* de hospedagem de código, desempenham um papel central nas atividades colaborativas de desenvolvimento de software. No entanto, as mídias sociais proveem diferentes recursos para dar suporte à colaboração e disseminação de conhecimentos (Storey et al., 2014). Essas ferramentas e canais tornaram-se uma extensão das equipes, pois fornecem dados que as auxiliam a (1) manterem-se atualizadas sobre tecnologias, práticas e ferramentas para desenvolvimento de software; (2) tirar dúvidas; (3) aprender; (4 e 5) identificar e conectar-se com outros desenvolvedores; (6) obter e dar *feedback*; (7) publicar atividades de desenvolvimento; (8) monitorar as atividades da equipe; (9) mostrar resultados e habilidades; (10) avaliar; e, (11) coordenar atividades (Storey et al., 2016). Diante destas oportunidades, muitos times de desenvolvimento vêm utilizando ferramentas *on-line* de comunicação síncrona, como Slack, Telegram e WhatsApp, e ferramentas assíncronas, como os fóruns do tipo Q&A voltados para desenvolvedores (Stack Overflow e GitHub *Discussions*) (Storey et al., 2016; Pérez-Soler et al., 2018; Hata et al., 2022).

Já no contexto específico das equipes de desenvolvimento de software *open source*, cujo ambiente de desenvolvimento se caracteriza por ser altamente distribuídos e geralmente compostos por equipes autodirigidas (Chen et al., 2013; Tantisuwankul et al., 2019), o uso de *e-mails*, *chats* e fóruns é uma prática bem consolidada para conduzir discussões abertas e públicas (Guzzi et al., 2013; Storey et al., 2016). Historicamente, as listas de discussão eram o canal preferido para comunicação de tais projetos (Rigby and Hassan, 2007; Guzzi et al., 2013; Vasilescu et al., 2014). No entanto, com o surgimento de outros meios de comunicação social, como os fóruns de discussões, observou-se uma mudança de preferência (Guzzi et al., 2013; Vasilescu et al., 2014). Na plataforma GitHub, por exemplo, repositórios de *Issues* e *Pull Requests* (PRs) desempenham papéis relevantes na comunicação das comunidades (Brisson et al., 2020). Esses canais são considerados uma rica fonte de informações sobre o desenvolvimento dos projetos (Guzzi et al., 2013; Storey et al., 2016; Brisson et al., 2020).

Storey et al. (2016) destacam que, quando nos remetemos às ferramentas de desenvolvimento de software, há uma tendência em pensar em ambientes para

desenvolvimento de códigos, depuradores de código, códigos-fonte, sistemas de controle de versão e rastreadores de erros. Contudo, os autores referenciam a pesquisa de Naur (1985) para destacar que o desenvolvimento de software extrapola as barreiras dos códigos que estão sendo feitos e que o desenvolvimento de software também envolve os conhecimentos registrados nas mentes dos desenvolvedores (conhecimento tácito) e nas documentações (conhecimento explícito) que acompanham o projeto. Assim, ferramentas de gerenciamento de projetos e canais de comunicação, como listas de discussões, serviços de *microblogs*, ferramentas de bate-papo (*chats*) e fóruns são exemplos de ferramentas passíveis de registrarem informações (conhecimentos) sobre o software desenvolvido. Levy and Hazzan (2009) destacam que o conhecimento é o principal fator competitivo de uma organização, permitindo que a empresa seja produtiva e que entregue produtos competitivos.

Neste cenário, a área de gestão do conhecimento desempenha um papel importante, objetivando a identificação, organização e compartilhamento do conhecimento acumulado durante o processo de desenvolvimento de projetos de software (Komi-Sirvio et al., 2002). A gestão do conhecimento em equipes de desenvolvimento de software apoia (1) aceleração de aprendizado e familiarização de novos membros, (2) melhoria contínua das atividades de desenvolvimento e manutenção, (3) estabelecimento de boas práticas, (4) colaboração entre membros dos times, (5) resolução de problemas, (6) inovação, (7) documentação, (8) adaptação a mudanças e (9) construção de um base de conhecimento institucional (Natali and de Almeida Falbo, 2002; Levy and Hazzan, 2009; De Vasconcelos et al., 2017).

Esta pesquisa contribui para a crescente literatura sobre como os desenvolvedores de software adotam meios de comunicação social para apoiar o desenvolvimento colaborativo de software e coordenar as tarefas dos membros da equipe e, ainda, como é possível apoiar a determinação de conhecimentos de projetos de software a partir de tais meios. Embora as mensagens instantâneas e os fóruns do tipo PCQA não sejam os únicos meios de comunicação adotados pelas equipes, o foco desta pesquisa é apoiar a determinação de conhecimentos de projetos de software a partir dos registros de comunicação dos times presentes nestes canais. Com isso, esta pesquisa visa fornecer meios para a determinação do *Project Lost Knowledge*, *Project Frequent Knowledge* e *Project Related Knowledge*.

2.2 Repositórios de Engenharia Software

Devido à natureza colaborativa que permeia o processo de desenvolvimento de projetos de software, este produz diferentes fontes de dados resultantes da interação diária dos membros das equipes e das mudanças corretivas e evolutivas nos artefatos de software produzidos (Chen et al., 2016).

As fontes de dados geradas são classificadas por Chen et al. (2016) como fontes estruturadas e fontes não estruturadas. As fontes estruturadas têm um formato estrutural conhecido, como, por exemplo, as árvores de análise de códigos-fonte (*source code parse trees*), os *logs* de execução e rastreios (*execution logs and traces*), os metadados de lista de discussões (*mailing list metadata*) e os metadados dos *logs* de bate-papo. Já as fontes não estruturadas são formadas principalmente por textos em linguagem natural, como as descrições de relatórios de erros (*bug reports*), comentários adicionados aos códigos-fonte, documentação de requisitos, conteúdo de listas de discussão e conteúdo das mensagens de bate-papo. Pesquisas investigam a utilidade de fontes de dados estruturadas e não estruturadas geradas durante o processo de desenvolvimento de software (Rastkar et al., 2014; Francois et al., 2015; Alkadhi et al., 2018; Viviani et al., 2018).

No entanto, Bavota (2016) destaca que o crescimento da geração de dados não estruturados cria repositórios de dados de software que devem ser minerados. Em sua pesquisa, o autor descreve a aplicação de *Mining Unstructured Data* (MUD) em diversos contextos da Engenharia de Software (ES). O autor elenca técnicas utilizadas para minerar dados a partir de fontes não estruturadas, mais especificamente, são destacados repositórios de dados não estruturados de ES disponíveis para mineração e possíveis aplicações do MUD na área de ES. Alkadhi et al. (2017) investigaram formas de extrair *design rationale* das mensagens de bate-papo dos desenvolvedores. Viviani et al. (2018) pesquisam formas de extrair automaticamente informações de *design* das discussões registradas em *pull request*. Francois et al. (2015) criaram um sistema para recuperar *design rationale* a partir de *e-mails* corporativos. Além disso, diferentes pesquisas mineram repositórios de ES para tratar o problema da duplicação de informação em sistemas de rastreamento de *bugs* (Alipour et al., 2013; Kukkar et al., 2020; Cooper et al., 2021), em fóruns de perguntas e respostas(Silva et al., 2018; Wang et al., 2020; Pei et al.,

2021) e em dados gerados em plataformas de desenvolvimento colaborativo como GitHub (Yu et al., 2018; Wang et al., 2019; Lima and Soares, 2022).

Conforme Seção 2.1, estudos mostram que as equipes estão cada vez mais usando ferramentas de colaboração para fins de comunicação. Tais ferramentas apoiam a troca de mensagens que podem compreender perguntas e esclarecimentos rápidos, alinhamento de atividades de coordenação de equipes e, ainda, atividades sociais. No entanto, o principal objetivo destas mensagens é a discussão de questões técnicas sobre o trabalho desenvolvido (Isaacs et al., 2002).

O cenário no qual as empresas de software estão inseridas é comum que estas recorram a diferentes ferramentas de comunicação para facilitar a interação entre os membros da equipe, gerenciar as atividades realizadas e discutir decisões de projeto (Storey et al., 2014; Giardino et al., 2014; Hata et al., 2022), gerando artefatos (repositórios) não estruturados que registram conhecimentos acerca do desenvolvimento dos projetos de software.

Os repositórios de Engenharia de Software desempenham um papel crucial na gestão eficaz do desenvolvimento de software, facilitando o compartilhamento de código, a colaboração em equipe, a detecção e resolução de conflitos, a rastreabilidade de alterações e a manutenção de um histórico detalhado do progresso do projeto. Esta pesquisa de doutorado tem como objetivo minerar conhecimentos de projeto de software a partir de registros não estruturados de comunicação de equipes de desenvolvimento gerados por meio de ferramentas de bate-papo (*chats*) e de fóruns do tipo PCQA.

2.3 Mineração de Dados aplicada a ES

A mineração de dados (*data mining*) é um processo que consiste em descobrir estruturas interessantes, inesperadas ou valiosas em grandes conjuntos de dados (Zaki and Meira, 2014). É uma área interdisciplinar que combina conceitos provenientes da área de estatística e matemática com ferramentas e técnicas oriundas de grandes áreas da computação, como a aprendizagem de máquina, banco de dados e áreas de análise de dados, a fim de manipular dados e explicitar informações. A mineração de dados constitui um processo de descoberta de conhecimento, que engloba tarefas de pré-processamento, como extração e limpeza de dados, e

pós-processamento, como a interpretação dos padrões e modelos minerados (Zaki and Meira, 2014).

Técnicas de mineração de dados vêm sendo aplicadas em ES com objetivo de apoiar a realização de tarefas de desenvolvimento e como instrumento para a realização de estudos empíricos (Bavota, 2016). Algoritmos de mineração de dados estão sendo usados para: (1) gerar documentos, por exemplo, sumarizando conteúdo de *bug reports* (Rastkar et al., 2014); (2) determinar *links* de rastreabilidade entre artefatos de software (Jain et al., 2014); (3) gerar recomendações para desenvolvedores e gerentes para execução de tarefas (Alkhazi et al., 2020); (4) ranquear artefatos textuais (Jindal and Kaur, 2020); (5) avaliar e melhorar a qualidade de desenvolvimento de códigos-fonte (Srinivas et al., 2015); e, (6) realizar predição de falhas (Batool and Khan, 2022) .

Nesta pesquisa são usadas quatro estratégias amplamente aplicadas à mineração de dados textuais não estruturados: Correspondência de Padrões (*Pattern Matching*), Recuperação de Informação (RI), Processamento de Linguagem Natural (PLN) e *Deep Learning* (DL). Algoritmos de *Pattern Matching* são desenvolvidos visando verificar da existência de um padrão específico (uma sequência de *tokens*) em um texto. Esta técnica é geralmente implementada usando expressões regulares que definem a sequência de *tokens* que se deseja identificar em um texto. No geral, algoritmos de RI combinam diversas evidências para identificar documentos relevantes de uma base de dados. As técnicas de PLN permitem, entre outras funcionalidades, extrair automaticamente o significado de partes do texto escrito em linguagem natural. Já as estratégias baseadas em *Deep Learning* visam criar sistemas capazes de aprender a partir dos dados, permitindo que máquinas compreendam, representem e resolvam problemas complexos, contribuindo para avanços na área de processamento de linguagem natural, reconhecimento de padrões e outros.

Na implementação do *framework Miner4DevTeam* foram usados:

- **Algoritmos de Ranque (*Ranking*)**

Os algoritmos de ranque são usados para classificar automaticamente os objetos de acordo com sua relevância. Gambhir and Gupta (2017) relatam que muitos problemas de Recuperação de Informação, como resumo de texto (*Text*

Summarization), são por natureza problemas de ranque. No contexto desta tese, o algoritmo *Text Summarization* é utilizado para identificar mensagens relevantes nos registros de comunicação dos desenvolvedores, formando um ranque ordenado de sentenças relevantes (Seção 3.2).

O algoritmo de sumarização aqui usado determina o valor de relevância das mensagens baseado na importância dos termos que as compõem. Para tanto, é utilizada a métrica $TF \times IDF$, métrica de ponderação de termos adotada na avaliação de sistemas de mineração de dados. A frequência do termo (TF) premia os termos que aparecem em um documento com uma alta frequência, enquanto a frequência inversa do documento (IDF) penaliza os termos que aparecem em muitos documentos, isto é, termos não discriminatórios como artigos e preposições. Assim, um termo é considerado relevante para representar o conteúdo do documento se ocorrer várias vezes no documento e estiver contido em um pequeno número de documentos da coleção analisada (Robertson, 2004). O cálculo de $TF \times IDF$ é feito conforme a Equação 2.1.

$$TF \times IDF(t) = TF(t) \times log\frac{N}{n(t)}, \tag{2.1}$$

onde t é o termo sobre o qual se deseja calcular o valor de $TF \times IDF$, N corresponde ao total de documentos na coleção e $n(t)$ é o número de documentos na coleção que contém o termo t.

- **Algoritmos para geração de *Word Cloud***

Word Cloud (*tag cloud* ou nuvem de palavras) compreende uma estratégia de representação visual para uma coleção de documentos textuais que usa diferentes tamanhos de fonte, cores e espaços para organizar e representar palavras relevantes relacionadas à coleção analisada (Chi et al., 2015). Consequentemente, as *word clouds* podem ser utilizadas como estratégias para sumarizar o conteúdo de uma coleção de textos (Chi et al., 2015). De acordo com Khusro *et al.* (Khusro et al., 2018), as *word clouds* estão sendo usadas para sumarizar conteúdos semânticos em mídias sociais que, geralmente, possuem grandes coleções de textos. Porém, os autores afirmam que além de sumarizar conteúdos textuais, as *word clouds* podem ser usadas como recursos para busca, navegação, formação de opinião e

reconhecimento de correspondência. Como ferramenta de busca, os termos da *word clouds* são usados como *hyperlynk* que referenciam a informação buscada. Como estratégia de navegação, os termos das nuvens são usados como índices, através dos quais usuários podem navegar pelos recursos de seu interesse. Na formação de opiniões, usuários podem observar o conjunto de termos de uma nuvem e formar opiniões acerca da entidade (pessoa, empresa, instituição, assunto) associada a nuvem. No reconhecimento de correspondência, usuários podem usar nuvens para determinar o contexto de algo.

Na área de ES, as *word clouds* vêm sendo utilizadas na manipulação de dados provenientes do processo de desenvolvimento de software, como, por exemplo, na visualização de informações registradas em sistemas de rastreamento de erros (*bug report systems*) apoiando o processo de deduplicação e correção dos erros reportados (Sharma and Sharma, 2015). Nesta tese, *word clouds* são utilizadas como estratégias de visualização para demonstrar o escopo de desenvolvimento de projetos de software, sumarizando informações semânticas e técnicas referente ao projeto desenvolvido (Seção 3.3). Informações semânticos revelam características de especificação, planejamento, desenvolvimento, manutenção e gerenciamento dos projetos de software. Informações técnicas revelam características relacionadas ao uso de tecnologias e decisões técnicas no desenvolvimento dos projetos, como, por exemplo, o sistema operacional sobre o qual o software será executado, *Application Programming Interface* (APIs) utilizadas e detalhes de implementação.

- **Algoritmos de Distância de Similaridade de *Strings***

Algoritmos usados para calcular a distância de edição entre sentenças, usam funções de similaridade de texto para quantificar a similaridade entre duas *strings* e obter uma correspondência ou comparação aproximada das *strings* analisadas (Lu et al., 2013). De acordo com Lu et al. (2013), estratégias para cálculo de similaridade de *strings* são muito usadas em áreas como integração de dados, deduplicação de dados e mineração de dados. Os autores afirmam que a maioria dos trabalhos existentes considera similaridades sintáticas entre as *strings* e usam funções como, distância de *Levenshtein*, distância de *Hamming*, similaridade de *cosseno* e Coeficiente de *Jaccard* para determinar tal distâncias (Lu et al., 2013).

Nesta pesquisa, um algoritmo que calcula a distância de edição foi utilizado na implementação do *FK-Component* para identificar a similaridade entre mensagens registradas nos *logs* de comunicação dos desenvolvedores. O algoritmo utilizado baseia-se na métrica de similaridade de *cosseno*, técnica usada para calcular a distância entre dois pontos (vetores), conforme Equação 2.2 (Tan et al., 2006).

$$cos(\theta) = \frac{AB}{||A||^2||B||^2}, \tag{2.2}$$

onde A e B são vetores numéricos que representam as mensagens enviadas pelas equipes. A fórmula retorna como resultado valores de similaridade entre [0-1], onde um valor próximo a 0 indica ortogonalidade (sem correlação) e um valor próximo a 1 indica similaridade de itens (Tan et al., 2006). O algoritmo implemento utiliza a similaridade de *cosseno* para identificar mensagens que pertençam a categorias pré-definidas, conforme descrito na Seção 3.3.

Como exemplo, cita-se que o algoritmo pode identificar que a frase *"- Bom dia, equipe!"* é semelhante à *"- Bom dia, time!"* e assim determinar possíveis mensagens que expressam saudações.

Além disso, a similaridade de cosseno também foi utilizada no *RK-Component* com objetivo de identificar de postagens de discussões relacionadas ocorridas em fóruns de discussões de desenvolvedores. No *RK-Component*, a similaridade de cosseno é usada para quantificar a semelhança entre *embeddings* de postagens de discussões (Seção 3.4).

- **Algoritmos de *Part-of-speech***

Algoritmos de *Part-of-speech* são usados no processamento de textos escritos em linguagem natural para determinar a classe gramatical das palavras em uma frase, ou seja, são usados para identificar substantivos, verbos, adjetivos, advérbios, pronomes, etc. (Bavota, 2016). Nesta tese, algoritmos de *part-of-speech* que identificam a classe gramatical de palavras escritas em português são utilizados a fim de identificar substantivos relevantes que descrevam o escopo de desenvolvimento de projetos de software.

A estratégia adotada para a identificação de termos relevantes ao escopo de desenvolvimento do software, a partir de registros de bate-papo de desenvolvedores,

é implementada pelo *FK-Component* descrito na Seção 3.3. Os termos relevantes identificados são utilizados na geração de *word clouds* que visam sumarizar o escopo de desenvolvimento de projetos de software.

Como exemplo, seja M_1 uma mensagem enviada por um membro da equipe:

M_1: - *"Dependendo dos pontos ao clicar na opção Troque Agora na próxima janela iria aparecer somente os Prêmios que estaria dentro da pontuação disponível. Então o usuário iria selecionar e finalizar o Resgate. E os prêmios estariam na Guia Meus Prêmios!"*.

Ao ser utilizado algoritmos de definição de *part-of-speech*, pode-se identificar:

Artigos e preposições:
M_1: - *"Dependendo dos pontos ao clicar na opção Troque Agora na próxima janela iria aparecer somente os Prêmios que estaria dentro da pontuação disponível. Então o usuário iria selecionar e finalizar o Resgate. E os prêmios estariam na Guia Meus Prêmios! "*.

Verbos e suas conjugações:
M_1: - *"Dependendo dos pontos ao clicar na opção Troque Agora na próxima janela iria aparecer somente os Prêmios que estaria dentro da pontuação disponível. Então o usuário iria selecionar e finalizar o Resgate. E os prêmios estariam na Guia Meus Prêmios!"*.

Advérbios:
M_1: - *"Dependendo dos pontos ao clicar na opção Troque Agora na próxima janela iria aparecer somente os Prêmios que estaria dentro da pontuação disponível. Então o usuário iria selecionar e finalizar o Resgate. E os prêmios estariam na Guia Meus Prêmios!"*.

Pronomes:
M_1: - *"Dependendo dos pontos ao clicar na opção Troque Agora na próxima janela iria aparecer somente os Prêmios que estaria dentro da pontuação disponí-*

vel. Então o usuário iria selecionar e finalizar o Resgate. E os prêmios estariam na Guia Meus Prêmios!".

Assumindo que os termos categorizados nas classes gramaticais acima não expressam significativo conteúdo semântico (Gambhir and Gupta, 2017), ao serem eliminados da mensagem M_1 obtém-se a seguinte sequência de palavras:

M_1: - *"pontos opção próxima janela Prêmios pontuação disponível. usuário Resgate prêmios Guia Prêmios!"*

As palavras acima podem ser usadas para sumarizar o conteúdo da mensagem M_1. É possível perceber que a palavra "prêmios" é a que mais se repete, possuindo três ocorrências. Logo, tem-se indício de que, nesta mensagem, o assunto frequentemente tratado está relacionado à "prêmios" dentro do escopo de desenvolvimento do software.

- **Modelos genéricos de *Deep Learning***

Modelos genéricos de *Deep Learning* são projetados de forma flexível e parametrizável, permitindo que sejam adaptados e ajustados para diferentes conjuntos de dados e tarefas. Nesta pesquisa, o *RK-Component* usa um modelo *Sentence-BERT* genérico, `all-mpnet-base-v2`[1], para calcular *embeddings* semanticamente significativo das discussões dos desenvolvedores (Seção 3.4). O modelo mapeia frases e parágrafos (*sentences*) para um espaço vetorial denso de 768 dimensões, sendo projetado para codificar parágrafos curtos. Textos com mais de 384 palavras são truncados. No momento da realização desta pesquisa, dentre os modelos treinados pelo *sentence-Bert*, o `all-mpnet-base-v2` fornecia a melhor qualidade na geração das *sentences embeddings* e no desempenho de tarefas de pesquisas semânticas (Reimers, 2021).

O uso de modelos genéricos se aplica ao contexto em que não existe uma base pré-rotulada a partir da qual abordagens automáticas possam ser treinadas ou otimizadas. Modelos genéricos não se restringem a um contexto especí-

[1]https://huggingface.co/sentence-transformers/all-mpnet-base-v2

fico, pois são treinados e otimizados em diferentes bases de dados. Especificamente, o modelo `all-smpnet-base-v2` foi treinado usando como base o modelo `microsoft/mpnet-base` e otimizado usando mais de 1 bilhão de pares de frases coletadas de diferentes bases de dados (*Reddit comments (2015 – 2018), WikiAnswers Duplicate question pairs, Stack Exchange Duplicate questions (titles+bodies)*, etc) (Face, 2021). Modelos específicos são treinados e otimizados de acordo com o contexto para o qual são criados, não garantindo eficácia quando utilizados fora do contexto de treinamento. Além disso, a utilização de modelos genéricos proporciona vantagens como: (1) a não dependência de estruturas computacionais complexas para treino, validação e teste dos modelos, (2) a não necessidade de otimização dos modelos, e (3) a não necessidade de retreino, revalidação e reteste do modelo sempre que o contexto para o qual foi desenvolvido sofrer alterações (Polyzotis et al., 2017; Zhou et al., 2017; Schelter et al., 2018; Lee and Shin, 2020). O dinamismo da área de desenvolvimento de software pode ser considerado fator motivador para o uso de modelos genéricos. Além disso, o uso de modelos públicos traz flexibilidade para as abordagens desenvolvidas. À medida que novos modelos forem gerados e suas respectivas qualidades forem comprovadas, as abordagens podem ser configuradas para utilizá-los.

2.3.1 Métricas de avaliação

Métricas de avaliação determinam valores através dos quais a qualidade dos sistemas pode ser avaliada e comparada. A análise quantitativa da eficácia dos componentes do *framework Miner4DevTeam* será avaliada de acordo com a eficácia obtida pelos algoritmos mineração de dados. Contudo, existem várias métricas usadas para estimar a eficácia dos algoritmos utilizados segundo o contexto de aplicação destes. Dentre as métricas utilizadas nesta tese, destacam-se: precisão, revocação, precisão@k, acurácia e *Mean Reciprocal Rank*(MRR).

A precisão (*Precision*) é uma medida de qualidade que mede o número total de objetos classificados corretamente em relação ao número total de objetos existentes (Kim, 2018). Isto é, mensura a qualidade das soluções propostas através da proporção de objetos classificados corretamente pelos algoritmos. Sua fórmula é dada por:

$$Precision = \frac{TruePositives(tp)}{TruePositives(tp) + FalsePositives(fp)} \tag{2.3}$$

Sendo a *Precision* uma métrica que pode ser utilizada para quantificar a porção de objetos classificados como relevantes, a $Precision@k$ ($P@k$) é uma métrica usada para mensurar a fração de objetos relevantes retornadas nos primeiros k elementos do topo de uma lista ranqueada de objetos (McFee and Lanckriet, 2010). $P@k$ captura a qualidade do ranque para soluções em que apenas os primeiros k resultados são importantes. Sua fórmula é uma adaptação da Equação 2.3, sendo:

$$Precision@k = \frac{TruePositives_k(tp_k)}{TruePositives_k(tp_k) + FalsePositives_k(fp_k)} \tag{2.4}$$

A revocação (*Recall*) é uma medida de completude que mensura a fração das instâncias que foram classificados corretamente (Kim, 2018). Seu valor é obtido pela seguinte fórmula:

$$Recall = \frac{TruePositives(tp)}{TruePositives(tp) + FalseNegatives(fn)} \tag{2.5}$$

A faixa de valores que tanto a precisão quanto a revocação podem assumir é o intervalo [0, 1]. Uma precisão perfeita, igual a 1, significa que cada elemento categorizado pertence realmente ao critério pesquisado. Uma revocação perfeita, de valor 1, significa que todos os elementos que deveriam pertencer a uma determinada categoria foram realmente identificados.

A medida MRR indica a posição de classificação do primeiro objeto relevante em sistemas cujo objetivo é ranquear objetos de acordo com seus respectivos valores de relevância. Para uma lista ranqueada de objetos l, a posição no ranque do primeiro objeto relevante da lista é denotada por $r(l)$, sendo assim o valor de MRR é calculado da seguinte forma:

$$MRR = \frac{1}{r(l)} \tag{2.6}$$

Se em todos as listas de objetos ranqueadas por um sistema, os objetos relevantes estiverem no topo da lista, o valor do MRR será 1 e atingirá seu valor máximo (Liu, 2011).

Medidas baseadas em posição, como precisão@k (*Precision@k*) e *Mean reciprocal rank* (MRR), são usadas para avaliar o desempenho dos modelos de ranque (Liu, 2011). Nesta tese, ambas as métricas são utilizadas para avaliar a eficácia do componente do *Miner4DevTeam* que identifica sentenças relevantes a partir dos *logs* de comunicação dos desenvolvedores.

Por fim, utilizou-se a acurácia (*Accuracy*) a qual é uma medida quantitativa da eficácia de um classificador $\hat{f}$ (Kim, 2018). Dado um conjunto de dados x com n instâncias, a acurácia equivale à proporção de instâncias classificadas corretamente por $\hat{f}$. Na Equação 2.7, $I(y_i = \hat{f}(x_i))$ vale 1 se uma instância x_i foi classificada corretamente por $\hat{f}$; caso contrário, esse termo vale 0.

$$accuracy(\hat{f}) = \frac{1}{n}\sum_{i=1}^{n} I(y_i = \hat{f}(x_i)) \tag{2.7}$$

2.4 Trabalhos Relacionados

Por ser uma atividade colaborativa, o processo de desenvolvimento de software gera diferentes fontes de dados que possuem informações relacionadas ao projeto construído. Diante desse fato, a inteligência artificial vem sendo usada para identificar, extrair e aprender informações úteis desses repositórios e apoiar as tarefas das equipes de desenvolvimento (Alkadhi et al., 2017; Viviani et al., 2018; Alkadhi et al., 2018; Wang et al., 2020; Pei et al., 2021).

Em uma investigação conduzida para apoiar a construção da base de conhecimento e do domínio de aplicação desta pesquisa, verificou-se que abordagens baseadas em aprendizado de máquina (*Machine Learning* - ML) fornecem soluções eficientes aplicadas com sucesso a diferentes domínios da Engenharia de Software, melhorando e automatizando várias atividades de desenvolvimento, como classificação de requisitos, refatoração, previsão de defeitos e estimativa de esforço (Borges et al., 2022). Os resultados mostram os benefícios que aplicação de ML têm propiciado para automatizar e melhorar o *design* de software (por exemplo, descoberta de arquitetura), desenvolvimento (por exemplo, reutilização de código), teste (por exemplo, previsão de defeitos), manutenção (por exemplo, refatoração), gerenciamento (por exemplo, estimativa de esforço) e tarefas de qualidade Borges et al. (2022).

Jindal and Kaur (2020) propuseram o uso de palavras e frases relevantes na criação de uma abordagem que evidencia conhecimentos registrados em *bug reports*. Os autores afirmam que a abordagem proposta gera resumos precisos, permitindo sua utilização em qualquer projeto de software. Lima et al. (2019c) avaliaram o uso de algoritmos de aprendizagem de máquina na classificação automática de requisitos de software em diferentes granularidades: requisitos funcionais e requisitos não funcionais de disponibilidade, segurança, portabilidade, usabilidade, operacional, escalabilidade, desempenho, tolerância a falha, aparência, disponibilidade e manutenibilidade. Os autores destacam a importância da classificação automática de requisitos de software para a engenharia de requisitos. Kleebaum et al. (2018) visam apoiar as tarefas dos desenvolvedores usando técnicas de sumarização de texto para promover a captura e o uso de conhecimentos registrados nas decisões diárias dos desenvolvedores (*decision knowledge*). Os autores argumentam que a identificação e apresentação de tais conhecimentos é útil no contexto da Engenharia de Software Contínua (*Continuous software engineering*), pois apoiam na análise de impacto de alterações, na validação de requisitos, nas manutenções a longo prazo e mantém os desenvolvedores informados sobre as decisões de arquitetura tomadas. Srinivas et al. (2015) usaram algoritmos de mineração de texto para agrupar projetos de software e apoiar a reutilização de componentes de software. O agrupamento é feito baseado em vetores de características que descrevem os componentes analisados. Os autores afirmam que os grupos formados podem ser usados para revelar conhecimentos e comportamentos oculto de projetos de software.

No contexto específico da mineração de repositórios que registram discussões de equipes de desenvolvimento, pesquisadores mineram os canais de comunicação de desenvolvedores de software visando tratar o problema da duplicação de informação em sistemas de rastreamento de *bugs* (Alipour et al., 2013; Kukkar et al., 2020; Cooper et al., 2021), fóruns do tipo PCQA (Silva et al., 2018; Wang et al., 2020; Pei et al., 2021) e em plataformas de desenvolvimento colaborativo como o GitHub (Yu et al., 2018; Wang et al., 2019; Lima and Soares, 2022). Tais pesquisas usam técnicas de RI, algoritmos ML, modelos de aprendizado de máquina profundo (DLM) e até mesmo uma combinação deles para tratar o problema da duplicação de dados em contextos de ES. Wang et al. (2020) utilizaram soluções

de *deep learning* para identificar postagens duplicadas no *Stack Overflow*. Os autores afirmam que estratégias que se limitam a extrair e manipular recursos textuais para detectar automaticamente perguntas duplicadas são abordagens limitadas por perderem informações semânticas acerca do conteúdo analisado. Sendo assim, propuseram três diferentes abordagens (WV-CNN, WV-RNN e WV-LSTM) que se baseiam em *word embeddings*, *Convolutional Neural Networks* (CNN), *Recurrent Neural Networks* (RNN) e *Long Short-Term Memory* (LSTM) para identificar duplicadas. Li et al. (2017) e Ren et al. (2019) propuseram abordagens automáticas baseadas em técnicas tradicionais de RI e PLN para detectar duplicadas em *Pull Requests* (PRs) no GitHub. Já Zhang et al. (2020) propôs o `iLinker`, uma abordagem para detectar *issues* relacionados no GitHub. A abordagem alcançou *recall@5* de 38%-51% e *recall@10* de 45%-61%. Li et al. (2021) apresenta uma abordagem que combina $TF - IDF$ e técnicas de aprendizado profundo para detectar contribuições duplicados. O método produz uma lista de contribuições duplicadas candidatas mais semelhantes à nova contribuição.

Já no contexto de mineração de conhecimento de projetos de software a partir de *e-mails* e ferramentas de bate-papo, Francois et al. (2015) propuseram o sistema KTR (*Knowledge Trace Retrieval*) para recuperar conhecimentos relacionados a soluções de problemas de software (*problem-solving knowledge*) a partir de *e-mails* institucionais. Os autores mapearam o desafio de recuperar tais conhecimentos em um problema de ranque e propuseram o KTR, sistema que determina a relevância de uma mensagem de *e-mail* para um tópico específico através da combinação linear de vetores de características. Alkadhi et al. (2017) reportaram um estudo exploratório que examina a frequência, a completude e a viabilidade de extração automática de *design rationale* a partir de mensagens de bate-papo de equipes de desenvolvedores que faziam parte de um *capstone course*. Os autores afirmam que as mensagens de bate-papo contêm informações sobre questões discutidas, alternativas consideradas e argumentações que embasam as decisões tomadas durante o desenvolvimento do software. Para tanto, os autores usaram algoritmos de aprendizagem de máquina e processamento de linguagem natural (PLN) para analisar o conteúdo 8.700 mensagens. Os mesmos autores analisaram como desenvolvedores de *Open Source Software* (OSS) discutem *design rationales* em canais de IRC (*Internet Relay Chat*) (Alkadhi et al., 2018). Os autores afirmam

que desenvolvedores de OSS (*open source software*) usam canais de IRC para discutir detalhes de desenvolvimento e implementação de projetos de software e, ainda, para trocar conhecimentos e ideias com outros desenvolvedores. Eles obtiveram sucesso ao utilizarem algoritmos de aprendizagem de máquina na identificação e extração automática de elementos de *rationale* a partir de 7.500 mensagens.

Objetivando ajudar desenvolvedores a capturarem, com pouco esforço, *design rationale* a partir de mensagens de bate-papo, Alkadhi et al. (2017) também propuseram um sistema chamado REACT. O sistema se baseia na anotação manual de mensagens de bate-papo que contém *rationale* pelos próprios desenvolvedores. Ao usarem o REACT, os desenvolvedores anotam suas próprias mensagens de bate-papo com e-mojis projetados para identificar cinco elementos que definem um *rationale*. Os desenvolvedores também podem anotar mensagens escritas por outros membros da equipe. Os autores afirmam que o REACT é o primeiro passo para aprimorar a captura de *rationale* das mensagens de bate-papo dos desenvolvedores, que as anotações são fáceis de aprender e simples de aplicar e que o REACT pode ser facilmente integrado às ferramentas de mensagens instantâneas.

Viviani et al. (2018) afirmam que, frequentemente, as informações de projeto de software estão incorporadas nas discussões dos desenvolvedores e muitas tarefas de desenvolvimento poderiam ser facilitadas se tais informações pudessem ser identificadas automaticamente e apresentadas aos desenvolvedores. Os autores propõem o uso de palavras-chave relacionadas ao projeto de software para extrair automaticamente informações de planejamento a partir de discussões registradas em *pull resquest*.

O objetivo desta tese é definir mecanismos para mineração de conhecimentos de projetos de software a partir dos registros não estruturados de comunicação de equipes de desenvolvimento, gerados através do uso de *chats* e fóruns do tipo PCQA.

Na mineração realizada em *chats*, esta pesquisa difere de Francois et al. (2015) pois os registros de dados aqui utilizados são provenientes dos *logs* de conversas das equipes de desenvolvimento. Diferentemente à Alkadhi et al. (2017) que primeiramente analisaram o *log* de comunicação de equipes de desenvolvimento formadas a partir de *capstone courses* e, em seguida, analisaram a comunicação

de equipes *open source* (Alkadhi et al., 2018), os *logs* utilizados na avaliação dos mecanismos de mineração de *chats* aqui propostos são provenientes da interação de equipes de desenvolvimento atuantes na indústria de software comercial. Assim como proposto por Viviani et al. (2018), Kleebaum et al. (2018) e Jindal and Kaur (2020), os mecanismos de mineração de *chats* aqui desenvolvidos usam técnicas de mineração de dados baseadas na extração de palavras e frases relevantes, para determinar conhecimentos de projetos de software.

Na mineração feita em fóruns do tipo PCQA, as abordagens desenvolvidas comumente usam bases de dados pré-rotuladas para treinar ou otimizar formas de detectar duplicadas nos canais de comunicação dos desenvolvedores. O principal desafio desta pesquisa, neste cenário, está relacionado à diversidade de repositórios hospedados no GitHub, que fornecem diferentes contextos de projetos de software. Esses contextos destacam a relevância de projetar um método adequado para replicação em outros repositórios do GitHub. Portanto, para detectar postagens relacionadas em fóruns de discussão do tipo PCQA, a estratégia aqui desenvolvida (1) não depende de um conjunto de dados pré-rotulado; (2) não depende de modelos específicos de domínio; e (3) é baseada em modelos de aprendizagem profunda de uso geral.

Visando apoiar a mineração de diferentes tipos de conhecimentos de software foi planejado, desenvolvido e avaliado o *framework Miner4DevTeam* que será descrito no próximo capítulo desta tese. Os conhecimentos identificados são usados para revelar o escopo de desenvolvimento dos projetos, apoiar as tarefas de desenvolvimento e gerenciamento realizadas por diferentes membros das equipes e dar suporte ao compartilhamento e reuso do conhecimento dos projetos.

Capítulo 3

Framework: Miner4DevTeam

Este capítulo apresenta o *Miner4DevTeam*, um *framework* projetado para identificar discussões relevantes, referentes ao desenvolvimento e ao gerenciamento de projetos de software, em registros de comunicação de desenvolvedores. Discussões relevantes são aquelas que viabilizam a construção e a disseminação de conhecimentos de projetos de software entre integrantes de equipes distribuídas ou colocalizadas. Ou seja, discussões que revelam informações acerca da análise, do planejamento, desenvolvimento, utilização, validação, lições aprendidas e gerenciamento do software (Levy and Hazzan, 2009). Para tal fim, são utilizados técnicas e algoritmos da área de Recuperação de Informação (RI), Processamento de Linguagem Natural (PLN) e *Deep Learning* (DL). Destaca-se que o *Miner4DevTeam* é o artefato produzido como resultado da metodologia DSR adotada nesta pesquisa.

O capítulo está organizado da seguinte forma: a Seção 3.1 apresenta o *Miner4DevTeam*. Nela são discutidas as decisões de projeto tomadas acerca da construção do *framework*. As Seções 3.2, 3.3 e 3.4 descrevem os componentes que formam o *Miner4DevTeam*: o *Lost Knowledge Component*, o *Frequent Knowledge Component* e o *Related Knowledge Component*.

3.1 Minerando Conhecimentos de software

A mineração (identificação e extração) de decisões e discussões relevantes de projetos de software viabiliza a disseminação e o reuso de conhecimentos acerca

dos projetos, apoiando as equipes de desenvolvimento na execução de suas tarefas. Gerentes ou líderes de equipes podem se beneficiar com tais informações para (1) apoiar a gestão dos projetos, (2) recuperar informações técnicas já discutidas, (3) controlar a qualidade do produto, (4) delegar tarefas e (5) acompanhar o desenvolvimento dos projetos. Membros das equipes podem se beneficiar com tais informações para (1) conhecer o projeto e (2) compreender o raciocínio acerca de decisões já tomadas. Para tanto, foi desenvolvido o *framework Miner4DevTeam*, artefato produzido para apoiar a solução do problema identificado no domínio de aplicação desta tese (Figura 1.1). Neste estudo, é utilizada uma definição clássica de *framework*, a qual define *framework* como sendo uma estrutura de alto nível ou arquitetura de aplicativo composta por módulos projetados para serem refinados e usados como um grupo (Wirfs-Brock and Johnson, 1990). Como fonte de dados, o *framework* utiliza os registros (históricos) de discussões gerados pela interação de desenvolvedores através do uso de ferramentas de comunicação, como ferramentas de mensagens instantâneas (WhatsApp[1] e Slack[2]) e fórum de discussões voltado para desenvolvedores (GitHub *Discussions*[3]).

O *Miner4DevTeam* é composto por componentes independentes que encapsulam funcionalidades específicas e que, juntos, formam o *framework*, são eles: *Lost Knowledge Component* (*LK-Component*), *Frequent Knowledge Component* (*FK-Component*) e *Related Knowledge Component* (*RK-Component*). Esta decisão de projeto permite que as equipes de desenvolvimento utilizem o componente mais adequado à realidade do time, independentemente do uso dos demais. A estratégia de construção do *Miner4DevTeam* baseia-se em um modelo incremental e evolutivo, onde os componentes são construídos e suas versões são avaliadas e evoluídas de forma contínua até que os critérios de aceitação, estabelecidos no domínio de aplicação desta tese (Figura 1.1), fossem satisfeitos.

A Figura 3.1 ilustra o *Miner4DevTeam*. Os históricos de discussões dos desenvolvedores, realizados por *chats* ou fóruns do tipo PCQA, constituem fonte de entrada de dados para o *framework*. Tal escolha é motivada pelo fato de que os registros de comunicação de desenvolvedores constituem rica fonte de informação

[1]https://www.whatsapp.com/
[2]https://slack.com/
[3]https://docs.github.com/pt/discussions

acerca do produto desenvolvido (Alkadhi et al., 2017, 2018; Brisson et al., 2020; Hata et al., 2022).

Dentre os três componentes que compõem o *Miner4DevTeam*, o *LK-Component* e o *FK-Component* exploram o conteúdo textual de mensagens instantâneas trocadas por equipes atuantes na indústria de software. Ambos possuem o objetivo de identificar e extrair discussões relevantes referentes ao planejamento e ao gerenciamento de projetos de software. Contudo, os componentes implementam estratégias diferenciadas sendo projetados visando atender a demanda de públicos-alvos diferentes. Dado que discussões e decisões relevantes de projetos de software podem extrapolar os limites dos times de software, o terceiro componente, *RK-Component*, identifica postagens de discussões relacionadas trocadas por comunidades *open source* em fóruns de discussões do tipo PCQA. A estratégia de implementação do *RK-Component* baseia-se em algoritmos de aprendizagem profunda de máquina e os resultados, por ele gerado, atendem às demandas de mantenedores, contribuidores e usuários de projetos hospedados na plataforma GitHub. Apesar dos componentes possuírem propósitos similares, estes atuam sobre categorias, aqui pré-determinadas, de conhecimentos de software distintas:

1. ***Project Lost Knowledge* (PLK):** o PLK compreende as decisões e discussões de projetos feitas pelas equipes de desenvolvimento, através de mensagens instantâneas, que não foram formalmente registradas na documentação oficial do projeto ou, ainda, as decisões e discussões que, simplesmente, caíram no esquecimento dos membros das equipes sendo consideradas "perdidas". O PLK é um conhecimento relevante para que gerentes e líderes de equipe (público alvo) possam promover a qualidade do software, apoiar a evolução dos produtos desenvolvidos e apoiar o processo de tomada de decisões estratégicas nas empresas de software (Lima et al., 2019a).

2. ***Project Frequent Knowledge* (PFK):** o PFK compreende as decisões de desenvolvimento e gerenciamento do projeto que já foram bem discutidas pela equipe. Supõe-se que tais decisões já se tornaram claras e intrínsecas ao time de desenvolvimento. Contudo, estas podem não ter sido formalmente registradas na documentação oficial do projeto. O PFK é relevante para apoiar a familiarização de novos membros (público alvo) às equipes, pois

evidencia o escopo de desenvolvimento do software (Lima et al., 2020).

3. ***Project Related Knowledge* (PRK):** o PRK compreende as postagens de discussão relacionadas, que podem ser postagens duplicadas ou postagens quase duplicadas. Postagens duplicadas são aquelas com o mesmo conteúdo. Os itens de postagem duplicados (título, descrição e autor) podem ser cópias exatas ou próximas. Os autores das postagens podem reescrever alguns itens adicionando ou excluindo informações. A detecção de postagens duplicadas é essencial para mitigar o problema de duplicação em fóruns de discussão colaborativa de desenvolvedores. Postagens quase duplicadas são aquelas postagens com tópicos semelhantes. Diferentes usuários com interesses, perguntas ou ideias semelhantes criam e comentam sobre eles. Itens de postagens quase duplicadas (título, descrição e autor) não são os mesmos, mas compartilham informações relacionadas ao conteúdo um do outro. A detecção de postagens quase duplicadas é essencial para disseminar o conhecimento do projeto. Para fins práticos, nesta tese, postagens duplicadas e quase duplicadas são consideradas postagens relacionadas. Mantenedores, líderes de projetos ou moderadores são o público alvo deste tipo de conhecimento (Lima et al., 2023).

O *Miner4DevTeam* provê informações que viabilizam a determinação de um dos três tipos de conhecimento do projeto de software que se almeja identificar. Os dados são visualmente formatados e apresentados aos membros das equipes que, através de habilidades cognitivas de leitura e interpretação, transformam tais dados em conhecimentos sobre os projetos analisados.

Conforme a metodologia *Design Science Research* utilizada nesta pesquisa, a avaliação do *framework* foi feita de forma contínua, obedecendo aos critérios de aceitação pré-determinados (Figura 1.1). Sendo assim, optou-se por efetuar a avaliação do *framework* em duas etapas. Primeiramente foram conduzidos estudos em laboratório avaliam a eficácia das estratégias desenvolvidas. Em seguida, estudos realizados com equipes da indústria de software e mantenedores de comunidades *open source* (1) endossaram os resultados inicialmente obtidos, (2) validaram a adequação dos componentes em ambientes onde as equipes usam ferramentas de comunicação colaborativas e (3) identificaram oportunidades de aperfeiçoamento

do *framework*.

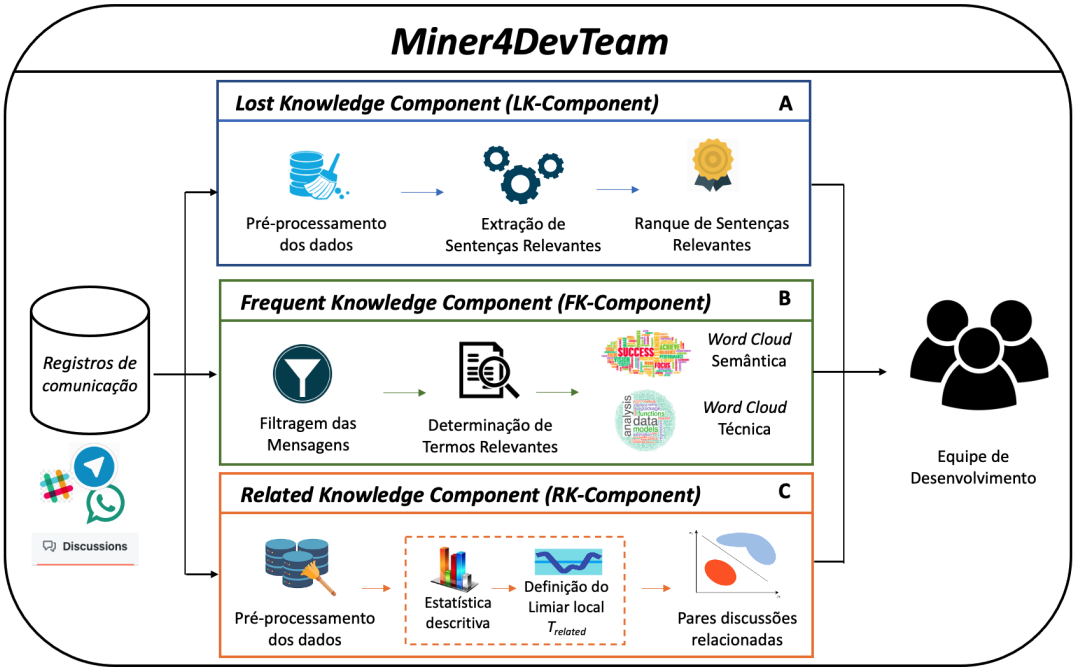

Figura 3.1: Componentes do *framework Miner4DevTeam*

A geração de cada componente corresponde a um ciclo de *design* da metodologia DSR que fundamenta a construção do *Miner4DevTeam*. As seções seguintes visam descrever as decisões de projeto realizadas para a criação de cada componente.

3.2 *Lost Knowledge Component*

Os registros de comunicação de times de desenvolvimento podem conter mensagens que levam a identificação de conhecimentos de projetos perdidos no ponto de vista de alguns *stackholders*, conhecimentos aqui denominados *Project Lost Knowledge* (PLK). Diante deste cenário, o *LK-Component* (Figura 3.1 - A) tem por objetivo identificar e extrair mensagens relevantes dos *logs* de comunicação das equipes de desenvolvimento que viabilizam a determinação do *Project Lost Knowledge* (PLK), sendo o PLK as decisões e discussões de projeto que não foram registradas na documentação oficial do projeto ou aquelas que caíram no esquecimento dos

membros da equipe. Para atingir tal objetivo, o *LK-Component* executa três etapas:

1. **Pré-processamento de dados**: nesta etapa, os dados a serem analisados são tratados e filtrados. São aplicadas técnicas de processamento natural de linguagem que incluem segmentação de frases, remoção de palavras irrelevantes e remoção de sinais de pontuação. Esta etapa é necessária para otimizar a execução do algoritmo de ranque executado na etapa seguinte (Gambhir and Gupta, 2017; Haiduc et al., 2010).

   Inicialmente, o *log* é segmentado em sentenças. Cada sentença é composta pelo conjunto de mensagens consecutivas enviadas por um mesmo emissor. Esta decisão de projeto foi tomada, pois uma análise feita manualmente nos *logs* de comunicação das equipes revelou que um emissor pode enviar várias mensagens consecutivas a fim de expressar sua decisão, opinião ou indagação a respeito de um assunto. Em seguida, são eliminados verbos, pronomes, artigos, preposições, dados não alfabéticos e mensagens administrativas enviadas pelas ferramentas de mensagens instantâneas, pois são considerados termos ruidosos que não contribuem para a seleção de sentenças relevantes a ser realizada na etapa seguinte (Gambhir and Gupta, 2017).

   Os dados, aqui tratados, são utilizados como fonte de entrada para a etapa seguinte executada pelo componente.

2. **Extração de sentenças relevantes**: esta etapa visa identificar sentenças relevantes úteis para a determinação o PLK.

   Uma forma de identificar e extrair sentenças relevantes de um texto é mapear o problema de extração em um problema de ranque. Sendo assim, a estratégia utilizada para determinar as sentenças mais relevantes registradas nos *logs* de comunicação das equipes de desenvolvimento foi a sumarização de texto, que cria resumos de grandes coleções de dados textuais a partir da identificação de sentenças relevantes capazes de descrever o domínio do assunto abordado. O algoritmo de ranque utilizado na implementação do *LK-Component* foi o *Text Summarization* (Seção 2.3). O algoritmo calcula a relevância dos termos das sentenças baseado no valor de $TF \times IDF$, onde TF

corresponde à frequência do termo e IDF e à frequência inversa do termo na coleção. Sabendo que as sentenças são compostas por diferentes termos, a medida de $TF \times IDF$ de cada termo é usada para calcular a pontuação de relevância das sentenças determinadas na etapa de pré-processamento.

O algoritmo *Text Summarization* fornece o valor de relevância de cada sentença analisada. Este valor é usado para ranquear as discussões importantes feitas pela equipe de desenvolvimento.

3. **Ranque de sentenças relevantes**: nesta etapa, as N sentenças identificadas como sendo as mais relevantes são determinadas. N é um parâmetro configurável e determinado pelo usuário do componente. As sentenças identificadas são usadas na determinação do PLK.

A identificação do PLK, através da determinação de sentenças relevantes, presentes nos registros de comunicação das equipes de desenvolvimento, revela alguns desafios próprios: (1) requer análise semântica dos dados, (2) requer conhecimento prévio sobre o projeto analisado e (3) envolve julgamento humano, uma vez que o conceito de relevância depende da perspectiva, perfil e maturidade dos avaliadores. Diferentes pessoas, com diferentes perfis, podem possuir diferentes visões acerca da determinação do PLK. Sendo assim, os dados gerados a partir do *LK-component* foram projetados para serem consumidos pelos gerentes de projetos das equipes, membros que possuem um conhecimento mais amplo dos projetos.

Para avaliar o *LK-Component* foram realizados estudos em dois contextos distintos:

- **Indústria**: Os registros de comunicação de dois times da indústria foram submetidos ao processamento do *LK-Component* e os gerentes identificaram o PLK. Os *logs* analisados foram gerados na ferramenta de comunicação WhatsApp e foram cedidos por duas *startups* que atuam no polo digital de Manaus. Nesse contexto, o PLK pode ser usado para apoiar a evolução dos produtos de software desenvolvidos e o processo estratégico de tomada de decisão das empresas. Mais detalhes sobre este estudo encontram-se descritos no Capítulo 4, na Seção 4.2.

- **Academia**: Dados de monitoramento de três equipes de projetos provenientes de um curso interdisciplinar, realizado com alunos do curso de computação, gerados a partir do uso da ferramenta Slack, foram analisados. No contexto acadêmico, o objetivo do estudo foi avaliar a utilidades dos *logs* como fontes de informação relevantes para monitorar o desenvolvimento dos projetos de software, monitorar o progresso dos alunos e evidenciar oportunidades de intervenção para os professores nas turmas. Mais detalhes sobre este estudo encontram-se descritos no Capítulo 4, na Seção 4.3.

3.3 *Frequent Knowledge Component*

Os registros de comunicação de times de desenvolvimento também podem conter mensagens que levam a identificação de conhecimentos frequentemente debatidos pelas equipes, conhecimentos aqui denominados *Project Frequent Knowledge* (PFK). Sendo assim, com o objetivo de capturar os conhecimentos frequentemente discutidos nos *logs* de comunicação de desenvolvedores de software, foi desenvolvido o *Frequent Knowledge Component* (Figura 3.1 - B). Por serem discutidos com frequência pelas equipes, os conhecimentos evidenciados pelo PFK são úteis para a integração de novos membros aos times de desenvolvimento, proporcionando-lhes oportunidades para conhecer o escopo de desenvolvimento dos projetos.

O escopo de desenvolvimento dos projetos de software tratados nesta tese limita-se às funcionalidades, restrições, dificuldades e desafios ocorridos durante o desenvolvimento do software e registrados nos *logs* de comunicação das equipes. A determinação do PFK é feita através da identificação dos assuntos frequentemente discutidos pelos times e da associação de tais assuntos aos trechos de discussões das equipes. Para tanto, o *FK-Component* executa três etapas:

1. **Filtragem de mensagens**: Diferentemente da etapa de pré-processamento executada pelo *LK-Component*, esta etapa visa identificar e excluir sentenças ruidosas e irrelevantes para a determinação do PFK. Para tanto são excluídas frases que expressam saudações, concordância, tempo e mensagens sem contexto.

   Durante a execução do processo de filtragem, as mensagens dos *logs* são di-

vididas em frases. Utilizam-se técnicas de PLN que, explorando pontuações gráficas, determinam as diversas frases que compõem uma mensagem. Posteriormente, são utilizados algoritmos de distância de edição (Seção 2.3) para identificar e excluir as frases ruidosas (frases de saudações, concordância, que expressam tempo e frases sem contexto). Amostras aleatória das frases a serem excluídas devem ser selecionadas dos *logs* analisados, em seguida, baseando-se na métrica de similaridade de *cosseno* (Seção 2.3), o algoritmo de cálculo de distância de edição identifica e exclui tais frases. Como resultado desta etapa, são identificadas frases possíveis de serem usadas na determinação do escopo de desenvolvimento dos projetos analisados.

2. **Determinação de termos relevantes**: nesta etapa, o conjunto de mensagens provenientes da etapa anterior é transformado em um conjunto de palavras. A classe gramatical de cada palavra é determinada. Em seguida, palavras consideradas de baixo valor informacional são eliminadas. Palavras de baixo valor informacional são aquelas que, muitas vezes, não possuem significado relevante para o contexto do problema (Gambhir and Gupta, 2017). As palavras enquadradas nas seguintes classes gramaticais são excluídas: artigos, preposições, pronomes, verbos e advérbios. Para atingir este objetivo, a técnica de PLN *part-of-speech* é utilizada (Seção 2.3). Também são retiradas sequências de caracteres que não agregam informações substanciais sobre o projeto, como: “rsrs”, “:(”, e-mojis, endereços eletrônicos, URLs e sequências de caracteres numéricos. Por fim, cada palavra passa a ser representada pela tupla $\langle t, f \rangle$, onde t representa um termo e f representa a frequência deste termo no conjunto de palavras, dando origem a uma lista ordenada L de assuntos frequentemente discutidos pela equipe. O objetivo desta etapa é gerar um conjunto de termos que possam ser usados para representar o escopo do software desenvolvido.

3. **Sumarização e externalização do PFK**: para sumarizar e externar o PFK são utilizadas *word clouds* (Seção 2.3). O componente gera duas *word clouds* que externam o escopo do projeto de software analisado:

    - **Semântica**: composta por termos, frequentemente usados nas discus-

sões da equipe, que tentam evidenciar características de especificação, planejamento, desenvolvimento, manutenção e gerenciamento do projeto de software.

- **Técnica**: composta por termos técnicos, frequentemente usados nas discussões da equipe, que tentam evidenciar características relacionadas ao uso de tecnologias e decisões técnicas no desenvolvimento do projeto de software.

A opção por gerar duas *word clouds* deu-se pois termos técnicos sobressaíram-se na geração de uma *word cloud* única. Sendo assim, as *word clouds* são formadas por termos excludentes. Além da lista L (assuntos frequentemente discutidos pela equipe) uma segunda lista T é utilizada na determinação do conteúdo das *word clouds*. A lista T é composta pelo conjunto de *tags* usadas pelo *Stack Overflow*[4] para categorizar as perguntas semelhantes que ocorrem no *site*, como, por exemplo: `Python`, `Ajax`, `Android`, etc. Por se tratar de um *site* de perguntas e respostas para programadores e profissionais da área de computação, as *tags* do *Stack Overflow* descrevem conteúdo técnico da área.

Contudo, verificou-se que alguns termos técnicos usados nas discussões das equipes não possuíam correspondência direta com as *tags* da lista T, como, por exemplo, os termos `dados`, `arquitetura`, `error`, `store`, etc. Porém, algumas *tags* podem ser usadas para descrever tais termos, como por exemplo `modelagem-de-dados`, `arquitetura-de-software`, `internal-server-error`, etc. Sendo assim, optou-se por desmembrar as *tags* compostas do *Stack Overflow* em *tags* contendo um único termo. Os termos desmembrados foram adicionados à lista T, disponível em `https://tinyurl.com/y36vdo4n`. A geração da lista T é feita apenas uma vez pelo componente, contudo esta deve ser mantida atualizada para contemplar as categorias definidas no *Stack Overflow*. As listas L e T foram usadas na determinação do conteúdo das *word clouds* da seguinte forma:

- **Semântica**: composta por termos pertencentes à L e não pertencentes

[4] https://pt.stackoverflow.com/tags

à T, $L - T$.

- **Técnica**: composta por termos pertencentes à interseção de L com T, $L \cap T$.

Estudos realizados para validar o *FK-Component* mostram que o componente é capaz de extrair assuntos relevantes presente nos *logs* de comunicação das equipes e que o conhecimento identificado (PFK) é útil para a integração de novos membros às equipes de desenvolvimento (Seção 5.2.2). Notou-se, ainda, que o uso do PFK não se restringe aos novos membros, membros antigos também podem se beneficiar com o conhecimento extraído para recordar decisões já tomadas. Mais detalhes sobre o estudo executado são discutidos no Capítulo 5.

3.4 *Related Knowledge Component*

Os registros de comunicação de times de desenvolvimento também podem conter mensagens repetidas que levam a identificação de conhecimentos aqui denominados *Project Related Knowledge* (PRK). Sendo assim, com o objetivo de capturar os conhecimentos relacionados discutidos por desenvolvedores de software, foi desenvolvido o *Related Knowledge Component* (Figura 3.1 - C). O *RK-Component* visa identificar discussões (ou postagens) relacionadas criadas em fóruns de discussões de desenvolvedores. O PRK compreende as discussões duplicadas e as discussões quase duplicadas ocorridas nos fóruns.

Conceituam-se postagens duplicadas e quase duplicadas da seguinte maneira.

- *Postagens duplicadas* são aquelas com o mesmo conteúdo. Os itens de postagem duplicados (título, descrição e autor) podem ser cópias exatas ou próximas. Os autores das postagens podem reescrever alguns itens adicionando ou excluindo informações. A detecção de postagens duplicadas é essencial para mitigar o problema de duplicação em fóruns de discussão colaborativa de desenvolvedores.

- *Postagens quase duplicadas* são aquelas postagens com tópicos semelhantes. Diferentes usuários com interesses, perguntas ou ideias semelhantes criam e comentam sobre eles. Itens de postagens quase duplicadas (título, descrição

e autor) não são os mesmos, mas compartilham informações relacionadas ao conteúdo um do outro. A detecção de postagens quase duplicadas é essencial para disseminar o conhecimento do projeto.

Nesta pesquisa, qualquer postagem duplicada ou quase duplicada é considerada como "postagem relacionada". O *RK-Component* fornece um conjunto de candidatos para postagens relacionadas denotado por R.

O componente se baseia na similaridade semântica existente entre o conteúdo das postagens para identificar pares de discussões candidatas a relacionadas. Assume-se que quanto maior o valor da similaridade semântica existente entre duas postagens, maiores são as chances de que estas sejam relacionadas. O valor de similaridade discussões duplicadas e discussões quase duplicadas, conforme a definição apresentada de PRK (Seção 3.1). Para fins práticos, discussões relacionadas são as duplicadas e as quase duplicadas. O *RK-Component* fornece como saída um conjunto de discussões candidatas a relacionadas, chamado de *RD-Candidates* (*Related Discussion candidates*). Para tando, o *RK-Component* é composto pelas etapas de pré-processamento de dados e verificador de similaridade, ambas descritas a seguir.

1. **Pré-processamento de dados**:

   A identificação de discussões relacionadas é feita por meio de algoritmos que calculam a similaridade semântica entre tais postagens. Contudo, a otimização da execução destes algoritmos também requer o pré-processamento dos dados manipulados (Zhou et al., 2017) para remover informações ruidosas e formatar os dados a serem processados. São executadas as seguintes subetapas de pré-processamento:

   Extração de atributos: nesta etapa, são extraídos os atributos de cada postagem de discussão. O componente usa os valores dos atributos para selecionar as postagens de discussão que devem ser avaliadas. São extraídos o nome do repositório, o identificador (ID), a categorias, o autor, a data de criação e o título de cada discussão.

   Seleção dos dados: nesta etapa, o conteúdo das discussões é formatado para minúsculo. Em seguida, são extraídos o título e o corpo (*body text*)

das discussões. Esta etapa se faz necessária, pois a similaridade entre as discussões analisadas é mensurada usando o título e o corpo (*body text*) das discussões principais (primeira postagem). Os comentários adicionados às *threads* de discussões não são considerados, pois correspondem a *feedbacks* ou raciocínios feitos em decorrência do tópico principal abordado na primeira postagem da discussão.

Remoção de código-fonte e URLs: nesta etapa são removidos os trechos de código-fonte presentes no corpo das discussões. Esta etapa se faz necessária, pois o modelo de aprendizagem de máquina utilizado no cálculo de similaridade das discussões é um modelo de uso geral, treinado para tratar problemas de PLN. Além disso, tanto a estrutura quanto o vocabulário de textos escritos em linguagem natural diferem de trechos de códigos (Sirres et al., 2018). O *RK-Component* analisa os documentos HTML de cada postagem de discussão e remove trechos de código e URLs incorporados em tags HTML específicas, por exemplo, `<code>` e `<a href>`.

Remoção de pontuação, emoji, números e palavras irrelevantes: nesta etapa, com o objetivo de eliminar ruídos, são removidos pontuações, *emojis*, números e *stopwords*. A remoção de pontuações, *emojis*, números e *stopwords* é feita usando a *Python NLTK Toolkit* [5] (Bird, 2006). Não foram removidos os símbolos de pontuações "." e "_" por serem usados na concatenação de palavras.

Lematização e Remoção de postagens vazias: Após as subetapas de remoção de ruído, o texto resultante passa pelo processo de *lematização* (Zhang et al., 2017). Por fim, as postagens de discussão de comprimento zero são desconsideradas.

2. **Verificador de similaridade:**

   Nesta etapa são identificados os pares de postagens de discussões candidatas a relacionadas. Como entrada são fornecidos os dados provenientes da etapa de pré-processamento. Como saída é gerado o conjunto R contendo pares de discussões candidatas a relacionadas. Esta etapa é dividida em duas

[5] https://www.nltk.org/

subetapas: (1) medição de similaridade e (2) seleção dos pares de discussões candidatas a relacionadas. As subetapas são descritas a seguir.

Medição de similaridade: o algoritmo utilizado para calcular a similaridade entre postagens do fórum de discussão é um verificador de similaridade semântica de textos (Agirre et al., 2015) (*semantic text similarity (STS)*) que utiliza modelos genéricos de aprendizagem de máquina profunda para gerar os *embeddings* das postagens que, posteriormente, possuem a similaridade mensurada através da métrica de similaridade de cosseno.

O modelo usado pelo *RK-Component* é o `all-mpnet-base-v2` [6], gerado a partir do *framework Python SentenceTransformers* (Reimers and Gurevych, 2019). O `all-mpnet-base-v2` é um modelo genérico pré-treinado a partir do *sentence-Bert*[7]. O modelo é usado para derivar *embeddings* semanticamente significativas de cada postagem analisada. O modelo mapeia frases e parágrafos (*sentences*) para um espaço vetorial denso de 768 dimensões. Pesquisadores otimizaram o `all-mpnet-base-v2` com mais de 1 bilhão de pares de frases coletadas de vários conjuntos de dados (Face, 2021). Na data de construção deste componente, dentre os modelos treinados pelo *sentence-Bert*, o `all-mpnet-base-v2` fornecia a melhor qualidade na geração das *embeddings* e no desempenho de tarefas de pesquisas semânticas (Reimers, 2021). Contudo, uma restrição imposta pelo modelo é o mesmo ser projetado para codificar frases e parágrafos curtos, truncando textos com mais de 384 palavras.

Modelos genéricos não se restringem a um contexto específico, pois são treinados e otimizados em diferentes bases de dados. Especificamente, o modelo `all-smpnet-base-v2` foi treinado usando como base o modelo `microsoft/mpnet-base` e otimizado usando mais de 1 bilhão de pares de frases coletadas de diferentes bases de dados (*Reddit comments (2015 – 2018), WikiAnswers Duplicate question pairs, Stack Exchange Duplicate questions (titles+bodies)*, etc.) (Face, 2021). Modelos específicos são treinados e otimizados de acordo com o contexto para o qual são criados, não garantindo

[6] https://huggingface.co/sentence-transformers/all-mpnet-base-v2
[7] https://www.sbert.net/

eficácia quando utilizados fora do contexto de treinamento. Além disso, a utilização de modelos genéricos proporciona vantagens como: (1) a não dependência de estruturas computacionais complexas para treino, validação e teste dos modelos, (2) a não necessidade de otimização dos modelos, e (3) a não necessidade de retreino, revalidação e reteste do modelo sempre que o contexto para o qual foi desenvolvido sofrer alterações (Polyzotis et al., 2017; Zhou et al., 2017; Schelter et al., 2018; Lee and Shin, 2020). O dinamismo da área de desenvolvimento de software pode ser considerado fator motivador para o uso de modelos genéricos. For fim, o uso de modelos públicos traz flexibilidade para o componente. À medida que novos modelos, compatíveis com o `all-smpnet-base-v2`, forem gerados e suas respectivas qualidades forem comprovadas, o componente pode ser configurado para utilizá-los.

Já a métrica de similaridade de cosseno é usada para computar a semelhança entre pares de *embeddings* gerados e definir o valor de similaridade entre pares de discussões, $similarity(master,target)$. A similaridade de cosseno gera valores entre zero e um. O valor 1 caracteriza que as postagens de discussões, *master* e *target*, são idênticas e o valor 0 caracteriza discussões dissimilares. Não foi definida relação de hierarquia ou prioridade entre as discussões *masters* e *targets*. Sendo assim, o valor de similaridade obtidos para o par $(master_i,target_j)$ é igual ao valor de similaridade obtido para o par $(target_j,master_i)$.

Sendo assim, cada par de discussão $(master,target)$ teve seu valor de similaridade calculado. Ou seja, dada uma discussão d_i, é calculada a similaridade de d_i contra todas as discussões pertencentes a coleção de discussões. Os valores de similaridade gerados são armazenados e usados como entrada para a etapa de seleção de candidatas a relacionadas.

3. *Seleção das candidatas a relacionadas*

   As postagens candidatas a relacionadas são identificadas a partir dos valores de similaridade calculados na etapa anterior. O componente utiliza um limiar para determinar se duas postagens estão ou não relacionadas. No entanto, não é utilizado um limiar fixo e universal.Optou-se por utilizar um limiar "local" ao conjunto de dados analisado, o $T_{related}$. $T_{related}$ é definido com

base na estatística descritiva dos valores de similaridades do conjunto de discussões analisado. Bases de entrada de dados diferentes têm valores de limiares diferentes. O uso de um limiar local proporciona flexibilidade ao *RK-Component* e possibilita a sua utilização em diferentes contextos.

O cálculo do $T_{related}$ é feito conforme os quatro passos descritos a seguir.

(a) **Determinação do valor de K:** K é um parâmetro da abordagem. O seu valor é usado para delimitar o espaço de procura por pares de discussões candidatas a relacionadas. Com o objetivo de selecionar os valores de similaridade das top-K discussões *targets* mais similares a cada discussão presente na base de dados, é determinado o valor de K. Quanto maior for o valor de K, maior é o número de valores de similaridades selecionados. Fazendo $K = 5$, serão selecionados os valores de similaridades das top-5 discussões *targets* mais similares a cada uma das n discussões analisadas. Fazendo $K = 10$ serão selecionados os top-10 valores de similaridade das discussões *targets* mais similares a cada discussão analisada. O valor de K é configurável e constitui uma entrada de dados para a abordagem.

(b) **Criação da distribuição S:** com base no valor de K são calculados os valores de similaridade que formarão a distribuição S. A distribuição S é formada pelos valores de similaridade das K discussões *targets* mais semelhantes a cada uma das n discussões *masters*.

Sendo n o total de discussões presentes na base de dados, K o número de discussões *targets* mais semelhantes a cada discussão *master* da base de dados e $value_{i\_j}$ o valor de similaridade entre as discussões $master_i$ e $target_j$, tem-se que S é dada por:

$$S =< value_{1\_1}, value_{1\_2},...,value_{1\_K}, value_{2\_1}, value_{2\_2}, ..., value_{2\_K},...,value_{n\_1}, value_{n\_2},...,value_{n\_K} >$$

(c) **Cálculo da estatística descritiva de S:** nesta etapa, é calculada a estatística descritiva da distribuição S. São identificados os valores correspondentes ao primeiro ($Q1$), segundo ($Q2$) e terceiro ($Q3$) quartis e o valor do intervalo interquartil (IQR) da distribuição S. Tais valores

são usados para calcular o valor do *upper inner fence* (Equação 3.1) que identificam os *outliers* da distribuição S (Tukey et al., 1977).

$$Upper\ inner\ fence = Q3 + (1.5 * IQR) \tag{3.1}$$

(d) **Determinação do limiar local ($T_{related}$):** ao assumir que quanto maior o valor da similaridade semântica existente entre duas postagens, maiores são as chances de que estas sejam relacionadas, o limiar usado para identificar os pares de discussões relacionadas é definido como sendo o valor de *Upper inner fence*. Sendo assim, tem-se que:

$$T_{related} = Upper\ inner\ fence \tag{3.2}$$

A definição do valor de K impacta na distribuição S gerada que, por sua vez, impacta no cálculo de $Q1$, $Q2$, $Q3$ e IQR, e, consequentemente, no valor do limiar $T_{related}$ definido. Por refletir características próprias da distribuição S, o limiar calculado é chamado de "limiar local".

Após a definição do valor do limiar local, é feita a determinação dos pares de discussão candidatas a relacionadas. Os pares de discussões (*master*,*target*) com valor de similaridade igual ou superior ao limiar local são destacadas pelo componente como sendo pares com discussões candidatas a relacionadas e formam o conjunto R. As postagens de discussões relacionadas são aquelas identificadas como sendo *outliers*. Calefato et al. (2022) também usaram a estatística descritiva para identificar os períodos reais de inatividade de desenvolvedores *open source*. Sempre que uma nova postagem de discussão for avaliada, sua similaridade deve ser avaliada com todas as postagens anteriores.

A eficácia do *RK-Component* foi avaliada em um estudo feito com dados de três projetos hospedados na plataforma GitHub e que usam o GitHub*Discussions*. O *Discussions* é um fórum nativo usado para intermediar discussões entre membros e usuários das comunidades hospedadas na plataforma GitHub (Hata et al., 2022). Destacado como sendo "*a new way for software communities to collaborate*

outside the codebase" Niyogi (2020), o fórum foi criado com objetivo de prover oportunidades para que membros e usuários das comunidades possam interagir colaborativamente e discutir tópicos relacionados aos projetos, como, por exemplo, a correção e ocorrência de defeitos (*bugs*), compreensão de discrepâncias e exceções encontradas, planejamento de novas funcionalidades (*features*) e, ainda, solicitar revisões de códigos, emitir anúncios, divulgar informações, realizar recrutamentos e outros inúmeros tópicos que envolvam um projeto de software (Hata et al., 2022; Liu, 2022).

Três mantenedores de software livre e três pesquisadores de ES (Engenharia de Software) avaliaram 14 conjuntos de postagens candidatas a discussões relacionadas. Foi medida a precisão (*precision*), a acurácia e a revocação (*recall*) do componente. Os avaliadores julgaram o relacionamento entre os pares de discussões candidatos a relacionadas. Mais detalhes sobre o estudo executado são discutidos no Capítulo 6.

Capítulo 4

Minerando *Project Lost Knowledge*

Este capítulo apresenta discussões relacionadas a aplicação e avaliação do *Lost Knowledge Component* (*LK-Component*) em dois diferentes contextos: na indústria de software e no contexto acadêmico. O estudo realizado na indústria revelou que o componente proposto foi eficaz na determinação de discussões que poderiam ser associadas a conhecimentos de projetos antes perdidos ou esquecidos do ponto de vista de gerentes de projetos. Tais discussões são relevantes para que gerentes e líderes de equipe promovam a qualidade do software, apoiar a evolução dos produtos desenvolvidos e apoiar o processo de tomada de decisões estratégicas nas empresas de software. Já no contexto acadêmico, a aplicação do componente oportunizou aos professores formas de acompanhamento mais efetivas do processo de desenvolvimento de projetos de software realizados por alunos matriculados em um curso interdisciplinar. Este tipo de curso tem o objetivo de propiciar a aplicação prática, de maneira integrada, do conhecimento e das habilidades que os alunos adquiriram durante o programa acadêmico.

O capítulo está organizado da seguinte forma: a Seção 4.1 apresenta o *Visual Abstract* das pesquisas realizadas com objetivo de definir e validar o *LK-Component*. A Seção 4.2 descreve o contexto de aplicação do *LK-Component* na indústria de software, o processo adotado para determinar o PLK no contexto especificado, o estudo de avaliação do componente, os resultados obtidos e as

discussões dos resultados gerados. A Seção 4.3 apresenta e discute a utilização e a avaliação do mesmo componente no contexto acadêmico. Por fim, a Seção 4.5 apresenta as considerações finais do capítulo.

4.1 *LK-Component - Visual Abstract*

O *LK-Component* é resultado do primeiro ciclo de *design* realizado nesta pesquisa. Visando apresentar um resumo dos principais pontos de idealização, desenvolvimento e validação do componente, a Figura 4.1 apresenta o *Visual Abstract* (Storey et al., 2017) das pesquisas que descrevem o *LK-Component* (Lima et al., 2019a,b). Nela são apresentados o objetivo, o contexto, a relevância, o rigor e a novidade do componente.

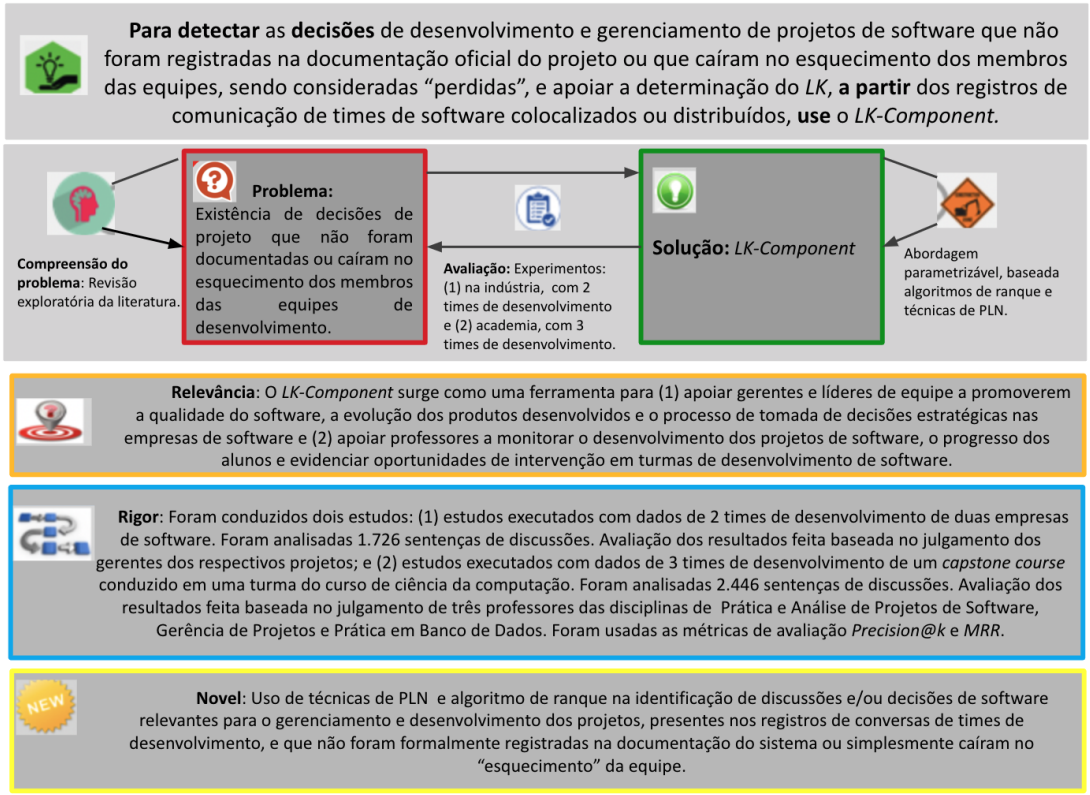

Figura 4.1: *Visual Abstract* do desenvolvimento do *LK-Component*.

As principais contribuições da execução do primeiro ciclo de *design* para a

indústria são:

- O desenvolvimento e avaliação de uma abordagem semi-automática, baseada em técnicas de PLN e algoritmos de ranque (*ranking*) para apoiar a identificação de decisões de projetos que não foram formalmente registradas na documentação do software e podem estar "perdidas" sob a perspectiva dos gerentes de projeto.
- Evidências experimentais sobre a perda de conhecimento dos projetos de software em mensagens de bate-papo das equipes de desenvolvimento.
- Evidências experimentais sobre as aplicações práticas do *LK-Component* em empresas que fazem uso de tais ferramentas de comunicação.

As principais contribuições da execução do primeiro ciclo de *design* para a academia são:

- Evidências experimentais sobre como estudantes de graduação em computação usam Slack para apoiar o desenvolvimento de projetos de software.
- Evidências experimentais sobre como o registro de comunicação dos estudantes podem dar suporte às atividades pedagógicas de professores de computação.
- Evidências experimentais sobre a adequação do *LK-Component* para apoiar as atividades dos professores, detectando necessidades técnicas dos alunos.

Como oportunidade de pesquisas futuras, destaca-se a necessidade de combinar diferentes evidências no cálculo do ranque das decisões mais relevantes e, como consequência, a otimização dos resultados gerados pelo *LK-Component*.

4.2 *LK-Component* aplicado à Indústria

Conforme descrito na Seção 3.2, o *LK-Component* visa permitir a determinação de conhecimento sobre projetos de software que esteja "perdido" em meio ao grande volume de mensagens registradas nos *logs* de comunicação das equipes de

desenvolvimento e que não foram formalmente registrados na documentação oficial do projeto. Para tanto, as N mensagens mais relevantes referentes a um período P de interação da equipe são determinadas e expostas aos gerentes de projetos que, ao lê-las, identificam o conhecimento de projeto perdido: o PLK. Tal estratégia é usada para evidenciar as discussões e decisões de projetos feitas pelas equipes e que caíram no esquecimento do time, apoiando equipes que lidam diariamente com as dificuldades e desafios próprios do contexto de desenvolvimento de projetos de software.

Um período P é determinado por um conjunto de dias $D = \{d_1, \ldots, d_n\}$, onde cada d_i é definido por um conjunto $M_{d_i} = \{m_{(1,d_i)}, \ldots, m_{(k,d_i)}\}$ de mensagens, e cada $m_{(j,d_i)}$ é uma tupla $\langle sender_{(j,d_i)}, date_{(j,d_i)}, content_{(j,d_i)} \rangle$. Sendo $sender_{(j,d_i)}$ um atributo que define o nome do remetente da mensagem, $date_{(j,d_i)}$ um atributo que define a data e hora de envio da mensagem e $content_{(j,d_i)}$ um atributo que define o conteúdo da mensagem. O conteúdo de um conjunto de mensagens enviadas em datas pré-definidas é analisado pelo componente com o objetivo de extrair as discussões de projeto de software relevantes.

Objetivando avaliar a efetividade do *LK-Component* como meio para viabilizar a determinação de PLK em projetos de software reais, foi realizado um estudo que analisou os *logs* de comunicação de duas equipes de desenvolvedores provenientes de duas empresas de software situadas no polo digital de Manaus. A metodologia, os resultados e as discussões geradas a partir deste estudo foram publicados em (Lima et al., 2019a).

Especificamente, as seguintes questões de pesquisa (QP) guiaram a investigação de validade do *LK-Component* no contexto da indústria de software:

- **QP1:** Pode-se extrair sentenças relevantes, relacionadas às discussões de projetos, dos *logs* de mensagens instantâneas dos desenvolvedores usando técnicas automatizadas?
- **QP2:** Pode-se identificar o PLK das discussões relevantes entrevistando os gerentes de projeto?

Os resultados obtidos revelam que o algoritmo de *Text Summarization* usado foi eficaz na determinação das N sentenças mais relevantes registradas nos *logs* de

comunicação das equipes e que os gerentes foram capazes de determinar o PLK dos projetos através da leitura e análise de tais sentenças. Os gerentes também enfatizaram ter interesse em adotar a abordagem que implementa o processo de identificação do PLK em outros projetos sob suas respectivas responsabilidades.

4.2.1 Determinação do PLK

A identificação do PLK é um processo semi-automático, no qual o *LK-Component* é usado para identificar automaticamente as sentenças mais relevantes, registradas em trechos do *log*, referentes ao desenvolvimento de projetos de software. A Figura 4.2 identifica as etapas da abordagem proposta para permitir a determinação do PLK. Neste processo, o gerente de projetos é o ator que interage ativamente na abordagem.

As seguintes etapas descrevem o processo apresentado na Figura 4.2:

1. Coleta de Dados do Projeto

Conforme ilustrado na Figura 4.2 - A, os gerentes devem fornecer os arquivos de *log* que desejam analisar. Em seguida, é gerado e exibido um gráfico contendo a distribuição do número de mensagens que a equipe de desenvolvimento enviou, por dia, ao longo do tempo, conforme Figura 4.3. Baseado no gráfico de distribuição, os gerentes de projetos devem selecionar o(s) período(s) de interesse para realizarem a análise do conteúdo dos *logs*. São definidas as datas limítrofes do(s) período(s).

2. Extração das Sentenças Relevantes

Nesta etapa (Figura 4.2 - B), as mensagens relacionadas ao(s) período(s) selecionado(s) são processadas pelo *LK-Component*, conforme descrição feita na Seção 3.2. Como resultado, o componente retorna, para cada período selecionado, as N sentenças mais relevantes.

3. Determinação do PLK

A determinação do PLK é feita por um avaliador que, através de um processo cognitivo, determina o *lost knowledge* associado às N sentenças mais relevantes

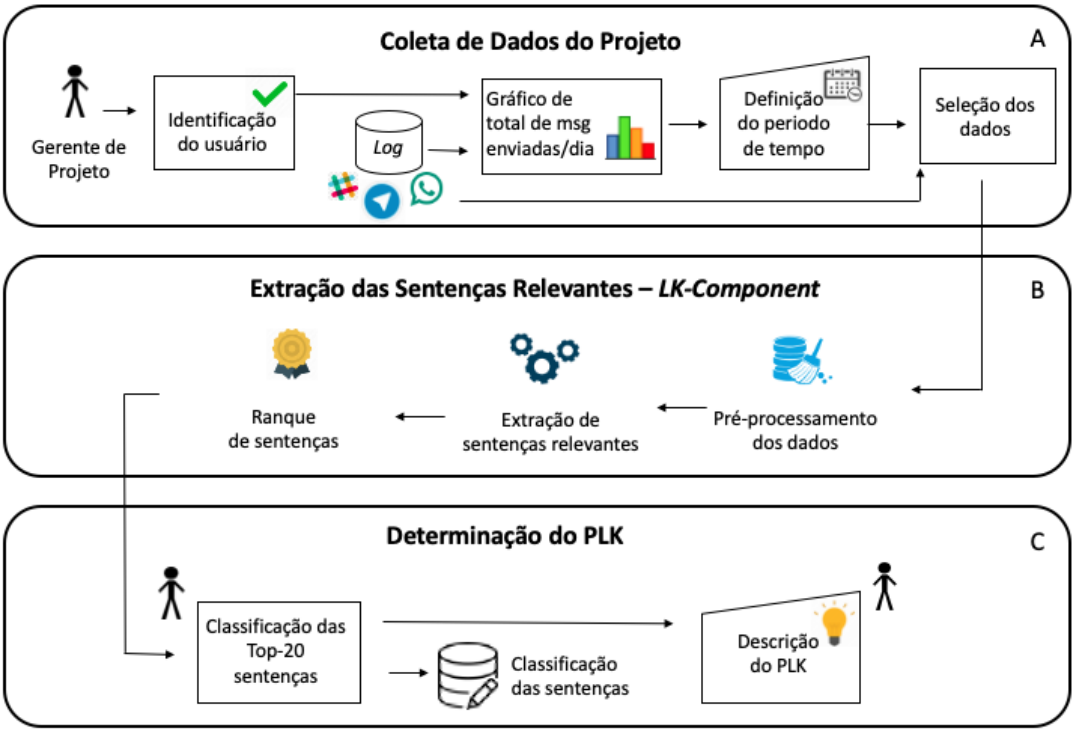

Figura 4.2: Processo de determinação do *Project Lost Knowledge*.

identificadas (Figura 4.2 - C). Para tanto, as sentenças relevantes são apresentadas aos gerentes para que estes possam determinar a importância de cada uma para o projeto. Por fim, após a leitura das sentenças relevantes, os gerentes descrevem o PLK identificado.

4.2.2 Avaliação do *LK-Component*

Um estudo replicado em duas empresas de desenvolvimento de software avaliou a efetividade do *LK-Component*. Foram analisados os *logs* de comunicação de duas equipes (Equipe A e Equipe B) pertencentes a empresas atuantes na indústria de software do polo digital de Manaus. Portanto, os resultados do estudo conduzido baseiam-se nas perspectivas de dois gerentes de projetos. O gerente da Equipe A possui 20 anos de experiência e o gerente da Equipe B atua há 10 no mercado de desenvolvimento de software.

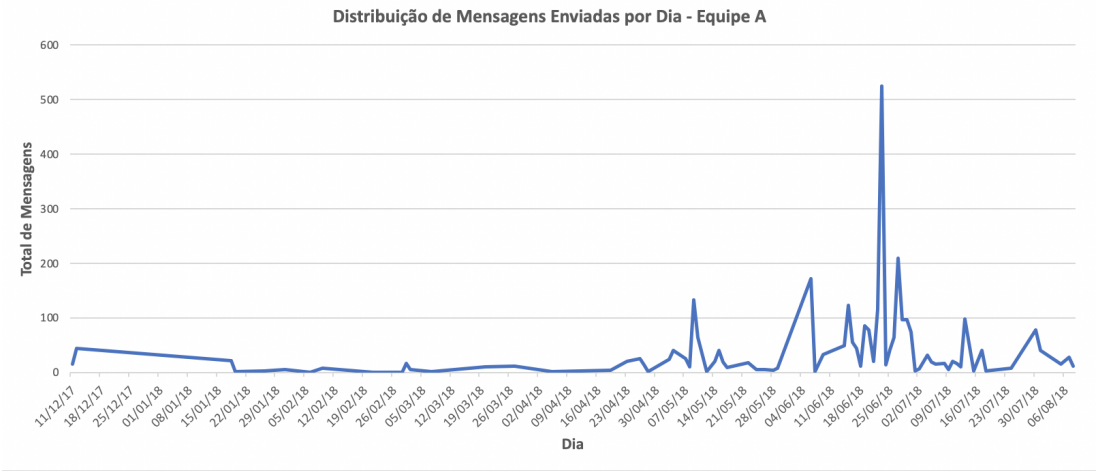

Figura 4.3: Gráfico de distribuição de mensagens enviadas por dia.

A seguir são apresentadas as etapas executadas durante a realização do estudo de avaliação do *LK-Component*. Apesar de serem apresentadas cinco etapas, as duas últimas etapas aqui descritas não são contempladas pela abordagem apresentada na seção anterior (Seção 4.2.1). Tanto a quarta quanto a quinta etapa aqui descritas são executadas apenas durante a valiadação do *LK-Component*.

1. Determinação da coleção de dados

Nesta etapa, foram coletados e analisados os *logs* de comunicação das Equipes A e B. Os registros de troca de mensagens fornecidos foram gerados através da ferramenta *WhatsApp*. Embora não se caracterizassem como times distribuídos, alguns membros, de ambas as equipes, possuíam horário de trabalho diferenciado dos demais. Sendo assim, as equipes utilizavam a ferramenta de mensagens instantâneas para se comunicar, tomar decisões em relação ao projeto desenvolvido e agendar reuniões. Os projetos foram indicados pelos próprios gerentes.

A Equipe A é composta por cinco membros: um gerente de negócios, um gerente de projetos, dois desenvolvedores e um testador. A equipe é especializada no desenvolvimento de tecnologias computacionais que promovem o engajamento e a interação de multidões em eventos culturais e esportivos como concertos, festivais e jogos. O Projeto A, desenvolvido por esta equipe, lidou com requisitos

incompletos, inconsistentes e passíveis de evolução. A interação da equipe era realizada por meio de reuniões presenciais e pela ferramenta de comunicação síncrona. O *log* de comunicação da Equipe A registra mensagens trocadas por, aproximadamente, um ano. Durante este período, a equipe manteve contato via a ferramenta por 104 dias e 3.325 mensagens foram enviadas no total. O Projeto A visa permitir a interação dos participantes do Festival Folclórico de Parintins através de dispositivos móveis.

A Equipe B é composta por seis membros: um *project owner* (PO), dois *scrum masters*, um *designer* de interface, dois desenvolvedores e um testador. A empresa da Equipe B provê serviços de desenvolvimento de soluções de software para outras empresas. O Projeto B, desenvolvido pela equipe, corresponde a um aplicativo de comércio eletrônico para dispositivos móveis. No *log* de comunicação da equipe foram registrados dados referentes a 285 dias de interação. No total, foram enviadas 5.256 mensagens. A empresa da Equipe B atua no mercado de desenvolvimento de software há seis anos. Atualmente, é composta por, aproximadamente, onze funcionários e adota o *Scrum* como processo de desenvolvimento de software.

2. Extração de sentenças relevantes

Nesta etapa, as mensagens presentes nos trechos de *log* referente ao(s) período(s) no(s) qual(is) o gerente expressa interesse em analisar são avaliadas automaticamente pelo *LK-Component*. As N sentenças mais relevantes são identificadas.

O componente foi avaliado usando as medidas *Precision@k* (P@k) e MRR descritas na Seção 2.3.1. Os valores de $P@3$, $P@5$ e $P@10$ foram calculados para medir a eficácia do *LK-Component* ao identificar as três, cinco e dez sentenças mais relevantes referentes aos trechos dos *logs* analisados.

3. Determinação do conhecimento identificado (PLK)

Nesta etapa, os gerentes dos projetos analisados descreveram o PLK identificado.

A determinação do PLK é feita a partir da leitura das N sentenças relevantes identificadas pelo *LK-Component*. Por se tratar de uma etapa que necessita de um conhecimento prévio e abrangente do projeto, os dados gerados a partir

do *LK-component* foram projetados para serem consumidos pelos gerentes de projetos das equipes. Contudo, a utilização de tais informações não se restringe apenas aos gerentes, outros membros que possuam familiaridade com o projeto desenvolvido podem se beneficiar dos conhecimentos técnicos extraídos dos *logs* de comunicação das equipes.

4. Análise manual dos dados

Nesta etapa, a fim de compreender e identificar os diversos tipos de discussões que podem estar registradas nos *logs* de comunicação de equipes de desenvolvimento, foi realizado um processo de codificação aberta em uma amostra aleatória das mensagens enviadas por ambas as equipes. Como resultado, foi determinado um conjunto de categorias que descrevem os tipos de discussões registadas nos *logs*. Em seguida, utilizando o conjunto de categorias, dois avaliadores codificaram amostras de sentenças separadamente. As discordâncias geradas foram discutidas e novas etapas de codificações foram executadas, até que a medição do grau de concordância entre os avaliadores atingisse um valor substancial. O grau de concordância foi mensurado através do coeficiente de *Kappa Cohens* (Cohen, 1960) e a interpretação de concordância definida por Landis e Koch em (Landis and Koch, 1977).

5. Realização de entrevista de acompanhamento

Por fim, com objetivo de compreender as necessidades por abordagens automáticas que viabilizem a identificação automática de conhecimentos de projetos de software, os gerentes foram solicitados a responder as seguintes perguntas:

- Você ficou surpreso ao ver que o projeto sob sua responsabilidade estava perdendo conhecimentos nos *logs* de comunicação dos desenvolvedores?
- O que você fez após saber que conhecimentos relacionados ao projeto estavam sendo “perdidos”?
- Você, como gerente de projetos, gostaria de usar a ferramenta de captura do conhecimento perdido em outros projetos?

4.2.3 Resultados

Como resultado da primeira etapa do estudo conduzido, os gerentes das equipes A e B determinaram o(s) período(s) de tempo sobre os quais foram executadas as análises das discussões das equipes e, consequentemente, identificados os PLKs.

O gerente de projetos da equipe A (Gerente-A) selecionou cinco diferentes períodos que, no total, correspondem a 49 dias de registros de comunicação, o que significa que o Gerente-A analisou 47.11% do total de dias em que a Equipe A estabeleceu trocas de mensagens através da ferramenta. Já o gerente de projetos da Equipe B (Gerente-B) selecionou sete períodos diferentes, totalizando 57 dias de registro de comunicação do time B, o que equivale a 20% do total de dias em que a Equipe B usou a ferramenta para se comunicar. A Tabela 4.1 sumariza os números que caracterizam os *logs* e os valores que mensuram a porcentagem de dados analisadas.

Na segunda etapa do estudo realizado, o *LK-Component* mensurou a relevância de 1.726 sentenças diferentes, sendo 845 sentenças originárias do *log* da Equipe A e 881 do *log* da Equipe B. Como, para cada período, o componente foi configurado para determinar as 20 sentenças mais relevantes o Gerente-A avaliou a relevância de 100 sentenças e o Gerente-B avaliou a relevância de 140 sentenças, conforme Tabela 4.1. Em média, os gerentes alocaram 22 minutos, por período, para avaliar as sentenças e determinar do PLK. A relevância avaliada está relacionada com a importância do conteúdo textual das sentenças para o projeto de software desenvolvido.

Tabela 4.1: Caracterização dos *logs* analisados no estudo da indústria.

	Equipe A	**Equipe B**
CARACTERIZAÇÃO DOS *LOGS*		
Total Dias de Interação Registrados	104	285
Total de Mensagens Trocadas	3.325	5.256
DADOS ANALISADOS		
Períodos Selecionados	5	7
Dias Analisados	49	57
%Dias Analisados	47.11%	20%
Sentenças Analisadas	845	881
Sentenças Relevantes Identificadas	100	140

A Tabela 4.2 sumariza a média dos valores obtidos pelas métricas usadas para quantificar a eficácia do *LK-component*. A média é obtida conforme número de período(s) analisado(s) por gerente.

A partir do ponto de vista do gerente de projetos da Equipe A, as três primeiras sentenças (top-3), dos cinco períodos avaliados, estão relacionadas a discussões relevantes para o projeto, logo o valor de $Precision@3$ atingiu valor máximo de 100% . Analisando as primeiras cinco sentenças mais relevantes (top-5), dos cinco períodos analisados, o Gerente-A afirma que, em média, 88% são relevantes para o projeto ($Precision@5 = 88\%$). Considerando as top-10, 74% delas foram avaliadas como sendo relevantes, atingindo o valor de $Precision@10 = 74\%$. Já o gerente de projetos da Equipe B destacou que, em média, 85% das três primeiras sentenças, referentes aos sete períodos analisados, são relevantes. Em relação as top-10 sentenças, 75% eram realmente relevantes. A medida em que o valor de k aumenta ($k = 2$, 5 e 10) é perceptível que o valor de precisão diminui, tal comportamento se justifica pelo fato de serem introduzidas sentenças julgadas como sendo não relevantes pelos gerentes. Contudo, a Tabela 4.2 também mostra que a abordagem proposta alcançou, em média, o melhor valor de MRR ($MRR = 1$) isso significa que todas as sentenças do topo do ranque foram consideradas relevantes para o projeto, de acordo com as perspectivas dos gerentes.

Tabela 4.2: Valores das métricas de avaliação.

	Equipe A	**Equipe B**
MRR	1	1
Precision@3	100%	85%
Precision@5	88%	80%
Precision@10	74%	75%

A seguir são listas exemplos de três sentenças identificadas pelo *LK-Component* como sendo relevantes. Todas as sentenças tiveram suas relevâncias endossadas pelos gerentes de seus respectivos projetos. Por questão de brevidade, são transcritos apenas trechos das sentenças identificadas:

- ***Log* Equipe A:**

  - *"...Já vimos que não da para deixar que os usuários tenham opção de ajustar o que precisamos ou não (hora, luminosidade, etc)... temos que*

'forçar' com que isto seja feito (automaticamente)... questões de design... Tem que ser mais profissional!...".

- "Por fim, observamos que o efeito de side de 3s está muito rápido e corre o grande risco de não sincronizar legal... o ideal é deixar 10s (e seguir com a troca de lados como está)... o ideal é incluir também o efeito 'estrala piscante' (azul e preto aleatórios... e azul e branco aleatórios)!!".

- "...se combinarmos uma pulseira com um app de celular? ela se comunicando com o smartphone ao invés de diretamente com uma estrutura. assim teríamos o celular livre pra fotos e mais mobilidade no caso de eventos sem sensibilidade de localização".

- ***Log* Equipe B:**

- "Características dos aparelhos que podem ser usados para testes do apk android:. - smartphone - versões do android a partir do 5.0. - resoluções de tela... vamos testar a partir das características dos smartphones listados e gerar novas versões... Correções feitas nessa versão:. - distorção da imagem do banner. - campo de telefone sem obrigatoriedade. - pesquisa por cep. - atualização das cores padrão".

- "... As 3h da tarde vou ver se já estamos vendo as mensagens dos medicamentos... A publicação é dividida em etapas. Lembro que vcs já me passaram por e-mail as credenciais da empresa...".

- "O Membro_J precisa testar se o aplicativo está funcionando corretamente, na hora de pagamento em convenio. Pensei em gerar regras para apenas 2 usuários, com poucos produtos, para fins de validação. Essa funcionalidade tbm e importante para o fechamento do projeto e implantação do aplicativo.".

Portanto, respondendo à QP1 deste estudo: Pode-se extrair sentenças relevantes, relacionadas às discussões de projetos, dos *logs* de mensagens instantâneas dos desenvolvedores usando técnicas automatizadas?

Sim. O estudo realizado evidenciou ser possível utilizar o algoritmo de *Text Summarization* para extrair automaticamente sentenças relevantes dos *logs* de comunicação de equipes de desenvolvimento de software.

Na terceira etapa do estudo realizado, foi solicitado que os gerentes de projeto de ambas as equipes descrevessem o *Project Lost Knowledge* identificado após a leitura das sentenças relevantes de cada período avaliado. A seguir são transcritos os PLKs determinado pelos gerentes de projetos das equipes participantes do estudo. Como a determinação do PLK é feita por período, o Gerente-A descreveu cinco PLKs e o Gerente-B descreveu sete PLKs, conforme citações abaixo:

- **Citações do Gerente-A:**

  Período 1: *"Definição da mudança da duração e velocidade dos efeitos. Motivo de bloqueio do app em dias que não deve ser utilizado (uso apenas nos ensaios técnicos e nos dias de apresentação)".*

  Período 2: *"Necessidade de dar menos opções de intervenção pro usuário no app. Ajustes de data/hora devem ser automatizados pelo próprio app. Possibilidade/viabilidade de desenvolvimento de gadgets específicos para uso em multidões".*

  Período 3: *"Acabei lembrando de negociações que foram feitas para possível uso da tecnologia em um evento cultural no Rio de Janeiro".*

  Período 4: *"Teste de iluminação para verificar se conseguíamos ver as cores na tela do celular em diferentes horas do dia, com diferentes condições de iluminação. Também precisávamos verificar se conseguíamos distinguir cada cor a uma certa distância do celular".*

  Período 5: *"Evidenciou a necessidade de integração dos efeitos visuais com sons. Coisa que até o momento ainda não foi realizada".*

- **Citações do Gerente-B:**

  Período 1: *"Algumas decisões de projeto foram alteradas para melhorar o serviço de webservice, e algumas alterações de navegação e usabilidade*

foram tomadas para melhorar a experiência do usuário durante a compra de um produto.".

Período 2: *"As regras de usuários para convênios não tinham sido entendidas ou definidas. Assim, a equipe não sabia como a aplicação deveria tratar ou se comportar quando os usuários que tivessem convênio logassem na aplicação para fazer a compra de produtos. Essa regra ainda não tinha sido implementada na aplicação, faltava definir pelo cliente como seria feito".*

Período 3: *"Foi importante lembrar que as mensagens que seriam exibidas para os medicamentos na funcionalidade de detalhe, foram definidas devido uma norma do Ministério da Saúda. Só foi visto essa necessidade durante os testes de aceitação do aplicativo pelo cliente".*

Período 4:*"Foi importante pois foi definido como seria consumido a parte de configuração do perfil do usuário. Além disso, não lembrava que tinha sido feito casos de testes para o requisito de 'usuário com convênio'."*

Período 5: *"Eu não lembrava que o cliente tinha uma pendência sobre o status do pedido, que teria que ser disponibilizado via webservice. Além disso, ocorreu alguns problemas de acesso no serviço disponibilizado via webservice".*

Período 6: *"Na nova versão do aplicativo tinham sido incluídos as funcionalidades de 'Fidelidade' e 'Resgatar Prêmios'. Além disso, algumas regras de usuários com convênios foram definidas".*

Período 7: *"Descrição dos testes não funcionais que tinham sido feitos em diferentes aparelhos na versão Android da aplicação, estava ocorrendo erros de distorção de imagem e a tela fechava.".*

Pode-se notar que o Gerente-A identificou algumas decisões importantes de projeto em suas citações de PLK. Analisando-as, nota-se que tais decisões podem ser usadas para: (1) identificar detalhes de implementação do código-fonte, quando o Gerente-A registra: - *"...mudança da duração e velocidade dos efeitos. Motivo de bloqueio do app..."*, (2) oferecer suporte à elicitação de requisitos, quando Gerente-A afirma: - *"Ajustes de data/hora devem ser automatizados pelo próprio app..."*,

(3) determinar decisões gerenciais, quando Gerente-A afirma - *"...lembrando de negociações que foram feitas para possível uso da tecnologia..."*, (4) determinar necessidade de testes do app, quando Gerente-A afirma - *"... teste de iluminação para verificar se conseguíamos ver as cores na tela do celular em diferentes horas do dia... verificar se conseguíamos distinguir cada cor ..."*, e (5) oportunidades de evolução para o produto desenvolvido, quando Gerente-A afirma - *"...necessidade de integração dos efeitos visuais com sons..."*.

Já as citações do Gerente-B identificaram PLKs relevantes em relação às (1) atualizações do projeto de software, quando o Gerente-B afirma: - *"...Algumas decisões de projeto foram alteradas para melhorar..."*, (2) incompletude dos requisitos do sistema, quando Gerente-B afirma: - *"...Essa regra ainda não tinha sido implementada na aplicação, faltava definir pelo cliente como seria feito..."* , (3) testes do produto, quando Gerente-B escreve: - *"... não lembrava que tinha sido feito casos de testes para o requisito de 'usuário com convênio'... "*, e (4) necessidade de manutenção, quando o mesmo diz: - *"...estava ocorrendo erros de distorção de imagem e a tela fechava."*. O PLK, determinado pelo Gerente-B, também permitiu compreender algumas regras de negócio adotadas que foram baseadas em normatizações legais associadas ao contexto de desenvolvimento do projeto. Fato explicitamente escrito na citação referente ao período 3: -*"...devido uma norma do Ministério da Saúde..."*. Esta citação identifica e justifica as mensagens de descrição de medicamentos exibidas pelo aplicativo.

Os PLKs determinados por ambos os gerentes podem ser usados para promover a transferência de conhecimentos de software, disseminar informações entre membros da equipe e apoiar a criação de novas ideias que possam ser implementadas em novas versões dos produtos desenvolvidos. Apesar de trabalharem em contextos de projetos diferentes, observa-se que ambas as equipes discutem e tomam decisões importantes através de ferramentas de mensagens instantâneas. Sendo estas últimas, ricas fontes de informação acerca dos projetos desenvolvidos.

Respondendo à QP2 deste estudo: Pode-se identificar o PLK das discussões relevantes entrevistando os gerentes de projeto?

Sim. É possível identificar *Project Lost Knowledge* a partir de sentenças relevantes identificadas nos *logs* de comunicação de equipes de

desenvolvimento de software.

A próxima etapa da avaliação foi conduzida com o objetivo de compreender os tópicos discutidos pelas equipes de desenvolvimento por meio das ferramentas de mensagens instantâneas. Um processo de codificação aberta identificou as categorias de discussões mais frequentes ao contexto analisado, conforme demonstrado na Tabela 4.3. Em seguida, dois avaliadores codificaram amostras de sentenças conforme categorias apresentadas na Tabela 4.3. O processo de codificação executado foi respaldado pelo valor $Kappa = 0{,}61$, para a codificação executada no *log* da Equipe A, e $kappa = 0{,}64$, para a codificação executada no *log* da Equipe B. Os dois valores indicam concordância substancial de acordo com a interpretação proposta por Landis e Koch (Landis and Koch, 1977).

Tabela 4.3: Tipos de discussões ocorridas em *log*

Categoria	**Descrição**
Requisitos	Definições de funcionalidades, serviços e restrições do software
Planejamento	Definições de planejamento e projeto do software
Desenvolvimento	Definições de configuração e implementação do software
Deployment	Instalação do Software
Delivery	Entrega de uma *release* do Software
Teste	Teste do software
Aceitação do software	*Feedback* dos usuários
Suporte	Assistência de instalação, configurações e solução de problemas
Negócios Empresariais	Gerenciamento da Empresa
Gerência	Gerência do projeto
Marketing	*Marketing* do projeto
Visão de Produto	Definições de *features* para novas versões e discussões sobre produtos similares
Nome do Produto	Definição do nome do aplicativo
Reuniões	Coordenação de reuniões de trabalho
Integração da Equipe	Discussões relacionadas à interação social da equipe

Em seguida, com objetivo de identificar as categorias de PLK mais frequentes, os gerentes foram solicitados a identificar os tópicos associados a cada transcrição de PLK feita. A Tabela 4.4 resume a frequência de cada categoria encontrada

Tabela 4.4: Tipos de discussões reveladas pelo *PLK*

PLK Categoria	**Gerente-A**	**Gerente-B**
Requisito	1	7
Teste	2	3
Planejamento	2	1
Visão de Produto	2	1
Gerência	1	1
Desenvolvimento	0	1
Delivery	0	1
Aceitação do software	0	1
Negócios Empresariais	1	0

na identificação dos PLKs. De acordo com a perspectiva do Gerente-A, PLKs referentes a elicitação de requisitos, ao planejamento, ao teste, ao gerenciamento do sistema foram identificados. Já o tipo de PLK mais comum determinado pela perspectiva do Gerente-B abrange a elicitação de requisitos. Em cada transcrição de PLK é possível identificar mais de uma categoria de tipos de discussão.

Por fim, a avaliação do componente foi concluída com a realização de entrevistas de acompanhamento com os gerentes, com o objetivo de compreender quais suas respectivas percepções em relação aos PLKs identificados. A seguir são apresentados trechos das transcrições das entrevistas realizadas:

1. **Você ficou surpreso ao ver que o projeto sob sua responsabilidade estava perdendo conhecimento nos *logs* de comunicação dos desenvolvedores?**

   Gerente-A: - *"... ao ler as mensagens relevantes recuperadas, lembrei detalhes sobre as decisões que tomamos e testes relacionados ao software que eram importantes para o produto. Também lembrei de aspectos de gerência da empresa, como, por exemplo, ações que tivemos que tomar para capturar novos clientes que poderiam se interessar pelo aplicativo. Foi surpreendente observar coisas das quais não me lembrava, apesar de ter selecionado apenas cinco períodos de tempo para analisar. Acho que se tivesse escolhido mais períodos, teria encontrado mais PLK."*.

   Gerente-B: - *"Fiquei surpresa pela quantidade de conhecimentos que encontramos durante essa análise. Do conhecimento que encontrei, muitas*

coisas eram acordadas somente através da ferramenta, sem ser por uma reunião formal!".

2. **O que você fez após saber que conhecimentos relacionados ao projeto estavam sendo "perdidos"?**

   Gerente-A:- *"O software evoluiu (a equipe implementou novos lançamentos) e alguns PLKs foram levados em consideração... O conhecimento identificado foi essencial, pois lidamos com um público diverso com diferentes conhecimentos tecnológicos. Outros aspectos que discutimos, como a interação de efeitos de iluminação com sons, estão atualmente em desenvolvimento para uma nova versão de nossa tecnologia.".*

   Gerente-B - *"Este projeto específico terminou e eu não tenho mais contato com os desenvolvedores deste projeto. Porém, podemos usar o conhecimento identificado como fonte de informação para resgatar as decisões tomadas e usar o aprendizado adquirido em novos projetos.".*

3. **Você, como gerente de projetos, gostaria de usar a ferramenta de captura do conhecimento perdido em outros projetos?**

   Gerente-A: - *"Seria fantástico uma ferramenta que extrai automaticamente o PLK dos logs de mensagens dos desenvolvedores e em outras fontes de dados, como os e-mails. Eu, como gerente de projetos, usaria como padrão para todos os projetos desenvolvidos na minha empresa.".*

   Gerente-B: - *"Sim! Na minha opinião, é importante o time ter um apoio ferramental, principalmente para aplicativos de mensagens! Muita coisa se perde durante o projeto ao utilizar aplicativo de mensagens. Hoje em dia se utiliza muito desses aplicativos, justamente para tentar melhorar a comunicação do time e cliente.".*

Uma análise nas citações dos gerentes revela que ambos foram capazes de determinar ou lembrar decisões de projeto importantes. Fato que pode evidenciar que os próprios gerentes não estavam cientes do quão rico, em termos de fonte de conhecimentos, podem ser os *logs* de discussões das equipes de desenvolvimento. Ao serem questionados sobre suas respectivas posturas diante da identificação dos

"conhecimentos perdidos", os gerentes entrevistados destacaram oportunidades de intervenções distintas. O gerente da Equipe A destacou que o PLK foi usado como fonte de dado para *features* do novo sistema construído, criando oportunidades de evolução para projeto. Já o gerente da Equipe B destacou que o processo de identificação do PLK cria oportunidades para a transferência de conhecimentos adquiridos. Quando indagados sobre a possibilidade de utilizarem a abordagem de captura de PLK em outros projetos, ambos demonstraram interesse. O Gerente-A destacou o interesse em analisar outras fontes de dados, como e-mail. Já o Gerente-B enfatizou a importância da criação de ferramentas que apoiem as equipes de desenvolvimento de software na realização de suas tarefas.

4.2.4 Discussões

O estudo executado revelou o uso promissor do algoritmo de *Text Summarization* para extrair sentenças relevantes registradas nos *logs* de bate-papo de equipes de desenvolvimento. Os gerentes identificaram o PLK associado a cada período de comunicação analisado. Em média, os gerentes levaram 22 minutos para avaliar cada conjunto de 20 sentenças relevantes e identificar o PLK a elas associado. Métricas para avaliações quantitativas e análises feitas nas citações dos gerentes indicam que as sentenças relevantes extraídas pelo *LK-Component* permitem a determinação do PLK. Contudo, o algoritmo de ranque utilizado é sensível a erros ortográficos, pois a métrica $TF \times IDF$, usada para mensurar a relevância das sentenças analisadas, bonifica termos raros. Sendo assim, direcionamentos futuros apontam para a necessidade de uso de novas métricas ou para a combinação de diferentes métricas no cálculo do ranque das sentenças.

Muitas decisões e discussões relevantes de projetos podem ser identificadas durante o processo de determinação do PLK, criando oportunidades para transferência de conhecimentos, transformação de conhecimentos e criação de novas ideias. A transferência de conhecimentos pode ajudar a disseminar os conhecimentos do projeto por toda a equipe e reduzir a chance de introduzir inconsistências e erros no código. A transformação de conhecimentos permite atualizar o conhecimento do software e aprofundar o entendimento da equipe sobre os projetos desenvolvidos. A determinação do PLK também pode ajudar a conceituar novas

ideias e desenvolver novas maneiras de auxiliar os negócios da empresa.

A perda de conhecimentos leva a uma variedade de problemas, como a perda do raciocínio que envolve as decisões de projeto tomadas, a diminuição da compreensão do software, a degradação do compartilhamento de conhecimentos e, ainda, dificulta a avaliação do software (Manteuffel et al., 2014; Soria and van der Hoek, 2019). Observou-se também que, embora o PLK mais frequente estivesse relacionado à definição de requisitos, cada projeto possui uma distribuição diferente dos tópicos dos PLKs identificados. Isso destaca que o projeto pode perder conhecimentos em diferentes estágios do ciclo de vida de desenvolvimento do software e o PLK não está necessariamente vinculado a nenhuma etapa específica.

A pesquisa aqui apresentada rastreou apenas uma vertente de troca de conhecimentos: a feita via ferramenta de mensagens instantâneas. Não foram exploradas outras fontes de dados a fim de identificar a viabilidade de captura de conhecimentos em outras mídias, como e-mail, Slack, comentário de código-fonte, *backlogs* ou *daily meetings*. Sendo assim, não é possível afirmar que a solução proposta contempla todas as possibilidades de identificação de conhecimentos perdidos de projetos de software.

Contudo, entrevistas com os gerentes de projetos revelaram que ambos tiveram uma boa experiência ao identificar o PLK. Eles destacaram que o PLK é muito importante para promover a qualidade do software, adquirir conhecimentos e produzir novos lançamentos de produtos. Afirmaram, ainda, que gostariam de adotar ferramentas que os informam automaticamente sobre conhecimentos perdidos para apoiar o processo de desenvolvimento de software de suas empresas.

Dado que foram examinados os *logs* de apenas dois projetos, não é garantido que os resultados aqui obtidos possam ser generalizados para todos os projetos de software, porém o estudo realizado demonstrou que o *LK-Component* é útil para apoiar a determinação de conhecimentos perdidos de projetos de software.

4.3 *LK-Component* aplicado na academia

Com o objetivo de avaliar a aplicabilidade do *LK-Component* no apoio às tarefas pedagógicas de professores da área de Engenharia de software, o componente foi utilizado e avaliado como módulo integrante de uma ferramenta analítica de-

senvolvida para evidenciar conhecimentos de software a partir dos registros de comunicação de alunos matriculados em um curso interdisciplinar. A ferramenta utiliza técnicas estatísticas e algoritmos de mineração de dados para analisar dados provenientes dos *logs* de comunicação dos alunos e evidenciar padrões comportamentais que apoiem o monitoramento das equipes por parte dos professores.

No estudo aqui apresentado, o *LK-Component* foi utilizado para analisar os registros de comunicação produzidos pela interação de estudantes matriculados em um curso interdisciplinar que envolveu as disciplinas de Prática e Análise de Projetos de software, Gerência de Projetos e Prática em Banco de Dados. O *Project-based Learning* (PBL) foi adotado como metodologia de ensino norteadora das atividades realizadas no curso. Foi solicitado que as equipes utilizassem as mesmas ferramentas utilizadas por equipes da indústria para dar suporte ao processo de planejamento, desenvolvimento, verificação, validação, teste e gerência dos projetos de software desenvolvidos. São elas: (1) Github [1], usado para apoiar o desenvolvimento colaborativo e controlar o versionamento dos códigos desenvolvidos; (2) Slack, usado para apoiar a comunicação das equipes; (3) Google Drive, serviço usado para dar suporte ao armazenamento e compartilhamento de artefatos produzidos; e (4) Trello [2], ferramenta utilizada para dar suporte ao gerenciamento das tarefas das equipes.

A Figura 4.4 demonstra a ferramenta analítica desenvolvida. Um estudo de avaliação conduzido avaliou a ferramenta proposta como recurso para análise dos *logs* de comunicação das equipes, fornecendo informações para apoiar o monitoramento do desenvolvimento dos projetos realizados. Foi analisada a eficácia da ferramenta em prover meios para detectar demandas de cunho técnico e social geradas por parte dos alunos.

Especificamente, este estudo planeja responder às seguintes questões de pesquisa (QP):

- **QP1:** Pode-se usar os metadados das ferramentas de comunicação para analisar e compreender o padrão de comportamento das equipes de estudantes?
- **QP2:** Pode-se usar técnicas de mineração de dados para identificar e ex-

[1] https://github.com/
[2] https://trello.com/pt-BR

trair sentenças relevantes, sob a perspectiva do professor, dos registros de comunicação das equipes de estudantes?

A metodologia e os resultados gerados a partir deste estudo foram publicados em (Lima et al., 2019b).

4.3.1 Aplicação do *LK-Component* na educação

No contexto da educação, o *LK-Component* é usado como módulo autônomo na proposta de uma ferramenta analítica desenvolvida para apoiar as atividades pedagógicas, de monitoramento, de professores de ES (Figura 4.4 - C).

Os *logs* analisados foram produzidos a partir da interação de equipes formadas por estudantes de ES. Por se tratar de uma experiência que visava o desenvolvimento de soluções para necessidades reais, as equipes precisaram compreender, abstrair, projetar, implementar e validar soluções eficientes e com qualidade. Durante o desenvolvimento dos projetos, as equipes possuíam autonomia para decidir a melhor forma de atender os requisitos solicitados pelos clientes. Os próprios membros, monitorados pelos professores, propunham discussões e tomavam decisões a respeito de todo o processo de desenvolvimento e gerenciamento do software desenvolvido. Conforme metodologia de desenvolvimento de software adotada, o SCRUM, os projetos desenvolvidos eram divididos em etapas (*sprints*). A cada *sprint*, novos requisitos de software poderiam surgir e novos desafios técnicos poderiam ser enfrentados pelas equipes. Ao final do curso, as equipes apresentaram os resultados obtidos aos professores e aos seus clientes.

Com o objetivo de viabilizar a comunicação das equipes, os grupos usaram a ferramenta Slack, sendo esta projetada para facilitar a colaboração entre grupos de pessoas, mediante recursos de bate-papo, compartilhamento de mídias e integração com outros serviços. A ferramenta Slack disponibiliza informações de metadados referentes ao funcionamento de cada canal de comunicação criado (subgrupos que reúnem participantes com objetivos comuns) e a participação de cada usuário no ambiente. O Slack foi utilizado com ferramenta de apoio para discutir atividades e tomar decisões em relação ao processo de desenvolvimento e gerenciamento do projeto de software. Em consenso com os estudantes, o professor da disciplina Prática e Análise de Projetos de software foi adicionado aos grupos de discussões

criados no Slack. Sendo assim, os estudantes estavam cientes de que o professor tinha acesso às mensagens trocadas pelas equipes através da ferramenta.

4.3.1.1 Ferramenta analítica para apoio educacional

A ferramenta analítica desenvolvida extrai dados dos registros de comunicação das equipes, calcula e reporta informações estatísticas em formato de gráficos e tabelas e identifica as discussões mais relevantes registradas nos *logs*. A ferramenta fornece meios para identificar as dificuldades técnicas e de interação das equipes, os mal-entendidos e as dúvidas, evidenciando oportunidades para intervenções por parte dos professores.

A Figura 4.4 demonstra a ferramenta analítica desenvolvida e os recursos, por ela, oferecidos. Ao acessarem a ferramenta os professores devem se identificar e registrar os *logs* que serão analisados (Figura 4.4 - A). Após determinar o *log* que se deseja analisar, um gráfico exibe a distribuição de mensagens enviadas por dia pela equipe, semelhante ao gráfico mostrado na Figura 4.3. O gráfico de distribuição mostra a troca de mensagens realizada no canal *general* do *workspace* das equipes. O gráfico apoia a seleção dos períodos P dos *logs* a serem analisados. Após a definição dos períodos, são selecionas as amostras de dados a serem analisadas. Em seguida, a ferramenta disponibiliza duas opções de análise sobre os dados: estatística (Figura 4.4 - B) e mineração de dados 4.4 - C).

1. Módulo Estatístico:

O módulo estatístico (Figura 4.4 - B) tem o objetivo de sumarizar e relatar o comportamento dos membros das equipes sob a perspectiva individual e coletiva de interação destes com o grupo. Os dados gerados pelos recursos disponibilizados neste módulo apoiam o monitoramento o progresso das equipes durante todas as etapas do desenvolvimento dos projetos. Os recursos do módulo sintetizam os valores que as variáveis analisadas assumem e geram gráficos e tabelas estatísticos baseados nos metadados, providos pela ferramenta Slack, e na frequência dos termos das mensagens enviadas. O módulo oferece os seguintes recursos:

- **Gráfico de distribuição de mensagens:** Gráfico usado para descrever a distribuição de mensagens enviadas pela equipe durante um período predefinido de tempo. Considerando que o desenvolvimento de um software trata-se

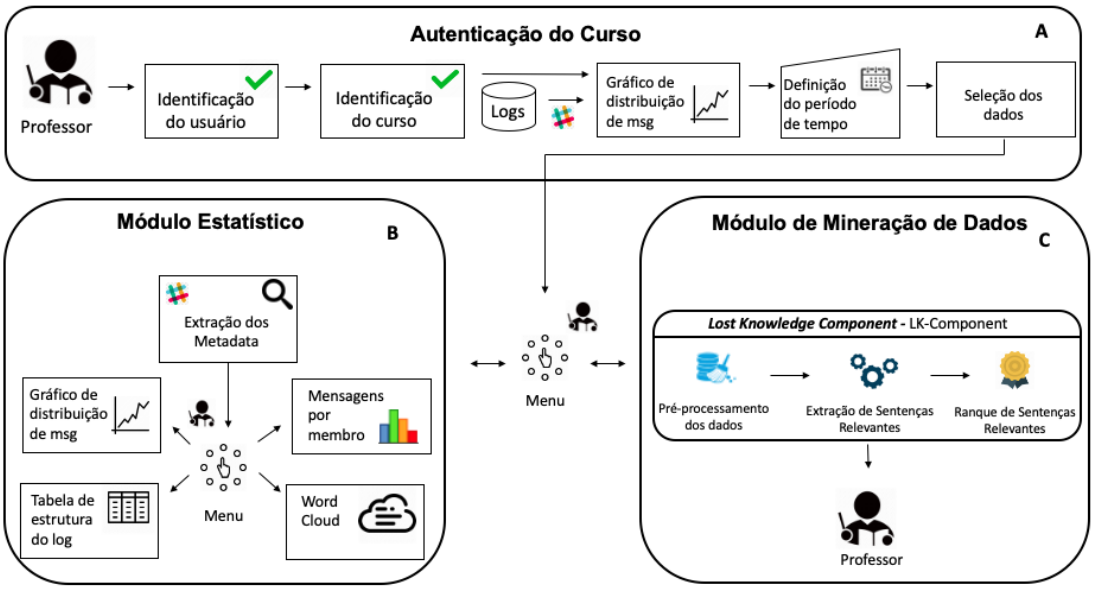

Figura 4.4: Ferramenta analítica para apoio educacional.

de um trabalho colaborativo e que o time deve interagir para distribuir tarefas, discutir ideias e tomar decisões, este recurso permite que os professores observem e analisem o perfil de interação da equipe em função do tempo. O número de mensagens trocadas por dia pode dar indício sobre a organização de trabalhos da equipe frente aos prazos de cada *sprint*. De posse dessa informação, os professores possuem indícios sobre o comportamento ativo ou procrastinador dos membros das equipes.

- **Estrutura do *Log*:** Uma tabela é utilizada para sumarizar a estrutura do *log*. Nela é resumido o conjunto de dados relacionados, a forma com que a equipe configurou e, consequentemente, organizou o meio pelo qual se comunicam. Os dados resumem informações acerca dos canais criados, como: os dias em que houve interação da equipe nesses canais, número de membros em cada canal e número de mensagens trocadas por canal, podem evidenciar a forma como as equipes estão gerenciando suas tarefas. De posse dessa informação, os professores podem refletir acerca do comportamento das equipes frente ao gerenciamento das discussões de planejamento, implementação e teste do produto desenvolvido.

- **Word Clouds:** Tem o objetivo de fornecer uma visão geral das discussões ocorridas. A ideia por trás da geração das nuvens de palavravas é prover uma representação visual de termos cuja relevância é medida através da frequência destes no texto e destacada através da formatação de apresenta (Seção 2.3). É útil resumir o conteúdo textual da comunicação dos membros das equipes. Os professores podem analisar, de forma mais objetiva, as dificuldades, dúvidas e sucesso das atividades realizadas e expressadas através das mensagens enviadas.

- **Mensagens por membro:** Tem o objetivo de contar e reportar o total de mensagens enviadas por cada participante de um canal previamente especificado. Este recurso dá indícios sobre o perfil do participante sob a perspectiva de interação individual deste com a equipe. Professores são providos de dados e podem detectar alunos com perfis de interação oposto aos demais membros. De posse dessas informações, os professores podem detectar e intervir em situações onde o comportamento de alguns membros das equipes possam prejudicar o desenvolvimento dos projetos. O recurso provê indícios para detecção de posturas ativas e passivas, desistências e baixa produtividade, revelando a presença deste no ambiente de comunicação.

2. Módulo de Mineração de Dados:

O módulo de mineração de dados (Figura 4.4 - C) tem o objetivo de analisar o conteúdo textual dos registros de mensagens dos alunos e detectar as discussões mais relevantes do ponto de vista dos professores. Este módulo foi projetado para executar o *LK-Component*, explicado na Seção 3.2. O objetivo é que o componente forneça dados que atendam as necessidades informacionais dos professores a fim de apoiá-los na realização de suas atividades pedagógicas de monitoramento das equipes e, consequentemente, ensino. O componente também foi configurado para retornar as 20 sentenças mais relevantes referentes aos períodos pré-selecionados.

4.3.2 Avaliação do *LK-Component* na educação

Um estudo foi realizado com o objetivo de investigar a utilidade dos registros de comunicação das equipes como fonte de informações relevantes para apoiar as

atividades pedagógicas de professores de graduação de computação e avaliar a utilização do *LK-Component* no contexto acadêmico.

Os *logs* analisados foram produzidos a partir da interação de três equipes: Arara, Boto e Onça. O SCRUM norteou os processos de gerenciamento e desenvolvimento dos projetos (Schwaber, 1997).

O estudo executado foi composto por três etapas:

1. Determinação da coleção de dados

Nesta etapa, os três professores definiram os períodos sobre os quais demonstravam interesse em analisar os dados dos *logs* de comunicação gerados (Figura 4.4 - A).

Cada equipe é composta por cinco membros. A interação de seus membros ocorreu por meio de reuniões presenciais (em sala de aula) e através da ferramenta Slack. Contudo, os metadados extraídos dos *logs* de comunicação das equipes registram a participação de seis membros em cada equipe, dos quais cinco correspondem aos alunos e um corresponde ao professor da disciplina Prática e Análise de Projetos de software. Os registros coletados datavam de 13 de setembro de 2018 a 17 de dezembro de 2018. Nesse período, a equipe Arara registrou 50 dias de interação e trocou 713 mensagens, a equipe Boto registrou 58 dias de comunicação e trocou 1.319 mensagens e a equipe Onça registrou 33 dias de conversa e trocou 421 mensagens. O total de troca de mensagens contabiliza as mensagens enviadas através de todos os canais de comunicação criados pelas equipes no *workspace* da ferramenta *Slack*. A Tabela 4.5 resume os números que caracterizam as amostras dos *logs* analisadas.

2. Utilização da ferramente

Nesta etapa, os professores foram solicitados a explorar os recursos providos pela ferramenta (Figura 4.4 - B e C). A ferramenta proposta caracteriza-se como um recurso adicional que visa auxiliar o monitoramento do desempenho das equipes através da geração de gráficos e tabelas estatísticas, e ainda, através da identificação de mensagens relevantes trocadas pelos membros.

3. Realização de entrevista de acompanhamento

Tabela 4.5: Caracterização dos *logs* analisados no estudo da educação.

	Arara	**Boto**	**Onça**
CARACTERIZAÇÃO DOS *LOGS*			
Total Dias de Interação Registrados	50	58	33
Total de Mensagens Trocadas	713	1.319	421
Total de Canais no *Workspace*	4	4	7
CARACTERIZAÇÃO DO CANAL *GENERAL*			
Total de participantes	6	6	6
Total de dias de interação registrados	50	58	33
Total de mensagens trocadas	635	1.149	307

Ao final, visando analisar a efetividade dos recursos providos, os três professores responderam um questionário de acompanhamento:

- Qual sua percepção sobre os recursos oferecidas pela ferramenta: são úteis para suas atividades de ensino?
- Você usaria a ferramenta para apoiá-lo em seus próximos cursos?

4.3.3 Resultados

Na primeira etapa do estudo, os professores determinaram o(s) período(s) de tempo sobre os quais foram feitas as análises das discussões. A Tabela 4.6 sumariza as amostras de dados selecionadas que compreendem entrada de dados para os módulos estatístico e de mineração de dados da ferramenta analítica.

Cada professor escolheu um total de 12 períodos, todos provenientes do canal *general* (conforme Tabela 4.5). A partir dos períodos selecionados é possível afirmar que o professor A analisou uma porção referente a 26%, 29,30% e 33,33% dos *logs* das equipes Arara, Boto e Onça, respectivamente. O professor B analisou aproximadamente 32%, 34% e 39% dos *logs* das equipes Arara, Boto e Onça. Já o professor C analisou cerca de 26%, 29% e 15% dos referidos *logs*, respectivamente.

Na segunda etapa do estudo, a fim de responder às questões de pesquisa levantadas neste estudo, os dois módulos que compunham a ferramenta analítica foram avaliados separadamente.

Tabela 4.6: Caracterização das amostras de dados analisadas no estudo da educação.

	Arara	**Boto**	**Onça**
CARACTERIZAÇÃO DAS AMOSTRAS			
Professor A			
Períodos Selecionados	4	4	4
Dias Analisados	13	17	11
%Dias Analisados (canal *general*)	26%	29,3%	33,33%
Total de sentenças	110	249	72
Professor B			
Períodos Selecionados	3	4	5
Dias Analisados	16	20	13
%Dias Analisados (canal *general*)	32%	34,48%	39,39%
Total de sentenças	186	281	93
Professor C			
Períodos Selecionados	4	4	4
Dias Analisados	13	17	5
%Dias Analisados	26%	29,31%	15,15%
Total de sentenças	140	154	23

Inicialmente, os metadados e as mensagens provenientes dos períodos selecionados foram usados para gerar gráficos estatísticos e evidenciar o engajamento dos membros das equipes. Para cada período P selecionado pelos três professores das disciplinas, o módulo estatístico foi usado para gerar os 4 recursos gráficos por ele providos, conforme Seção 4.3.1.1.

A Figura 4.5 apresenta o gráfico de distribuição de mensagens trocadas pela equipe Arara no período de 10/10/18 à 23/10/18. Para geração deste gráfico foram utilizados os metadados que armazenam a data de envio das mensagens e o canal usado para estabelecer a comunicação. Com o gráfico de distribuição de mensagens, os professores possuem evidências acerca da distribuição do esforço de trabalho da equipe ao longo do tempo.

A Figura 4.6 sumariza dados relacionados a forma como a equipe organizou o ambiente utilizado para a interação de seus membros. Uma tabela que evidencia a estrutura do *log* da equipe é criada. A geração deste recurso extraiu metadados providos pelos arquivos de configurações *channels.json* e pelos arquivos **.json*

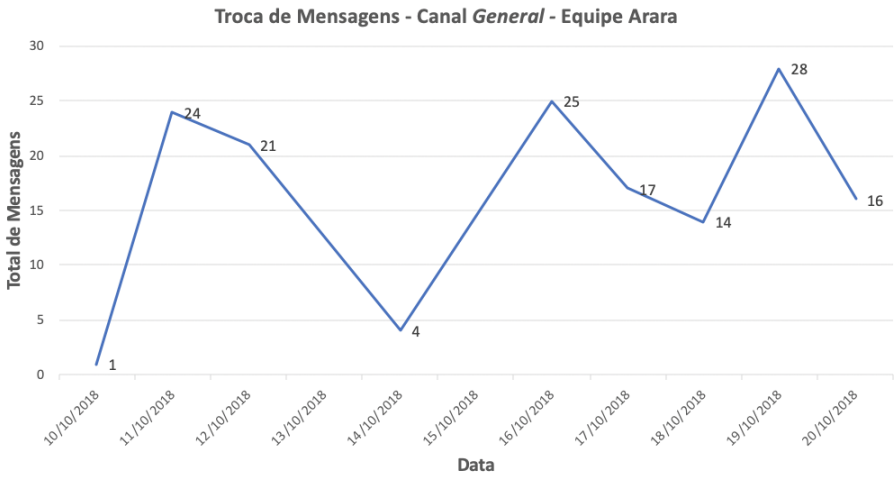

Figura 4.5: Gráfico de distribuição de mensagens: equipe Arara.

reportados em cada canal de interação criado pela equipe.

Organização do *log* - Equipe Onça					
Canais do Slack	**Total de membros**	**Dias de interação**	**Primeiro dia de interação**	**Último dia de interação**	**Total de mensagens enviadas**
random	6	4	2018-09-13	2018-10-05	9
importantdates	6	4	2018-09-13	2018-10-31	12
general	6	33	2018-09-13	2018-12-17	307
division-of-tasks	6	2	2018-09-14	2018-10-09	12
dailymeetings	5	19	2018-09-20	2018-12-12	53
git-repository	6	4	2018-09-24	2018-10-10	21
sprint-goals	6	1	2018-10-06	2018-10-06	7
Total					**421**

Figura 4.6: Tabela de estrutura do *log*: equipe Onça.

O terceiro recurso gera uma nuvem de palavras com o objetivo de sumarizar o conteúdo das mensagens enviadas pelos membros das equipes. A geração deste recurso consome dados oriundos do texto das mensagens. A geração das nuvens é baseada na frequência com que os termos aparecem no texto das mensagens e podem ser usadas como forma de sumarizar o conteúdo das mensagens.

O quarto recurso é o gráfico que demonstra o total de mensagens enviadas por cada membro da equipe, evidenciando o nível de interação dos membros através da ferramenta de comunicação adotada. A criação do gráfico ilustrado na Figura 4.7 usa os metadados que determinam o dia e o emissor das mensagens. É feito um

parser no arquivo do *log* com o objetivo de determinar a quantidade de mensagens enviadas por um determinado membro, em um período de tempo P, em um canal específico.

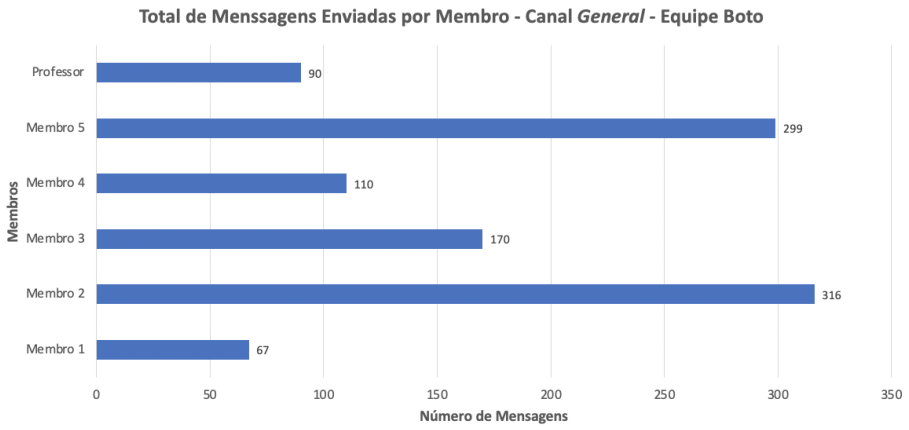

Figura 4.7: Gráfico de mensagens enviadas por membro: equipe Boto.

Após analisarem os gráficos e tabelas gerados pelo módulo estatístico, os professores descreveram suas respectivas percepções em relação às informações visualizadas. A seguir, são transcritos trechos das respostas dos mesmos:

> **Professor A:** - *"Os dados relatados em tabelas e gráficos cobriam aspectos de planejamento, desenvolvimento, releases do projeto e interação entre os membros das equipes."*.
>
> **Professor B:** - *"Gostei muito de ver os gráficos e tabelas antes de olhar as sentenças. Do ponto de vista de Gerência de Projetos, dá pra perceber se a equipe é organizada ou não pelo número de canais usados no Slack e pela quantidade de mensagens trocadas perto das datas de entrega de cada sprint. Dá pra verificar o engajamento de cada membro da equipe através da quantidade de interações de cada aluno em um período. Enfim, as informações estatísticas e gráficos são muito úteis. Claro que as informações são complementadas pela análise das sentenças relevantes."*.
>
> **Professor C:** - *"Os gráficos sozinhos não dão tanta informação sobre o empenho das equipes, mas ele se torna muito útil quando combinado com*

> *o texto das mensagens... As Word Clouds são úteis para identificar o teor principal das mensagens e os aspectos mais desafiadores para a equipe... Porém, ficou claro para mim a importância da análise das mensagens para o acompanhamento e avaliação dos alunos.".*

Analisando as citações dos professores, é possível destacar que: (1) sob a perspectiva dos três professores, o módulo estatístico foi capaz de prover informações revelantes acerca do comportamento e comprometimento das equipes; (2) conforme o Professor B, os recursos estatísticos podem ser analisados a fim de evidenciar a organização das equipes; (3) os recursos estatísticos providos oportunizam um monitoramento raso das equipes; e (4) conforme os professores B e C, é necessário que as informações providas pelo módulo estatístico sejam enriquecidas pelas informações do módulo de mineração de dados a fim de proporcionar melhores formas de monitoramento e oportunidades de intervenção.

Portanto, respondendo à QP1 deste estudo: Pode-se usar os metadados das ferramentas de comunicação para analisar e compreender o padrão de comportamento das equipes de estudantes?

Sim. O estudo realizado mostra que os gráficos estatísticos baseados nos metadados do Slack e na frequência do termo do texto são úteis para analisar o comportamento das equipes. No entanto, eles forneceram conhecimento superficial sobre as atividades das equipes.

Em seguida, a eficácia do *LK-component* foi avaliada com o intuito de identificar e recuperar frases relevantes dos registros de comunicação dos alunos. Para tanto, foi solicitando que os três professores classifiquem a relevância das 20 principais sentenças identificadas pelo *LK-component* de acordo com sua utilidade nas atividades pedagógicas.

O *LK-Component* calculou a pontuação de relevância de 1.308 sentenças. Um total de 564 sentenças relevantes foram identificadas, extraídas e analisadas. No total, o Professor A analisou a relevância de 186 sentenças, o Professor B analisou a relevância de 202 sentenças e o Professor C analisou 176 sentenças.

Conforme ilustra a Tabela 4.7, o Professor A destacou que as três principais sentenças referentes aos *logs* de cada equipe analisada eram úteis para suas

tarefas pedagógicas, atingindo valores de precisão de $P@3_{Arara} = 91\%$, $P@3_{Boto} = 79\%$ e $P@3_{Onça} = 100\%$. Considerando as dez sentenças do topo do ranque, a avaliação do Professor A determinou que 79%, 80% e 82% das frases analisadas foram consideradas relevantes de acordo com cada equipe. O Professor A também classificou todas as sentenças do topo do ranque relevantes, a medida MRR alcançou seu melhor valor ($MRR = 1$) ao analisar os *logs* de todas as equipes. Do ponto de vista do Professor B, 88% das três principais frases identificadas da equipe Arara eram relevantes ($P@3_{Arara} = 88\%$). Por outro lado, ele classificou apenas 50% das três principais frases da equipe Boto como relevantes. Do ponto de vista dos Professores B, nem todas as sentenças classificadas em primeiro lugar eram relevantes, como se pode notar pelos valores de MRR (0,62 para a equipe Boto e 0,80 para a equipe Onça). O Professor C destacou que 83% das 3 principais sentenças e 87% das 10 melhores da equipe Arara eram relevantes. Do ponto de vista do Professor C, a abordagem alcançou a melhor eficiência na avaliação da equipe Onça, onde os valores de MRR, $p@3$ e $p@5$ alcançaram seus melhores valores, revelando que todas as cinco sentenças do topo do ranque foram consideradas relevantes de acordo com as perspectivas dos professores. Pode-se notar que, em alguns casos, os valores de precisão da avaliação dos professores diferem. Esse fenômeno se deve ao fato de o julgamento da relevância envolver imprecisão humana, uma vez que o conceito de relevância depende da perspectiva dos avaliadores.

Tabela 4.7: Métricas de avaliação do *LK-Component* aplicado à educação

	Equipe	**Sentenças Relevantes**	**MRR**	**P@3**	**P@5**	**P@10**
Professor A	Arara	68	1	91%	85%	79%
	Boto	62	1	79%	77%	80%
	Onça	56	1	100%	90%	82%
Professor B	Arara	60	1	88%	80%	73%
	Boto	65	0.62	50%	50%	47%
	Onça	77	0.80	73%	65%	53%
Professor C	Arara	73	0.87	83%	90%	87%
	Boto	80	1	91%	95%	92%
	Onça	23	1	100%	100%	80%
	Média			83%	81%	74%
TOTAL		564				

Os professores também descreveram suas respectivas percepções em relação às informações reveladas pelo módulo de mineração de dados. A seguir, são

transcritas as respostas dos mesmos:

> **Professor A:** - *"As sentenças relevantes ajudam o professor a se concentrar nos elementos que são importantes no processo de ensino-aprendizagem. Neste curso, os alunos devem entregar um software a um cliente real, e frases relevantes ajudam a reduzir o risco de uma falha na entrega."*
>
> **Professor B:** - *"As sentenças relevantes extraídas do registro de comunicação dos alunos foram essenciais para identificar oportunidades de intervenção. Como professor da matéria de Gerência de Projetos, as discussões relevantes foram úteis para avaliar o desempenho do Scrum Master."*
>
> **Professor C:** - *"As sentenças relevantes são úteis para revelar o nível de engajamento das equipes. Pude identificar problemas de comunicação e falhas técnicas. As sentenças são úteis para me ajudar a melhorar o feedback que dou aos alunos no final de cada sprint. Eu pude apontar os aspectos positivos e negativos relacionados a cada equipe. Através das sentenças, também pude perceber que os alunos da equipe Boto tiveram muita dificuldade técnica e na última sprint houve muita procrastinação. Alguns alunos se destacaram mais, e por isso usaram bastante o sistema de mensagens para cobrar e estimular o empenho dos demais. As sentenças deixam claro que uma única nota para todos os membros do grupo pode ser injusto com os alunos que fizeram muito esforço individual."*

Analisando a resposta dos professores, é possível notar que a percepção da eficácia das informações providas pelo módulo de mineração de dados no apoio às atividades pedagógicas dos professores foi unânime.

Portanto, respondendo à QP2: Pode-se usar técnicas de mineração de dados para identificar e extrair sentenças relevantes, sob a perspectiva do professor, dos registros de comunicação das equipes de estudantes?

Sim. O algoritmo implementado pelo *LK-Component* identificou e extraiu sentenças relevantes a partir dos *logs* de comunicação das

equipes. Sentenças úteis para apoiar as atividades pedagógicas dos professores.

Na terceira etapa do estudo foram realizadas entrevista de acompanhamento com os professores envolvidos. A entrevista visava investigar as percepções dos professores a respeito da utilidade da ferramenta proposta e avaliar o interesse destes na utilização da ferramenta analítica em outros cursos. A seguir são apresentados trechos das respostas obtidas:

1. **Qual sua percepção sobre os recursos oferecidas pela ferramenta: são úteis para suas atividades de ensino?**

   Professor A: - *"Sim. A ferramenta me ajudou a identificar os pontos mais relevantes e os pontos onde eu deveria agir para apoiar a experiência positiva dos alunos durante o curso. Ele permitiu identificar quais tópicos precisam de um "reforço" de nossa parte, com objetivo de realizar a entrega dos projetos produzidos pelos alunos. A ferramenta ajudou na identificação de barreiras que impossibilitam os alunos de avançar no conhecimento prático e que podem ser barreiras que eles encontrarão na indústria."*

   Professor B: - *"Sim. É bastante útil uma vez que é possível ter um panorama do trabalho das equipes, suas dificuldades (técnicas, gerenciais ou de relacionamento) e seu engajamento. É possível verificar quem está de fato envolvido no desenvolvimento do projeto. A ferramenta possibilita identificar momentos de intervenção do professor durante o curso. Do ponto de vista de Gerenciamento de Projetos, que é a minha disciplina, a ferramenta me permite avaliar o Scrum Master de cada equipe e a forma de como ele está gerenciando o desenvolvimento do projeto como um todo."*

   Professor C: - *"Sim!! A ferramenta me fornece informações sobre a cooperação dos alunos no projeto. Analisando os dados relatados, eu pude projetar novas atividades em sala de aula para esclarecer as dúvidas técnicas dos alunos. A análise dos recursos providos pela ferramenta demonstrou ser importante, pois me mostraram que alguns alunos procrastinaram e no final tiveram que abandonar as regras de trabalho estipuladas pelos professores."*

2. **Você usaria a ferramenta para apoiá-lo em seus próximos cursos?**

> **Professor A:** - *"Sim. Claro! Talvez a ferramenta possa incluir frases classificadas por sentimentos, as postagens relevantes com sentimento negativo podem ser priorizadas e tratadas pelos professores para criar experiências positivas durante o curso."*
>
> **Professor B:** - *"Sim! Eu usaria com certeza. Uma dificuldade que se tinha era a de acompanhar de fato o andamento das atividades uma vez que as reuniões presenciais nem sempre aconteciam no horário da minha disciplina. Como o curso envolveu 3 disciplinas, coordenar as atividades é uma tarefa difícil e pode ser minimizada através do uso da ferramenta."*
>
> **Professor C:** -*"Sim!! A ferramenta proposta é essencial para avaliar e acompanhar os alunos."*

Analisando as respostas dos professores, observa-se que todos identificaram interesse em adotar a ferramenta analítica proposta em seus futuros cursos. Além disso, os professores destacam o uso da ferramenta como uma oportunidade para identificar barreiras técnicas e sociais que possam prejudicar o desenvolvimento dos projetos realizados pelas equipes, oferecendo oportunidades para identificação de pontos de intervenção.

4.3.4 Discussões

O estudo realizado revelou o uso promissor do *LK-Component* como módulo integrante da ferramenta de análise de dados desenvolvida. Os resultados do estudo sugerem que as informações de metadados e a frequência dos termos de texto providos pela ferramenta Slack podem ser usadas para criar relatórios gráficos estatísticos e revelar a estrutura organizacional das equipes, os padrões de interação entre os membros da equipe e sumarizar as discussões dos times. No entanto, os professores indicaram que os gráficos estatísticos gerados fornecem conhecimento superficial sobre as atividades das equipes e que sozinhos não oferecem informações suficientes para apoiar as suas respectivas atividades pedagógicas.

Os resultados também denotaram um uso positivo do *LK-Component* para identificar e extrair frases relevantes do registro de comunicação das equipes feitos através da ferramenta Slack. A análise quantitativa indica que o módulo

de mineração de dados alcançou uma precisão média de 83%, 81% e 74% na identificação das sentenças relevantes ranqueadas entre as 3, 5 e 10 principais nos *logs*, respectivamente. Valores que apontam a necessidade de aprimoramento da estratégia adotada a fim de aumentar a precisão das frases relevantes extraídas.

Sob a perspectiva de avaliação dos professores, estes destacaram que os gráficos estatísticos e os relatórios de frases relevantes são recursos que se complementam, fornecendo uma visão geral do progresso individual dos alunos e das equipes formadas. No entanto, as citações dos professores revelaram interesse em explorar outros tipos de informações que podem ser mineradas a partir dos *logs* de comunicação das equipes, como as informações oriundas da execução de análises de sentimento.

Dado que estudantes podem ser comparados aos desenvolvedores iniciantes (Salman et al., 2015), este estudo revelou a efetividade de uso do *LK-Component* na análise de *logs* de comunicação de equipes de desenvolvimento que fazem uso da ferramenta Slack para apoiar a comunicação entre membros de equipes de desenvolvimento. Também revelou que o uso de metadados para computar a frequência de mensagens enviadas ou frequência de interações entre os participantes trouxe um conhecimento superficial do padrão de comportamento das equipes, sendo necessária a adoção de novas estratégias para obtenção de resultados mais precisos.

4.4 Ameaças à Validade

Dado que foram analisados dados de dois projetos, não é possível garantir a generalização dos resultados obtidos. Além disso, o processo de categorização das discussões por tipo foi realizada manualmente por dois avaliadores. Porém, com o objetivo de minimizar possíveis vieses introduzidos durante o processo, foi mensurado o nível de concordância entre os avaliadores.

Ademais, este estudo analisou apenas uma vertente da troca de conhecimento, via IM, e não procurou ver se algum desse conhecimento poderia ser capturado em outras mídias, como *e-mail*, Slack, comentários de código, documento de *backlog* ou relatórios de *daily*.

Por fim, o algoritmo de *ranking* utilizado se baseia no valor de $TF \times IDF$.

Contudo, $TF \times IDF$ é uma métrica sensível a erros ortográficos. Como o estudo trata-se de uma investigação para entender se os projetos perdem conhecimento e se estes podem ser identificados usando uma abordagem semiautomática, acredita-se que os resultados reportados não são impactados por esta característica da métrica.

4.5 Considerações

Este capítulo apresentou o *LK-Component*, um dos três componente do *framework Miner4DevTeam* aqui construído. O *LK-Component* tem por objetivo apoiar a determinação do *Project Lost Knowledge* (Seção 3). Neste componente, o problema de detecção de decisões e/ou discussões de projetos a partir dos registros de comunicação de times de desenvolvimento foi modelado como um problema de ranque (*ranking*). Para tanto, foram usadas técnicas de PLN e algoritmo de *text summarization*. O *LK-Component* é uma abordagem parametrizável na qual o usuário consumidor dos resultados gerados deve definir (1) a janela de tempo que delimita o conjunto de dados a ser analisado e (2) o valor de N usado na determinação das N sentenças mais relevantes registradas nos *logs* de comunicação das equipes. O componente foi avaliado tanto no contexto da indústria de software quanto na academia.

Estudos feitos na indústria indicam que os projetos perdem conhecimento à medida que decisões, raciocínios e justificativas são discutidas em ferramentas de comunicação colaborativa. Gerentes de projetos podem usar o *LK-Component* para identificar PLK do projeto. O componente pode apoiar a identificação e reuso de conhecimentos em *startups* e outras empresas de software que usam ferramentas de bate-papo como canal de comunicação. Em média, os gerentes dos projetos levaram 22 minutos para avaliar as sentenças de cada período selecionado e identificar o PLK. A avaliação do componente atingiu uma precisão de 75%. Por fim, o PLK pode ser utilizado para apoiar a evolução dos produtos de software e também para apoiar o processo de tomada de decisão estratégica nas empresas de software nas quais desenvolvedores usam ferramentas de mensagens instantâneas para intermediar a comunicação do time.

Estudos na academia mostram que o *LK-Component* pode ser usado para

apoiar as atividades pedagógicas dos professores. Foi investigado se os registros de comunicação de alunos, matriculados em um curso interdisciplinar, feitos através da ferramenta Slack, são uma importante fonte de evidência para monitorar o progresso das equipes de desenvolvimento. Observou-se que o componente foi capaz de recuperar informações relevantes, dos registros de discussões das equipes, que podem ser usadas pelos professores para monitorar e identificar oportunidades de intervenção em sala de aula.

Contudo, os resultados apresentados indicam a necessidade de uma pesquisa extensa para entender como o conhecimento de projetos de software é perdido em outros canais de comunicação usados por equipes de desenvolvimento, como IRC, lista de discussão, fórum de desenvolvedores e outros. Além disso, esta pesquisa também oportuniza direcionamentos futuros para estudos de outros tipos de conhecimento de projeto de software que podem ser extraídos dos registros de comunicação dos times de desenvolvimento.

Capítulo 5

Minerando *Project Frequent Knowledge*

Este capítulo apresenta discussões relacionadas a aplicação e avaliação do *Frequent Knowledge Component* (*FK-Component*). Um estudo realizado com profissionais da indústria de software revelou resultados promissores ao utilizar conhecimentos de projeto extraídos pelo componente na integração de novos membros às equipes de desenvolvimento de software.

O capítulo está organizado da seguinte forma: a Seção 5.1 apresenta o *visual abstract* da pesquisa realizada com objetivo de definir e validar o *FK-Component*. A Seção 5.2 descreve o contexto de aplicação do *FK-Component*. A Subseção 5.2.1 descreve o processo de determinação do PFK. A Subseção 5.2.2 descreve o estudo realizado com objetivo de avaliar a eficácia do *FK-Component*. As Subseções 5.2.3 e 5.2.4 apresentam os resultados obtidos e as discussões referentes ao estudo realizado, respectivamente. Por fim, as Seções 5.3 e 5.4 apresentam as ameaças à validade do estudo realizado e as considerações finais do capítulo, respectivamente.

5.1 *FK-Component - Visual Abstract*

O *FK-Component* é resultado do segundo ciclo de *design* realizado nesta pesquisa. Para apresentar um resumo dos principais pontos de idealização, desenvolvimento e validação do componente, a Figura 5.1 apresenta o *Visual Abstract* (Storey

et al., 2017) da pesquisa que descreve o *FK-Component* (Lima et al., 2020). Nela são apresentados o objetivo, o contexto, a relevância, o rigor e a novidade do componente.

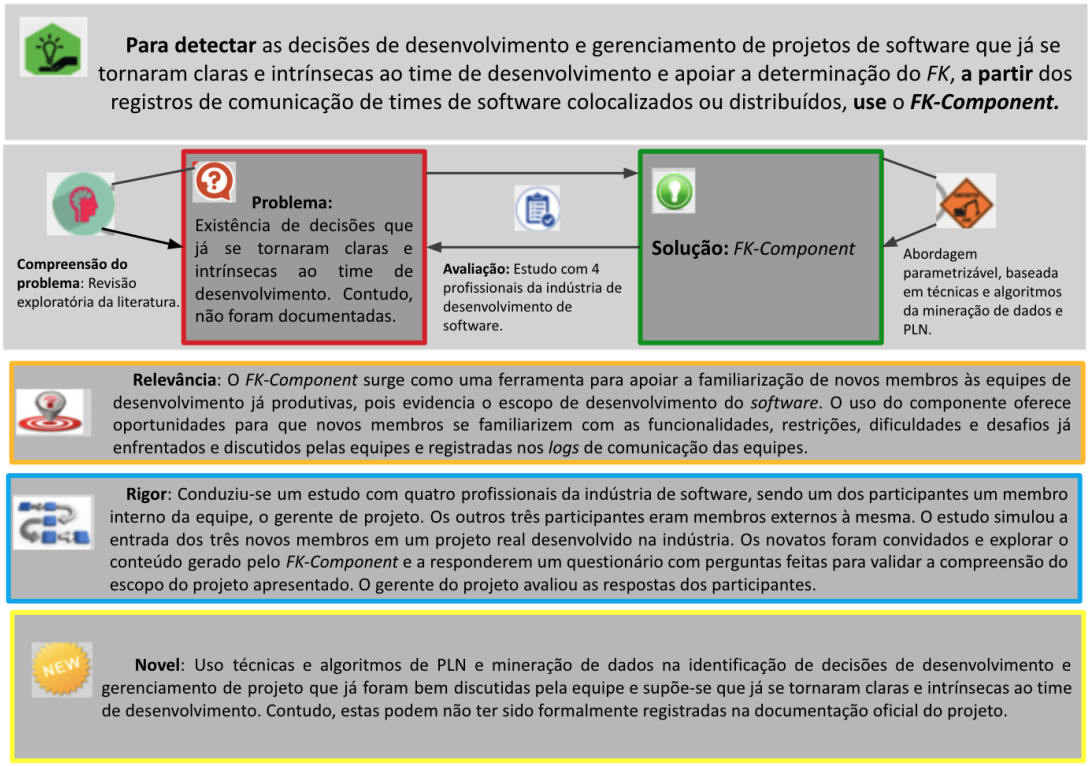

Figura 5.1: *Visual Abstract* do desenvolvimento do *FK-Compoenent*.

As principais contribuições deste segundo ciclo de *design* são:

- O desenvolvimento e avaliação de uma abordagem semi-automática, baseada em PLN e técnicas de mineração de dados para apoiar a identificação de decisões de projetos que não foram formalmente registradas na documentação do software e que podem ser usadas para apoiar a entrada de novos membros às equipes de desenvolvimento.
- Evidências experimentais sobre o uso de ferramentas de bate-papo (*chats*) nas equipes de desenvolvimento.
- Evidências experimentais sobre as aplicações práticas do *FK-Component* em empresas que recorrem a tais ferramentas de comunicação.

Direcionamentos futuros apontam para a necessidade de aperfeiçoamento do componente através da otimização das técnicas de PLN utilizadas, da combinação de diferentes fontes de dados (como o *backlog* e o *sprint planning*) e a exploração de recursos como *emojis*, trechos de códigos e metadados.

5.2 *FK-Component* aplicado à Indústria

Conforme descrito na Seção 3.3, o *FK-Component* visa apoiar a identificação de conhecimentos de projetos de software baseando-se em discussões frequentes feitas pelos times de desenvolvimento por meio de ferramentas de comunicação síncronas, determinando o que aqui é chamado de *Project Frequent Knowledge* (Seção 3.1). Para tanto, o *FK-Component* determina os assuntos frequentemente discutidos e expõe trechos de mensagens associados a tais assuntos. Tal estratégia é usada para evidenciar o escopo de desenvolvimento dos projetos, oferecendo oportunidades para que novos membros se familiarizem com as funcionalidades, restrições, dificuldades e desafios já enfrentados e discutidos pelas equipes.

Sendo assim, as seguintes questões de pesquisa (QP) guiam a investigação de validade do *FK-Component* aqui proposto:

- **QP1:** A identificação e a extração dos assuntos frequentemente discutidos nos registros de comunicação de desenvolvedores auxiliam na determinação do escopo de desenvolvimento de projetos de software?
- **QP2:** Novatos conseguem compreender o escopo de desenvolvimento do projeto através da análise das mensagens associadas aos assuntos frequentemente discutidos pela equipe?

Para avaliar o componente, conduziu-se um estudo com quatro profissionais da indústria de desenvolvimento de software. No contexto do projeto de software analisado, um dos participantes do estudo era membro interno da equipe, o gerente de projeto, os outros três eram membros externos à mesma. Os resultados revelaram que o componente pode extrair o PFK presente nos *logs* de comunicação das equipes e o conhecimento identificado é útil para a integração de novos membros às equipes de desenvolvimento. Durante o estudo, os participantes enfatizaram

que o processo de identificação do *Project Frequent Knowledge* também fornece oportunidade para transferência de conhecimentos técnicos, sociais e organizacionais e, ainda, que a utilidade do PFK não se limita aos novatos, membros antigos podem usá-lo para recordar decisões já tomadas pela equipe. A metodologia, os resultados e as discussões geradas a partir deste estudo foram publicados em (Lima et al., 2020).

5.2.1 Determinação do PFK

A determinação do PFK é um processo no qual o *FK-Component* é usado para identificar discussões frequentes, relacionadas aos projetos de software, feitas por equipes de desenvolvimento através do uso de ferramentas de mensagens instantâneas. A Figura 5.2 identifica as etapas da abordagem desenvolvida para permitir a abstração, sumarização, externalização e utilização do PFK. Tanto o gerente de projeto, conhecedor do software desenvolvido, quanto os novatos, potenciais beneficiários do PFK extraído, são atores que interagem ativamente na abordagem.

As seguintes etapas descrevem o processo apresentado na Figura 5.2:

1. Coleta de Dados do Projeto

Com o objetivo de prover informações que evidenciem a evolução temporal do projeto analisado, o gerente de projeto deve fornecer o *log* de comunicação da equipe e definir uma data limítrofe que delimite o passado remoto e o passado recente do projeto de software (Figura 5.2 - A). A divisão em dois períodos fornece uma abstração da evolução do projeto analisado, sendo útil para melhor revelar o escopo de desenvolvimento deste. Uma análise mais profunda das discussões recentes tende a ser mais produtiva, já que pode revelar funcionalidades, restrições e desafios técnicos sobre os quais a equipe está trabalhando ou irá trabalhar. O passado remoto revela discussões ocorridas acerca de decisões e problemas que, provavelmente, já foram implementados e/ou mitigados e que podem evidenciar lições aprendidas pela equipe. As mensagens referentes aos períodos identificados são selecionadas e constituem a entrada de dados para a etapa de extração do PFK.

2. Extração do PFK

A extração do PFK (Figura 5.2 - B) é realizada através da determinação dos assuntos frequentemente discutidos em cada período analisado (passado remoto e passado recente). O PFK é utilizado para evidenciar o escopo do projeto em desenvolvimento. Para tanto, as mensagens selecionadas na etapa anterior são submetidas ao *FK-Component*.

Como saída, o componente gera *word clouds* que sumarizam e externalizam o PFK identificado. Para cada período, definidos na etapa de coleta de dados do projeto, são geras duas *word clouds*: a Semântica e a Técnica. Conforme Seção 3.3, a *word cloud* semântica é utilizada para evidenciar características de especificação, planejamento, desenvolvimento e manutenção dos projetos de software, já a *word cloud* técnica evidencia características relacionadas ao uso de tecnologias e decisões técnicas no desenvolvimento dos projetos de software.

3. Apresentação do PFK

Nesta etapa, são apresentados os assuntos frequentemente discutidos pelas equipes e as mensagens a eles associadas (Figura 5.2 - C). As *word clouds* são utilizadas como recurso de busca, através das quais os novatos podem localizar trechos de discussões acerca de um assunto específico frequentemente discutido pela equipe (Figura 5.3). Cada trecho de discussão inclui a mensagem que contém o assunto alvo de interesse do novato e, visando expandir o contexto da mensagem alvo, também são exibidas três mensagens anteriores e três mensagens posteriores à mensagem alvo. O número de mensagens usadas para mostrar o contexto da mensagem alvo é configurável.

Ao iterarem pelos termos das *word clouds* os novatos são apresentados a trechos de discussões que revelam as decisões e discussões frequentes da equipe e, ainda, revelam o raciocínio acerca das decisões de projeto tomadas, expressando o conhecimento do projeto analisado. Uma ferramenta *web* foi gerada para facilitar a apresentação do PFK extraído.

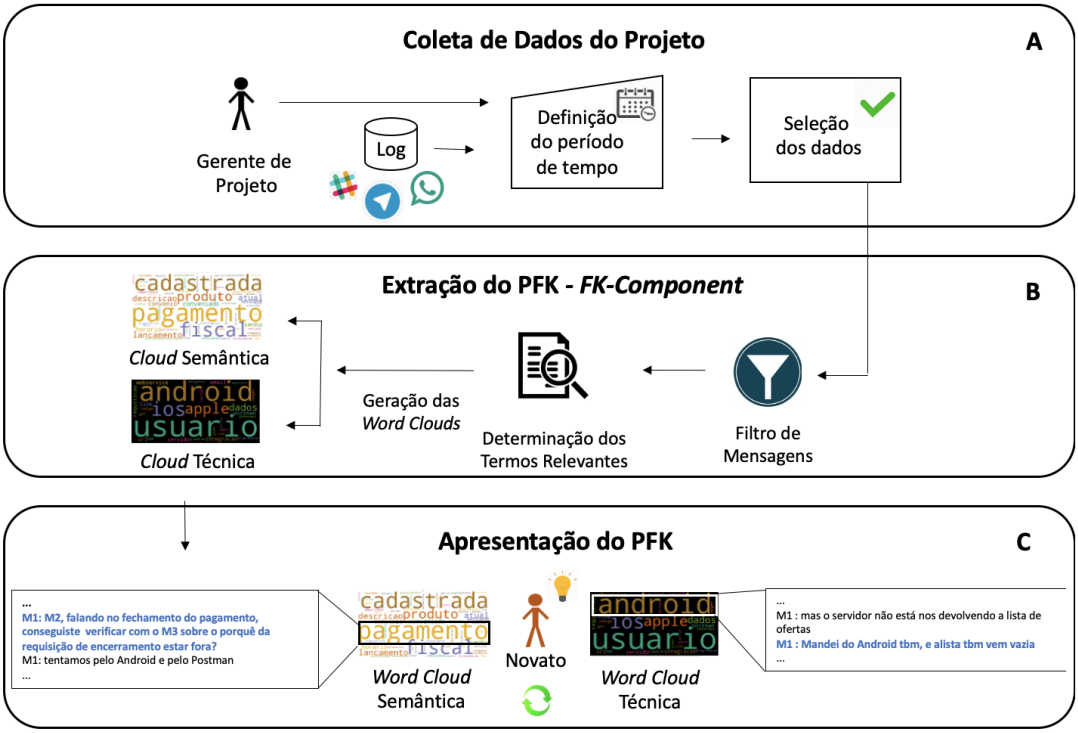

Figura 5.2: Processo de determinação do *Project Frequent Knowledge*.

5.2.2 Avaliação do *FK-Component*

Um estudo realizado visou avaliar a efetividade do *FK-Component* usado na abordagem e, ainda, avaliar a compreensão do escopo de desenvolvimento do projeto de software por novos membros que ingressam em uma equipe cujas atividades já estão em andamento. Para tanto, o estudo simulou a entrada de três novos membros (Novato1, Novato2 e Novato3) em um projeto desenvolvido na indústria. Os três novatos são profissionais da indústria de software e possuem diferentes perfis de atuação no processo de desenvolvimento. Novato1 é testador e possui 4 anos de experiência na área, Novato2 é programador com 4 anos de experiência e Novato3 é engenheiro de requisitos com experiência de 9 anos.

Os dados analisados referem-se a um projeto real, tendo sido fornecidos por uma *startup* que atua no mercado de desenvolvimento de software há seis anos e adota o *Scrum* como processo de desenvolvimento. Embora não fosse um time distribuído, alguns membros do projeto possuíam horário de trabalho diferenciado.

Sendo assim, a equipe utilizava a ferramenta WhatsApp para apoiar o processo de desenvolvimento e gerenciamento das atividades. O *log* foi obtido através do recurso de exportação de dados fornecido pela própria ferramenta. Áudios e imagens foram filtrados.

O projeto analisado foi indicado pelo gerente. A equipe era composta por um *Project Owner*, dois *Scrum Master*, um *designer* de interface, dois desenvolvedores e um testador. Como projeto, a equipe desenvolveu um software de comércio eletrônico para dispositivos móveis. Por questões de sigilo contratual e visando apoiar o estreitamento entre a indústria e a academia, o gerente optou por analisar o *log* de um projeto já concluído. Sendo assim, uma amostra do *log* referente a 10 meses de interação da equipe foi selecionada, momento em que algumas funcionalidades do projeto já estavam implementadas, porém, não concluídas. As mensagens dos 8 primeiros meses foram usadas para descrever o passado remoto do projeto e as mensagens dos 2 últimos meses descrevem o passado recente.

O estudo executado foi composto por quatro etapas:

1. Caracterização do projeto e treino de utilização da ferramenta

Nesta etapa, foi apresentada aos novatos uma visão geral do projeto no qual estes seriam alocados, assim como, o objetivo e o funcionamento da ferramenta *web* criada para permitir a navegação do novato pelos termos que determinam os assuntos frequentemente discutidos pela equipe. Esta etapa simula as primeiras horas de trabalho de um novato em um projeto. Momento em que, geralmente, um membro da equipe é designado para apresentar o objetivo do projeto em andamento e fornecer informações gerais do *status* de trabalho do time.

2. Exploração dos dados

Nesta etapa, foi solicitado que os novatos explorassem, individualmente, as *word clouds* apresentadas e, ainda, navegassem pelo conteúdo do *log* através do recurso de busca embutido nas *clouds*. O objetivo desta etapa é permitir que os novos membros entrem em contato com o conteúdo das mensagens enviadas pela equipe e registradas no *log* de comunicação, familiarizando-se com as discussões e decisões de projeto tomadas através da ferramenta utilizada.

3. Determinação do conhecimento adquirido

Nesta etapa, os novatos preencheram um questionário contendo cinco perguntas. As perguntas foram feitas com objetivo de validar a compreensão do escopo do projeto apresentado e a utilidade do componente desenvolvido. Os novatos foram informados que suas respostas deveriam apresentar suas percepções em relação conteúdo analisado e deveriam apontar o que cada um aprendeu sobre os itens questionados. Sendo assim, após leitura do conteúdo das *word clouds*, solicitou-se aos participantes do estudo que respondessem as seguintes perguntas:

1. Quais funcionalidades do sistema são discutidas?
2. Quais restrições do sistema são discutidas?
3. Quais dificuldades técnicas são discutidas?
4. Quais desafios são discutidos?
5. Na sua visão, a externalização do PFK lhe ajudou a compreender melhor o projeto no qual você vai trabalhar?

4. Avaliação do componente

A compreensão do PFK exige que os novos membros das equipes analisem semanticamente o conteúdo dos dados apresentados, contudo requer apenas um conhecimento geral prévio sobre o projeto analisado. Não envolve o julgamento humano, uma vez que a relevância dos termos analisados baseia-se na frequência destes, porém envolve a imprecisão de compreensão de um novo membro. Para lidar com tal cenário e validar o componente desenvolvido, o gerente de projeto do sistema analisado avaliou pontos do processo em dois momentos. Primeiramente, o gerente classificou os 20 primeiros termos usados para compor cada *word cloud* gerada como sendo relevantes ou não-relevantes para descrever os conteúdos semântico e técnico de cada período do *log* analisado. Em um segundo momento, o gerente analisou o conteúdo das respostas dos questionários preenchidos pelos participantes.

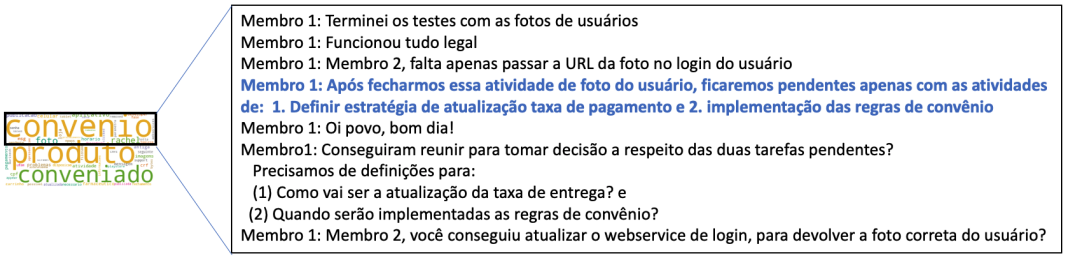

Figura 5.3: Recurso de busca embutido nas *Word Clouds*.

5.2.3 Resultados

A primeira etapa do estudo conduzido foi realizada individualmente com cada participante. Em média, cada sessão durou 15 minutos. As instruções descritas na etapa de caracterização do projeto e treino de utilização da ferramenta (Seção 5.2.2) foram executadas.

Para gerar as *word clouds* utilizadas na segunda etapa do estudo, a amostra do *log*, referente aos 10 meses de interação da equipe, foi analisada. A amostra era composta 896 mensagens, das quais 750 descreviam o passado remoto e 146 descreviam o passado recente do projeto. A Figura 5.4 ilustra as *word clouds* semânticas e técnicas geradas para revelar os assuntos, frequentemente discutidos pela equipe, referentes ao passado recente e remoto do projeto analisado. Já a Tabela 5.1 lista os 20 assuntos mais discutidos pela equipe de desenvolvimento usados na geração das *word clouds*. No total, foram computados os índices de relevância de 5.846 palavras candidatas a assuntos frequentemente discutidos, sendo 5.032 palavras referentes ao passado remoto e 814 palavras relativas ao passado recente do projeto. Os participantes Novato1, Novato2 e Novato3 relataram que alocaram 40min, 1h15min e 1h40min nesta etapa, respectivamente.

Na terceira etapa do estudo, os participantes responderam o questionário solicitado. Por questões de brevidade, a seguir são apresentadas as perguntas e trechos das respostas.

1. **Quais funcionalidades do sistema são discutidas?**

   Novato1: *"Produtos podem ser consultados... inserir/visualizar produtos no carrinho de compras... realizar pagamento, cadastrar usuários conve-*

Figura 5.4: *Word Clouds* Semânticas e Técnicas.

niados e não conveniados. Os usuários podem fazer login no app.".

Novato2: *"Mostrar economia na tela do produto. Ao comprar um produto o usuário pode ver o quanto está economizando.... Desconto de produto conforme convênio do usuário. Cada usuário pode ter descontos diferentes, isso muda conforme o convênio em que ele está vinculado... O sistema deve permitir que os usuários ligados a um convênio cadastrem seu CPF ou número do cartão do dependente... Cadastro de 2 tipos de usuários (com convênio e sem)".*

Novato3: *"Login, cadastro de usuários, farmacêuticos e produtos, esqueci minha senha, consulta e detalhes dos produtos, carrinho de compras... pagamento para usuários com convênio e sem convênio."*

Os participantes expressam visões complementares a respeito do sistema analisado, apresentando funcionalidades discutidas pela equipe. Neste caso, percebe-se que os participantes puderam tomar conhecimento sobre as funcionalidades do sistema registradas na amostra do *log* analisado. A aquisição de um conhecimento amplo e profundo do desenvolvimento de um software demanda vivência com o projeto. Neste caso, os participantes alocaram, em média, 1h11min na análise das

Tabela 5.1: Assuntos frequentemente discutidos.

Word Clouds	**Assuntos Frequentes**
Semântica Passado Recente	Pagamento, cadastrada, fiscal, produto, lançamento, descrição, atual, horário, conveniado, convênio, senha, mensagem, importante, individual, company, dólares, certinho, novidade, preparativos, reconfigurar
Semântica Passado Remoto	Convênio, produto, conveniado, foto, Rachel, aplicativo, publicação, celular, msg, horário, cpf, pagamento, atividade, crf, problema, loja, farmacêutico, antiga, mensagem, carrinho.
Técnica Passado Recente	Usuário, Android, iOS, Apple, dados, integração, webservice, servidor, postman, URL, requisição, link, e-mail, app, amigável, imagem, get, cartão, error, post
Técnica Passado Remoto	Imagem, app, play, usuário, dados, webservice, login, *store*, integração, jpg, serviço, valor, e-mail, JSON, Android, parâmetros, arquitetura, validação, cartão, padrão.

word clouds e suas citações, posteriormente analisadas pelo gerente de projeto da equipe, revelam que todos adquiriram conhecimentos sobre as funcionalidades do projeto.

2. **Quais restrições do sistema são discutidas?**

   Novato1: *"Usuários conveniados e não conveniados consultam/pagam produtos com diferentes valores."*.

   Novato2: *"O saldo para compras do titular e dependentes deve ser compartilhado e não individual... As condições de pagamento só devem ser apresentadas caso o usuário esteja logado."*.

   Novato3: *"Usuário com convênio tem desconto na compra Usuário pode consultar preços sem logar Caso usuário finalizar a compra sem logar, não terá desconto do convênio. Caso logue com o convênio, o usuário terá parte do valor pago pelo convênio. Se não logado com o convênio, não terá direito ao benefício... O sistema apresenta na hora do pagamento uma parte que será paga pelo convenio e a outra o cliente deve escolher a forma de pagamento... Sobre cadastro de usuários não conveniados e conveniados: o usuário informar o cpf ou nro do cartão e seus dados são baixados a partir do webservice da empresa o saldo do conveniado é compartilhado entre titulares e dependentes ..."*.

Ao serem perguntados sobre as restrições do sistema, os participantes apon-

taram algumas regras de negócio praticadas pela equipe, como, por exemplo, visibilidade das informações no *app* e tipo de pagamento diferenciados por usuário.

O componente proposto tem o objetivo de poder ser usado para a ambientação de diferentes perfils de profissionais atuantes no desenvolvimento de software. Sendo assim, a experiência técnica anteriormente adquirida pelo profissional pode influenciar a análise e compreensão do escopo do projeto apresentado através das *word clouds*. Neste sentido, é normal que a maturidade, evidenciada pelo tempo de experiência e papel desempenhado, de cada participante seja fator de influência na identificação de diferentes tipos de restrições relatadas.

Ao analisarmos as respostas dos três profissionais é possível observar quão rica é a resposta do engenheiro de requisitos (Novato3). Além de destacar restrições e regras de negócios, sua resposta revela fluxos de interações que podem ocorrer entre objetos de um cenário, o que ocorre quando o mesmo descreve as etapas de concessão de desconto conforme tipo de usuário e sobre a realização de cadastro de usuários.

3. **Quais dificuldades técnicas são discutidas?**

   Novato1: *"Problemas com o Android. Definição das imagens na app do play store."*.

   Novato2: *"Qual preço mostrar para o usuário? Me parece que existem dois campos de preço, um campo para o preço com desconto e outro campo com o preço integral do produto..."*.

   Novato3: *"Quais campos (banco de dados) do preço diz respeito ao preço para quem tem convênio. Especificações dos Webservices RESTful salvar a foto com o padrão: idProduto + ".jpg"e idUsuario + ".jpg- necessidade de salvar as imagens em pastas diferentes, para que a foto do produto com id=1 não substitua a foto do usuário com id=1, ou vice-versa. Como é o retorno para o JSON response (WebService) alterar a URI para : .../rest/usuario/login"*.

Neste caso, dos participantes Novato1 e Novato3 identificaram dificuldades técnicas com a manipulação de imagens em diferentes momentos do desenvolvi-

mento do projeto. A citação do Novato2 destaca dificuldade na compreensão de requisitos e regras de negócio, também destacada pelo Novato3. Já a citação do Novato3, além de evidenciar os pontos já citados, ainda destaca padrões e justificativas de codificação que a equipe adota ao manipular imagem e ao estabelecer comunicação com *WebService*.

Os termos `foto`, `imagem`, `webservice` e `JSON` foram identificados como assuntos frequentemente discutidos pela equipe (Tabela 5.1). Porém, destaca-se que a forma de navegação utilizada pelos participantes, para acessar os conteúdos associados aos termos das *word clouds*, pode ser apontada como fator que influencia o processo de formação do conhecimento. Não foi imposto aos participantes navegar em todos os termos das *clouds*, também não foi pré-definida uma ordem de visitação dos termos e nem mesmo sugerido o tipo de *word cloud* que eles deveriam analisar. Os participantes do estudo escolheram os termos conforme seus respectivos interesses.

4. **Quais desafios são discutidos?**

   Novato1: *"...implementar a funcionalidade de pagamento... estabelecer as regras de negócio relacionadas aos usuários conveniados e não conveniados. Comunicação dos responsáveis pelo servidor com a equipe de desenvolvimento do app. A equipe realiza consultas e os resultados não mostravam dados ou ocorriam erros."*.

   Novato2: *"Nada a declarar"*.

   Novato3: *"Houve muitos desafios quanto ao fechamento das regras de negócio: se o usuário conveniado poderia alterar o valor do seu cpf diretamente no aplicativo, quais dados poderiam ser alterados por usuários conveniados, valor da taxa de entrega e questões de publicitação do app."*.

Nesta questão, ficou claro para o testador (Novato1) e para o engenheiro de requisitos (Novato3) que algumas regras de negócio estavam pendentes de serem resolvidas, contudo, o programador (Novato2) destacou não conseguir elencar desafios enfrentados pela equipe, porém demonstrou evidências da pendência identificada pelo Novato1 e pelo Novato3 quando, na questão anterior, destacou uma dificuldade de entendimento de requisitos e regras de negócio.

Uma análise nas respostas dos participantes demonstra que estes conseguiram capturar funcionalidades, restrições, dificuldades técnicas e desafios enfrentados pela equipe durante o desenvolvimento do projeto. Destaca-se, ainda, que dois desses quatro itens capturados pelos participantes podem evidenciar raciocínios que embasam a tomada de decisões da equipe frente às tarefas de desenvolvimento, são eles: dificuldades técnicas e desafios enfrentados.

5. **A externalização do PFK lhe ajudou a compreender melhor o projeto no qual você vai trabalhar?**

   Novato1: *"Eu acredito que compreendi as principais funcionalidades do app... comecei pesquisando as informações dos últimos 2 meses."*.

   Novato2: *"Sim. O sistema ajudou a entender os requisitos do aplicativo que são discutidos entre os membros da equipe, me ajudou a entender o escopo do sistema, qual o objetivo do aplicativo em produção, quem era a pessoa que sempre foi consultada para as tomadas de decisão, e acho que consegue identificar quem era o dev, testador e design... Achei interessante que tanto no primeiro período quanto no segundo houve discussão dos membros da equipe sobre uma tela do sistema em que não deveria ser apresentado o valor do produto com desconto caso o usuário não estivesse logado. Parece que esse requisito do sistema não estava muito claro para eles... um ponto de dificuldade é que li o mesmo trecho de mensagem na lista de diferentes 'palavras frequentes' ... Analisei mais o período 1 (8 primeiros meses) do que o período 2 (2 últimos meses), pois continha mais informações."*.

   Novato3: *"Ajudou sim, pude verificar as principais funcionalidades e regras de negócio implementadas e os problemas ocorridos no projeto... Analisei todas as word clouds dos dois períodos. As que me deram mais informações foram a do primeiro período. A do segundo período só consegui informações de implantação, nada novo de funcionalidade... A word cloud que mais me trouxe informação foi a semântica, a técnica tinha muita informação duplicada da semântica. Houve palavras que me fizeram 'perder tempo', pois não traziam informação relevante como: msg, horário, atividades, loja..."*.

Apesar de todos os participantes estarem em consenso em relação à utilidade do componente em auxiliar a melhor compreender o projeto, seus relatos indicam necessidade de diminuição nas repetições das informações apresentadas e refinamento de termos apresentados nas *word clouds*. A fim de minimizar a repetição dos trechos de discussões apresentados, pode-se evitar a apresentação de discussões que se sobrepõem e aumentar o número de mensagens usadas para descrever o contexto das discussões apresentadas. Para minimizar o problema de palavras que, de acordo com Novato3, *"não traziam informação relevante"* estratégias futuras para eliminar substantivos que não caracterizam o desenvolvimento de um software podem ser adotadas. Além disso, os três participantes relataram ter níveis de interesse diferentes nas *word clouds* geradas. O Novato1 (testador) relatou ter interesse maior no passado recente do projeto, já o Novato2 (programador) relatou seu interesse maior nos dados do período remoto e Novato3 (engenheiro de requisitos) expressou interesse em analisar igualmente os dados dos dois períodos evidenciados. Tal fato endossa a importância de fornecer uma visão sobre a evolução temporal do projeto, baseando-se nos passados remoto e recente deste.

Na última etapa do estudo realizado, o gerente de projeto do sistema analisado avaliou a efetividade do componente desenvolvido. Para avaliar a precisão das informações reveladas pelas *word clouds*, o gerente de projeto foi solicitado a classificar a relevância dos 20 primeiros termos que compunham cada *word cloud* e que correspondem aos assuntos frequentemente discutidos pela equipe. Os resultados indicam que 70% e 75% dos termos presentes na *word cloud* semântica e na *word cloud* técnica do período referente ao passado remoto foram relevantes para a contextualização do projeto desenvolvido, respectivamente. Assim como, 50% e 85% dos termos apresentados nas *word clouds* semântica e técnica referente ao período de passado recente do projeto foram consideradas relevantes, respectivamente. A porcentagem de termos relevantes determinada pelo gerente de projetos endossa o relato feito pelo Novato3 ao afirmar a existência de *"termos que não traziam nenhuma informação relevante"* e corrobora com a necessidade de aprimoramento da etapa de extração do PFK.

Contudo, ao ser questionado sobre alguns termos da *word cloud* semântica com menor porcentagem de relevância (50%), o gerente de projeto destacou que existem termos na *cloud* semântica que deveriam ocorrer juntamente com termos

da *cloud* técnica. E, ainda, que a existência de alguns termos isolados não fazia sentido, pois estavam relacionados a um escopo específico do projeto. Tais observações mostram oportunidades para aprimoramento do componente através do uso de termos compostos na representação dos assuntos frequentemente discutidos. Tal estratégia se justifica, pois alguns termos isolados não possuem sentido completo.

Respondendo à QP1 deste estudo: A identificação e a extração dos assuntos frequentemente discutidos nos registros de comunicação de desenvolvedores auxiliam na determinação do escopo de desenvolvimento de projetos de software?

Sim. O estudo evidenciou que a identificação e a extração dos assuntos frequentemente discutidos pelas equipes auxiliam na determinação do escopo de desenvolvimento de projetos de software. Contudo, é necessário realizar aprimoramentos nos algoritmos a fim de maximizar a relevância dos assuntos identificados.

Ao analisar as respostas dos participantes, o gerente relatou que todos conseguiram compreender o escopo do projeto e, ainda, que a estratégia adotada é útil do ponto de vista técnico e social, já que permite o acesso às decisões técnicas e permite que novatos se familiarizem com papéis desempenhados pelos demais membros da equipe. Uma entrevista com o gerente de projeto revelou seu interesse em adotar o processo de externalização do PFK proposto em futuros projetos sob sua responsabilidade e, ainda, que a utilidade do PFK extraído não se limita aos novatos, membros antigos podem usar o processo de externalização do PFK para recordar decisões já tomadas.

Respondendo à QP2 deste estudo: Novatos conseguem compreender o escopo de desenvolvimento do projeto através da análise das mensagens associadas aos assuntos frequentemente discutidos pela equipe?

Sim. O estudo evidenciou que a abordagem revela resultados promissores ao possibilitar a compreensão do PFK pelos novatos. Porém, experiências prévias dos novatos podem interferir na compreensão do PFK.

5.2.4 Discussões

De acordo com Steinmacher *et. al* muitos são os desafios enfrentados por novatos ao ingressarem em uma equipe de desenvolvimento (Steinmacher et al., 2015). Como destaca Dagenais *et. al* (Dagenais et al., 2010), a familiarização com o escopo de desenvolvimento do software é uma das principais barreiras enfrentadas. Sendo assim, conhecer as demandas dos novatos e propor soluções que apoiem a adaptações destes a um projeto em andamento pode trazer vantagens para a equipe.

Nesse contexto, o *FK-Component* utiliza algoritmos de processamento de linguagem natural para tratar o conteúdo de mensagens provenientes dos *logs* de interação das equipes de desenvolvimento e revelar as discussões frequentes e raciocínios feitos pelo time frente às tarefas de desenvolvimento executadas. Os resultados obtidos são úteis para promover a transferência de conhecimentos de software, aumentar a possibilidade de compreensão do projeto e, consequentemente, melhorar a integração de novatos.

Como a determinação do PFK é um processo no qual o *FK-Component* é usado para identificar discussões frequentes, relacionadas aos projetos de software, não é garantida a identificação de todas as funcionalidades, restrições, dificuldades técnicas e desafios relatados pela equipe no registro de comunicação. Sendo assim, a utilização do componente não exclui a necessidade de leitura e compreensão das documentações oficiais dos projetos (método tradicionalmente utilizado), ele fornece meios para oportunizar a transferência de conhecimentos técnicos, sociais e organizacionais registrados nos *logs* de interação da equipe. Além de expor as decisões de projeto tomadas, as mensagens também revelam os nomes e os papeis de atuação dos membros na equipe de desenvolvimento. Além disso, sabendo-se que a aquisição de um conhecimento amplo e profundo do projeto de software em desenvolvimento demanda vivência com o problema tratado e maturidade, adquirida com tempo de trabalho, pressupõe-se que o *FK-Component* possa ser útil para diminuir a curva de aprendizado sobre o projeto. Por fim, destaca-se que apesar de ser, inicialmente, projetado para beneficiar novos membros, citações do gerente de projeto entrevistado destaca que o conhecimento revelado pelo *FK-Component* pode ser utilizado tanto por novatos quanto membros antigos.

5.3 Ameaças à Validade

Alguns pontos do estudo reportado podem influenciar em sua validade. Destaca-se: (1) quantidade de participantes; (2) *log* analisado; e (3) a entrada de novatos na equipe foi simulada.

Para minimizar tais ameaças, três perfis diferentes de profissionais com experiência na indústria de *software* foram utilizados, o *log* de interação pertence a um projeto real da indústria de Software e o gerente de projeto da equipe avaliou os resultados obtidos. Contudo, novos estudos devem ser realizados com o objetivo de avaliar a eficácia do componente em diferentes contextos de equipes de desenvolvimento.

5.4 Considerações

Este capítulo apresentou o *FK-Component*, o segundo dos três componente do *framework Miner4DevTeam*. O *FK-Component* tem por objetivo apoiar a determinação do *Project Frequent Knowledge* (Seção 3). O foco do componente é atender às necessidades de um público específico: os novatos de uma equipe de desenvolvimento. Neste componente, o problema de detecção de decisões e/ou discussões de projetos foi tratado considerando a frequência dos termos presentes nos registros de comunicação dos times. Para tanto, foram utilizas técnicas de mineração de dados para tratar o conteúdo das mensagens e revelar as discussões e raciocínios feitos pela equipe frente às tarefas de desenvolvimento executadas.

Baseando-se em palavras-chave que descrevem o contexto semântico e técnico do software, foram geradas *word clouds* que viabilizam a externalização do conhecimento e apoiam a determinação do PFK. Conhecer as demandas dos novatos e propor soluções que apoiem a adaptação destes a um projeto em andamento pode trazer vantagens para a equipe, como a diminuição do tempo de adaptação destes. O estudo conduzido apontou resultados promissores. Tanto os novatos quanto o gerente de projeto entrevistados destacaram a utilidade da abordagem para a integração de novos membros às equipes. Embora algumas empresas ofereçam suporte à integração de novos membros, nem sempre esta prática é uma realidade. Como contraexemplo citam-se as *startups* que, normalmente, necessitam atender

às demandas imediatas do mercado onde atuam.

O componente revela oportunidades para a transferência de conhecimento de software, aumento da compreensão do projeto e, consequentemente, a melhora da integração de novatos. Do ponto de vista do gerente de projeto, em média, 70% dos termos usados nas *word clouds* foram considerados relevantes para determinar o PFK do projeto. Contudo, o uso do componente não exclui a necessidade de compreensão da documentação oficial dos projetos, ele fornece oportunidades para transferência de conhecimento técnicos, sociais e organizacionais.

Direcionamentos futuros apontam para o aprimoramento dos algoritmos utilizados, a fim de minimizar a repetição dos trechos de discussões apresentados e termos com baixa relevância evidenciados pelas *word clouds*. Além disso, para estudos de outros tipos de conhecimento de projeto de software que possam estar registrados nos *logs* de comunicação dos times de desenvolvimento.

Capítulo 6

Minerando *Project Related Knowledge*

Este capítulo apresenta discussões relacionadas a aplicação e avaliação do *Related Knowledge Component* (*RK-Component*). O *RK-Component* foi projetado visando atender demandas provenientes do contexto de comunidades de projetos de software que usam fóruns de discussões. Um estudo realizado com três mantenedores de equipes de Software livre (OSS) hospedadas no GitHub revelou resultados promissores ao utilizar modelos de aprendizagem profunda genérico na identificação de postagens relacionadas (duplicadas e quase duplicadas) em fóruns de desenvolvedores. Especificamente, o planejamento, o desenvolvimento e a avaliação do *RK-Component* foram feitos considerando as características e limitações do fórum *GitHub Discussions*. Comunidades OSS podem se beneficiar com resultados gerados para minimizar uma possível degradação do fórum causada pela ocorrência de postagens duplicadas e discussões descentralizadas de tópicos iguais.

O capítulo está organizado da seguinte forma: a Seção 6.1 apresenta o *Visual Abstract* da pesquisa realizada com objetivo de definir e validar o *RK-Component*. A Seção 6.2 descreve o contexto de aplicação do *RK-Component*. A seção 6.3 apresenta o planejamento e os resultados de um estudo exploratório sobre as atividades de compartilhamento de *links* ocorridas no GitHub *Discussions*. Por fim, a Seção 6.4 apresenta as considerações finais do capítulo.

6.1 *RK-Component* - Visual Abstract

O *RK-Component* é resultado do terceiro ciclo de *design* realizado nesta pesquisa. Com o objetivo de apresentar um resumo dos principais pontos de idealização, desenvolvimento e validação do componente, a Figura 6.1 apresenta o *Visual Abstract* (Storey et al., 2017) da pesquisa que descreve o *RK-Component* (Lima et al., 2023). Nela são apresentados o objetivo, o contexto, a relevância, o rigor e a novidade do componente.

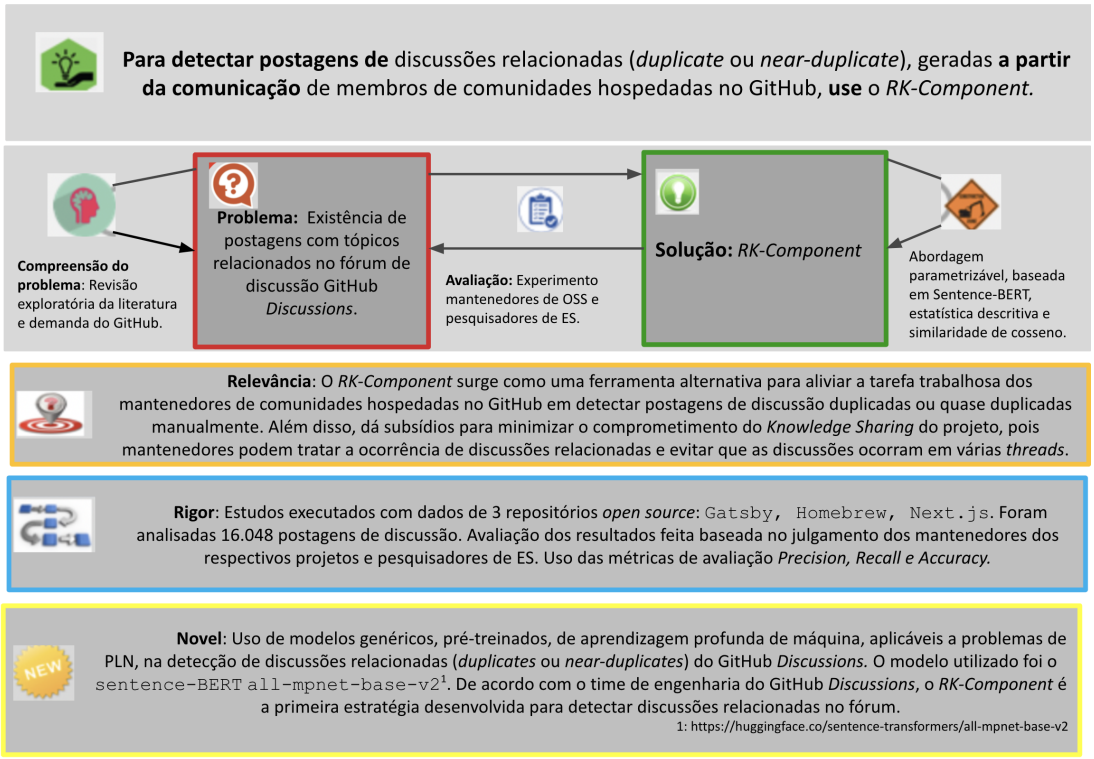

Figura 6.1: *Visual Abstract* do desenvolvimento do *RK-Component*.

As principais contribuições deste terceiro ciclo de *design* são:

- O desenvolvimento e avaliação de uma abordagem baseada em modelos de aprendizado de máquina profunda para detectar postagens de discussão relacionadas em fóruns do tipo PCQA.
- Evidências experimentais sobre o uso de um modelo de aprendizado de

máquina de propósito genérico para detectar postagens de discussão relacionadas criadas por desenvolvedores.

- Evidências experimentais sobre as aplicações práticas do *RK-Component* para as comunidades hospedadas na plataforma GitHub, sob a perspectiva de mantenedores OSS.

Como oportunidade de pesquisas futuras, destaca-se a otimização do modelo genérico utilizado pelo componente. Um estudo de viabilidade conduzido com dados extraídos do GitHub *Discussions* visou compreender o impacto das atividades de compartilhamento de *links* no compartilhamento de conhecimentos dos projetos hospedados na plataforma (Seção 6.3). Como resultado, observou-se que os usuários do *Discussions* compartilham *links* em *threads* de discussões para apoiar o entendimento correto de conhecimentos do projeto, frequentemente criando *links* entre postagens de discussões relacionadas. Este fenômeno viabiliza a utilização da rede de *links* geradas na construção de bases de dados rotuladas com pares de discussões relacionadas e, como consequência, possibilita a otimização do modelo de aprendizagem profunda utilizado pelo *RK-Component*.

6.2 *RK-Component* aplicado ao contexto *OSS*

Conforme descrito na Seção 3.4, o *RK-Component* visa identificar discussões (ou postagens) relacionadas criadas em fóruns de discussões do tipo Q&A voltados para profissionais de desenvolvimento de software. Especificamente, pretende-se identificar postagens duplicadas ou quase duplicadas, aqui chamadas postagens relacionadas que determinam o *Project Related Knowledge* (PRK). Postagens duplicadas são aquelas com o mesmo conteúdo. Os itens de postagem duplicados (título, descrição e autor) podem ser cópias exatas ou próximas. Os autores das postagens podem reescrever alguns itens adicionando ou excluindo informações. A detecção de postagens duplicadas é essencial para mitigar o problema de duplicação em fóruns de discussão colaborativa de desenvolvedores. Postagens quase duplicadas são aquelas postagens com tópicos semelhantes. Diferentes usuários com interesses, perguntas ou ideias semelhantes criam e comentam sobre eles. Itens de postagens quase duplicadas (título, descrição e autor) não são os mesmos,

mas compartilham informações relacionadas ao conteúdo um do outro. A detecção de postagens quase duplicadas é essencial para disseminar o conhecimento do projeto.

Mantenedores de comunidades *open source* podem se beneficiar dos resultados gerados para minimizar (1) a sobrecarga de trabalho associada a detecção manual de discussões relacionadas, (2) o retrabalho causado pela necessidade de responder diversas vezes a mesma pergunta, (3) a poluição da plataforma e (4) uma provável degradação na disseminação do conhecimento do projeto, já que este ocorre em diferentes postagens. Sendo assim, a seguinte questão de pesquisa (QP) guiou a investigação de validade do *RK-Component* no contexto das comunidades hospedadas no GitHub:

QP: Modelos genéricos de aprendizagem de máquina profunda, aplicáveis a problemas de processamento de linguagem natural (NLP), são eficazes na identificação de postagens de discussões relacionadas no GitHub *Discussions*?

Objetivando avaliar a efetividade do *RK-Component* como meio para viabilizar a determinação de PRK, foi realizado um estudo que analisou os registros de comunicação, realizados através do fórum *Discussions* de três comunidades OSS hospedadas na plataforma GitHub. O estudo realizado foi conduzido com três mantenedores de projetos OSS e três pesquisadores da área de Engenharia de Software. A eficácia do componente foi mensurada com base na perspectiva dos seis participantes que julgaram o relacionamento existente entre pares de discussões candidatas a relacionadas e apontaram evidências sobre os desafios e aplicações práticas associados a identificação do *Project Related Knowledge* no fórum *GitHub Discussions*. Os resultados obtidos mostram que o componente conseguiu identificar pares de discussões relacionadas. O componente atingiu valores de precisão que variam de 77% a 100% e *recall* de 66%. A metodologia, os resultados e as discussões geradas a partir deste estudo foram publicados em (Lima et al., 2023).

6.2.1 Forum GitHub *Discussions*

O *GitHub Discussions* é um fórum nativo disponibilizado para intermediar as discussões colaborativas entre membros e usuários das comunidades hospedadas

na plataforma GitHub (Hata et al., 2022). A própria empresa GitHub sugere que a utilização do fórum seja feita para fazer e responder perguntas, compartilhar informações, fazer anúncios e conduzir ou participar de discussões específicas sobre os projetos (GitHub, 2021a). O *Discussions* é apontado como um ambiente para que mantenedores, colaboradores e usuários discutam a utilização, o desenvolvimento e a atualização dos projetos em um único local, sem a necessidade da adoção de ferramentas de terceiros (GitHub, 2021a). Além disso, o fórum vem se destacando por permitir a separação entre discussões cotidianas de usuários do projeto e discussões direcionadas às equipes de desenvolvimento (*Issues* ou *Pull Request*) (Hata et al., 2022).

Para incentivar a comunicação entre membros e usuários das comunidades hospedadas no GitHub, que já utilizam o *Discussions*, ambos podem participar das *threads* de discussões criadas no fórum. A participação pode ser feita de quatro formas: criando, comentando, reagindo a uma postagem de discussão ou, simplesmente, lendo (GitHub, 2021a). Os usuários do fórum também podem pesquisar tópicos selecionados em postagens de discussões antigas. Para tanto, devem especificar palavras-chave no mecanismo de busca disponível no GitHub. Também é possível a utilização de qualificadores que restringem os resultados das buscas ao título, ao *body text* ou aos comentários das discussões (GitHub, 2021c). Para criar uma nova discussão, os autores devem especificar o título, o *body text* e a categoria a qual pertence a nova postagem (Figura 6.2).

As categorias são atributos obrigatórios das discussões que auxiliam a organizá-las em classes predefinidas, permitindo que os membros da comunidade conversem no lugar certo e encontrem discussões com características semelhantes (GitHub, 2021b). Conforme as necessidades do projeto, membros com autorização podem definir, criar ou excluir categorias para um repositório. Por padrão, o GitHub *Discussions* oferece cinco tipos de categorias: *Announcements*, *General*, *Ideas*, *Questions and answers* (Q&A) e *Show and tell* (GitHub, 2021b). Mantenedores podem criar discussões do tipo *Announcements* para relatar atualizações e notícias do projeto (GitHub, 2021a). Discussões do tipo Q&A são criadas para fazer perguntas, sugerir respostas e votar na resposta mais apropriada. As discussões do tipo *Ideias* podem ser criadas para relatar e compartilhar ideias de melhorias para o projeto. Já as discussões do tipo *Show and tell* são para discutir criações,

Figura 6.2: Exemplo de uma postagem de discussão.

experimentos ou testes relevantes. Por fim, as discussões do tipo *General* abordam qualquer assunto pertinente ao projeto (GitHub, 2021b).

Mantenedores relatam a aceitação positiva do uso do GitHub *Discussions* em seus projetos[1] e destacam que a ferramenta permitiu o crescimento das comunidades no mesmo ambiente onde há o desenvolvimento do projeto, melhorou e aumentou o engajamento dos membros e usuários das comunidades e permitiu separar *issues*, perguntas, *feature requests* e "*general chit-chat*". Mantenedores destacam ainda que as *threads* de discussões lhes dão acesso ao encadeamento do que já foi questionado, proposto ou sugerido acerca do tópico discutido. Dessa forma, estes podem tratar as demandas levantadas individualmente sem perdê-las em discussões mais amplas.

Neste estudo, as postagens de discussões públicas que ocorrem através do uso do GitHub *Discussions* são a fonte de entrada de dados para a avaliação do *RK-Component*. São analisadas as discussões de três diferentes projetos hospedados no GitHub: `Gatsby`, `Homebrew` e `Next.js`, conforme Tabela 6.1.

[1] https://github.blog/2021-08-17-github-discussions-out-of-beta/

6.2.2 Determinação do PRK

A determinação do PRK é um processo no qual o *RK-Component* é usado para detectar pares de discussões (ou postagens) relacionadas criadas em fóruns de discussões colaborativa de comunidades hospedadas no GitHub. O componente apresentado na Seção 3.4 foi aplicado no contexto de comunidades *open source*. Visando identificar discussões relacionadas criadas no fórum *GitHub Discussions*, o componente processou a similaridade de discussões criadas por usuários e membros de três diferentes projetos hospedados na plataforma (Tabela 6.1). Os resultados gerados foram avaliados por mantenedores das respectivas comunidades e por pesquisadores da área de Engenharia de Software (ES).

A Figura 6.3 mostra o processo geral conduzido para detectar os conjuntos com pares de postagens relacionadas. O processo compreende três etapas: Coleta de Dados (Figura 6.3 - A), pré-processamento dos dados (Figura 6.3 - B) e Determinação do PRK (Figura 6.3 - C).

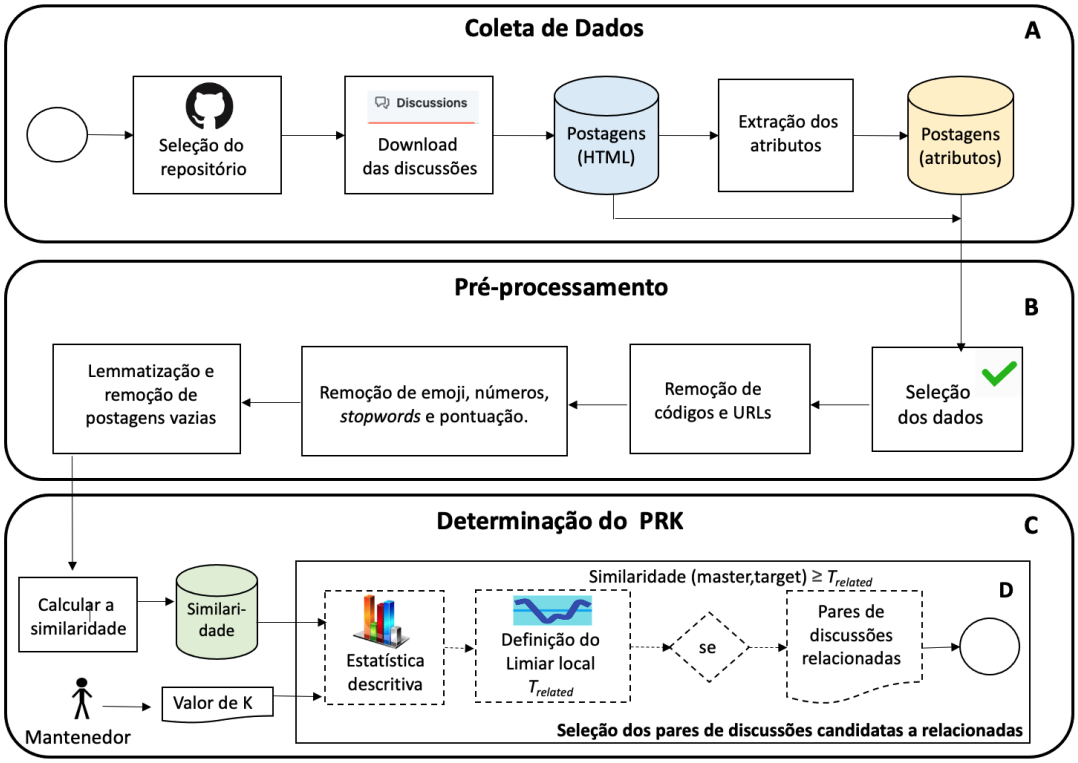

Figura 6.3: Processo de determinação do *Project Related Knowledge*.

Tabela 6.1: Repositórios selecionados para avaliação do *RK-Component*

Repositório	#*Threads* de Discussões
`Gatsby` [2]	1.886
`Homebrew`[3]	2.490
`Next.js`[4]	11.672
Total	**16.048**

Uma base de dados D foi construída para avaliar o *RK-Component*. Foram coletados e armazenados localmente todas as postagens de discussões públicas dos repositórios `Gatsby` ($D_{p=Gatsby}$), `Homebrew` ($D_{p=Homebrew}$) e `Next.js` ($D_{p=Next.js}$), conforme Tabela 6.1. Na etapa de coleta de dados (Figura 6.3 - A) foram coletados os arquivos HTML com as *threads* de discussões (discussão principal e comentários) e extraídos o conteúdo dos seguintes metadados: ID da discussão, a categoria a qual a discussão pertence, o criador (autor) da discussão, a data de criação, o número de comentários adicionados à discussão, o título da mesma e conteúdo textual da discussão principal. A seleção dos repositórios baseia-se no (1) uso do fórum *Discussions*, (2) indicação do time de profissionais da empresa GitHub e (3) na disponibilidade dos mantenedores das comunidades. As coletas foram executadas em diferentes ciclos, foram coletadas todas as postagens de discussão, públicas, disponíveis nos fóruns dos repositórios até setembro de 2022, totalizando 16.048 discussões.

6.2.2.1 Caracterização da base de dados

De acordo com a documentação da comunidade `Gatsby`, o `Gatsby` é um *framework* gratuito e de código aberto baseada em `React` que ajuda desenvolvedores a criarem sites e aplicativos extremamente rápidos [5] ("*Gatsby is a free and open source framework based on React that helps developers build blazing fast websites and apps.*"). O conjunto de discussões coletadas do projeto forma a base $D_{p=Gatsby}$. Na data da coleta dos dados, foram obtidas 1.886 discussões do projeto, sendo assim $|D_{p=Gatsby}| = 1.886$ (Tabela 6.1). O fórum contém discussões no nível do reposi-

[5] https://github.com/gatsbyjs/gatsby/blob/master/README.md

tório, o que significa que todas as postagens de discussão referentes ao projeto são acessíveis por membros e visitantes do repositório. A Figura 6.4-(a) mostra o número de novas discussões criadas na comunidade ao longo de 32 meses (01/2020 a 08/2022). De acordo com a janela de tempo analisada, a variação média da taxa de crescimento do uso do *Discussions* no `Gatsby` (considerando o número de novas discussões criadas) é de 13,4%. Este número reflete a aceitação e a utilização do fórum pelas comunidades do projeto. Os tipos de categorias, até então, disponibilizadas para a classificação das discussões do projeto eram `community`, `help`, `ideas-feature-requests`, `rfc` e `umbrella-discussions`. As discussões do tipo `help` predominavam no fórum do projeto, totalizando 76,03%, seguidas pelas discussões do tipo `ideas-fearute-requests`, `umbrella-discussions`, `rfc` e `community`, totalizando 19,24%, 1,85%, 1,48% e 1,37% das discussões, respectivamente (Figura 6.4-(b)).

O `Homebrew` é um projeto *open source* que facilita a instalação de ferramentas UNIX que a Apple não incluiu no macOS. Ele também pode ser usado para instalar software não empacotado na distribuição Linux sem a necessidade do `sudo`[6] (*"install the UNIX tools Apple didn't include with macOS. It can also install software not packaged for your Linux distribution to your home directory without requiring sudo*"). O conjunto de discussões coletadas do repositório `Homebrew` é denominado $D_{Homebrew}$. Na data de criação da base de dados, foram coletadas 2.490 discussões do projeto (Tabela 6.1), sendo assim $|D_{p=Homebrew}| = 2.490$. A Figura 6.4-(c) mostra a distribuição de novas discussões criadas no projeto ao longo de 24 meses (09/2020 a 08/2022). Apesar de estar disponível desde janeiro de 2020, a postagem de discussão mais antiga coletada do projeto `Homebrew` é de setembro de 2020. A variação média da taxa de crescimento do uso do fórum (considerando o número de novas discussões criadas) é de 18,02%. Os meses com maior número de discussões criadas foram dezembro/20, janeiro e fevereiro de 2021, totalizando 148, 151 e 195 novas discussões, respectivamente. Tais valores representam o maior pico mostrado pela Figura 6.4-(c). Diferentemente do `Gatsby` e do `Next.js`, a organização das discussões do projeto `Homebrew` é feita por tipo de problema e não por tipo de pergunta. O `Homebrew` possui um repositório específico, chamado "discussions", contendo as discussões ao nível de organização para o projeto. Este

[6]https://docs.brew.sh/

repositório, “discussions”, substitui o fórum obsoleto da comunidade. As categorias disponibilizadas pelo projeto até o momento da coleta de dados eram `casks`, `Getting-started`, `tap-maintenance-and- brew- development`, `daily-usage`, `linux`, `polls` e `Writing-formulae-casks`. As discussões do tipo `daily-usage` são as mais comuns no conjunto de dados $D_{Homebrew}$, totalizando 47,30%. Seguido por `Getting-started`, `casks`, `tap-maintenance-and-brew-development`, `linux`, `writing-formulae-casks` e `polls` totalizando 17,46%, 13,65%, 9,27%, 6,90%, 5,34% e 0,04% das discussões, respectivamente (Figura 6.4-(d)).

O `Next.js` é um projeto de código aberto que fornece uma solução para construir um aplicativo web completo com `React`[7] (”*provides a solution to build a complete web application with React*“). A comunidade `Next.js` se destaca por apoiar, desde o início, o lançamento do GitHub *Discussions*. O `Next.js` foi o projeto com maior número de discussões analisadas. Na data de coleta dos dados foram obtidas 11.672 postagens de discussões públicas, logo tem-se que $|D_{p=Next.js}| = 11.672$. A Figura 6.4-(e) mostra a distribuição de novas discussões criadas no projeto ao longo de 32 meses (01/2020 a 08/2022). Neste período, a variação média da taxa de crescimento do uso do fórum (considerando o número de novas discussões criadas) foi de 30,43%, sendo o projeto com a maior taxa de crescimento média dentre os três analisados. Os meses com maior número de discussões criadas foram maio, junho e outubro de 2020 com 589, 544 e 562, respectivamente (Figura 6.4-(e)). As categorias disponibilizadas pelo projeto, no momento da coleta dos dados, eram `ideas`, `help`, `polls`, `react-server-components`, `rfc`, e `show-and-tell`. As discussões do tipo `help` eram predominantes na base de dados do `Next.js`, totalizando 85,82% das discussões, seguidas pelas discussões do tipo `ideas`, `show-and-tell`, `rfc`, `react-server-components` e `polls`, totalizando 12,73%, 1,03%, 0,21%, 0,17% e 0,01% das discussões, respectivamente (Figura 6.4-(f)).

Por fim, a base de dados considerada na avaliação do *RK-Component* é composta pelas discussões coletadas dos três projetos descritos. Sendo assim, a base de dados D é dada por $D = D_{p=Gatsby} \cup D_{p=Homebrew} \cup D_{p=Next.js}$.

As seguir são descritas as etapas de pré-processamento e de determinação do PRK apresentadas na Figura 6.3-(b) e 6.3-(c), respectivamente.

[7] https://nextjs.org/learn/basics/create-nextjs-app

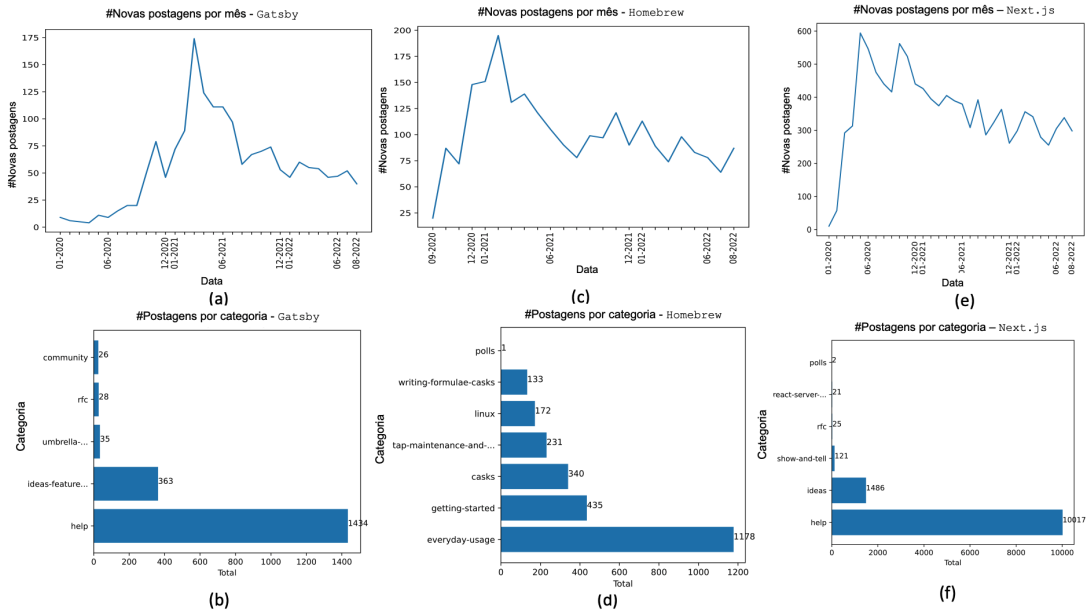

Figura 6.4: Caracterização da base de dados - `Gatsby`, `Homebrew` e `Next.js`

1. Pré-processamento

A coleção de dados gerada, D, foi submetida à etapa de pré-processamento. Primeiramente, foram aplicados filtros que têm por objetivo particionar a base de dados em subconjuntos de dados. A aplicação de filtros impacta na identificação dos conjuntos de pares de discussões candidatas a relacionadas, R. Foram utilizados dois filtros: (1) filtro de projeto e (2) filtro de categorias. O filtro de projeto, p, é utilizado para selecionar discussões de um projeto específico. Já o filtro de categorias, c, foi usado para delimitar o tipo das discussões, conforme categoria, que serão submetidas à análise do componente. O filtro de categorias pode receber quaisquer rótulos disponibilizado por um projeto p. Os filtros usados são apresentados na forma $p = \alpha | c = \beta$, onde p indica o nome do projeto (ex.: `Gatsby`, `Homebrew` ou `Next.js`) e c identifica o nome da categoria. A definição de ambos os filtros facilita a avaliação e a discussão dos resultados alcançados.

Para padronizar a escrita dos resultados e facilitar a leitura dos mesmos, optou-se por usar os seguintes rótulos de categoria: `Q&A`, `Ideas` e `ALL`. O rótulo `Q&A` é usado para referenciar as discussões do tipo "*Question and Answering*" (`help`) dos projetos. O rótulo `Ideas` é usado para referenciar as discussões do tipo

`ideas` ou `ideas-fearute-requests`. O rótulo `Q&A` refere-se à categoria `help` para os projetos `Gatsby` e `Next.js`, e o rótulo `Ideas` refere-se às categorias `ideas-feature-requests` e `ideas` dos projetos `Gatsby` e `Next.js`, respectivamente. O rótulo `ALL` é usado para indicar o processamento que todas as discussões, independente do tipo de categoria a qual pertença. Por exemplo, o rótulo de categoria `ALL` compreende todas as postagens de discussão pré-classificadas em `ideas`, `help`, `polls`, `react-server-components`, `RFC` e `show-and-tell` do repositório `Next.js`. Como descrito na Seção 6.2.2.1, a categorização das discussões do projeto `Homebrew` segue uma organização diferenciada, motivo pelo qual não foi aplicado filtro de categorias às discussões do projeto. Sendo assim, os resultados do projeto `Homebrew` são reportados considerando a configuração $c = ALL$. A divisão do conjunto de dados por categorias justifica-se, pois as discussões do tipo `Q&A` e `Ideas` são a maioria nos projetos `Gatsby` e `Next.js`. A união das discussões de tais categorias representa 95,28% e 98,55% das postagens de discussão presentes em $D_{p=Gatsby}$ e $D_{p=Next.js}$, respectivamente. Além disso, a documentação do *Discussions* encoraja o uso de ambas as categorias para propósitos diferentes: discussões do tipo `Q&A` para pedir ajuda à comunidade em questões relacionadas ao projeto e `Ideas` para compartilhar e discutir ideias para novos recursos (GitHub, 2021a).

Durante o pré-processamento também foram selecionados o título e o conteúdo (*body text*) das discussões. No total, foram gerados sete subconjuntos de dados sobre os quais a eficácia do componente foi avaliada. Os subconjuntos gerados são listados na na terceira coluna da Tabela 6.2, onde $D_{p=Gatsby|c=Q\&A}$ corresponde à coleção de discussões formada pelas postagens do tipo `help` do projeto `Gatsby`, $D_{p=Gatsby|c=Ideas}$ corresponde à coleção de discussões do tipo `ideas-feature-requests` do projeto `Gatsby` e assim sucessivamente.

Seguindo as subetapas de pré-processamento (Figura 6.3-B), os subconjuntos de discussões gerados passaram por diferentes fases de remoção de ruídos e formatação. A última etapa de pré-processamento desconsidera as discussões de tamanho zero. Foram desconsideradas 2 discussões do projeto `Gastby`, 3 do projeto `Homebrew` e 6 do projeto `Next.js`.

2. Determinação do PRK

Após a etapa de pré-processamento, o algoritmo verificador de postagens

relacionadas calculou os valores da similaridade semântica dos pares de discussões pertencentes aos sete subconjuntos de dados gerados (Tabela 6.2, coluna 3). O cálculo de similaridade foi feito conforme Seção 3.4.

Conforme mostra Figura 6.3-C, a determinação dos pares de postagens relacionadas depende da definição do parâmetro K, sendo o conjunto de pares de discussões candidatas a relacionadas, R, impactado por este. O valor de K é usado na seleção dos valores de similaridade das top-K discussões *targets* mais similares a cada discussão *master* presente no conjunto de dados analisado. Os valores de K testados foram 5 e 10. Para cada $D_{p=\alpha|c=\beta}$ subconjunto de dados apresentado na Tabela 6.2, o componente foi configurado para ser executado com $K = 5$ e $K = 10$. A configuração de K restringe o universo de busca por discussões relacionadas. Fazendo $K = 5$ e $K = 10$ a eficácia do componente será avaliado considerando as 5 e as 10 discussões mais similares ao conjunto de discussões analisadas, respectivamente. Valores maiores que 10 expandem o universo de busca e não foram considerados, pois o componente visa atingir maiores valores de precisão na identificação de pares de postagens relacionadas, em detrimento à abrangência. Por se tratar de um parâmetro que exige configuração, espera-se que os mantenedores de projetos OSS possam melhor definir o valor de K de acordo com suas respectivas necessidades. Os mantenedores podem diminuir o valor de K para aumentar as chances de detectar discussões duplicadas ou aumentar K para explorar ocorrências de discussões com tópicos semelhantes.

No total, a abordagem foi executada sob 14 diferentes configurações (Tabela 6.2, coluna 4). Para cada configuração foi calculado o valor do limiar local, $T_{related}$, e gerado o conjunto R de discussões relacionadas. A quarta coluna da Tabela 6.2 lista os conjuntos de discussões candidatas a relacionadas gerados a partir das configurações aplicadas. O conjunto $R_{p=Gatsby|c=Q\&A|K=5}$ corresponde aos pares de discussões candidatas a relacionadas identificadas para o projeto `Gatsby`, ao fazer $c = Q\&A$ e parametrizar $K = 5$. O conjunto $R_{p=Gatsby|c=Q\&A|K=10}$ corresponde aos pares de discussões candidatas a relacionadas identificadas para o projeto `Gatsby`, ao fazer $c = Q\&A$ e parametrizar $K = 10$, e assim sucessivamente.

Tabela 6.2: Configurações de avaliação do *RK-Component* : repositórios, categorias e K

Repositório	Categoria	Base de dados	Conjunto de candidatas a discussões relacionadas
`Gatsby`	`Q&A`	$D_{p=Gatsby\|c=Q\&A}$	$R_{p=Gatsby\|c=Q\&A\|K=5}$ $R_{p=Gatsby\|c=Q\&A\|K=10}$
	`Idea`	$D_{p=Gatsby\|c=idea}$	$R_{p=Gatsby\|c=idea\|K=5}$ $R_{p=Gatsby\|c=idea\|K=10}$
	`ALL`	$D_{p=Gatsby\|c=ALL}$	$R_{p=Gatsby\|c=ALL\|K=5}$ $R_{p=Gatsby\|c=ALL\|K=10}$
`Homebrew`	ALL	$D_{p=Homebrew\|c=ALL}$	$R_{p=Homebrew\|c=ALL\|K=5}$ $R_{p=Homebrew\|c=ALL\|K=10}$
`Next.js`	`Q&A`	$D_{p=Next.js\|c=Q\&A}$	$R_{p=Next.js\|c=Q\&A\|K=5}$ $R_{p=Next.js\|c=Q\&A\|K=10}$
	`Idea`	$D_{p=Next.js\|c=idea}$	$R_{p=Next.js\|c=idea\|K=5}$ $R_{p=Next.js\|c=idea\|K=10}$
	`ALL`	$D_{p=Next.js\|c=ALL}$	$R_{p=Next.js\|c=ALL\|K=5}$ $R_{p=Next.js\|c=ALL\|K=10}$

6.2.3 Avaliação do *RK-Component*

Visando avaliar a efetividade do *RK-Component* foram realizados estudos no contexto de comunidades *open source*. Visando identificar discussões relacionadas postadas no fórum *GitHub Discussions*, o componente processou a similaridade de discussões públicas criadas por usuários e membros de três diferentes projetos hospedados na plataforma (Tabela 6.1). Os resultados gerados foram avaliados pelos mantenedores das respectivas comunidades (no total, 3 participantes) e por 3 pesquisadores da área de Engenharia de Software (ES). Tanto os mantenedores quanto os pesquisadores de SE classificaram manualmente pares de candidatos a discussão relacionados como duplicados, quase duplicados ou não relacionados. Os mantenedores julgaram os pares de candidatos de discussão relacionados de acordo com seu projeto de trabalho. Os pesquisadores de SE atuaram na avaliação dos três projetos.

Os mantenedores, M_Gatsby, M_Homebrew e M_Next.js, foram contatados via time de profissionais do GitHub. M_Gatsby é um colaborador ativo da comu-

nidade de `Gatsby`. Ele cria conteúdo e temas para o projeto. O número de *commits* feitos por M_Gatsby ultrapassa o valor 600 nos últimos cinco anos. M_Homebrew é o líder do projeto `Homebrew`. Ele se atua ativamente nas atividades do projeto e já realizou mais de 7.000 *commits* nos últimos 11 anos. Por fim, M_Next.js também contribui ativamente para a comunidade de `Next.js`, tendo realizado mais de 500 *commits* no repositório. Os três pesquisadores, SE_R1, SE_R2 e SE_R3, contatados possuem diferentes *expertises* em desenvolvimento de software. SE_R1 é um profissional atuante na indústria de software e pesquisador da área de Engenharia de Software, SE_R2 é um colaborador e pesquisador de comunidades *open source* e SE_R3 é pesquisador da área de Engenharia de Software. Todos os pesquisadores possuem mais de cinco anos de experiência em suas respectivas áreas de atuação.

Os avaliadores receberam documentos *on-line* contendo instruções para avaliar os pares de postagens candidatas a relacionadas detectados pelo `RK-Component`, configurando o valor de K para $K = 5$ e $K = 10$. Os documentos (1) descreviam o conceito de postagens de discussão duplicadas e quase duplicadas e (2) listavam pares de candidatos a postagens relacionadas. Cada par de postagem candidato a relacionado foi descrito através dos ID e o título das discussões *master* e *target*. O ID era um link a partir do qual os avaliadores podiam acessar as postagens originais. Os avaliadores foram instruídos a adicionar o rótulo "D" para duplicadas, "R" para quase duplicadas e "N" para postagens não relacionadas. Os avaliadores podiam acrescentar comentários para justificar seus julgamentos.

O valor de precisão (*precision*) do componente foi calculado em comparação ao julgamento dos mantenedores de OSS e dos pesquisadores do SE. Os mantenedores de OSS possuem uma visão global do projeto. Eles poderiam julgar os pares de candidatos a discussão relacionados que requeriam um conhecimento mais aprofundado do contexto do projeto. Os pesquisadores de SE avaliaram pares de discussões já sabidamente relacionadas e apoiaram a avaliação de candidatos a discussões relacionadas do projeto `Next.js`. O julgamento dos mantenedores identificaram as previsões verdadeiramente positivas (*true-positive*).

6.2.4 Resultados

Como o modelo genérico de aprendizagem de máquina profunda usado trunca textos de entrada com mais de 384 palavras, primeiramente, foram analisados os comprimentos dos textos pertencentes as base de dados dos três projetos. Foram identificadas as porcentagens de postagens que não atendiam ao critério de possuir 384 palavras ou menos. No total, 21, 15 e 38 postagens nas bases de dados dos projetos `Gatsby`, `Homebrew` e `Next.js` possuem mais de 384 palavras, respectivamente. Esses valores representam menos de 0,70% do total de postagens de cada projeto, não representando um problema para os resultados aqui reportados.

As Tabelas 6.3, 6.4 e 6.5 mostram os valores dos limiares locais, $T_{related}$, e os números de pares de discussões candidatas a relacionadas identificadas, $|R|$, para os projetos `Gatsby`, `Homebrew` e `Next.js`, respectivamente. Nas tabelas também estão descritos, para cada combinação de filtro (projeto, categoria da discussão, e valor de K) os valores correspondentes ao tamanho das distribuições ($size(D)$) e as estatísticas descritivas de S (IQR, $Q1$, $Q2$ e $Q3$) usadas no cálculo dos limiares locais (Equação 3.2).

Para todos os filtros aplicados, o aumento do valor de K causou a diminuição dos valores de limiares locais $T_{related}$ (Tabelas 6.3, 6.4 e 6.5). A diminuição no valor do limiar faz com que o componente detecte novos pares de discussões candidatas a relacionadas. Os novos pares são aqueles cujo valor de similaridade antes não eram considerados *outliers*, porém passaram a ser com a alteração do valor de *Upper inner fence*, conforme Equações 3.1 e 3.2.

A Tabela 6.3 mostra os resultados obtidos para o projeto `Gatsby`. Para $K = 5$ e $c = Q\&A$, o valor do limiar foi calculado baseado nos valores de similaridade de 6.061 pares únicos de postagens de discussões, $size(D)$, gerando $T_{related} = 0{,}9493$ e detectando 3 pares de discussões candidatas a relacionadas, $|R|$. As parametrizações $|p = Gatsby|c = Q\&A|K = 10$ fizeram com que valor do limiar, $T_{related} = 0{,}9348$, fosse calculado baseado nos valores de similaridade de 12.111 pares de discussões e 5 pares de candidatas a relacionadas foram identificados. Comparando o limite local $T_{related}$ entre $Q\&A$ e $Ideas$, nota-se que o limite local $Ideas$ é menor tanto para $K = 5$ quanto para $K = 10$, sugerindo a necessidade de investigação da(s) diferença(s) existentes entre postagens de discussão `Q&A` e

Tabela 6.3: Estatística descritiva, valores de $T_{related}$ e $|R|$ - `Gatsby`

	`Gatsby` - Valores de similaridade					
	c = Q&A		c = Ideas		c = ALL	
	$K = 5$	$K = 10$	$K = 5$	$K = 10$	$K = 5$	$K = 10$
Size(S)	6.061	12.111	1.425	2.826	7.892	15.729
IQR	0,147	0,151	0,134	0,140	0,139	0,142
$Q1$	0,579	0,555	0,513	0,476	0,586	0,560
$Q2$	0,663	0,641	0,593	0,560	0,665	0,642
$Q3$	0,727	0,706	0,648	0,617	0,725	0,703
$T_{related}$	0,9493	0,9348	0,8498	0,8287	0,9339	0,9181
$\|R\|$	3	5	6	9	7	9

`Ideas`.

Tabela 6.4: Estatística descritiva, valores de $T_{related}$ e $|R|$ - `Homebrew`

`Homebrew` - Valores de similaridade		
	c = ALL	
	$K = 5$	$K = 10$
Size(S)	10.386	20.611
IQR	0,128	0,129
$Q1$	0,550	0,525
$Q2$	0,617	0,593
$Q3$	0,678	0,654
$T_{related}$	0,8717	0,8476
$\|R\|$	20	34

A Tabela 6.4 mostra os resultados obtidos para o projeto `Homebrew`. O componente quase duplicou as postagens avaliadas para detectar 14 novos pares de candidatos a postagens relacionadas ao usar $K = 5$ e $K = 10$. Além disso, o valor do limiar diminuiu em 0,024 unidades. Embora o número de pares avaliados tenha duplicado, o número de candidatos a postagens de discussão relacionadas não duplicou. Isso sugere que aumentar K também aumenta o número de instâncias de postagem de discussão não relacionadas na distribuição S.

Os números apresentados na Tabela 6.5, referentes ao tamanho da distribuição,

Tabela 6.5: Estatística descritiva, valores de $T_{related}$ e $|R|$ - `Next.js`

	`Next.js` - Valores de similaridade					
	c = Q&A		c = Ideas		c = ALL	
	$K = 5$	$K = 10$	$K = 5$	$K = 10$	$K = 5$	$K = 10$
Size(S)	41.426	82.033	5.832	11.476	48.120	95.327
IQR	0,110	0,113	0,112	0,112	0,108	0,110
$Q1$	0,589	0,567	0,560	0,532	0,597	0,574
$Q2$	0,647	0,625	0,619	0,590	0,653	0,632
$Q3$	0,700	0,680	0,673	0,644	0,705	0,685
$T_{related}$	0,8650	0,8501	0,8415	0,8130	0,8679	0,8509
$\|R\|$	132	175	90	151	220	309

Size(D), e quantidade de pares de discussões candidatas a relacionadas, $|R|$, do projeto `Next.js`, corroboram o cenário descrito por Ahasanuzzaman et al. (2016). Os autores destacam que o aumento da popularidade dos fóruns acarreta o aumento da quantidade de questões criadas e, consequentemente, pode aumentar o número de relacionadas (Ahasanuzzaman et al., 2016). Dado que o projeto `Next.js` é o maior em números de discussões criadas (Tabela 6.1), as chances de se ocorrer discussões relacionadas aumenta. O maior número de pares de discussões candidatas a relacionadas, para o projeto `Next.js`, foi obtida para a configuração $|p =$ `Next.js`$|c = ALL|K = 10$ onde 309 instâncias foram identificadas como pares de discussões relacionadas pelo componente (Tabela 6.5).

Para cada um dos 14 grupos de configuração apresentados na Tabela 6.2 coluna 4, o componente calculou diferentes valores de limites locais e detectou diferentes quantidades de pares de discussões candidatas a relacionadas. Destaca-se que todos os conjuntos de postagens candidatas a relacionadas detectados considerando $K = 5$ são subconjuntos dos conjuntos gerados por $K = 10$. Tabela 6.6 mostra o valor de precisão alcançado pelo componente considerando os 14 grupos de configuração. Ressaltamos que o componente é parametrizável. Os mantenedores podem definir o valor de K de acordo com seus respectivos interesses, podendo detectar postagens duplicadas ou similares.

Na Tabela 6.6 são apresentados os valores de precisão atingidos pela abordagem ao identificar pares de discussões candidatas à relacionadas. Conforme

Tabelas 6.3, 6.4 e 6.5, o menor conjunto de candidatas a relacionadas formado foi identificado para o projeto `Gatsby` fazendo $c = Q\&A$ e $K = 5$, logo temos $|R_{p=\texttt{Gatsby}|c=Q\&A|K=5}| = 3$. Contudo, para esta configuração o componente atingiu 100% de precisão (Tabela 6.6), ou seja, todos os pares identificam discussões relacionadas entre si. Por outro lado, o maior conjunto de candidatos a discussão relacionados detectado pelo componente foi obtido usando a configuração $|p = \texttt{Next.js}|c = ALL|K = 10$.No total, foram identificados 309 pares de discussões candidatas a relacionadas, $|R_{p=\texttt{Next.js}|c=ALL|K=10}| = 309$, dos quais 290 foram julgados pelos avaliadores como sendo relacionados. Com essa configuração o componente atingiu precisão de 93.85% (Tabela 6.6).

Tabela 6.6: Valores de Precisão atingidos pelo *RK-Component*

Categoria	$K = 5$	$K = 10$
	`Gatsby`	
Q&A	100% (3/3)	100% (5/5)
Ideas	83,33% (5/6)	77,78% (7/9)
ALL	100% (7/7)	100% (9/9)
	`Homebrew`	
ALL	95% (19/20)	91,17% (31/34)
	`Next.js`	
Q&A	93,94% (124/132)	89,14% (156/175)
Ideas	98,89% (89/90)	95,36% (144/151)
ALL	99,09% (218/220)	93,85% (290/309)

Os pares de discussões candidatas a relacionadas do projeto `Gatsby` foram avaliados sob a perspectiva de um dos mantenedores do projeto, M_Gatsby, e de dois pesquisadores de ES, SE_R1 e SE_R3. Os pesquisadores julgaram os pares com maior valor de similaridade, acima de 0,9415, e entraram em consenso sobre o relacionamento existente entre as discussões analisadas. Essa intervenção pontual foi solicitada pela equipe de profissionais do GitHub com objetivo de poupar o tempo e os esforços gastos pelo mantenedor do projeto `Gatsby` no julgamento de casos de duplicadas. O julgamento dos pares de discussões candidatas a relacionadas que demandavam conhecimento técnico sobre o projeto foi feito pelo

mantenedor M_Gatsby. No total, os pesquisadores julgaram 5 pares de candidatos a discussão relacionadas, concordando que todos os pares julgados continham discussões relacionadas. O mantenedor M_Gatsby julgou 13 pares que exigiam conhecimento técnico prévio sobre o projeto. Conforme Tabela 6.6, o componente atingiu o valor máximo de precisão (100%) na identificação de pares de discussões relacionadas do tipo `Q&A`. A categoria `Ideas` do projeto `Gatsby` apresentou precisão inferior às demais categorias. Analisando as previsões falso-positivas da categoria `Ideas`, notamos que o componente não capturou as especificidades de problemas relacionados ao projeto de dois pares. Embora os dois candidatos a postagens de discussão relacionadas abordassem o mesmo tópico e tivessem uma interseção de palavras-chave do projeto, eles abordaram problemas diferentes.

O julgamento dos pares de discussões candidatas a relacionadas do projeto `Homebrew` também foram julgados por dois pesquisadores da área de ES (SE_R1 e SE_R3) e por um dos mantenedores do projeto (M_Homebrew). Os pesquisadores julgaram pares de discussões candidatas a relacionadas com valor de similaridade acima de 0,9558. No total, os pesquisadores avaliaram 4 pares de duplicadas, concordando que todos os quatro pares continham postagens relacionadas. Já o mantenedor M_Homebrew julgou 30 pares de candidatas a postagens relacionadas. Os avaliadores julgaram que 19 (dos 20) pares com postagens candidatas a relacionadas apresentados em $R_{p=\texttt{Homebrew}|c=ALL|K=5}$ como relacionados. A partir da perspectiva dos avaliadores, 31 dos 34 pares de discussões são verdadeiramente relacionados. O mantenedor M_Homebrew julgou as discussões de três pares como não relacionadas. A categorização das discussões do projeto `Homebrew` não possui a mesma organização dos demais projetos aqui analisados. Por isso, os valores apresentados para o projeto são referentes ao cálculo de similaridade de todas as discussões presentes na base de dados do projeto ($c = ALL$), independente do tipo de categoria.

Por fim, o número de pares de discussões candidatas a relacionadas identificadas para o projeto `Next.js` exigiu maior participação dos pesquisadores no julgamento dos pares. Por se tratar de um número maior, o julgamento de todos os pares pelo mantenedor do projeto se tornou inviável. Sendo assim, dois pesquisadores (SE_R2 e SE_R3) avaliaram os pares contendo discussões candidatas a relacionadas do projeto `Next.js`. Contudo, para garantir a confiabilidade dos

valores reportados, os dois pesquisadores que participaram do julgamento dos pares tiveram consenso mensurado. A concordância entre os avaliadores foi medida usando o coeficiente *Cohen's Kappa* (Cohen, 1960), atingindo o valor 0,85. Esse valor indica concordância *almost perfect* conforme a interpretação proposta por Landis e Koch (Landis and Koch, 1977). No entanto, foi coletado o feedback do mantenedor do projeto `Next.js` em relação a uma amostra aleatória de 27 pares de postagens relacionadas e não relacionadas.

A Tabela 6.6 mostra que apesar do número de candidatas a relacionadas obtido com a variação do valor de K aumentar, a precisão atingida sofre uma degradação. No total, para o projeto `Gatsby`, o componente apontou 18 pares únicos de discussões candidatas a relacionadas, para o projeto `Homebrew` foram apontas 34 pares únicos de candidatas a relacionadas e para o projeto `Next.js` o componente apontou 344 pares. Em relação aos resultados do projeto `Gatsby`, dois pares de candidatos a discussão relacionados foram classificados como não relacionados pelo mantenedor do M_Gatsby. Assim, 16 dos 18 pares detectados estavam realmente relacionados, alcançando 88,88% de acurácia. Da perspectiva do mantenedor M_Homebrew, dos 34 pares de candidatos à discussão relacionados detectados, três são falsos positivos. Portanto, o componente previu com 91,17% de acurácia as postagens de discussão verdadeiramente relacionadas. Por fim, o *RK-Component* previu com acurácia de 91,86% as postagens de discussão relacionadas considerando o projeto `Next.js`. Dos 344 pares detectados, 28 são falsos positivos.

A taxa de *recall* (Equação 2.5) foi medida para avaliar a eficácia do *RK-Compoenet* na identificação de postagens relacionadas. Para isso, selecionamos, para cada projeto, uma amostra de aproximadamente 400 pares de postagens de discussão cuja similaridade foi calculada pelo componente, considerando todas as postagens de discussão nas bases de dados dos projetos ($c = ALL$) e configurando $K = 10$. Os pares de postagens de discussão foram selecionados aleatoriamente. Definimos o tamanho da amostra calculando o tamanho de uma amostra representativa de discussões em cada repositório, considerando uma margem de erro de 10% e um nível de confiança de 95%. No total, selecionamos quase 1.200 pares de discussões. Dois pesquisadores de SE classificaram manualmente (independentemente) os pares de postagens de discussão da amostra. Quantificamos a

Tabela 6.7: Avaliação do *RK-Component* - Análise de Amostras

K	$\|R\|$	*Precision*	*Recall*
	`Gatsby`		
10	4	100% (4/4)	11,76% (4/34)
20	5	100% (5/5)	14,70% (5/34)
30	6	100% (6/6)	17,64% (6/34)
40	7	100% (7/7)	20,58% (7/34)
50	9	100% (9/9)	26,47% (9/34)
60	10	100% (10/10)	29,41% (10/34)
70	11	100% (11/11)	32,35% (11/34)
80	14	100% (14/14)	41,17% (14/34)
90	16	100% (16/16)	47,05% (16/34)
100	19	89.47% (17/19)	50,00% (17/34)
110	21	85.71% (18/21)	52,94% (18/34)
120	29	72.41% (21/29)	61,76% (21/34)
	`Homebrew`		
10	6	100% (6/6)	10.00% (6/60)
20	23	95.65% (22/23)	36,66% (22/60)
30	42	78.57% (33/42)	55,00% (33/60)
40	60	66.66% (40/60)	66,66% (40/60)
	`Next.js`		
10	24	100% (24/24)	23,07% (24/104)
20	42	76.19% (32/42)	30,76% (32/104)
30	60	65.00% (39/60)	37,50% (39/104)
40	75	60.00% (45/75)	43,26% (45/104)

concordância entre os pesquisadores envolvidos na classificação manual para minimizar vieses e erros de classificação usando a medida Kappa de Cohen (Cohen, 1960). Os valores de Cohen's Kappa alcançados foram 0,90 para o projeto `Gatsby`, 0,62 para o `Homebrew` e 0,85 para o projeto `Next.js`. Os valores mostram concordância entre os avaliadores, os quais são quase perfeitos, substanciais e quase perfeitos (Landis and Koch, 1977). A análise da fração de pares relacionados no universo representado pela amostra é baixa. Para o projeto `Gatsby`, apenas 12% dos pares de discussão na amostra são relacionados. Para o projeto `Homebrew`, 14% dos pares de discussão são relacionados. Além disso, apenas 23% dos pares de discussão da amostra `Next.js` são relacionados.

A Tabela 6.7 mostra as taxas de *recall* alcançadas. Os três projetos apresentaram comportamento semelhante em relação às métricas *Precision* e *Recall* com base nas amostras analisadas. A precisão (*precision*) diminui à medida que a revocação (*recall*) aumenta. O valor de K causa essa variação. Valores menores de K limitam o espaço de pesquisa e aumentam o valor do limite local, detectando postagens relacionadas com alta precisão, mas comprometendo a detecção de todos os pares de postagens relacionadas. Valores baixo de K são úteis para detectar discussões com conteúdo textual próximo (duplicadas) e fornecem evidências para os mantenedores excluírem postagens de discussão do fórum da comunidade. Neste caso, um alto valor de *recall* pode não ser necessário, já que os mantenedores do OSS geralmente não querem todas as postagens de discussão relacionadas, preferindo apenas aquelas que podem ser duplicadas.

Por outro lado, valores maiores de K expandem o espaço de busca para postagens relacionadas e reduzem o valor do limite local, detectando novos pares de postagens relacionadas e comprometendo a precisão da abordagem. Os mantenedores podem definir valores altos de K para avaliar os posts de discussão de forma abrangente. Nesse caso, os mantenedores do OSS podem usar os resultados do *RK-Component* para detectar tópicos comumente discutidos no fórum e planejar intervenções apropriadas. Os mantenedores interessados em realizar uma análise profunda das postagens de discussão podem variar o valor de K de acordo com suas necessidades. Taxas de *recall* superiores a 60% são alcançadas ao definir $K = 120$ para `Gatsby` e $K = 40$ para o projeto `Homebrew`. Em ambos os projetos, o valor de K permitiu a taxa de precisão superior a 60%. Esses valores de *recall*

Tabela 6.8: Tempo de execução do *RK-Component* - $c = ALL$ e $K = 10$

#Discussões	**Criação dos embeddings (s)**	**Cálculo de similaridade (s)**
`Gatsby`		
1.883	645,55	488,55
`Homebrew`		
2.488	495,51	557,12
`Next.js`		
11.666	1833,03	1854,15

denotam recuperação melhor que aleatória (Buckland and Gey, 1994).

Embora a taxa de *recall* seja uma métrica relevante, o planejamento do *RK-Component* visou no desenvolvimento de uma abordagem mais conservadora que detecte postagens verdadeiramente relacionadas para apoiar o processo de tomada de decisão dos mantenedores do OSS. Além disso, o componente não faz intervenções automáticas, como excluir ou mesclar postagens de discussão. Uma abordagem conservadora evita que os mantenedores do OSS gastem muito tempo moderando pares de discussão verdadeiramente relacionados. Dado que o componente é uma abordagem parametrizável, os mantenedores podem ajustar o valor de K (aumentar ou diminuir) para expandir ou comprimir o espaço de pesquisa por postagens relacionadas e, consequentemente, alterar o valor do limiar local usado para detectar pares de postagens relacionadas. As amostras rotuladas de pares (não)relacionados de postagens estão disponíveis no pacote de reprodução[8].

Por fim, foi medido o tempo para criar os *embeddings* de sentença de todas as postagens e o tempo para calcular todas as taxas de similaridade entre todas as postagens. A tabela 6.8 mostra o tempo de execução do componente considerando $K = 10$ e $c = ALL$. O tempo de execução depende do número de postagens. Como o projeto `Next.js` é o maior, o *RK-Component* gastou muito mais tempo na análise do `Next.js` do que os outros projetos. A medida de tempo usada foi segundos (s). Foi utilizada uma máquina Intel Core i5 Dual-Core de 2,3 GHz

[8]https://zenodo.org/badge/latestdoi/635559152

com 8 GB para calcular esses valores. No entanto, é desnecessário recriar todos os *embeddings* de sentenças toda vez que o componente for executado. Pode-se salvar as *embeddings* para uso posterior.

Portanto, respondendo à QP deste estudo: Modelos genéricos de aprendizagem de máquina profunda, aplicáveis a problemas de processamento de linguagem natural (NLP), são eficazes na identificação de postagens de discussões relacionadas no GitHub *Discussions*?

Sim. O modelo pré-treinado, de propósito geral, aplicável a problemas de PLN, do *Sentence-BERT*, pode ser usado na detecção de postagens relacionadas realizadas no fórum GitHub *Discussions*. Os resultados apresentados nas Tabelas 6.6 e 6.8 mostram a eficácia do uso do modelo `all-mpnet-base-v2` para determinar os *embeddings* das postagens e detectar discussões relacionadas usando a similaridade de cosseno.

6.2.5 Discussões

A seguir, são discutidos os impactos causados pela alteração do valor K, os falsos positivos detectados, as perspectivas dos mantenedores em relação às postagens relacionadas detectadas, a eficácia do *RK-Component* comparado a outras abordagens e os benefícios propiciados pelo *RK-Component*.

6.2.5.1 Impactos causados pela alteração do valor *K*

O valor K delimita os limites de busca para pares de postagens candidatas a relacionadas. À medida que o valor de K aumenta, os limites do espaço de busca por postagens relacionadas também aumentam. Por outro lado, à medida o valor de K diminui tais limites diminuem, aumentando as chances de detectar duplicadas. Logo, a alteração do valor de K provoca variações no valor do limiar local $T_{related}$ e, consequentemente, no número de pares de discussões candidatas a relacionadas detectados. Ao configurar o componente para executar com $K = 5$ e $K = 10$, notou-se que os conjuntos com discussões candidatas a relacionadas criados para $K = 5$ são subconjuntos de $K = 10$.

Como consequência, há o risco de propagação em cascata de previsões falso-positivas através dos conjuntos de candidatos de discussões relacionadas. Fato que ocorreu nos conjuntos de discussões relacionadas detectados para $p = Gatsby$ e $c = Ideas$. O mesmo par (*master*,*target*) não relacionado, identificado por M_Gatsby, ocorre em $R_{p=\texttt{Gatsby}|c=Ideas|K=5}$ e $R_{p=\texttt{Gatsby}|c=Ideas|K=10}$. Embora a taxa de precisão tenda a diminuir, a abordagem detecta novos pares de discussões relacionadas quando o valor de K varia de 5 para 10.

Projeto `Gatsby`: Analisando $R_{p=\texttt{Gatsby}|c=Q\&A|K=5}$ e $R_{p=\texttt{Gatsby}|c=Q\&A|K=10}$ foram detectados dois novos pares de discussões relacionadas ao alterar o valor K de 5 para 10. O mantenedor do projeto `Gatsby`, M_Gatsby, julgou as postagens de ambos os pares como relacionadas. Considerando $p = \texttt{Gatsby}|c = Ideas|K = 5$ e $p = \texttt{Gatsby}|c = Ideas|K = 10$, o componente também detectou dois novos pares de discussões candidatas a relacionadas. No entanto, um dos dois novos pares foi julgado não relacionado por M_Gatsby. Dos sete pares de candidatas a relacionadas identificadas para $R_{p=\texttt{Gatsby}|c=ALL|K=5}$, houve uma interseção de cinco pares com $R_{p=\texttt{Gatsby}|c=Q\&A|K=10}$. Todos esses cinco pares foram considerados relacionados na avaliação do conjunto $R_{p=\texttt{Gatsby}|c=Q\&A|K=10}$. As discussões dos dois novos pares foram julgadas por M_Gatsby como sendo relacionadas. Contudo, as discussões dos novos pares foram criadas em categorias diferentes: na `Ideas` e na `Q&A`. Este cenário revela que usuários estão criando discussões relacionadas em categorias diferentes. Em relação ao conjunto de relacionadas $R_{p=\texttt{Gatsby}|c=ALL|K=10}$, dos nove pares identificados, sete já haviam sido julgados devido à interseção dos conjuntos. Os novos dois foram identificados como sendo relacionados pelo mantenedor M_Gatsby.

Projeto `Homebrew`: Analisando os conjuntos de pares com discussões candidatas a relacionadas criados considerando as configurações $p = \texttt{Homebrew}|c = ALL|K = 5$ e $p = \texttt{Homebrew}|c = ALL|K = 10$, o componente detectou 14 novos pares de discussões candidatas a relacionadas ao alterar o valor K de 5 para 10. Dos novos 14 candidatos, M_Homebrew julgou dois pares como não relacionados.

Projeto `Next.js`: O componente detectou 43 novos pares candidatos a discussões relacionadas ao variar o valor K e fixar $p = Next.js|c = Q\&A$. Dos 43 novos candidatos, os pesquisadores julgaram 11 pares como não relacionados. Isso significa que o componente detectou 32 novos pares de discussões relacionadas,

aumentando o valor de K. No entanto, os falsos positivos diminuíram o valor de precisão da abordagem de 94% para 89%, aproximadamente. Este cenário se repete para $p = Next.js|c = Ideas$ e $p = Next.js|c = ALL$. Ao alterar o valor K de 5 para 10, o componente detectou 61 e 89 novos pares de discussões candidatas a relacionadas para $c = Ideas$ e $c = ALL$, respectivamente. No total, seis e 14 novos candidatos para $c = Ideas$ e $c = ALL$ foram julgados não relacionados pelos avaliadores, respectivamente. Em ambos os casos, a taxa de precisão diminuiu.

Devido à relação de interseção entre os conjuntos com pares de discussões candidatas a relacionadas, os avaliadores já haviam julgado 208 dos 220 pares do conjunto $R_{p=\texttt{Next.js}|c=ALL|K=5}$. Os pesquisadores avaliaram os novos pares como relacionados. Entre os novos 12 pares, dez pares tiveram discussões criadas pelo mesmo usuário, dez pares tiveram uma das discussões criada como `Q&A` e a outra como `Ideas`, e um par teve uma das discussões criadas como `Q&A` e a outra como `show-and-tell`. Esta descoberta corrobora as descobertas do projeto `Gatsby`: os usuários criam postagens de discussão relacionadas em diferentes categorias. Em relação ao conjunto $R_{p=\texttt{Next.js}|c=ALL|K=10}$, os avaliadores já haviam julgado 303 dos 309 pares de discussões candidatas a relacionadas. Dois novos pares foram julgados como sendo não relacionados.

Os mantenedores de projetos hospedados na plataforma podem definir o valor de K de acordo com seus respectivos interesses. Diminuir o valor K aumenta o valor de precisão do componente. Valores de precisão mais altos garantem maior assertividade na detecção de verdadeiros positivos. Por outro lado, aumentar o valor K pode reduzir o valor de precisão. No entanto, aumentar o valor K também aumenta o número de pares de discussões candidatas a relacionadas detectados.

Valores menores de K restringem o espaço de pesquisa e aumentam o valor do limiar local, resultando em maiores valores de precisão, mas comprometendo a detecção de todos os pares de postagens relacionados. Valores baixos de K são benéficos para identificar duplicadas, fornecendo evidências para que mantenedores removerem postagens do fórum da comunidade. Nesse cenário, um alto valor de *recall* pode não ser relevante, pois os mantenedores geralmente buscam apenas postagens relacionadas que podem ser duplicadas. Por outro lado, valores maiores de K expandem o espaço de busca de postagens relacionadas e detectam novos pares de postagens relacionadas e comprometendo a precisão da aborda-

gem. Os mantenedores podem definir valores altos de K para avaliar as postagens abrangentemente. Neste caso, os mantenedores podem usar os resultados do *RK-Component* para percorrer as postagens (Buckland and Gey, 1994), detectando tópicos comumente discutidos e planejar intervenções apropriadas. Os mantenedores interessados em fazer uma análise profunda das postagens podem variar o valor de K de acordo com suas respectivas necessidades.

6.2.5.2 Predições falso-positivas

Objetivando compreender os falsos positivos identificados pelo componente, os pesquisadores analisaram e discutiram o conteúdo dos pares de discussões identificados como sendo falso-positivos. Indícios que justificam a ocorrência de falso-positivos levam a identificação de limitações na abordagem, apresentadas a seguir por projeto.

Analisando as previsões falso-positivas, notou-se que o *RK-Component* identificou postagens com tópicos semelhantes. No entanto, não conseguiu capturar a especificidade do problema do projeto. Neste sentido, o *RK-Component* pode falhar em tratar contextos particulares de projetos de software, sugerindo postagens que abordam a mesma característica do projeto como sendo relacionadas, porém, diferem na especificidade do assunto. Esta limitação é chamada de "limitação de especificidade do projeto".

Também notou-se que os autores das postagens usaram capturas de tela para detalhar ou descrever os problemas enfrentados e, ainda, usam descrições de log de erros para mostrar o rastreamento do erro. Além disso, foi observada a predominância de palavras-chave do *template* e palavras-chave dos projetos nas postagens identificadas como sendo falso-positivos.

O componente remove capturas de tela e descrições de erro incorporadas em tags HTML durante a etapa de pré-processamento. O componente também não usa imagens ou descrições de log de erros como fonte de evidência. No entanto, remover as capturas de tela e os erros de log pode eliminar a especificidade do problema. Além disso, após o pré-processamento, as palavras-chave do projeto podem se destacar do conteúdo real da discussão. Foi identificada a predominância das palavras-chave de *template* nos pares falso-positivos.

6.2.5.3 Perspectiva dos mantenedores de OSS em relação às discussões relacionadas detectadas

Mantenedores adicionaram comentários aos pares de discussões candidatas a relacionadas por eles avaliados. Tais comentários evidenciam (1) desafios no julgamento de discussões duplicadas e quase duplicadas, (2) motivos associados à existência de discussões relacionadas (duplicadas e quase duplicadas) e (3) aplicações práticas do componente.

- Desafios no julgamento de discussões duplicadas

Como limitações, destaca-se a imprecisão dos conceitos "discussões relacionadas" que pode depender da perspectiva dos avaliadores. Tal limitação pode introduzir vieses na avaliação do componente.

Com base nos comentários dos mantenedores sobre a amostra de discussões relacionadas analisadas, foram identificadas quatro motivos pelos quais os mantenedores classificaram pares de postagens como sendo duplicatos:

1. *São cópias exatas umas das outras* - "*...são duplicatas idênticas*" (M_Next.js).

2. *Abordam o mesmo problema* - "*O problema inicial é essencialmente o mesmo e eles têm a mesma solução primária...*" (M_Homebrew), "*As duas discussões são sobre o mesmo problema e a solução é funcionalmente a mesma, embora os comandos específicos dados sejam ligeiramente diferentes...*" (M_Homebrew), "*O problema que o usuário na discussão target está enfrentando parece ser idêntico (como observado por eles em sua descrição)...*" (M_Homebrew), "*Esses tópicos são idênticos...*" (M_Homebrew), "*Contexto diferente, mas mesmo problema e solução.*" (M_Next.js).

3. *Possuem interseção nos assuntos abordado* - "*...a discussão master também discute um problema secundário que foi levantado durante a correção inicial.*" (M_Homebrew), "*Mesma coisa exposta usando uma biblioteca externa para busca de dados*" (M_Next.js).

4. *Podem ser criadas pelo mesmo usuário* - "*...e foram abertos pelo mesmo usuário*" (M_Homebrew), "*Postado pelo mesmo autor também.*" (M_Next.js).

A conceituação de duplicadas tende a variar de acordo com critérios objetivos e subjetivos. O primeiro critério exposto pelos mantenedores—cópias exatas—é objetivo e fácil de ser julgado, pois julgar duplicadas que são cópias exatas não exige que os avaliadores façam uma análise semântica profunda do conteúdo das discussões. No entanto, o segundo e o terceiro critérios exigem que os avaliadores analisem as discussões semanticamente para identificar a ocorrência de tópicos sobrepostos. Embora o quarto critério seja objetivo, os avaliadores não podem considerá-lo isoladamente para julgar duplicadas.

Por outro lado, foram identificadas cinco motivos pelos quais os mantenedores classificaram pares de postagens como sendo quase duplicadas:

1. *Não são necessariamente cópias exatas umas das outras* - "*...problemas semelhantes, mas não uma duplicada exata*" (M_Homebrew), "*Como usuários diferentes estão em threads diferentes, elas nunca serão 100% duplicadas.*" (M_Homebrew), "*Estes estão todos relacionados a imagens, então eu diria que estão 'relacionadas', mas não são exatamente iguais*" (M_Next.js).

2. *Abordam problemas similares* - "*Mesma solução e problemas semelhantes...*" (M_Homebrew), "*Estas (discussões) estão discutindo a mesma coisa...*" (M_Homebrew), "*Ambos os usuários estão enfrentando o mesmo problema inicial...*" (M_Homebrew), "*...Estão todas relacionadas a imagens...*" (M_Next.js).

3. *Podem abordar assuntos complementares* - "*A discussão de destino é na verdade uma continuação da discussão principal...*" (M_Homebrew), "*Esta é a RFC inicial de uma implementação de recurso, a outra discussão é uma questão pós-release.*" (M_Next.js), "*Esta discussão fez com que o RFC 14890 fosse criada*" (M_Next.js).

4. *Podem abordar especificidades diferentes dentro de um mesmo tópico* - "*Problema muito semelhante que se relaciona com o mesmo problema inicial, mas o problema real subjacente nos dois era totalmente diferente*" (M_Homebrew), "*...Um usuário abriu a discussão para obter ajuda porque a solução sugerida não funcionou. O outro usuário está perguntando por que o problema ocorreu em primeiro lugar.*" (M_Homebrew), "*Um pedido de recurso para o recurso de otimização de imagem, relacionado, mas falando*

sobre coisas diferentes." (M_Next.js), "*Portanto, o tópico geral (guia) é o mesmo, o guia sobre o qual deveria tratar é diferente.*" (M_Gatsby).

5. *Podem abordar o mesmo problema em contextos diferentes* - "*...é exatamente o mesmo bug... mas contexto diferente.*" (M_Next.js)..

Dada a imprecisão dos conceitos e baseando-se nos comentários dos mantenedores, nota-se que os mesmos também enfrentaram desafios na classificação dos pares de discussão como duplicadas ou quase duplicadas — " *Isso se confunde um pouco entre duplicado e quase duplicado ...*"(M_Homebrew), "*Definitivamente, há alguma semelhança entre os dois, mas não está muito claro se o problema é o mesmo... Então, é difícil dizer, mas não acho que sejam o mesmo.*" (M_Homebrew), "*...Eu os consideraria duplicados sem o contexto adicional do que era necessário para corrigir o problema na discussão target.*" (M_Homebrew), "*Eu estou lutando com este um pouco. São perguntas tecnicamente diferentes, então entendo porque o usuário abriu as duas (ambas têm o mesmo autor)...*" (M_Homebrew), "*É muito parecido, quase duplicado... mas ele tem uma solução diferente...*"(M_Next.js). Esta imprecisão endossa a decisão de considerarmos as postagens duplicadas e quase duplicadas como sendo relacionadas.

Outra limitação observada refere-se como usuários descrevem suas dúvidas/-problemas/solicitações, pois pode interferir tanto nos resultados gerados pelo componente quanto no julgamento dos avaliadores. Às vezes, pode ser um desafio para os usuários expressarem o problema claramente e com detalhes contextuais suficientes — "*Ambas as pessoas tiveram um problema semelhante, mas descrito tão vagamente que poderia ser qualquer coisa.*"(M_Homebrew).

- Motivos associados à existência de discussões relacionadas

Comentários do mantenedor do projeto `Homebrew` evidenciam três motivos pelos quais usuários do *Discussions* criam postagens de discussão relacionadas. Embora não se trate de uma lista exaustiva, os três itens a seguir permitem que os mantenedores de OSS entendam as razões que levam a existência de discussões relacionadas e possam planejar ações para lidar com a situação. Os motivos foram discutidos com a equipe de engenharia do GitHub *Discussions* e formalmente

comprovados através de trechos de postagens coletadas no fórum dos projetos analisados. Logo, usuários do *Discussions* criam discussões relacionadas porque:

1. *Desejam reforçar um pedido de ajuda* - "*... Depois que os mantenedores pararam de responder, o usuário abriu uma nova discussão na tentativa de continuar.*" (M_Homebrew).

2. *Desejam acrescentar novas informação às postagens anteriormente criadas* - "*O usuário nunca obteve uma resposta, então abriu um novo problema com mais algumas informações algumas semanas depois.*" (M_Homebrew).

3. *Buscam alternativas diferentes para a solução do problema* - "*...mas eles estão insatisfeitos com a solução apresentada na discussão principal e gostariam de uma alternativa.*" (M_Homebrew).

Especificamente no projeto `Homebrew`, o mantenedor destacou o cenário de múltiplos pares de discussões duplicadas entre si. O mantenedor afirma que se trata de uma situação peculiar, "*Estes são um pouco complicado...*". Apesar de serem discussões relacionadas, "*Então, eu chamaria os problemas de duplicadas porque todos eles são o mesmo problema...*", a solução dos problemas levantados depende da configuração de hardware e software das máquinas, "*..cada situação é diferente depende totalmente da máquina e configuração específica do usuário.*". O mantenedor afirma ainda que as soluções são diferentes e que não há forma única de ajudar com a demanda, "*As soluções reais podem ser diferentes, mas não temos como ajudar com isso...*" e que, apesar disso, a solução dada através do fórum é a mesma, "*...do nosso ponto de vista, o conselho que podemos dar é o mesmo*" (M_Homebrew). A identificação deste cenário pode ser usada para justificar uma necessidade de intervenção ou priorização, por parte dos mantenedores, em pontos específicos do projeto, objetivando a solução de problemas recorrente enfrentados pelas comunidades OSS.

- Aplicações práticas do *RK-Component*

Por fim, como aplicações práticas do componente, os mantenedores sugerem:

1. *Possibilidade de união das threads de discussões relacionadas* — "*Essas (discussões) poderiam ter sido reunidas em uma discussão e faria sentido...*" (M_Homebrew), "... Mas, se dependesse de mim, elas deveriam ter sido juntas na mesma discussão." (M_Homebrew)..

2. *Possibilidade de remanejamento das discussões como comentários umas das outras* — "*...o novo problema provavelmente deveria ter sido postado como um comentário na discussão principal*" (M_Homebrew), "*...(eles) teriam recebido uma atenção melhor como um comentário um sobre o outro.*" (M_Next.js), "*Esta discussão poderia ter sido suficiente como um comentário sobre 21633.*" (M_Next.js)..

3. *Possibilidade de recrutamento de colaboradores para tarefas específicas* — "*...pode ser útil para pessoas que procuram outros guias para contribuir (neste caso).*" (M_Gatsby).

Embora os mantenedores tenham identificado algumas aplicações práticas para o *RK-Component*, a abordagem não visa excluir, mesclar ou organizar postagens de discussão automaticamente. No entanto, os resultados gerados pelo componente podem apoiar os mantenedores na realização de tais atividades.

6.2.5.4 Comparando a eficácia do *RK-Component*

Nesta seção, as medidas de eficácia relativa obtidas pelo componente são comparadas com os métodos anteriores desenvolvidos para detectar postagens duplicadas em diferentes canais de comunicação usados por desenvolvedores de software descritos na Tabela 6.9. As taxas de *recall* reportadas nas pesquisas são comparadas com as taxas de *recall* alcançadas pelo *RK-Component*. Não é objetivo identificar qual abordagem é a melhor. São fornecidas evidências da adequação do *RK-Component* para detectar postagens duplicadas e quase duplicadas baseando-se em um modelo de uso genérico e pré-treinado do `Sentence-BERT`.

Algumas pesquisas avaliaram as abordagens propostas usando medidas de precisão@k e recall@k. Ambas as métricas são normalmente usadas para medir a relevância dos principais k resultados para as preferências ou necessidades do usuário, especialmente em cenários em que se espera que o sistema forneça uma

lista ordenada de itens ou resultados (problemas de ranque). Deve-se ressaltar que o valor de K neste estudo tem uma interpretação diferente da explicada anteriormente de k (precisão@k e recall@k). Usamos o valor K para delimitar os limites de pesquisa para candidatos a postagens relacionadas e criar a distribuição S. O *RK-Component* usa o valor K para selecionar os valores de similaridade das K principais postagens mais semelhantes a cada postagem de discussão no conjunto de dados de entrada. Quanto maior o valor de K, maior o número de valores de similaridade selecionados. O *RK-Component* não foi avaliado com base nas principais recomendações de K. Nosso objetivo é avaliar todo o conjunto de candidatos relacionados retornados pela abordagem, pois não consideramos R uma lista ranqueada de postagens relacionadas.

Primeiramente, a precisão do componente é comparada com os valores de precisão relatados por pesquisas que visam identificar duplicadas em sistemas de rastreamento de bugs. A precisão do *RK-Component* varia entre 88% (`Gatsby`) a quase 92% (`Homebrew` e `Next.js`) e 66% de revocação. Alipour et al. (2013), Kukkar et al. (2020) e Cooper et al. (2021) relataram que suas respectivas abordagens atingiram 88%, 85%-99% e 84% de precisão. Runeson et al. (2007) relatam que atingiram entre 24%-42% de revocação. Sendo assim, a faixa de valores alcançada sugere a aplicabilidade do componente na identificação de postagens relacionadas.

Em segundo lugar, os valores relativos de precisão e *recall* alcançados pelo *RK-Component* são comparados com abordagens que identificam duplicadas nos fóruns de perguntas e respostas voltados para desenvolvedores. A abordagem proposta por Zhang et al. (2015), `DupPredictor`, alcançou uma taxa de *recall* de 63,8%. A abordagem de Mizobuchi and Takayama (2017) alcançou valores de *recall* variando de 15,48% a 43,13%. A abordagem de Zhang et al. (2018) alcançou *recall* entre 66% e 86%. Wang et al. (2020) obteve os melhores valores para *recall@20*, variando de 76% a 79%. Gao et al. (2022) relatam ter alcançado entre 68%-79% de *recall*. Por fim, Pei et al. (2021) relatou ter alcançado um *recall* de 82,28%. Observa-se que as abordagens variam em relação ao valor de *recall* relatado. Conforme a Tabela 6.7, o *RK-Component* alcançou uma taxa de *recall* de quase 67% para o projeto `Homebrew`. Acreditamos que esse valor seja porque o componente usa um modelo de uso geral. Todas as pesquisas contrastadas usam bases pré-rotuladas para treinar ou otimizar modelos de *machine* ou *deep*

learning. Criar um conjunto de dados rotulados para o fórum de discussões é uma oportunidade de pesquisa. No entanto, o dinamismo e a variedade de contextos de software no GitHub são um desafio para criar tal conjunto de dados. O conjunto de dados rotulados criado para medir a taxa de *recall* do *RK-Component* está disponível no pacote de reprodução[9].

Por fim, os valores relativos de *recall* de abordagens que visam remover a duplicação de *issues* e *pull requests* na plataforma GitHub são comparados com as taxas de *recall* alcançadas pelo componente. Yu et al. (2018) relatou alcançar 70% de *recall*, a abordagem de Li et al. (2017) alcançou entre 55,3% e 71%, por fim, e Zhang et al. (2020) relatou atingir entre 38%-51% de *recall@5* e 45-65% de *recall@10*. Logo, tem-se que *RK-Component* pode auxiliar os mantenedores na detecção de postagens duplicadas ou quase duplicadas no fórum *Discussions*.

Com base nos valores de precisão relatados (Tabela 6.6) e na taxa de *recall* alcançada (Tabela 6.7), afirma-se que o *RK-Component* se adequa para detectar postagens relacionadas (duplicadas e quase duplicadas) no GitHub *Discussions*. No entanto, pretende-se melhorar a precisão da abordagem e os valores de *recall* alcançados. Planeja-se evoluir o algoritmo do componente para que este considere trechos de código e imagens na etapa de medição de similaridade das postagens.

A diversidade de contextos de projetos hospedados na plataforma GitHub, gera oportunidades para o desenvolvimento de abordagens automáticas de identificação de postagens relacionadas no fórum *Discussions*. Contudo, o número significativo de comunidades de OSS, a singularidade dos projetos e as limitações mencionadas desafiam o desenvolvimento e validação de tais sistemas. Esses motivos endossaram a decisão de design do *RK-Component* de usar um modelo de aprendizado de máquina de propósito geral, calcular valores de limiares locais e enfatizar o aumento dos valores de taxas de precisão alcançados.

6.2.5.5 Implicações do *RK-Component*

O *RK-Component* se caracteriza pelo (1) uso de modelos genéricos de aprendizagem de máquina profunda, aplicados a problemas de PLN; (2) não dependência de uma base pré-rotulada com exemplos de discussões duplicadas ou relaciona-

[9]https://zenodo.org/badge/latestdoi/635559152

Tabela 6.9: Sumário de pesquisas que visam detectar duplicadas em canais de comunicação de desenvolvedores.

Foco da pesquisa	Referencia	Como os autores modelaram o problema?	Dataset	*Precision (P)*	*Recall (R)*	*Accuracy*	Comentários
Bug reports	Runeson et al. (2007)	Problema de ranque	Relatórios de defeitos de desenvolvimento.		24%-42%		Abordagem baseada em PLN.
	Alipour et al. (2013)	Tarefa de classificação	Repositório de bugs do Android ecosystem.			91%	A abordagem se baseia em algoritmos de ML.
	Kukkar et al. (2020)	Tarefa de classificação	Seis bases de (Lazar et al., 2014), (Lerch and Mezini, 2013)		79%-94% R@20	85%-99%	Abordagem baseada em CNN.
	Cooper et al. (2021)	Problema de ranque	RICO dataset (Deka et al., 2017)			83% top-2	A abordagem usa técnicas de visão computacional, reconhecimento óptico e recuperação de texto.
Fóruns do tipo Q&A	Zhang et al. (2015)	Problema de ranque	Base do Stack Overflow rotulada		64% R@20		A abordagem combina o valor de similaridade de 4 evidências
	Mizobuchi and Takayama (2017)	Problema de ranque	Base do Stack Overflow rotulada		43% R@20		Abordagem baseada em modelos Word2vec.
	Zhang et al. (2018)	Tarefa de ranque-classificação	Base do Stack Overflow rotulada	75%-86%	66%-86%		A abordagem usa estratégias de ranque, DL e RI.
	Wang et al. (2020)	Tarefa de classificação	Base do Stack Overflow rotulada		76%-79% R@5		Abordagem baseada em CNNs, RNNs e LSTMs
	Pei et al. (2021)	Tarefa de classificação	Base do Stack Overflow rotulada	82%	82%		Abordagem baseada em *Attention-based Sentence* e modelos ASIM
	Gao et al. (2022)	Tarefa de classificação	Base do Stack Overflow rotulada		68%-79%		Abordagem baseada em word embedding e CNNs
Atividades no GitHub	Wang et al. (2019)	Tarefa de classificação	DupPR (Yu et al., 2018)	73% P@1	65% R@1		Abordagem baseada no algoritmo AdaBoost.
	Li et al. (2017)	Problema de ranque	Os autores construíram um dataset com PRs duplicados		54%-83% R@20		Abordagem baseada em RI e PLN
	Zhang et al. (2020)	Terefa de recomendação	Dataset próprio de duplicadas[10]		45%-61% R@10		Abordagem baseada em RI e DL.

dos; (3) não dependência de modelos treinados para compreender um contexto específico; (4) possibilidade de ser utilizado nos diferentes contextos de software retratados nas comunidades OSS; e (5) prioriza maiores taxas de precisão na identificação de pares de postagens relacionadas (*precision*), em detrimento à abrangência (*recall*). Tais características visam minimizar desafios associados (1) à necessidade de bases pré-rotuladas, através da qual modelos de aprendizagem de máquina possam ser treinados e/ou otimizados; (2) à criação de tais bases, pois trara-se de uma tarefa custosa, e pode sempre ser considerada ineficiente e inadequada; e (3) à grande diversidade de contextos gerados pelas singularidades dos projetos de software.

Pesquisas anteriores mostram que os mantenedores de OSS precisam gerenciar múltiplos aspectos dos projetos para garantir que a visão dos projetos perdure (Guizani et al., 2022) e a sustentabilidade de longo prazo dos projetos (Dias et al., 2021). Para isso, os mantenedores executam diferentes atividades, abrangendo tarefas de código e não-codificação (Dias et al., 2021; Trinkenreich et al., 2021). Tais responsabilidades intensificam a carga de trabalho dos mantenedores de OSS. Tan and Zhou (2020) destaca a proposição de ferramentas como uma prática recomendada para descentralizar as responsabilidades dos mantenedores. Além disso, Dias et al. (2021) relata que "*para sustentar uma visão de longo prazo do projeto, os mantenedores devem delegar tarefas.*" Desta forma, o *RK-Component* surge como uma alternativa ferramental para aliviar a tarefa trabalhosa dos mantenedores de detectar postagens duplicadas ou quase duplicadas.

Novas oportunidades de pesquisa podem surgir diante da diversidade e singularidade das comunidades OSS hospedadas no GitHub. Surgem oportunidades para entender como as comunidades usam o fórum de discussões do GitHub e como as categorias de discussão diferem. Os comentários dos mantenedores trazem *insigths* para a proposição de melhorias do *RK-Component*. Por exemplo, pode-se usar os julgamentos dos mantenedores para otimizar o classificador fornecendo amostras de discussões relacionadas e não relacionadas. Pode-se também planejar estratégias para minimizar a limitação especificidade do projeto, tratando a predominância de palavras-chave de projetos e palavras-chave de *templates*.

6.2.6 Ameaças à Validade

Embora o componente seja uma abordagem parametrizável, baseada em modelos de aprendizado de máquina genérico e estatísticas descritivas, o estudo realizado pode apresentar limitações.

O *RK-Component* foi avaliado em três comunidades OSS. Devido a diversidade e singularidade das comunidades hospedadas no GitHub, faz-se necessária a execução de novos estudos para avaliar os resultados em diferentes comunidades. Para minimizar esta limitação, a equipe de engenharia do GitHub sugeriu a seleção das comunidades analisadas. Além disso, o conjunto de dados refere-se a uma janela de tempo específica que não reflete o momento atual dos fóruns de discussão do GitHub. Porém, o projeto `Next.js` se destaca por possuir uma alta taxa de utilização do fórum.

Além disso, destaca-se que julgar a relação entre as postagens é uma atividade subjetiva e pode introduzir vieses na avaliação do componente. A avaliação do *RK-Component* apresenta alguns desafios, como: (1) os avaliadores devem fazer uma análise semântica do conteúdo das postagens; (2) o julgamento dos aspectos técnicos presentes nas postagens requer conhecimento prévio do projeto; e (3) o julgamento envolve (in)precisão humana quanto ao conceito de "relacionado", embora tenhamos definido o conceito de postagens relacionadas, a interpretação depende da perspectiva dos avaliadores. Para minimizar esta ameaça, introduzimos o significado de "postagens relacionadas" para os mantenedores de OSS e pesquisadores de SE antes que estes classificassem as postagens como sendo relacionadas ou não. Também entramos em contato com mantenedores de OSS selecionados para a realização do julgamento dos pares candidatos a discussões relacionadas.

O cálculo do limiar local também pode ser considerado uma limitação. Postagens relacionadas aqueles pares identificados como *outliers* em uma distribuição S. S contém os valores de similaridade das K postagens mais semelhantes a cada postagem no conjunto de dados. À medida que se aumenta o valor de K, o valor da mediana da distribuição diminui, assim como o valor do limiar local. Objetivando a melhora nos valores de precisão atingidos, o valor de K foi configurado para 5 e 10. A definição do valor de K também é uma limitação. O *RK-Component*

maximiza a taxa de precisão definindo o valor K próximo a 1. À medida que o valor de K aumenta, o espaço de busca para postagens relacionadas fica mais extenso e a abordagem pode detectar novos pares de itens relacionados. Os valores relativos da avaliação do componente foram comparados com os valores relativos da avaliação de outras abordagens. Como esta é a primeira tentativa de detectar postagens relacionadas no *Discussions*, não se pode decidir qual abordagem tem o melhor desempenho.

Além disso, a taxa de *recall* do *RK-Component* foi calculada com base em uma amostra de 400 pares de candidatos a postagens relacionadas, selecionados para cada repositório. Embora a taxa de *recall* tenha atingido 66%, os valores de precisão alcançados (Tabela 6.6) mostram que o *RK-Component* é eficaz na detecção de postagens relacionadas no *Discussions*. Mantenedores podem parametrizar o componente de acordo com suas respectivas necessidades.

Por fim, o componente considera apenas o primeiro *post* das discussões para medir a similaridade entre pares de postagens. Argumenta-se que comentários e respostas são *feedback* ou raciocínio acerca dos tópicos principais das postagens. No entanto, uma análise mais aprofundada deve ser feita. Além disso, o modelo trunca o texto de entrada com mais de 384 palavras. Uma análise mostrou que 21, 15 e 38 postagens nos conjuntos de dados `Gatsby`, `Homebrew` e `Next.js` têm mais de 384 palavras, respectivamente. Esses valores representam menos de 0,70% do total de postagens em cada projeto. Embora não represente um problema para este estudo, poderia ser uma limitação em fóruns com postagens mais longas.

6.3 Compartilhando *Project Knowledge* através de *Hyperlinks*

O fórum *Discussions* tem se destacado por ser um local confiável e flexível para intermediar a comunicação de comunidades OSS hospedadas na plataforma GitHub, o que possibilitou o crescimento das comunidades e aumentou o engajamento dos membros[11]. Mantenedores enfatizam a facilidade com que a *interface* do *Discussions* permite que usuários adicionem *hyperlinks* (ou simplesmente, "*links*") entre

[11] https://github.blog/2021-08-17-github-discussions-out-of-beta/

discussões e *issues*.

Pesquisas anteriores demonstram que o compartilhamento de *links* é uma prática comum entre de usuários da plataforma GitHub (Chopra et al., 2021; Zhang et al., 2020; Li et al., 2018; Zampetti et al., 2017). Os desenvolvedores compartilham referências de *links* nas *issues* e nos PRs para documentar PRs (Zampetti et al., 2017), vincular *issues* e PRs relevantes juntos (Zhang et al., 2020) e referenciar informações entre projetos (Li et al., 2018). No entanto, as atividades de compartilhamento de *links* no fórum de discussões carecem de investigação. De fato, os usuários do GitHub *Discussions* compartilham *links* nas postagens de discussão. Acredita-se que as atividades de compartilhamento de *links* no fórum ajudam a estabelecer um entendimento comum e apoiam a disseminação do conhecimento dos projetos.

Diante deste cenário, foi realizado um estudo exploratório sobre as atividades de compartilhamento de *links* ocorridas no GitHub *Discussions*. Objetiva-se investigar o propósito, as intenções associadas aos *links* compartilhados e o impacto destes no compartilhamento de conhecimentos de projetos de software. Especificamente, pretende-se responder às seguintes questões de pesquisa (QPs):

- **QP1:** Como as *threads* de discussões estão lincadas (vinculadas) a outros recursos? As *threads* de discussões compreendem tanto a postagem principal quantos os comentários adicionados às discussões. Esta questão de pesquisa visa investigar o tipo de recurso referenciado e determinar a extensão da disseminação do conhecimento do projeto por meio dos *links* compartilhados. A QP1 foi dividida em três subquestões:

  QP1.1: Com que frequência os usuários compartilham *links* no *Discussions*?

  QP1.2: Qual é a proporção de *links* internos e externos ao GitHub?

  QP1.3: Quais são os tipos de recursos vinculados?

- **QP2:** Quais são as intenções dos usuários ao compartilhar *links* no *Discussions*? Além de investigar o tipo de recursos referenciados, é necessário saber o motivo que fez com que os usuários do *Discussions* compartilhassem *links* nas postagens de discussão.

Para responder às perguntas de pesquisa, foi realizada uma análise quantitativa e uma qualitativa dos *links* compartilhadas no fórum. Na análise quantitativa foi investigada a frequência com que 10 comunidades OSS compartilham *links* em *threads* de discussão e os tipos de recursos referenciados. Foram analisados 65.621 *links* extraídos de 29.251 *threads* de discussão pública ocorridas no *Discussions*. Na análise qualitativa, foram avaliados 1.106 *links* selecionados do projeto `Next.js` a fim de investigar os motivos do compartilhamento dos *links*. As respostas às questões de pesquisas expostas acima proveem informações acerca da criação e do uso da rede de *links* gerados a partir do *Discussions*. As comunidades OSS podem se beneficiar dos resultados gerados para melhor entender o impacto do fórum na disseminação do conhecimento dos projetos e promover a adoção do fórum como o local oficial de interação das comunidades. Os usuários do OSS podem se beneficiar dos resultados para melhor compreender o relacionamento entre os diferentes recursos do projeto (*issues*, PRs, outras postagens de discussão, documentação, site do projeto, etc.). Por fim, a equipe de engenharia do próprio GitHub pode se beneficiar dos resultados para melhor entender a importância e o uso do fórum. Além disso, este estudo é o primeiro passo para coletar evidências acerca da viabilidade de uso da rede de *links* na construção de bases de dados rotuladas com pares de discussões relacionadas. O que permitiria a otimização do modelo de aprendizagem profunda utilizado pelo *RK-Component*. A metodologia, os resultados e as discussões geradas a partir deste estudo foram submetidos para o processo de revisão do journal *Journal of Systems and Software* (JSS).

6.3.1 Metodologia

A Figura 6.5 mostra o processo geral executado para responder às perguntas de pesquisa. Com objetivo de compreender as atividades de compartilhamento de *links* no *Discussions*, foram executadas as etapas de (1) criação da amostra de dados, (2) determinação dos tipos de recursos referenciados e (3) determinação do motivo pelo qual o link foi compartilhado.

- **Amostra de dados**

A Figura 6.5-(A) mostra o processo de criação da amostra de dados considerada neste estudo. Foram analisadas as discussões de dez comunidades OSS

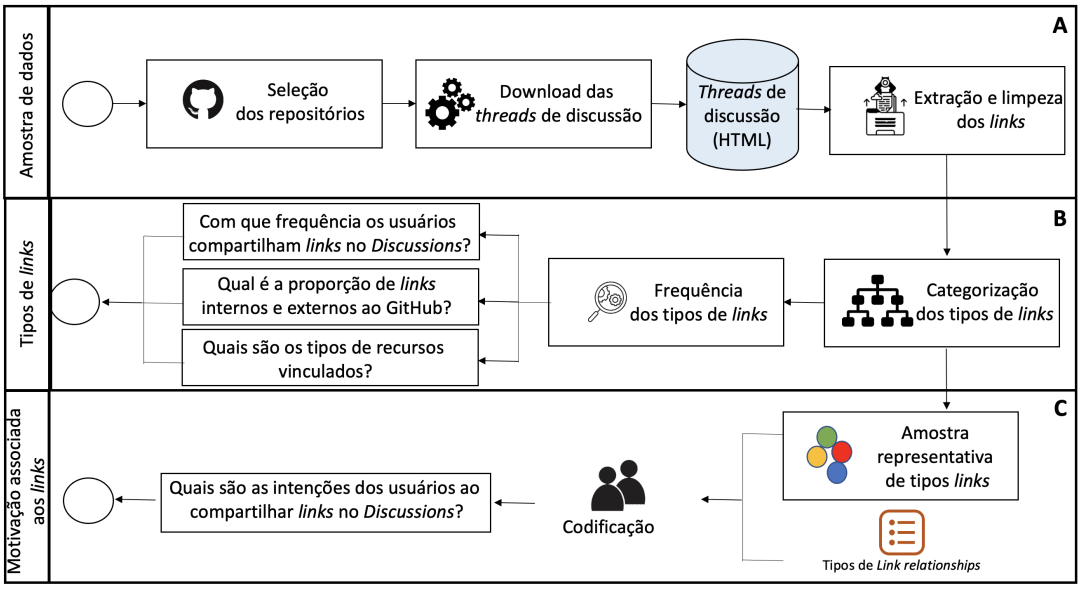

Figura 6.5: Processo da análise de *links* no GitHub *Discussions*

(Tabela 6.10). Os repositórios foram selecionados com base (1) no uso do fórum *Discussions* (Hata et al., 2022) e (2) nas recomendações do time de engenheiros do GitHub *Discussions*. No total, foram analisadas 29.251 postagens de discussão. Todas as discussões públicas das 10 comunidade OSS, disponíveis até setembro de 2022, foram coletadas. A biblioteca Python Beautiful Soup[12] foi usada para extrair as referências de *links* das discussões. Foram considerando os *links* disponíveis na primeira postagem da discussão e nos comentários.

Após a extração dos *links*, foram excluídos os *links* que não agregavam informações valiosas ao contexto de interesse da pesquisa. No total, foram extraídos 65.621 *links* que compõem a amostragem de dados. Não foram considerados os seguintes tipos de *links*:

1. *Links Mailto*. Os *links* do tipo *mailto* ativam o cliente de e-mail padrão no computador para enviar um e-mail. Em página HTML, as especificações de *links* "mailto" são apresentadas da seguinte forma: $< a\ href =$"$mailto:name@email.com$"$>\ texto\ < /a >$.

2. *Links embutidos em e-mail*. Os *links* de cancelamento de assinatura e *web-*

[12]https://beautiful-soup-4.readthedocs.io/en/latest/

Tabela 6.10: Repositórios usados na criação da amostra de dados

Repositório	#*Threads* de discussão	#*links*	Tempo (anos)	Linguagem predominante
`dotnet/csharplang`	2.728	18.273	6	C#
`gatsbyjs/gatsby`	1,666	5.381	8	JavaScrit
`laravel/framework`	1.548	1.724	10	PHP
`livewire/livewire`	1.842	3.788	4	Blade
`prisma/prisma`	2.286	3.952	4	TypeScript
`react-hook-form/react-hook-form`	2.522	4.449	4	TypeScript
`tanStack/react-table`	946	929	6	TypeScript
`tailwindlabs/tailwindcss`	3.505	5.345	5	JavaScript
`vercel/vercel/`	1.674	2.386	7	TypeScript
`vercel/next.js`	10.534	19.394	6	JavaScript
soma	**29.251**	**65.621**		

push são referências automáticas incorporadas em e-mails. Esses *links* aparecem em *threads* de discussão quando os usuários copiam o conteúdo do corpo do e-mail nas discussões.

3. *Links de imagens*. Não foram considerados *links* que apontam para recursos sob o domínio "$user-images.githubusercontent.com/$" por serem *links* para recursos de imagem.

4. *Links de busca*. Os *links* de busca ativam a máquina de pesquisa do GitHub, por exemplo, "$https://github.com/vercel/next.js/search?q=*$".

- **Tipos de recursos referenciados**

Nesta etapa são identificados os dados necessários para responder a QP1: Como as *threads* de discussões estão lincadas (vinculadas) a outros recursos? A Figura 6.5-(B) mostra o processo executado na determinação dos tipos de recursos referenciados.

Os *links* foram categorizados de acordo com seus respectivos tipos, definidos pelos endereços de URL referenciado. Foi realizada uma análise quantitativa dos resultados para determinar:

QP1.1: *Com que frequência os usuários compartilham links no Discussions?*

Para responder esta questão, foi determinado o número de discussões com *links* e calculado a frequência com que usuários compartilham *links* nas *threads* de discussões. Os resultados ajudam a determinar se o compartilhamento de *links* é uma prática comum entre os usuários do fórum *Discussions*.

QP1.2: *Qual é a proporção de links internos e externos ao GitHub?*

Os *links* no domínio "github.com" foram categorizados como *links* Internos (ou seja, *links* para recursos hospedados na plataforma GitHub). Caso contrário, os *links* foram classificados como Externos. *Link* cuja URL absoluta não faz referência explícita ao domínio "github.com" é classificado como sendo Externo. Essa categorização é a primeira etapa para identificar os tipos de recursos referenciados com mais frequência pelos usuários do *Discussions*. Pretende-se identificar se os recursos referenciados estão ou não hospedados na plataforma GitHub. A proporção de *links* internos e externos permite entender a extensão do compartilhamento de *links*. Os resultados determinam se o compartilhamento de conhecimento dos projetos extrapola os limites dos repositórios OSS e da plataforma GitHub.

QP1.3: *Quais são os tipos de recursos vinculados?*

Após identificar se os *links* são internos ou externos, estes foram classificados conforme o endereço URL. Esta é uma estratégia bem conhecida para realizar inferências com base nas atividades de compartilhamento de *links* em fóruns colaborativos. Chopra et al. (2021) analisaram as atividades de compartilhamento de *links* dos desenvolvedores no GitHub *Pull Request*. Os autores construíram duas taxonomias orientadas à referência para descrever (1) os tipos de informações referenciadas e (2) como o recurso referenciado é codificado nos PRs. Para fazer isso, os autores conduziram uma análise de conteúdo qualitativa em *links* extraídos de PRs públicos do GitHub.

Para mapear o padrão de disseminação do conhecimento dos projetos (1) nos repositórios dos projetos OSS, (2) entre repositórios da plataforma GitHub e (3) com recursos externos foi realizada uma análise qualitativa nos *links* extraídos. Dois pesquisadores identificaram, codificaram e refinaram iterativamente os tipos de *links* internos e externos. Como resultado, foram definidas subcategorias para *links* internos e externos. Uma vez definidas as categorias e as subcategorias, os 65.621 *links* (Tabela 6.10) foram classificados automaticamente. Assim, foi possível determinar os tipos de recursos mais frequentemente compartilhado nas

postagens de discussão.

- **Motivação associada ao compartilhamento dos *links***

Visando identificar os principais motivos para o compartilhamento de *links* no GitHub, Li et al. (2018) categorizaram e conceituaram os tipos de relacionamento entre *issues units* (*issues* e PRs). Os autores identificaram seis tipos de relacionamentos (ou causas) que motivam a criação de *links* entre *issues* e PRs, incluindo relacionamentos que expressam (1) dependência - *Dependent relationship*, (2) duplicação - *Duplicate relationship*, (3) relevância - *Relevant relationship*, (4) referência - *Referenced relationship*, (5) soluções - *Fixed relationship* e (6) aprimoramento - *Enhanced relationship* (Li et al., 2018). O relacionamento *Dependent relationship* expressa que a resolução da *issue unit* onde o link foi compartilhado depende da *issue unit* de destino. O relacionamento *Duplicate relationship* indica que as duas *issues units* vinculadas pelo *link* discorrem o mesmo tópico. O relacionamento *Relevant relationship* indica que as *issues units* vinculadas abortam tópicos semelhantes, mas não iguais. O relacionamento *Referenced relationship* é usado para designar que o conhecimento registrado na *issue unit* referenciada pode servir para entender a *issue unit* de origem. O relacionamento do tipo *Fixed* indica que a *issue unit* de origem apresenta uma solução para resolver o problema relatado na *issue unit* destino. Por fim, o *Enhanced relationship* expressa que a *issue unit* onde ocorreu o compartilhamento do link realiza algumas alterações na *issue unit* de destino para torná-la robusta ou atender aos requisitos.

As seis categorias apresentadas por Li et al. (2018) foram utilizadas como base na análise qualitativa dos *links* compartilhados no *Discussions*. Para tanto, foi necessário reformular os conceitos das relações *relationships* identificadas por Li et al. (2018) ao escopo do GitHub *Discussions*. A Tabela 6.11 apresenta as definições reformuladas dos tipos de relacionamento e exemplos explicativos.

A análise qualitativa dos *links* visa responder a QP2 — Quais são as intenções dos usuários ao compartilhar *links* no *Discussions*? — Pretende-se identificar as razões que levaram os usuários do GitHub *Discussions* compartilhar *links*. Para isso, uma amostra de *links* foi analisada com base nos seis principais motivos para o comportamento de *links* apontados na Tabela 6.11.

Tabela 6.11: Conceituação dos tipos de relacionamento associados a motivação do compartilhamento de *links* no *Discussions*

Tipos de Relacionamentos Derivados	**Definição Derivada**	**Exemplo**
Dependent	A resolução do problema discutido depende do recurso de destino.	Isso será corrigido depois do . *merge* deste PR #21930
Duplicate	A postagem de discussão e o recurso de destino discutem o mesmo problema.	Postado originalmente por @Fan..ton0 em #27650 (comment)
Relevant	A postagem de discussão e o recurso de destino discutem tópicos semelhantes. Seus conteúdos estão relacionados.	Acho que isso está relacionado a #18419
Reference	O conhecimento no recurso de destino pode ser útil para entender o assunto discutido.	Como está descrito em #15453
Fixing Proposal	O recurso de destino fornece informações que podem resolver o problema discutido.	Você pode conseguir esse comportamento hoje usando a propriedade loader
Enhancement Proposal	A discussão sugere (ou faz) alterações no recurso de destino para torná-lo mais robusto.	Eu tinha postado uma pergunta inicial aqui, mas seria bom...

Para entender os motivos do compartilhamento de *links* em postagens de discussão, foi analisada manualmente uma amostra de 1.106 *links* selecionados entre os *links* do projeto `Next.js`. Primeiramente, foi criada uma amostra representativa de *links* para cada tipo de link identificado na QP1.3. O tamanho da amostra foi definido configurando a margem de erro amostral em 10% e o nível de confiança em 95%. Em seguida, a amostra foi codificada conforme os "tipos derivados de relacionamentos" descritos na Tabela 6.11. Dois pesquisadores classificaram uma amostra aleatória de *links* separadamente. A concordância entre os pesquisadores foi medida usando o coeficiente Kappa de Cohen (Cohen, 1960), obtendo um valor de 0,77. Este valor indica concordância substancial conforme a interpretação proposta por Landis e Koch (Landis and Koch, 1977).

6.3.2 Resultados

Visando investigar as atividades de compartilhamento de *links* no fórum *Discussions* do GitHub e verificar o impacto de tais compartilhamentos no compartilhamento de conhecimentos dos projetos, foram realizados experimentos em

laboratório visando compreender os tipos de recursos mais referenciados e o motivo pelo qual os usuários do *Discussions* compartilham *links* para tais recursos. A seguir, são apresentados os resultados das questões de pesquisa norteadoras deste estudo.

- **QP1:** Como as *threads* de discussões estão lincadas (vinculadas) a outros recursos?

Para entender como as discussões estão vinculadas (lincadas) a outros recursos, primeiro é identificada a porcentagem de *threads* de discussões com *links*. Em seguida, os *links* são categorizados de acordo com sua direção (endereço) e é medida a proporção de *links* internos e externos ao GitHub. Por fim, são determinados os tipos de recursos-alvo (recursos referenciados) e calculada a frequência dos *links* em cada categoria de tipo de link.

QP1.1: *Com que frequência os usuários compartilham links no fórum de discussões?*

A Tabela 6.12 apresenta o percentual de discussões com *links*. É determinado a taxa de ocorrência de *links* na postagem principal da discussão e nos comentários. Em média, 59,12% das 29.251 *threads* de discussões compartilham *links* - Tabela 6.12, coluna 2. Os 3 principais projetos no ranque de discussões com *links* são os projetos `dotnet/csharplang`, `react-hook-form/react-hook-form` e o `gatsbyjs/gatsby`, todos com um valor percentual superior a 61%. Considerando a amostra analisada, as discussões possuem, em média, 2,24 *links* (Tabela 6.10). Investigando o local exato de compartilhamento dos *links*, verifica-se que, das discussões que compartilham *links*, quase 60% contêm *links* na postagem principal e 63,09% nos comentários (Tabela 6.12, colunas 3 e 4). Como as discussões podem ter *links* tanto na postagem principal quanto nos comentários, a soma dos valores percentuais é maior que 100%. Verificou-se ainda que, em média, 94,65% das discussões compartilham de 1 a 10 *links*, 3,53% das discussões compartilham de 11 a 20 *links* e apenas 0,28% das discussões possuem mais de 50 *links* (Tabela 6.13). Também foi identificada a ocorrência de *links* repetidos na mesma *thread* de discussão (por exemplo, projeto `Next.js` - discussão #12414).

QP1.2: *Qual é a proporção de links internos e externos do GitHub?*

Tabela 6.12: Porcentagem de *threads* de discussões com *links*

Repositório	**%*Threads* de discussões com *Link***	***Threads* de discussões com *links***	
		%*Threads* de discussões com *Link* na postagem principal	**%*Threads* de discussões com *Link* nos comentários**
dotnet/csharplang	80,93%	46,28%	88,08%
gatsbyjs/gatsby	64,70%	73,93%	60,85%
laravel/framework	44,37%	72,19%	42,64%
livewire/livewire	62,48%	39,96%	81,58%
prisma/prisma	60,27%	39,25%	76,41%
react.../react-hook-form	67,52%	73,16%	55,72%
TanStack/react-table	45,45%	66,74%	49,06%
tailwindlabs/tailwindcss	59,28%	63,07%	59,94%
vercel/vercel/	50,77%	60,70%	57,76%
vercel/next.js	55,42%	63,97%	58,89%
média	**59,12%**	**59,93%**	**63,09%**

Tabela 6.13: Porcentagem de *links* compartilhados nas discussões.

	***#links* por discussões**					
Repositório	**1—10**	**11—20**	**21—30**	**31—40**	**41—50**	**>50**
dotnet/csharplang	77,17%	14,13%	5,11%	1,90%	0,36%	1,31%
gatsbyjs/gatsby	90,63%	5,56%	1,39%	1,11%	0,64%	0,64%
laravel/framework	98,25%	1,16%	0,29%	None*	None*	0,29%
livewire/livewire	95,74%	3,64%	0,52%	0,08%	None*	None*
prisma/prisma	97,75%	0,94%	0,58%	0,43%	0,21%	0,07%
react.../react-hook-form	98.17%	1,35%	0,41%	None*	None*	0,05%
tanStack/react-table	98,60%	1,39%	None*	None*	None*	None*
tailwindlabs/tailWindCss	97,40%	2,11%	0,24%	0,14%	0,04%	0,04%
vercel/vercel	96,94%	2,58%	0,35%	None*	None*	0,11%
vercel/next.js	95,83%	2,48%	0,99%	0,27%	0,11%	0,29%
média	**94,65%**	**3,53%**	**0,99%**	**0,39%**	**0,13%**	**0,28%**

* Não foi identificada a ocorrência de *links* nessa faixa.

A Figura 6.6 apresenta a porcentagem de *links* internos e externos por repositório analisado. Em média, 62,17% dos *links* compartilhados são internos e 37,82% são externos. Esses resultados indicam que os usuários do *Discussions* referenciam recursos hospedados tanto dentro quanto fora da plataforma GitHub para dar suporte às suas discussões. Logo, o conhecimento disseminado por meio de *links* compartilhados ultrapassa o limite da plataforma GitHub. Porém, em alguns projetos, o número de *links* internos pode ser oito vezes maior que o número de *links* externo, por exemplo, o projeto `dotnet/csharplang`. Por outro lado, observa-se que a proporção de *links* externos no projeto `react-hook-form/react-hook-form` é 1,4 vezes maior que a interna. Os resultados endossam a singularidade das comunidades OSS. Com relação à proporção de *links* internos e externos do GitHub, não se pode estabelecer um comportamento padrão para compartilhamento de *links* nas *threads* de discussões.

QP1.3: *Quais são os tipos de recursos vinculados?*

Visando conhecer o destino dos *links* e o tipo de informação que eles referenciam, os *links* foram categorizados de acordo com seu endereço URL. Os *links* internos foram subdivididos em três categorias, as quais foram subdivididas em sete subcategorias (Figura 6.8). Os *links* externos foram subdivididos em oito subcategorias (Figura 6.8). As categorias emergiram a partir de uma análise manual dos *links* extraídos. São elas:

Links internos. Os *links* internos são aqueles sob o domínio GitHub ("github.com"). Esta categoria é dividida em três subcategorias:

1. *@-mentions*. A categoria *@-mentions* surge a partir da funcionalidade de referenciamento disponibilizada pela plataforma GitHub. A funcionalidade detecta referências prefixadas com "@" e as converte em *links* automaticamente (Chopra et al., 2021).

2. *Within-repo*. Esta categoria agrupa *links* para recursos hospedados no repositório do projeto. Para mapear a extensão de tais *links* estes foram classificados em *Issues*, *Discussions*, *Wiki*, *Code* e *Pull Requests* (PRs).

   Code. Esta categoria agrupa os *links* para unidades de código dentro dos repositórios (*branches*, *hooks*, *code containers*, *libraries* e *elements*). Os *links* desta categoria foram identificados através de expressões regulares que

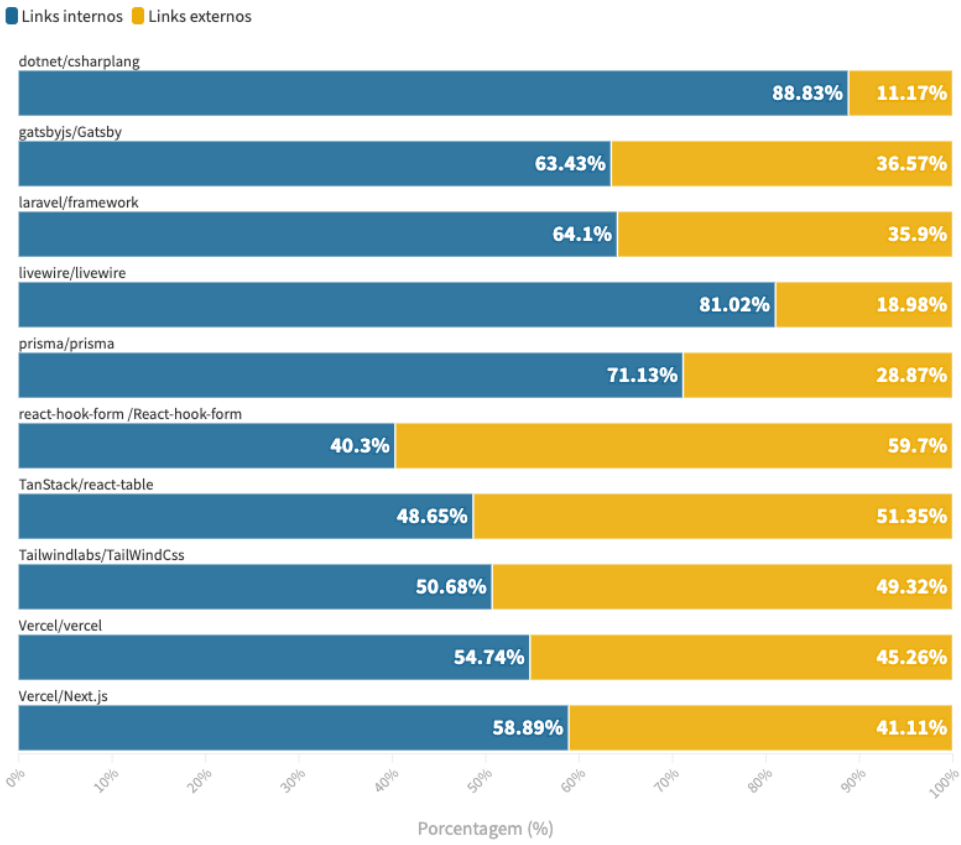

Figura 6.6: Proporção de *links* internos e externos

casam com o endereço URL do repositório (domínio do repositório), seguido pelas palavras-chave `commit`, `tree/`, `blob/` ou `releases/`.

Issues. Esta categoria agrupa os *links* para *issues* dentro do repositório. Para isso, foi aplicada uma expressão regular para identificar os *links* que correspondem ao endereço URL do repositório (domínio do repositório), seguido pela palavra-chave `issues/`.

Discussions. Esta categoria agrupa os *links* para *threads* de discussões dentro do repositório. Detecta-se esse tipo de *link* aplicando uma expressão regular para identificar os *links* que correspondam ao endereço URL do repositório do repositório de origem, seguido pela palavra-cheve `discussions/`.

Pull Requests (PRs). Esta categoria agrupa os *links* para PRs. Para identificar esses *links*, também aplica-se expressões regulares procurando por *links* que correspondam ao endereço URL do repositório, seguido da palavra-chave `pull/`.

Wiki. Esta categoria agrupa os *links* para páginas Wiki do repositório. Para identificar tais *links*, aplica-se expressões regulares procurando por *links* que correspondem ao endereço URL do repositório, seguido da palavra-chave `wiki/`.

3. *Cross-repo*. A categoria *cross-repo* agrupa os *links* que referenciam recursos hospedados fora do repositório do projeto, porém na plataforma GitHub. Para tanto, esta categoria foi dividida em duas subcategorias:

   Within-organization. Esta categoria agrupa os *links* que referenciam recursos hospedados sob a URL da organização do projeto. Por exemplo, um link do projeto `Next.js` para o projeto `vercel` é classificado como *within-organization*, pois ambos os projetos estão sob o domínio da mesma organização, a `vercel` (Tabela 6.10).

   Cross-organization. Os *links* desta categoria referenciam recursos hospedados no GitHub porém fora do domínio da organização do repositório de origem.

Links externos. Para mapear os tipos de recursos frequentemente referenciados

fora da plataforma GitHub, esta categoria foi subdividida em oito subcategorias, conforme a seguir:

1. *Documentation*. Esta categoria agrupa os *links* para a documentação do projeto que está hospedada fora da plataforma GitHub. Para extrair esses *links*, primeiro foi identificado o endereço URL da documentação do projeto. Esta identificação foi feita manualmente a partir dos arquivos README.md dos projetos. Em seguida, definiu-se uma expressão regular para buscar os *links* que correspondem ao domínio da documentação. Frequentemente, os documentos podem ser encontrados no endereço "https://<project_site>/docs/*".

2. *on-line code editor/Playground*. Os *links* classificados nesta categoria apontam para trechos de código compartilhados em ferramentas *on-line*, como, por exemplo, *codeSandBox*[13], *CodePen*[14] e *TypeScript Playground*[15].

3. *Project Website* (Site do projeto). Os *links* desta categoria apontam para o site do projeto hospedado fora da plataforma GitHub. A URL do site do projeto também foi identificada, manualmente, a partir do arquivo README.md de cada projeto analisado.

4. *Stack Overflow*. Esta categoria agrupa *links* para perguntas do Stack Overflow [16]. Perguntas da plataforma *Stack Overflow* são frequentemente referenciadas nas postagens de discussão. Tais *links* também foram identificados por expressões regulares usadas para casar a URL dos *links* com o domínio "stackoverflow.com/".

5. *Twitter*. O *Twitter*[17] é uma plataforma de mídia social popular entre os desenvolvedores de software. Como a plataforma impacta o dia-a-dia de trabalho dos desenvolvedores de software (Fang et al., 2020), uma subcategoria específica foi definida para agrupar *links* que referenciam postagens no *Twitter*.

[13]https://codesandbox.io/
[14]https://codepen.io/
[15]https://www.typescriptlang.org/play
[16]https://stackoverflow.com/
[17]https://twitter.com/

6. *Video platforms* (Plataformas de vídeo). Esta categoria surge agrupando os *links* para plataformas *on-line* de compartilhamento de vídeos, como o *YouTube*[18].

7. *Wikipedia*. A *Wikipedia* é uma enciclopédia *on-line* de uso geral. Foi identificado que os usuários do *Discussions* também compartilham *links* para páginas da *Wikipedia*.

8. *Outros*. Esta categoria engloba os *links* que não se enquadram nas demais categorias de *links* externos.

Após a identificação das categorias, um *script* Python foi usado para classificar automaticamente os *links* extraídos. A Tabela 6.14 mostra a frequência dos *links* extraídos de acordo com seu endereço URL. Através da análise quantitativa realizada, observou-se que:

- *@-mentions* são os tipos de *links* mais frequentes entre os *links* internos, 49,81%. Os *links within-repo* e *cross-repo* representam 28,35% e 21,84%, respectivamente. Os *links* do tipo *@-mentions* diferem de outros, pois vinculam, principalmente, a perfis de usuário. No entanto, os usuários do *Discussions* também usam notações *@-reference* para vincular às páginas de repositórios hospedados no GitHub e a recursos da linguagem de programação.

- Os *links* para *issues* são os mais frequente. No entanto, o número de *links* para *issues* extraídos do projeto `dotnet/csharplang` impacta o valor da soma. Com base no arquivo README.md do projeto, a comunidade `dotnet/csharplang` abre *issues* para propor novos recursos para a linguagem C# e usa o fórum *Discussions* para discutir as propostas. Esse comportamento da comunidade justifica o alto número de *links* entre as *threads* de discussões e *issues* no projeto. A Figura 6.7 mostra a distribuição de *links* do tipo *within-repo*.

- Os números apresentados na Tabela 6.14 mostram que os usuários frequentemente compartilham *links* entre postagens de discussão. Ao considerar

[18]https://www.youtube.com/

links do tipo *within-repo*, a frequência de *links* do tipo *discussions* sobressai em 10 dos 5 projetos analisados. Esse resultado sugere que os usuários estão cientes (conhecem) sobre as discussões que ocorrem no fórum. Além disso, o número pode ser interpretado como uma evidência de que os membros das comunidades OSS, consciente ou inconscientemente, promovem a reutilização do conhecimento do projeto, compartilhando *links* para postagens de discussão que consideram de alguma forma relevantes para o tópico da discussão onde o link foi compartilhado.

- Em relação aos *links* do tipo *code*, foram encontrados *links* para *commits*, *branches*, *packages*, arquivos de código-fonte, linhas de código-fonte, *releases* e recursos que dão suporte o uso de códigos-fonte, como arquivos contendo exemplos de codificações.

- Ao comparar a frequência de *links* do tipo *within-organization* com *links* do tipo *cross-organization*, os resultados sugerem que os usuários citam recursos fora do domínio da organização do projeto de origem. Por outro lado, a taxa de porcentagem de *links* do tipo *within-organization* do projeto `dotnet/csharplang` é quatro vezes maior que a dos *links* do tipo *cross-organization*. Aproximadamente 80% dos *links* do tipo *witihn-organization* do projeto `dotnet/csharplang` são para o repositório `dotnet/roslyn`. Esse resultado sugere um relacionamento próximo entre ambos os repositórios. O arquivo README.md `dotnet/csharplang` indica que o repositório `dotnet/roslyn` contém a implementação de referência da linguagem C#, sendo uma fonte de informação relevante para o projeto `dotnet/csharplang`. Este valor evidencia a eficácia da disseminação do conhecimento do projeto `dotnet/csharplang` por meio do fórum *Discussions* e, ainda, que a disseminação do conhecimento do projeto ultrapassa os limites do repositório.

- Os *links* externos (*external*) referenciam, frequentemente, a documentação do projeto. Esse resultado corrobora a falta de referências explícitas às páginas do GitHub Wiki (*within-repo*). Somente o projeto `vercel/vercel` tem uma página Wiki que possui um *links* para a página de documentação do projeto localizada fora da plataforma.

- A documentação dos projetos nem sempre está na URL "https://<project_site>/docs/*". É por isso que a Tabela 6.14 possui o valor zero para a frequência do tipo de link *documentation* para projetos `react-hook-form/react-hook-form` e `dotnet/csharplang`.

- Os usuários do *Discussions* compartilham *links* para editores de código *on-line* ou *playgrounds* para compartilhar trechos de código por eles implementados. A maioria dos *links* externos (*external*) dos projetos `react-hook-form/react-hook-form` e `tanStack/react-table` (69% e 49%, respectivamente) referenciam editores de código *on-line*. Foram encontrados *links* para os editores *Codesandbox*, *CodePen*, *laravelplayground.com*, *play.tailwindcss.com*, *jsfiddle.net*, *stackblitz.com*, *sharplab.io* e *dotnetfiddle.net*.

- Também foi identificado que os usuários compartilham *links* para plataformas de compartilhamento de vídeos. Vídeos técnicos podem conter tópicos introdutórios ou mais complexos de engenharia de software. Estes são comumente utilizados por desenvolvedores com objetivo de aprender ou aperfeiçoar habilidades técnicas (Poche et al., 2017; Ponzanelli et al., 2017).

- Na subcategoria "outros", identificou-se *links* para a plataforma *Spectrum*: um ambiente utilizado pelas comunidades do GitHub para tirar dúvidas, reportar *bugs* e discutir com a equipe. No momento de desenvolvimento desta pesquisa, a plataforma *Spectrum* era uma fonte de informação somente para leitura. Ela está sendo descontinuada em favor do fórum de discussões do GitHub.

- Na subcategoria "outros", também foram encontrados *links* para documentação e site de terceiros, páginas com descrição de empregos, canais do *Slack*, páginas de tutoriais, comunidades DEVs, plataformas de aprendizado *on-line*, perfis do *LinkedIn*, etc.

- Por fim, foram identificados a ocorrência de *links* inativos. Os usuários podem compartilhar *links nulos* para exemplificar a ocorrência de um *bug* ou até mesmo solicitar um recurso referente ao uso de URLs. *links* inativos não referenciam páginas válidas da Web, por exemplo, "http://example.ca/fr".

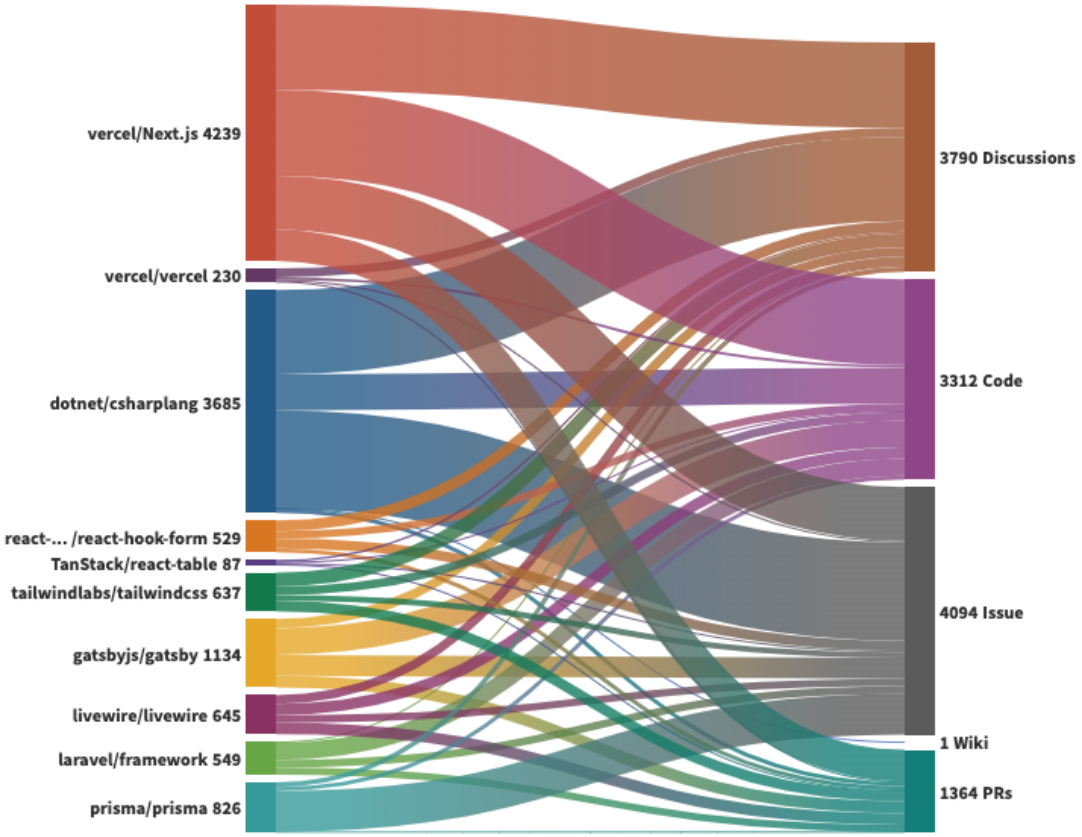

Figura 6.7: Distribuição de *links* da categoria *within-repo*.

Respondendo à QP1 deste estudo: Como as *threads* de discussões estão lincadas (vinculadas) a outros recursos?

Usuários do *Discussions* frequentemente compartilham *links* ao criarem *threads* de discussões. Eles compartilham *links* para recursos internos e externos ao GitHub. Considerando a amostra de dados analisada (Tabela 6.10), as *threads* de discussão têm em média 2,24 *links*. Embora a maioria dos *links* no repositório faça referência a *issues*, os usuários frequentemente fazem referência a outras discussões. Os *links* são, geralmente compartilhados, para páginas e documentação externa do projeto. Nos dados analisados, *links* para páginas Wiki da plataforma GitHub são raros.

- **QP2:** Quais são as intenções dos usuários ao compartilhar *links* nas posta-

Tabela 6.14: Frequência de *links* por categoria

LINKS INTERNOS

	Within-repo						Cross-repo		#Links internos
	@-mention	**Code**	**Issues**	**PRs**	**Discussions**	**Wiki**	**Within-organization**	**Cross-organization**	
dotnet/csharplang	9517 [58.63%]	596 [3.67%]	1626 [10.02%]	67 [0.41%]	1395 [8.59%]	1 [0.01%]	2447 [15.08%]	583 [3.59%]	16232
gatsbyjs/gatsby	1406 [41.20%]	440 [12.89%]	344 [10.08%]	195 [5.71%]	155 [4.54%]	0 [0.00%]	48 [1.41%]	825 [24.17%]	3413
laravel/framework	326 [29.50%]	283 [25.61%]	126 [11.40%]	120 [10.86%]	20 [1.81%]	0 [0.00%]	58 [5.25%]	172 [15.57%]	1105
livewire/livewire	2150 [70.06%]	187 [6.09%]	123 [4.01%]	188 [6.13%]	147 [4.79%]	0 [0.00%]	22 [0.72%]	252 [8.21%]	3069
prisma/prisma	1552 [55.21%]	64 [2.28%]	675 [24.01%]	8 [0.28%]	79 [2.81%]	0 [0.00%]	192 [6.83%]	241 [8.57%]	2811
react.../react-hook-form	1086 [60.57%]	124 [6.92%]	164 [9.15%]	65 [3.63%]	176 [9.82%]	0 [0.00%]	25 [1.39%]	153 [8.531%]	1793
tanStack/react-table	249 [55.09%]	16 [3.54%]	26 [5.75%]	12 [2.65%]	33 [7.30%]	0 [0.0%]	0 [0.0%]	116 [25.66%]	452
tailwindlabs/tailWindCss	1188 [43.85%]	140 [5.17%]	96 [3.54%]	174 [6.42%]	227 [8.38%]	0 [0.0%]	104 [3.84%]	780 [28.79%]	2709
vercel/vercel	637 [48.77%]	41 [3.14%]	26 [1.99%]	16 [1.23%]	147 [11.26%]	0 [0.0%]	87 [6.66%]	352 [26.95%]	1306
vercel/next.js	3959 [34.66%]	1421 [12.44%]	888 [7.78%]	519 [4.54%]	1411 [12.35%]	0 [0.0%]	231 [2.02%]	2992 [26.20%]	11421
Total	**22,070**	**3,312**	**4,094**	**1,364**	**3,790**	**1**	**3,214**	**6,466**	**44,311**
%	49.807	7.474	9.239	3.078	8.553	0.002	7.253	14.592	

LINKS EXTERNOS

	Project Website		Stack Overflow	Videos Platform	Twitter	Online Code Editor/ Playground	Wikipedia	other	#Links externos
	/docs/	**other**							
dotnet/csharplang	—*	—*	159 [7.79%]	25 [1.22%]	22 [1.08%]	296 [14.50%]	113 [5.54%]	1426 [69.87%]	2041
gatsbyjs/gatsby	437 [22.21%]	428 [21.75%]	56 [2.85%]	19 [0.97%]	24 [1.22%]	26 [1.32%]	2 [0.10%]	976 [49.59%]	1968
laravel/framework	221 [35.70%]	38 [6.14%]	43 [6.95%]	8 [1.29%]	16 [2.58%]	3 [0.48%]	4 [0.65%]	286 [46.20%]	619
livewire/livewire	213 [29.62%]	33 [4.59%]	23 [3.20%]	25 [3.48%]	43 [5.98%]	77 [10.71%]	3 [0.42%]	302 [42.00%]	719
prisma/prisma	620 [54.34%]	52 [4.56%]	38 [3.33%]	36 [3.16%]	18 [1.58%]	1 [0.09%]	0 [0.0%]	376 [32.95%]	1141
react.../react-hook-form	—*	452 [17.02%]	32 [1.20%]	26 [0.98%]	18 [0.68%]	1843 [69.39%]	1 [0.04%]	284 [10.69%]	2656
tanStack/react-table	94 [19.71%]	1 [0.21%]	19 [3.98%]	1 [0.21%]	0 [0.0%]	238 [49.90%]	1 [0.21%]	123 [25.79%]	477
tailwindlabs/tailWindCss	765 [29.02%]	27 [1.02%]	76 [2.88%]	80 [3.03%]	45 [1.71%]	544 [20.64%]	1 [0.04%]	1098 [41.65%]	2636
vercel/vercel	283 [26.20%]	195 [18.06%]	16 [1.48%]	3 [0.28%]	12 [1.11%]	0 [0.0%]	0 [0.0%]	571 [52.87%]	1080
vercel/next.js	2031 [25.47%]	310 [3.89%]	255 [3.20%]	62 [0.78%]	91 [1.14%]	214 [2.68%]	18 [0.23%]	4992 [62.61%]	7973
Total	**4,664**	**1,536**	**717**	**285**	**289**	**3,242**	**143**	**10,434**	**21,310**
%	21.88	7.20	3.36	0.13	1.35	15.21	0.67	48.96	

* We could not determine this value.

gens de discussão?

Para responder a QP2, foi realizada uma análise qualitativa em 1.106 *links* extraídos do projeto `Next.js`. Os *links* foram classificados manualmente os nas categorias de tipos de relacionamento derivadas (descritas na Tabela 6.11). Visando dar suporte a classificação feita, foi mensurado o valor do consenso da classificação

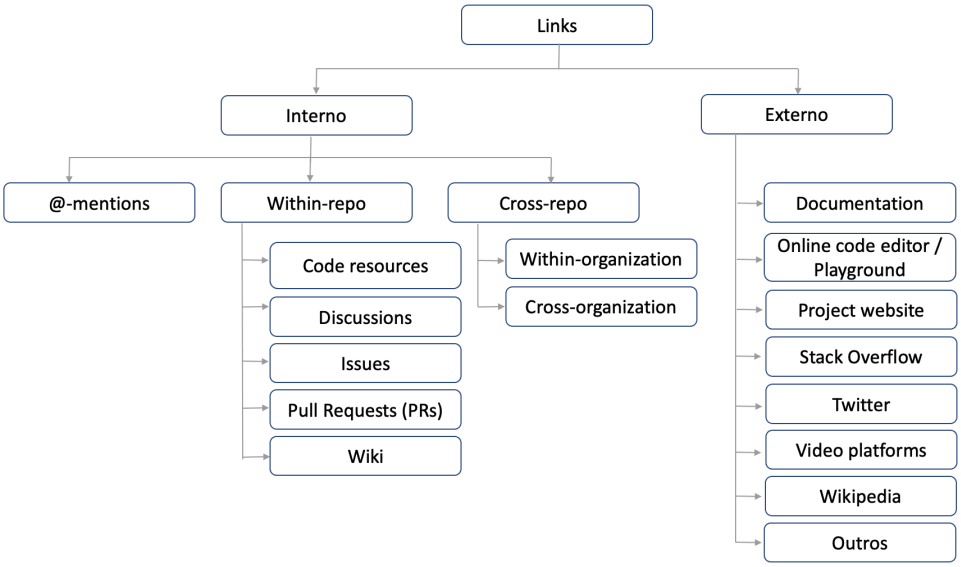

Figura 6.8: Tipos de recursos referenciados

feita por dois pesquisadores que, de forma independente, avaliaram uma amostra de *links* randomicamente selecionada dentre os 1.106 *links* analisados. O valor do consenso foi mensurado com o coeficiente de Cohen Kappa (Cohen, 1960), obtendo valor de 0,78. Esse valor indica concordância substancial (*substantial agreement*) conforme a interpretação proposta por Landis e Koch (Landis and Koch, 1977). Além disso, citações de usuários extraídas das *threads* de discussões são apresentadas com objetivo de endossar os resultados.

1. Relacionamento do tipo Dependent.

Relacionamentos do tipo *Dependent* são raros. Dos 1.106 *links* analisados, foram identificados três *links* nessa categoria. Os usuários compartilharam *links* para PRs para destacar que a solução para o problema discutido depende do *merge* de um PR — "*(O RFC) não será mais necessário quando #36436 estiver disponível no estável em Next.js 13*" (`Next.js` #8207).

2. Relacionamento do tipo Duplicate.

Os usuários do *Discussions* duplicam postagens de discussão em diferentes plataformas com objetivo de obter vários *feedbacks* sobre suas perguntas — "*Eu também fiz essa pergunta no SO (Stack Overflow), mas não obtive nenhuma resposta lógica*" (`Next.js` #24452).

Os usuários também duplicam discussões em diferentes plataformas porque têm dúvidas sobre o melhor lugar para criá-las — "*Eu não sabia onde postar*" (`Next.js` #20502). Além disso, os usuários duplicam discussões para enfatizar sua necessidade de ajuda — "*Estou basicamente fazendo a mesma pergunta aqui: #24659*" (`Next.js` #23737). Também descobrimos que os usuários do *Discussions* duplicam intencionalmente o conteúdo das discussões no/do *Stack Overflow* — "*Isto é postado também no Stack Overflow...*" (`Next.js` #22715). Os usuários ainda podem triplicar o conteúdo das discussões — "*Obrigado por fazer esta pergunta. Percebi que suas postagens no Reddit e no Stack Overflow também não têm respostas...*" (`Next.js` #35554). Além disso, verificou-se que os usuários também podem destacar a ocorrência de duplicadas ocorridas entre postagens de discussão — "*Estou basicamente fazendo a mesma pergunta aqui: #24659*" (`Next.js` #23737).

3. Relacionamento do tipo Relevant.

O relacionamento do tipo *Relevant* é o mais frequente entre os *links* analisado, alcançando 48% dos *links* do classificados como *within-repo*, subcategoria *Discussions*. Os usuários do fórum compartilham *links* para outras postagens de discussão que versam sobre *features* similares, que compartilham o mesmo tópico ou que abordam *bugs* relacionados — "*Acredito que a solicitação é bastante semelhante a esta: #34567*" (`Next.js` #34543).

Os usuários frequentemente compartilham *links* entre postagens de discussão para identificar as discussões relacionadas. Além disso, os usuários referenciam as discussões do tipo *Relevant* para (1) destacar RFCs (*Request for Comments*) já criados e que possuem tópicos semelhantes ao tópico da discussão — "*já temos RFC nº 8981 semelhante*" (`Next.js` #16854) —, (2) chamar a atenção para outras pessoas que enfrentaram problemas semelhantes — "*E eu procuro pessoas que também enfrentaram esse problema*" (`Next.js` #24437) — e (3) preencher campos de *templates* com discussões similares já conhecidas — "*Relacionadas*" (`Next.js` #11822).

4. Relacionamento do tipo Reference.

Os *links* classificados com sendo do tipo *Reference* são os mais frequentes. Entre os 1.106 *links* analisados, 78% referenciam fontes de informação que ajudam a esclarecer o conteúdo da postagem. Os usuários compartilham *links* para a

Wikipédia para conceituar termos e esclarecer as ideias propostas nas discussões. Embora não seja uma mídia de propósito restrito à programação, os pesquisadores de engenharia de software usam as páginas da *Wikipedia* para treinar modelos de *word embeddings* (Mishra and Sharma, 2021), detectar ambiguidade na engenharia de requisitos (Ferrari and Esuli, 2019) e estudar as atividades dos *bots* (Zheng et al., 2019). Os usuários também compartilham *links* para a *Wikipedia* para questionar a corretude da documentação do projeto — "*Na minha opinião, isso certamente não é uma função pura (e de fato na visão da Wikipedia), pois viola o seguinte:...*" (`Next.js` #20015).

Os usuários do *Discussions* também referenciam editores de código *on-line* com objetivo de explicar os *bugs* que estão enfrentaram e permitir que outras pessoas os reproduzam. Ao utilizarem editores de código *on-line*, os usuários evitam copiar e colar trechos de código no corpo da discussão, fornecendo um código executável.

Além disso, os usuários referenciam a documentação do projeto para enfatizar que não conseguiram encontrar uma solução para o problema, comunicar que estão seguindo um tutorial oficial e declarar que a documentação do projeto precisa de atualizações — "*Não consigo encontrar nenhum documento oficial*" (`Next.js` #25091), "*Estou usando NextJS + Electron e tentando seguir <link> mas assim que adiciono ..., as coisas começam a falhar*" (`Next.js` #10829), "*Acho que também seria bom adicionar exemplos de como customizar... aos documentos oficiais*" (`Next.js` #24115).

Quando os usuários não encontram as informações desejadas na documentação do projeto, estes podem recorrer aos fóruns de perguntas e respostas voltados para a comunidade de programação (PCQA), como *Stack Overflow* — "*Não consigo encontrar nenhum documento oficial. Então eu sigo este tópico: https://stackoverflow.com/questions/60459747/connect-apollo-with-mongodb*" (`Next.js` #25091).

Há usuários que preferem criar uma pergunta do *Stack Overflow* e referenciá-la na discussão. Nesse caso, os usuários detalham o problema em uma pergunta do *Stack Overflow* e criam uma discussão para declarar que precisam de ajuda sobre o assunto. Sendo assim, os usuários não duplicam a descrição dos problemas na discussão — "*Oi. Postei uma pergunta sobre como adicionar um script com onload em Next.js no Stackoverflow. Alguém poderia ajudar/comentar? Obrigado!*"

(`Next.js` #27197); "*Aqui estão os detalhes do meu bug.*" (`Next.js` #2884).

Também foi identificado que os usuários do fórum usam o link do tipo *@-mentions* para chamar a atenção dos membros da comunidade ou enfatizar um repositório específico — "*@jnv, @nevenein Usando-o [na versão 9.5.3] (not canary), sem as. Funciona muito bem!*" (`Next.js` #8207), "*Legal. Este problema de RFC pode ser finalmente encerrado. Estou certo, @Timer?*" (`Next.js` #8207).

Além disso, os usuários podem compartilhar *links* para exemplificar questões relacionadas ao próprio uso de *links*. Os usuários criam referências para esclarecer as soluções propostas, ilustrar a ocorrência de um *bug*, apresentar uma pergunta ou solicitar um novo recurso — "'*Como o nextjs faz isso https://nextjs.org/showcase e https://nextjs.org/showcase/?*" (`Next.js` #23988).

Por fim, os usuários do *Discussions* podem adicionar referências a outros recursos para pedir ajuda — "*Alguém pode me ajudar a corrigir esse problema. #21890*" (`Next.js` #23184).

5. Relacionamento do tipo Fixing Proposal.

O tipo *Fixing Proposal* é frequente entre os *links* de discussões para PRs, ocupando a segunda posição logo após o tipo *reference*. No entanto, os usuários do *Discussions* encontram possíveis soluções para problemas discutidos em diferentes lugares, como, *YouTube*, documentação, *Stack Overflow* — "*Verifique esta resposta [no Stack Overflow], espero que ajude você*" (`Next.js` #11351) —, *issues*, PRs e *Twitter* — "*Houve um tópico interessante no Twitter sobre isso, envolvendo... e contendo alguns pontos para possíveis soluções...*" (`Next.js` #22715).

6. Relacionamento do tipo Enhancement Proposal.

Assim como os *links* do tipo *dependent*, os *links* do tipo *enhancement proposal* são raros. Dos 1.106 *links* analisados, foram classificados apenas três *links* nesta categoria. Os usuários do fórum criam uma nova discussão com o objetivo de adicionar informações a postagens criadas anteriormente — "*Eu postei uma pergunta inicial aqui, cuja sugestão era basicamente 'use gSSP', que também é uma solução possível, mas seria bom não gerar novamente a página do zero toda vez que é basicamente estático.*" (`Next.js` #17111). Nos três casos, foi identificado que os usuários não invalidaram as discussões originais.

Além dos seis propósitos mencionados acima, foi identificado que os usuários

podem referenciar várias vezes o mesmo recurso para expressar diferentes intenções. Na mesma discussão, foram encontramos *links* duplicados, os quais foram classificados como sendo do tipo *Duplicate* e *Fixing Proposal*, respectivamente.

1. *Duplicate*: "*Postei esta pergunta aqui (vercel/swr#362) 2 dias atrás, mas não recebi nenhuma resposta, por isso postei aqui.*" (`Next.js` #12414).

2. *Fixing Proposal*: "*Foi respondida aqui: vercel/swr#362*" (`Next.js` #12414).

Também foi identificado um motivo diferente para o compartilhamento de *links*: identificar o local correto para uma discussão. Esse tipo de link foi chamado de relacionamento do tipo *localization* — "*Sugiro adicionar sua configuração a #30174 para que possa ser priorizada*" (`Next.js` #30862), "*Se você é uma empresa que deseja publicar uma função, use #32937*" (`Next.js` #32938).

Respondendo à QP2 deste estudo: Quais são as intenções dos usuários ao compartilhar *links* nas postagens de discussão?

Embora a maioria dos *links* compartilhados referenciem recursos utilizados para melhor esclarecer a discussão, os usuários do *Discussions* também compartilham *links* para identificar dependências de problemas, destacar duplicações, apontar tópicos relacionados, sugerir propostas de correção e identificar propostas de aprimoramento. Descobrimos que os *links* do tipo *dependent* e de *enhancement proposal* são raros. Os usuários podem duplicar intencionalmente suas postagens no *Stack Overflow*. *links* do tipo *fixing proposal* podem referenciar diferentes recursos, como vídeos, documentação, *Stack Overflow*, *issues*, PRs e *Twitter*.

6.3.3 Discussões

Os usuários do fórum *Discussions* frequentemente compartilham *links* em *threads* de discussões. Aproximadamente 59% das discussões da amostra analisada possui, pelo menos, 1 link compartilhado. Os usuários compartilham *links* para recursos que julgam relevantes para o tópico que estão discutindo. Além disso, os usuários que adicionam comentários às *threads* de discussões também compartilham *links* para dar suporte ao raciocínio descrito nos comentários. Sendo assim, verifica-se

que os usuários do *Discussions* compartilham *links* em *threads* de discussões para apoiar o entendimento correto de conhecimentos do projeto entre os membros da comunidade OSS.

Em contraste com os usuários do *Modern Code Review* (MCR) (Wang et al., 2021), os usuários do *Discussions* frequentemente compartilham *links* que são direta ou indiretamente relacionados aos projetos. Eles compartilham *links* para recursos hospedados fora e dentro do domínio do repositório, assim como, para recursos hospedados fora da plataforma GitHub. A diferença entre a proporção de *links* internos e externos ao GitHub mostra que os usuários do *Discussions* ultrapassam os limites da plataforma para promover o compartilhamento de conhecimento dos projetos.

Foram observadas semelhanças entre as categorizações de tipos de link aqui encontradas (Figura 6.8) e a taxonomia de tipos de referência criada por Chopra et al. (2021). As subcategorias *issues* e PRs, aqui identificadas, são semelhantes as também definidas por Chopra et al. (2021). No entanto, visando compreender as atividades de compartilhamento de *links* nos repositórios de origem, primeiramente, neste estudo, os *links* foram filtrados conforme as categorias *within-repo* e *cross-repo*. Por exemplo, considerando o link "https://github.com/vercel/next.js/issues/37793" extraído de uma discussão do projeto `Next.js`, o link é classificado na categoria *within-repo*, subcategoria *issues*. No entanto, o link "https://github.com/vercel/ next-plugins/issues/507", também extraído de uma discussão do `Next.js`, é classificado na categoria *cross-organization* porque o domínio do repositório do link difere do repositório `Next.js`.

O conceito da subcategoria *code resources* mescla as definições das categorias "*Version Control Entity*" e "*Source Code*" de Chopra et al. (2021). Segundo os autores, a categoria "*Version Control Entity*" compreende as entidades relacionadas ao sistema de controle de versão do software (*branches*, *hooks* e conflitos de *merge*). Além disso, a categoria "*Source code*" inclui quaisquer artefatos no projeto que representem o código-fonte (*code containers*, *code elements*, *code libraries*, API, teste). Por fim, a subcategoria "*external documentation*" aqui apresentada é semelhante à subcategoria "*Client Documentation*" definida por Chopra et al. (2021).

Além disso, assim como os usuários do *Modern Code Review* (Wang et al., 2021)

e os usuários de repositórios de código-fonte (Hata et al., 2019), os usuários do *Discussions* também compartilham *links* para as perguntas do *Stack Overflow* e a página web do projeto de software. Por fim, assim como ocorre nos repositórios de código-fonte (Hata et al., 2019), também foi observada a ocorrência de *links* nulos. No entanto, observou-se que os usuários do *Discussions* podem compartilhar intencionalmente *links* nulos ou quebrados para descrever problemas de URL relacionados ao contexto dos projetos.

Embora tenhamos conduzido uma análise de métodos mistos e selecionado uma amostra estatisticamente significativa no estudo qualitativo, este estudo pode apresentar limitações, como (1) a amostragem pode não conter todos os *links* compartilhados nas *threads* de discussão, (2) assumiu-se que todos os links extraídos impactam a disseminação do conhecimento do projeto, (3) classificação manual para identificar as intenções dos usuários por trás do compartilhamento de links e (4) investigação da disseminação do conhecimento do projeto sob a ótica do compartilhamento de *links*. Por fim, as discussões podem variar significativamente entre diferentes comunidades, então não se pode garantir que as descobertas se apliquem a todos os projetos hospedados no GitHub. Além disso, a análise foi feita em dados de apenas 10 repositórios do GitHub com foco em codificação, embora muitos repositórios estejam disponíveis[19].

Contudo, conforme resultados apresentados, os usuários do fórum *Discussions* frequentemente compartilham *links* entre postagens de discussões para identificar as *discussões relacionadas*. Sendo assim, pode-se afirmar que a rede de *links* formada pelas discussões dos membros de comunidades *open source*, no GitHub *Discussions*, podem ser utilizadas na construção de bases de dados rotuladas com pares de discussões relacionadas e, como consequência, viabilizar a otimização do modelo de aprendizagem profunda utilizado pelo *RK-Component*. Contudo, investigações mais detalhadas são necessárias e constituem direções futuras desta pesquisa.

[19]https://github.com/

6.3.4 Ameaças à Validade

Embora tenha sido conduzida uma análise quantitativa e uma qualitativa dos *links* compartilhadas no fórum, esta pesquisa pode apresentar limitações.

Destacam-se dois eventos que podem afetar a validade do estudo em relacionadas à etapa de extração dos *links*: (1) a amostragem pode não conter todos os *links* compartilhados nas discussões e (2) foi assumido que todos os *links* extraídos impactam a disseminação do conhecimento dos projetos.

Além disso, pode haver limitações em relação a classificação manual feita para identificar as intenções dos usuários por trás do compartilhamento de *links*. Para minimizar essa limitação e reduzir vieses, foi analisada uma amostra representativa de *links* de projetos `Next.js` e, ainda, calculada a taxa de concordância entre avaliadores do processo de classificação. O projeto `Next.js` se destaca por apresentar uma alta taxa de utilização do *Discussions*. Outra ameça identificada foi a investigação da disseminação do conhecimento do projeto na perspectiva do compartilhamento de *links*. É sabido que existem diferentes formas de apoiar a disseminação do conhecimento nas comunidades OSS.

Por fim, os resultados apresentados podem não generalizar. As atividades de compartilhamento de *links* e as postagens de discussões podem variar significativamente entre diferentes comunidades, por isso não se pode garantir que as descobertas reportadas se aplicam a todos os projetos hospedados no GitHub. Além disso, foram analisados apenas dados de 10 repositórios que se concentram em codificação (principalmente em JavaScript e TypeScript). Tais repositórios foram selecionados com base nas recomendações da equipe de engenharia do GitHub e no número de postagens de discussão existente no fórum *Discussions*. A amostra analisada compreende 29.251 postagens de discussão das quais foram extraídos 65.621 *links*. No entanto, são necessários novos estudos para avaliar os resultados em diferentes comunidades com o objetivo de compreender as semelhanças e diferenças entre as comunidades.

6.4 Considerações

Este capítulo apresentou o *RK-Component*, o terceiro componente do *framework Miner4DevTeam*. O *RK-Component* tem por objetivo apoiar a determinação do *Project Related Knowledge* (Seção 3). O foco do componente é detectar postagens relacionadas no GitHub *Discussions* e apoiar mantenedores na tarefa trabalhosa de detectar manualmente postagens relacionadas.

Este componente usa um modelo, de aprendizagem profunda genérico, de acesso público, pré-treinado pelo Sentence-BERT (SBERT) para calcular *embeddings* de sentenças semanticamente significativas de postagens e a similaridade de cosseno para comparar pares de postagens. Os modelos de aprendizado de máquina disponíveis publicamente trazem flexibilidade à abordagem. À medida que os pesquisadores lançam modelos melhores, pode-se atualizar o *RK-Component*. Além disso, o uso de modelos de uso geral oferece vantagens como (1) não há necessidade de estruturas computacionais complexas locais para treinamento, validação e teste de modelos, (2) não há necessidade de parametrização do modelo e (3) não há necessidade de retreino do modelo, revalidação e novo teste sempre que o contexto mudar (Polyzotis et al., 2017; Zhou et al., 2017; Schelter et al., 2018; Lee and Shin, 2020). O dinamismo com que as comunidades no GitHub crescem justifica o uso de modelos genéricos de aprendizado de máquina.

O componente foi avaliado em discussões públicas coletadas em três comunidades OSS. No total, foram avaliadas a similaridade semântica de 16.048 postagens. Três mantenedores de OSS e três pesquisadores de SE julgaram as postagens relacionadas detectadas. Os resultados mostram que pode-se usar um modelo de aprendizado de máquina profunda de uso geral aplicado a problemas de NLP para detectar postagens relacionadas no *Discussions*. O *RK-Component* alcançou uma taxa de precisão média de 92,47%, considerando os 10 pares de discussão mais semelhantes para cada postagem de discussão no conjunto de dados D ($K = 10$). Com base em uma amostra rotulada, foi avaliada a taxa de *recall* do componente, que atingiu aproximadamente 67% para o projeto `Homebrew`.

Os resultados aqui apresentados foram discutidos com a equipe de engenharia do GitHub *Discussions*. Consequentemente, a equipe de engenharia está testando algumas mudanças na interface do fórum como, por exemplo, ao criarem novas

postagens, os usuários devem confirmar que pesquisaram tópicos semelhantes antes de criar novas postagens.

Acredita-se que esta pesquisa traz futuras oportunidades para permitir a aquisição e transferência de conhecimento do projeto, pois pode fornecer aos usuários questões relacionadas ao projeto e tornando o conhecimento do projeto fácil de encontrar. Por fim, os resultados possibilitam a reutilização do conhecimento do projeto, pois os usuários poderão acessar postagens relacionadas que já foram perguntadas e respondidas. Como direcionamentos futuros visa-se a otimização do modelo e a construção de um Bot para relatar automaticamente a ocorrências de postagem relacionadas no fórum. A otimização do modelo usado poderá ser feita explorando a rede de *links* criada nas postagens de discussões. Observou-se que os usuários do *Discussions* frequentemente compartilham *links* entre postagens relacionadas, esse achado permite a criação automática de bases rotuladas com discussões relacionadas, que podem ser exploradas na otimização do modelo.

Capítulo 7

Conclusões

Este capítulo apresenta a conclusão desta tese (Seção 7.1), responde à questão de pesquisa norteadora do trabalho (Seção 7.1.1), apresenta as suas contribuições (Seção 7.2) e discorre sobre oportunidades futuras (Seção 7.3).

7.1 Conclusão

Ferramentas como *chats* e fóruns do tipo PCQA têm se tornado essenciais para as equipes de desenvolvimento de software, sejam estas distribuídas ou colocalizadas. Tais ferramentas oferecem um meio eficiente de comunicação e facilitam a colaboração entre membros de equipes, pois permitem a troca rápida de informações, compartilhamento de ideias e resolução de problemas, o que certamente contribui para o sucesso dos projetos e aprimoramento da produtividade das equipes.

Quando desenvolvedores usam tais ferramentas, discussões relevantes sobre a idealização, funcionalidades, planejamento, desenvolvimento, utilização e gerenciamento dos projetos ficam registradas nos sistemas de armazenamento de conteúdo desses recursos. Contudo, essas informações podem se tornar "perdidas", não implementadas, esquecidas ou difíceis de serem encontradas em meio a grande quantidade de mensagens trocadas pelos usuários dos canais de comunicação. Podem, ainda, já terem sido amplamente discutidas e/ou amadurecidas pelos integrantes das equipes, porém não registradas na documentação oficial do projeto, ou podem ser discutidas repetidas (duplicadas) vezes por desinformação

dos usuários. Em todos os cenários destacados, a perda e a duplicação de informações podem comprometer o compartilhamento e o reuso de conhecimentos dos projetos de software (*project knowledge sharing* e *project knowledge reuse*) e, consequentemente, o produto desenvolvido, pois a capacidade de interagir e compartilhar informações entre membros de times de desenvolvimento é apontada como requisito principal para o sucesso de projetos de software (Chen et al., 2013; Tantisuwankul et al., 2019).

Dado que o problema desta pesquisa é de cunho prático das equipes de desenvolvimento de software, sejam elas colocalizadas ou distribuídas, utilizou-se a metodologia DSR (*Design Science Research*). A metodologia apoia a produção de artefatos inovadores para problemas práticos. A solução para o problema aqui abordado foi concretizada em um *framework* projetado, desenvolvido e avaliado para apoiar a determinação de diferentes tipos de conhecimentos de projetos de software, a partir dos registros de comunicação das equipes de desenvolvimento, o *framework Miner4DevTeam* (Figura 7.1). Foram utilizados conceitos, algoritmos e técnicas de mineração de dados, oriundos das áreas de recuperação de informação, processamento de linguagem natural e *deep learning* visando a determinação do *Project Lost Knowledge* (PLK), *Project Frequent Knowledge* (PFK) e *Project Related Knowledge* (PRK) (Capítulo 3). O *framework* é formado por três componentes: o *Lost Knowledge Component*, que apoia a determinação do PLK, o *Frequent Knowledge Component*, que apoia a identificação do PFK, e o *Related Knowledge Component*, que apoia a identificação do PRK.

As conclusões desta pesquisa são apresentadas a seguir, conforme os tipos de conhecimento investigado. Com respeito ao *Project Lost Knowledge* (PLK), tem-se que:

- Os membros das equipes de desenvolvimento usam ferramentas de comunicação síncronas, especificamente *chats*, para discutir diferentes tópicos acerca dos projetos realizados, tais como requisitos, planejamentos, desenvolvimento, testes, etc. Contudo, as equipes também discutem tópicos não relevantes para a determinação de conhecimentos de projetos de software, como discussões relacionadas à integração social das equipes e discussões de coordenação de reuniões de trabalho.

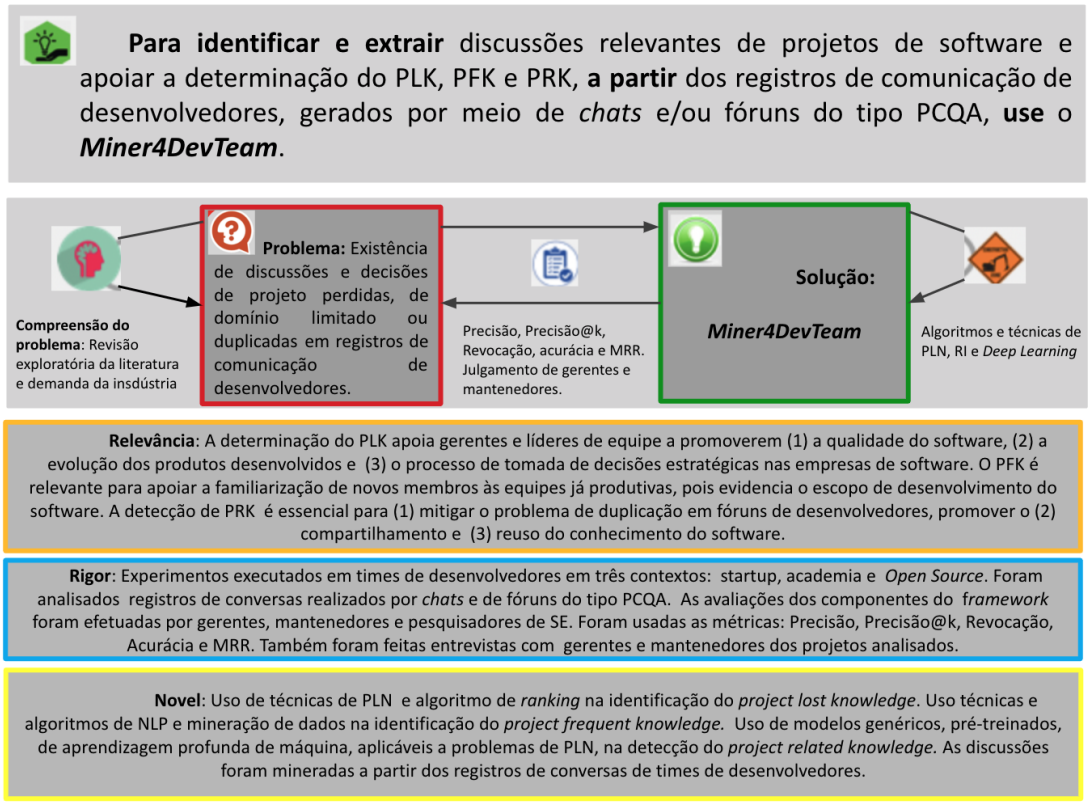

Figura 7.1: *Visual Abstract* desta pesquisa de doutorado.

- O *LK-Component* demonstrou resultados promissores ao determinar automaticamente decisões de projeto a partir dos registros de comunicação de equipes de desenvolvimento e oportunizar a identificação do PLK.
- É possível utilizar o algoritmo de *Text Summarization* para extrair decisões relacionadas aos projetos de software, a partir dos registros de comunicação dos times de desenvolvimento que usam *chats*.
- Gerentes de projetos identificaram *Project Lost Knowledge* (PLK) a partir de decisões de projetos relevantes extraídas.
- Gerentes de projetos destacaram ter interesse em adotar o processo de identificação do PLK em outros projetos sob suas respectivas responsabilidades.
- O PLK pode ser usado para promover a transferência de conhecimentos de

software, disseminar informações entre membros das equipes e promover a oportunidade de criação de novas ideias que possam ser implementadas em novas versões dos produtos desenvolvidos.

- Os PLKs identificados continham diferentes tipos de discussões de projetos de software, destacando que o conhecimento pode ser perdido em diferentes estágios do ciclo de vida de desenvolvimento do software e o PLK não está, necessariamente, vinculado à nenhuma etapa específica.
- A utilização do *LK-Component* não se limita ao contexto da indústria de software. Professores que atuam no ensino da ES também podem ser beneficiados ao utilizarem o componente para extrair discussões relevantes dos *logs* de comunicação de equipes de estudantes, para apoiar as atividades pedagógicas dos professores e monitorar o desenvolvimento dos projetos de software realizados pelos estudantes.

Com respeito ao *Project Frequent Knowledge* (PFK), tem-se que:

- O PFK pode ser determinado através dos assuntos frequentemente discutidos pelas equipes de desenvolvimento. O PFK evidencia o escopo de desenvolvimento dos projetos de software.
- O *FK-Component* demonstrou resultados promissores ao externar decisões de projeto frequentemente discutidas nos registros de comunicação de equipes de desenvolvimento e oportunizar a determinação do PFK.
- As barreiras enfrentadas por novos membros, ao chegarem em uma nova equipe de desenvolvimento, podem ser amenizadas com a determinação do PFK.
- O processo de identificação do PFK fornece oportunidade para transferência de conhecimentos técnicos, sociais e organizacionais.
- A utilização do PFK não se limita aos novatos, membros antigos podem usá-lo para recordar decisões já tomadas pelas equipes.

Com respeito ao *Project Related Knowledge* (PRK), tem-se que:

- Usuários do fórum GitHub *Discussions* duplicam postagens quando (1) desejam reforçar um pedido de ajuda, (2) desejam acrescentar novas informações às postagens anteriormente criadas ou (3) quando buscam alternativas diferentes para a solução do problema.

- Postagens relacionadas ocorridas no *Discussions* podem ser reorganizadas, unidas ou remanejadas como comentários umas das outras.

- O dinamismo com que as comunidades no GitHub crescem justifica o uso do modelo de aprendizado de máquina genéricos.

- O *RK-Component* usa um modelo, de aprendizagem profunda genérico, de acesso público, pré-treinado pelo Sentence-BERT (SBERT) para calcular *embeddings* de sentenças semanticamente significativas de postagens. Esta decisão de planejamento trouxe flexibilidade à abordagem. À medida que os pesquisadores lançam novos modelos melhores, pode-se atualizar o *RK-Component*.

- Os mantenedores de projetos hospedados na plataforma GitHub podem se beneficiar do *RK-Component* para lidar com a tarefa trabalhosa de detectar manualmente postagens relacionadas.

- Os resultados gerados pelo *RK-Component* podem ser usados por mantenedores para priorizar o desenvolvimento ou atualizar postagens específicas do projeto frequentemente discutidas, controlar a propagação de postagens relacionadas e apoiar o compartilhamento e reuso de conhecimento do projeto.

Contudo, os estudos realizados evidenciaram necessidades de aprimoramento dos mecanismos propostos. A manipulação de textos provenientes dos registros de comunicação das equipes revelou alguns desafios que foram considerados, porém, necessitam de maior atenção na criação ou evolução dos mecanismos de mineração aqui propostos, são elas:

- Apesar de serem feitas por especialistas (membros das equipes de desenvolvimento ou desenvolvedores), as discussões registradas nos *logs* normalmente

seguem um padrão informal de conversação. Isso significa que as normas, critérios e jargões técnicos usados na escrita da documentação oficial do sistema podem ser desprezados pelos membros.

- Mensagens de bate-papo costumam ser escritas sem rigor ortográfico e podem não seguir um fluxo sequencial de emissão. Apesar de ser um desafio para as estratégias automáticas de manipulação de tais dados, os membros das equipes são perfeitamente capazes de compreender e se comunicar nesses contextos. Além disso, é comum a presença de gírias, jargões, erros ortográficos, e-mojis, onomatopeias nos textos das mensagens, assim como o uso de abreviações para agilizar a escrita. Esses tipos de mensagens podem não ser corretamente interpretadas pelos algoritmos implementados.

- Desenvolvedores podem discutir diferentes assuntos nos meios de comunicação utilizados, gerando registros de discussões relacionadas aos projetos e registros de discussões fora do escopo dos projetos.

- Modelos genéricos de *deep learning* podem falhar na manipulação de contextos específicos de projetos de software, pois podem não identificar as especificidades dos assuntos discutidos.

- Os estudos também evidenciaram que o perfil e as experiências profissionais dos membros que usaram os mecanismos propostos podem interferir no processo de determinação ou identificação dos diferentes tipos de conhecimentos aqui explorados.

- Não é garantida a identificação exaustiva de todos os conhecimentos aqui elencados, sendo recomendada a leitura e compreensão das documentações oficiais dos projetos.

7.1.1 Resposta da Questão de Pesquisa

Nesta seção, será respondida a questão de pesquisa desta tese. Sendo assim, tem-se:

QP: Como utilizar métodos automáticos para identificar discussões relevantes de projetos de software a partir dos registros de comunicação de desenvolvedores?

Os estudos realizados evidenciaram resultados positivos na utilização de técnicas e algoritmos das áreas de mineração de dados, mais especificamente Recuperação de Informação, Processamento de Linguagem Natural e *Deep Learning*, para a identificação e extração de discussões de projetos relevantes a partir dos registros de comunicação de desenvolvedores. No contexto desta pesquisa, discussões relevantes são aquelas que viabilizam a disseminação e o reuso de conhecimentos de projetos de software entre integrantes de equipes distribuídas ou colocalizadas. Ou seja, discussões que revelam informações acerca da análise, do planejamento, desenvolvimento, utilização, validação, lições aprendidas e gerenciamento do software. Nesta pesquisa foram utilizadas abordagens baseadas em (1) ranque, o *LK-Component*, (2) processamento de linguagem natural otimizados, *FK-Component* e (3) modelos de PLN pré-treinadas, *RK-Component*.

A identificação de discussões relevantes a partir dos registros de comunicação de equipes de desenvolvimento de software promove a captura de conhecimento, dá suporte à tomada de decisão, pode aumentar a eficiência da equipe e contribui para a qualidade do desenvolvimento de projetos de software. Especificamente, esta tese foi motivada pela oportunidade de:

- Promover o compartilhamento e reuso de conhecimento: A identificação de discussões relevantes que antes estavam "perdidas" do ponto de vista de membros das equipes de software facilita o compartilhamento e reuso do conhecimento adquirido durante o desenvolvimento. Isso pode ajudar, dentre outras oportunidades, (1) gerentes a resgatarem o histórico de discussões relevantes sobre a tomada de decisão dos projetos e (2) novos membros da equipe a se atualizarem e a entenderem os contextos e decisões anteriores.

- Evitar repetição de discussões: A identificação de discussões relacionadas pode evitar a repetição de tópicos já abordados. Isso economiza tempo e recursos, permitindo que a equipe se concentre em discussões novas ou mais urgentes e ajuda mantenedores de equipes que usam o GitHub *Discussions* a lidarem com os problemas de degradação do fórum causado pela ocorrência

de discussões duplicadas.

Contudo, a eficácia desses métodos depende da qualidade dos dados, da quantidade de dados disponíveis e da escolha apropriada das técnicas de processamento. Além disso, a manipulação de textos provenientes de *chats* de desenvolvedores e fóruns do tipo PCQA constituem desafios próprios do problema.

Os estudos realizados evidenciaram que as discussões relevantes de projetos de software identificadas impactaram positivamente na identificação do *Project Lost Knowledge* (PLK), na determinação do *Project Frequent Knowledge* (PFK) e na detecção do *Project Related Knowledge* (PRK). Tais conhecimentos evidenciam informações acerca de projetos dos software desenvolvidos, como informações referentes à definição, planejamento, configurações, testes, aceite, tecnologias utilizadas, gerenciamento, lições aprendidas e entrega do projeto que promovem o compartilhamento e o reuso do conhecimento do projeto.

Argumenta-se que o compartilhamento e o reuso do conhecimento de projetos de software é importante para garantir pontos como:

- Continuidade: O compartilhamento de conhecimento garante que, mesmo que membros da equipe saiam ou novos entrem, o conhecimento presente nos registros de comunicação das equipes não seja perdido. Isso assegura a continuidade do projeto e reduz a dependência de indivíduos específicos.

- Qualidade: O compartilhamento de lições aprendidas pode melhorar a qualidade do trabalho da equipe. A equipe pode aprender com erros passados e adotar práticas bem-sucedidas.

- Colaboração: O conhecimento compartilhado facilita que os membros das equipes entendam melhor o contexto, as decisões tomadas e o raciocínio envolvido, promovendo um ambiente de trabalho colaborativo.

- Redução de Riscos: O compartilhamento de conhecimento ajuda a evitar a repetição de erros técnicos e de gerenciamento já cometidos.

- Novas ideias: Ao expor algumas discussões e decisões já feitas, a equipe pode encontrar novas formas de abordar problemas e explorar soluções mais eficazes.

- Aprendizado: Compartilhar conhecimento proporciona oportunidades para que outros membros da equipe aprendam com as experiências, erros e sucessos anteriores.

- Documentação: O conhecimento compartilhado pode ser documentado, contribuindo diretamente os demais pontos citados.

Por outro lado, a não identificação de conhecimentos de projetos de software ou a perda de tais conhecimentos, podem levar a uma variedade de problemas, como a perda do raciocínio referente às decisões de projeto, a diminuição da compreensão dos sistemas desenvolvidos, a degradação do compartilhamento de conhecimentos e dificulta a avaliação do software e o impacto das estimativas de mudanças.

Sendo assim, o *framework Miner4DevTeam* surge como uma abordagem para apoiar equipes de desenvolvimento na gestão do conhecimento dos projetos de software dentro das organização. Podendo o *LK-Component* ser utilizado durante todo o processo de desenvolvimento com o objetivo de auxiliar a identificação de conhecimentos esquecidos e que, possivelmente, impactaram em mudanças entre o produto inicialmente definido e o produto final gerado. Para isso, o *LK-Component* deve receber como entrada os registros de conversas da equipe de desenvolvimento e as datas inicial e final que delimitam o período de tempo sobre o qual o componente fará a análise das mensagens registradas. Como saída, o componente fornece uma lista ranqueada com as N sentenças (sequência de mensagens enviadas pelo mesmo usuário) mais relevantes identificadas no período analisado. A partir de tais sentenças, o gerente de projeto da equipe pode determinar o *Project Lost Knowledge*. Já o *FK-Component*, que também pode ser utilizado durante todo o processo de desenvolvimento do software, tem o objetivo de assistir a entrada de novos membros a equipes já produtivas, pois facilita a familiarização destes ao novo contexto de trabalho, tornando-os produtivo mais rapidamente. Para isso, com o objetivo de prover informações que evidenciem a evolução temporal do projeto analisado, o gerente de projeto da equipe deve fornecer o registro de comunicação da equipe e definir datas limítrofes que delimitam o passado remoto e o passado recente do projeto de software. Em seguida, o componente gera duas *word clouds* que externam o escopo do projeto de software analisado: a **Semân-**

tica, composta por termos que tentam evidenciar características de especificação, planejamento, desenvolvimento, manutenção e gerenciamento do projeto de software e a **Técnica**, composta por termos que tentam evidenciar características relacionadas ao uso de tecnologias e decisões técnicas no desenvolvimento do projeto. Os termos presentes nas *word clouds* podem ser utilizadas como índices que permitem a navegação do usuário pelo *log* de conversas da equipe. A partir de tais *word clouds*, os novos membros podem consumir o *Project Frequent Knowledge*. Por fim, o *RK-Component* apoia a diminuição de possíveis inconsistências geradas pela duplicidade de informações acerca dos projetos. Desta forma, sua utilização também pode ser feita durante todas as etapas do processo de desenvolvimento do software. Como entrada, o componente recebe uma lista contendo postagens de discussões de desenvolvedores feitas em fóruns de comunicação. O componente analisa o conteúdo das postagens e fornece um conjunto R com pares de discussões candidatas a discussões relacionadas.

7.2 Contribuições

Esta seção apresenta as contribuições que esta pesquisa de doutorado trouxe para a área de Engenharia de Software. Ao longo desta pesquisa, foram identificadas e desenvolvidas diversas contribuições que ampliam o entendimento sobre a ocorrência de conhecimentos de projetos de software nos registros de comunicação de desenvolvedores. Essas contribuições abrangem desde evidências experimentais sobre o que os desenvolvedores discutem até mecanismos robustos baseados em *deep learning* para detectar o *project knowledge* em fóruns de desenvolvedores. A seguir, são categorizadas e descritas as contribuições desta tese:

- **Contribuições teóricas:**

    1. Evidências experimentais sobre a perda de conhecimento dos projetos de software em mensagens de bate-papo (*chats*) das equipes de desenvolvimento.
    2. Evidências experimentais sobre as aplicações práticas do *LK-Component* em empresas que usam ferramentas de bate-papo na comunicação da equipe.

3. Evidências experimentais sobre as aplicações práticas do *FK-Component* em empresas que usam *chats* como ferramentas de comunicação.
4. Evidências experimentais sobre o uso de um modelo de aprendizado de máquina de propósito genérico para detectar postagens de discussão relacionadas criadas por desenvolvedores.
5. Evidências experimentais sobre as aplicações práticas do *RK-Component* para as comunidades hospedadas na plataforma GitHub, sob a perspectiva de três mantenedores OSS.
6. Evidências experimentais sobre as atividades de compartilhamento de links no fórum de discussões do GitHub.
7. Um estudo exploratório para identificar os tipos de recursos comumente referenciados no *Discussions* e as razões para o compartilhamento de tais referências.
8. Evidências experimentais sobre como estudantes de graduação em computação usam Slack para apoiar o desenvolvimento de projetos de software.
9. Evidências experimentais sobre como o registro de comunicação dos estudantes podem dar suporte as atividades pedagógicas de professores da área de computação.

- **Contribuições práticas:**

1. O desenvolvimento e avaliação de uma abordagem semi-automática, baseada em técnicas de PLN e algoritmos de ranque para apoiar a identificação de decisões de projetos que não foram formalmente registradas na documentação do software e podem estar "perdidas" sob a perspectiva dos gerentes de projeto.
2. O desenvolvimento e avaliação de uma abordagem semi-automática, baseada em PLN e técnicas de mineração de dados otimizadas para apoiar a identificação de decisões de projetos que já se tornaram claras e intrínsecas ao time de desenvolvimento, porém não foram formalmente registradas na documentação do software. Tais informações podem

ser usadas para apoiar a entrada de novos membros às equipes de desenvolvimento.

3. O desenvolvimento e avaliação de uma abordagem baseada em modelos de aprendizado de máquina profunda para detectar postagens de discussão relacionadas em fóruns do tipo PCQA.

- **Publicações aceitas:**

1. Lima, M., Valle, V., Costa, E., Lira, F., and Gadelha, B. (2019c). Software engineering repositories: Expanding the PROMISE database. In Proceedings of the XXXIII Brazilian Symposium on Software Engineering, SBES 2019, Salvador, Brazil, September 23-27, 2019, pages 427–43.
2. Lima, M., Ahmed, I., Conte, T., Nascimento, E., Oliveira, E., and Gadelha, B. (2019a). Land of lost knowledge: An initial investigation into projects lost knowledge. In 2019 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM), pages 1–6. IEEE.
3. Lima, M., Fontão, A., Fernandes, D., Conte, T., and Gadelha, B. (2019b). How are my students going? a tool to analyse students' interactions on capstone courses. In Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE), volume 30, page 1611.
4. Lima, M., Oliveira, E., Conte, T., and Gadelha, B. (2020). Clouds are heavy! a storm of relevant project-related terms to support newcomers' onboarding. In Proceedings of the 34th Brazilian Symposium on Software Engineering, SBES 2020, Natal, Brazil, October 19-23, 2020, pages 319–324.
5. Borges, O., Lima, M., Couto, J., Gadelha, B., Conte, T., and Prikladnicki, R. (2022). Ml@se: What do we know about how machine learning impact software engineering practice? In 2022 17th Iberian Conference on Information Systems and Technologies (CISTI), pages 1–7. IEEE.

6. Lima, M., Steinmacher, I., Ford, D., Liu, E., Vorreuter, G., Conte, T., and Gadelha, B. (2023). Looking for Related Discussions Posts on GitHub Discussions. PeerJ Computer Science. Aceito para publicação.
7. Pessoa, M., Lima, M., Pires, F., Haydar, G., Melo, R., Rodrigues, L., Oliveira, D., Oliveira, E., Galvão, L., Gadelha, B., Isotani, S., Gasparini, I., Conte, T. (2023). A Journey to Identify Users' Classification Strategies to Customize Games and Gamified Environments. IEEE Transactions on Learning Technologies. DOI: 10.1109/TLT.2023.3317396. IEEE.

- **Publicações submetidas:**

1. Lima, M., Steinmacher, I., Ford, D., Vorreuter, G., Gonçalves, L., Conte, T., and Gadelha, B. (2023). How are discussions linked? A link analysis study on the GitHub Discussions forum. Submetido para revisão no Journal of Systems and Software.

- **Impactos na indústria de software:** Os estudos aqui realizados impactaram na comunidade *open source*. Como evidência, o GitHub destacou a importância da pesquisa realizada em seu blog oficial[1]. Além disso, os resultados gerados motivaram a implementação de novas *features* no GitHub *Discussions*, como:

1. *Checkbox for duplicates* - *Checkbox* que destaca a realização da busca por postagens duplicadas antes que uma nova postagem seja criada pelos usuários do *Discussions*.
2. *Most Helpful* - quadro que destaca os usuários que mais contribuíram para responder perguntas realizadas no fórum.

7.3 Trabalhos Futuros

Embora esta tese tenha abordado a identificação de conhecimentos de projetos de software a partir dos registros de comunicação de desenvolvedores, há diversas

[1] https://github.blog/2022-07-20-heres-how-academic-research-is-shaping-github-discussions/

oportunidades para futuras investigações e desenvolvimentos que podem expandir e aprofundar ainda mais o entendimento do tema. Nesta seção, apresentamos algumas sugestões para trabalhos futuros, cada um deles com potencial para apoiar as tarefas realizadas por diferentes membros de equipes de desenvolvimento.

Novas pesquisas podem focar no aprimoramento dos componentes aqui propostos. Dado que todos os componentes abordam problemas que envolvem PLN, desafios oriundos da própria área oportunizam o desenvolvimento de novas pesquisas. A ambiguidade linguística (palavras e frases com múltiplos significados), desafios multilíngues (vários idiomas), a falta de dados rotulados e os recursos semânticos limitados (nuances e a semântica profunda da linguagem humana) constituem desafios a serem tratados. Além disso, o contexto de aplicação das soluções propostas, registros de comunicação de times de desenvolvimento de software, agregam desafios ainda maiores, pois nesse contexto geralmente há a descrição de trechos de códigos cuja estrutura e vocabulário diferem das linguagens naturais (Sirres et al., 2018). Sendo assim, o entendimento contextual no qual as palavras ou frases são usadas é desafiador. Superar esses desafios requer pesquisa contínua em técnicas de PLN.

Outras pesquisas podem focar em apoiar a determinação do PLK, PFK e PRK a partir de diferentes repositórios de dados ou a partir da combinação de dados neles armazenados. Diferentes repositórios podem ser gerados a partir da colaboração de equipes de desenvolvimento de software. Neles estão armazenadas informações sobre o processo de desenvolvimento e as interações da equipe. Como visto na Seção 2.2, tais repositórios são valiosas fontes de informações sobre o projeto. Sendo assim, criam-se oportunidades para a mineração de PLK, PFK e PRK em repositórios de controle de versão, plataformas de gerenciamento de projetos, sistemas de rastreamento de *bug*, outros fóruns e outras ferramentas de discussão, *e-mails*, redes sociais corporativas e documentação.

Vislumbram-se também oportunidades de novas pesquisa para apoiar a área de *knowledge managemente* em projetos de software. Com este foco, é importante implementar estratégias que não somente facilitem a captura do conhecimento, mas também apoiem a organização, o compartilhamento e uso deste na equipe. A partir da identificação do PLK, PFK e PRK novos pesquisas podem focar em documentar e catalogar automaticamente tais conhecimentos, criando bases de

conhecimento dos projetos de software, onde as informações relevantes podem ser armazenadas e facilmente acessadas por todos os membros da equipe. Pode-se criar estratégias eficientes que incentivem a transformação de conhecimentos de projetos tácitos (propriedade intelectual) em explícitos (escritos). Por fim, apoiar a transferência de conhecimento entre membros experientes e novos membros da equipe. É fundamental dar suporte à gestão do conhecimento em projetos de software, visando maximizar a eficiência das equipes, promover a inovação e garantir que o conhecimento valioso seja retido e utilizado (Levy and Hazzan, 2009; Natali and de Almeida Falbo, 2002).

Por fim, acredita-se que novas pesquisas possam explorar o "conhecimento social" presente nos registros de conversas de times de desenvolvimento. O conhecimento social do projeto compreenderia o conhecimento evidenciado através do padrão comportamental dos membros das equipes, como por exemplo, o papel desempenhado por um membro específico frente às discussões de projeto feitas pelas equipes. Visualizando as equipes de desenvolvimento como uma comunidade é possível mapear os registros de conversas destas equipes em uma rede de conexão, gerando grafos que evidenciam os relacionamentos existentes entre os membros dos times. Tais grafos podem ser usados para modelar e compreender as conexões estabelecidas pelos membros das equipes e, ainda, derivar métricas da área de Análise de Redes Sociais (ARS), como a coesão, a centralidade e a densidade das redes geradas(Bosu and Carver, 2014), que evidenciem novos conhecimentos sobre os projetos de software.

Apêndice A

Lista de termos técnicos usada pelo *Frequent Knowledge Component*

Este apêndice apresenta a lista de termos T usada pelo *Frequent Knowledge Component* para apoiar a determinação dos assuntos técnicos discutidos pelas equipes de desenvolvimento.

A.1 Lista T

A lista T foi gerada a partir do conjunto de *tags* usadas pelo *Stack Overflow* para categorizar as perguntas semelhantes, feitas por profissionais de computação, que ocorrem no *site*. A lista T é uma adaptação da lista de *tags* do *Stack Overflow*. Tal adaptação foi realizada com objetivo de viabilizar a determinação dos assuntos técnicos frequentemente discutidos pelas equipes de desenvolvimento e registrados nos *logs* de conversas destas, conforme explicado na Seção 3.3.

```
1  php
2  javascript
3  java
4  c#
5  html
6  jquery
7  mysql
8  android
9  css
10 python
11 sql
12 c
13 html5
14 asp.net-mvc
15 ajax
16 banco-de-dados
17 json
18 laravel
19 python-3.x
20 array
21 sql
22 server
23 .net
24 angularjs
25 c++
26 node.js
27 r
```

28 asp.net
29 css3
30 angular
31 string
32 wordpress
33 twitter
34 bootstrap
35 orientacao-a-objetos
36 delphi
37 postgresql
38 entity-framework
39 winforms
40 android
41 studio
42 regex
43 query
44 funcoes
45 api
46 oracle
47 codeigniter
48 excel
49 xml
50 jsf
51 linux
52 asp.net-mvc-5
53 typescript
54 ionic
55 webservice
56 hibernate
57 pdo
58 mysqli
59 aplicacao
60 web
61 django
62 git
63 react
64 lista
65 visual-studio
66 ruby-on-rails
67 ios
68 laravel-5
69 primefaces
70 swing
71 select
72 firebase
73 spring
74 android
75 mvc
76 vba
77 vb.net
78 formulario
79 imagem
80 windows
81 ruby
82 vue.js
83 htaccess
84 arquivo
85 apache
86 classes
87 java-ee
88 sqlite
89 algoritmo
90 jpa
91 asp.net-core
92 variaveis
93 linq
94 loop
95 apache-cordova
96 react-native
97 spring-boot
98 google-maps
99 bootstrap-3
100 rest
101 mobile
102 mongodb
103 terminologia
104 xamarin
105 swift
106 eclipse
107 bootstrap
108 python-2.7
109 http
110 front-end
111 desempenho
112 asp.net-web-api
113 ponteiro
114 data
115 matriz
116 wpf
117 listview
118 input
119 session
120 laravel-eloquent
121 datetime
122 post
123 ubuntu
124 thread
125 jsp
126 seguranca
127 cakephp
128 servidor
129 email
130 if
131 entity-framework-6
132 webforms
133 android-activity
134 razor
135 pandas
136 pdf
137 date
138 struct
139 script
140 github
141 firebird
142 socket
143 matematica
144 url
145 android-fragment
146 logica
147 for
148 validacao
149 curl
150 excecao
151 upload
152 facebook
153 javafx
154 asp
155 modal
156 express
157 linguagem
158 phpmyadmin
159 modelagem
160 tipagem
161 estrutura-de-dados
162 maven
163 login
164 expressoes
165 lambda
166 bootstrap-4
167 foreach
168 spring-mvc
169 checkbox
170 tomcat
171 docker
172 objective-c
173 ionic3
174 criptografia
175 csv
176 jquery-datatables
177 div
178 classificacao
179 unity3d
180 while
181 operadores
182 selenium
183 join
184 excel-vba
185 google
186 bash
187 table
188 netbeans
189 memoria
190 jsf-2.2
191 design-pattern
192 google-chrome
193 cmd
194 tkinter
195 phpmailer
196 menu
197 engenharia-de-software
198 metodo
199 iis
200 shell-script
201 tabela-banco-de-dados
202 sql-update
203 sintaxe
204 xamarin-forms
205 aws
206 arraylist
207 pl-sql
208 dom
209 javascript-eventos
210 xcode
211 character-encoding
212 busca
213 ionic2
214 layout-responsivo
215 estilo-de-codificacao
216 datatable
217 heranca
218 sql-select
219 objetos
220 soap
221 jquery-ui
222 arduino
223 recursao
224 visual-studio-2015
225 batch
226 visual-basic-6
227 arquitetura-de-software
228 jdbc

229	envio-de-email	296	azure	363	modelo-relacional
230	testes	297	stored	364	ado.net
231	conversao-de-tipo	298	procedures	365	fullcalendar
232	datagridview	299	materialize	366	gerenciamento-de-memoria
233	conexao	300	html	367	hash
234	iframe	301	select	368	matlab
235	trigger	302	formatacao	369	ireport
236	shell	303	versionamento	370	autocompletar
237	restful	304	laravel-blade	371	video
238	navegador	305	mysql	372	processamento-de-imagens
239	random	306	workbench	373	firefox
240	autenticacao	307	relacionamento	374	switch
241	visual-studio-code	308	asp.net-identity	375	deploy
242	nullpointerexception	309	debug	376	numpy
243	parametros	310	rstudio	377	terminal
244	canvas	311	laravel	378	sqlite3
245	tabela	312	nginx	379	junit
246	multithreading	313	assembly	380	ddd
247	java-8	314	dart	381	internet-explorer
248	visual-studio-2013	315	spring-security	382	angular-6
249	flutter	316	angularjs-directives	383	plugin-wordpress
250	lua	317	dll	384	float
251	sql-insert	318	datepicker	385	redes
252	framework	319	asp.net-mvc-4	386	assincronismo
253	servlet	320	ssl	387	construtor
254	http-request	321	web	388	nfe
255	qt	322	scrap	389	android-recyclerview
256	get	323	return	390	gridview
257	npm	324	dropdown	391	windows-phone
258	selenium-webdriver	325	relatorio	392	grid
259	file-upload	326	laravel	393	log
260	controller	327	xaml	394	chartjs
261	composer	328	google	395	nhibernate
262	opencv	329	console	396	localstorage
263	delphi-7	330	async	397	xamarin.android
264	app	331	flask	398	url-amigavel
265	view	332	alocacao	399	hospedagem
266	lista-encadeada	333	ecmascript-6	400	cors
267	interface	334	dicionario	401	grafico
268	button	335	redirecionamento	402	jboss
269	rolamento	336	jtable	403	jpa-2.0
270	php	337	request	404	wcf
271	access	338	ftp	405	char
272	wordpress	339	utf-8	406	null
273	cookies	340	certificado	407	procedure
274	magento	341	download	408	heroku
275	paginacao	342	cache	409	biblioteca
276	webview	343	otimizacao	410	geolocalizacao
277	kotlin	344	condicao	411	int
278	ggplot2	345	entity-framework-core	412	https
279	xampp	346	.net-core	413	template
280	eventos	347	axios	414	enums
281	txt	348	visual	415	gradle
282	jasper-reports	349	studio	416	declaracao-de-variavel
283	animacao	350	pagseguro	417	namespace
284	mongoose	351	maps	418	gui
285	svg	352	filtro	419	diretorio
286	testes-unitarios	353	android	420	tags
287	combobox	354	adapter	421	time
288	woocommerce	355	sass	422	instalacao
289	crud	356	layout	423	ide
290	link	357	websocket	424	impressao
291	data.frame	358	array	425	xhtml
292	seo	359	multidimensional	426	indices
293	plugin	360	comparacao	427	google-maps-api-3
294	compilacao	361	reflexao	428	oracle11g
295	windows	362	genericos	429	group-by

430	jogos	497	aplicacao-desktop	564	base64
431	wildfly	498	sistema-operacional	565	socket.io
432	zend	499	reportviewer	566	jquery-select2
433	highcharts	500	media-queries	567	genexus
434	geracao-de-pdf	501	dplyr	568	gson
435	spring-data	502	asynctask	569	cordova-plugin
436	facebook-graph-api	503	latex	570	electron
437	mariadb	504	processo	571	opengl
438	thymeleaf	505	slim	572	unity-5
439	matplotlib	506	delphi-10	573	collection
440	cakephp-2	507	mpdf	574	site
441	stack	508	grafo	575	google-chrome-extensao
442	migrations	509	services	576	gps
443	django-templates	510	else	577	dompdf
444	doctrine	511	jquery-mobile	578	arvore-binaria
445	conversao	512	independente-de-linguagem	579	mod-rewrite
446	visualg	513	timestamp	580	posicionamento
447	back-end	514	spring-jpa	581	textview
448	push-notification	515	replace	582	machine-learning
449	docker-compose	516	firebase-database	583	propriedade
450	dataframe	517	crystal-reports	584	symfony
451	tsql	518	uml	585	inteligencia-artificial
452	filter	519	token	586	fluent-api
453	firemonkey	520	flexbox	587	progress-bar
454	header	521	calculo	588	try-catch
455	iphone	522	complexidade-de-algoritmo	589	slider
456	importacao	523	smtp	590	jar
457	oauth2	524	sistema-de-arquivos	591	sequelize-js
458	ip	525	rotas	592	count
459	jwt	526	cron	593	plot
460	route	527	xmlhttprequest	594	html5-video
461	concatenacao-de-string	528	backup	595	path
462	acentuacao	529	responsivo	596	google-analytics
463	powershell	530	where	597	callback
464	textbox	531	webpack	598	stream
465	radiobutton	532	angular-routes	599	gcc
466	programacao-funcional	533	localhost	600	delete
467	orm	534	cakephp-3	601	wsdl
468	osx	535	configuracao	602	uwp
469	ssh	536	impressora	603	angular-material
470	carousel	537	escopo-de-variaveis	604	build
471	retrofit	538	chave-estrangeira	605	imageview
472	golang	539	apk	606	google-maps-android-api-2
473	cdi	540	jasper	607	paralelismo
474	youtube	541	web	608	hover
475	c++11	542	crawler	609	auth
476	syntax-error	543	ejb	610	arquivo-zip
477	teclado	544	team-foundation-server	611	transacoes
478	gulp	545	wamp	612	dominio
479	sql-server-2012	546	popup	613	pascal
480	edittext	547	jframe	614	haskell
481	ux	548	macro	615	web.config
482	jquery-validate	549	http	616	pygame
483	primaverabss	550	google	617	mask
484	dns	551	play	618	webbrowser
485	glassfish	552	serializacao	619	polimorfismo
486	nosql	553	permissoes	620	bitbucket
487	onclick	554	django-2.0	621	doctrine-2
488	include	555	rails4	622	angularjs-scope
489	audio	556	vector	623	label
490	portugol	557	css-hover	624	openssl
491	injecao-de-dependencia	558	cliente-servidor	625	mvvm
492	centos	559	cast	626	sql-server-2014
493	promises	560	dialog	627	order-by
494	symfony-2	561	background	628	referencia
495	code-first	562	demoiselle	629	atributos
496	vue.js-2	563	hashmap	630	python-requests

631 ciencia-da-computacao
632 timer
633 ckeditor
634 sublime-text
635 windows-phone-8-1
636 sql-server-2008
637 linha-de-comando
638 raspberry-pi
639 scanf
640 proxy
641 calendar
642 revisao-codigo
643 dapper
644 nav-bar
645 runtime
646 vraptor
647 projeto-de-software
648 debian
649 windows-10
650 image
651 intel-xdk
652 compiladores
653 instanciar-objeto
654 drag-n-drop
655 actionscript-3
656 scrapy
657 bluetooth
658 laravel-5.4
659 android-notification
660 active-record
661 click
662 parser
663 documentacao
664 pyqt
665 repository
666 volley
667 getters-setters
668 blob
669 streaming
670 commit
671 criteria
672 textarea
673 integracao
674 contador
675 linq-to-entities
676 material-design
677 codeigniter-3
678 fluent-nhibernate
679 yii
680 persistencia
681 intellij-idea
682 redux
683 python-kivy
684 licenca
685 sdk
686 bitmap
687 semantica
688 joomla
689 active-directory
690 devexpress
691 automatizacao
692 jenkins
693 codeblocks
694 android-xml
695 delphi-xe8
696 emulador
697 svn
698 gitlab
699 id
700 booleano
701 inner
702 url-rewrite
703 preg-match
704 redes-neurais
705 requisicao
706 single-page-application
707 md5
708 pip
709 tdd
710 firedac
711 cloud
712 sharepoint
713 twitter
714 jspdf
715 sharedpreferences
716 navigation-drawer
717 jpql
718 youtube-api
719 recaptcha
720 moodle
721 cursor
722 css-font-face
723 serial
724 entidade-relacionamento
725 realtime-database
726 chat
727 alert
728 pycharm
729 concorrencia
730 android-camera
731 assinatura-digital
732 regressao
733 html5-audio
734 jersey
735 dev-c++
736 parse
737 groovy
738 xpath
739 decimal
740 react-navigation
741 adonisjs
742 atualizacao
743 componentes
744 annotation
745 flash
746 blogger
747 prompt
748 printf
749 apple
750 scss
751 microsservicos
752 fila
753 gmail
754 owl-carousel
755 pwa
756 scanner
757 mac
758 merge
759 paradigmas
760 garbage-collector
761 senhas
762 packages
763 eventos-do-mouse
764 value
765 whatsapp
766 less
767 facebook-sdk
768 task
769 delphi-xe3
770 unix
771 http-status
772 ffmpeg
773 oauth
774 cross-platform
775 vagrant
776 odbc
777 binary
778 static
779 beautifulsoup
780 android-intent
781 lado-do-servidor
782 ascii
783 model
784 swift-3
785 postman
786 slide
787 automapper
788 object
789 font
790 cpanel
791 xls
792 itextsharp
793 hexadecimal
794 online
795 classes
796 abstratas
797 laravel-rotas
798 grunt
799 case
800 delphi
801 admob
802 google
803 drive
804 api
805 firestore
806 intent
807 kendo-ui
808 connectionstring
809 chave-primaria
810 visual-studio-2012
811 actionbar
812 microsoft
813 zend-framework-2
814 spinners
815 caracteres
816 css-animations
817 testes-automatizados
818 encapsulamento
819 alinhamento
820 constantes
821 numeros-aleatorios
822 css-transicoes
823 django-rest-framework
824 sms
825 nomenclatura
826 treeview
827 e-commerce
828 meta-tags
829 iterator
830 binding
831 jaspersoft-studio

832 text
833 redirecionamento-de-url
834 kivy
835 angular-7
836 dinamico
837 padrao-dao
838 array-associativo
839 img
840 pyqt-5
841 word
842 linq-to-sql
843 react-router
844 import
845 vbs
846 virtualhost
847 chatbot
848 inner-join
849 map
850 desserializacao
851 qrcode
852 perl
853 sqldeveloper
854 paypal
855 advpl
856 nuget
857 ionic4
858 echo
859 inserir
860 editor-de-texto
861 opencart
862 knockoutjs
863 jvm
864 aprendizagem-de-maquina
865 lazarus
866 galeria
867 android-eclipse
868 boleto
869 fpdf
870 webdesign
871 exportacao
872 datasnap
873 order
874 foreign-key
875 hql
876 dataset
877 cross-browser
878 diagrama
879 font-awesome
880 teoria-da-computacao
881 powerbi
882 append
883 sql-injection
884 gerenciamento-de-projetos
885 sublime-text-2
886 unicode
887 seletores
888 uitableview
889 cores
890 undefined
891 fopen
892 datagrid
893 module
894 twig
895 ec2
896 pivot
897 submenu
898 website
899 malloc
900 laravel-5.6
901 slimframework
902 phpexcel
903 io
904 argumento
905 number-format
906 apache-poi
907 valor-monetario
908 acessibilidade
909 file-get-contents
910 laravel-5.2
911 double
912 php-autoload
913 bat
914 asp-net-core
915 phpunit
916 tooltip
917 correios
918 quebras-de-linha
919 object-pascal
920 android-permission
921 viewmodel
922 google-sheets
923 timezone
924 sincronizacao
925 arquitetura-computadores
926 instagram
927 devise
928 atom
929 scraping
930 require
931 jqgrid
932 elasticsearch
933 makefile
934 comentarios
935 chart
936 dataannotations
937 ejs
938 tensorflow
939 ng-repeat
940 s3
941 shiny
942 android-service
943 stream-de-arquivos
944 widget
945 porcentagem
946 google-api
947 byte
948 programacao-estruturada
949 alertdialog
950 gem
951 cep
952 play-framework
953 lumen
954 laravel-5.5
955 icons
956 notepad++
957 buffer
958 auto-incremento
959 android-manifest
960 mingw
961 organizacao-de-projeto
962 bower
963 virtualbox
964 password
965 vbscript
966 ui
967 planilhas
968 controle-de-acesso
969 primefaces-3.5
970 owin
971 cms
972 preg-replace
973 entrada-de-dados
974 .exe
975 grails
976 gtk
977 cross-domain
978 metodologia
979 django-admin
980 xsd
981 mock
982 component
983 collapse
984 android-alarmmanager
985 extensao
986 firebase-cloud-messaging
987 apache2
988 scheduler
989 clone
990 broadcastreceiver
991 ssis
992 inteiros
993 subdominio
994 refatoracao
995 multiple-file-upload
996 prolog
997 singleton
998 safari
999 angularjs-services
1000 windows-service
1001 iis-7
1002 core-data
1003 mockito
1004 armazenamento
1005 tupla
1006 scala
1007 sqlalchemy
1008 texto
1009 gulpjs
1010 d3.js
1011 querystring
1012 jcombobox
1013 integracao-continua
1014 sublime-text-3
1015 artisan
1016 option
1017 django-queryset
1018 jdk
1019 redirect
1020 load
1021 attribute
1022 sobrecarga
1023 android-webview
1024 jsx
1025 metodos-estaticos
1026 signalr
1027 captcha
1028 jax-rs
1029 responsive
1030 jasminsoftware
1031 vuetify
1032 firebird-2.5

1033 jbutton
1034 fortran
1035 drawable
1036 funcao-anonima
1037 encode
1038 normalizacao
1039 silex
1040 i18n
1041 resultset
1042 oledb
1043 delphi-xe6
1044 awt
1045 placeholder
1046 arvore
1047 google-app-engine
1048 momentjs
1049 soma
1050 efd-reinf
1051 keras
1052 arredondamento
1053 cucumber
1054 do-while
1055 tcp
1056 phonegap-buillder
1057 locaweb
1058 babel
1059 publish
1060 redis
1061 cmake
1062 vim
1063 webdriver
1064 sed
1065 esocial
1066 resources
1067 vps
1068 jtextfield
1069 requirejs
1070 xamarin.ios
1071 lado-do-cliente
1072 variaveis-globais
1073 timeout
1074 jquery-plugins
1075 ms-dos
1076 phpstorm
1077 big-data
1078 handlebars.js
1079 asp.net-core-web-api
1080 mono
1081 reporting-services
1082 ecmascript-5
1083 wysiwyg
1084 facebook-javascript-sdk
1085 winapi
1086 telerik
1087 fancybox
1088 javamail
1089 dockerfile
1090 angular-5
1091 totalcross
1092 onesignal
1093 style
1094 dropdownlist
1095 range
1096 background-image
1097 html-table
1098 markdown
1099 qml
1100 app.config
1101 marker
1102 ponto-flutuante
1103 inicializacao
1104 angular-8
1105 react-hooks
1106 series-temporais
1107 solid
1108 listener
1109 library
1110 office
1111 jasmine
1112 sweetalert
1113 lua-table
1114 porta-serial
1115 soapclient
1116 jstl
1117 wampserver
1118 expo
1119 listbox
1120 graphql
1121 agendamento
1122 notificacao
1123 panel
1124 java-bean
1125 height
1126 zurb-foundation
1127 indentacao
1128 background-color
1129 ssms
1130 lib
1131 area-de-transferencia
1132 extjs
1133 jackson
1134 closures
1135 oracle10g
1136 eval
1137 resolucao-de-tela
1138 testes-funcionais
1139 sum
1140 substring
1141 yii2
1142 split
1143 service-worker
1144 swagger
1145 redimensionamento
1146 x86
1147 modulo
1148 pgadmin
1149 outlook
1150 facebook-api
1151 itext
1152 guzzle
1153 mean
1154 xmldocument
1155 tcpclient
1156 histograma
1157 geometria
1158 c++14
1159 operador-like
1160 mint
1161 coordenadas
1162 anaconda
1163 digitalocean
1164 host
1165 ui-router
1166 taxonomia
1167 setinterval
1168 codigo-de-barras
1169 replicacao
1170 quicksort
1171 prestashop
1172 angular-module
1173 oracle12c
1174 jest
1175 sql-server-express
1176 warnings
1177 tinymce
1178 parallax
1179 internet-explorer-8
1180 rspec
1181 lazy-loading
1182 rmi
1183 multi-tenancy
1184 clr
1185 touch
1186 twitter-api
1187 set
1188 pseudocodigo
1189 sortable
1190 aes
1191 constraint
1192 helper
1193 distinct
1194 middleware
1195 webrequest
1196 diff
1197 webserver
1198 jee
1199 jquery-masked-input
1200 bcrypt
1201 php-5
1202 websql
1203 game
1204 vue-router
1205 azure-app-service
1206 tidyverse
1207 custom-post-types
1208 404
1209 eclipse-link
1210 reconhecimento-de-voz
1211 android-sdk
1212 gzip
1213 meteor
1214 lightbox
1215 migracao
1216 jade
1217 settimeout
1218 branch
1219 toggle
1220 uri
1221 casos-de-uso
1222 cielo
1223 rmarkdown
1224 chatterbot
1225 design-de-linguagem
1226 polymer
1227 localizacao
1228 rss
1229 mime-type
1230 sistemas-distribuidos
1231 fxml
1232 plpgsql
1233 crontab

1234 alamofire
1235 thumbnail
1236 using
1237 datapicker
1238 resize
1239 controles
1240 ipython-notebook
1241 delphi-berlin
1242 gd
1243 router
1244 pyqt-4
1245 picturebox
1246 passagem-por-referencia
1247 ioc
1248 yaml
1249 httpclient
1250 querybuilder
1251 virtualenv
1252 heap
1253 redmine
1254 iis-express
1255 multiple
1256 temas
1257 controle-de-fluxo
1258 sap
1259 log4j
1260 metodologia-agil
1261 display
1262 driver
1263 ksoap2
1264 popover
1265 imagemagick
1266 align
1267 avd
1268 awk
1269 asmx
1270 ocr
1271 csrf
1272 fast-reports
1273 jaxb
1274 prototipo
1275 laravel-auth
1276 usabilidade
1277 xss
1278 apply
1279 xml-schema
1280 sdl
1281 vuex
1282 async-await
1283 footer
1284 yarn
1285 ping
1286 formdata
1287 player
1288 union
1289 preprocessador
1290 response
1291 sha1
1292 informix
1293 allegro
1294 network
1295 getjson
1296 primefaces-5.2
1297 jtextarea
1298 kernel
1299 iso-8859-1
1300 3d
1301 autenticacao-http
1302 varchar
1303 fetch
1304 sqlfiddle
1305 webclient
1306 convencoes
1307 tclientdataset
1308 variaveis-de-ambiente
1309 html2canvas
1310 joptionpane
1311 scenebuilder
1312 blog
1313 ad
1314 strtotime
1315 divisao
1316 mips
1317 intellij
1318 graphics
1319 android-gradle
1320 autoload
1321 fragment
1322 ecmascript
1323 java-processing
1324 angular-cli
1325 razor-pages
1326 fatorial
1327 delphi-xe4
1328 combinacoes
1329 indy-10
1330 qtgui
1331 erp
1332 html-option
1333 rsa
1334 visao-computacional
1335 cdn
1336 authorization
1337 html-agility-pack
1338 window-builder
1339 fullscreen
1340 busca-binaria
1341 internacionalizacao
1342 jquery-animate
1343 visual-studio-2010
1344 dependencias
1345 transition
1346 bitwise
1347 coffeescript
1348 microsoft-word
1349 python-internals
1350 rich-faces
1351 pagina-de-erro
1352 codigo-aberto
1353 bit
1354 dbgrid
1355 struts2
1356 chromium
1357 aspx
1358 sklearn
1359 adminlte
1360 delegate
1361 stringbuilder
1362 jlabel
1363 react-dom
1364 jekyll
1365 bundle
1366 jquery.validate
1367 prepared-statement
1368 nodemailer
1369 scaffold
1370 pom
1371 typeahead
1372 xslt
1373 windows-8
1374 bootstrap-glyphicons
1375 validator
1376 openshift
1377 composicao
1378 summernote
1379 64-bits
1380 innodb
1381 favicon
1382 numeros
1383 jsoup
1384 parse.com
1385 genymotion
1386 h2database
1387 easyphp
1388 visibilidade
1389 libgdx
1390 model-validation
1391 trait
1392 pentaho
1393 business-intelligence
1394 pic
1395 minificacao
1396 job
1397 mapas
1398 pyinstaller
1399 xdebug
1400 android-ndk
1401 geolocation
1402 file-permission
1403 robots
1404 event-listener
1405 phantomjs
1406 advanced-custom-fields
1407 private
1408 webrtc
1409 windows-7
1410 acl
1411 recuperacao-de-erros
1412 administracao-postgresql
1413 principios-de-programacao
1414 firewall
1415 chave-composta
1416 rgb
1417 db2
1418 interpolacao-de-string
1419 lamp
1420 ninject
1421 sftp
1422 webkit
1423 carregamento
1424 apple-push-notification
1425 meteorjs
1426 indexof
1427 drawing
1428 uitableviewcell
1429 jsonp
1430 java-ee-7
1431 2d
1432 sha-256
1433 locale
1434 compressao

1435	servidor-de-aplicacao	1502	memory-leaks	1569	traducao
1436	simplexml	1503	angular-ui-bootstrap	1570	assinatura
1437	css-border	1504	partialview	1571	internet
1438	formulario-submeter	1505	indexeddb	1572	maskmoney
1439	brasil	1506	escopo	1573	engenharia-reversa
1440	release	1507	html-helpers	1574	microsoft-edge
1441	fibonnaci	1508	google-cloud	1575	c++-builder
1442	deprecated	1509	neo4j	1576	engine
1443	collation	1510	.net-assembly	1577	mdi
1444	web2py	1511	slideshow	1578	scrum
1445	bxslider	1512	formula	1579	barplot
1446	imap	1513	dotnet-cli	1580	rotacao
1447	paypal-api	1514	travis-ci	1581	w3c-validador
1448	contact-form-7	1515	newsletter	1582	regra-de-negocio
1449	maps-javascript-api	1516	pentaho-kettle	1583	timepicker
1450	mercadopago	1517	hsqldb	1584	campos
1451	npgsql	1518	wget	1585	32-bits
1452	void	1519	numeros-binarios	1586	matematica-financeira
1453	elixir	1520	sfml	1587	asp.net-membership
1454	identity	1521	jax-ws	1588	javadoc
1455	deep-learning	1522	zeos	1589	try-finally
1456	maquina-virtual	1523	antivirus	1590	django-1.8
1457	sybase	1524	decorator-pattern	1591	interpretador
1458	cpu	1525	jsfiddle	1592	google-api-client
1459	animate	1526	asp.net-mvc-routing	1593	nan
1460	zabbix	1527	sas	1594	interoperabilidade
1461	generators	1528	vertical-align	1595	rds
1462	dispose	1529	automatos	1596	stylus
1463	smarty	1530	richtextbox	1597	jni
1464	expressoes	1531	shortcode	1598	forms-authentication
1465	oracle11c	1532	soa	1599	pagespeed-insights
1466	mongodb-csharp	1533	css-display	1600	ldap
1467	software	1534	php-domdocument	1601	default
1468	tuple	1535	appstore	1602	microcontrolador
1469	each	1536	jquery-mask-plugin	1603	emberjs
1470	selected-option	1537	tortoisesvn	1604	vpn
1471	semantic-ui	1538	create	1605	webcam
1472	url-amigavel	1539	openldap	1606	intel
1473	requisitos	1540	x64	1607	arm
1474	hadoop	1541	falha-de-segmentacao	1608	xamarin-studio
1475	css-seletores	1542	data-science	1609	yield
1476	ini	1543	restricao-unico	1610	pouchdb
1477	scipy	1544	tinymce-4	1611	couchdb
1478	deteccao-de-colisao	1545	bdd	1612	app-compat-v7
1479	escalabilidade	1546	input-output	1613	applet
1480	viewport	1547	google-calendar	1614	alias
1481	okhttp	1548	elementos	1615	flex
1482	json.net	1549	pyautogui	1616	simple-injector
1483	asp.net-web-api-2	1550	data-warehouse	1617	transform
1484	diretivas-de-compilacao	1551	pasta	1618	sobrescrita
1485	tablemodel	1552	clojure	1619	pseudo-classes
1486	jpanel	1553	zoom	1620	tls
1487	posix	1554	realmdb	1621	body
1488	laravel-5.8	1555	pixels	1622	remoto
1489	http-post	1556	google-location-api	1623	telegram-bot
1490	dump	1557	webgrid	1624	bootstrap-datepicker
1491	actions	1558	algebra-booleana	1625	flyway
1492	mysqldump	1559	google-cloud-platform	1626	keyframes
1493	resharper	1560	webhook	1627	firebase-cloud-functions
1494	reprodutor-de-video	1561	passport-js	1628	php-7.1
1495	embarcadero	1562	tabs	1629	quasar
1496	z-index	1563	embed	1630	boot
1497	compressao-de-imagem	1564	intellisense	1631	php-fpm
1498	stl	1565	imutabilidade	1632	inno-setup
1499	lisp	1566	modularizacao	1633	ddos
1500	google-analytics-api	1567	firefox-extensao	1634	adapter
1501	linked-server	1568	silverlight	1635	jinja2

1636 css-font-size
1637 selecteditem
1638 mamp
1639 addclass
1640 mootools
1641 sonarqube
1642 laravel-6
1643 drupal
1644 vtex
1645 clickonce
1646 cassandra
1647 storyboard
1648 efeito
1649 adobe-air
1650 jtree
1651 whm
1652 navegacao
1653 extension-method
1654 nullable
1655 database-first
1656 ant
1657 lucene
1658 culture
1659 sidebar
1660 powerpoint
1661 dropzonejs
1662 adobe-flex
1663 ciclo-infinito
1664 wmi
1665 observer-pattern
1666 scripting
1667 sql-view
1668 internet-explorer-9
1669 google-search
1670 java2d
1671 nltk
1672 checkout
1673 rust
1674 native
1675 userform
1676 ibm
1677 ancora
1678 postgis
1679 newtonsoft
1680 exec
1681 symfony4
1682 linguagens-formais
1683 c++17
1684 jinternalframe
1685 psql
1686 diagrama-de-classes
1687 multer
1688 tampermonkey
1689 marcadores
1690 laravel-collective
1691 break
1692 pug
1693 simpledateformat
1694 nativescript
1695 jdialog
1696 reactive-programming
1697 monodevelop
1698 uitextfield
1699 google-cloud-messaging
1700 viewpager
1701 rvest
1702 routing
1703 countdown
1704 construct-2
1705 programacao-dinamica
1706 open-graph
1707 grep
1708 psr-4
1709 breakpoints
1710 swiftmailer
1711 cobol
1712 protecao
1713 postgresql-10
1714 camera
1715 dev-tools
1716 select-limit
1717 handler
1718 angularjs-factorys
1719 padding
1720 xml-namespaces
1721 stdclass
1722 moq
1723 colecao
1724 app-hibrido
1725 flexslider
1726 api-youtube
1727 mailer
1728 inputstream
1729 span
1730 pyopengl
1731 cgi
1732 ngroute
1733 when
1734 swingworker
1735 console-application
1736 swift-playground
1737 mapkit
1738 android-styles
1739 jfreechart
1740 group-concat
1741 uinavigationcontroller
1742 sensibilidade-de-caixa
1743 formatacao-condicional
1744 jmeter
1745 android-context
1746 html5-canvas
1747 monitoramento
1748 laravel-mix
1749 iot
1750 cardinalidade
1751 telegram
1752 input-mask
1753 devops
1754 prisma
1755 mqtt
1756 calendario
1757 array-push
1758 chosen-jquery-plugin
1759 google-storage
1760 libreoffice
1761 dados-hierarquicos
1762 pseudo-elementos
1763 delphi-2010
1764 watson
1765 nodemon
1766 delphi-tokyo
1767 sendmail
1768 java-9
1769 angular-fire-2
1770 usercontrol
1771 windows-ce
1772 openmp
1773 rtf
1774 ipv6
1775 sql-server-2005
1776 flot
1777 abap
1778 thread-safety
1779 metaprogramacao
1780 abstracao
1781 weka
1782 angular-providers
1783 wxwidgets
1784 palavras-reservadas
1785 css-border-radius
1786 equacoes
1787 proguard
1788 setup
1789 internal-server-error
1790 case-insensitive
1791 fade
1792 erb
1793 equals
1794 activex
1795 guid
1796 animate.css
1797 sitemap
1798 report-builder
1799 codigo-postal
1800 solr
1801 clipper
1802 ajaxtoolkit
1803 edicao-de-imagens
1804 enumerable
1805 spam
1806 goto
1807 fabric.js
1808 delphi-6
1809 office-2013
1810 webmethods
1811 extjs-4
1812 posicao-do-cursor
1813 lapply
1814 http-metodos
1815 erlang
1816 kohana
1817 summary
1818 ses
1819 opacity
1820 lm
1821 jquery-ui-multiselect
1822 eslint
1823 rabbitmq
1824 floatingactionbutton
1825 k-means
1826 excel-userform
1827 procv
1828 ansi
1829 jsonobject
1830 comandos
1831 formula
1832 keystore
1833 three.js
1834 offset
1835 mkdir
1836 timespan

1837 web-component
1838 slice
1839 longblob
1840 utf8-decode
1841 mouseover
1842 angular2-http
1843 linkagem
1844 progress-4gl
1845 elastic-beanstalk
1846 hashcode
1847 http-status-code-403
1848 bind
1849 web-worker
1850 portabilidade
1851 uuid
1852 jodatime
1853 blender
1854 anuncio
1855 queue
1856 feed
1857 threadpool
1858 angular2-directives
1859 phpmail
1860 operador-logico
1861 camel-case
1862 user-agent
1863 asp.net-mvc-helper
1864 scope
1865 boleto.net
1866 with
1867 tabela-temporaria
1868 der
1869 spark
1870 projeto-de-compiladores
1871 skype
1872 azure-devops
1873 material
1874 usb
1875 lattice
1876 benchmark
1877 longpolling
1878 crosswalk
1879 analise-de-dados
1880 financas
1881 spl
1882 delay
1883 metodos-magicos
1884 t4
1885 horizontal
1886 sql-server-localdb
1887 spring-tools-suite
1888 google-plus
1889 basic
1890 sudo
1891 gdb
1892 jquery-validation-engine
1893 transicao
1894 ado
1895 actionscript
1896 sweetalert2
1897 java-11
1898 karma
1899 styled-components
1900 rxjs
1901 servlet-3
1902 orientacao-a-aspectos
1903 httr
1904 tesseract
1905 factory
1906 fadein
1907 sanitize
1908 hyper-v
1909 mcrypt
1910 const
1911 legendas
1912 mormot
1913 angularjs-components
1914 nasm
1915 demoiselle-certificate
1916 bootgrid
1917 margin
1918 rotativa
1919 amp
1920 passagem-por-valor
1921 lock
1922 homestead
1923 progressive-web-apps
1924 zend-framework-3
1925 anova
1926 tempo-de-vida
1927 dbeaver
1928 estatistica-bootstrap
1929 reshape
1930 icone
1931 clang
1932 assert
1933 rxjava
1934 inline
1935 aba
1936 c3p0
1937 github-pages
1938 ui-draggable
1939 descompilacao
1940 ofuscacao
1941 vmware
1942 cycle2
1943 html5-history
1944 data.table
1945 arquivo-header
1946 memory-limit
1947 mobile-safari
1948 pojo
1949 directx
1950 blazor
1951 restsharp
1952 cpython
1953 ormlite
1954 mercadolivre
1955 gateway
1956 accordion
1957 underscore.js
1958 vimeo
1959 delphi-xe2
1960 ionic-popup
1961 wildcard
1962 context-menu
1963 kinect
1964 camera
1965 rtti
1966 file-put-contents
1967 afnetworking
1968 memcached
1969 hashset
1970 jndi
1971 logcat
1972 telnet
1973 fila-thread
1974 webgl
1975 apache-solr
1976 representacao-numerica
1977 bytecode
1978 simplexml-load-string
1979 phtml
1980 photoshop
1981 f#
1982 lubridate
1983 saas
1984 dreamweaver
1985 actionlistener
1986 firebug
1987 serverless
1988 env
1989 httpd
1990 crawling
1991 data-attribute
1992 mocha
1993 anexo
1994 class-library
1995 scilab
1996 hive
1997 token-ajax
1998 cosmos-db
1999 videojs
2000 liquibase
2001 editor
2002 fisica
2003 contains
2004 notifyicon
2005 hack
2006 padrao-de-texto
2007 pthreads
2008 cli
2009 sefaz
2010 try-with-resources
2011 grid-layout
2012 clusterizacao
2013 tempo-real
2014 8086
2015 tcpdf
2016 arrow-functions
2017 www
2018 natural-language
2019 jboss-weld
2020 full-text
2021 tray
2022 pyodbc
2023 hdf5
2024 geckodriver
2025 blockchain
2026 compartilhar
2027 match
2028 child
2029 mouse
2030 jmenu
2031 sip
2032 jpg
2033 self
2034 make
2035 laravel-5.1
2036 ci
2037 codeigniter-2

2038 orientation
2039 office-open-xml
2040 empty
2041 toolbox
2042 jquery-remodal
2043 portugues
2044 idle-ide
2045 bootstrap-table
2046 dql
2047 phalcon-php
2048 resteasy
2049 jquery-toggleclass
2050 postfix
2051 visual-studio-online
2052 destrutor
2053 sat-cfe
2054 route53
2055 spl-autoloader-register
2056 onblur
2057 dax
2058 export
2059 lawtex
2060 kubernetes
2061 main
2062 pymongo
2063 ng-controller
2064 procedural
2065 julia
2066 psr
2067 extension
2068 google-adsense
2069 globalizacao
2070 metroframework
2071 war
2072 bootsfaces
2073 contexto
2074 html-meta
2075 jcrop
2076 smartdevice
2077 jaas
2078 itunes
2079 event-dispatch-thread
2080 autohotkey
2081 data-manipulation
2082 programacao-imperativa
2083 analise-lexica
2084 pipe
2085 toasts
2086 validation
2087 profiling
2088 lwjgl
2089 pagedlist
2090 django-1.9
2091 java-ee-6
2092 standalone
2093 background-services
2094 xna
2095 codificacao-de-conteudo
2096 spss
2097 unobtrusive
2098 historico
2099 google-sites
2100 django-filter
2101 log4net
2102 owl
2103 publicidade
2104 jms
2105 local-notification
2106 kali-linux
2107 php-cli
2108 sourcetree
2109 php-domxpath
2110 application-service
2111 rtsp
2112 openlayers-3
2113 partial
2114 symfony3
2115 compass
2116 openpyxl
2117 schema
2118 global
2119 nuxt
2120 visual-studio-2019
2121 this
2122 workspace
2123 drawer-layout
2124 angular-google-maps
2125 jstree
2126 userscripts
2127 pydev
2128 prototypejs
2129 php-cpp
2130 biginteger
2131 logo
2132 dropbox
2133 samba
2134 dateinterval
2135 portugues-do-brasil
2136 stack-trace
2137 angularfire
2138 dry
2139 iis-web-deploy
2140 inphinit
2141 active-model-serializers
2142 rowsorter
2143 gulpfile
2144 glmer
2145 sendgrid
2146 msys2
2147 datetime-local
2148 pooling
2149 dynamics-crm
2150 variavel-estatica
2151 numeracao
2152 jdesktoppane
2153 semaforo
2154 typedef
2155 .net-compiler-platform
2156 quickreport
2157 backgroundworker
2158 sqlite-studio
2159 picasso
2160 codigo-fonte
2161 interop
2162 asterisk
2163 bitcoin
2164 rowfilter
2165 git-flow
2166 removechild
2167 union-all
2168 apex
2169 vhdl
2170 code-golf
2171 plotly
2172 powerquery
2173 fs
2174 glpi
2175 tela
2176 tabela-verdade
2177 hangfire
2178 ratchet
2179 mustache.js
2180 jdatechooser
2181 pyside
2182 uiviewcontroller
2183 programacao-reativa
2184 superclasse
2185 bottomnavigationview
2186 real-time
2187 pep8
2188 301
2189 render
2190 sh
2191 jqplot
2192 fastcgi
2193 logging
2194 gwt
2195 filezilla
2196 htmlentities
2197 hhvm
2198 restify
2199 factors
2200 android-billing
2201 varbinary
2202 protected
2203 public
2204 spritesheet
2205 variancia
2206 caminho-relativo
2207 look-and-feel
2208 gpl
2209 proguard-android
2210 yeoman
2211 biometria
2212 material-ui
2213 var-dump
2214 bittorrent
2215 getters
2216 dbase
2217 imagick
2218 css-padding
2219 log4j2
2220 trello
2221 tempo-de-resposta
2222 uisearchbar
2223 plesk
2224 c++-cli
2225 tipo-de-conteudo
2226 wordcloud
2227 aop
2228 microcontrolador-8051
2229 nspredicate
2230 capistrano
2231 sslstream
2232 uiview
2233 play-store-console
2234 store
2235 flyweight
2236 es6
2237 gorm
2238 unreal-engine

2239 system-tray
2240 minitest
2241 path-separator
2242 azure-tables
2243 backbone.js
2244 pdoexception
2245 referencia-circular
2246 manipulador-de-evento
2247 tarefa
2248 read.table
2249 ipv4
2250 windows-mobile
2251 .env
2252 interceptor
2253 habtm
2254 rollback
2255 polyfill
2256 analise-numerica
2257 quartz-scheduler
2258 android-room
2259 webp
2260 lavarel
2261 jta
2262 udp
2263 deepin
2264 picture
2265 android
2266 resources
2267 metodo
2268 abstrato
2269 w3c
2270 datasource
2271 boost
2272 crystal
2273 bootstrap-validator
2274 vue-resource
2275 dynamic
2276 glm
2277 bot-framework
2278 jcalendar
2279 framework7
2280 geany
2281 checksum
2282 bits
2283 linkedin
2284 extern
2285 data
2286 class
2287 kcfinder
2288 fullpage.js
2289 patch
2290 sns
2291 api
2292 key
2293 python
2294 content-type
2295 pgadmin3
2296 cefsharp
2297 tomcat8
2298 linguagem-de-marcacao
2299 delphi-seatle
2300 npoi
2301 scrollto
2302 xamarin.portable
2303 strategy-pattern
2304 slimframwork
2305 sql
2306 server
2307 cout
2308 kml
2309 kendo-asp.net-mvc
2310 angular
2311 voip
2312 tabbarcontroller
2313 jlist
2314 memo
2315 confirm
2316 harbour
2317 parsley
2318 putty
2319 psd
2320 spliterator
2321 margens
2322 frames
2323 mschart
2324 mercurial
2325 css-centering
2326 bibtex
2327 bem
2328 jelastic
2329 asp.net-profile
2330 poco
2331 axis2
2332 django-generic-views
2333 cuda
2334 jssc
2335 print
2336 urllib
2337 descriptors
2338 leaflet
2339 spyder
2340 ambiente
2341 http2
2342 bubblesort
2343 pynput
2344 operators
2345 pupeteer
2346 tablelayout
2347 prototipacao
2348 angular-chart
2349 inappbrowser
2350 lollipop
2351 google-translate
2352 balanceamento-carga
2353 tortoisegit
2354 iptables
2355 breadcrumbs
2356 mediawiki
2357 slf4j
2358 loopback
2359 edi
2360 workflow
2361 sublimerepl
2362 xtrareport
2363 qtwebkit
2364 smart-pointer
2365 fluentvalidation
2366 watch
2367 w3.css
2368 literal
2369 cte
2370 opencsv
2371 user-interface
2372 spring-rest
2373 capitalizacao
2374 sql-table
2375 jquery-data
2376 bigint
2377 buffer-overflow
2378 analise-estatica
2379 hibernate-envers
2380 grafico-de-dispersao
2381 bbcode
2382 advogado-de-linguagem
2383 trunk
2384 maquina-de-estado
2385 libvlc
2386 opencl
2387 qcamera
2388 wicket
2389 dsp
2390 site-links
2391 opensuse
2392 ddl
2393 octal
2394 morris
2395 moip
2396 sqlbulkcopy
2397 pechkin
2398 android-databinding
2399 onedrive
2400 teste-de-mesa
2401 fetchall
2402 php-document
2403 semver
2404 sso
2405 elm
2406 curto-circuito
2407 nfs-e
2408 cqrs
2409 laravel-5.7
2410 reportlab
2411 snmp
2412 circuitos-logicos
2413 logback
2414 react-select
2415 condicao-de-corrida
2416 ecmascript-7
2417 dtd
2418 esb
2419 flat-ui
2420 onsen
2421 multi-core
2422 apache-tiles
2423 sdn
2424 android-assets
2425 dnx
2426 abntex2
2427 xmpp
2428 sbt
2429 alfresco
2430 inferencia
2431 rgraph
2432 landscape
2433 doctype
2434 caelum-stella
2435 dkim
2436 contents
2437 use-strict
2438 mathematica
2439 playlist

2440	uiwebview	2507	love	2574	mysql-num-rows
2441	mask-plugin	2508	cracker	2575	jalert
2442	clock	2509	delphi-2007	2576	docker-windows
2443	october-cms	2510	plyr	2577	es5
2444	cortex-a	2511	federation-authentication	2578	wildfly13
2445	mlp	2512	sse	2579	pkcs11
2446	uitabbar	2513	captura-de-tela	2580	postgresql-11
2447	jit	2514	nuvem-de-tags	2581	seaborn
2448	glob	2515	data-uri	2582	react-testes
2449	jdom	2516	irb	2583	api-gateway
2450	salt	2517	jquery-blockui	2584	knex.js
2451	acid	2518	poligono	2585	xib
2452	inducao-de-tipo	2519	triangulo	2586	codename-one
2453	ashx	2520	masked-input-plugin	2587	sparql
2454	ontologia	2521	heredoc	2588	formview
2455	plone	2522	ntp	2589	soundex
2456	xcode-8	2523	pop3	2590	customelements
2457	xeditable	2524	java-me	2591	easeljs
2458	whmcs	2525	cx-freeze	2592	variaveis-de-instancia
2459	java-10	2526	metaclass	2593	windows-xp
2460	next	2527	tapply	2594	dbunit
2461	configurationmanager	2528	webstorage	2595	l10n
2462	jslider	2529	html2pdf-js	2596	freetds
2463	teste-stress	2530	django-2.2	2597	pinvoke
2464	application-insights	2531	keylogger	2598	openfire
2465	powerpivot	2532	stackexchange	2599	windows-mobile-5
2466	webengine	2533	jetty	2600	mapply
2467	database-partitioning	2534	end-to-end	2601	sapply
2468	prism	2535	defer	2602	minimos-quadrados
2469	cypher	2536	desinstalacao	2603	volatile
2470	e2e	2537	imgur	2604	boilerplate
2471	ipad	2538	sinatra	2605	mtv
2472	google-play-services	2539	crux-framework	2606	sonarlint
2473	jsonarray	2540	syntactic-sugar	2607	rnorm
2474	dbset	2541	pjax	2608	qtranslate
2475	floating	2542	nagios	2609	apc
2476	key-value	2543	throw	2610	internet-explorer-7
2477	bookmarklet	2544	apns	2611	arvores-de-expressao
2478	openlayers	2545	stringbuffer	2612	modelo-de-atores
2479	phaser-js	2546	htpasswd	2613	froala-editor
2480	opera	2547	uninfe	2614	criptomoeda
2481	cartao-de-credito	2548	physx	2615	hoisting
2482	dpi	2549	packagist	2616	mahapps
2483	ng-animate	2550	mql4	2617	mediarecorder
2484	webstorm	2551	google-openid	2618	getusermedia
2485	hugo	2552	uiimage	2619	postsharp
2486	cross-compiler	2553	arc	2620	django-channels
2487	corotina	2554	iup	2621	identificador
2488	.net-standard	2555	libgit2sharp	2622	crystal-lang
2489	tipo-anonimo	2556	syslog	2623	browserify
2490	pagina-web	2557	iphone-4	2624	teste-de-carga
2491	delphi-xe7	2558	graphael	2625	godaddy
2492	jigsaw	2559	durandal	2626	django-annotate
2493	luis	2560	gammu	2627	dados-abertos
2494	subclasse	2561	vbulletin	2628	mstest
2495	hapijs	2562	angular-kendo	2629	create-project
2496	syntax-highlight	2563	mdx	2630	lombok
2497	amazon-cloud-front	2564	olap	2631	funcoes-puras
2498	chosen	2565	series-temporais	2632	gamificacao
2499	jira	2566	yml	2633	azure-sql
2500	vulnerabilidade	2567	idempotencia	2634	d
2501	logica-difusa	2568	indicative	2635	webassembly
2502	glsl	2569	ts	2636	overhead
2503	lme4	2570	discord.py	2637	sizeof
2504	greensock-gsap	2571	formik	2638	document-db
2505	rscript	2572	react-router-dom	2639	phplot
2506	querydsl	2573	rvm	2640	febraban

2641 zlib
2642 snake-case
2643 identity-server
2644 servicebus
2645 nightwatch
2646 fusioncharts
2647 latin1
2648 boletophp
2649 ncurses
2650 minizinc
2651 mixin
2652 drc
2653 rgss3
2654 aspxrichedit
2655 socialite
2656 dagger
2657 relacao
2658 foreach-jstl
2659 dryioc
2660 virtualenvwrapper-win
2661 polly
2662 circuitbreaking
2663 kanban
2664 keystonejs
2665 vuetable
2666 pgadmin4
2667 drive
2668 apache-kafka
2669 previous
2670 apache-nifi
2671 delphi-4
2672 scratch
2673 catalina
2674 android-content-provider
2675 tableau
2676 metronic
2677 debug-historico
2678 nicedit
2679 intellitrace
2680 angular-views
2681 multiplicidade
2682 jmenubar
2683 uglify-js
2684 entity-framework-v1
2685 keyup
2686 pear
2687 lftp
2688 xdocument
2689 marshalling
2690 baas
2691 cygwin
2692 bison
2693 geometria-computacional
2694 monogame
2695 python2exe
2696 paradox
2697 poppler
2698 j2me
2699 gpgpu
2700 love2d
2701 algoritmo-fonetico
2702 foundation
2703 demoiselle-nimble
2704 aptana
2705 pdt
2706 espaco-em-branco
2707 backbone
2708 slim-lang
2709 fixtures
2710 db4o
2711 liferay
2712 ecm
2713 abot
2714 log-de-eventos-windows
2715 microsoft-help-viewer
2716 twitter-stream-api
2717 stmt
2718 vuforia
2719 castle-windsor
2720 rtmpdump
2721 cufon
2722 microcontroladores-8bits
2723 multimedia
2724 dwoo
2725 demoiselle-infra
2726 wtforms
2727 lucid
2728 aspectj
2729 sistemas-digitais
2730 touch-events
2731 css-shape
2732 carbon
2733 msbuild
2734 autofocus
2735 outofrange
2736 analise-de-texto
2737 bottle
2738 ocaml
2739 supervisor
2740 contratos
2741 serilog
2742 nunit
2743 explode
2744 fsockopen
2745 resiliencia
2746 aprendizado-supervisionado
2747 biblioteca-estatica
2748 seeders
2749 portugol-studio
2750 oracle18c
2751 mineracao-de-textos
2752 tablesorter
2753 scheme
2754 shiny-server
2755 php-doc
2756 rhino
2757 newline
2758 processwire
2759 microdata
2760 sped
2761 efd
2762 minify-maven-plugin
2763 selectize.js
2764 pubcenter
2765 rich-snippets
2766 xcache
2767 exploit
2768 redips.drag
2769 j
2770 integridade
2771 testcase
2772 input-type-date
2773 powerpanel
2774 nesbox
2775 jboss-modules
2776 grafico-condicional
2777 ogg
2778 bass
2779 tor
2780 ipython
2781 pgm
2782 indy-9
2783 amd
2784 homebrew
2785 html5-data
2786 grafana
2787 valgrind
2788 delphi-5
2789 funcao-discriminante
2790 react-navigation-fluid-transitions
2791 paginator
2792 brainfuck
2793 principio-liskov
2794 magic-quotes-gpc
2795 fabrio.io
2796 sql-server-2000
2797 iron-python
2798 bundler
2799 randomforest
2800 java-12
2801 wcag
2802 x-tags
2803 infotype
2804 jquery-1.6
2805 mpeg
2806 quantmod
2807 cocos2dx
2808 nfp
2809 xplaned
2810 opcode
2811 mandrill
2812 vitamio
2813 ember-data
2814 bitnami
2815 alamofireimage
2816 umbraco
2817 principal
2818 acbrsat
2819 php-8
2820 migrate-invoicy
2821 testes-unitarios
2822 medoo
2823 dml
2824 x++
2825 cocoapods
2826 kable
2827 pattern-matching
2828 http3
2829 jsonwebtoken
2830 sounddevice
2831 mapa-karnaugh
2832 numeros-magicos
2833 svn2git
2834 gif
2835 iscroll5
2836 eai
2837 ftprush
2838 hammerjs
2839 restfb
2840 carga-preguicosa

carga-antecipada
jmf
md6
expansion-files
apache-storm
twitter4j
httpapplication
sparkleshare
msdn
runloop
unidade-de-medida
dexie
portrait
shebang
boto
hasclass
qhull
flood
wowza
linkedhashmap
vxml
qxorm
winrar
netezza
gridlookupedit
cortana
ideone
descontinuado
service-container
pervasive
python-igraph
lita
bootstrap-toggle
ean-13
instant-app
visualbasic-powerpacks
ooxml
figma
cloud-search
snep
guava
jsonb
zinnia
q.js
devexpress-asp.net-ajax
torrent
heremaps
lookbehind
imagination-augmented
wow.js
code128
fasta
delphi-2006
stringr
edit-and-continue
gml
small-basic
dfsort
ocl
musl
shim
dpkg
gravatar
vsts
cpuid
documentfragment
variaveis-locais
calendario-gregoriano
tabela-zebrada
pagemethods
qftp
infomaker
mysql-find-in-set
izpack
android-loader
dispositivos-moveis
debounce
qliksense
elixir-mix
resample
php7.2
spring-web
spring-batch
solidity
ethereum
livelock
vaporware
entidade
scrutinizer
p2p
enzyme
eclipse-plugin
grails-plugin
wkhtmltopdf
godot
gdscript
microprogramacao
linguagem-de-maquina
validate
extend
jre
gnuplot
define-method
caret
java-13
zookeper
helix
digital-mesh
azuread
computacao-quantica
q#
oracle.manageddataaccess
.net-5
sqale
unsigned
ssr
traefik
whatwg
win10toast
uint8array
minio
influxdb
switchery
android-r8
bar-chart
code-coverage
rgl
yup
hta
renomear
redux-dev-tools
tailwindcss
sql2o
atribuicao-via-desestruturacao
pentester
hacking
togaf
numeraljs
npx
g++
gfortran
bigquery
pillow
rdd
flutter-desktop
color-thief
r-server
f-string
untagged
printwriter
typeorm
to-date
pydroid
windows-10-home
flexdashboard
previsao
forecast
wish
apache-phoenix
haxeflixel
ios-7
nsis
team-build
asr
transcricao-de-audio
tika
webix
icalendar
assincrono
assincrono
expressao-regular
expressao-regular
performance
base-de-dados
clipboard
azure
memoria
pesquisa
angular4
angular-4
webconfig
miniaturas
msqli
instalador
blade
finally
ficheiro
arvore-binaria
dictionary
multiplataforma
convencoes-de-nome
video
win32
subversion
asp.net-core-webapi
recursividade
report
blogspot
estrutura-de-codigo

3040	debugger	3107	exception	3174	regexp
3041	lambda	3108	hibridos	3175	sessao
3042	vs	3109	iterador	3176	sessao
3043	aspnet	3110	licenciamento	3177	testes-de-unidade
3044	mysql-order-by	3111	arrendondamento	3178	charset
3045	search	3112	sistemas-operacionais	3179	express.js
3046	realm	3113	db	3180	interface-grafica
3047	windowsforms	3114	dinamico	3181	certificado-digital
3048	windows-forms	3115	insert	3182	testflight
3049	rails	3116	mongo	3183	tfs
3050	decorator	3117	nullreferenceexception	3184	events
3051	index	3118	user-experience	3185	ie7
3052	permissao	3119	access	3186	zip
3053	responsive-design	3120	complexidade	3187	tipo-de-dados
3054	lexer	3121	relationship	3188	laraval
3055	servico	3122	xwindows	3189	node
3056	client-side	3123	asp.net-5	3190	encoding
3057	expressoes-booleanas	3124	enum	3191	codigo-postal
3058	ordenar	3125	codigo-livre	3192	perfilamento
3059	phonegap	3126	opengles	3193	animation
3060	asp.net-webforms	3127	procurar	3194	math
3061	mysql-update	3128	web	3195	mssql
3062	anuncio	3129	teoria-dos-grafos	3196	amazon
3063	sessoes	3130	estrutura-de-repeticao	3197	compilador
3064	imagens	3131	requisicao	3198	vuejs
3065	syntax	3132	depuracao	3199	character-set
3066	pre-processador	3133	dominio	3200	conversao
3067	sql-management-studio	3134	laravel4	3201	cordova
3068	swift3	3135	asp.net-mvc-6	3202	server-side
3069	funcoes	3136	php-unit	3203	printer
3070	indice-unico	3137	vue	3204	ie
3071	exe	3138	class	3205	classe
3072	nuvem	3139	codificacao-de-caracteres	3206	concatenacao
3073	conexao	3140	python2	3207	processing
3074	asp.net-vnext	3141	frontend	3208	calculo
3075	ie9	3142	indice	3209	diretiva
3076	vetor	3143	e-mail	3210	server
3077	visual-basic	3144	persistence	3211	case-sensitive
3078	net	3145	reactjs	3212	database
3079	winrt	3146	depurar	3213	tag
3080	function	3147	paradigma	3214	font-size
3081	nodejs	3148	variaveis	3215	fields
3082	logica	3149	iteracao	3216	relatorios
3083	modelagem-de-dados	3150	arquitetura	3217	activity
3084	animacoes	3151	audio	3218	google-drive
3085	generics	3152	primary-key	3219	prototype
3086	list	3153	paradigma-de-programacao	3220	csharp
3087	python3	3154	modelagem-de-bd	3221	reflexao
3088	expressoes-regulares	3155	grafico	3222	unique
3089	oop	3156	properties	3223	editorconfig
3090	opensource	3157	interface-de-usuario	3224	unity
3091	formulario	3158	usuario	3225	oo
3092	.htaccess	3159	idusuario	3226	pilha
3093	localizar	3160	usuarios	3227	chrome
3094	radio	3161	poo	3228	retorno
3095	carrossel	3162	mysql-table	3229	estrutura-linguagem
3096	url-reescrever	3163	recursivo	3230	api-stackexchange
3097	google-drive-sdk	3164	variavel	3231	compiler
3098	web-api	3165	gerencia-de-projeto	3232	left-join
3099	eloquent	3166	react.js	3233	elasticsearc
3100	metodos	3167	update	3234	encriptacao
3101	unity5	3168	sort	3235	traits
3102	ie8	3169	vscode	3236	normalizar
3103	funcao	3170	anotacoes	3237	design-de-software
3104	expressjs	3171	web.conf	3238	microservicos
3105	lacos	3172	construtores	3239	stream-de-video
3106	memory	3173	macos	3240	asp.net-core-mvc

3241	classe-abstrata	3266	select2	3291	android-recycler-view
3242	composer.json	3267	lambda-expressions	3292	constants
3243	js	3268	sonar	3293	operador
3244	mail	3269	caminho	3294	sinatxe
3245	vb	3270	htacess	3295	amazon-aws
3246	estrutura-de-diretorio	3271	bd	3296	semantica
3247	actionscript-2	3272	padroes-projetos	3297	htm
3248	importacao	3273	position	3298	matrix
3249	ast	3274	tags-input	3299	spa
3250	asynchronous	3275	go	3300	asp-classico
3251	autocomplete	3276	sped-efd	3301	.net-framework
3252	desktop	3277	java8	3302	automato
3253	insensibilidade-de-caixa	3278	bean	3303	filesystem
3254	mysql-select	3279	localizacao-geografica	3304	progress
3255	sqlsrv	3280	file	3305	speech-recognation
3256	folha-de-estilo	3281	ciencia-de-dados	3306	tempo-de-execucao
3257	persistencia	3282	controle-de-versao	3307	java10
3258	postgres	3283	depurador	3308	acentos
3259	variables	3284	angular2	3309	condicao
3260	browser	3285	angular-2	3310	consulta
3261	scroll	3286	sgbd	3311	asp.net-web-api2
3262	entity-framework-6.1	3287	vetores	3312	cross-join
3263	vb6	3288	bug	3313	transaction
3264	ordenacao	3289	keyboard		
3265	ordenacao	3290	reflection		

Referências Bibliográficas

Agirre, E., Banea, C., Cardie, C., Cer, D., Diab, M., Gonzalez-Agirre, A., Guo, W., Lopez-Gazpio, I., Maritxalar, M., Mihalcea, R., Rigau, G., Uria, L., and Wiebe, J. (2015). SemEval-2015 task 2: Semantic textual similarity, English, Spanish and pilot on interpretability. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 252–263, Denver, Colorado. Association for Computational Linguistics.

Ahasanuzzaman, M., Asaduzzaman, M., Roy, C. K., and Schneider, K. A. (2016). Mining duplicate questions of stack overflow. In *2016 IEEE/ACM 13th Working Conference on Mining Software Repositories (MSR)*, pages 402–412. IEEE.

Alipour, A., Hindle, A., and Stroulia, E. (2013). A contextual approach towards more accurate duplicate bug report detection. In *2013 10th Working Conference on Mining Software Repositories (MSR)*, pages 183–192. IEEE.

Alkadhi, R., Johanssen, J. O., Guzman, E., and Bruegge, B. (2017). React: An approach for capturing rationale in chat messages. In *2017 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, pages 175–180.

Alkadhi, R., Lata, T., Guzmany, E., and Bruegge, B. (2017). *Rationale in development chat messages: an exploratory study.* IEEE.

Alkadhi, R., Nonnenmacher, M., Guzman, E., and Bruegge, B. (2018). How do developers discuss rationale? In *2018 IEEE 25th International Conference on Software Analysis, Evolution and Reengineering (SANER)*, pages 357–369. IEEE.

Alkhazi, B., DiStasi, A., Aljedaani, W., Alrubaye, H., Ye, X., and Mkaouer, M. W. (2020). Learning to rank developers for bug report assignment. *Applied Soft Computing*, 95:106667.

Batool, I. and Khan, T. A. (2022). Software fault prediction using data mining, machine learning and deep learning techniques: A systematic literature review. *Computers and Electrical Engineering*, 100:107886.

Bavota, G. (2016). Mining unstructured data in software repositories: Current and future trends. In *2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, volume 5, pages 1–12. IEEE.

Bird, S. (2006). Nltk: the natural language toolkit. In *Proceedings of the COLING/ACL 2006 Interactive Presentation Sessions*, pages 69–72.

Borges, O., Lima, M., Couto, J., Gadelha, B., Conte, T., and Prikladnicki, R. (2022). Ml@se: What do we know about how machine learning impact software engineering practice? In *2022 17th Iberian Conference on Information Systems and Technologies (CISTI)*, pages 1–7. IEEE.

Bosu, A. and Carver, J. C. (2014). Impact of developer reputation on code review outcomes in oss projects: An empirical investigation. In *Proceedings of the 8th ACM/IEEE international symposium on empirical software engineering and measurement*, pages 1–10.

Brisson, S., Noei, E., and Lyons, K. (2020). We are family: analyzing communication in github software repositories and their forks. In *2020 IEEE 27th International Conference on Software Analysis, Evolution and Reengineering (SANER)*, pages 59–69. IEEE.

Buckland, M. and Gey, F. (1994). The relationship between recall and precision. *Journal of the American society for information science*, 45(1):12–19.

Burge, J. E. and Brown, D. C. (2008). Seurat: Integrated rationale management. In *Proceedings of the 30th International Conference on Software Engineering*, ICSE '08, pages 835–838, New York, NY, USA. ACM.

Calefato, F., Gerosa, M. A., Iaffaldano, G., Lanubile, F., and Steinmacher, I. (2022). Will you come back to contribute? investigating the inactivity of oss core developers in github. *Empirical Software Engineering*, 27(3):76.

Chen, T.-H., Thomas, S. W., and Hassan, A. E. (2016). A survey on the use of topic models when mining software repositories. *Empirical Software Engineering*, 21(5):1843–1919.

Chen, X., Li, X., Clark, J. G., and Dietrich, G. B. (2013). Knowledge sharing in open source software project teams: A transactive memory system perspective. *International Journal of Information Management*, 33(3):553–563.

Chi, M.-T., Lin, S.-S., Chen, S.-Y., Lin, C.-H., and Lee, T.-Y. (2015). Morphable word clouds for time-varying text data visualization. *IEEE transactions on visualization and computer graphics*, 21(12):1415–1426.

Chopra, A., Mo, M., Dodson, S., Beschastnikh, I., Fels, S. S., and Yoon, D. (2021). "@ alex, this fixes# 9": Analysis of referencing patterns in pull request discussions. *Proceedings of the ACM on Human-Computer Interaction*, 5(CSCW2):1–25.

Cohen, J. (1960). A coefficient of agreement for nominal scales. *Educational and psychological measurement*, 20(1):37–46.

Cooper, N., Bernal-Cárdenas, C., Chaparro, O., Moran, K., and Poshyvanyk, D. (2021). It takes two to tango: Combining visual and textual information for detecting duplicate video-based bug reports. In *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*, pages 957–969. IEEE.

Dagenais, B., Ossher, H., Bellamy, R. K., Robillard, M. P., and De Vries, J. P. (2010). Moving into a new software project landscape. In *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering-Volume 1*, pages 275–284.

De Vasconcelos, J. B., Kimble, C., Carreteiro, P., and Rocha, Á. (2017). The application of knowledge management to software evolution. *International Journal of Information Management*, 37(1):1499–1506.

Deka, B., Huang, Z., Franzen, C., Hibschman, J., Afergan, D., Li, Y., Nichols, J., and Kumar, R. (2017). Rico: A mobile app dataset for building data-driven design applications. In *Proceedings of the 30th annual ACM symposium on user interface software and technology*, pages 845–854.

Dias, E., Meirelles, P., Castor, F., Steinmacher, I., Wiese, I., and Pinto, G. (2021). What makes a great maintainer of open source projects? In *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*, pages 982–994. IEEE.

Face, H. (2021). sentence-transformers/all-mpnet-base-v2.

Fang, H., Klug, D., Lamba, H., Herbsleb, J., and Vasilescu, B. (2020). Need for tweet: How open source developers talk about their github work on twitter. In *Proceedings of the 17th International Conference on Mining Software Repositories*, pages 322–326.

Ferrari, A. and Esuli, A. (2019). An nlp approach for cross-domain ambiguity detection in requirements engineering. *Automated Software Engineering*, 26(3):559–598.

Francois, R., Nada, M., and Hassan, A. (2015). How to extract knowledge from professional e-mails. In *2015 11th International Conference on Signal-Image Technology & Internet-Based Systems (SITIS)*, pages 687–692. IEEE.

Gambhir, M. and Gupta, V. (2017). Recent automatic text summarization techniques: a survey. *Artificial Intelligence Review*, 47(1):1–66.

Gao, W., Wu, J., and Xu, G. (2022). Detecting duplicate questions in stack overflow via source code modeling. *International Journal of Software Engineering and Knowledge Engineering*, 32(02):227–255.

Giardino, C., Unterkalmsteiner, M., Paternoster, N., Gorschek, T., and Abrahamsson, P. (2014). What do we know about software development in startups? *IEEE Software*, 31(5):28–32.

GitHub, I. (2021a). Github discussions.

GitHub, I. (2021b). Managing categories for discussions in your repository.

GitHub, I. (2021c). Searching discussions.

Guizani, M., Zimmermann, T., Sarma, A., and Ford, D. (2022). Attracting and retaining oss contributors with a maintainer dashboard. *arXiv preprint arXiv:2202.07740*.

Guzzi, A., Bacchelli, A., Lanza, M., Pinzger, M., and Van Deursen, A. (2013). Communication in open source software development mailing lists. In *2013 10th Working Conference on Mining Software Repositories (MSR)*, pages 277–286. IEEE.

Haiduc, S., Aponte, J., and Marcus, A. (2010). Supporting program comprehension with source code summarization. In *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering-Volume 2*, pages 223–226. ACM.

Hata, H., Novielli, N., Baltes, S., Kula, R. G., and Treude, C. (2022). Github discussions: An exploratory study of early adoption. *Empirical Software Engineering*, 27(1):1–32.

Hata, H., Treude, C., Kula, R. G., and Ishio, T. (2019). 9.6 million links in source code comments: Purpose, evolution, and decay. In *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*, pages 1211–1221. IEEE.

Hevner, A. R. (2007). A three cycle view of design science research. *Scandinavian journal of information systems*, 19(2):4.

Isaacs, E., Walendowski, A., Whittaker, S., Schiano, D. J., and Kamm, C. (2002). The character, functions, and styles of instant messaging in the workplace. In *Proceedings of the 2002 ACM conference on Computer supported cooperative work*, pages 11–20. ACM.

Jain, R., Ghaisas, S., and Sureka, A. (2014). Sanayojan: a framework for traceability link recovery between use-cases in software requirement specification and

regulatory documents. In *Proceedings of the 3rd International Workshop on Realizing Artificial Intelligence Synergies in Software Engineering*, pages 12–18.

Jindal, S. G. and Kaur, A. (2020). Automatic keyword and sentence-based text summarization for software bug reports. *IEEE Access*, 8:65352–65370.

Khusro, S., Jabeen, F., and Khan, A. (2018). Tag clouds: Past, present and future. *Proceedings of the National Academy of Sciences, India Section A: Physical Sciences*, pages 1–13.

Kim, Y. (2018). Improving classifiers for semantic annotation of software requirements with elaborate syntatic structure. *International Journal of Advanced Science and Technology, ISSN 2005-4238 IJAST, Vol. 112, Nº. 44, 2009, pags. 123-136*, page 14.

Kleebaum, A., Johanssen, J. O., Paech, B., Alkadhi, R., and Bruegge, B. (2018). Decision knowledge triggers in continuous software engineering. In *Proceedings of the 4th International Workshop on Rapid Continuous Software Engineering*, RCoSE '18, pages 23–26, New York, NY, USA. ACM.

Komi-Sirvio, S., Mantyniemi, A., and Seppanen, V. (2002). Toward a practical solution for capturing knowledge for software projects. *IEEE software*, 19(3):60–62.

Kukkar, A., Mohana, R., Kumar, Y., Nayyar, A., Bilal, M., and Kwak, K.-S. (2020). Duplicate bug report detection and classification system based on deep learning technique. *IEEE Access*, 8:200749–200763.

Landis, J. R. and Koch, G. G. (1977). The measurement of observer agreement for categorical data. *biometrics*, pages 159–174.

Lazar, A., Ritchey, S., and Sharif, B. (2014). Generating duplicate bug datasets. In *Proceedings of the 11th working conference on mining software repositories*, pages 392–395.

Lee, I. and Shin, Y. J. (2020). Machine learning for enterprises: Applications, algorithm selection, and challenges. *Business Horizons*, 63(2):157–170.

Lerch, J. and Mezini, M. (2013). Finding duplicates of your yet unwritten bug report. In *2013 17th European conference on software maintenance and reengineering*, pages 69–78. IEEE.

Levy, M. and Hazzan, O. (2009). Knowledge management in practice: The case of agile software development. In *2009 ICSE Workshop on Cooperative and Human Aspects on Software Engineering*, pages 60–65. IEEE.

Li, L., Ren, Z., Li, X., Zou, W., and Jiang, H. (2018). How are issue units linked? empirical study on the linking behavior in github. In *2018 25th Asia-Pacific Software Engineering Conference (APSEC)*, pages 386–395. IEEE.

Li, Z., Yin, G., Yu, Y., Wang, T., and Wang, H. (2017). Detecting duplicate pull-requests in github. In *Proceedings of the 9th Asia-Pacific Symposium on Internetware*, pages 1–6.

Li, Z.-X., Yu, Y., Wang, T., Yin, G., Mao, X.-J., and Wang, H.-M. (2021). Detecting duplicate contributions in pull-based model combining textual and change similarities. *Journal of Computer Science and Technology*, 36:191–206.

Lima, C. and Soares, D. (2022). On the nature of duplicate pull-requests: An empirical study using association rules. In *Proceedings of the 16th Brazilian Symposium on Software Components, Architectures, and Reuse*, pages 68–75.

Lima, M., Ahmed, I., Conte, T., Nascimento, E., Oliveira, E., and Gadelha, B. (2019a). Land of lost knowledge: An initial investigation into projects lost knowledge. In *2019 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, pages 1–6. IEEE.

Lima, M., Fontão, A., Fernandes, D., Conte, T., and Gadelha, B. (2019b). How are my students going? a tool to analyse students' interactions on capstone courses. In *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE)*, volume 30, page 1611.

Lima, M., Oliveira, E., Conte, T., and Gadelha, B. (2020). Clouds are heavy! a storm of relevant project-related terms to support newcomers' onboarding. In

Proceedings of the 34th Brazilian Symposium on Software Engineering, SBES 2020, Natal, Brazil, October 19-23, 2020, pages 319–324.

Lima, M., Steinmacher, I., Ford, D., Liu, E., Vorreuter, G., Conte, T., and Gadelha, B. (2023). Looking for related discussions on github discussions. *PeerJ Computer Science*.

Lima, M., Valle, V., Costa, E., Lira, F., and Gadelha, B. (2019c). Software engineering repositories: Expanding the PROMISE database. In *Proceedings of the XXXIII Brazilian Symposium on Software Engineering, SBES 2019, Salvador, Brazil, September 23-27, 2019*, pages 427–436.

Liu, E. (2022). How five open source communities are using github discussions.

Liu, T.-Y. (2011). *Learning to rank for information retrieval*. Springer Science & Business Media.

Lu, J., Lin, C., Wang, W., Li, C., and Wang, H. (2013). String similarity measures and joins with synonyms. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, pages 373–384.

Manteuffel, C., Tofan, D., Koziolek, H., Goldschmidt, T., and Avgeriou, P. (2014). Industrial implementation of a documentation framework for architectural decisions. In *2014 IEEE/IFIP Conference on Software Architecture*, pages 225–234. IEEE.

McFee, B. and Lanckriet, G. R. (2010). Metric learning to rank. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 775–782.

Mishra, S. and Sharma, A. (2021). Crawling wikipedia pages to train word embeddings model for software engineering domain. In *14th Innovations in Software Engineering Conference (formerly known as India Software Engineering Conference)*, pages 1–5.

Mizobuchi, Y. and Takayama, K. (2017). Two improvements to detect duplicates in stack overflow. In *2017 IEEE 24th International Conference on Software Analysis, Evolution and Reengineering (SANER)*, pages 563–564. IEEE.

Natali, A. C. C. and de Almeida Falbo, R. (2002). Knowledge management in software engineering environments. In *Anais do XVI Simpósio Brasileiro de Engenharia de Software*, pages 238–253. SBC.

Naur, P. (1985). Programming as theory building. *Microprocessing and microprogramming*, 15(5):253–261.

Niyogi, S. (2020). New from satellite 2020: Github discussions, codespaces, securing code in private repositories, and more.

Pei, J., Wu, Y., Qin, Z., Cong, Y., and Guan, J. (2021). Attention-based model for predicting question relatedness on stack overflow. In *2021 IEEE/ACM 18th International Conference on Mining Software Repositories (MSR)*, pages 97–107. IEEE.

Pérez-Soler, S., Guerra, E., and de Lara, J. (2018). Collaborative modeling and group decision making using chatbots in social networks. *IEEE Software*, 35(6):48–54.

Poche, E., Jha, N., Williams, G., Staten, J., Vesper, M., and Mahmoud, A. (2017). Analyzing user comments on youtube coding tutorial videos. In *2017 IEEE/ACM 25th International Conference on Program Comprehension (ICPC)*, pages 196–206. IEEE.

Polyzotis, N., Roy, S., Whang, S. E., and Zinkevich, M. (2017). Data management challenges in production machine learning. In *Proceedings of the 2017 ACM International Conference on Management of Data*, pages 1723–1726.

Ponzanelli, L., Bavota, G., Mocci, A., Oliveto, R., Di Penta, M., Haiduc, S., Russo, B., and Lanza, M. (2017). Automatic identification and classification of software development video tutorial fragments. *IEEE Transactions on Software Engineering*, 45(5):464–488.

Rastkar, S., Murphy, G. C., and Murray, G. (2014). Automatic summarization of bug reports. *IEEE Transactions on Software Engineering*, 40(4):366–380.

Reimers, N. (2021). Sentencetransformers documentation.

Reimers, N. and Gurevych, I. (2019). Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*.

Ren, L., Zhou, S., Kastner, C., and Wasowski, A. (2019). Identifying redundancies in fork-based development. In *2019 IEEE 26th International Conference on Software Analysis, Evolution and Reengineering (SANER)*, pages 230–241. IEEE.

Rigby, P. C. and Hassan, A. E. (2007). What can oss mailing lists tell us? a preliminary psychometric text analysis of the apache developer mailing list. In *Fourth International Workshop on Mining Software Repositories (MSR'07: ICSE Workshops 2007)*, pages 23–23. IEEE.

Robertson, S. (2004). Understanding inverse document frequency: on theoretical arguments for idf. *Journal of documentation*, 60(5):503–520.

Runeson, P., Alexandersson, M., and Nyholm, O. (2007). Detection of duplicate defect reports using natural language processing. In *29th International Conference on Software Engineering (ICSE'07)*, pages 499–510. IEEE.

Salman, I., Misirli, A. T., and Juristo, N. (2015). Are students representatives of professionals in software engineering experiments? In *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, volume 1, pages 666–676. IEEE.

Schelter, S., Biessmann, F., Januschowski, T., Salinas, D., Seufert, S., and Szarvas, G. (2018). On challenges in machine learning model management.

Schwaber, K. (1997). Scrum development process. In *Business object design and implementation*, pages 117–134. Springer.

Sharma, A. and Sharma, S. (2015). Bug report triaging using textual categorical and contextual features using latent dirichlet allocation. *International Journal for Innovative Research in Science and Technology (IJIRST)*, 1(9):85–96.

Silva, R. F., Paixão, K., and de Almeida Maia, M. (2018). Duplicate question detection in stack overflow: A reproducibility study. In *2018 IEEE 25th inter-*

national conference on software analysis, evolution and reengineering (SANER), pages 572–581. IEEE.

Sirres, R., Bissyandé, T. F., Kim, D., Lo, D., Klein, J., Kim, K., and Traon, Y. L. (2018). Augmenting and structuring user queries to support efficient free-form code search. *Empirical Software Engineering*, 23(5):2622–2654.

Soria, A. M. and van der Hoek, A. (2019). Collecting design knowledge through voice notes. In *Proceedings of the 12th International Workshop on Cooperative and Human Aspects of Software Engineering*, pages 33–36. IEEE Press.

Srinivas, C., Radhakrishna, V., and Rao, C. G. (2015). Clustering software project components for strategic decisions and building reuse libraries. In *Proceedings of the The International Conference on Engineering & MIS 2015*, pages 1–5.

Steinmacher, I., Conte, T., Gerosa, M. A., and Redmiles, D. (2015). Social barriers faced by newcomers placing their first contribution in open source software projects. In *Proceedings of the 18th ACM conference on Computer supported cooperative work & social computing*, pages 1379–1392.

Storey, M.-A., Engstrom, E., Höst, M., Runeson, P., and Bjarnason, E. (2017). Using a visual abstract as a lens for communicating and promoting design science research in software engineering. In *2017 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, pages 181–186. IEEE.

Storey, M.-A., Singer, L., Cleary, B., Figueira Filho, F., and Zagalsky, A. (2014). The (r) evolution of social media in software engineering. In *Future of Software Engineering Proceedings*, pages 100–116.

Storey, M.-A., Zagalsky, A., Figueira Filho, F., Singer, L., and German, D. M. (2016). How social and communication channels shape and challenge a participatory culture in software development. *IEEE Transactions on Software Engineering*, 43(2):185–204.

Tan, P.-N. et al. (2006). *Introduction to data mining*. Pearson Education India.

Tan, X. and Zhou, M. (2020). Scaling open source software communities: Challenges and practices of decentralization. *IEEE Software*, 39(1):70–75.

Tantisuwankul, J., Nugroho, Y. S., Kula, R. G., Hata, H., Rungsawang, A., Leelaprute, P., and Matsumoto, K. (2019). A topological analysis of communication channels for knowledge sharing in contemporary github projects. *Journal of Systems and Software*, 158:110416.

Trinkenreich, B., Guizani, M., Wiese, I. S., Conte, T., Gerosa, M., Sarma, A., and Steinmacher, I. (2021). Pots of gold at the end of the rainbow: What is success for open source contributors. *IEEE Transactions on Software Engineering*.

Tukey, J. W. et al. (1977). *Exploratory data analysis*, volume 2. Reading, Mass.

Vasilescu, B., Serebrenik, A., Devanbu, P., and Filkov, V. (2014). How social Q&A sites are changing knowledge sharing in open source software communities. In *Proceedings of the 17th ACM conference on Computer supported cooperative work & social computing*, pages 342–354.

Viviani, G., Janik-Jones, C., Famelis, M., Xia, X., and Murphy, G. C. (2018). What design topics do developers discuss? In *Proceedings of the 26th Conference on Program Comprehension*, pages 328–331. ACM.

Wang, D., Xiao, T., Thongtanunam, P., Kula, R. G., and Matsumoto, K. (2021). Understanding shared links and their intentions to meet information needs in modern code review. *Empirical Software Engineering*, 26(5):1–32.

Wang, L., Zhang, L., and Jiang, J. (2020). Duplicate question detection with deep learning in stack overflow. *IEEE Access*, 8:25964–25975.

Wang, Q., Xu, B., Xia, X., Wang, T., and Li, S. (2019). Duplicate pull-request detection: When time matters. In *Proceedings of the 11th Asia-Pacific symposium on internetware*, pages 1–10.

Wirfs-Brock, R. J. and Johnson, R. E. (1990). Surveying current research in object-oriented design. *Communications of the ACM*, 33(9):104–124.

Yazdaninia, M., Lo, D., and Sami, A. (2021). Characterization and prediction of questions without accepted answers on stack overflow. In *2021 IEEE/ACM 29th International Conference on Program Comprehension (ICPC)*, pages 59–70. IEEE.

Yu, Y., Li, Z., Yin, G., Wang, T., and Wang, H. (2018). A dataset of duplicate pull-requests in github. In *Proceedings of the 15th International Conference on Mining Software Repositories*, pages 22–25.

Zaki, M. J. and Meira, W. (2014). *Data mining and analysis: fundamental concepts and algorithms*. Cambridge University Press.

Zampetti, F., Ponzanelli, L., Bavota, G., Mocci, A., Di Penta, M., and Lanza, M. (2017). How developers document pull requests with external references. In *2017 IEEE/ACM 25th International Conference on Program Comprehension (ICPC)*, pages 23–33. IEEE.

Zhang, W. E., Sheng, Q. Z., Lau, J. H., and Abebe, E. (2017). Detecting duplicate posts in programming qa communities via latent semantics and association rules. In *Proceedings of the 26th International Conference on World Wide Web*, pages 1221–1229.

Zhang, W. E., Sheng, Q. Z., Lau, J. H., Abebe, E., and Ruan, W. (2018). Duplicate detection in programming question answering communities. *ACM Transactions on Internet Technology (TOIT)*, 18(3):1–21.

Zhang, Y., Lo, D., Xia, X., and Sun, J.-L. (2015). Multi-factor duplicate question detection in stack overflow. *Journal of Computer Science and Technology*, 30(5):981–997.

Zhang, Y., Wu, Y., Wang, T., and Wang, H. (2020). ilinker: a novel approach for issue knowledge acquisition in github projects. *World Wide Web*, 23(3):1589–1619.

Zheng, L., Albano, C. M., Vora, N. M., Mai, F., and Nickerson, J. V. (2019). The roles bots play in wikipedia. *Proceedings of the ACM on Human-Computer Interaction*, 3(CSCW):1–20.

Zhou, L., Pan, S., Wang, J., and Vasilakos, A. V. (2017). Machine learning on big data: Opportunities and challenges. *Neurocomputing*, 237:350–361.