UNIVERSIDADE FEDERAL DO MARANHÃO

Programa de Pós-Graduação em Ciência da Computação

Bruno Carvalho da Silva

ReqCluster4IoT: Um método de Agrupamento de Requisitos para aplicações de IoT

São Luís - MA

2025

Bruno Carvalho da Silva

ReqCluster4IoT: Um método de Agrupamento de Requisitos para aplicações de IoT

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da UFMA, como parte dos requisitos necessários para a obtenção do Título de Mestre em Ciência da Computação.

Programa de Pós-Graduação em Ciência da Computação

Universidade Federal do Maranhão

Orientador: Prof. Dr. Davi Viana dos Santos

Coorientador: Prof. Dr. Rodrigo Pereira dos Santos

São Luís - MA

2025

Ficha gerada por meio do SIGAA/Biblioteca com dados fornecidos pelo(a) autor(a).
Diretoria Integrada de Bibliotecas/UFMA

Silva, Bruno Carvalho da.
ReqCluster4IoT: Um método de Agrupamento de Requisitos para aplicações de IoT / Bruno Carvalho da Silva. - 2025.
110 f.

Coorientador(a) 1: Rodrigo Pereira dos Santos.
Orientador(a): Davi Viana dos Santos.
Dissertação (Mestrado) - Programa de Pós-graduação em Ciência da Computação/ccet, Universidade Federal do Maranhão, São Luís, 2025.

1. Engenharia de Requisitos. 2. Internet das Coisas. 3. Processamento de Linguagem Natural. 4. Bases de Requisitos. 5. Agrupamento de Requisitos. I. Santos, Davi Viana dos. II. Santos, Rodrigo Pereira dos. III. Título.

Bruno Carvalho da Silva

ReqCluster4IoT: Um método de Agrupamento de Requisitos para aplicações de IoT

Avaliação de dissertação em 09 de abril, 2025, em São Luís - MA:

Prof. Dr. Davi Viana dos Santos
Orientador
Universidade Federal do Maranhão

Prof. Dr. Rodrigo Pereira dos Santos
Coorientador
Universidade Federal do Estado do Rio de Janeiro

Prof. Dr. Geraldo Braz Junior
Examinador Interno
Universidade Federal do Maranhão

Prof. Dr. Paulo Robson Campelo Malcher
Examinador Externo
Universidade Federal Rural da Amazônia

Profa. Dra. Tayana Uchôa Conte
Examinadora Externa
Universidade Federal do Amazonas

São Luís - MA
2025

À minha família e amigos, que sempre estiveram ao meu lado nesta caminhada, oferecendo apoio, carinho e incentivo.

Agradecimentos

Agradeço aos meus pais, Raimundo Antero da Silva Neto e Iranilde Braga Carvalho, por todo apoio, cuidado, paciência e carinho dado em todo esse tempo. E aos meus irmãos, Rafael e Taffarel, por estarem sempre ao meu lado, para me apoiar no possível. Agradeço também a minha avó, Maria das Graças, por todo amor, carinho e cada ligação.

Sou bastante agradecido ao meu orientador, Davi Viana, por todas as orientações, oportunidades e ensinamentos dados sobre como ser um bom pesquisador, profissional e pessoa. E a Rodrigo Santos, pela coorientação e reuniões mensais, sempre cheias de ideias e orientações, que muito apoiaram este trabalho.

À Izadora de Paula, meu amor e namorada, que esteve ao meu lado em cada etapa desta jornada no mestrado. Sua paciência, carinho e apoio foram fundamentais nos momentos desafiadores, sempre me ouvindo, me reconfortando, me incentivando e, com seu jeitinho único, me lembrando da importância de relaxar. Obrigado por ser essa companheira incrível, que celebra comigo as conquistas e me dá forças nas dificuldades. Sou imensamente grato por ter você ao meu lado.

À minha segunda família, meus amigos de longa data – Lethycia, Bruno Marllon, Marlon, Marcos Vinícius, Jamil, Matheus, Erik, Lucas Feitosa e Jhones –, por toda amizade, apoio incondicional e por compartilharem comigo tantas memórias inesquecíveis. Cada conversa, conselho e momento juntos foram essenciais nessa caminhada. E à Fernanda Assunção, minha amiga desde o início desta minha trajetória na UFMA, sempre surtando junto e proporcionando diversos momentos de descontração e felicidade.

Ao Laboratório de Sistemas Distribuídos Inteligentes (LSDi) pelo espaço e oportunidades oferecidas. Aos professores Francisco José da Silva e Silva e Luciano Reis Coutinho, pelos ensinamentos passados. E aos colegas, Rodrigo Nascimento, Lucas Abreu, Igor Reis, Claudio Aroucha, Luis Laurindo, André Barreto e, especialmente, Bruno Roberto, meu amigo que muito me incentiva, me inspira e me aconselha sobre diversos aspectos da vida.

Por último, mas com igual importância, expresso minha profunda gratidão a todos que, de alguma forma, contribuíram para minha formação acadêmica.

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) - Código de Financiamento 001. Além de agradecermos ao PROCAD-Amazônia da CAPES (88887.200532/2018-00). Agradecemos a FAPEMA (BEPP-03906/23) e FAPEMA/EMAP(APP-09405/22). Por fim, agradecemos ao INCT de Internet do Futuro para Cidades Inteligentes (CNPq proc. 465446/2014-0, FAPESP proc. 14/50937-1, and FAPESP proc. 15/24485-9).

"Faça ou não faça. Tentativa não há."

(Mestre Yoda)

"Se você está comprometido o suficiente, você pode fazer qualquer coisa."

(Saul Goodman)

Resumo

A Engenharia de Requisitos é uma importante etapa do ciclo de vida de desenvolvimento de software, visando que o produto final atenda às necessidades dos *stakeholders*. Em projetos de software, os requisitos são frequentemente escritos em linguagem natural não estruturada e organizados em documentos de especificação de requisitos. Tal característica oferece praticidade no momento da criação do documento; contudo, torna a análise manual difícil e demorada, sobretudo em projetos com grande número de requisitos. Contextos contemporâneos de software, como Internet das Coisas (do inglês, *Internet of Things* ou IoT), tornam necessário que o processo de desenvolvimento de software seja repensado continuamente. A análise de requisitos em IoT é desafiadora devido à alta diversidade de elementos presentes nesse domínio complexo. Nestas circunstâncias, uma abordagem que pode dar suporte para a análise destes requisitos é a utilização de algoritmos de agrupamento, para organizá-los em grupos de contextos similares. Tal agrupamento pode ajudar na compreensão do sistema a ser projetado. Foi proposto então o ReqCluster4IoT, um método de agrupamento de requisitos funcionais para aplicações de IoT. O ReqCluster4IoT está fundamentado em uma análise exploratória da literatura para identificar características de IoT e também em experimentos em classificação de requisitos e em similaridade semântica de textos. Para prover melhores índices na classificação e na computação de similaridade semântica, foram desenvolvidos dois conjuntos de dados, a Promise+ e o ReqFuncSimDataset. Os experimentos demonstraram a superioridade de modelos baseados em Transformers em detrimento as abordagens tradicionais. O método proposto foi avaliado quanto à sua aceitação, considerando sua utilidade. Para isso, foi conduzida uma pesquisa de opinião e um grupo focal. A pesquisa opinião visava obter a aceitação de 10 desenvolvedores de software para IoT. Por sua vez, o grupo focal foi utilizado para obter mais informações sobre os resultados providos pelo método. Os resultados mostraram que o método proposto consegue capturar com eficácia as relações entre requisitos para agrupá-los corretamente. Os resultados dos experimentos demonstraram ainda que o ReqCluster4IoT pode auxiliar os profissionais no agrupamento de requisitos de software para IoT, bem como a identificação de característica de IoT no grupo pode dar mais informações para a análise destes requisitos. O método obteve também altos índices de aceitação quanto à sua utilidade, ao seu potencial de redução de esforço e intenção de uso.

Palavras-chave: Engenharia de Requisitos, Internet das Coisas, Processamento de Linguagem Natural, Bases de Requisitos, Agrupamento de Requisitos.

Abstract

Requirements Engineering is a critical phase in the software development lifecycle, ensuring the final product meets stakeholder needs. In software projects, teams often write requirements in unstructured natural language and organize them into requirement specification documents. While this approach simplifies document creation, it makes manual analysis difficult and time-consuming, especially in projects with numerous requirements. Modern software contexts, such as the Internet of Things (IoT), demand continuous reevaluation of software development processes. Requirement analysis in IoT poses challenges due to the high diversity of elements in this complex domain. In such scenarios, clustering algorithms can support requirement analysis by grouping them into similar contexts, improving system understanding. To address this, we propose ReqCluster4IoT, a functional requirement clustering method for IoT applications. The approach builds on an exploratory literature analysis to identify IoT characteristics, along with experiments on requirement classification and text semantic similarity. To enhance classification and semantic similarity computation, we developed two datasets: Promise+ and ReqFuncSimDataset. Experiments showed that Transformer-based models outperform traditional approaches. We evaluated the method's acceptance and usefulness through a survey and a focus group. The survey gathered feedback from 10 IoT software developers, while the focus group provided deeper insights into the method's results. Findings confirmed that ReqCluster4IoT effectively captures relationships between requirements for accurate grouping. Additionally, the method helps professionals cluster IoT software requirements and identify IoT-specific features within groups, improving requirement analysis. The method achieved high acceptance rates in terms of usefulness, effort reduction potential, and intention to use.

Keywords: Requirements Engineering, Internet of Things, Natural Language Processing, Requirements Bases, Requirements Clustering.

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

AB	*Ada Boost*
AM	*Aprendizado de Máquina*
BoW	*Bag-of-word*
DT	*Decision Tree*
GB	*Gradient Boosting*
IA	*Inteligencia Artificial*
IoT	*Internet of Things*
KNN	*K-Nearest Neighboor*
LLM	*Large Language Model*
LR	*Logistic Regression*
LSTM	*Long Short Term Memor*
MNB	*Multinomial Naive Bayes*
MVS	*Máquina de Vetor de Suporte*
PA	*Passive Agressive*
PLM	*Pre-treined Language Model*
RF	*Random Forest*
RNN	*Recurrent Neural Network*
SGD	*Stochastic Gradient Descent*
SRS	*Software Requirements Specification*
TAM	*Technology Acceptance ModeL*
TF-IDF	*Term Frequency-Inverse Document Frequency*
UML	*Unified Modeling Language*

Sumário

1 Introdução

No ciclo de desenvolvimento de software, a Engenharia de Requisitos desempenha um importante papel, sendo considerada um dos processos mais importantes. Este processo é comumente dividido nas etapas de: elicitação dos requisitos, análise dos requisitos, documentação dos requisitos, verificação dos requisitos e validação dos requisitos (ISO/IEC/IEEE 29148, 2018). Na etapa de análise dos requisitos, os requisitos são classificados entre funcionais, que descrevem as funções ou tarefas do software, bem como seus componentes, e requisitos não funcionais que especificam as limitações do software e ajudam a garantir que o software cumpra com as necessidades dos usuários. É importante que os requisitos sejam corretamente especificados, pois caso um erro de requisito se propague para outras fases do desenvolvimento e chegue à etapa de produção, o custo de correção pode ser muito maior comparado ao que seria se fosse corrigido na fase de requisitos (ARYA; NIGAM; NIGAM, 2012).

1.1 Contextualização

Novos contextos de software, como o de Internet das Coisas (do inglês, *Internet of Things* ou IoT), tornam necessário que a Engenharia de Software repense continuamente o desenvolvimento de sistemas para se adequar aos novos paradigmas (FAHMIDEH et al., 2022). Como destacado por Giray, Tekinerdogan e Tüzün (2018), é importante que o software para IoT seja desenvolvido de forma sistemática, a fim de alcançar um sistema adequado no que diz respeito a requisitos funcionais e não funcionais. Ressalta-se ainda a não existência de um conjunto de conceitos bem definidos na Engenharia de Requisitos para aplicações de contextos contemporâneos, como IoT (SILVA et al., 2021).

A IoT perpassa os mais diversos aspectos do cotidiano, envolvendo áreas como casas e prédios inteligentes, agricultura, redes elétricas inteligentes (do inglês, *smart grids*), hospitais e espaços públicos (MURALIDHARAN; ROY; SAXENA, 2018). Estas aplicações objetivam aprimorar a qualidade de vida das pessoas, oferecendo soluções tecnológicas que aumentam a eficiência, o conforto e a segurança em diferentes contextos.

O avanço da IoT, contudo, enfrenta vários desafios técnicos e operacionais: (1) a disponibilidade dos sistemas é dependente dos protocolos de comunicação empregados, exigindo que eles sejam eficientes e confiáveis para suportar as demandas de tempo real; (2) a confiabilidade dos dados é um aspecto crítico, pois decisões automatizadas baseadas em dados imprecisos ou inconsistentes podem comprometer a segurança e a eficácia das soluções; (3) a escalabilidade das soluções é essencial, especialmente em aplicações que demandam a integração de muitos dispositivos. Esses sistemas precisam ser projetados

para crescer sem diminuir o desempenho ou funcionalidade; e (4) a interoperabilidade entre dispositivos heterogêneos no ecossistema da IoT, considerando a ampla variedade de fabricantes e tecnologias (KHANNA; KAUR, 2020).

1.2 Problema e Justificativa

Em projetos de software, os requisitos ficam contidos em um documento chamado Especificação de Requisitos de Software (*Software Requirements Specification* ou SRS). A SRS é geralmente escrita em linguagem natural, conforme Franch et al. (2023), facilitando sua produção, contudo, agrega os problemas inerentes desta linguagem, tais quais ambiguidades e linguagem específica do domínio, por exemplo. Um número grande de requisitos dificulta o gerenciamento manual dos requisitos e prejudica a compreensão sobre a expectativa e escopo do sistema projetado (KOCHBATI et al., 2021). A análise destes requisitos, que inclui a classificação dos requisitos e identificação de como os requisitos se relacionam entre si, é geralmente realizada manualmente por meio de um processo demorado (MISRA et al., 2022).

A classificação manual de requisitos é uma tarefa desafiadora, demorada e suscetível a erros, especialmente em projetos de software que possuem uma grande quantidade de requisitos. Dessa forma, a necessidade de métodos automáticos para classificar requisitos se torna evidente, promovendo pesquisas que explorem abordagens baseadas em regras e aprendizado de máquina (ZHAO et al., 2021). Essa busca por automação é ainda mais relevante em domínios complexos, como o desenvolvimento de software para IoT, que apresenta desafios adicionais para a Engenharia de Requisitos, incluindo a volatilidade dos requisitos e a heterogeneidade de requisitos e dispositivos (SOUZA et al., 2025).

A complexidade do domínio de IoT advém da característica distribuída e heterogênea destes sistemas, que integram diversos dispositivos de hardware e software conectados com capacidades computacionais limitadas. Conforme Silva, Gonçalves e Rocha (2019), estes fatores tornam as etapas de especificação e análise de requisitos complexas, principalmente na identificação e descrição dos elementos de hardware e software, bem como à gestão das interações entre esses elementos. Outro problema relatado na literatura para o domínio de IoT foi em relação às decisões de *design*, estilos e arquitetura para estes softwares (MOTTA; OLIVEIRA; TRAVASSOS, 2019).

Em Fahmideh et al. (2022), um questionário online foi respondido por 127 profissionais de software especializados em IoT, e 40 desses participantes destacaram a elicitação e validação de requisitos como uma tarefa importante. Os participantes relataram algumas razões para esta dificuldade, como requisitos não claros, que levam a mudanças de requisitos ao longo do projeto. Além disso, a gestão e análise de requisitos para IoT apresentam dificuldades, uma vez que é necessária experiência em diversas áreas do contexto, como

eletrônica e comunicação. No contexto de IoT, onde a interoperabilidade e a escalabilidade são fatores críticos, tratar o software modularmente pode ser essencial para lidar com a complexidade crescente e atender aos requisitos de sistemas distribuídos e interconectados.

Nesse contexto, técnicas de agrupamento de requisitos surgem como uma abordagem promissora para enfrentar tais desafios. Algoritmos de agrupamento têm sido amplamente aplicados em diversas áreas da engenharia de software, conforme Shtern e Tzerpos (2012), oferecendo suporte para atividades como análise de reflexão, evolução de software e recuperação de informações. No domínio da Engenharia de Requisitos, o uso de algoritmos de agrupamento tem se mostrado particularmente útil para organizar e estruturar documentos de especificação, uma vez que estes documentos são escritos majoritariamente em linguagem natural, logo, algoritmos de agrupamento, especialmente de textos, são de grande proveito (CASAMAYOR; GODOY; CAMPO, 2012a).

Uma das aplicações promissoras em agrupamento de requisitos é a possibilidade de tratar cada grupo de requisitos funcionais como subprojetos ou componentes, facilitando a divisão do projeto inicial em partes menores e mais gerenciáveis (KUHN; DUCASSE; GîRBA, 2007). O agrupamento é realizado somente em requisitos funcionais, pois os não funcionais ditam características e restrições gerais aos sistemas, podendo, portanto, alcançar um ou mais grupos. Essa abordagem pode trazer diversos benefícios para o processo de desenvolvimento. Primeiro, ao identificar grupos de requisitos relacionados, é possível alocar subprojetos a times de desenvolvimento especializados. Segundo, o agrupamento pode incentivar a reutilização de componentes previamente desenvolvidos, aumentando a eficiência e reduzindo custos. Terceiro, a identificação clara das responsabilidades de cada subprojeto auxilia na definição de limites entre os módulos do sistema, promovendo maior modularidade (SALMAN et al., 2018). Agrupamento de requisitos pode ainda gerar uma visão global de alto nível para interdependência de requisitos (MISRA et al., 2022).

Por exemplo, um grupo de requisitos voltado para dispositivos IoT pode ser tratado por especialistas nessa tecnologia, enquanto outro grupo relacionado à interação entre os dispositivos IoT pode ser atribuído para profissionais com experiência em conectividade e protocolos de comunicação. Além disso, o agrupamento de requisitos pode ter impacto nas decisões arquiteturais. Ao identificar funcionalidades ou componentes coesos, permite-se estruturar o software de maneira mais modular, com interfaces bem definidas entre os módulos (AL-OTAIBY; ALSHERIF; BOND, 2005; GALSTER; EBERLEIN; JIANG, 2013).

Isso não apenas facilita a manutenção e evolução do sistema, mas também apoia decisões como a escolha de tecnologias, *frameworks* e padrões de design que melhor atendem às necessidades específicas de cada módulo. Portanto, um método foi proposto ao identificar que existe uma lacuna em métodos de agrupamento de requisitos que contemplem o domínio de IoT de maneira específica.

1.3 Objetivos

1.3.1 Objetivo Geral

O objetivo desta dissertação é apoiar analistas de requisitos e desenvolvedores de aplicações de IoT com um método de agrupamento de requisitos funcionais de software.

1.3.2 Objetivos Específicos

- Realizar a expansão de uma base de requisitos de software para classificação, incorporando novas fontes e técnicas de pré-processamento de texto.
- Investigar a eficácia de modelos baseados em *transformers* (como BERT e RoBERTa) para computação de similaridade semântica entre requisitos de software.
- Propor um método de agrupamento de requisitos funcionais baseado em similaridade semântica.
- Avaliar o método proposto por meio de métricas de qualidade de agrupamento (como *Silhouette Score*) e utilizando pesquisa de opinião e grupo focal.

1.4 Metodologia

A metodologia é dividida em três etapas principais: (1) Análise de trabalhos relacionados, (2) Proposta de método e (3) Avaliações experimentais.

- **Análise de trabalhos relacionados**: esta etapa foi realizada para verificar na literatura quais abordagens constam para agrupamento de requisitos. A análise da literatura foi utilizada para decidir qual abordagem de agrupamento seria utilizada e quantos e quais módulos o método precisaria. A análise dos trabalhos está descrita no Capítulo 3.
- **Proposta de método**: uma vez analisada a literatura de trabalhos de agrupamento, foi montado um *pipeline* com quatro módulos: extração de requisitos funcionais, computação da similaridade semântica, agrupamento de requisitos e identificação de características de IoT. Este processo e seus resultados são descritos no Capítulo 4.
- **Avaliações Experimentais**: o método proposto foi avaliado utilizando dois estudos experimentais. O primeiro foi uma pesquisa de opinião com desenvolvedores de aplicações de IoT para avaliar a percepção dos desenvolvedores sobre os resultados do método. Já o segundo foi um grupo focal com engenheiros de requisitos para obter informações mais detalhadas sobre os resultados do método. Os estudos de avaliação são detalhados no Capítulo 5.

1.5 Organização da Dissertação

Esta dissertação contribui para a pesquisa e prática em Engenharia de Software, especialmente para a Engenharia de Requisitos. Um método de agrupamento de requisitos funcionais para IoT, denominado ReqCluster4IoT, é descrito. Também são descritos os estudos experimentais que serviram de avaliação para o método. Esta dissertação está organizada em 6 capítulos, uma breve descrição do conteúdo de cada capítulo é apresentada a seguir.

- Capítulo 2 de Fundamentação teórica: são apresentados os conceitos de Engenharia de Requisitos, IoT, algoritmos de classificação, similaridade semântica e algoritmos de agrupamento;
- Capítulo 3 de Trabalhos relacionados: são apresentados trabalhos da literatura que propõem métodos de agrupamento e há uma comparação entre estes trabalhos e o proposto nesta dissertação;
- Capítulo 4 do ReqCluster4IoT: é apresentado o método desenvolvido, incluindo o processo de criação e experimentos realizados em cada uma das etapas;
- Capítulo 5 de Avaliações experimentais: são detalhados os estudos experimentais efetuados, incluindo o contexto e os requisitos. Também é detalhado o perfil dos participantes, o grupo focal com engenheiros de requisitos e o formulário online com desenvolvedores de IoT;
- Capítulo 6 de Conclusão e trabalhos futuros: apresenta as conclusões desta dissertação, as limitações e desafios da pesquisa e trabalhos futuros.

2 Fundamentação Teórica

Neste capítulo, são apresentados os conceitos de Engenharia de Requisitos, IoT e suas características, além de Algoritmos de Classificação, Similaridade Semântica e Algoritmos de Agrupamento. Na Seção 2.1 são trabalhados os conceitos principais da etapa de Engenharia de Requisitos; Na Seção 2.2 são apresentadas definições de IoT. A Seção 2.3 apresenta algoritmos de classificação de aprendizado de máquina. Modelos de Linguagem pré-treinados são apresentados na Seção 2.4. A Seção 2.5 apresenta os algoritmos de agrupamento utilizados.

2.1 Engenharia de Requisitos

Engenharia de Requisitos é a área de engenharia de sistemas que lida com o processo de desenvolvimento e verificação de requisitos do sistema (HERSMAN; FOWLER, 2010). Esta área trata sobre o que as partes interessadas querem de um sistema e como entender o que suas necessidades significam em termos de projeto (SUTCLIFFE; GULLIKSEN, 2012). A engenharia de requisitos provê também um meio sistematizado para o adequado entendimento do domínio e contexto do sistema sob desenvolvimento (SIAKAS et al., 2024). Conforme IEEE Computer Society Staff (2024), os processos de engenharia de requisitos podem ser divididos em desenvolvimento e gerência de requisitos. No desenvolvimento, as atividades são: elicitação, análise, especificação e validação. Já a gerência trata da depuração, controle de mudanças e correspondência de escopo (IEEE Computer Society Staff, 2024). Mais detalhes nos itens citados abaixo.

- **Elicitação**: também chamada de captura de requisitos, versa sobre a obtenção de candidatos a requisitos. Nesta etapa, um dos problemas a serem mitigados é de requisitos incompletos, o que pode levar a produção de software incompatível com as necessidades do cliente. Existem diversas formas para realizar a elicitação, como: entrevistas, grupos focais, pesquisas de opinião, prototipação e mapeamento de histórias de usuários (do inglês, *user storys*);
- **Análise:** esta atividade ajuda os desenvolvedores no entendimento e na aplicação dos requisitos elicitados, tanto individualmente quanto em seu conjunto. A análise visa garantir que os requisitos sejam: (1) completos, se adequando aos limites da aplicação; (2) concisos, sem conteúdo estranho aos requisitos; (3) sem conflitos com outros requisitos; (4) sem conflitos com documentos do projeto; e, por fim; (5) viáveis, a solução encontrada deve estar nas restrições de custo e de equipe;

- **Especificação**: versa sobre o registro e comunicação dos requisitos. Esta atividade é influenciada por fatores como familiaridade do engenheiro de requisitos com o negócio, grau de risco de requisitos incompletos e distribuição geográfica dos membros do projeto. As formas de especificação podem ser: linguagem natural estruturada ou não estruturada e baseada em modelos, como diagramas UML;

- **Validação**: trata de aferir se os requisitos documentados representam as verdadeiras necessidades das partes interessadas no produto. Métodos utilizados são: revisão de requisitos, simulação, execução e prototipação;

- **Depuração**: é uma atividade de gerência de requisitos que visa encontrar o menor conjunto de requisitos que atenderão às necessidades das partes interessadas. Fazer isso reduz o tamanho e complexidade da solução, minimizando o esforço e custo para entregá-la;

- **Controle de Mudanças**: é a atividade central da gerência de requisitos. Todas as partes interessadas devem entender e aceitar a mudança, acordando os impactos destas mudanças no cronograma, recursos e no escopo do projeto;

- **Correspondência de escopo**: envolve garantir que o escopo dos requisitos não exceda as restrições de custo, cronograma ou equipe. Sempre que possível, a correspondência de escopo deve ser quantitativa e não qualitativa.

2.2 Internet das Coisas

IoT é um paradigma computacional no qual objetos cotidianos, como relógios e roupas, podem ser equipados com identificação, sensitividade, atuação, conexão e processamento para permitir que eles se comuniquem entre si ou com outros dispositivos e serviços com a utilização de Internet (WHITMORE; AGARWAL; XU, 2014).

Tendo surgido há pouco mais de 20 anos, este campo tecnológico tem se tornado influente nos mais diversos campos de aplicação, tais quais: mobilidade, casas/prédios inteligentes, agricultura, dentre outros. Uma característica notável deste domínio é a sua diversidade, representada na Figura 1 por três grandes campos: Dispositivos IoT, Tecnologias e Aplicações (KHANNA; KAUR, 2020). Dispositivos de IoT tratam-se dos dispositivos que efetivamente coletarão dados e agir sobre o ambiente, como microcontroladores e smartphones. As tecnologias são os elementos fundamentais que dão infraestrutura para as aplicações, contendo desde os protocolos para movimentação dos dados até as plataformas de desenvolvimento em nuvem. Juntos, dispositivos IoT e tecnologias se unem para o desenvolvimento das aplicações de IoT que igualmente são diversas em finalidade.

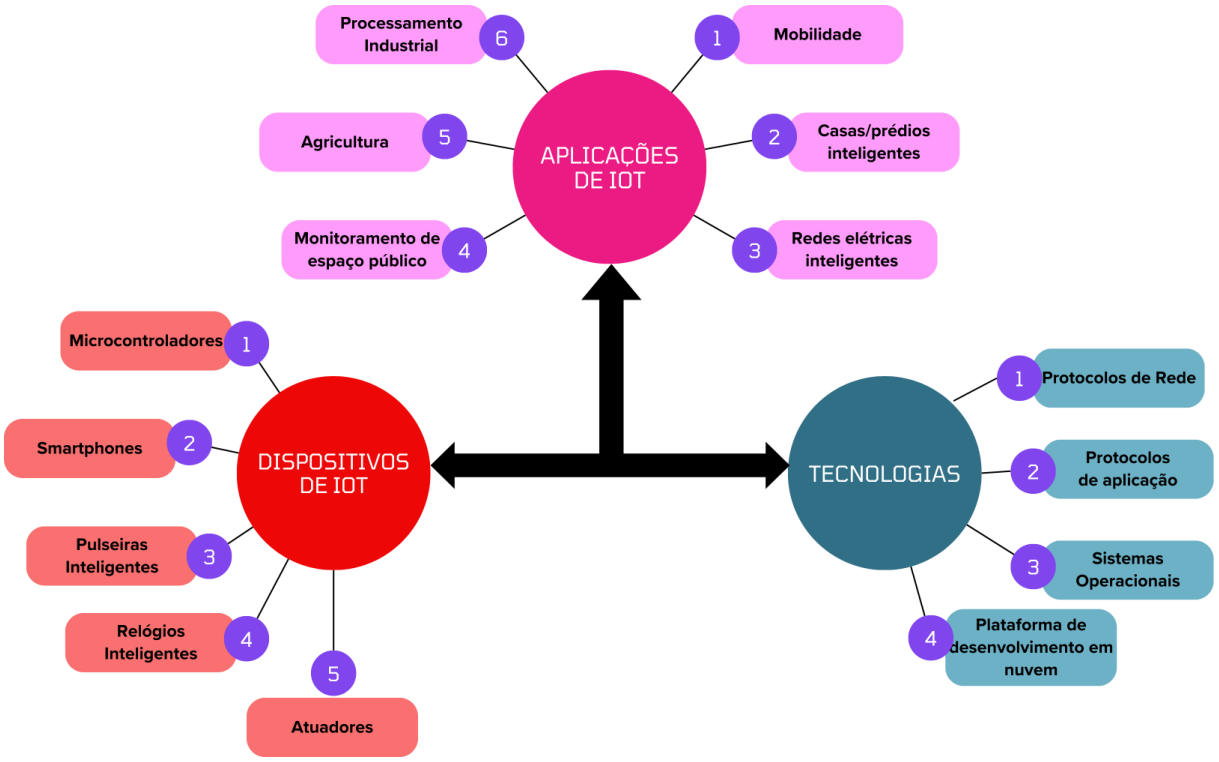


Figura 1 – Diversidade do domínio de IoT.

A falta de padronização de conceitos e características em IoT, mencionada anteriormente, reflete um desafio central para pesquisadores e profissionais da área. Esse cenário instiga a busca por uma maior organização e consenso, motivando diversos trabalhos a propor revisões das definições existentes. Assim, foi conduzida uma análise exploratória da literatura, que identificou três estudos relevantes que realizaram análises aprofundadas sobre definições de IoT, culminando na elaboração de frameworks que destacam características fundamentais dessa área. Na Tabela **??**, tem-se o compilado dos três trabalhos (MOTTA; OLIVEIRA; TRAVASSOS, 2019; FIROUZI et al., 2020; SORRI; MUSTAFEE; SEPPANEN, 2022). Para a compilação destas características, foram consideradas as definições individuais que passaram pelo seguinte processo:

- Cada característica de cada trabalho era comparada com todas as características;
- Se uma definição de característica abordasse o mesmo tema, essas características seriam consideradas iguais e somente uma delas seria incluída no compilado final;
- Se uma característica estiver presente em somente um trabalho, ela também será incluída ao compilado final.

Com base nas definições apresentadas por esses estudos, foi realizado um compilado que sintetiza as principais características de IoT, resultando em 12 categorias distintas.

Durante este processo, observou-se que algumas características eram descritas fragmentadamente ou com terminologias variadas entre os trabalhos. No entanto, por meio de uma análise das definições, foi possível combinar características similares em categorias comuns. A seguir, todas as definições das características de IoT são apresentadas:

- **Interação**: a interação em sistemas IoT exige um meio que permita a comunicação entre os dispositivos, envolvendo uma rede que suporte o desenvolvimento de aplicações, sem se limitar à Internet. Para garantir a troca de informações, os dispositivos IoT precisam comunicar-se eficientemente, viabilizando o uso dos dados coletados. Isso requer que cada dispositivo esteja equipado com um módulo compatível com o protocolo de comunicação apropriado;

- **Coisas**: os objetos IoT englobam tags, sensores, atuadores e diversos tipos de hardware capazes de substituir computadores, ampliando a conectividade. Conforme descrito em Sorri, Mustafee e Seppanen (2022), esses objetos podem ser classificados em duas categorias: o objeto virtual, que inclui atuadores, sensores, dispositivos inteligentes, embarcados e microcomputadores; e o objeto físico, que se refere ao local onde o objeto virtual está integrado ou incorporado;

- **Serviços**: refere-se às aplicações inovadoras, ao aprimoramento digital e ao suporte à tomada de decisão que os serviços IoT oferecem, gerando valor estratégico para as organizações;

- **Tecnologias Padronizadas**: a padronização é fundamental para garantir a conectividade entre os objetos IoT, abrangendo esquemas de endereçamento, protocolos de comunicação, arquiteturas, interfaces e tecnologias habilitadoras;

- **Dados**: refere-se às atividades voltadas para o tratamento, processamento e análise dos dados capturados do ambiente e de outros dispositivos. Esses dados, coletados diretamente dos dispositivos, representam o primeiro passo para a tomada de decisões e podem ser utilizados para enviar comandos de retorno, como ajustes para reduzir o consumo de energia, por exemplo;

- **Ubiquidade**: refere-se à rede de informações e à infraestrutura de conectividade que possibilita uma comunicação em tempo real e de maneira pervasiva. Essa conectividade exige que os dispositivos estejam acessíveis a qualquer momento e em qualquer lugar, garantindo que possam se conectar à rede sempre que necessário;

- **Singularidade**: todos os objetos IoT devem ser unicamente endereçáveis e devem ter identificação automática, além de serem identificados claramente;

- **Ação**: representam uma etapa subsequente ao processamento e referem-se às ações automáticas que podem ser realizadas pelo ou no dispositivo. Além disso, incluem também ações executadas por outras partes interessadas no ecossistema IoT;
- **Heterogeneidade**: refere-se à diversidade de dispositivos que operam em diferentes redes e plataformas, exigindo que todos sejam interoperáveis, capazes de se conectar e trocar dados de forma coordenada e eficiente;
- **Ecossistema**: os dispositivos IoT, os protocolos utilizados, as plataformas que eles operam, as comunidades interessadas nos dados, bem como os objetivos de interesse das partes, todos formam o ecossistema IoT;
- **Inteligência**: Inteligência Artificial (IA) e o aprendizado de máquina possibilitam interações eficazes e a incorporação de inteligência aos dispositivos IoT. Para um sistema ser considerado inteligente, é essencial realizar um conjunto de ações voltadas para o tratamento dos dados e a tomada de decisões;
- **Ambiente**: o ambiente são os locais onde as coisas estão, as ações acontecem, os eventos ocorrem e as pessoas estão. Os sistemas devem ter um conjunto de objetos IoT capazes de sentir, reagir, colaborar e agir sobre um ambiente.

2.3 Algoritmos de Classificação

Algoritmos de aprendizado de máquina são utilizados para automatizar o processo de classificação de requisitos. A tarefa de classificação em algoritmos de aprendizado de máquina se insere no campo da aprendizagem supervisionada, sendo assim, o modelo treina com um conjunto de dados para poder inferir rótulos a dados desconhecidos. Kaur e Kaur (2024) relatam que os algoritmos de aprendizado de máquina que mais são utilizados para a tarefa de classificação binária de requisitos são Máquina de Vetor de Suporte, Multinomial Naive Bayes, Regressão Logística, Árvore de Decisão, *Stochastic Gradient Descent*, *Random Forest*, Passivo Agressivo, *Ada Boost*, *Gradient Boosting*, *K-Nearest Neighbor*.

- **Máquina de Vetor de Suporte (MVS)**: do inglês, *Support Vector Machine* (SVM) esse algoritmo realiza a classificação utilizando um hiperplano em um espaço N-dimensional. Este hiperplano é capaz de separar os dados em diferentes categorias (RYU; CHOI; BAIK, 2016). Um aspecto crucial dessa técnica é a seleção da função de *Kernel*, que define a função de decisão. A escolha do *Kernel* deve ser cuidadosamente ajustada, pois sua adequação depende das características do problema e do conjunto de dados. As MVSs são particularmente eficientes em espaços de alta dimensionalidade e possuem boa utilização de memória computacional. No

entanto, apresentam limitações em relação a conjuntos de dados desbalanceados, sendo vulneráveis à influência de *outliers*;

- **Multinomial Naive Bayes (MNB)**: os métodos de Naive Bayes baseiam-se na aplicação do teorema estatístico de Bayes, partindo do pressuposto de independência condicional entre pares de características (como palavras). O MNB é uma variação projetada para lidar com dados distribuídos segundo uma multinomial, sendo amplamente utilizado na classificação de textos. Esse método apresenta alta eficiência computacional e é bem adaptado para trabalhar com vetores de dados esparsos, uma característica comum em contextos textuais devido à ausência de certas palavras em diversos documentos. Contudo, a hipótese de independência condicional entre palavras frequentemente não se sustenta em textos reais, e o MNB tende a desconsiderar palavras raras ou reduzir sua importância, o que pode impactar negativamente o desempenho do algoritmo (XU; LI; WANG, 2017);

- **Regressão Logística (RL)**: do inglês, *Logistic Regression* (LR), esse algoritmo é classificado como linear, baseado na premissa de que o valor alvo pode ser expresso como uma combinação linear das características. Ele utiliza a função logística para estimar a probabilidade da variável dependente. A regressão logística gera coeficientes interpretáveis, permitindo identificar quais palavras têm maior influência na classificação de texto. Além disso, destaca-se por sua eficiência computacional em cenários com grandes volumes de dados textuais. Contudo, esse modelo apresenta limitações, sendo sensível a *outliers*, inadequado para classes desbalanceadas e pouco eficiente para lidar com problemas de natureza não linear (WANG et al., 2017);

- **Árvore de Decisão**: do inglês, *Decision Tree* (DT) é um método de tomada de decisão que classifica os textos em uma estrutura hierárquica por meio da aplicação de regras de decisão (KAUR; KAUR, 2024). É um método robusto para dados ruidosos e consegue aprender expressões disjuntivas. Para realizar classificação, uma DT ordena suas instâncias de baixo para cima em uma árvore, na qual cada nó corresponde a um teste de um determinado atributo;

- ***Stochastic Gradient Descent* (SGD)**: é uma técnica simples, mas eficaz, de otimização numérica, amplamente utilizada para ajustar classificadores que utilizam funções de perda convexas. Ele é particularmente adequado para tarefas de classificação de texto devido ao seu bom desempenho em dados de alta dimensionalidade e com alta esparsidade. Uma das principais vantagens do SGD é sua flexibilidade, permitindo que, dependendo da função de perda escolhida, assuma características de diferentes modelos, como lineares, quadráticos, classificadores, probabilísticos ou mesmo *perceptrons*. No entanto, a sensibilidade do SGD à inicialização dos hiperparâmetros pode ser um desafio, já que seus resultados podem variar de forma

significativa com ajustes na regularização, número de iterações ou taxa de aprendizado (TIWARI et al., 2020; AL-ANZI, 2022);

- **Passivo Agressivo**: do inglês, *Passive Agressive* (PA) esse algoritmo pertence à categoria de aprendizado de máquina conhecida como aprendizado online, na qual os dados são processados sequencialmente e o modelo é atualizado incrementalmente. Essa abordagem é especialmente vantajosa ao lidar com grandes volumes de dados. O modelo permanece inalterado quando a previsão está correta, mas é ajustado sempre que ocorre um erro. Além disso, esses algoritmos tendem a "esquecer" exemplos anteriores à medida que novos dados são incorporados, o que pode impactar seu desempenho. Embora amplamente aplicados em tarefas de classificação binária, enfrentam desafios ao lidar com problemas de múltiplas classes (TIWARI et al., 2020);

- ***Random Forest* (RF)**: é um método do tipo *bagging*, ou seja, uma amostra aleatória de dados em um conjunto de treinamento é selecionada com substituição. O método RF gera um subconjunto aleatório de recursos e gera diversas árvores de decisão. Para uma tarefa de classificação, é feita uma votação majoritária para definir qual árvore será utilizada para a classificação (KAUR; KAUR, 2024). Nessa abordagem, o risco de *overfitting* é reduzido e é fácil determinar quais recursos contribuem mais para o modelo;

- ***Ada Boost* (AB)**: é um método do tipo *"meta-learning"*, no qual o aprendizado ocorre de maneira iterativa (KUMAR et al., 2022b). O AB se utiliza de diversos classificadores "fracos" para criar um classificador "forte". Algoritmos do tipo boosting tentam reduzir os erros dos modelos anteriores, até que os dados do treinamento sejam previstos perfeitamente, ou o número máximo de modelos tenha sido adicionado. Uma das vantagens deste método é a baixa quantidade de hiperparâmetros porém, para um correto desenvolvimento, é necessário que os dados sejam de qualidade, sendo, portanto, sensível a *outliers* e ruídos;

- ***Gradient Boosting* (GB)**: assim como o AdaBoost, o GB é um método de "boosting" que constrói modelos sequenciais para corrigir os erros dos modelos anteriores. No entanto, o GB otimiza uma função de perda explícita usando técnicas de gradiente, onde cada novo modelo é ajustado para minimizar os resíduos (erros) do modelo anterior. Apesar de poderoso, o GB pode ser computacionalmente intensivo e, assim como o AdaBoost, também pode ser sensível a ruídos e *outliers*, especialmente sem o uso de funções de perda robustas;

- ***K-Nearest Neighbor* (KNN)**: esse algoritmo é classificado como pertencente ao grupo de aprendizado não generalizante, caracterizado por não construir um modelo interno generalizado, mas sim por armazenar as instâncias dos dados de treinamento.

A etapa de classificação consiste em determinar a classe do dado com base na maior frequência entre os vizinhos mais próximos. Um parâmetro crítico nesse método é o número de vizinhos considerados, representado pelo valor de K. Essa técnica é especialmente adequada para lidar com classes multimodais, embora sua eficiência seja relativamente baixa (HUANG et al., 2017).

Para que textos possam ser processados nos algoritmos de aprendizado de máquina, é necessário criar um vetor de características que transforme os textos em vetores numéricos, também conhecidos como *word embeddings*. Para a classificação de requisitos, as técnicas mais utilizadas em *pipelines* com os algoritmos descritos anteriormente são as técnicas tradicionais de seleção de características *Bag-of-words* (BoW) e *Term Frequency-Inverse Document Frequency* (TF-IDF) (KAUR; KAUR, 2024).

- ***Bag-of-words***: é uma maneira simples de representar texto em um vetor numérico. Para um dado conjunto de textos, o vocabulário é extraído e cada palavra vira uma componente do vetor. A partir disso, é feita a contagem de cada palavra presente no texto e assim o vetor de característica criado. Esta abordagem é ineficiente computacionalmente ao criar vetores esparsos e de alta dimensionalidade, do tamanho do vocabulário (PATIL et al., 2023);
- ***Term Frequency-Inverse Document Frequency***: é um método baseado em BoW, mas que utiliza um score estatístico para calcular pesos para cada palavra individualmente encontrada em diversos documentos. É composto por dois scores : (1) frequência do termo que mede com que frequência a palavra aparece em um documento; e (2) frequência inversa que mede a importância de um termo globalmente (PATIL et al., 2023).

2.4 Transformers e Similaridade Semântica

Modelos de Linguagem Grandes (do inglês, *Large Language Models* ou LLMs) são sistemas baseados em IA feitos para compreender, gerar e manipular textos em linguagem natural. Estes modelos aprendem padrões, estruturas gramaticais e o contexto das palavras em grandes volumes de dados textuais. LLMs são geralmente baseadas em aprendizado profundo.

Inicialmente, o processamento de texto entendia as sentenças como uma sequência de palavras e as processava sequencialmente utilizando Redes Neurais Recorrentes (do inglês, Recurrent Neural Network ou RNN), como proposto por Rumelhart, Hinton e Williams (1986). Após o uso de RNN, uma abordagem que proporcionou melhora de desempenho foi a utilização de camadas na rede com Long Short Term Memory (LSTM), proposto por Hochreiter e Schmidhuber (1997).

A abordagem utilizando RNN possuía alguns problemas como: explosão ou desaparecimento de gradientes, dificuldade em capturar dependências de longo prazo, alto consumo de memória e treinamento demorado. Neste contexto, surgiu a arquitetura Transformers proposta por Vaswani et al. (2017), que utilizava o mecanismo de atenção para capturar dependências de longo prazo e processava palavras de uma sentença paralelamente e não sequencial. Na Figura 2 tem-se a arquitetura Transformers, à esquerda tem-se o *encoder* e à direita tem-se o *decoder*.

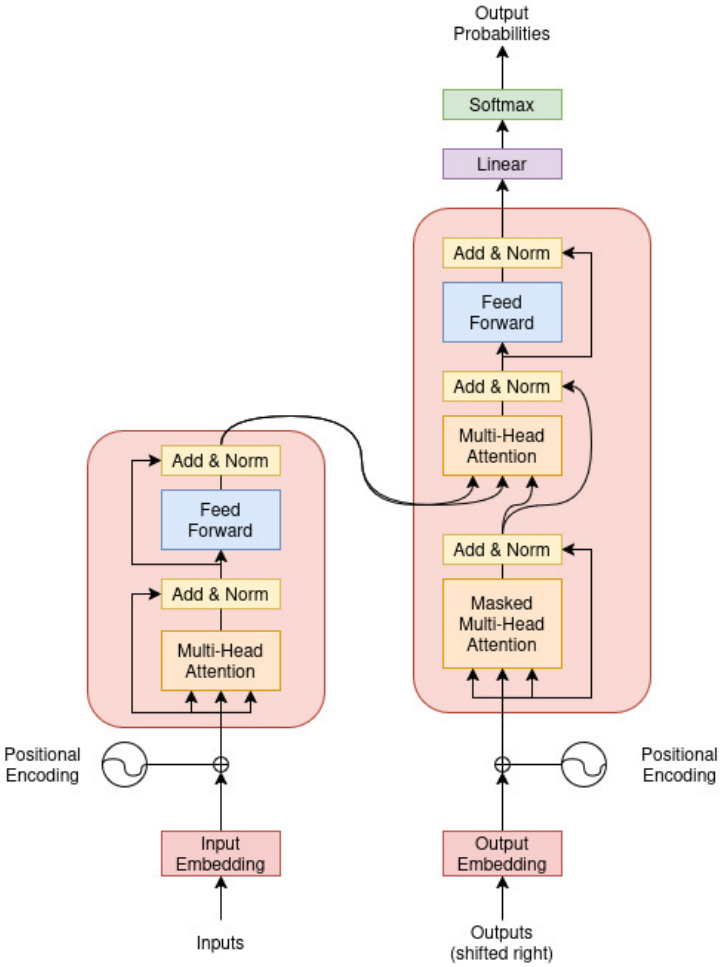


Figura 2 – Arquitetura Transformers, fonte Vaswani et al. (2017).

- ***Encoder*:** é composto de uma pilha de seis camadas idênticas. O objetivo desta etapa é a codificação da sequência de entrada para uma representação vetorial de tamanho constante. Alguns modelos são somente encoders como o RoBERTa (Meta), BERT (Google), DeBERTa (Microsoft), ALBERT (Google), DistilBERT (HuggingFace), XLM-RoBERTa (Meta) e XLNet (Google);

- ***Decoder*:** também composto por uma pilha de seis camadas. Esta etapa consegue gerar tokens ou símbolos baseados em vetores de contexto. Alguns modelos são

somente decoders sendo utilizados para gerar texto a partir de texto, como GPT (OpenAI), PaLM (Google) e Claude (Anthropic).

A utilização da arquitetura Transformers possibilitou a existência dos Modelos de Linguagem pré-treinados (do inglês, *Pre-treined Language Model* ou PLM). Estes modelos possibilitam representações de palavras conforme o contexto, sendo efetivos para extração de características semânticas de maneira geral. Para especializar estes modelos na resolução de tarefas específicas ou de contextos específicos, era necessário executar um ajuste fino (do inglês, *fine tunning*) nos PLM (HOU et al., 2024). Desta forma, tarefas como classificação, análise de sentimentos e similaridade semântica poderiam ser resolvidas com melhor desempenho.

O processo de ajuste fino nestes modelos é a etapa crucial que adapta esses modelos para tarefas específicas, ajustando seus parâmetros em um conjunto de dados menor e especializado. O fine-tuning é um caso específico de aprendizado por transferência, onde um modelo pré-treinado em uma tarefa fonte (modelagem de linguagem) é adaptado para uma tarefa alvo (classificação de texto). Segundo (HOWARD; RUDER, 2018), o ajuste fino pode ser categorizado em três tipos:

- Clássico ou padrão: todos os parâmetros são atualizados. É o método mais simples, porém pode levar ao esquecimento de conhecimento pré-treinado, conhecido como *catastrophic forgetting* (KIRKPATRICK et al., 2016). A seguinte fórmula representa o processo de aprendizado, no qual η é a taxa de aprendizado.

$$\theta_{t+1} = \theta_t - \eta \nabla_\theta \mathcal{L}(\theta_t)$$

- Ajuste discriminativo: cujas taxas de aprendizado são diferenciadas por camadas. A seguinte formula representa o processo de aprendizado para as camadas menores. Nesta fórmula η_l representa a taxa de aprendizado da camada l e L representa o número total de camadas.

$$\eta_l = \frac{\eta_{\text{base}}}{2^{L-l}}$$

- Ajuste com adaptadores: no qual pequenos módulos treináveis são adicionados sem modificar os parâmetros originais. A seguinte formula representa esse processo, no qual, W_1, W_2 são pesos treináveis e h é a ativação da camada.

$$h' = h + f(W_2 \cdot \text{ReLU}(W_1 \cdot h))$$

O ajuste fino minimiza uma função de perda específica para a tarefa. Para classificação, usa-se frequentemente a entropia cruzada (do inglês, *cross-entropy*):

$$\mathcal{L}(\theta) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{C} y_{i,c}\log(p_{i,c})$$

onde:

- θ: parâmetros do modelo
- N: número de exemplos
- C: número de classes
- $y_{i,c}$: rótulo verdadeiro (one-hot)
- $p_{i,c}$: probabilidade prevista

Existem algumas métricas de similaridade para elementos de linguagem natural. As baseadas em léxicos mensuram a similaridade com base no compartilhamento de termos. Já as baseadas em semântica mensuram a similaridade com base no significado dos textos. Para computar a similaridade semântica, é necessária a conversão destes textos em vetores numéricos definidos a partir de um modelo de linguagem, que pode ser desde modelos estatísticos (BoW ou TF-IDF) até os PLMs do tipo *encoder* (BERT ou RoBERTa).

Na Engenharia de Software, alguns trabalhos propuseram a utilização de PLMs na tarefa de similaridade semântica de requisitos de software. Em (ABBAS et al., 2022), os autores investigaram a relação entre requisitos similares e softwares similares, visando o reuso de código a partir de projetos anteriores de uma empresa de ferrovia. Para tal, foram investigados diferentes modelos de linguagem, indo desde TF-IDF e Indice de similaridade de Jaccard (ISJ) até PLMs como Doc2vec, FastText e BERT. Excetuando o ISJ, em todos os outros a similaridade é dada pelo cosseno do ângulo entre os vetores. Os resultados evidenciaram que o BERT tem melhores resultados tanto para correlação entre dois softwares quanto entre requisitos e software.

2.5 Algoritmos de Agrupamento

Algoritmos de agrupamento objetivam agrupar elementos similares de acordo com suas características em grupos. Por grupo entende-se como uma agregação de pontos em um espaço multidimensional no qual a distância entre dois pontos no grupo é menor do que a distância entre um ponto que está no grupo e outro ponto que não está (JAIN; DUBES, 1988). A literatura mostra que existem três tipos principais de algoritmos de agrupamento, cada um com sua particularidade, campo de aplicação, vantagens e desvantagens. Os três tipos principais são: baseados em centroide, baseados em densidade e baseados em hierarquia.

Os algoritmos baseados em centroides partem de uma determinada quantidade de grupos para encontrar quais são os centros geométricos (centroides) que representam cada grupo. Em qualquer algoritmo baseado em centroides, o tema principal é o cálculo da distância entre os objetos de um determinado conjunto de dados. Alguns cálculos utilizados são: distância Euclidiana, Minkowski e Manhattan (UPPADA, 2014). Alguns dos algoritmos desta classe são:

- **K-Means:** é um algoritmo que agrupa os dados tentando separar as amostras em n grupos de variância igual, minimizando um critério de inércia. A inércia assume que o formato dos clusters é convexo, não se adaptando bem a formatos irregulares. Este algoritmo requer que o número de clusters seja especificado. O K-means não se adapta bem a um ambiente de alta dimensionalidade portanto, algum algoritmo de redução de dimensionalidade contribui bastante para o desempenho. O algoritmo é altamente dependente da inicialização dos centroides e um método para melhorar o desempenho é a variação K-means++ que inicializa os centroides distantes um do outro ao invés da aleatoriedade. A complexidade média é dada por O(KnT) sendo n o número de amostras e T o número de iterações;
- **Mini Batch Kmeans**: é uma variação do K-Means que utiliza subconjuntos dos dados de entrada, amostrados aleatoriamente, para reduzir o tempo de computação, enquanto tenta otimizar a função objetvo.

Os algoritmos baseados em densidade possuem a capacidade de identificar grupos com geometrias diversas com base nas regiões de alta concentração de pontos. Tais algoritmos permitem a identificação de *outliers*. Algoritmos baseados em densidade são eficientes para grandes conjuntos de dados (KRIEGEL et al., 2011). Alguns algoritmos desta classe são:

- **DBSCAN:** visualiza os clusters como áreas de alta densidade separadas por áreas de baixa densidade portanto, para este algoritmo os clusters podem ser de qualquer formato. Existem dois parâmetros para o funcionamento do algoritmo, amostras mínimas e eps. A amostra mínima define o número mínimo de vizinhos que um determinado ponto deve possuir para ser um ponto central é o responsável por definir a tolerância do algoritmo em relação a ruído. Já o eps define o limite máximo de distância entre dois pontos para que estes sejam considerados vizinhos é crucial para a escolha do conjunto de dados, sendo necessária uma avaliação criteriosa deste elemento;
- **OPTICS:** é similar ao DBSCAN contudo, relaxa o parâmetro eps e define como único parâmetro o número mínimo de membros para definição de um cluster. No

OPTICS permite-se a extração clusters de densidade variável dentro do mesmo conjunto de dados. Uma das principais diferenças entre este algoritmo e o DBSCAN é quanto a rotulagem da periferia e pontos de ruído e já o OPTICS é mais adequado para uso em grandes conjuntos de dados.

Por fim, têm-se os algoritmos baseados em hierarquia, que são utilizados ao ter um grande número de partições. Cada partição está associada a um nível da hierarquia. Ao contrário dos algoritmos baseados em centroides, não requerem o conhecimento da quantidade de grupos à priori. Tais algoritmos são caracterizados como gulosos, feitos a partir de uma sequência de etapas irreversíveis para construir a estrutura de dados desejada (MURTAGH; CONTRERAS, 2017). Alguns algoritmos desta classe são:

- **Ward:** é um algoritmo de agrupamento hierárquico que constrói clusters com base em divisões sucessivas. Ward visa minimizar a variância em todos os clusters, uma abordagem semelhante ao K-means. É ideal para um ambiente com muitos clusters e com esses clusters desiguais. Para utilizá-lo, são necessários os parâmetros do número de clusters ou do limite da distância entre os pontos;
- **Algoritmos de link:** nessa categoria estão três tipos de algoritmos: *Complete Linkage*, que visa minimizar a distância entre dois grupos; *Average Linkage*, que minimiza a média das distâncias entre dois grupos; e, por fim, *Single Linkage*, que minimiza as distâncias entre os pontos mais próximos de dois clusters.

Algoritmos de agrupamento hierárquico têm início com uma matriz de similaridade ou dissimilaridade que irá computar o quão próximos ou distantes cada um dos objetos está dos demais. Dois objetos estão próximos se sua dissimilaridade é pequena ou sua similaridade é grande (OTI; OLUSOLA, 2024). A matriz resultado possui o tamanho nxn, sendo n a quantidade de objetos ou observações. Em aplicações de agrupamento hierárquico existem diversas formas de se computar a similaridade ou dissimilaridade associada a dois pares de observações. A Tabela 1 exibe algumas medidas utilizadas para similaridade e dissimilaridade de vetores com valores numéricos, bem como suas vantagens e desvantagens:

A partir da matriz de similaridade, o processo de um Algoritmo Hierárquico Aglomerativo, sintetizado por Johnson e Wichern (2002), para agrupar n objetos, é seguinte:

1. Assuma que cada objeto é o seu próprio grupo;
2. Procure na matriz de similaridade o par de grupos mais suscetível a fusão, conforme o algoritmo empregado (Ward, *Complete Linkage*, *Average Linkage*, *Single Linkage*);

Tabela 1 – Métricas de Similaridade.

Métrica	Dissimilaridade (A, B)	Similaridade (A, B)	Vantagens	Desvantagens
Cosseno	$1 - \text{similaridade}(A, B)$	$\frac{\sum_i A_i B_i}{\sqrt{\sum_i A_i^2} \cdot \sqrt{\sum_i B_i^2}}$	• Invariante à magnitude dos vetores, capturando a similaridade de direção. • Útil para dados esparsos (e.g., TF-IDF).	• Não captura a diferença de magnitude entre os vetores. • Pode ser sensível a pequenas mudanças em dados esparsos.
Euclidiana	$\sqrt{\sum_i (A_i - B_i)^2}$	$\frac{1}{1+\text{dissimilaridade}(A,B)}$	• Intuitiva. • Considera diferenças absolutas.	• Sensível à escala e magnitude dos dados. • Não funciona bem para dados esparsos.
Manhattan	$\sum_i \lvert A_i - B_i \rvert$	$\frac{1}{1+\text{dissimilaridade}(A,B)}$	• Mais robusta que a Euclidiana para dados com outliers. • Simples e eficiente.	• Não considera a direção dos vetores. • Sensível à escala.
Minkowski	$(\sum_i \lvert A_i - B_i \rvert^p)^{\frac{1}{p}}$	$\frac{1}{1+\text{dissimilaridade}(A,B)}$	• Generaliza Euclidiana (p=2) e Manhattan (p=1). • Flexível para diferentes valores de pp.	• Exige escolha apropriada de pp. • Sensível à escala.

3. Mescle o grupo a e b. Rotule o grupo recém-formado (ab). Atualize as entradas na matriz de distância, excluindo as linhas e colunas correspondentes ao cluster a e b, adicionando uma linha e coluna que forneça a distância entre o cluster (ab) e os clusters restantes;

4. Repita as etapas (2) e (3) para um total de $n - 1$ vezes, assim todos os objetos estarão em um único grupo após o término do algoritmo. Registre a identidade dos clusters que são mesclados e os níveis (distâncias ou similaridades) nos quais as fusões ocorrem.

Com base no passo 4, o registro das combinações dos grupos, pode-se formar uma representação gráfica disso, os chamados Dendrogramas. Dendrogramas mostram como os elementos ou grupos são agrupados em níveis sucessíveis. Em um Dendrograma, o eixo vertical representa a dissimilaridade entre os grupos e o eixo horizontal representa os grupos ao longo do tempo. Este tipo de gráfico pode ser utilizado para a escolha de um ponto de corte, o qual determina o número de grupos a ser formado (SALMAN et al., 2018). A Figura 3 tem uma representação de uma árvore de dendrograma com um ponto de corte ilustrado.

2.6 Considerações Finais

Este capítulo forneceu uma visão geral sobre diversos conceitos utilizados ao longo da dissertação. Foram trabalhadas as etapas de engenharia de requisitos, diversos

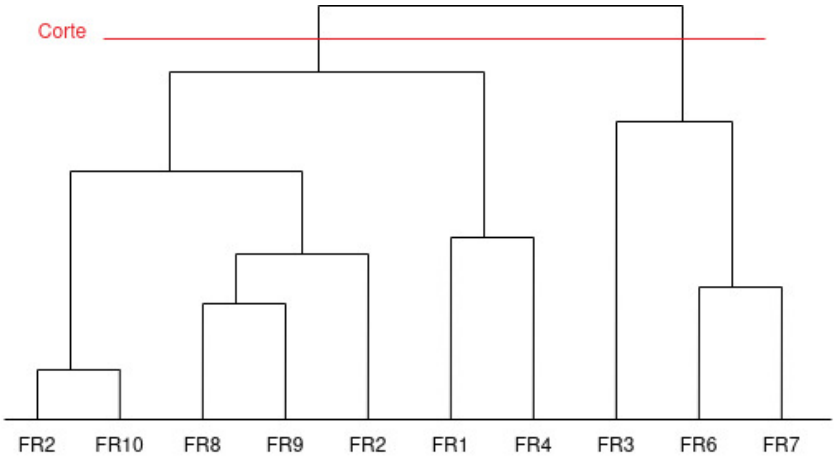


Figura 3 – Representação Gráfica de um Dendrograma. Fonte Salman et al. (2018).

algoritmos de classificação, similaridade semântica e agrupamento, todos com foco em suas aplicações em requisitos.

Adicionalmente, foram levantados conceitos importantes do domínio de IoT e as doze características que serão utilizadas na inferência de características aos grupos, a saber: Interação, Coisas, Serviços, Tecnologias Padronizadas, Dados, Ubiquidade, Singularidade, Ação, Heterogeneidade, Ecossistema, Inteligência e Ambiente. A identificação destas características poderá auxiliar os engenheiros de requisitos na etapa de análise.

Os conceitos apresentados neste capítulo serão necessários para o correto entendimento do próximo capítulo. Neste capítulo, são apresentados os trabalhos encontrados na literatura que realizam agrupamento de requisitos. Os trabalhos foram comparados quanto ao tipo de requisito processado, como é feita a classificação de requisito, como é feita a computação de similaridade entre os requisitos e o contexto de software que o método dá suporte.

3 Trabalhos relacionados

A literatura apresenta um conjunto de métodos e ferramentas que visam apoiar o agrupamento de requisitos para diferentes finalidades, em diferentes contextos e avaliados de diferentes formas. Nesse contexto, destacam-se as aplicações de agrupamento que organizam requisitos elicitados em linguagem natural em grupos com base em suas características, visando decompor o software em subsistemas.

3.1 Técnicas baseadas em algoritmos não hierárquicos

Esta subseção apresenta as abordagens que empregam algoritmos de agrupamento não hierárquicos na organização de requisitos de software. Esses métodos incluem algoritmos clássicos como K-Means, DBSCAN e métodos baseados em coesão temática. Tais técnicas requerem normalmente a definição prévia ou automática do número de grupos e favorecem maior escalabilidade em comparação com os métodos hierárquicos. Além disso, permitem maior flexibilidade na combinação com diferentes vetores de representação e técnicas de pré-processamento.

Em Casamayor, Godoy e Campo (2012b), foi proposto um método para minerar documentos de requisitos escritos em linguagem natural, visando a detecção semiautomática de componentes. Inicialmente, um conjunto de requisitos é processado e classificado em funcionais e não funcionais utilizando uma técnica semiautomática. Após isso, cada requisito passa por um etiquetador de *part-of-speech* e é posteriormente processado com um conjunto de regras para identificar cada responsabilidade. No momento final, as responsabilidades são agrupadas utilizando algoritmos não supervisionados (*Expectation-maximization*, COBWEB, X-Means e DBSCAN). O método foi avaliado em três estudos de caso e os resultados relatam que os grupos obtidos corresponderam aos componentes esperados previamente pelos analistas.

Em Galster, Eberlein e Jiang (2013), os autores propõem uma técnica que se divide em três fases. Na primeira, chamada de *preparation*, os requisitos são atomizados, sem diferenciação entre requisitos funcionais e não funcionais. Nesta etapa, os requisitos são avaliados quanto à sua relevância de maneira manual. Na segunda fase, chamada de agrupamento, na qual os requisitos vão ser agrupados com base em seus atributos. Por fim, manualmente os clusters são descritos com informações sobre quando, onde, como e por que aquele grupo de requisitos será implementado. A técnica é avaliada mediante a um estudo de caso com 72 requisitos. O estudo de caso visava identificar se a técnica auxiliava na transição de software para arquitetura e se era viável na prática. Quanto à viabilidade, relatam que os requisitos precisam ser atomizados, o que pode gerar trabalho

extra; e pode-se ter problemas na avaliação manual dos requisitos e a técnica é sensível a esta avaliação.

Em Kumar et al. (2022a), os autores propõem uma forma de utilizar o algoritmo K-Means para essa tarefa de agrupamento de requisitos no contexto de engenharia de requisitos ágeis. O processo é dividido em três fases: (1) pré-processamento, onde os requisitos passam por tokenização, remoção de stopwords e lematização, acontece nesta fase também a vetorização com Bag-of-words simples e utilizando TF-IDF; (2) tem-se o agrupamento a partir da utilização do K-Means; (3) na validação, tenta-se encontrar o melhor valor para número de clusters com base no coeficiente de silhueta. Para avaliar a proposta, foram utilizadas 98 user story e para a análise de clustersótimos,s foram testados de 2 a 14 como valores de k.

3.2 Técnicas baseadas em algoritmos hierárquicos

Esta subseção reúne os trabalhos que empregam algoritmos de agrupamento hierárquico para a organização de requisitos de software. Essas abordagens compartilham a característica de estruturar os requisitos em forma de dendrogramas, permitindo uma visualização gradual e interpretável dos agrupamentos. Entre os algoritmos utilizados, destacam-se o *Hierarchical Agglomerative Clustering* com variações como *Single Linkage*, *Complete Linkage* e *Average Linkage*. Os trabalhos aqui analisados aplicam diferentes técnicas de pré-processamento e cálculo de similaridade, mas têm em comum o uso de hierarquia como estratégia principal de agrupamento ou como critério de comparação.

Em Al-Otaiby, AlSherif e Bond (2005), os autores propõem uma técnica de agrupamento utilizando como base algoritmos hierárquicos, voltada exclusivamente para requisitos escritos na forma de cenários. A técnica proposta se divide em três fases. Primeiro identificam-se os objetos que farão parte da relação entre os cenários. Por "objeto" entende-se pelo resultado da ação de um verbo na frase. O produto final da primeira fase é uma tabela cenário x atributos. Após isso, tem-se a medição da similaridade por meio da tabela anterior. O produto desta fase é a geração de uma tabela cenário x cenário, computando a similaridade entre os requisitos. Na última fase é aplicado o algoritmo hierárquico para agrupar os requisitos com base na tabela anterior. Os autores avaliaram a técnica por meio de um estudo de caso com 20 cenários obtidos de um livro. Foram utilizadas as três variações do algoritmo hierárquico: *Complete Linkage*, *Average Linkage* e *Single Linkage*. Há uma variação em relação ao algoritmo inicialmente proposto no artigo, pois a dissimilaridade foi usada no lugar da similaridade. Concluiu-se após as comparações entre as variações que o *Complete Linkage* obtém os melhores resultados e que é consistente visando decompor o sistema em subsistemas.

Em Misra, Sengupta e Podder (2016), é proposta uma técnica de agrupamento

que utiliza o modelo semântico latente LDA, voltada para requisitos que estejam livres de erros linguísticos e de ambiguidade. A técnica proposta divide-se em três etapas mais gerais. A primeira é responsável pelo pré-processamento do corpus de requisitos, aqui serão usadas técnicas de PLN como POS-tagging, chunking, *lemmatization* e *tokenization*. Após isso é gerado um corpus com os termos dos requisitos sem *stopwords*. Na segunda etapa, gera-se o modelo latente, com a medição da similaridade semântica através do cosseno do ângulo dos vetores LDA, para verificar a relação semântica entre os termos e os requisitos. Na terceira etapa, identifica-se o conjunto de tópicos e seus clusters. As relações entre requisitos são montadas como um grafo não direcionado ponderado. Os autores avaliaram sua técnica por meio de um experimento com especificações de requisitos, que totalizaram 41 requisitos. Para a definição do número alvo de grupos, os autores utlizaram classificações do próprio documento, totalizando 10. Foram comparados os algoritmos de agrupamento: *Theme based Clustering*, *Hierarquical average linkage* e *Particional Clustering*. As métricas utilizadas foram Pureza, Pureza Inversa e F-1. O algoritmo proposto no artigo, *theme based*, obteve os melhores resultados.

Em Salman et al. (2018), os autores propõem também uma forma de agrupamento de requisitos funcionais. O processo é dividido em quatro partes: (1) o documento de especificação de requisitos é obtido sendo selecionados manualmente os requisitosfuncionais; (2) os requisitos são *tokenizados*, as *stopwords* são removidas sendo realizado *stemming*; (3) é feita a computação da similaridade semântica entre os requisitos baseando-se em quando dois ou mais requisitos funcionais compartilham muitos *tokens*, espera-se que esses requisitos estejam relacionados à mesma tarefa; (4) é utilizado um algoritmo hierárquico aglomerativo para agrupar os requisitos similares em grupos. Para a avaliação de resultados, os autores realizam um estudo de caso com quatro documentos de requisitos de diferentes domínios e tamanhos, investigou-se o número de clusters estão semanticamente corretos e se o número de clusters identificados está próximo do número real, tal qual no documento original. Para a primeira avaliação, os autores utilizam as métricas de precisão e recall para concluir que a proposta identifica os grupos corretamente conforme a semântica. Já quanto ao número de clusters identificados, concluiu-se que eles são iguais ou muito semelhantes ao número real.

Em Kochbati et al. (2021), os autores propõem uma forma de agrupar requisitos utilizando algoritmos hierárquicos aglomerativos. São propostas três etapas: (1) há um pré-processamento dos requisitos realizando tokenização, remoção de stopwords, remoção de pontuação e stemming; (2) há computação de similaridade semântica em dois níveis, localmente utilizando word2vec e similaridade semântica entre cada par de vetores pertencentes a dois requisitos, e globalmente, feita com base na similaridade local e utilizando idf; (3) a fase de agrupamento, utilizou-se como critério de agrupamento a matriz de similaridade semântica dos requisitos, para automatizar o número ótimo de grupos utilizou-se o índice de Dunn, um índice de validade interna usado para avaliar o

resultado do agrupamento, quanto maior este índice, melhor a solução do agrupamento. Como forma de avaliação, os autores propõem um estudo de caso com quatro documentos de requisitos. Todas as análises são feitas comparando com o documento de especificação de requisitos inicial. A classificação dos requisitos funcionais é feita manualmente.

3.3 Análise comparativa

As Tabelas 2 e 3 apresentam uma análise comparativa entre os trabalhos encontrados, objetivando apresentar como os estudos desenvolvidos abordam questões de tipo de requisito processado, classificação de requisito, computação de similaridade e contexto do software. Os critérios adotados para realizar a comparação entre os trabalhos são:

- **tipo de requisito processado:** verifica qual tipo de requisito elicitado em linguagem natural é processado pelo método.
- **classificação de requisito:** verifica se o método realiza classificação dos requisitos em funcional e não funcional antes de realizar o agrupamento.
- **computação de similaridade:** verifica qual técnica é utilizada para realizar a computação da similaridade entre os requisitos.
- **contexto de software:** verifica qual contexto de software o método é especialista para realizar o agrupamento.

Tabela 2 – Comparação entre trabalhos relacionados (Parte 1).

Referência	Tipo de requisito processado	Classificação de requisito
Al-Otaiby, AlSherif e Bond (2005)	Cenário	Não realiza
Casamayor, Godoy e Campo (2012b)	Não estruturada	Semi-automática
Galster, Eberlein e Jiang (2013)	Não estruturada	Manualmente
Misra, Sengupta e Podder (2016)	Não estruturada	Não realiza
Salman et al. (2018)	Não estruturada	Manualmente
Kochbati et al. (2021)	Não estruturada	Manualmente
Kumar et al. (2022a)	User story	Não realiza
Método Proposto	Não estruturada	Automática

O tipo de requisito processado no método é determinante para o seu uso. Conforme Franch et al. (2023), após uma série de entrevistas realizadas em 12 empresas de software, o método de especificação de requisitos mais utilizado é o de requisitos não estruturados no formato de sentenças em linguagem natural. Dos trabalhos identificados na literatura, somente dois trabalhos Al-Otaiby, AlSherif e Bond (2005), Kumar et al. (2022a) não abordam esse tipo de requisito. Na solução proposta neste trabalho, a ReqCluster4IoT, será contemplado o tipo não estruturado. Conforme IEEE Computer Society Staff (2024),

Tabela 3 – Comparação entre trabalhos relacionados (Parte 2).

Referência	Computação de Similaridade	Contexto de Software
Al-Otaiby, AlSherif e Bond (2005)	Baseado em atributos	Geral
Casamayor, Godoy e Campo (2012b)	Não realiza	Geral
Galster, Eberlein e Jiang (2013)	Baseado em atributos	Geral
Misra, Sengupta e Podder (2016)	Não realiza	Geral
Salman et al. (2018)	TF-IDF	Geral
Kochbati et al. (2021)	word2vec e IDF	Geral
Kumar et al. (2022a)	TF-IDF	Geral
Método Proposto	Transformers	IoT

uma especificação típica de requisitos de linguagem natural não estruturada é uma coleção de declarações em linguagem natural, como, “O sistema deve...”.

Classificação de requisitos é um processo realizado durante a etapa de análise dos requisitos. Esta classificação pode ser binária, envolvendo requisitos funcionais, que descrevem funções do software ou de seus componentes, e requisitos não-funcionais, que descrevem as restrições impostas ao software e ajudam a garantir que o software atenda às necessidades dos usuários. Dos trabalhos levantados na literatura, três realizam esta classificação manualmente. Em Casamayor, Godoy e Campo (2012b) é realizada uma classificação semi automática que considera a avaliação manual dos usuários do método.A classificação manual dos requisitos é uma tarefa difícil, demorada e propensa a erros, especialmente para projetos de software com muitos requisitos (CANEDO; MENDES, 2020; QUBA et al., 2021). Outros três trabalhos não realizam a classificação. Os requisitos não funcionais possuem aspecto horizontal ao sistema, ao não realizar esta distinção, restringir este tipo de requisito a um só grupo limita seu caráter horizontal. Portanto, nesta pesquisa optou-se por utilizar um processo automatizado com classificação feita utilizando Inteligência Artificial para a identificação dos requisitos funcionais.

Com relação à computação de similaridade, modelos baseados em Transformers superam métodos tradicionais, como TF-IDF e Word2Vec, tanto em precisão quanto em robustez ao lidar com variações contextuais. De acordo com Reimers e Gurevych (2019), Transformers ajustados para tarefas de similaridade semântica demonstraram melhores índices de correlação de Pearson e Spearman nas tarefas de Similaridade Textual Semântica (STS-Benchmark) e Paridade de Pergunta (QQP). Além disso, em Devlin et al. (2018), os autores destacaram que os Transformers conseguem captar nuances contextuais de palavras e frases ao considerar a bidirecionalidade, o que confere aos Transformers capacidade de compreender o significado de frases inteiras.

Com relação ao contexto de software, foi identificado que na literatura não existem métodos que sejam específicos para um contexto de software. A ReqCluster4IoT é um trabalho voltado para a análise de requisitos de IoT. Softwares para IoT apresentam maior complexidade na especificação em comparação com softwares tradicionais. Esse paradigma

apresenta diversos desafios, como a elicitação e descrição dos componentes de hardware e software, além do gerenciamento e detalhamento das interações entre esses elementos. A alta diversidade de elementos que estão presentes nas aplicações é um dos elementos que agravam esta dificuldade.

3.4 Considerações Finais

Este capítulo realizou uma análise de abordagens existentes para o agrupamento de requisitos, destacando suas características, limitações e contribuições. Este levantamento bibliográfico evidenciou que, embora existam diversas técnicas e métodos propostos, a maioria deles apresenta lacunas significativas, especialmente quanto ao processamento de requisitos não estruturados, à classificação automática de requisitos funcionais e não funcionais, e à aplicação em contextos específicos, como o de IoT.

A importância do método proposto neste trabalho, o ReqCluster4IoT, está justamente na abordagem dessas lacunas. Ao focar em requisitos não estruturados, os quais são os mais comumente utilizados na prática industrial. Além disso, a classificação automática de requisitos, realizada por meio de técnicas de Inteligência Artificial, elimina a dependência de processos manuais, que são propensos a erros e demandam tempo considerável. A utilização de modelos baseados em Transformers para a computação de similaridade semântica também representa um aprimoramento, pois esses modelos superam métodos tradicionais em termos de precisão e capacidade de capturar nuances contextuais.

Outro aspecto importante é a especialização do método para o contexto de IoT, um paradigma que apresenta desafios únicos devido à complexidade inerente à interação entre hardware e software, bem como à diversidade de elementos envolvidos. A ReqCluster4IoT se propõe a lidar com essas particularidades, oferecendo uma solução adaptada às necessidades específicas desse domínio.

No capítulo seguinte, é apresentada a construção do método ReqCluster4IoT, detalhando cada etapa do processo, desde o pré-processamento dos requisitos até a identificação de característica de IoT em cada grupo. Serão discutidas as decisões tomadas para cada etapa do método. Além disso, serão apresentados resultados de experimentos que servem de justificativa para quais modelos são utilizados em cada etapa.

4 O método ReqCluster4IoT

O objetivo desta dissertação é auxiliar analistas de requisitos e desenvolvedores de aplicações de IoT com um método de agrupamento de requisitos funcionais de software. O foco é que o método extraia os requisitos funcionais de forma automática a partir de uma lista de requisitos, agrupe-os em grupos de contextos semanticamente similares e, posteriormente, consiga indicar qual(is) característica(s) de IoT aquele grupo trata. Para tal foi desenvolvido o método ReqCluster4IoT, apresentado na Figura 4 com as suas cinco etapas, a saber: (1) Requisitos textuais elicitados; (2) Extração de Requisitos Funcionais, descrito com mais detalhes na Seção 4.1; (3) Computação da Similaridade Semântica, mais detalhes na Seção 4.2; (4) Agrupamento de Requisitos, detalhes na Seção 4.3; e (5) Identificação de Características de IoT, detalhes na Seção 4.4.

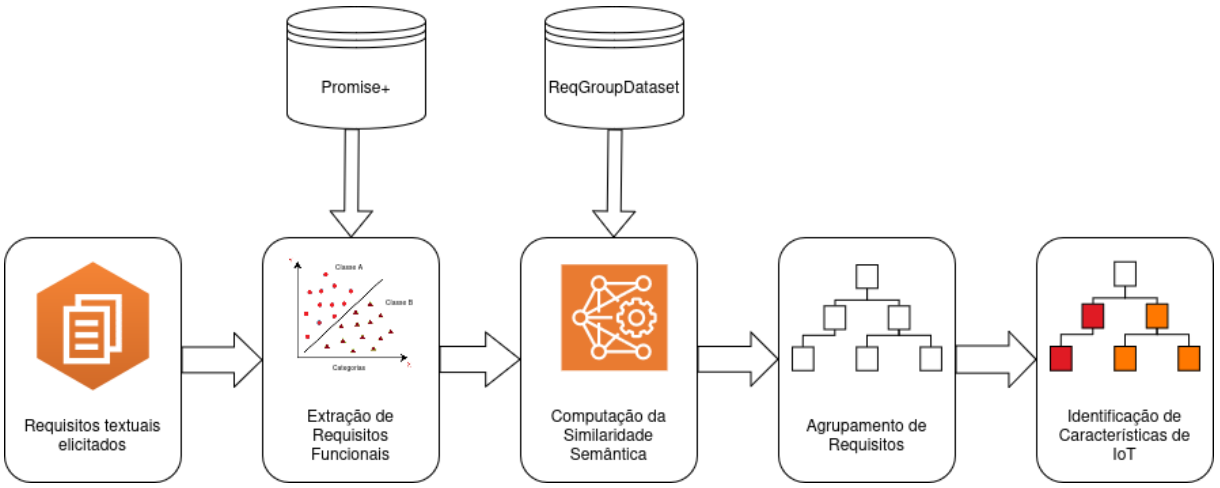


Figura 4 – Etapas da ReqCluster4IoT.

A primeira etapa trata da entrada dos requisitos no método, estes requisitos devem ser passados como uma lista, como sentenças em linguagem natural não estruturada. Os requisitos então são classificados de forma binária, em requisitos funcionais e não funcionais, sendo passados para a próxima etapa os funcionais. Uma vez identificados os requisitos funcionais, a próxima etapa é a computação da similaridade entre os requisitos. Em seguida, há a etapa de agrupamento de requisitos, na qual os requisitos são agrupados utilizando algoritmo de agrupamento hierárquico. Por fim, uma vez formados os grupos, é identificada a característica de IoT mais compatível com cada grupo.

4.1 Extração de Requisitos Funcionais

A tarefa de classificação em algoritmos de aprendizado de máquina se insere no campo da aprendizagem supervisionada, sendo assim, o modelo treina com um conjunto de dados para poder inferir rótulos a dados desconhecidos. O ponto central está, portanto, no conjunto de dados utilizados no treinamento. No contexto de requisitos, Kaur e Kaur (2024) indicam que um dos conjuntos de dados mais utilizados é o PROMISE Dataset (SHIRABAD; MENZIES, 2005). A base PROMISE possui 12 classes possíveis, sendo F para requisitos funcionais e o restante para diferentes classes de requisitos não-funcionais: A (*Availability*), L (*Legal*), LF (*Look and Feel*), MN (*Maintainability*), O (*Operacional*), PE (*Performance*), SC (*Scalability*), US (*Usability*), FT (*Fault Tolerance*) and PO (*Portability*).

Porém, Navarro-Almanza, Juarez-Ramirez e Licea (2017) indicam que essa base possui baixa quantidade de dados para aplicações de aprendizado de máquina. Assim, surgem outras iniciativas para expansão, como a PROMISE_exp em Lima et al. (2019) que incorporou requisitos de diferentes fontes. Apesar dos esforços, ainda há relatos sobre limitações dessa expansão, como conjuntos de dados desbalanceados, conforme destacado por Canedo e Mendes (2020) e Quba et al. (2021). Além disso, existem problemas como a baixa quantidade de dados para aplicações de aprendizado de máquina e conjuntos de dados com amostras pouco diversas em relação ao tipo de software.

A necessidade de avaliar como a ampliação dessas bases de dados impacta o desempenho de algoritmos de aprendizado de máquina em tarefas de classificação de requisitos se torna evidente, pois a introdução de novos dados pode influenciar diretamente na precisão e eficácia desses algoritmos. Outro aspecto que fortalece a proposição de novas expansões é a introdução de algoritmos de redes neurais e aprendizado profundo, algoritmos que necessitam de abundância de dados (TAYE, 2023). Desta forma, realizou-se uma nova expansão da base de requisitos, a Promise+ em Silva et al. (2024), tomando como ponto de partida a base PROMISE_exp. O processo e os principais resultados do artigo estão descritos na Seção 4.1.1.

Modelos de aprendizagem de máquina tradicionais são mais utilizados na literatura para esta tarefa, porém existe crescimento na utilização de modelos baseados em Transformers (KAUR; KAUR, 2024). Neste contexto, é necessário um experimento para investigar a eficácia de modelos pré-treinados baseados em Transformers na tarefa de classificação de requisitos, comparando-os com algoritmos de aprendizado de máquina tradicionais. O experimento é descrito na Seção 4.1.2. Os resultados deste experimento determinaram qual modelo foi utilizado para a classificação de requisitos no método final.

4.1.1 Promise+

A condução desta nova expansão da base de requisitos PROMISE teve como objetivo aumentar o número de requisitos, mantendo a qualidade e seguindo os princípios orientadores da primeira expansão, a PROMISE_exp (LIMA et al., 2019). O aumento quantitativo do número de instâncias do conjunto de dados também passou por um processo de validação e avaliação. Na etapa de validação, o objetivo é verificar o consenso de classificação entre os especialistas em relação à classificação do requisito de software. Já na etapa de avaliação, a expansão será analisada quanto à sua eficácia na geração de modelos classificadores. Esta expansão foi publicada em (SILVA et al., 2024).O esquema adaptado possui as seguintes etapas (LIMA et al., 2019):

- Busca focada na web por SRS;
- Análise dos SRS encontrados;
- Extração de dados dos SRS;
- Classificação dos requisitos que não possuem uma classe válida;
- Validação dos requisitos classificados;
- Geração da Promise+;
- Avaliação da Promise+.

O motor de busca da Google[1] foi usado para recuperar documentos de novas especificações de requisitos. As buscas foram realizadas utilizando a sequência "*Software Requirements Specification*". Inicialmente, a única restrição foi quanto ao uso da língua inglesa no documento, por ser a linguagem encontrada tanto na PROMISE quanto na PROMISE_exp. Qualquer documento que violasse essa restrição não seria considerado para análise. Além disso, era restrita também a duplicação de documentos cujos requisitos já estivessem na base.

As etapas de análise de documentos e extração de dados foram realizadas de maneira simultânea. Na análise de documentos, o objetivo era verificar tanto a estrutura quanto a forma de documentação dos requisitos. Apesar da existência de normas e recomendações para a criação de documentos de requisitos, como as presentes nas *International Organization for Standardization* ou ISOs ISO/IEC/IEEE 29148 (2018) e IEEE 830 (1998), é comum encontrar documentos com formatos confusos ou improvisados. Esta etapa visou eliminar documentos com estruturas incompreensíveis. Outro aspecto analisado foi como os requisitos são documentados, uma vez que existem diferentes métodos

[1] www.google.com.br

de documentação (protótipos, diagramas de caso de uso, histórias de usuário, entre outros). O único método válido em nosso contexto era considerar documentos nos quais os requisitos fossem documentados na forma de frases.

Na extração de dados, dois pesquisadores preencheram um formulário com os seguintes dados:

- Ano: ano de publicação do documento;
- Descrição do Software: os pesquisadores selecionavam os campos de resumo e introdução dos documentos. Este item foi posteriormente utilizado para realizar a análise quanto ao tipo de software;
- Requisitos classificados: os pesquisadores selecionavam os requisitos que estivessem descritos em um tipo válido (Funcional ou alguma das onze classes de não-funcionais contidas na PROMISE_exp);
- Requisitos não classificados: os pesquisadores selecionavam os requisitos descritos com um tipo diferente das 12 categorias definidas na PROMISE_exp.

Assim como na primeira expansão, neste trabalho também optou-se por preservar as classificações trazidas no documento, caso fossem válidas, ou seja, requisitos com alguma das 12 classes da PROMISE_exp. Portanto, os requisitos já classificados foram inseridos diretamente na Promise+. No entanto, os requisitos não classificados precisavam ser classificados por especialistas.

Nesta etapa, entraram cinco especialistas, cada um classificou aproximadamente 20% dos requisitos extraídos. Os especialistas receberam uma lista com os requisitos, uma descrição destes requisitos e eram instruídos a classificar os requisitos conforme as classes pré-definidas. Os avaliadores também eram encorajados a realizar um comentário caso encontrassem algum problema de escrita ou detectassem que aquele requisito era de uma classificação diferente às classes pré-determinadas.

Para validar a classificação realizada pelos cinco especialistas, um sexto especialista classificou todos os requisitos. Para analisar o consenso na classificação, utilizou-se o teste Kappa (COHEN, 1960). Os resultados da aplicação do teste Kappa são detalhados na Seção 4.1.1.1. A experiência de cada especialista em Engenharia de Requisitos é listada na Tabela 4. Destaca-se que os especialistas têm vasta experiência trabalhando com Engenharia de Requisitos em projetos de desenvolvimento e/ou orientações e com produção científica.

Para a avaliação da Promise+, foram realizadas as três tarefas de classificação de requisitos de software Lima et al. (2019): (1) classificação binária entre requisitos funcionais e não-funcionais; (2) classificação multiclasse entre os requisitos não-funcionais

Tabela 4 – Experiência dos especialistas

Esp	Escolaridade	Experiência (em anos)	Artigos em ER
A	Dr.	13	12
B	Me.	11	3
C	Dr.	14	3
D	Dr.	24	36
E	Me.	6	2
F	Dr.	14	14

e, por fim, (3) classificação multiclasse com todas as classes disponíveis da base. Os resultados comparados são entre a PROMISE_exp e a Promise+. Com os requisitos passando por pré-processamento com: tokenização, remoção de *stopwords*, lematização e vetorização. Foram utilizados os algoritmos de AM (KNN, MVS, RL, SGD, MNB, PA), instanciados sob as mesmas condições de hiper parâmetros, com divisão treinamento/teste numa proporção 90%/10%. Utilizou-se também o processo de validação cruzada Kfold no treinamento, com k=10. Os resultados apresentados neste trabalho são da aplicação dos algoritmos treinados com o melhor *fold* e aplicados nos dados de teste.

Para a análise quanto ao tipo de software, foram utilizadas as seguintes classificações de software: sistema, aplicativo, científico/de engenharia, embarcado, linha de produtos, web e inteligência artificial (PRESSMAN; MAXIM, 2016).

4.1.1.1 Resultados

Após a busca no Google, 85 documentos foram encontrados. No entanto, após excluir seis documentos com requisitos já presentes na PROMISE_exp, apenas os documentos identificados por Ferrari, Spagnolo e Gnesi (2017), no *dataset* PURE, permaneceram na análise deste trabalho. O *dataset* PURE consiste em 79 documentos de especificação de requisitos públicos coletados na web e escritos em linguagem natural. Os autores tinham como objetivo fornecer um conjunto de dados para tarefas típicas de PLN em RE, como identificação de requisitos e avaliação da estrutura do documento. Eles também previram que os requisitos poderiam ser manualmente anotados para servir como *benchmarks* para tarefas como classificação de requisitos.

Na Figura 5, apresentam-se a condução e os resultados das seis iniciais etapas da metodologia. Após análise dos documentos, 37 foram rejeitados e 42 foram aceitos. Dos 42 aceitos, 57% (24) são documentos de software da Indústria e 43% (18) são de documentos de software de projetos universitários. O dataset PURE possui requisitos espalhados em seus documentos. Para a correta extração dos requisitos, dois pesquisadores analisaram todos os documentos internos conforme a classificação das seções nos documentos (e.g., *"Requirements", "Non-Functional", "Functional requirements", "Quality requirements"*). Assim, foram extraídos os requisitos de cada documento com a sua respectiva classificação.

Portanto, dos documentos aceitos, foram extraídos 2703 requisitos, sendo que 53,6% (1448) continham uma classe válida e foram diretamente para a Promise+. Os 46,4% (1255) restantes eram requisitos sem classificação, enviados para os especialistas.

Na Figura 5, tem-se também o resultado da validação do sexto especialista com os cinco especialistas. Para essa validação, utilizou-se o resultado do índice Kappa que foi, respectivamente: 0,6462, 0,7213, 0,7112, 0,4126 e 0,4094. Conforme Cohen (1960), houve, portanto, acordo substancial para os três primeiros e acordo moderado para os dois últimos. Para os casos em que a concordância foi moderada (Kappa <0,6), optou-se por inserir na base a classificação informada pelo especialista de maior experiência.

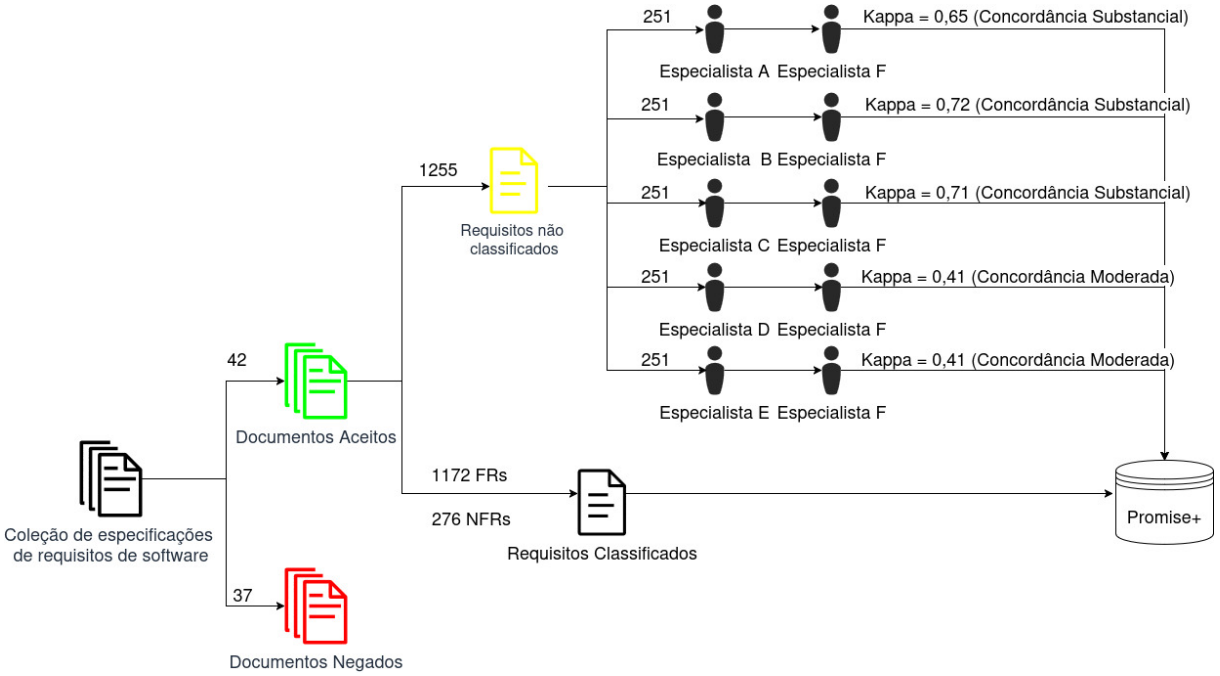


Figura 5 – Resultados passo a passo da metodologia.

No momento da classificação manual, os especialistas encontraram requisitos que, em seus entendimentos, pertenceriam a uma classe diferente das pré-determinadas. As classes de requisitos não funcionais citadas foram: hardware, linguagem de programação, integração entre sistemas e interoperabilidade. Por vezes, alguns requisitos foram também considerados regras de negócio. Os especialistas entraram em consenso que 25 requisitos são de classe inválida. Estes requisitos estão adicionados à base em separado. A Promise+ está disponível para uso e consulta[2].

Na Figura 6 é apresentada a distribuição anual dos documentos. Percebe-se que boa parte dos documentos tem data após 2005 (72%). Dos documentos encontrados, somente em três não se teve nenhuma identificação direta do ano em que foi feito.

[2] https://zenodo.org/records/12805484

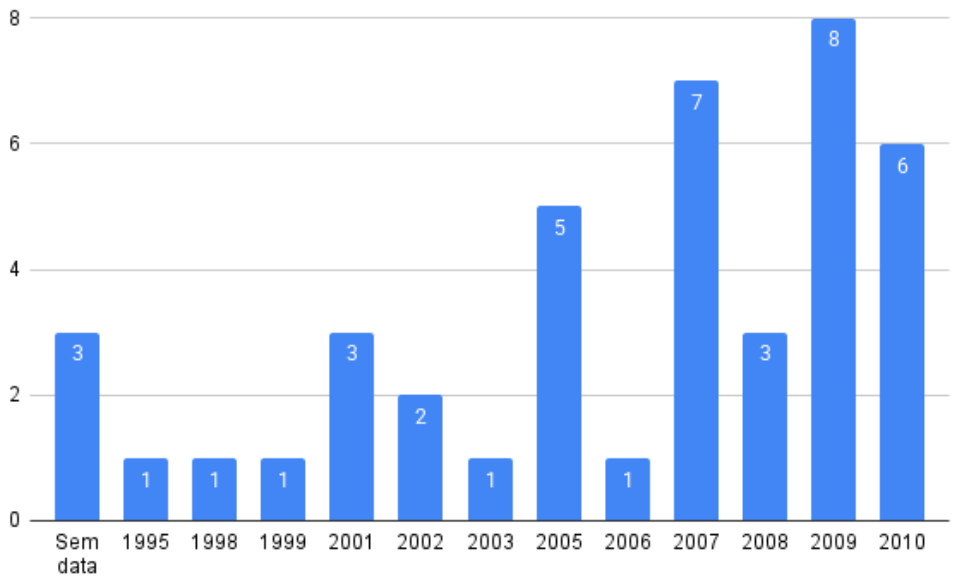


Figura 6 – Lista anual de documentos.

Quanto aos tipos de software identificados na base, observou-se que somente 4 dos 7 tipos de software foram encontrados, conforme Figura 7. Não foram encontrados os tipos de software científico/de engenharia, software de linha de produtos e software de inteligência artificial. A ausência dos dois primeiros tipos se deve, provavelmente, ao fato de que esse tipo de software geralmente resulta em patente ou propriedade intelectual. Já a ausência do último tipo pode estar relacionada ao período, conforme visto anteriormente, os documentos encontrados datam até 2010, momento em que esse tipo de software (inteligência artificial) não era tão difundido.

Dos tipos de software encontrados, tem-se: (1) Software de Sistema são programas feitos para atender outros programas, como compiladores, funções utilitárias, componentes de sistema operacional, dentre outros; (2) Software de Aplicação são programas independentes que solucionam uma necessidade específica do negócio; (3) Software Web contempla aplicativos voltados para navegadores; e, por fim (4) Software Embarcado é uma aplicação que reside em um produto ou sistema sendo utilizado para controlar características para o próprio sistema.

A Tabela 5 exibe a comparação quantitativa entre a PROMISE_exp e a Promise+. Na PROMISE_exp as classes minoritárias eram L (15 amostras), PO (12 amostras) e FT (18 amostras), as duas primeiras receberam um aumento significativo em suas amostras, respectivamente, 1286,67% e 516,67%. Cabe destacar, também, o aumento expressivo da classe MN que foi de 587,5%. A classe de maior aumento absoluto em amostras foi a classe de requisitos funcionais, no valor de 1814. Verifica-se, também, que o aumento de classes que possuíam, na base anterior, acima de 150 amostras, saiu de 1 para 7, e que a

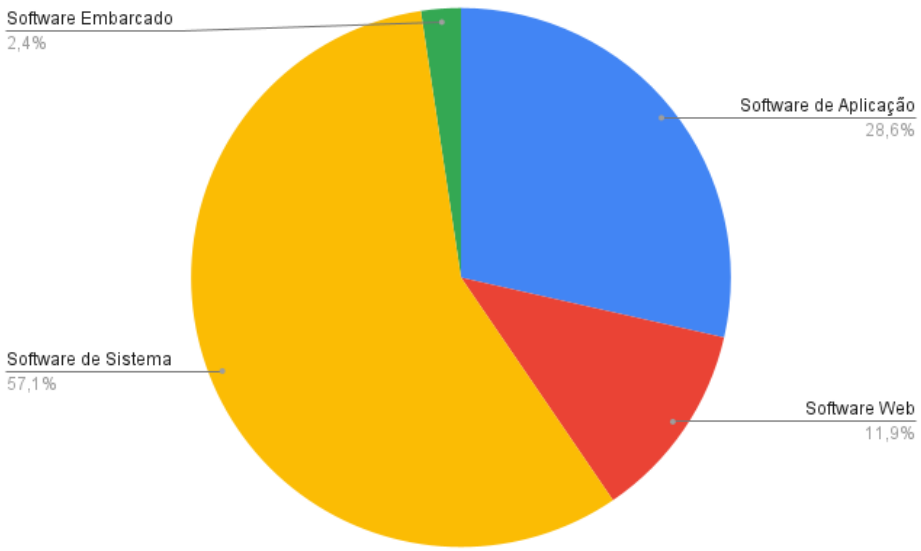


Figura 7 – Distribuição de tipos de software.

classe com menos amostras, na base anterior, saiu de 12 para 35 na Promise+. A Figura 8 apresenta uma visualização do acréscimo realizado na nova base de requisitos para os requisitos não funcionais.

Tabela 5 – Comparação quantitativa da expansão

Classes	PROMISE_exp	Promise+	Taxa de Aumento
F	444	2258	408,56%
A	31	71	129%
L	15	209	1293,33%
LF	42	102	142,85%
MN	24	75	212,5%
O	77	157	103,9%
PE	72	163	126,38%
SC	22	95	331,81%
SE	125	237	89,6%
US	85	224	163,53%
FT	18	35	94,44%
PO	12	76	533,33%
Total	969	3677	279,46%

Para a comparação entre os resultados da execução dos algoritmos de classificação em ambas as bases, foi utilizada a diferença de F1-scores entre o resultado na Promise+ e o resultado na PROMISE_exp. Portanto, um resultado positivo indica melhor execução da Promise+, do contrário, indica melhor resultado da PROMISE_exp. O *F1-score* é uma métrica que quantifica o equilíbrio entre precisão e *recall*, sendo calculada como a média harmônica destas duas métricas. *F1-score* é útil quando existe um desequilíbrio

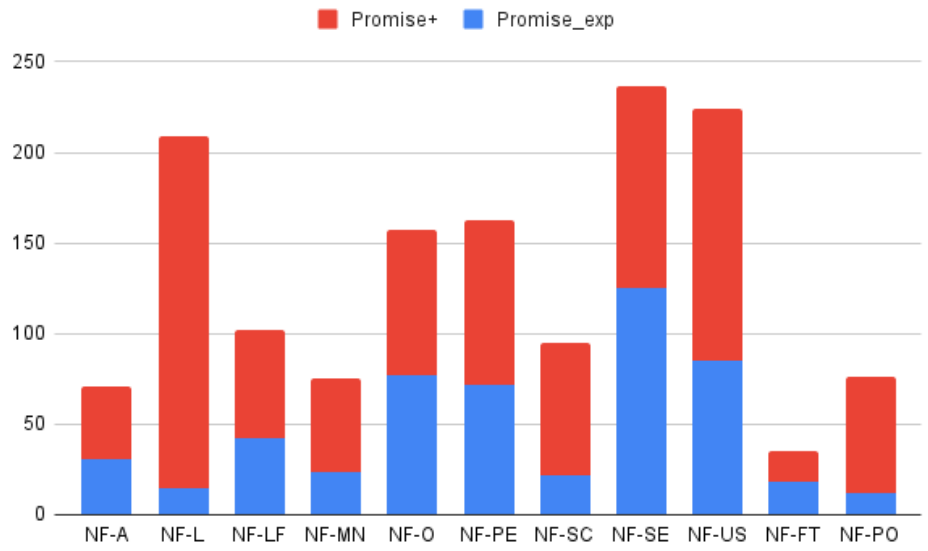


Figura 8 – Comparação entre as bases de requisitos para requisitos não-funcionais.

significativo entre as classes. No contexto do presente trabalho, esta métrica será utilizada para a comparação entre o desempenho dos algoritmos em cada *dataset*, por ser uma forma simples e direta de fazer esta comparação (GERON, 2019). Outro ponto que reforça a escolha desta métrica é que ela consegue quantificar melhor os desempenhos dos algoritmos na situação de desbalanceamento dos *datasets* (LIPTON; ELKAN; NARYANASWAMY, 2014).

Na Figura 9, tem-se o resultado da classificação binária. Para o algoritmo KNN, MNB e RL tem-se um desempenho melhor na PROMISE_exp para requisitos não funcionais, enquanto a Promise+ melhora para os funcionais. No MVS, SGD e PA o desempenho fica bem nivelado nos não funcionais, com melhora nos índices de requisito funcional. De maneira geral, a Promise+ oferta uma melhora nos índices relativos à classe funcional.

Os resultados da classificação de requisitos não funcionais em 11 classes estão na Figura 10. Para as classe US e L têm-se, de maneira geral, melhores índices na utilização da PROMISE_exp, exceto na utilização de RL e MNB. Para a classe FT no KNN e PA a execução é melhor com a PROMISE_exp, enquanto no SGD fica melhor com a Promise+. Para a classe L nos algoritmos KNN, MVS, SGD e PA ao usar a PROMISE_exp tem-se desempenho melhor, enquanto no RL e MNB tem-se desempenho melhor na Promise+. Em relação a Promise+, o desempenho é mais uniforme em todos os algoritmos. Para a classe LF, a Promise+ melhora em na maioria dos algoritmos, com exceção do KNN e PA, cujo melhor desempenho fica com a PROMISE_exp. Na classe MN, a aplicação da Promise+ resulta em um desempenho superior em todos os algoritmos. Para as classes O e SE, o desempenho é melhor com a aplicação da PROMISE_exp, com exceção dos

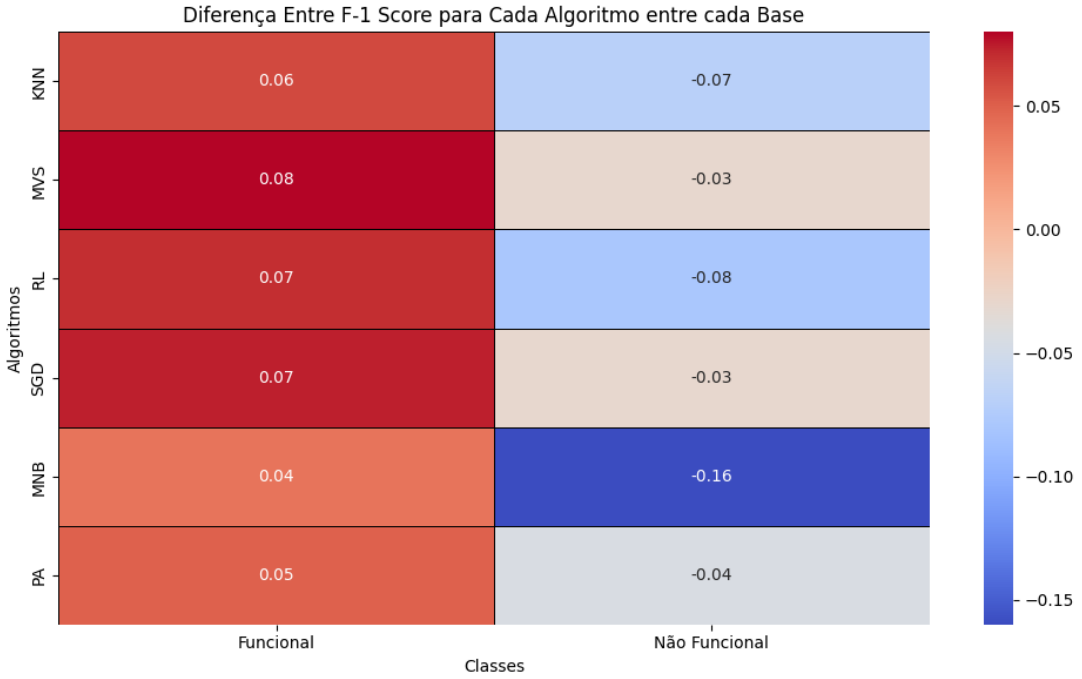


Figura 9 – Resultado da Classificação Binária.

algoritmos PA e SGD. Já para a classe PE, o desempenho com a Promise+ melhora em todos os algoritmos, com exceção dos algoritmos RL e PA. Nas classes PO e SC, o desempenho melhora com a utilização da Promise+. Para a classe US, o desempenho é melhor na PROMISE_exp, com exceção ao algoritmo RL.

O resultado da classificação com todas as 12 classes (incluindo requisitos funcionais) é apresentado na Figura 11. Para as classes F e L, a Promise+ é melhor em todas as classes, à exceção de SGD. Para a classe FT e US, em todos os algoritmos, o desempenho é de 0%. Na classe L, o desempenho na Promise+ é substancialmente maior na maioria dos algoritmos. Na classe LF tem-se um desempenho melhor na Promise+, com exceção dos algoritmos SGD e PA. Nas Classes MN e PO, ha um melhor desempenho na Promise+ nos algoritmos KNN, MVS e PA. Nas classes O, PE, SE e SC, a PROMISE_exp desempenha melhor na maioria dos algoritmos.

4.1.2 Experimento Classificação de Requisitos

Para a determinação de qual algoritmo seria o mais adequado para compor o módulo de Extração de Requisitos Funcionais, foi conduzido este experimento para verificar se classificadores baseados em Transformers podem fornecer melhora de desempenho na classificação binária de requisitos, em relação aos algoritmos de classificação tradicionais. Algoritmos de classificação baseados em aprendizagem de máquina, presentes na Seção 2.3, são amplamente utilizados na tarefa de classificação de requisitos (CANEDO; MENDES,

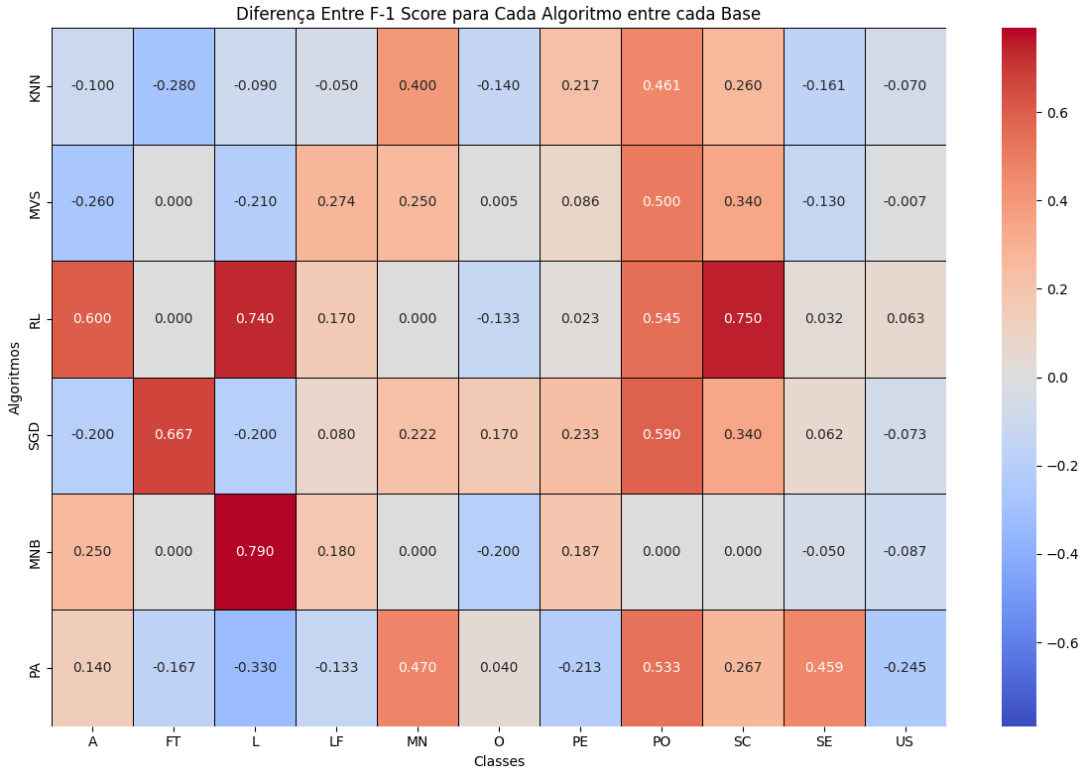


Figura 10 – Resultado da Classificação em Requisitos Não Funcionais.

2020; SILVA GEOVANE MIGUEL DA SILVA; MADEIRO, 2021).

A utilização de Transformers, contudo, cresce nas mais diversas áreas, incluindo classificação de texto (KORA; MOHAMMED, 2023). Em Singh e Kumar (2023), os autores compararam os algoritmos Naive Bayes, RL e SVM com o modelo baseado em Transformers, DistilBERT, na tarefa de análise de sentimentos em textos. Os resultados demonstram um desempenho maior do DistilBERT. Já em Kaur e Kaur (2023), os autores utilizaram uma combinação de BERT com redes neurais profundas na tarefa de classificação de requisitos. Os resultados demonstraram também superioridade de desempenho em relação aos algoritmos de classificação tradicionais.

Neste contexto, surge a necessidade de verificar se os modelos de Transformers possuem melhor desempenho para classificação de requisitos. Para poder verificar isso, um experimento foi feito para responder à seguinte questão de pesquisa:

- Questão de Pesquisa de Classificação de Requisitos (QP-CR1): Classificadores baseados em Transformers podem fornecer melhor desempenho na classificação binária de requisitos (Funcional e Não Funcional)?
  - Subquestão de pesquisa 1: Classificadores baseados em Transformers podem fornecer melhor acurácia na classificação binária de requisitos?

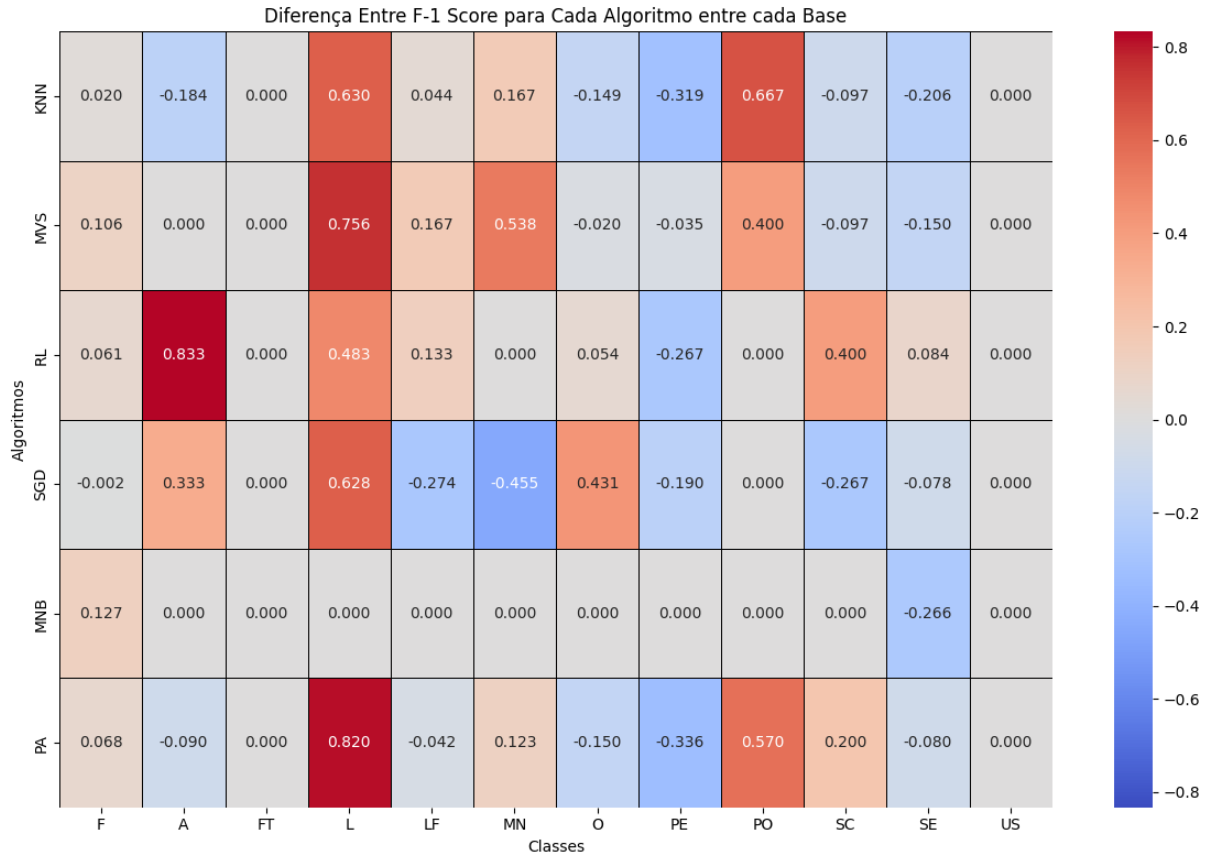


Figura 11 – Resultado da Classificação com todas as classes de requisitos.

– Subquestão de pesquisa 2: Classificadores baseados em Transformers podem fornecer melhor f1-score na classificação de Requisitos Funcionais?

– Subquestão de pesquisa 3: Classificadores baseados em Transformers podem fornecer melhor f1-score na classificação de Requisitos Não Funcionais?

Para responder à QP-CR1, a metodologia apresentada na Figura 12 foi utilizada para gerar dados para análise. Para os 10 algoritmos de classificação tradicionais, descritos na Seção 2.3, os requisitos passam por processos tradicionais de *pipelines*, como: tokenização, remoção de palavras irrelevantes, lematização e vetorização com TF-IDF. Por fim, cada classificador passa por um *Grid Search* com K-Fold, para realizar otimização de hiper parâmetros, bem como validação cruzada para maximizar o desempenho dos algoritmos. Foi utilizado K=10 no K-Fold.

Para os 10 modelos Transformers do tipo encoder e algumas variações, descritos na Seção 2.4, os requisitos passam pelo Transformer para obter sua representação vetorial, depois há um processo de ajuste fino (*fine-tunning*) no *pipeline* de classificação de texto da biblioteca HuggingFace[3]. Neste processo, a saída do encoder passa por uma camada

[3] https://huggingface.co/

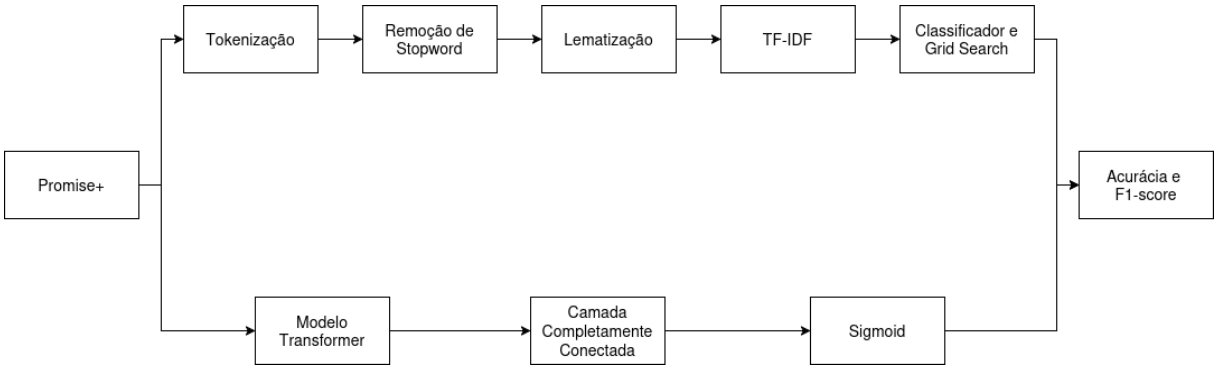


Figura 12 – Metodologia para experimento de classificação de requisitos.

completamente conectada e em seguida passa por uma camada com sigmoid, para inferir a classe. Em ambos os *pipelines*, as métricas aferidas nos modelos foram acurácia e F1-score.

Para realizar a comparação entre os dois conjuntos de classificadores, foi utilizado o teste de hipótese *Wilcoxon signed-rank test*, o qual é um teste não paramétrico usado para comparar duas amostras emparelhadas de alguma forma, neste caso, pelo uso do mesmo *dataset*. Este teste é aplicado neste caso, pois se tem uma classificação binária, com as métricas Acurácia e F1-score, com a comparação entre dois tipos de modelo, conforme Rainio, Teuho e Klén (2024). O nível de significância adotado é de 0,05 e as hipóteses são as seguintes:

- Hipótese nula (H0): Não há diferença significativa na acurácia (ou F1-score) entre os modelos Transformers e os modelos de aprendizado de máquina tradicionais.
- Hipótese alternativa (H1): Há uma diferença significativa na acurácia (ou F1-score) entre os modelos Transformers e os modelos de Aprendizado de Máquina tradicionais.

4.1.2.1 Resultados

Nas Tabelas 6 e 7 são demonstrados os resultados da execução dos classificadores para a tarefa de classificação binária entre requisitos funcionais e requisitos não funcionais. Para os algoritmos de Aprendizado de Máquina, tem-se que os de melhor acurácia são NB e SVM, tendo LR, SGD e PA vindo logo em seguida e bem próximos. A acurácia média dos modelos é 0,826. Com relação aos Requisitos Funcionais, os classificadores de melhor desempenho são NB, SVM, KNN e SGD. O F1-score médio para Requisitos Funcionais é 0,862. Por fim, para os requisitos Não Funcionais, o melhor desempenho foi com NB e SVM, com KNN, LR, SGD e PA vindo logo em seguida. O F1-score médio para Requisitos Não Funcionais é 0,763.

Para os modelos baseados em Transformers, cujos resultados estão na Tabela 7, os de melhor acurácia, F1-score Funcional e F1-score Não Funcional, são albert-large-

Tabela 6 – Resultado da classificação binária nos algoritmos de Aprendizado de Máquina.

Aprendizado de Máquina			
Modelo	Acurácia	F1-score Funcional	F1-score Não Funcional
SVM	0,866848	0,894624	0,819188
KNN	0,837840	0,883227	0,792453
LR	0,836210	0,879828	0,792593
SGD	0,838368	0,882729	0,794007
PA	0,837204	0,878788	0,795620
RF	0,799712	0,860125	0,739300
DT	0,641607	0,729258	0,553957
GB	0,810991	0,864407	0,757576
AB	0,793903	0,845987	0,741818
NB	0,872828	0,903930	0,841727

v2, distilbert-base-uncased e bert-base-multilingual-uncased. A acurácia média entre os modelos é 0,8845. O F1-score médio para Requisitos Funcionais é 0,904. O F1-score médio para Requisitos Não Funcionais é 0,856. Comparando as 20 execuções, os melhores desempenhos de acurácia, F1-score Funcional e F1-score Não Funcional são, em ordem, bert-base-multilingual-uncased, albert-large-v2 e distilbert-base-uncased.

Tabela 7 – Resultado da classificação binária nos modelos baseados em Transformers

Transformers			
Modelo	Acurácia	F1-score Funcional	F1-score Não Funcional
bert-base-uncased	0,883152	0,901602	0,856187
bert-large-uncased	0,883152	0,897375	0,864353
roberta-base	0,877717	0,896074	0,851485
distilbert-base-uncased	0,896739	0,912844	0,873333
albert-base-v2	0,885870	0,905405	0,856164
albert-large-v2	0,894022	0,913140	0,864111
xlm-roberta-base	0,866848	0,891832	0,826855
bert-base-multilingual-cased	0,872283	0,896247	0,833922
bert-base-multilingual-uncased	0,896739	0,914027	0,870748
bart-base	0,888587	0,907449	0,860068

O resultado da aplicação do teste de hipótese de Wilcoxon é apresentado na Tabela 8, onde estão descritos os valores de *p-value* obtidos para cada métrica analisada. Dado que o nível de significância adotado para o teste foi de 0,05, observa-se na tabela que todos os p-values são inferiores a esse limite. Isso permite rejeitar a hipótese nula, que pressupõe a inexistência de diferença significativa entre os modelos Transformers e os de Aprendizado de Máquina Tradicionais. Assim, conclui-se haver evidências estatísticas suficientes para afirmar a existência de diferenças significativas em todas as métricas analisadas.

A Figura 13 demonstra essas diferenças por meio de um diagrama de caixa, que apresenta a distribuição das métricas para os dois grupos de modelos. Nessa visualização,

Tabela 8 – Resultado do teste de hipótese.

Teste de Hipótese	
Métrica	p_value
Acurácia	0,000926
F1-score Funcional	0,000996
F1-score Não Funcional	0,000876

nota-se a superioridade dos modelos Transformers, que apresentam não apenas melhores valores médios, mas também menor variabilidade em diversas métricas. Isso reforça a robustez desses modelos quando comparados aos métodos tradicionais de Aprendizado de Máquina.

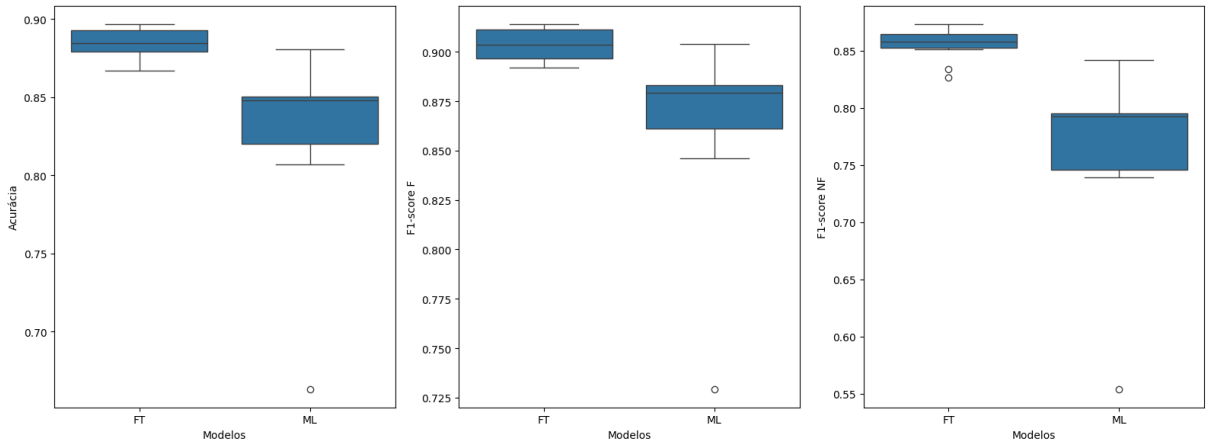


Figura 13 – Comparação da distribuição das métricas nos modelos.

(QP-CR1) Classificadores baseados em Transformers podem fornecer melhor desempenho na classificação binária de requisitos (Funcional e Não Funcional)? Os resultados indicam que classificadores baseados em Transformers superam significativamente os algoritmos tradicionais de Aprendizado de Máquina na classificação binária de requisitos, com média de acurácia de 0,8845 contra 0,826 dos modelos tradicionais. Além disso, os Transformers obtiveram F1-score superior tanto para requisitos funcionais (0,904 vs. 0,862) quanto não funcionais (0,856 vs. 0,763). O teste de hipótese de Wilcoxon confirmou diferenças estatisticamente significativas, reforçando a eficácia dos Transformers nessa tarefa.

4.1.3 Discussão

Com relação a expansão da base de requisitos Promise+, mais da metade dos documentos de requisitos de software aceitos são de projetos da indústria, o que por sua vez aproxima os resultados obtidos ao ambiente real de produção. Salienta-se que mesmo os projetos acadêmicos também representam projetos de software. Em ambos os casos, são

documentos estruturados para garantir o desenvolvimento de um software. Além disso, ao considerar estes projetos, aumenta-se o quantitativo de dados disponíveis, uma vez que há a carência deste tipo de dado publicamente na indústria.

Cabe destacar também que a maior concentração de documentos está entre os anos de 2005 a 2010. Ressalta-se que, apesar do tempo, tais documentos continuam relevantes, pois os requisitos de software não possuem prazo de validade. Os requisitos não estão vinculados a uma tecnologia específica, mas sim às necessidades dos clientes/usuários do sistema, mantendo sua finalidade de especificação, principalmente os requisitos funcionais. Outro fator que justifica o uso desses documentos é a escassez de documentos de requisitos públicos.

Em termos quantitativos, a expansão realizada foi alta em comparação com a PROMISE_exp. Neste trabalho, o aumento absoluto foi de 2255 novas instâncias, representando 6,5 vezes mais do que foi realizado na primeira expansão. Todas as classes receberam aumento relativo de, no mínimo, 86%, chegando até 1286,67%, como o ocorrido com a classe L. A expansão contribui na promoção ao acesso de mais informações sobre requisitos e, portanto, diversidade na base de dados. A diversidade promove convergência em algoritmos de aprendizado de máquina ao disponibilizar um número maior de exemplos com maior diversidade. Esses fatores contribuem para a redução de *overfiting* e capacidade de generalização dos algoritmos (GONG; ZHONG; HU, 2019).Contudo, como o maior aumento absoluto foi na classe F (1339 instâncias) o desbalanceamento, que já era presente, se intensificou. Portanto, trabalhos futuros que utilizem a Promise+ devem observar tal desbalanceamento.

Na etapa de classificação manual, a expansão da Promise+ evidencia alguns problemas como escrita dos requisitos e classes diferentes das classes originais da PROMISE. Para o problema da escrita, isso pode impactar tanto na rotulação manual quanto na utilização dos algoritmos de classificação. Para o problema de classes diferentes, isso pode indicar que uma nova avaliação pode ser feita futuramente, para que todos os requisitos aqui encontrados. Um dos objetivos desta nova avaliação pode ser rotular novas classes e, assim, ter um *dataset* mais granular e mais diverso em relação ao tipo de requisitos não-funcionais. Ao final da classificação manual, a validação conduzida com o índice Kappa demonstrou-se bem aceitável, com índices de concordância majoritariamente substanciais.

O Teorema *No Free Lunch* estabelece não haver um método de aprendizado de máquina que possa ser considerado superior a todos os outros para resolver todos os problemas existentes (WOLPERT; MACREADY, 1997). Sendo assim, um determinado método pode ser superior a outro dependendo do ajuste que se faz no treinamento. Portanto, a *baseline* utilizado a Promise+ foi a base PROMISE_exp. Optou-se também por instanciar os classificadores sob as mesmas condições de hiper parâmetros para tentar diminuir fatores extras na análise e comparar somente as bases. Contudo, dado que o

TF-IDF foi utilizado como método de vetorização, a presença de mais dados resulta em vetores de maior dimensão. Na Promise+, os vetores possuem mais dimensões e isso pode impactar negativamente nos resultados dos classificadores.

Para a classificação binária, em todos os algoritmos, a Promise+ fornece melhora no desempenho dos índices para a classe F. Este resultado deve-se a mudança de classe majoritária, que na PROMISE_exp era a NF, e na Promise+ passou a ser a classe F. Vale destacar que o algoritmo MVS é o que melhor desempenha para ambas as classes (NF e F).

Para a classificação dos não-funcionais em 11 classes, a Promise+ traz melhores resultados para a maioria das classes. Vale destacar que nas classes MN e PO, a Promise+ representou sair do rendimento 0% para ter rendimento razoável. Vale destacar que os algoritmos SGD e MVS são os que possuem melhor desempenho para as classes MN e PO.

Por fim, na classificação em 12 classes (com todas as classes da base), a PROMISE_exp desempenha melhor em três dos seis algoritmos, enquanto a Promise+ lidera em dois algoritmos. Acredita-se que tal rendimento da PROMISE_exp se deva por ser um *dataset* menos desbalanceado do que a Promise+. Vale destacar que esta classificação é a mais afetada pelo desbalanceamento, mas a diferença não é substancial, conforme relatado. Entretanto, em trabalhos futuros, deve-se utilizar métodos que compensem tal desbalanceamento para uma melhor qualidade dos resultados.

Com relação ao experimento de classificação de requisitos conduzidos, os modelos baseados em Transformers demonstraram desempenho superior aos algoritmos de classificação tradicionais na tarefa de classificação binária de requisitos, tanto para Requisitos Funcionais quanto para Não Funcionais. A acurácia média dos modelos Transformers foi maior, 0,8845 contra 0,826, em relação aos modelos tradicionais, além de apresentarem melhores valores médios de F1-score para ambos os tipos de requisitos, sendo Funcionais 0,904 para Transformers contra 0,862 para modelos tradicionais e Não Funcionais 0,856 para Transformers contra 0,763 para modelos tradicionais.

A aplicação do teste de Wilcoxon confirmou que as diferenças entre os modelos Transformers e os algoritmos de Aprendizado de Máquina tradicionais são estatisticamente significativas, rejeitando a hipótese nula (H0). Assim, conclui-se que os Transformers são uma escolha mais adequada para compor o módulo de Extração de Requisitos Funcionais. Os melhores desempenhos em acurácia, F1-score Funcional e F1-score Não Funcional, foram alcançados pelos modelos Transformers, especificamente albert-large-v2, distilbert-base-uncased e bert-base-multilingual-uncased. Sendo o modelo bert-base-multilingual-uncased utilizado por ser o que possui o melhor F1-score para identificação de requisitos funcionais (0,9140).

Como implicações para a pesquisa, tem-se que a Promise+ pode incentivar a

utilização de algoritmos de aprendizado de máquina para classificação de requisitos. Pode também ser utilizada para validar e comparar métodos já existentes de classificação de requisitos, amenizando a ameaça à validade do problema da quantidade de amostras (CANEDO; MENDES, 2020; SILVA GEOVANE MIGUEL DA SILVA; MADEIRO, 2021; QUBA et al., 2021). Por fim, o experimento utilizando Transformers, reforça a validade dessa abordagem para a tarefa de classificação de requisitos e abre caminho para novas pesquisas que explorem o uso de transformers em outras atividades de Engenharia de requisitos, como priorização de Requisitos e detecção de ambiguidades.

Como implicações para a prática de classificação e análise de requisitos, tem-se que a Promise+ pode auxiliar na melhoria da precisão de modelos de classificação. Tal melhoria foi demonstrada no experimento de classificação, com os modelos baseados em Transformers atingindo F1-score de 0,914. Esta maior acurária e F1-score indicam que os Transformers podem reduzir erros de classificação, o que é crucial para a automação de tarefas de Engenharia de Requisitos. Isso pode levar a uma redução de custos e tempo no processo de desenvolvimento de software, além de aumentar a confiança na automação. Embora os Transformers tenham demonstrado superioridade, tais modelos geralmente exigem mais recursos computacionais em comparação com algoritmos tradicionais. Portanto, é necessário considerar a compensação entre desempenho e custo computacional ao adotar esses modelos em projetos reais.

4.2 Computação de Similaridade Semântica

Conforme relatado na Seção 2.5, os algoritmos de agrupamento hierárquico tem a necessidade da formação de uma matriz de similaridade ou dissimilaridade, como entrada. Este módulo da ReqCluster4IoT é necessário, portanto, para receber os requisitos funcionais, convertê-los para vetores numéricos e computar a similaridade entre cada requisito par a par. Neste estudo, requisitos que descrevem elementos de uma mesma funcionalidade de software foram analisados de forma semelhante em termos de semântica.

Como discutido na Seção 3, a maioria dos trabalhos em agrupamento de requisitos utiliza como *word embedding* métodos baseados em atributos ou usa TF-IDF. Porém, este tipo gera vetores esparsos e com baixo poder semântico. Neste contexto, modelos baseados em Transformers conseguem capturar dependências de longo alcance e geram vetores de tamanho fixo, presente na Seção 2.4. Portanto, para a ReqCluster4IoT foram investigados se tais modelos conseguem gerar representações significativas para servir de entrada para a computação da similaridade.

Modelos baseados em Transformers podem ser pré-treinados para uma determinada tarefa específica, no contexto deste trabalho são interessantes modelos treinados em computação de similaridade semântica. Estes modelos identificados podem ainda passar por

ajuste fino para o contexto de requisitos de software. Uma abordagem que pode ser utilizada para realizar este ajuste é utilizando redes siamesas. O conceito de redes siamesas foi inicialmente apresentado em Bromley et al. (1993), com ampla utilização e processamento de imagem (MELEKHOV; KANNALA; RAHTU, 2016; FEDELE; GUIDOTTI; PEDRESCHI, 2024), porém, existem aplicações em texto (RANASINGHE; ORASAN; MITKOV, 2019; LEI; MENG, 2022).

Redes siamesas são um tipo de arquitetura de rede neural na qual duas redes neurais compartilham os mesmos pesos e parâmetros. São utilizadas para aprender representações discriminativas para medir a similaridade ou diferença entre entradas (SCHROFF; KALENICHENKO; PHILBIN, 2015). A rede siamesa com função de perda tripla, representada na Figura 14, é uma variação que possui três entradas: (1) imagem âncora, a qual é o ponto de partida da comparação; (2) imagem positiva, a qual é um exemplo semelhante à âncora e (3) imagem negativo, um exemplo diferente da âncora.

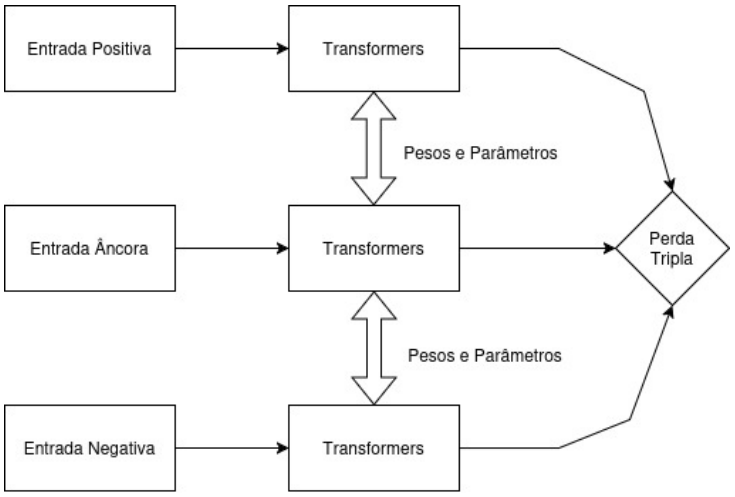


Figura 14 – Rede Siamesa com Função de Perda Tripla.

A função de perda tripla é dada pela seguinte equação (SCHROFF; KALENICHENKO; PHILBIN, 2015):

$$L = \max(0, d(a,p) - d(a,n) + \alpha)$$

Sendo:

- d(a,p): Distância entre o embedding da âncora e o positivo.
- d(a,n): Distância entre o embedding da âncora e o negativo.
- α : Margem, um hiper parâmetro que garante que o negativo esteja a uma distância suficiente da âncora.

A rede tenta minimizar a distância entre a âncora e o positivo, enquanto maximiza a distância entre a âncora e o negativo, com um espaçamento de pelo menos α. Neste sentido, os modelos de Transformers têm parte de seus pesos ajustados para se adequar ao contexto do treinamento.

Um ponto importante neste tipo de treinamento é a escolha dos tripletos (Âncora, Positivo e Negativo). Segundo Schroff, Kalenichenko e Philbin (2015), existem duas formas: (1) geração offline, na qual os tripletos são gerados antes do treinamento, sendo salvos como um conjunto de dados; (2) geração online, na qual os tripletos são gerados durante o treinamento, geralmente com base nas representações intermediárias aprendidas pela rede. A biblioteca *Sentence Transformers* [4] possui as seguintes opções de computação da função de perda a partir de uma determinada formação de tripletos:

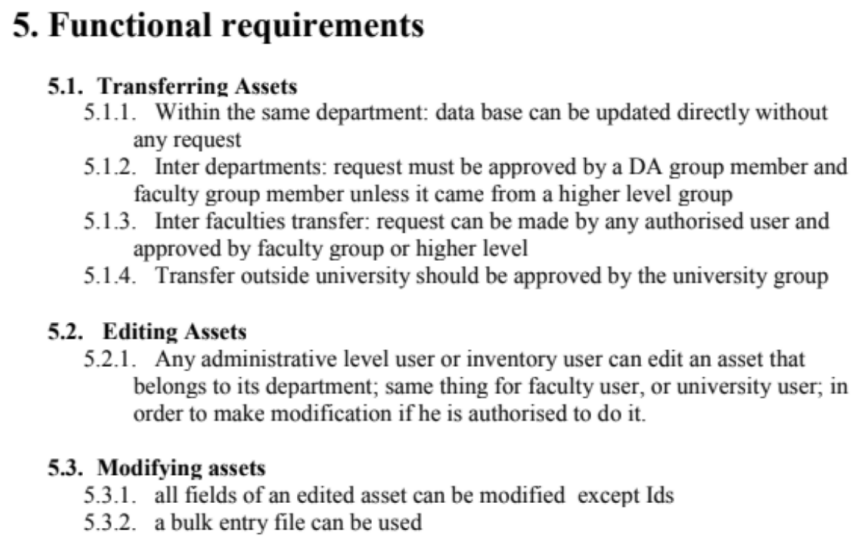
- BatchAllTripletLoss: computa todas as possíveis combinações de tripletos dentro de um determinado conjunto. É aproximado da geração offline. A função de perda permanece a mesma;
- BatchHardTripletLoss: seleciona apenas os tripletos difíceis utilizando uma margem fixa. Para cada registro, este método seleciona o exemplo positivo mais distante (dentro da mesma classe) e seleciona o exemplo negativo mais próximo (fora da classe). A função de perda permanece a mesma;
- BatchHardSoftMarginTripletLoss: foca também nos tripletos difíceis, porém utiliza a função de perda com margem suave. A margem suave evita instabilidades ao lidar com exemplos outliers extremos. Segue a seguinte fórmula: $\log\left(1 + \exp\left(d(a,p) - d(a,n)\right)\right)$;
- BatchSemiHardTripletLoss: considera os tripletos nos quais o exemplo negativo é mais distante que o positivo, mas ainda próximo o suficiente para contribuir com o aprendizado. Seleciona negativos cuja distância seja maior que a do positivo, mas menor que d(a,p)+α: $d(a,p) < d(a,n) < d(a,p) + \alpha$. A função de perda permanece a mesma.

Neste contexto, surge a necessidade da criação de um conjunto de dados de requisitos de diferentes funcionalidades, cujos requisitos de mesma funcionalidade sejam considerados de mesma classe. De modo que possa ser utilizado no ajuste dos Transformers. Para isso, foi gerado o ReqFuncSimDataset, um conjunto de dados que contém requisitos de software obtidos de diferentes SRSs.

Para a formação do conjunto de dados ReqFuncSimDataset seriam necessários documentos de requisitos que elicitassem suas funcionalidades como um conjunto de requisitos funcionais. Os documentos utilizados são os mesmos utilizados na Promise+,

[4] https://sbert.net/index.html

no caso o PURE Dataset (FERRARI; SPAGNOLO; GNESI, 2017). Na formação do ReqFuncSimDataset foram considerados somente os documentos de requisitos que seguissem a estrutura presente na Figura 15. Dos 79 documentos, somente 13 elicitam seus requisitos da forma apropriada. Foram extraídos 587 requisitos, divididos em 59 funcionalidades. O ReqFuncSimDataset não foi avaliado antes de sua aplicação contudo, os requisitos contidos no dataset PURE, para além de terem sido utilizados na Promise+ (SILVA et al., 2024), já foram utilizados em outros trabalhos (SHREDA; HANANI, 2021; RAHMAN et al., 2023; TALELE et al., 2023). O ReqFuncSimDataset está disponível em um repositório do zenodo [5]



5. Functional requirements

5.1. Transferring Assets

5.1.1. Within the same department: data base can be updated directly without any request

5.1.2. Inter departments: request must be approved by a DA group member and faculty group member unless it came from a higher level group

5.1.3. Inter faculties transfer: request can be made by any authorised user and approved by faculty group or higher level

5.1.4. Transfer outside university should be approved by the university group

5.2. Editing Assets

5.2.1. Any administrative level user or inventory user can edit an asset that belongs to its department; same thing for faculty user, or university user; in order to make modification if he is authorised to do it.

5.3. Modifying assets

5.3.1. all fields of an edited asset can be modified except Ids

5.3.2. a bulk entry file can be used

Figura 15 – Exemplo de documento cujos requisitos funcionais foram elicitados como componentes de uma funcionalidade. Fonte (FERRARI; SPAGNOLO; GNESI, 2017)

Uma vez gerado o conjunto de dados, foi conduzido um experimento para verificar se o treinamento com o ReqFuncSimDataset fornece melhora na similaridade de requisitos de mesma funcionalidade e, também, verificar se modelos baseados em Transformers conseguem computar a similaridade melhor do que métodos clássicos como o TF-IDF.

4.2.1 Experimento Similaridade

Para a definição de qual modelo seria o mais adequado para compor o módulo de similaridade semântica, foi conduzido o presente experimento visando avaliar se o tratamento na base de dados ReqFuncSimDataset fornece melhora na computação de similaridade de requisitos de mesma funcionalidade e, outrossim, verificar se os modelos baseados em Transformers conseguem computar a similaridade melhor do que o método tradicional com TF-IDF. Para atingir estes objetivos, as seguintes questões de pesquisa para o experimento foram formuladas:

[5] https://zenodo.org/records/14883596?token=eyJhbGciOiJIUzUxMiJ9.eyJpZCI6IjhhNDJmMzdjLTJjNzYtNGUyYS1

- Questão de Pesquisa de Similaridade de Requisitos 1(QP-SR1): O treinamento na base de dados ReqFuncSimDataset melhora a capacidade dos modelos de aprender e identificar similaridade entre requisitos que pertencem à mesma funcionalidade em comparação com modelos não treinados?
- Questão de Pesquisa de Similaridade de Requisitos 2 (QP-SR2): Modelos baseados em Transformers apresentam melhor desempenho na computação de similaridade entre requisitos funcionais em comparação ao método tradicional baseado em TF-IDF?

Para responder a QP-SR1, foram escolhidos 10 modelos disponíveis na biblioteca *Sentence Transformers* que são pré-treinados utilizando diversos datasets para computar similaridade semântica entre duas sentenças[6]. Estes modelos foram utilizados na sua versão padrão, sendo somente carregados usando a biblioteca. Os modelos são: *"all-MiniLM-L6-v2","all-mpnet-base-v2", "paraphrase-multilingual-mpnet-base-v2","paraphrase-multilingual-MiniLM-L12-v2","LaBSE","bert-base-nli-mean-tokens","distilbert-base-nli-mean-tokens","multi-qa -MiniLM-L6-cos-v1","msmarco-MiniLM-L-12-v3" e "paraphrase-TinyBERT-L6-v2"*. Quando este experimento foi realizado, em outubro de 2024, esses modelos apresentavam o melhor desempenho médio em *benchmarks* de *embedding* de sentença e busca semântica, conforme os resultados disponíveis na página do *Sentence Transformers*.

Estes modelos passaram pelo treinamento utilizando redes siamesas com função de perda tripla. Essa abordagem de rede siamesa, além de ser usada com frequência em similaridade de imagens (MELEKHOV; KANNALA; RAHTU, 2016; FEDELE; GUIDOTTI; PEDRESCHI, 2024), já foi usada para detectar similaridade de requisitos. Em Alnajem, Binkhonain e Hossain (2024), as redes siamesas foram usadas para detectar similaridade entre pares de requisitos. Diferentemente do contexto deste trabalho, que entende requisitos semanticamente similares quando dois requisitos descrevem elementos de uma mesma funcionalidade de software, o trabalho de Alnajem, Binkhonain e Hossain (2024) entende como similares requisitos que podem ser substituídos um pelo outro. A utilização de redes siamesas demonstrou-se promissora e com resultados superando os métodos estado da arte.

A formação dos tripletos seguiu as quatro possibilidades (BatchAllTripletLoss, BatchHardTripletLoss, BatchHardSoftMarginTripletLoss e BatchSemiHardTripletLoss), com exceção do SoftMargin, os outros modelos tiveram a margem variando entre 1, 2 e 3, resultando então em 10 variações para cada modelo. O número de épocas ficou em 800 e o tamanho de *batch* em 64. Para a comparação entre cada modelo treinado e sua versão base, os dois modelos foram aplicados sobre um conjunto de teste com 23 funcionalidades e 85 requisitos. Neste conjunto de teste foram montadas todas as combinações de pares, sendo similares os requisitos de mesma funcionalidade e não similares os requisitos de funcionalidades diferentes. Foi comparada então a diferença média entre os dois modelos,

[6] https://www.sbert.net/docs/sentence_transformer/pretrained_models.html

para verificar se o modelo treinado consegue aumentar a diferença entre seus registros similares e os não similares.

Para responder à QP-SR2, foram selecionados os 10 melhores modelos treinados e comparou-se com o método utilizando o TF-IDF, a partir do conjunto de teste descrito anteriormente. Foram considerados melhores modelos aqueles com maior diferença entre a média para pares similares e a média dos pares não similares. Sendo assim, os melhores modelos são aqueles que mais conseguem diferenciar pares similares de não similares. Em ambos os casos, foi utilizada a similaridade de cosseno para computar a similaridade.

4.2.1.1 Resultados

O resultado para a comparação entre o modelo pré-treinado e o ajustado, resultado da QP-SR1, está expresso na Figura 16. Na figura, tem-se a plotagem da diferença entre as diferenças de similaridade para os similares e não similares entre o modelo treinado e o pré-treinado original. Um registro positivo indica que o modelo treinado consegue diferenciar melhor similares e não similares, do contrário, o treinamento piorou esta diferenciação. Das 100 variações de modelos treinados, em 60 o treinamento melhora e em 40 o treinamento piora.

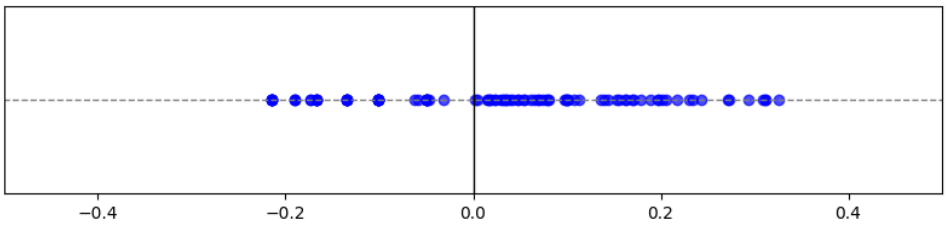


Figura 16 – Gráfico da diferença entre os modelos originais e os modelos treinados.

A Tabela 9 apresenta os 10 modelos com maior aumento de diferença. quando utilizada a similaridade de cosseno. Percebe-se que o modelo base *bert-base-nli-mean-tokens* é o mais beneficiado pelo treinamento, registrando 7 dos 10 maiores aumentos. Outros modelos também beneficiados são: *paraphrase-multilingual-MiniLM-L12-v2*, *all-MiniLM-L6-v2* e *paraphrase-multilingual-mpnet-base-v2*.

O resultado para a comparação entre o modelo pré-treinado e a computação por TF-IDF, resultado da QP-SR2, consta na Figura 16. Nesta figura, tem-se a plotagem da diferença entre as diferenças de similaridade para os similares e não similares entre o modelo treinado e o modelo baseado em TF-IDF. Um registro positivo indica que o modelo treinado consegue diferenciar melhor similares e não similares, do contrário, o TF-IDF é melhor na diferenciação. Das 100 variações de modelos treinados, em 60 o treinamento melhora e em 40 o treinamento piora.

Tabela 9 – Aumento de diferença entre similares e não similares após treinamento.

Modelo	Processo de Treinamento	Diferença no Treinado	Diferença no Pré-treinado	Aumento de Diferença
bert-base-nli-mean-tokens	hard-margin-3	0,375969	0,049699	0,326270
bert-base-nli-mean-tokens	hard-margin-2	0,361864	0,049699	0,312164
bert-base-nli-mean-tokens	soft-margin	0,361775	0,049699	0,312076
bert-base-nli-mean-tokens	hard-margin-1	0,375969	0,049699	0,309472
bert-base-nli-mean-tokens	semihard-margin-1	0,342682	0,049699	0,292983
bert-base-nli-mean-tokens	semihard-margin-2	0,322005	0,049699	0,272306
bert-base-nli-mean-tokens	semihard-margin-3	0,320766	0,049699	0,271067
paraphrase-multilingual-MiniLM-L12-v2	hard-margin-3	0,416379	0,172985	0,243394
all-MiniLM-L6-v2	all-triples-margin-2	0,434282	0,199824	0,234458
paraphrase-multilingual-mpnet-base-v2	hard-margin-1	0,397529	0,166790	0,230739

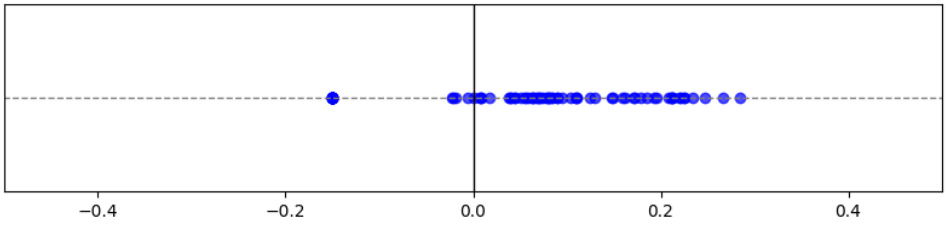


Figura 17 – Gráfico da diferença entre os modelos originais e a computação por TF-IDF.

A Tabela 10 apresenta os 10 modelos com maior aumento de diferença. A computação por TF-IDF é constante na tabela. Percebe-se que os modelos-base *paraphrase-multilingual-mpnet-base-v2* e *paraphrase-multilingual-MiniLM-L12-v2* após treinamento são os que mais têm vantagem em relação ao TF-IDF. Outros modelos também beneficiados são: *all-MiniLM-L6-v2* e *bert-base-nli-mean-token*. O modelo de maior diferença e, portanto, o escolhido para a computação de similaridade na ReqCluster4IoT é o *all-MiniLM-L6-v2-all_triplets-margin-2*.

Tabela 10 – Aumento de diferença entre similares e não similares com a computação por TFIDF.

Modelo	Processo de Treinamento	Diferença no Treinado	Diferença no TF-IDF	Aumento de Diferença
all-MiniLM-L6-v2	all_triplets-margin-2	0,434282	0,150205	0,284077
paraphrase-multilingual-MiniLM-L12	all_triplets-margin-3	0,416379	0,150205	0,266174
paraphrase-multilingual-mpnet-base-v2	hard-margin-1	0,397529	0,150205	0,247324
paraphrase-multilingual-mpnet-base-v2	hard-margin-2	0,384335	0,150205	0,234130
bert-base-nli-mean-tokens	hard-margin-3	0,375969	0,150205	0,225764
paraphrase-multilingual-MiniLM-L12	all_triplets-margin-2	0,375434	0,150205	0,225230
paraphrase-multilingual-mpnet-base-v2	soft-margin	0,372064	0,150205	0,221859
paraphrase-multilingual-MiniLM-L12-v2	semihard-margin-1	0,370440	0,150205	0,220236
paraphrase-multilingual-mpnet-base-v2	hard-margin-3	0,362983	0,150205	0,212778
paraphrase-multilingual-MiniLM-L12	hard-margin-3	0,362370	0,150205	0,212166

(QP-SR1): O treinamento na base de dados ReqFuncSimDataset melhora a capacidade dos modelos de aprender e identificar similaridade entre requisitos que pertencem à mesma funcionalidade em comparação com modelos não

treinados? Os resultados demonstram que o treinamento utilizando a ReqFuncSimDataset contribui para a melhoria da capacidade dos modelos em identificar similaridade entre requisitos que pertencem à mesma funcionalidade. Como ilustrado na Figura 16, o treinamento resultou em uma diferença positiva entre a capacidade de diferenciação de requisitos similares e não similares em relação ao modelo pré-treinado. Em 60 das 100 variações testadas, o modelo treinado demonstrou uma melhoria na capacidade de distinguir requisitos similares de não similares. A Tabela 9 revela que modelos como o bert-base-nli-mean-tokens se beneficiaram consideravelmente do treinamento, com aumentos substanciais na diferenciação entre requisitos similares e não similares.

(QP-SR2): Modelos baseados em Transformers apresentam melhor desempenho na computação de similaridade entre requisitos funcionais em comparação ao método tradicional baseado em TF-IDF? Os resultados indicam que os modelos baseados em Transformers superam o método tradicional de TF-IDF na computação de similaridade entre requisitos funcionais. A comparação entre os modelos ajustados e a computação por TF-IDF, conforme mostrado na Figura 17, revela que os modelos baseados em Transformers, após o treinamento, conseguem melhorar a diferenciação entre requisitos similares e não similares de forma mais significativa do que o TF-IDF. A Tabela 9 corrobora esse achado, com modelos como o all-MiniLM-L6-v2 e o paraphrase-multilingual-MiniLM-L12-v2 apresentando aumentos consideráveis na diferença de similaridade após o treinamento, superando a abordagem baseada em TF-IDF.

4.2.2 Discussão

Os resultados do experimento mostram que o treinamento com a base de dados ReqFuncSimDataset pode melhorar a capacidade dos modelos de aprenderem a identificar e diferenciar requisitos da mesma funcionalidade. Mesmo com nem todos os modelos treinados apresentarem melhorias, cerca de 60%, observa-se uma tendência geral de que os modelos ajustados conseguem aumentar a capacidade de diferenciação entre requisitos semelhantes e não semelhantes. Este resultado sugere que a utilização de um conjunto de dados específico, como o ReqFuncSimDataset, pode proporcionar um aprendizado mais específico às particularidades dos requisitos de software, gerando melhores representações semânticas geradas pelos modelos.

Além disso, quando comparados com o método tradicional baseado em TF-IDF, os modelos baseados em Transformers ajustados mostraram melhor desempenho na computação da similaridade semântica entre requisitos funcionais. Analisando os resultados, ficou evidente que, em sua maioria (60%), os modelos ajustados conseguem estabelecer melhores distinções entre requisitos que pertencem à mesma funcionalidade e aqueles que pertencem a funcionalidades diferentes. Isso indica o potencial dos modelos de Transformers, aliados ao treinamento específico, para superar abordagens clássicas e

melhorar a qualidade da análise de similaridade semântica em contextos de requisitos de software. O modelo escolhido para a computação de similaridade na ReqCluster4IoT é o *all-MiniLM-L6-v2-all_triplets-margin-2*.

4.3 Agrupamento de Requisitos

A terceira etapa do método ReqCluster4IoT, presente na Figura 4, é a formação dos grupos de requisitos. Para tal, foi utilizado o algoritmo de agrupamento hierárquico aglomerativo. O processo segue o seguinte fluxo, sendo um processo adaptado de Salman et al. (2018) e Kochbati et al. (2021):

- Fase 1: a matriz de dissimilaridade é obtida do módulo de similaridade semântica;
- Fase 2: é calculada a matriz de *linkage*, que recebe a matriz de dissimilaridade e retorna à representação hierárquica do agrupamento;
- Fase 3: são encontrados os pontos candidatos a corte no dendrograma para a determinação da quantidade de grupos;
- Fase 4: é encontrado o número de grupos mais adequado, considerando um índice de qualidade de agrupamento.

Para o cálculo da matriz de *linkage* foi utilizada a biblioteca scipy[7]. Este algoritmo começa tratando cada elemento como um grupo individual e, em cada iteração, combina os dois grupos mais próximos de acordo com um método específico (*single*, *complete*, *average* ou*ward*). Após a fusão, o algoritmo atualiza a matriz de distâncias para refletir as relações do novo grupo e os demais. A matriz de retorno tem em cada linha a representação da fusão de dois grupos e o ponto de fusão (distância entre os grupos). Esta matriz pode ser utilizada para construir dendrogramas que permitem visualizar a relação entre os grupos.

Na etapa 3, a partir da matriz de *linkage*, são obtidos um conjunto de valores de candidatos ao número ideal de grupos, ao invés de testar todas as possibilidades de grupos. Para tal, foi construído um método, demonstrado na Figura 18, que se baseia na maior diferença entre pontos consecutivos de fusão. Esta abordagem é útil, pois quando há uma grande diferença, isso indica haver uma separação significativa entre os grupos. O método então ordena as diferenças e escolhe, por padrão, as 10 maiores diferenças, que se tornam pontos de corte para os dendrogramas. Por fim, é verificado qual a quantidade de grupos presentes para cada ponto candidato.

[7] https://scipy.org/

```
def encontrar_clusters_por_distancia(dados, Z, num_candidatos=10):
    distancias = Z[:, 2]
    diferenciais = np.diff(distancias)
    índices_ordenados = np.argsort(diferenciais)[::-1]
    top_n = num_candidatos
    maiores_gaps_indices = indices_ordenados[:top_n]
    num_clusters = [
        len(np.unique(fcluster(Z, t=distancias[indice]+0.0001,
        criterion='distance')))
        for indice in maiores_gaps_indices]
    return num_clusters
```

Figura 18 – Função que encontra o número de clusters.

Na etapa 4, para cada quantidade de grupos candidata, são realizados os agrupamentos e retirado o índice de silhouette para determinar qual o melhor agrupamento. O agrupamento de maior índice de silhouette é então escolhido. O índice de silhouette mede o quão bem cada objeto está relacionado ao grupo ao qual ele pertence em comparação com outros grupos. Esta métrica varia de -1 a 1, na qual: 1 indica que o ponto está bem dentro do seu grupo e longe de outros grupos; 0 indica que o ponto está na fronteira entre grupos; e -1 indica que o ponto pode estar no grupo errado, ao estar mais próximo de outro cluster do que do seu próprio.

O cálculo do índice de silhouette para um ponto i envolve dois componentes principais: coesão (a(i)): É a distância média entre o ponto i e todos os outros pontos do mesmo grupo e separação (b(i)): É a distância média entre o ponto i e os pontos do grupo mais próximo. O índice de silhouette geral do agrupamento é a média dos scores de todos os pontos. Para cada ponto, o índice é conforme a seguinte equação:

$$s(i) = \frac{b(i) - a(i)}{\max(a(i), b(i))}$$

4.4 Identificação de Características de IoT

No módulo de Identificação de característica, foi utilizada uma abordagem de Modelagem de Tópicos Guiada, que orienta a abordagem de modelagem de tópicos, definindo vários tópicos para os quais o modelo irá convergir. O processo está representado na Figura 19. Pode-se dividir este processo em duas etapas principais. Primeiro, tem-se que criar incorporações para cada tópico, juntando-os e passando pelo gerador de *embedding* a base do modelo Bert. Essas incorporações serão comparadas com as incorporações dos documentos analisados através da semelhança do cosseno e atribui-se um rótulo. Se o

documento for mais parecido com um tópico semeado, ele receberá esse tópico; do contrário, se ele for mais parecido com a incorporação média, ele receberá o tópico -1.

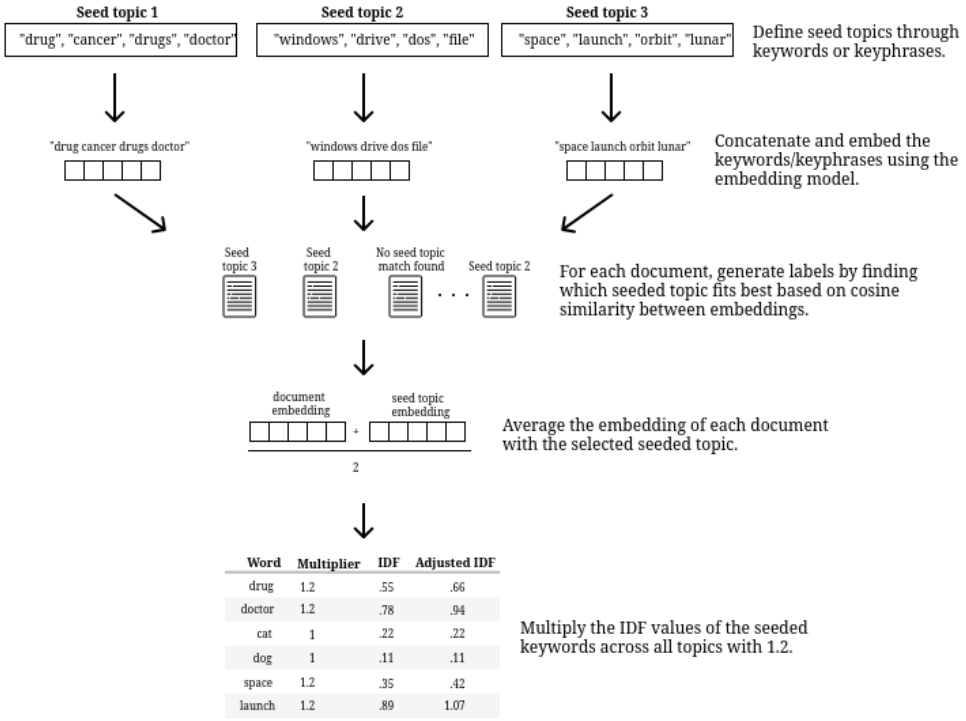


Figura 19 – Processo de representação de tópicos guiado.

Na segunda etapa, pegam-se todas as palavras da lista de tópicos semeados e um multiplicador maior do que 1 para cada palavra. Esse multiplicador é usado para aumentar os valores de IDF dessas palavras em todos os tópicos, favorecendo a presença dessas palavras nos tópicos desejados. Os tópicos utilizados para guiar são as 12 características de IoT, presentes na Seção 2.2, as quais são determinadas palavras e sentenças que descrevem cada característica.

4.5 Considerações Finais

Este capítulo apresentou todas as etapas do método proposto, o ReqCluster4IoT. O método é constituído de quatro etapas: extração de requisitos funcionais, computação de similaridade semântica, agrupamento de requisitos e identificação de características de IoT. Inicialmente, para prover melhora nos resultados dos algoritmos de classificação, foi realizada uma expansão da base de requisitos, a Promise+. Esta nova base de requisitos teve um aumento relativo de aproximadamente 280%, com requisitos anotados por especialistas. Após a formação da Promise+, um experimento foi feito comparando modelos baseados em Transformers e algoritmos de AM. Este experimento demonstrou que modelos baseados em Transformers possuem desempenho maior, estatisticamente

significativo. O modelo de maior desempenho e escolhido para compor o módulo de classificação foi o *bert-base-multilingual-uncased*.

Para a decisão do modelo para o módulo de computação de similaridade semântica, também foi feita uma base de dados, a ReqFuncSimDataset, que contém requisitos em linguagem natural divididos em funcionalidades. Para prover melhora na computação de similaridade, foi feito um ajuste fino nos modelos de Transformers utilizando redes siamesas com perda tripla. Este treinamento demonstrou que o conjunto de dados conseguiu prover melhora aos modelos em comparação ao modelo base. Outro experimento realizado foi para verificar se os modelos ajustados superam a prática mais comum nos trabalhos de agrupamento, similaridade usando TF-IDF. O experimento demonstrou que a maioria dos modelos ajustados supera amplamente o TF-IDF. O modelo *all-MiniLM-L6-v2* tinha a maior diferença entre similares e não similares e, portanto, foi escolhido para esta computação.

Para o agrupamento é utilizado o algoritmo de agrupamento hierárquico aglomerativo em diferentes configurações. O módulo utiliza um algoritmo baseado em dendrograma para obter candidatos a número de grupos. Ao final, a quantidade de grupos é determinada pelo índice de silhouette. Por fim, para a identificação de características de IoT, é utilizada uma abordagem de modelagem de tópicos guiada, que aplica o modelo Bert para gerar incorporações semânticas e identificar os tópicos semeados em documentos de texto.

No próximo capítulo, é realizada uma avaliação do método ReqCluster4IoT com um conjunto de requisitos de uma aplicação de IoT passando por todo o processo. Esta avaliação se dará de maneira qualitativa com estudos experimentais para obter a percepção de desenvolvedores e engenheiros de requisitos sobre os resultados do método.

5 Avaliações Experimentais

Neste capítulo, são apresentados dois estudos experimentais para avaliar os resultados da ReqCluster4IoT aplicada no contexto de automação residencial utilizando objetos IoT. Na Seção 5.1 é apresentada a contextualização e os requisitos da aplicação utilizada. Na Seção 5.2 é exposta à condução de uma pesquisa de opinião com desenvolvedores de software para IoT sobre os resultados produzidos pela ReqCluster4IoT. Por fim, na Seção 5.3 é apresentada a condução de um grupo focal com engenheiros de requisitos, para obter informações mais profundas sobre os resultados da ReqCluster4IoT. Todo o material utilizado nesta avaliação está disponível em um repositório no Zenodo [1].

5.1 Contexto: Automação Residencial utilizando IoT

Aplicações de IoT permitem a conexão de dispositivos e serviços por meio da conexão dos objetos. Um dos contextos de aplicação é o de automação residencial, que permite a criação de ambientes inteligentes que promovem conforto e segurança. Dispositivos deste contexto são termostatos, fechaduras inteligentes, câmeras de segurança e assistentes virtuais. Uma das grandes vantagens da automação residencial é a fácil gerência e controle de diferentes dispositivos, como *smartphones*, *laptops*, *tablets* e relógios inteligentes (STOLOJESCU-CRISAN; CRISAN; BUTUNOI, 2021).

Para os requisitos de software utilizados na avaliação, na ausência de um documento de especificação de requisitos de uma aplicação de IoT, eles foram gerados utilizando o LLM chatGPT 4.o. O uso da LLMs aplicados à engenharia de software tem crescido rapidamente pelos mais diversos domínios (HOU et al., 2024). Uma das aplicações é a geração de requisitos, conforme Gräßler et al. (2022), que utilizou desta geração para aumentar dados de treinamento com requisitos de diferentes domínios de software. Esta abordagem proporcionou melhora na tarefa de extração de requisitos, comprovando a eficácia desta abordagem.

O prompt utilizado no ChatGPT foi o seguinte: *"Given the context of a smart home with an automation process made using the internet of things. Elicit software requirements of this application as requirements in unstructured natural language."*. Os requisitos gerados constam no anexo C.

No âmbito desta dissertação, não foram realizadas comparações entre os resultados do método proposto e um possível processamento dos requisitos utilizando LLMs. A

[1] https://zenodo.org/records/14880051?token=eyJhbGciOiJIUzUxMiJ9.eyJpZCI6IjM0ZjNhYmE1LTBhOGItNGMyM
$Lx_6sjcm4U68VDQWPzad5dgjoKWEbidEZfMQwlQlHV - A$

exposição de dados sensíveis de negócio, como requisitos de software, a LLMs hospedados em nuvem pode representar riscos, incluindo possíveis vazamentos e não conformidade com a Lei Geral de Proteção de Dados (LGPD). Segundo Carlini et al. (2020), LLMs podem aprender e vazar dados confidenciais presentes tanto em seus conjuntos de treinamento quanto nas interações com usuários.

5.2 Pesquisa de Opinião

Para coletar as percepções de desenvolvedores de software para IoT, a estratégia utilizada foi de uma pesquisa de opinião online mediante um formulário. Segundo Punter et al. (2003), esta é uma estratégia de estudo experimental para obter uma descrição quantitativa de parte de uma população coletando dados mediante perguntas às pessoas. Para a realização desta pesquisa de opinião, foram seguidas as etapas definidas em Punter et al. (2003) e com subprocessos identificados em Molléri, Petersen e Mendes (2016), como objetivos de pesquisa e avaliação do instrumento de estudo. O planejamento resultou então nas seguintes etapas: definição do estudo, planejamento do estudo, implementação, execução e análise.

5.2.1 Planejamento

5.2.1.1 Definição do estudo

O objetivo deste estudo é coletar as percepções de desenvolvedores de aplicações IoT sobre os resultados da ReqCluster4IoT. Para fazer isso, o foco foi na avaliação do pertencimento dos requisitos no grupo ao qual foram alocados pelo método de agrupamento. Portanto, foi definida a seguinte questão de pesquisa: (QP-S1) Como é a aceitação dos resultados da ReqCluster4IoT entre desenvolvedores de aplicações de IoT?

A pesquisa foi conduzida em janeiro de 2025 com desenvolvedores de aplicações para IoT selecionados por meio de amostragem por conveniência (PUNTER et al., 2003). Os participantes foram convidados por meio de contatos pessoais e recomendações dos próprios participantes.

5.2.1.2 Planejamento do estudo

O questionário, presente no apêndice D, disponibilizado aos participantes, consiste em duas seções, na qual a primeira traz o resultado do processamento dos requisitos da Seção 5.1 e a segunda traz um questionário com escala Likert. Este questionário foi avaliado por um pesquisador Doutor em Engenharia de Software e com experiência na execução de pesquisas de opinião. Nesta etapa, o principal ajuste foi na seção com o resultado do processamento dos requisitos. Sendo decidido que para cada requisito em

cada grupo, os participantes tinham que julgar se o requisito: “Faz parte do grupo”, “Não faz parte do grupo” ou “Não sei opinar”. Os participantes foram encorajados a desenvolver melhor suas respostas sobre a qualidade dos grupos para mais entendimento, em uma seção de comentários após cada grupo.

Os 24 requisitos elicitados passaram pela etapa de extração de requisitos funcionais usando como classificador o modelo bert-base-multilingual-uncased, conforme definido na Seção 4.1.2. Após este processamento, 20 requisitos foram identificados como funcionais e então foram para a etapa de agrupamento. A ReqCluster4IoT dividiu em cinco grupos, com as seguintes características de IoT: “*Services*” com seis requisitos, *“Things”* com três requisitos, *“Data”* com quatro requisitos, *“Interaction”* com seis requisitos e *“Action”* com somente um requisito.

Após a avaliação dos requisitos em cada grupo, os participantes responderam questões primárias utilizando a escala de Likert (“Concordo Totalmente”, “Concordo”, “Não estou decidido”, “Discordo” ou “Discordo Totalmente”), para capturar a percepção geral dos participantes sobre um aspecto específico. Foram formuladas três afirmações sobre a ReqCluster4IoT. As afirmações foram inspiradas nos constructos do Modelo de Aceitação de Tecnologia (do inglês, *Technology Acceptance Model* ou TAM) (DAVIS; DAVIS, 1989). O TAM é um modelo amplamente utilizado para explicar como usuários aceitam e utilizam uma determinada tecnologia. As afirmações estão contidas na Tabela 11.

Tabela 11 – Afirmações relativas à ReqCluster4IoT.

ID	Declaração	Aspecto	Descrição
S1	Os resultados do método são úteis.	Utilidade	Refere-se ao grau em que um participante acredita que usar o método produziria resultados desejáveis.
S2	Usar os resultados do método poderia reduzir meu esforço em lidar com o problema	Potencial de redução de trabalho	Refere-se ao grau em que um participante acredita que usar o método produziria resultados desejáveis.
S3	Eu poderia usar esse método se tivesse oportunidade.	Intenção de uso	Refere-se à probabilidade de um participante usar o método no futuro.

5.2.1.3 Implementação do estudo

Antes da disponibilização dos formulários aos participantes, foi feita uma execução piloto com dois pesquisadores com experiência na condução de pesquisas de opinião. Neste processo, foi definido o formato para as questões. O formulário foi então disponibilizado

Tabela 12 – Perfil dos participantes da pesquisa de opinião.

ID	Grau Acadêmico	Experiência com Desenvolvimento	Experiência com IoT
01	Mestrado	Mais de 1 ano de trabalho	Mais de 1 ano de trabalho
02	Graduado	Até 6 meses de trabalho	Acadêmico e um projeto
03	Mestrado	Mais de 1 ano de trabalho	Acadêmico e um projeto
04	Ensino Médio	Mais de 1 ano de trabalho	Acadêmico e um projeto
05	Doutorado	Até 6 meses de trabalho	Acadêmico e um projeto
06	Mestrado	Mais de 1 ano de trabalho	Acadêmico e um projeto
07	Mestrado	Mais de 1 ano de trabalho	Mais de 1 ano de trabalho
08	Doutorado	Mais de 1 ano de trabalho	Acadêmico e um projeto
09	Graduado	Mais de 1 ano de trabalho	Acadêmico e um projeto
10	Graduado	Mais de 1 ano de trabalho	Mais de 1 ano de trabalho

aos participantes por meio do Google Forms[2]. Esta ferramenta permite uma rápida implementação e fácil coleta e armazenamento dos dados.

5.2.1.4 Execução e análise

Os participantes que aceitaram o convite receberam um e-mail contendo: (1) Termo de Consentimento Livre e Esclarecido (TCLE); (2) formulário de caracterização; (3) Um vídeo contendo breve explicação sobre a ReqCluster4IoT e conceitos de Engenharia de Requisitos e IoT; (4) Formulário do estudo e (5) dois documentos suplementares contendo descrição do software e seus requisitos, disponíveis no apêndice C, e documento contendo características de IoT, disponíveis na Seção 2.2. A análise dos dados considerou tanto os comentários quanto o quantitativo das respostas para cada grupo.

5.2.2 Resultados

Na Tabela 12 têm-se os perfis dos participantes da pesquisa de opinião. Quase todos possuem graduação na área de computação, com quatro mestres e dois doutores. O participante 04 ainda não possui graduação, porém está em fase de término de sua graduação em computação. Com relação à experiência em desenvolvimento de software, seis participantes possuem experiência profissional de mais de um ano, enquanto outros dois possuem conhecimento acadêmico e um projeto. Por fim, todos os participantes possuem experiência em pelo menos um projeto de IoT, sendo que três têm mais de um ano de experiência profissional na área.

Na Figura 20 tem-se o resultado para o grupo *Service*. Os requisitos 1, 4 e 14 foram unanimemente considerados pertencentes ao grupo. O requisito 9 apresentou a maior divergência, com somente seis participantes considerando pertencentes ao grupo. O requisito 9 possui a seguinte descrição: *"The application should use encryption to*

[2] https://docs.google.com/forms

protect data transmitted between the smart devices and the cloud, ensuring the privacy and security of user data. It should also detect and alert users about suspicious activities (e.g., unauthorized access attempts)". O participante 04 considerou que *"os reqs 1 e 9 são requisitos de segurança"*, ponto esse também destacado pelo participante 05, que sugeriu: *"Alguns dos requisitos estão relacionados à segurança, talvez segurança possa ser um grupo, analisar"*. Os requisitos 12 e 19 tiveram a maioria dos participantes considerando-os como pertencentes ao grupo. De maneira geral, houve um alto nível de concordância quanto à maioria dos requisitos, com o requisito 9 sendo o que mais gerou dúvidas entre os participantes.

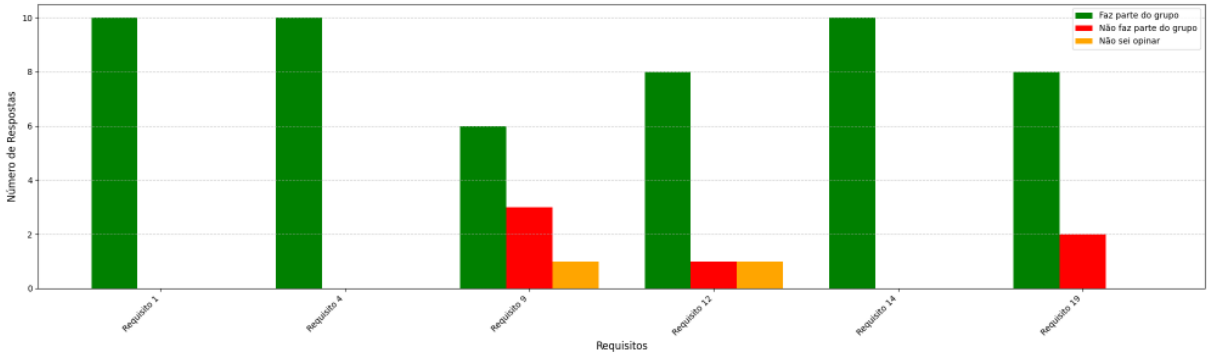


Figura 20 – Resposta para os participantes do formulário sobre o grupo de *Service*.

Na Figura 21 tem-se a apresentação para o grupo de *Things*. O requisito 2 foi amplamente considerado pertencente ao grupo, com sete participantes concordando, um discordando e dois sem opinião. O requisito 7 apresentou a maior divergência, com apenas quatro votos favoráveis e seis participantes afirmando que não faz parte do grupo. O requisito 07 possui a seguinte descrição: *"The application should provide both mobile and web interfaces that are responsive and user-friendly, allowing users to control smart home devices from any device with internet access"*. O participante 04 destacou que: *"O requisito 7 é a respeito do front-end e suas capacidades (req de usabilidade)"*. Já o requisito 20 teve ampla aceitação, com oito participantes concordando, um discordando e um sem opinião.

Na Figura 22 há os resultados para o grupo de *Data*. O requisito 15 foi unanimemente considerado pertencente ao grupo. O requisito 3 teve maioria favorável, com sete participantes concordando que faz parte do grupo, enquanto três discordaram. Já o requisito 5 apresentou maior divergência, com apenas quatro votos favoráveis e seis contrários. O requisito 5 possui a seguinte definição: *"The system shall allow users to create custom automation rules that define how devices should behave based on specific conditions or events (e.g., turning on lights when motion is detected or lowering the thermostat at a set time). Users should be able to combine multipleconditions and actions in their automation scripts"*. O participante 05 destacou que: *"o req 5 é uma funcionalidade do sistema, porém*

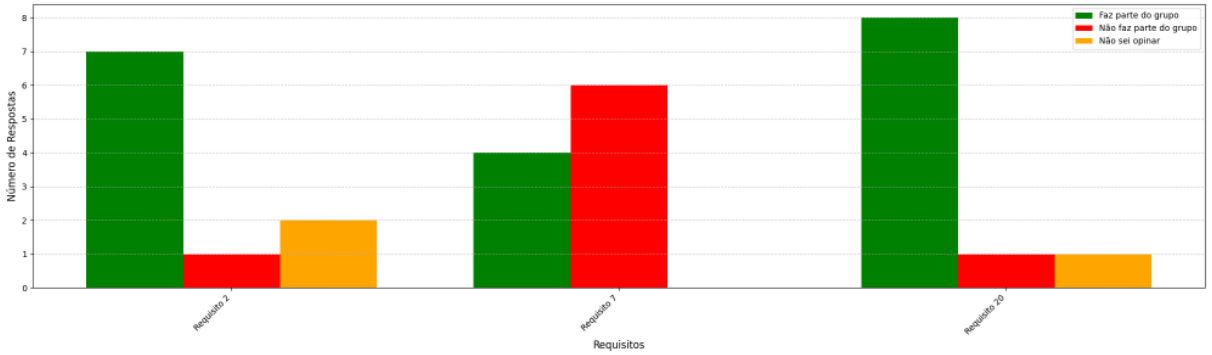


Figura 21 – Resposta para os participantes do formulário sobre o grupo de *Things*.

a funcionalidade inclui atuar sobre o ambiente". O requisito 13 ficou equilibrado, com cinco participantes considerando-o pertencente ao grupo e cinco discordando.

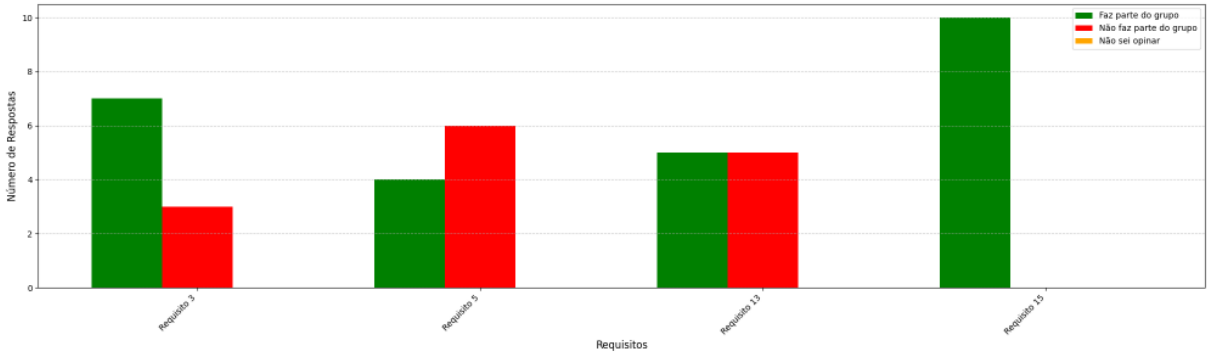


Figura 22 – Resposta para os participantes do formulário sobre o grupo de *Data*.

A Figura 23 apresenta o resultado para o grupo de *Interaction*. O requisito 6 foi amplamente considerado pertencente ao grupo, com nove participantes concordando e apenas um discordando. O requisito 8 apresentou grande divergência, com apenas três votos a favor, seis contra e um participante indeciso. O requisito 10 também gerou opiniões divididas, com cinco participantes afirmando que faz parte do grupo, quatro discordando e um sem opinião. O requisito 16 teve uma maioria favorável, com seis votos a favor e quatro contra. Já o requisito 17 foi o mais contestado, com apenas dois votos a favor e oito contra. Por fim, o requisito 18 foi amplamente aceito, com oito votos a favor e dois contra. No geral, houve consenso sobre os requisitos 6 e 18, enquanto o requisito 17 foi o mais rejeitado. O requisito 8 também apresentou forte discordância. O participante 04 destacou que os requisitos 8 e 17 seriam uma funcionalidade do sistema.

O último grupo, cuja característica é Action, é composto apenas pelo requisito 20. Neste grupo, oito participantes consideraram este requisito pertencente ao grupo, enquanto dois consideraram não pertencer. Na Figura 24, tem o resultado da aplicação do formulário de Likert. A maioria considerou os resultados do método úteis, enquanto um

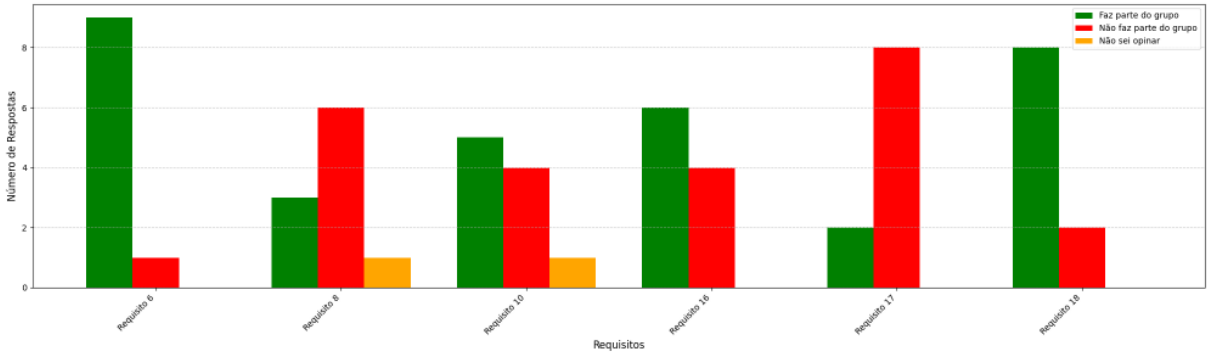


Figura 23 – Resposta para os participantes do formulário sobre o grupo de *Interaction*.

não está decidido. Quanto ao potencial de redução de trabalho, o todos os participantes concordaram que usar os resultados do método poderia reduzir seus esforços em lidar com o problema. Já para a utilização, todos os participantes concordaram que utilizariam o método, com predominância de concordância total.

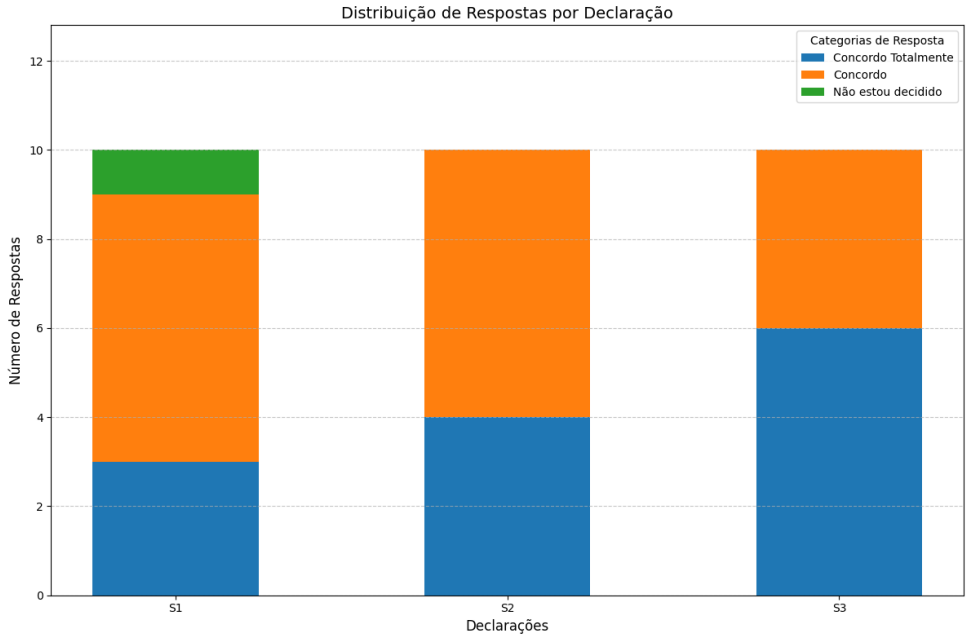


Figura 24 – Resposta dos participantes sobre as afirmações acerca da ReqCluster4IoT.

(QP-S1) Como é a aceitação dos resultados da ReqCluster4IoT entre desenvolvedores de aplicações de IoT? Os resultados desta pesquisa de opinião trazem que os desenvolvedores tiveram boa recepção ao método. Os participantes consideraram que a maioria dos requisitos está corretamente agrupada e pertence aos grupos propostos. A maioria dos participantes concordou com a utilidade do método, potencial de redução de trabalho e utilizaria o método caso tivessem a oportunidade. Os participantes também indicaram melhorias, como alguns requisitos não funcionais, dentre os indicados como funcionais e requisitos podendo pertencer a mais de um grupo.

5.3 Grupo Focal

O segundo estudo experimental para avaliar os resultados da ReqCluster4IoT foi um grupo focal. Um grupo focal é uma técnica de pesquisa qualitativa que reúne um pequeno grupo de pessoas para discutir um tópico específico, facilitado por um moderador, visando explorar percepções e experiências. Este método foi escolhido, por ser rápido, de baixo custo, pode prover conteúdo rico em informações qualitativas e revelar pontos de vista que são difíceis de serem capturados com outros métodos (KONTIO; LEHTOLA; BRAGGE, 2004). Além disso, este método é uma forma de entender melhor os resultados da ReqCluster4IoT, bem como o resultado da pesquisa de opinião com desenvolvedores. Para a realização deste grupo focal, foram seguidas as etapas definidas por Kontio, Lehtola e Bragge (2004): definição do problema a ser investigado, seleção dos participantes, coleta dos dados, execução do grupo focal e análise de dados.

5.3.1 Planejamento

5.3.1.1 Definição do problema a ser investigado

Este grupo focal objetiva obter as percepções sobre os resultados da ReqCluster4IoT com base em uma perspectiva intuitiva. A avaliação envolveu a participação de profissionais com experiência na área de Engenharia de Requisitos. A ReqCluster4IoT foi apresentada e demonstrada aos participantes. Este grupo focal se concentrou no entendimento da percepção dos participantes sobre utilidade, potencial para redução da carga de trabalho e intenção de uso dos resultados da ReqCluster4IoT. Estes critérios ajudam a avaliar a viabilidade para usuários pretendidos. Portanto, foi definida a seguinte questão de pesquisa: (QP-GF1) Como é a aceitação dos resultados da ReqCluster4IoT entre Engenheiros de Requisitos?

5.3.1.2 Seleção dos participantes

Quatro participantes foram convidados considerando suas experiências profissionais tanto na academia quanto na indústria, relacionadas à Engenharia de Requisitos. Hennink e Kaiser (2022) conduziram uma revisão sistemática da literatura no tamanho das amostras para saturação em pesquisa qualitativa. Os autores identificaram que a saturação pode ser alcançada em uma faixa de quatro a oito participantes em grupos focais. Os participantes receberam TCLE e não são identificados nas suas respostas e discussões durante a dinâmica de grupo devido a questões de confidencialidade e anonimato.

Na Tabela 13, têm-se os perfis dos participantes do grupo focal. Todos possuem Mestrado em Ciência da Computação, e o participante 4 possui Doutorado. Todos os participantes possuem experiência profissional com Engenharia de Requisitos. Com

Tabela 13 – Perfil dos participantes do grupo focal.

ID	Grau Acadêmico	Experiência com Requisitos	Experiência com IoT
01	Mestrado	Mais de 1 ano de trabalho	Até 6 meses de trabalho
02	Mestrado	Até 6 meses de trabalho	Até 6 meses de trabalho
03	Mestrado	Até 6 meses de trabalho	Conhecimento Acadêmico
04	Doutorado	Mais de 1 ano de trabalho	Conhecimento Acadêmico

relação à experiência em projetos de IoT, somente dois participantes possuem experiência profissional.

5.3.1.3 Coleta dos dados

Para além dos dados coletados durante a sessão de grupo focal, um questionário online foi respondido previamente pelos participantes. O questionário é o mesmo utilizado na pesquisa de opinião, cujo processo é apresentado na Seção 5.2.1.2 e está disponível no anexo D.

Após a execução do grupo focal, os participantes preencheram um formulário com questões primárias para obter respostas utilizando a escala de Likert, para capturar a percepção geral dos participantes sobre um aspecto específico, dados presentes no repositório do zenodo [3]. Foram formuladas três afirmações inspiradas no TAM (DAVIS; DAVIS, 1989) sobre a ReqCluster4IoT, sendo estas as mesmas afirmações utilizadas na pesquisa de opinião, que estão contidas na Tabela 11.

5.3.1.4 Execução do Grupo Focal

Considerando as diferentes localizações dos participantes, o grupo focal foi realizado utilizando o Google Meet[4] como ferramenta de videoconferência. Essa plataforma oferece maior flexibilidade para a participação, possibilitando a adaptação dos horários conforme a disponibilidade dos envolvidos e facilitando a colaboração entre os participantes.

Para garantir uma seção fluida, foi realizada uma seção de grupo focal piloto com dois pesquisadores em engenharia de software com experiência em condução de pesquisa qualitativa. Esta abordagem de condução de uma seção piloto seguiu os passos relatados em Abbas et al. (2022). Nesta seção piloto, foram refinados o formulário a ser preenchido pelos participantes, a explicação do método e as etapas do grupo focal.

Após esta avaliação inicial, foi iniciada então a execução do estudo. Dias antes da sessão, os participantes receberam e-mail com breve explicação do método e um documento contendo: o contexto do software (automação residencial), todos os requisitos

[3] https://shorturl.at/9e1qG
[4] https://meet.google.com

elicitados do software, lista de requisitos funcionais encontrados pelo método, os grupos e suas características de IoT. Outro documento enviado aos participantes foi um contendo explicações sobre as características de IoT, contidas na Seção 2.2. O grupo focal foi realizado numa sessão de 100 minutos. A sessão foi gravada com a permissão dos participantes e contou com a seguinte estrutura:

- Introdução do moderador e participantes;
- Esclarecimento sobre o contexto do estudo;
- Apresentação de conceitos de Engenharia de Requisitos e IoT;
- Apresentação do objetivo do estudo;
- Apresentação da estrutura e conteúdo do ReqCluster4IoT;
- Apresentação do software problema e da solução encontrada;
- Discussão relacionada às respostas dos participantes;
- Resposta à pergunta: "Os grupos encontrados ajudam na compreensão dos requisitos de software e da aplicação?"

5.3.1.5 Análise dos dados

A sessão foi gravada e transcrita, sendo empregados procedimentos de codificação aberta para dar suporte à análise das respostas dos participantes. Essa abordagem sistemática permite analisar as respostas e identificar temas e padrões comuns (CORBIN; STRAUSS, 2014). Como suporte para a codificação aberta, foi utilizada a ferramenta ATLAS.ti[5].

5.3.2 Resultados

O primeiro grupo de requisitos tratado no grupo focal foi o grupo cuja característica é *Service*. Este grupo é formado por seis requisitos. A Figura 25 apresenta os resultados do formulário prévio sobre o julgamento deste grupo. Com exceção do requisito 9, os demais foram unanimemente considerados parte do grupo. No grupo focal, o participante 03 comentou: *'Eu acho que a maioria desses requisitos que foram encaixados nesse grupo, eles estão realmente relacionados à funcionalidade desse sistema IoT que está sendo planejado".*

O requisito 9 que possui a seguinte descrição: *"The application should use encryption to protect data transmitted between the smart devices and the cloud, ensuring the privacy and security of user data. It should also detect and alert users about suspicious activities (e.g.,*

[5] https://atlasti.com/

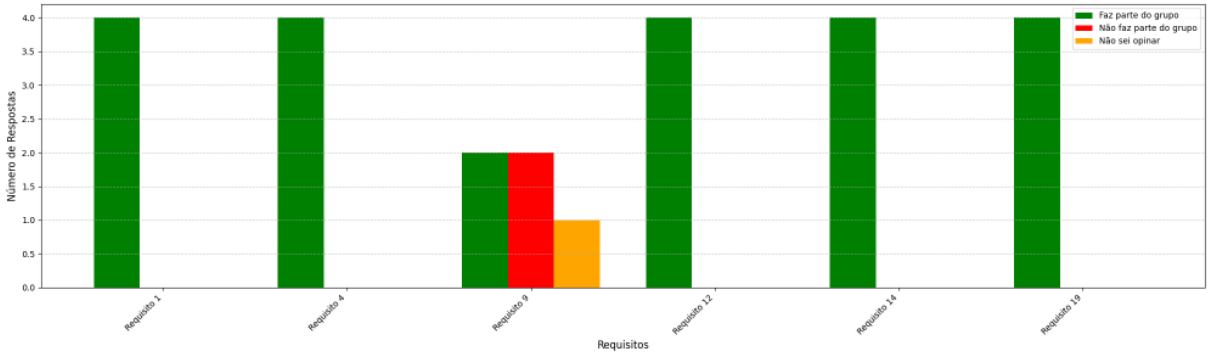


Figura 25 – Resposta para os participantes sobre o grupo de *Service*.

unauthorized access attempts)", os participantes 02 e 03 destacaram que este requisito não pertenceria ao grupo *Service* exclusivamente, mas sim que poderia ter mais de uma característica. O participante 03 complementou: *"Eu fiquei realmente na dúvida se realmente isso seria uma funcionalidade do Sistema ou seria algo que entraria na característica Data"*. Já o participante 01 considerou que este requisito seria não funcional, ao invés de funcional, visão corroborada pelos demais participantes. O participante 01 destacou: *"Eu apontei como um requisito não funcional porque eu identifiquei que embora ele seja importante relacionado ao requisito ele está mais relacionado à segurança do sistema que eu vejo mais como um aspecto transversal"*.

Neste momento instaurou-se uma discussão sobre a presença de requisitos não funcionais na lista de requisitos funcionais. Segundo o participante 01: *"No grupo de Things, por exemplo. Eu identifiquei que existe o Requisito 7, que ele fala muito da questão da interface, que ela deve ser responsável e amigável para o usuário. Eu identifiquei que isso está mais ligado a um requisito não funcional do que um requisito funcional."*. O requisito 7 que é elicitado da desta maneira: *"The application should provide both mobile and web interfaces that are responsive and user-friendly, allowing users to control smart home devices from any device with internet access"*. O participante 02 destacou: *"É difícil de fato ali a gente definir um requisito funcional de sistemas IoT, tirando os requisitos não funcionais"*. O participante 03 corroborou tal visão: *"Eu acho que realmente tem essa questão da transversalidade entre as características e a forma como os requisitos foram especificados"*. Ao final desta discussão, todos os participantes concordaram que quase a totalidade dos requisitos de fato eram funcionais.

O segundo grupo de requisitos discutido foi o grupo que tem como característica *Things*, formado por três requisitos. A Figura 26 apresenta os resultados do julgamento dos participantes. Neste grupo, houve uma unanimidade com relação de pertencimento ao grupo com relação ao requisito 2 e um predomínio com relação ao requisito 20. Com relação ao requisito 7, a discordância foi com relação a este requisito ser não funcional.

No mais, este foi o grupo no qual os participantes mais concordaram. O participante 03 apontou: *"eu acho que foi o grupo que eu mais assim concordei e ficou mais claro essa classificação"*.

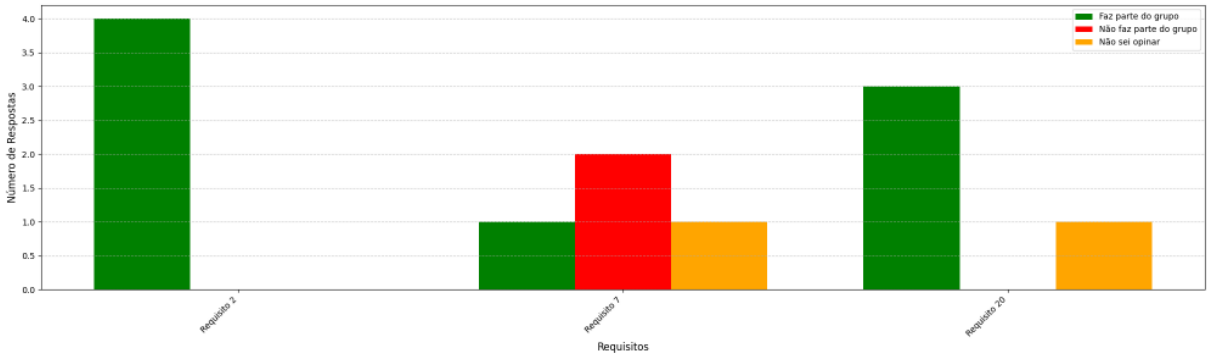


Figura 26 – Resposta para os participantes sobre o grupo de *Things*.

O terceiro grupo de requisitos formado pelo método e discutido no grupo focal foi o grupo que tem como característica *Data*, formado por quatro requisitos. A Figura 27 apresenta os resultados dos participantes. Houve unanimidade de pertencimento para o requisito 3 e requisito 15, e aí há discordância entre o 5 e 13. Para o requisito 5, os participantes 02 e 03 consideraram que este requisito estaria mais relacionado com *Service*. Para o requisito 15, elicitado como: *"The application should collect and store data from all connected devices for historical analysis. Users should be able to visualize this data through charts or reports to understand patterns in device usage"*, o participante 02 levantou a possibilidade dele ser de interação e o participante 03 considerou: *"Com relação ao 15, eu também tive a mesma percepção, porque eu fiquei na dúvida se isso não seria uma interação, a partir do momento que ele vai estar interagindo com outros tipos de dispositivos para obter essas informações externas"*.

O participante 01 considerou que o requisito 15 poderia ser tanto *Data* quanto *Interaction*: *"Eu vejo que seriam duas coisas, entendeu? Poderia ter essa parte dos dados, a partir do momento que você pensa na forma em que a pessoa quer visualizar o dado, mas também tem essa parte que o participante 02 comentou de, a partir do momento que você dá a funcionalidade de customização, poderia ser um serviço também. Então eu vejo que poderia ter duas características"*.

O quarto grupo formado pela ReqCluster4IoT foi o grupo cuja característica é *Interaction*. A Figura 28 traz os resultados desta avaliação. Neste grupo, os participantes tiveram maior grau de concordância. Houve unanimidade para o requisito 6 e maioria para o requisito 18 quanto ao pertencimento a este grupo. O requisito 17 ficou com unanimidade de não pertencimento. Nos requisitos 8, 10 e 16 os participantes divergiram e alguns participantes relataram dificuldade de entendimento com a definição da característica

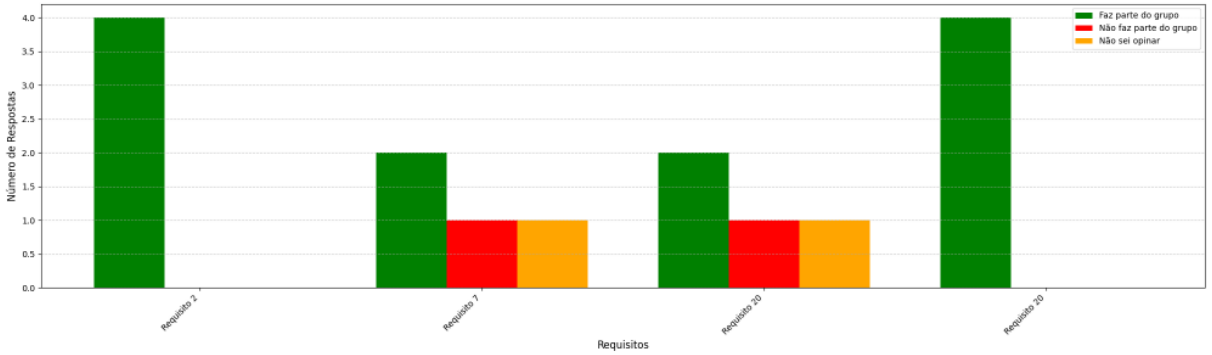


Figura 27 – Resposta dos participantes sobre o grupo de *Data*.

em si. O participante 04 destacou esta dificuldade de entendimento, já o participante 02 pontou que: *"Uma interação, por exemplo, seria um dispositivo enviar, por exemplo, alguma coisa pro broker. Pra mim seria uma interação"*.

O participante 02 ainda levantou que a definição da característica exclui o usuário da interação, ponto corroborado pelos demais participantes como elemento que pode ter causado a falta de entendimento. Neste momento, o participante 02 sugeriu: *"Eu acho que talvez na definição ali colocar não só os dispositivos interagindo entre si, então colocar com outras pessoas também, com usuários"*. O último grupo era composto por apenas um requisito, o requisito 20, cuja característica era *Action*. Os participantes tiveram unanimidade em declarar este requisito como pertencente ao grupo. Nenhum comentário mais foi feito durante a sessão por parte dos participantes.

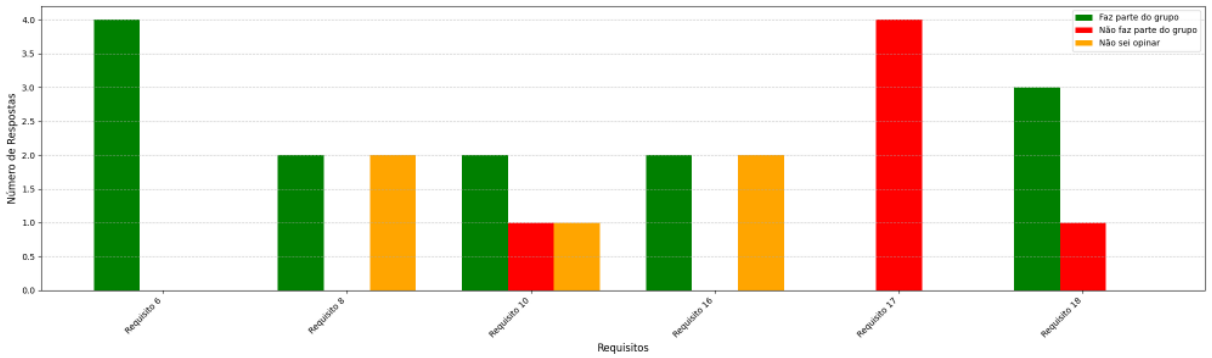


Figura 28 – Resposta dos participantes sobre o grupo de *Interaction*.

Com relação as respostas à pergunta realizada ao final do grupo focal, os participantes concordaram que o agrupamento pode auxiliar na compreensão dos requisitos de software do projeto. O participante 01 declarou: *"Acho que isso ajuda muito a gente ter uma ideia de quais são as interações, quais são os serviços, quais são os dados e quais são as ações que são necessárias, por exemplo, por um... um projeto ou um serviço que vai ser oferecido"*. O participante 03 seguiu na mesma direção: *"A partir do momento que você*

tem isso agrupado, você pode ter melhores estratégias para concentrar os esforços da equipe, observar as relações entre os próprios requisitos, já que eles estão dentro de um mesmo grupo".

Os participantes também relataram sobre o potencial do agrupamento para aplicações de IoT. O participante 04 destacou que: *"Quando a gente tem já um conteúdo específico, como o IoT, e consegue definir características relacionadas a esse contexto e assim classificar os requisitos, isso facilita seja a estratégia de implementação, seja a organização, seja a gerência desses requisitos em projeto".* Tais benefícios foram corroborados pelo participante 02, que afirmou: *"Pensando em sistemas IoT, que tem toda essa interação, esse ecossistema bem amplo de bastante coisas diferentes, de inteligência, de sensores, de atuadores, isso acaba levando e direcionando a gente a pensar nessas partes específicas. Por isso que acredito que ajuda bastante também".*

Por fim, o participante 03 destaca que: *"Pensando em sistemas IoT, que tem toda essa interação, esse ecossistema bem amplo de bastante coisas diferentes, de inteligência, de sensores, de atuadores, isso acaba levando e direcionando a gente a pensar nessas partes específicas. Por isso que acredito que ajuda bastante também. Isso de alguma forma acaba sendo facilitado, você saber qual requisito está mais ligado a outro, qual vai impactar mais diretamente outro".*

A Figura 29 apresenta os resultados para o questionário de Likert das declarações. Os participantes foram unânimes em concordar totalmente que os resultados do método são úteis e que utilizariam o método caso tivessem a oportunidade. Sobre o potencial de redução de trabalho, dois participantes concordaram totalmente e dois participantes concordaram.

(QP-GF1) Como é a aceitação dos resultados da ReqCluster4IoT entre Engenheiros de Requisitos? Os resultados deste grupo focal evidenciam que os engenheiros de requisitos tiveram boa recepção ao método. Os participantes avaliaram que a maioria dos requisitos foi adequadamente agrupada e corresponde aos grupos sugeridos. A maioria dos participantes reconheceu a utilidade do método, seu potencial para reduzir o trabalho e afirmou que o utilizaria caso tivesse a oportunidade. Os participantes também destacaram que a utilização do método no contexto de IoT, ao considerar as características específicas deste domínio, permite a identificação de pontos críticos e facilita na implementação, organização e verificação de conexão entre os requisitos. Os participantes também sugeriram melhorias, como requisitos não funcionais que foram classificados considerados funcionais, e a possibilidade de alguns requisitos pertencerem a mais de um grupo.

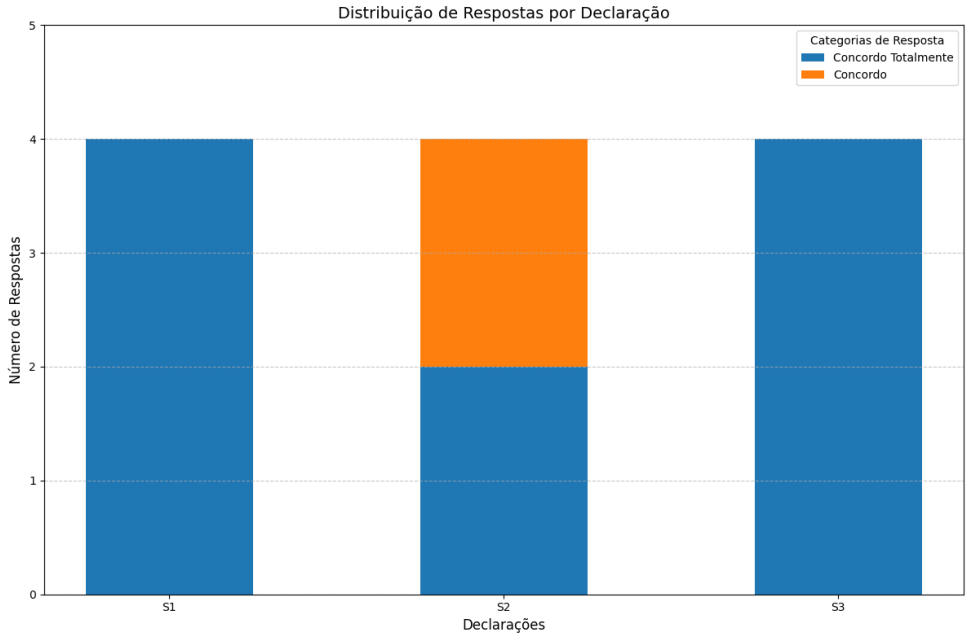


Figura 29 – Resposta dos participantes sobre as afirmações acerca da ReqCluster4IoT.

5.4 Discussão

Os resultados obtidos a partir da pesquisa de opinião fornecem uma visão sobre a percepção de desenvolvedores de software para IoT em relação ao método ReqCluster4IoT. Já o grupo focal foi realizado para obter as percepções de engenheiros de requisitos e coletar informações mais detalhadas sobre os resultados do método. Em ambos os casos, os resultados demonstraram que a maioria dos participantes reconheceu o pertencimento dos requisitos nos grupos propostos. Significando, portanto, que o método conseguiu capturar as relações entre os requisitos e criou grupos coesos.

No grupo de *Service*, tanto para desenvolvedores quanto para engenheiros de requisitos, houve ampla percepção de pertencimento dos requisitos. Para o requisito 9, ambos os grupos de participantes concordaram que se trataria de um requisito não funcional. A presença de alguns requisitos não funcionais é decorrência da classificação automática dos requisitos, porém, conforme destacado pelos engenheiros de requisitos, a ampla maioria dos requisitos de fato são funcionais. Isso indica que o algoritmo de classificação utilizado pelo método consegue, na maioria dos casos, discriminar os requisitos funcionais corretamente.

No grupo de *Things*, somente o requisito 07 foi tratado como não pertencente por ambos os grupos, por se tratar também de um não funcional. Um engenheiro do requisito tratou este grupo como o que ele mais concordou e que ficou clara a classificação do grupo. O grupo de *Interaction* foi o que mais houve discordância para ambos os grupos de participantes. No grupo focal, os engenheiros de requisitos destacaram que esta definição pode não estar clara o suficiente e exclui os usuários das interações, o que pode deixar

importantes interações passarem despercebidas.

Já para o grupo de *Data*, os engenheiros de requisitos destacaram que poderia haver uma interseção entre *Data* e *Interaction*, o que pode levar a uma revisão do processo de indicação de características para uma estrutura múltipla ou hierárquica. Esta interdependência de características pode ser uma decorrência da dificuldade de gestão da interação entre elementos de IoT (SILVA; GONçALVES; ROCHA, 2019). Por fim, no grupo Action, para ambos os grupos de participante, o requisito único foi considerado como pertencente a característica de ação.

Para a pergunta realizada aos engenheiros de requisitos sobre a utilização de grupos de requisitos na compreensão dos requisitos e da aplicação, todos os participantes destacaram a importância desta abordagem. Os participantes foram ao encontro dos trabalhos de agrupamento de requisitos (AL-OTAIBY; ALSHERIF; BOND, 2005; GALSTER; EBERLEIN; JIANG, 2013; SALMAN et al., 2018), destacando que o agrupamento contribui para: (1) melhores estratégias para esforços da equipe, (2) observar melhores relações entre os requisitos e, por fim, (3) identificar interações e serviços.

Quanto à aplicação no domínio de IoT, os engenheiros de requisitos destacaram que a abordagem de agrupamento pode contribuir para a análise dos requisitos neste ambiente de heterogeneidade de dispositivos e tecnologias, uma dificuldade presente no domínio de IoT (KHANNA; KAUR, 2020). Outra contribuição presente no domínio trata sobre a implementação, organização e gerência dos requisitos no projeto.

Com relação às respostas às afirmações baseadas nos constructos do TAM, tem-se que ambos os grupos de participantes tiveram recepção positiva dos resultados do ReqCluster4IoT. A ampla maioria dos participantes considerou o método útil (S1) e acredita que ele tem potencial para reduzir o esforço e todos os participantes indicaram intenção de uso da técnica em projetos futuros (S3). Tais resultados reforçam sua aplicabilidade prática no desenvolvimento de soluções IoT.

Como implicações para a pesquisa em Engenharia de Requisitos tem-se que: (1) o reconhecimento do pertencimento dos requisitos aos grupos propostos sugere que o método ReqCluster4IoT conseguiu capturar as relações entre os requisitos, validando sua eficácia no agrupamento automático. Esse resultado pode motivar novas pesquisas sobre aprimoramento de algoritmos de agrupamento e sua aplicabilidade em diferentes domínios; (2) a dificuldade na classificação de requisitos não funcionais, evidenciada pelos requisitos 07 e 09, sugere que a distinção entre requisitos funcionais e não funcionais ainda é um desafio, principalmente para domínios específicos, como IoT, motivando novas pesquisas para a geração de base de dados de requisitos de IoT e, por fim, (3) a percepção positiva dos engenheiros de requisitos sobre o uso de grupos reforça a importância da pesquisa em técnicas de agrupamento semântico para melhorar a análise e compreensão de requisitos.

Como implicações para a prática em Engenharia de Requisitos, destacam-se: (1) a abordagem de agrupamento pode contribuir para a organização eficiente de requisitos em projetos de IoT, facilitando a identificação de dependências e o planejamento de implementação; (2) a divisão de requisitos pode permitir estratégias mais direcionadas para equipes, otimizando a divisão de tarefas e melhorando a colaboração entre diferentes áreas do projeto e, por fim, (3) a heterogeneidade de dispositivos e tecnologias no contexto de IoT torna constitutiva a adoção de abordagens estruturadas para análise de requisitos. O agrupamento pode auxiliar na análise, melhorando na implementação e gerência dos requisitos.

5.5 Considerações Finais

Este capítulo apresentou dois estudos experimentais utilizados para avaliar o método ReqCluster4IoT. O primeiro estudo foi uma pesquisa de opinião com desenvolvedores de software para IoT. Pesquisas de opinião são um instrumento útil para obter uma descrição quantitativa para uma parte da população mediante perguntas às pessoas. O segundo estudo foi um grupo focal com engenheiros de requisitos. Grupo focal é uma técnica rápida, de baixo custo, que reúne um pequeno grupo de pessoas para discutir um tópico específico. O método também foi avaliado com base nos constructos do TAM quanto à utilidade, potencial de redução de trabalho e intenção de uso.

Os resultados das avaliações experimentais apresentadas evidenciam que o método de agrupamento de requisitos funcionais pode apoiar analistas de requisitos e desenvolvedores de aplicações de IoT. Conforme destacado no grupo focal, esta abordagem pode contribuir para melhores estratégias para o esforço da equipe, a observar relações entre requisitos, interações e serviços da aplicação IoT.

A avaliação experimental também evidencia que o método conseguiu criar grupos coesos com os requisitos, mediante a maioria de concordância na maior parte dos grupos por parte tanto dos engenheiros de requisitos quanto pelos desenvolvedores. O grupo de maior discordância foi o de *Interaction*, o que pode sugerir alterações em sua definição. O método também foi considerado útil, com potencial de redução de trabalho e a maioria dos participantes utilizaria se tivessem a oportunidade. O método ReqCluster4IoT apresenta novidades ao considerar características específicas do domínio de IoT, classificação automática de requisitos e uma nova abordagem para computar a similaridade semântica entre requisitos.

6 Conclusão e trabalhos futuros

IoT é um novo paradigma no desenvolvimento de software, no qual objetos heterogêneos do cotidiano podem se conectar entre si ou com serviços através da internet, para prover as mais diversas aplicações, atingindo os campos de agricultura, automação industrial, entre outros. Neste ambiente de alta heterogeneidade de dispositivos e aplicações, torna-se complexo a análise de requisitos de software.

O desenvolvimento de software para IoT agrega diversas complexidades, como heterogeneidade dos dispositivos e integração hardware e software, bem como demanda experiência de seus analistas em diversas áreas do contexto do software. Além disso, os domínios de aplicações destes softwares envolvem uma diversidade de áreas, como processamento industrial, mobilidade, agricultura e monitoramento de espaço público. Esta diversidade torna complexo o tratamento dos requisitos de software, sobretudo em projetos com alto número de requisitos.

Nesta dissertação, foi apresentado o método ReqCluster4IoT para agrupar requisitos funcionais e inferir característica IoT para o grupo. O foco do ReqCluster4IoT é o processamento de requisitos de software elicitados em linguagem natural não estruturada. O método proposto consegue ainda identificar de forma automática requisitos funcionais a partir da lista de requisitos elicitados. Foram realizados experimentos em classificação de requisitos, para identificação automática de requisitos funcionais, e em computação de similaridade semântica, para ajustar modelos baseados em Transformers. Ambos os experimentos demandaram a criação de bases de dados, a Promise+ para classificação e a ReqFuncSimDataset para ajuste dos modelos. Por fim, para a identificação de característica, foram utilizadas 12 características de IoT encontradas na literatura.

Para avaliar o método, foram feitos dois experimentos, uma pesquisa de opinião e um grupo focal, ambos com requisitos no contexto de uma aplicação de automação residencial utilizando IoT. Ambos os experimentos focaram na aceitação dos resultados do método. A pesquisa de opinião foi realizada com desenvolvedores de software para IoT, tendo eles recebido um formulário de caracterização, um vídeo com informações sobre o método e um formulário de avaliação dos grupos e do método. Já o grupo focal foi realizado com engenheiros de requisitos, tendo eles recebido um formulário de caracterização, um formulário de avaliação dos grupos e uma sessão virtual foi realizada. Todos os instrumentos utilizados nos estudos foram disponibilizados, promovendo a ciência aberta[1].

Os resultados da pesquisa de opinião demonstraram que o método conseguiu

[1] https://shorturl.at/9e1qG

agrupar requisitos similares, tendo, boa parte, confirmação de pertencimento. Houve também boa aceitação quanto à utilidade, potencial de redução de esforço e intenção de uso. No grupo focal, o resultado da avaliação dos Engenheiros de Requisitos foi similar aos desenvolvedores, houve também boa aceitação quanto à utilidade, potencial de redução de esforço e intenção de uso. Em ambos os estudos, pontos de melhoria foram citados, como requisitos não funcionais em meio aos funcionais, grupos de requisitos poderiam ter mais de uma característica de IoT e alterações nas definições de características de IoT.

6.1 Contribuições

Esta dissertação de mestrado apresentou algumas contribuições, incluindo:

- **Expansão de uma base de requisitos de software**: a geração da base Promise+, sendo uma expansão da base Primise_exp, a partir de anotações feitas por especialistas. Esta nova base apresenta um grande aumento percentual nas classes e pode ser utilizada pela comunidade de Engenharia de Software para prover melhora na classificação de requisitos;
- **Experimento de classificação de requisitos utilizando um novo método e comparação com os métodos estabelecidos**: o experimento realizado demonstrou que modelos baseados em Transformers são estatisticamente superiores aos algoritmos de aprendizado de máquina tradicionais. Tal resultado pode ensejar mais pesquisas utilizando estes modelos;
- **Criação de uma base para similaridade e experimento em similaridade de requisitos**: a base ReqFuncSimDataset agrega requisitos divididos em funcionalidade e pode ser utilizada para o treinamento de computação de similaridade semântica entre requisitos. O experimento conduzido demonstrou que a base proveu melhora para a maioria dos modelos pré-treinados. Após treinamento, esses modelos mostraram-se mais eficientes na computação de similaridade semântica em comparação com a aplicação tradicional com TF-IDF;
- **Abordagem de agrupamento de requisitos considerando características de IoT**: ao focar em aplicações de IoT, este trabalho apresenta uma abordagem nova para agrupamento de requisitos, o que pode servir para novos trabalhos focarem em contextos específicos de software.
- **Produção acadêmica**:
  - Publicação do artigo *"Promise+: expandindo a base de dados de requisitos de software Promise_exp"* In: SIMPÓSIO BRASILEIRO DE ENGENHARIA DE SOFTWARE (SBES), ocorrido em Curitiba,PR, Brasil (SILVA et al., 2024).

  - Citação completa:

    SILVA, Bruno; NASCIMENTO, Rodrigo; RIVERO, Luis; BRAZ, Geraldo; SANTOS, Rodrigo Pereira dos; MARTINS, Luiz E. G.; VIANA, Davi. Promise+: expandindo a base de dados de requisitos de software Promise_exp. In: SIMPÓSIO BRASILEIRO DE ENGENHARIA DE SOFTWARE (SBES), 38. , 2024, Curitiba/PR. Anais [...]. Porto Alegre: Sociedade Brasileira de Computação, 2024 . p. 291-301. DOI: <https://doi.org/10.5753/sbes.2024.3427.>

- Outra publicação realizada durante esta dissertação:

  - Artigo *"Assisting the Requirements Definition and Modeling of IoT applications through the ReqM4IoT tool"* In: SIMPÓSIO BRASILEIRO DE QUALIDADE DE SOFTWARE (SBQS), ocorrido em Salvador, BA, Brasil (NASCIMENTO et al., 2024).

  - Citação completa:

    NASCIMENTO, Rodrigo; SILVA, Bruno; AVELINO, Guilherme; VIANA, Davi. Assisting the Requirements Definition and Modeling of IoT applications through the ReqM4IoT tool. In: SIMPÓSIO BRASILEIRO DE QUALIDADE DE SOFTWARE (SBQS), 23. , 2024, Bahia/BA. Anais [...]. Porto Alegre: Sociedade Brasileira de Computação, 2024 . p. 286–297.

6.2 Limitações e ameaças à validade

Embora esta pesquisa tenha apresentado resultados positivos, é fundamental reconhecer suas limitações e possíveis ameaças à validade para permitir uma análise crítica de suas descobertas e contribuições. As ameaças identificadas foram classificadas em validade interna, externa, de construto e de conclusão, conforme descrito a seguir.

- **Validade Interna**: na pesquisa de opinião (Seção 5.2), os participantes receberam um vídeo explicativo sobre o método ReqCluster4IoT, o que pode ter influenciado nas respostas dos participantes. Para mitigar esta ameaça, foi seguido um roteiro de apresentação, supervisionado pelo orientador, que continha somente informações gerais sobre o método, visando somente alinhar conceitos. No grupo focal (Seção 5.3), um fator que pode influenciar a validade interna é o viés do pesquisador, pois a condução foi feita pelo pesquisador desta dissertação, o que pode gerar viés intencional nas respostas. Para mitigar esta ameaça, a condução foi feita em cima dos resultados prévios dos participantes sobre o pertencimento dos requisitos

aos grupos. Ademais, os participantes foram encorajados a refletirem sobre suas respostas durante a condução do grupo. Destaca-se também que a base de requisitos ReqFuncSimDataset (Seção 4.2) não foi avaliada antes de seu uso. Para diminuir esta ameaça, foram utilizados requisitos encontrados em uma base de dados já utilizada em outros trabalhos de engenharia de requisitos;

- **Validade externa**: Tanto a pesquisa de opinião (Seção 5.2), com dez participantes, quanto o grupo focal (Seção 5.3) com quatro participantes, representam uma amostra pequena de participantes, representam uma ameaça à validade externa. Para além do número de participantes, o método foi testado somente em um único cenário do domínio de IoT, podendo não ser representativo para os demais domínios. Para mitigar esta ameaça, foram selecionados profissionais e experts com diferente experiências tanto no desenvolvimento de software quanto em engenharia de requisitos. A realização de dois estudos, um focando nas percepções de desenvolvedores e outro focando nas percepções de engenheiros de requisitos, contribuiu para a coleta de informação de diferentes perspectivas sobre os resultados do método;

- **Validade de Construto**: A clareza do formulário de avaliação dos grupos (Apêndice D), que foram o primeiro foco de avaliação, correu de alguns construtos não terem sido bem entendidos. As diferentes familiaridades com conceitos de IoT e engenharia de requisitos, pode levar a variações de interpretações e respostas. Para mitigar esta ameaça foi disponibilizado um vídeo para os participantes da pesquisa de opinião e uma apresentação para os participantes do grupo focal sobre os principais conceitos envolvidos no método;

- **Validade de Conclusão**: as diferentes respostas obtidas a partir de participantes com diferentes experiências e interpretações sobre os grupos de requisitos, tanto na pesquisa de opinião quanto no grupo focal, pode ter causado impacto nas conclusões realizadas. Ademais, para os participantes do grupo focal, pode ter havido pressão por aceitação da opinião dominante do grupo. Para mitigar essas ameaças, foram fornecidas informações sobre o método e sobre os diferentes conceitos utilizados no método. No grupo focal, foi encorajada a discussão aberta, dando destaque para cada opinião dada e instigando a participação.

A principal limitação do método foi se restringir aos requisitos funcionais. Esta decisão foi tomada, pois o método foca na realização do agrupamento de requisitos, sendo isso possível somente para requisitos funcionais. No entanto, requisitos não funcionais são importantes para a construção do software, uma vez que no contexto de IoT estas características podem ser críticas, como aspectos de conectividade, tempo de resposta e disponibilidade.

Outra limitação do estudo foi quanto a avaliação, uma vez que ambos os experimentos foram feitos a partir de um contexto único e com requisitos gerados a partir de um LLM. É importante que o método possa ser aplicado nos mais diferentes contextos e com requisitos de projetos reais, com requisitos criados por humanos que podem ser ambíguos e incompletos.

6.3 Trabalhos Futuros

A atual versão da ReqCluster4IoT ainda não conta com um apoio ferramental, contudo uma ferramenta já está em andamento para ser uma plataforma online. Nesta aplicação, os usuários poderão gerenciar seus requisitos e visualizar os resultados dos agrupamentos. Tal implementação será importante para a facilitação da utilização do método.

Uma evolução possível para o método envolve a inclusão de requisitos não funcionais em consideração na análise dos grupos. Diferentemente dos requisitos funcionais, os requisitos não funcionais abrangem aspectos como desempenho, segurança, usabilidade e escalabilidade. Esses requisitos frequentemente possuem um caráter transversal, podendo impactar múltiplos grupos de requisitos funcionais. Assim, é necessário desenvolver um mecanismo que permita identificar e associar requisitos não funcionais a um ou mais grupos identificados.

Outra análise a ser feita é comparar os resultados obtidos com o método com os obtidos com modelos como o ChatGPT. O objetivo desta análise é verificar se o método desenvolvido consegue superar os modelos de linguagem existentes. É importante salientar que a exposição de dados sensíveis, como requisitos, a estas aplicações pode ser uma vulnerabilidade que deve ser considerada.

Adicionalmente, outro processo de possível melhoria seria o desenvolvimento de um método de escolha otimizada para o número de *clusters*. Atualmente, este processo envolve o corte de dendrogramas a partir da diferença de pontos de fusão para a escolha dos números em potencial e utilização do índice de silhouette para a determinação do número final. Este processo poderia passar por melhorias utilizando busca otimizada.

Por fim, um aspecto promissor para investigações futuras envolve a conexão entre os grupos de requisitos funcionais e a geração automática de código utilizando LLMs. A partir dos agrupamentos identificados, seria possível explorar como esses modelos poderiam auxiliar no processo de desenvolvimento de software, gerando automaticamente trechos de código correspondentes a cada conjunto de requisitos. Esse processo poderia acelerar o desenvolvimento e garantir maior aderência do código gerado às necessidades especificadas.

Referências

ABBAS, M.; FERRARI, A.; SHATNAWI, A.; ENOIU, E.; SAADATMAND, M.; SUNDMARK, D. On the relationship between similar requirements and similar software. *Requirements Engineering*, v. 28, p. 1–25, 01 2022. Quoted 2 times on page(s) 29 and 77.

AL-ANZI, F. S. An effective hybrid stochastic gradient descent for classification of short text communication in e- learning environments. In: *2022 8th International Conference on Control, Decision and Information Technologies (CoDIT)*. [S.l.: s.n.], 2022. v. 1, p. 1096–1101. Quoted on page 25.

AL-OTAIBY, T. N.; ALSHERIF, M.; BOND, W. P. Toward software requirements modularization using hierarchical clustering techniques. In: *Proceedings of the 43rd Annual Southeast Regional Conference - Volume 2*. New York, NY, USA: Association for Computing Machinery, 2005. (ACM-SE 43), p. 223–228. ISBN 1595930590. Disponível em: <https://doi.org/10.1145/1167253.1167305>. Quoted 5 times on page(s) 16, 35, 37, 38, and 84.

ALNAJEM, N. A.; BINKHONAIN, M.; HOSSAIN, M. S. Siamese neural networks method for semantic requirements similarity detection. *IEEE Access*, v. 12, p. 140932–140947, 2024. Quoted on page 61.

ARYA, N.; NIGAM, B.; NIGAM, A. Tool for automatic discovery of ambiguity in requirements. 09 2012. Quoted on page 14.

BROMLEY, J.; BENTZ, J.; BOTTOU, L.; GUYON, I.; LECUN, Y.; MOORE, C.; SACKINGER, E.; SHAH, R. Signature verification using a "siamese" time delay neural network. *International Journal of Pattern Recognition and Artificial Intelligence*, v. 7, p. 25, 08 1993. Quoted on page 58.

CANEDO, E. D.; MENDES, B. C. Software requirements classification using machine learning algorithms. *Entropy*, v. 22, n. 9, 2020. ISSN 1099-4300. Disponível em: <https://www.mdpi.com/1099-4300/22/9/1057>. Quoted 4 times on page(s) 38, 41, 50, and 57.

CARLINI, N.; TRAMÈR, F.; WALLACE, E.; JAGIELSKI, M.; HERBERT-VOSS, A.; LEE, K.; ROBERTS, A.; BROWN, T. B.; SONG, D.; ERLINGSSON, Ú.; OPREA, A.; RAFFEL, C. Extracting training data from large language models. *CoRR*, abs/2012.07805, 2020. Disponível em: <https://arxiv.org/abs/2012.07805>. Quoted on page 70.

CASAMAYOR, A.; GODOY, D.; CAMPO, M. Functional grouping of natural language requirements for assistance in architectural software design. *Knowledge-Based Systems*, v. 30, p. 78–86, 2012. ISSN 0950-7051. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0950705111002759>. Quoted on page 16.

CASAMAYOR, A.; GODOY, D.; CAMPO, M. Functional grouping of natural language requirements for assistance in architectural software design. *Knowledge-Based Systems*, v. 30, p. 78–86, 2012. ISSN 0950-7051. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0950705111002759>. Quoted 3 times on page(s) 34, 37, and 38.

COHEN, J. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, v. 20, p. 37 – 46, 1960. Disponível em: <https://api.semanticscholar.org/CorpusID:15926286>. Quoted 2 times on page(s) 43 and 45.

CORBIN, J.; STRAUSS, A. *Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory*. SAGE Publications, 2014. ISBN 9781483315683. Disponível em: <https://books.google.com.br/books?id=hZ6kBQAAQBAJ>. Quoted on page 78.

DAVIS, F.; DAVIS, F. Perceived usefulness, perceived ease of use, and user acceptance of information technology. *MIS Quarterly*, v. 13, p. 319–, 09 1989. Quoted 2 times on page(s) 71 and 77.

DEVLIN, J.; CHANG, M.; LEE, K.; TOUTANOVA, K. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018. Disponível em: <http://arxiv.org/abs/1810.04805>. Quoted on page 38.

FAHMIDEH, M.; AHMAD, A.; BEHNAZ, A.; GRUNDY, J.; SUSILO, W. Software engineering for internet of things: The practitioners' perspective. *IEEE Transactions on Software Engineering*, v. 48, n. 8, p. 2857–2878, 2022. Quoted 2 times on page(s) 14 and 15.

FEDELE, A.; GUIDOTTI, R.; PEDRESCHI, D. Explaining siamese networks in few-shot learning. *Machine Learning*, v. 113, p. 7723–7760, 04 2024. Quoted 2 times on page(s) 58 and 61.

FERRARI, A.; SPAGNOLO, G. O.; GNESI, S. Pure: A dataset of public requirements documents. In: *2017 IEEE 25th International Requirements Engineering Conference (RE)*. [S.l.: s.n.], 2017. p. 502–505. Quoted 3 times on page(s) 9, 44, and 60.

FIROUZI, F.; FARAHANI, B.; WEINBERGER, M.; DEPACE, G.; ALIEE, F. S. Iot fundamentals: Definitions, architectures, challenges, and promises. In: _____. *Intelligent Internet of Things: From Device to Fog and Cloud*. Cham: Springer International Publishing, 2020. p. 3–50. ISBN 978-3-030-30367-9. Disponível em: <https://doi.org/10.1007/978-3-030-30367-9_1>. Quoted on page 21.

FRANCH, X.; PALOMARES, C.; QUER, C.; CHATZIPETROU, P.; GORSCHEK, T. The state-of-practice in requirements specification: an extended interview study at 12 companies. *Requirements Engineering*, v. 28, n. 3, p. 377–409, Sep 2023. ISSN 1432-010X. Disponível em: <https://doi.org/10.1007/s00766-023-00399-7>. Quoted 2 times on page(s) 15 and 37.

GALSTER, M.; EBERLEIN, A.; JIANG, L. Structuring software requirements for architecture design. In: *2013 20th IEEE International Conference and Workshops on Engineering of Computer Based Systems (ECBS)*. [S.l.: s.n.], 2013. p. 119–128. Quoted 5 times on page(s) 16, 34, 37, 38, and 84.

GERON, A. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. 2nd. ed. [S.l.]: O'Reilly Media, Inc., 2019. ISBN 1492032646. Quoted on page 48.

GIRAY, G.; TEKINERDOGAN, B.; TüZüN, E. Iot system development methods. In: _____. [S.l.]: 9781498778510, 2018. p. 141–159. Quoted on page 14.

GONG, Z.; ZHONG, P.; HU, W. Diversity in machine learning. *IEEE Access*, v. 7, p. 64323–64350, 2019. Quoted on page 55.

GRäßLER, I.; PREUß, D.; BRANDT, L.; MOHR, M. Efficient extraction of technical requirements applying data augmentation. In: *2022 IEEE International Symposium on Systems Engineering (ISSE)*. [S.l.: s.n.], 2022. p. 1–8. Quoted on page 69.

HENNINK, M.; KAISER, B. N. Sample sizes for saturation in qualitative research: A systematic review of empirical tests. *Social Science Medicine*, v. 292, p. 114523, 2022. ISSN 0277-9536. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0277953621008558>. Quoted on page 76.

HERSMAN, C.; FOWLER, K. Chapter 5 - best practices in spacecraft development. In: FOWLER, K. (Ed.). *Mission-Critical and Safety-Critical Systems Handbook*. Boston: Newnes, 2010. p. 269–460. ISBN 978-0-7506-8567-2. Disponível em: <https://www.sciencedirect.com/science/article/pii/B9780750685672000056>. Quoted on page 19.

HOCHREITER, S.; SCHMIDHUBER, J. Long short-term memory. *Neural Computation*, v. 9, n. 8, p. 1735–1780, 11 1997. ISSN 0899-7667. Disponível em: <https://doi.org/10.1162/neco.1997.9.8.1735>. Quoted on page 26.

HOU, X.; ZHAO, Y.; LIU, Y.; YANG, Z.; WANG, K.; LI, L.; LUO, X.; LO, D.; GRUNDY, J.; WANG, H. *Large Language Models for Software Engineering: A Systematic Literature Review*. 2024. Disponível em: <https://arxiv.org/abs/2308.10620>. Quoted 2 times on page(s) 28 and 69.

HOWARD, J.; RUDER, S. Universal language model fine-tuning for text classification. In: GUREVYCH, I.; MIYAO, Y. (Ed.). *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Melbourne, Australia: Association for Computational Linguistics, 2018. p. 328–339. Disponível em: <https://aclanthology.org/P18-1031/>. Quoted on page 28.

HUANG, J.; KEUNG, J. W.; SARRO, F.; LI, Y.-F.; YU, Y.-T.; CHAN, W.; SUN, H. Cross-validation based k nearest neighbor imputation for software quality datasets: an empirical study. *Journal of Systems and Software*, Elsevier, v. 132, p. 226–252, 2017. Quoted on page 26.

IEEE 830. Ieee recommended practice for software requirements specifications. *IEEE Std 830-1998*, p. 1–40, 1998. Quoted on page 42.

IEEE Computer Society Staff. *SWEBOK Guide V4.0*. 445 Hoes Lane, Piscataway, NJ 08854-4141: IEEE Computer Society, 2024. Disponível em: <https://www.swebok.org>. Quoted 2 times on page(s) 19 and 37.

ISO/IEC/IEEE 29148. Iso/iec/ieee international standard - systems and software engineering – life cycle processes – requirements engineering. *ISO/IEC/IEEE 29148:2018(E)*, p. 1–104, 2018. Quoted 2 times on page(s) 14 and 42.

JAIN, A. K.; DUBES, R. C. *Algorithms for Clustering Data*. USA: Prentice-Hall, Inc., 1988. ISBN 013022278X. Quoted on page 29.

JOHNSON, R. A.; WICHERN, D. W. Applied multivariate statistical analysis. Prentice hall Upper Saddle River, NJ, 2002. Quoted on page 31.

KAUR, K.; KAUR, P. Improving bert model for requirements classification by bidirectional lstm-cnn deep model. *Computers and Electrical Engineering*, v. 108, p. 108699, 2023. ISSN 0045-7906. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0045790623001234>. Quoted on page 50.

KAUR, K.; KAUR, P. The application of ai techniques in requirements classification: a systematic mapping. *Artificial Intelligence Review*, v. 57, 02 2024. Quoted 5 times on page(s) 23, 24, 25, 26, and 41.

KHANNA, A.; KAUR, S. Internet of things (iot), applications and challenges: A comprehensive review. *Wireless Personal Communications*, v. 114, p. 1–76, 09 2020. Quoted 3 times on page(s) 15, 20, and 84.

KIRKPATRICK, J.; PASCANU, R.; RABINOWITZ, N. C.; VENESS, J.; DESJARDINS, G.; RUSU, A. A.; MILAN, K.; QUAN, J.; RAMALHO, T.; GRABSKA-BARWINSKA, A.; HASSABIS, D.; CLOPATH, C.; KUMARAN, D.; HADSELL, R. Overcoming catastrophic forgetting in neural networks. *CoRR*, abs/1612.00796, 2016. Disponível em: <http://arxiv.org/abs/1612.00796>. Quoted on page 28.

KOCHBATI, T.; GéRARD, S.; LI, S.; MRAIDHA, C. From word embeddings to text similarities for improved semantic clustering of functional requirements. *Proceedings of the International Conference on Software Engineering and Knowledge Engineering, SEKE 2021*, p. 285–290, 2021. Quoted 5 times on page(s) 15, 36, 37, 38, and 65.

KONTIO, J.; LEHTOLA, L.; BRAGGE, J. Using the focus group method in software engineering: obtaining practitioner and user experiences. In: *Proceedings. 2004 International Symposium on Empirical Software Engineering, 2004. ISESE '04.* [S.l.: s.n.], 2004. p. 271–280. Quoted on page 76.

KORA, R.; MOHAMMED, A. A comprehensive review on transformers models for text classification. In: *2023 International Mobile, Intelligent, and Ubiquitous Computing Conference (MIUCC).* [S.l.: s.n.], 2023. p. 1–7. Quoted on page 50.

KRIEGEL, H.-P.; KRöGER, P.; SANDER, J.; ZIMEK, A. Density-based clustering. *WIREs Data Mining and Knowledge Discovery*, v. 1, n. 3, p. 231–240, 2011. Disponível em: <https://wires.onlinelibrary.wiley.com/doi/abs/10.1002/widm.30>. Quoted on page 30.

KUHN, A.; DUCASSE, S.; GîRBA, T. Semantic clustering: Identifying topics in source code. *Information and Software Technology*, v. 49, n. 3, p. 230–243, 2007. ISSN 0950-5849. 12th Working Conference on Reverse Engineering. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0950584906001820>. Quoted on page 16.

KUMAR, B.; TIWARI, U. K.; DOBHAL, D. C.; NEGI, H. S. User story clustering using k-means algorithm in agile requirement engineering. In: *2022 International Conference on Computational Intelligence and Sustainable Engineering Solutions (CISES).* [S.l.: s.n.], 2022. p. 1–5. Quoted 3 times on page(s) 35, 37, and 38.

KUMAR, L.; BALDWA, S.; JAMBAVALIKAR, S. M.; MURTHY, L. B.; KRISHNA, A. Software functional and non-function requirement classification using word-embedding. In: BAROLLI, L.; HUSSAIN, F.; ENOKIDO, T. (Ed.). *Advanced Information Networking and Applications*. Cham: Springer International Publishing, 2022. p. 167–179. ISBN 978-3-030-99587-4. Quoted on page 25.

LEI, W.; MENG, Z. Text similarity calculation method of siamese network based on albert. In: *2022 International Conference on Machine Learning and Knowledge Engineering (MLKE)*. [S.l.: s.n.], 2022. p. 251–255. Quoted on page 58.

LIMA, M.; VALLE, V.; COSTA, E.; LIRA, F.; GADELHA, B. Software engineering repositories: Expanding the promise database. In: . [S.l.: s.n.], 2019. p. 427–436. ISBN 978-1-4503-7651-8. Quoted 3 times on page(s) 41, 42, and 43.

LIPTON, Z. C.; ELKAN, C.; NARYANASWAMY, B. Optimal thresholding of classifiers to maximize f1 measure. In: *Machine Learning and Knowledge Discovery in Databases*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014. p. 225–239. ISBN 978-3-662-44851-9. Quoted on page 48.

MELEKHOV, I.; KANNALA, J.; RAHTU, E. Siamese network features for image matching. In: *2016 23rd International Conference on Pattern Recognition (ICPR)*. [S.l.: s.n.], 2016. p. 378–383. Quoted 2 times on page(s) 58 and 61.

MISRA, J.; SENGUPTA, S.; PODDER, S. Topic cohesion preserving requirements clustering. In: *Proceedings of the 5th International Workshop on Realizing Artificial Intelligence Synergies in Software Engineering*. New York, NY, USA: Association for Computing Machinery, 2016. (RAISE '16), p. 22–28. ISBN 9781450341653. Disponível em: <https://doi.org/10.1145/2896995.2896998>. Quoted 3 times on page(s) 35, 37, and 38.

MISRA, N. N.; DIXIT, Y.; AL-MALLAHI, A.; BHULLAR, M. S.; UPADHYAY, R.; MARTYNENKO, A. Iot, big data, and artificial intelligence in agriculture and food industry. *IEEE Internet of Things Journal*, v. 9, n. 9, p. 6305–6324, 2022. Quoted 2 times on page(s) 15 and 16.

MOLLéRI, J. S.; PETERSEN, K.; MENDES, E. Survey guidelines in software engineering: An annotated review. In: *Proceedings of the 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*. New York, NY, USA: Association for Computing Machinery, 2016. (ESEM '16). ISBN 9781450344272. Disponível em: <https://doi.org/10.1145/2961111.2962619>. Quoted on page 70.

MOTTA, R.; OLIVEIRA, K.; TRAVASSOS, G. On challenges in engineering iot software systems. *Journal of Software Engineering Research and Development*, v. 7, p. 5, 09 2019. Quoted 2 times on page(s) 15 and 21.

MURALIDHARAN, S.; ROY, A.; SAXENA, N. Mdp-iot: Mdp based interest forwarding for heterogeneous traffic in iot-ndn environment. *Future Generation Computer Systems*, v. 79, p. 892–908, 2018. ISSN 0167-739X. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0167739X1731035X>. Quoted on page 14.

MURTAGH, F.; CONTRERAS, P. Algorithms for hierarchical clustering: an overview, ii. *WIREs Data Mining and Knowledge Discovery*, v. 7, n. 6, p. e1219, 2017. Disponível em:

<https://wires.onlinelibrary.wiley.com/doi/abs/10.1002/widm.1219>. Quoted on page 31.

NASCIMENTO, R.; SILVA, B.; AVELINO, G.; VIANA, D. Assisting the requirements definition and modeling of iot applications through the reqm4iot tool. In: *Anais do XXIII Simpósio Brasileiro de Qualidade de Software*. Porto Alegre, RS, Brasil: SBC, 2024. p. 286–297. ISSN 0000-0000. Disponível em: <https://sol.sbc.org.br/index.php/sbqs/article/view/32956>. Quoted on page 88.

NAVARRO-ALMANZA, R.; JUAREZ-RAMIREZ, R.; LICEA, G. Towards supporting software engineering using deep learning: A case of software requirements classification. In: *2017 5th International Conference in Software Engineering Research and Innovation (CONISOFT)*. [S.l.: s.n.], 2017. p. 116–120. Quoted on page 41.

OTI, E.; OLUSOLA, M. Overview of agglomerative hierarchical clustering methods. *British Journal of Computer, Networking and Information Technology*, v. 7, p. 14–23, 06 2024. Quoted on page 31.

PATIL, R.; BOIT, S.; GUDIVADA, V.; NANDIGAM, J. A survey of text representation and embedding techniques in nlp. *IEEE Access*, v. 11, p. 36120–36146, 2023. Quoted on page 26.

PRESSMAN, R.; MAXIM, B. *Engenharia de Software - 8ª Edição*. [s.n.], 2016. ISBN 9788580555349. Disponível em: <https://books.google.com.br/books?id=wexzCwAAQBAJ>. Quoted on page 44.

PUNTER, T.; CIOLKOWSKI, M.; FREIMUT, B.; JOHN, I. Conducting on-line surveys in software engineering. In: *2003 International Symposium on Empirical Software Engineering, 2003. ISESE 2003. Proceedings.* [S.l.: s.n.], 2003. p. 80–88. Quoted on page 70.

QUBA, G. Y.; QAISI, H. A.; ALTHUNIBAT, A.; ALZU'BI, S. Software requirements classification using machine learning algorithm's. In: *2021 International Conference on Information Technology (ICIT)*. [S.l.: s.n.], 2021. p. 685–690. Quoted 3 times on page(s) 38, 41, and 57.

RAHMAN, K.; GHANI, A.; ALZAHRANI, A.; TARIQ, M. U.; RAHMAN, A. U. Pre-trained model-based nfr classification: Overcoming limited data challenges. *IEEE Access*, v. 11, p. 81787–81802, 2023. Quoted on page 60.

RAINIO, O.; TEUHO, J.; KLéN, R. Author correction: Evaluation metrics and statistical tests for machine learning. *Scientific Reports*, v. 14, 07 2024. Quoted on page 52.

RANASINGHE, T.; ORASAN, C.; MITKOV, R. Semantic textual similarity with Siamese neural networks. In: MITKOV, R.; ANGELOVA, G. (Ed.). *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2019)*. Varna, Bulgaria: INCOMA Ltd., 2019. p. 1004–1011. Disponível em: <https://aclanthology.org/R19-1116/>. Quoted on page 58.

REIMERS, N.; GUREVYCH, I. Sentence-bert: Sentence embeddings using siamese bert-networks. *CoRR*, abs/1908.10084, 2019. Disponível em: <http://arxiv.org/abs/1908.10084>. Quoted on page 38.

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Learning representations by back-propagating errors. *Nature*, v. 323, p. 533–536, 1986. Disponível em: <https://api.semanticscholar.org/CorpusID:205001834>. Quoted on page 26.

RYU, D.; CHOI, O.; BAIK, J. Value-cognitive boosting with a support vector machine for cross-project defect prediction. *Empirical Software Engineering*, Springer, v. 21, p. 43–71, 2016. Quoted on page 23.

SALMAN, H. E.; HAMMAD, M.; SERIAI, A.-D.; AL-SBOU, A. Semantic clustering of functional requirements using agglomerative hierarchical clustering. *Information*, v. 9, n. 9, 2018. ISSN 2078-2489. Disponível em: <https://www.mdpi.com/2078-2489/9/9/222>. Quoted 9 times on page(s) 9, 16, 32, 33, 36, 37, 38, 65, and 84.

SCHROFF, F.; KALENICHENKO, D.; PHILBIN, J. Facenet: A unified embedding for face recognition and clustering. In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. [S.l.: s.n.], 2015. p. 815–823. Quoted 2 times on page(s) 58 and 59.

SHIRABAD, J. S.; MENZIES, T. *The PROMISE Repository of Software Engineering Databases*. 2005. School of Information Technology and Engineering, University of Ottawa, Canada. Disponível em: <http://promise.site.uottawa.ca/SERepository>. Quoted on page 41.

SHREDA, Q. A.; HANANI, A. A. Identifying non-functional requirements from unconstrained documents using natural language processing and machine learning approaches. *IEEE Access*, p. 1–1, 2021. Quoted on page 60.

SHTERN, M.; TZERPOS, V. Clustering methodologies for software engineering. *Adv. Soft. Eng.*, Hindawi Limited, London, GBR, v. 2012, jan 2012. ISSN 1687-8655. Disponível em: <https://doi.org/10.1155/2012/792024>. Quoted on page 16.

SIAKAS, E.; LAMPROPOULOS, G.; RAHANU, H.; GEORGIADOU, E.; SIAKAS, D.; SIAKAS, K. Refiot: A framework to combat requirements engineering in iot applications and systems. In: *Systems, Software and Services Process Improvement*. Cham: Springer Nature Switzerland, 2024. p. 80–96. ISBN 978-3-031-71139-8. Quoted on page 19.

SILVA, B.; NASCIMENTO, R.; RIVERO, L.; BRAZ, G.; SANTOS, R.; MARTINS, L.; VIANA, D. Promise+: expandindo a base de dados de requisitos de software $promise_e xp.In$: $Anais\ do\ XXXVIII\ Simpósio\ Brasileiro\ de\ Engenharia\ de\ Software.PortoAlegre, RS, Brasil$: $SBC, 2024.p.291 - -301.ISSN0000 - 0000.Disponívelem$: <>. Quoted 4 times on page(s) 41, 42, 60, and 87.

SILVA, D.; GONçALVES, T. G.; ROCHA, A. R. C. da. A requirements engineering process for iot systems. In: *Proceedings of the XVIII Brazilian Symposium on Software Quality*. New York, NY, USA: Association for Computing Machinery, 2019. (SBQS '19), p. 204–209. ISBN 9781450372824. Disponível em: <https://doi.org/10.1145/3364641.3364664>. Quoted 2 times on page(s) 15 and 84.

SILVA, D. V. da; SOUZA, B. P. de; GONÇALVES, T. G.; TRAVASSOS, G. H. A requirements engineering technology for the iot software systems. *Journal of Software Engineering Research and Development*, 2021. Quoted on page 14.

SILVA GEOVANE MIGUEL DA SILVA, G. F. a. d. M. F. C. M. B. L. M. Karolayne Teixeira da; MADEIRO, F. Algoritmos de aprendizagem supervisionada com conjuntos de dados desbalanceados para classificação de requisitos não-funcionais. In: *Anais do 15 Congresso Brasileiro de Inteligência Computacional*. Joinville, SC: SBIC, 2021. p. 1–7. Quoted 2 times on page(s) 50 and 57.

SINGH, A.; KUMAR, S. A comparison of machine learning algorithms and transformer-based methods for multiclass sentiment analysis on twitter. In: *2023 14th International Conference on Computing Communication and Networking Technologies (ICCCNT)*. [S.l.: s.n.], 2023. p. 1–9. Quoted on page 50.

SORRI, K.; MUSTAFEE, N.; SEPPANEN, M. Revisiting iot definitions: A framework towards comprehensive use. *Technological Forecasting and Social Change*, v. 179, p. 121623, 2022. ISSN 0040-1625. Disponível em: <https://www.sciencedirect.com/science/article/pii/S004016252200155X>. Quoted 2 times on page(s) 21 and 22.

SOUZA, S.; RODRIGUES, E.; MEIRELES, M.; LAUSCHNER, T.; CARVALHO, L.; MALDONADO, J. C.; CONTE, T. Techniques for eliciting iot requirements: Sensorina map and mind iot. *Journal of Systems and Software*, v. 222, p. 112323, 2025. ISSN 0164-1212. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0164121224003674>. Quoted on page 15.

STOLOJESCU-CRISAN, C.; CRISAN, C.; BUTUNOI, B.-P. An iot-based smart home automation system. *Sensors*, v. 21, n. 11, 2021. ISSN 1424-8220. Disponível em: <https://www.mdpi.com/1424-8220/21/11/3784>. Quoted on page 69.

SUTCLIFFE, A.; GULLIKSEN, J. Chapter 18 - user-centered requirements definition. In: BUIE, E.; MURRAY, D. (Ed.). *Usability in Government Systems*. Boston: Morgan Kaufmann, 2012. p. 285–300. ISBN 978-0-12-391063-9. Disponível em: <https://www.sciencedirect.com/science/article/pii/B978012391063900050X>. Quoted on page 19.

TALELE, P.; APTE, S.; PHALNIKAR, R.; TALELE, H. Semi-automated software requirements categorisation using machine learning algorithms. *International Journal of Electrical and Computer Engineering Systems*, v. 14, n. 10, p. 1107–1114, Nov. 2023. Disponível em: <https://ijeces.ferit.hr/index.php/ijeces/article/view/2711>. Quoted on page 60.

TAYE, M. M. Understanding of machine learning with deep learning: Architectures, workflow, applications and future directions. *Computers*, v. 12, n. 5, 2023. ISSN 2073-

431X. Disponível em: <https://www.mdpi.com/2073-431X/12/5/91>. Quoted on page 41.

TIWARI, S.; RATHORE, S. S.; SAGAR, S.; MIRANI, Y. Identifying use case elements from textual specification: A preliminary study. In: IEEE. *2020 IEEE 28th International Requirements Engineering Conference (RE)*. [S.l.], 2020. p. 410–411. Quoted on page 25.

UPPADA, S. K. Centroid based clustering algorithms-a clarion study. In: . [s.n.], 2014. Disponível em: <https://api.semanticscholar.org/CorpusID:1649872>. Quoted on page 30.

VASWANI, A.; SHAZEER, N.; PARMAR, N.; USZKOREIT, J.; JONES, L.; GOMEZ, A. N.; KAISER, L.; POLOSUKHIN, I. Attention is all you need. *CoRR*, abs/1706.03762, 2017. Disponível em: <http://arxiv.org/abs/1706.03762>. Quoted 2 times on page(s) 9 and 27.

WANG, W.; HUSSEIN, N.; GUPTA, A.; WANG, Y. A regression model based approach for identifying security requirements in open source software development. In: IEEE. *2017 IEEE 25th International Requirements Engineering Conference Workshops (REW)*. [S.l.], 2017. p. 443–446. Quoted on page 24.

WHITMORE, A.; AGARWAL, A.; XU, L. The internet of things—a survey of topics and trends. *Information Systems Frontiers*, v. 17, 04 2014. Quoted on page 20.

WOLPERT, D.; MACREADY, W. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, v. 1, n. 1, p. 67–82, 1997. Quoted on page 55.

XU, S.; LI, Y.; WANG, Z. Bayesian multinomial naïve bayes classifier to text classification. In: *Advanced Multimedia and Ubiquitous Engineering*. Singapore: Springer Singapore, 2017. p. 347–352. ISBN 978-981-10-5041-1. Quoted on page 24.

ZHAO, L.; ALHOSHAN, W.; FERRARI, A.; LETSHOLO, K. J.; AJAGBE, M. A.; CHIOASCA, E.-V.; BATISTA-NAVARRO, R. T. Natural language processing for requirements engineering: A systematic mapping study. *ACM Comput. Surv.*, Association for Computing Machinery, New York, NY, USA, v. 54, n. 3, apr 2021. ISSN 0360-0300. Disponível em: <https://doi.org/10.1145/3444689>. Quoted on page 15.

Apêndices

A Termo de Consentimento Livre e Esclarecido (TCLE)

Caro participante,

As informações listadas abaixo estão sendo fornecidas para sua participação voluntária neste estudo. As informações obtidas serão analisadas juntamente com as de outros participantes da pesquisa, garantindo confidencialidade das informações obtidas durante o trabalho. Ademais, Declaro que autorizo a disponibilização da transcrição da minha entrevista e/ou respostas de forma anônima no contexto da pesquisa intitulada "ReqCluster4IoT: Um método de agrupamento de requisitos funcionais para aplicações de IoT".

Esta transcrição segue as regras de privacidade dos participantes no Termo de Consentimento Livre e Esclarecido. Os pesquisadores mantêm a privacidade dos participantes sobre informações pessoais, como nome, iniciais, local de trabalho e país. Esta transcrição estará disponível em um repositório de acesso aberto (como ZENODO, FigShare, Github e entre outros)

1) OBJETIVO

Investigar a percepção dos resultados encontrados pela ReqCluster4IoT, com base em uma perspectiva intuitiva de Engenheiros de Requisitos e Desenvolvedores de software para IoT.

2) PROCEDIMENTOS

Os participantes participarão de um grupo focal utilizando a ferramenta Google Meet ou formulário online com perguntas relacionadas aos resultados obtidos pela ReqCluster4IoT, mediante a aplicação em um software proposto.

3) VOLUNTÁRIO

O participante voluntário será acompanhado por pelo menos um pesquisador, e os pesquisadores poderão esclarecer quaisquer dúvidas sobre a pesquisa/grupo focal usando as informações de contato disponíveis ao final deste documento.

4) PRIVACIDADE DOS PARTICIPANTES

Os pesquisadores garantem a confidencialidade em relação às informações obtidas, mantendo assim a privacidade dos participantes..

5) RETIRADA

O voluntário da pesquisa terá a liberdade de se retirar da participação a qualquer momento, mesmo que o trabalho esteja na fase final.

6) DESCONFORTOS ou RISCOS

Os seguintes riscos intelectuais e emocionais podem ser relatados durante a entrevista se o participante notar qualquer uma das seguintes condições e, a seu critério, interromper imediatamente a participação: sentimentos de embaraço, desconforto, medo, vergonha, estresse e cansaço.

7) REEMBOLSO OU COMPENSAÇÃO

Não há despesas pessoais para participar deste estudo, nem há compensação financeira, pois a

pesquisa não sugere nenhum ônus aos participantes.

8) A PESQUISA

O pesquisador tratará sua identidade com padrões profissionais de confidencialidade e integridade. Os resultados da pesquisa estarão disponíveis para você quando concluídos. Seu nome ou material que indique a participação não será divulgado sem sua permissão.

Declaro que fui informado dos objetivos deste trabalho de maneira clara e detalhada e que esclareci minhas dúvidas. Sei que, a qualquer momento, posso solicitar novas informações e mudar minha decisão de participar, se assim desejar.

Se tiver alguma dúvida, entre em contato conosco por e-mail: bruno.carvalho1@discente.ufma.br e davi.viana@ufma.br

Email: _____

Você concorda com este Termo de Consentimento Livre e Esclarecido?

[ ] Concordo [ ] Não concordo

B Formulário de Caracterização

Qual é a sua principal formação educacional?

[ ] Ensino médio [ ] Graduado [ ] Mestrado [ ] Doutorado

Você possui experiência com análise de requisitos?

[ ] 1 - em apenas conhecimento acadêmico

[ ] 2 - em conhecimento acadêmico e um projeto

[ ] 3 - em mais de um projeto e até seis meses de trabalho profissional

[ ] 4 - em mais de um ano de trabalho profissional

Possui experiência em desenvolvimento de software?

[ ] 1 - em apenas conhecimento acadêmico

[ ] 2 - em conhecimento acadêmico e um projeto

[ ] 3 - em mais de um projeto e até seis meses de trabalho profissional

[ ] 4 - em mais de um ano de trabalho profissional

Possui conhecimento sobre IoT ?

[ ] 1 - em apenas conhecimento acadêmico

[ ] 2 - em conhecimento acadêmico e um projeto

[ ] 3 - em mais de um projeto e até seis meses de trabalho profissional

[ ] 4 - em mais de um ano de trabalho profissional

C Requisitos da Aplicação IoT

C.1 Objetivo da avaliação experimental

Avaliar os resultados proporcionados pelo ReqCluster4IoT: método de agrupamento de requisitos funcionais para software de internet of things (IoT). Por resultado entende-se os grupos que são encontrados a partir do uso do ReqCluster4IoT. **O ReqCluster4IoT processa texto em inglês**.

C.2 Requisitos Elicitados

Um sistema de automação residencial baseado em IoT visa monitorar e gerenciar automaticamente diversos aspectos do ambiente doméstico para maximizar a eficiência, segurança e conforto.

Os requisitos do sistema são:

ID	Requisito
RQ-01	The system shall allow users to create accounts and log in using secure authentication methods. Users should be able to assign different access levels (e.g., admin, guest) to various members of the household.
RQ-02	The application should support integration with a wide range of smart home devices, including but not limited to smart lights, thermostats, security cameras, door locks, and appliances. It should provide an easy way for users to connect and configure new devices.
RQ-03	The system shall provide real-time monitoring of all connected smart devices, allowing users to view the current status and settings (e.g., whether lights are on, thermostat temperature, door lock status).
RQ-04	The application shall enable users to remotely control connected devices from anywhere through the internet. Users should be able to turn devices on or off, adjust settings, or activate/deactivate automation rules.
RQ-05	The system shall allow users to create custom automation rules that define how devices should behave based on specific conditions or events (e.g., turning on lights when motion is detected or lowering the thermostat at a set time). Users should be able to combine multiple conditions and actions in their automation scripts.
RQ-06	The system shall support integration with popular voice assistants like Amazon Alexa, Google Assistant, and Apple Siri, allowing users to control devices through voice commands.
RQ-07	The application should provide both mobile and web interfaces that are responsive and user-friendly, allowing users to control smart home devices from any device with internet access.
RQ-08	The system shall send notifications to users about important events, such as security alerts, changes in the status of devices, or failures in automation routines. Notifications should be customizable to allow users to choose the events they want to be notified about.
RQ-09	The application should use encryption to protect data transmitted between the smart devices and the cloud, ensuring the privacy and security of user data. It should also detect and alert users about suspicious activities (e.g., unauthorized access attempts).

RQ-10	The system shall provide features for monitoring energy usage of connected devices and suggest optimization strategies for reducing power consumption. Users should be able to view historical energy consumption data.
RQ-11	The application shall allow users to set schedules for connected devices (e.g., setting the lights to turn on at sunset or the thermostat to adjust temperature based on a daily routine).
RQ-12	The system should support multiple user profiles with different permissions, allowing for personalized automation settings for different members of the household.
RQ-13	The application should be able to connect with third-party services, such as weather forecasting services to adjust heating/cooling based on outdoor temperatures or geolocation services for setting home/away modes.
RQ-14	The system shall allow users to back up their automation rules, device configurations, and user settings. It should also provide a way to restore this data in case of data loss or migration to a new device.
RQ-15	The application should collect and store data from all connected devices for historical analysis. Users should be able to visualize this data through charts or reports to understand patterns in device usage.
RQ-16	The system should implement fail-safe mechanisms to handle situations where connectivity is lost, such as retaining the last known device state or automatically switching to a default state.
RQ-17	The application should include a feedback mechanism for users to report bugs, suggest improvements, and access help resources. A support channel should be available for troubleshooting common issues.
RQ-18	The system shall support various communication protocols (e.g., Wi-Fi, Zigbee, Z-Wave, Bluetooth) to ensure compatibility with a wide range of IoT devices.
RQ-19	The application should notify users about available firmware updates for connected devices and allow them to install updates through the interface. It should also support automatic updates for the application itself.
RQ-20	The system shall provide geofencing features that allow automation to be triggered based on the location of the user's smartphone (e.g., turning on the lights when arriving home).
RQ-21	The system shall achieve a response time of no more than 2 seconds for any user interaction, including logging in, loading dashboards, and executing commands for connected devices.
RQ-22	The application shall have an uptime of 99,9% or higher, with scheduled maintenance periods clearly communicated to users at least 24 hours in advance.
RQ-23	The system shall support a minimum of 1,000 concurrent users without degradation in performance or functionality.
RQ-24	The application interface shall adhere to accessibility standards (e.g., WCAG 2.1 Level AA) to ensure usability by individuals with disabilities.

D Avaliação de Grupos de Requisitos

Caro participante, no seguinte formulário você deverá julgar cada requisito em cada grupo de requisitos inferidos pelo ReqCluster4IoT. Você deve considerar o grupo como um todo e julgar se aquele requisito em específico está condizente com seu grupo.

Você também deverá julgar se a característica de IoT encontrada para aquele grupo é adequada. Ao final de cada julgamento haverá um campo de texto aberto, no qual aconselha-se que o usuário dê mais detalhes do porquê de suas escolhas.

Nome: _____

Legenda:

Faz parte do grupo (FPG), Não faz parte do grupo (NFG) e Não sei opinar (NSO)

Grupo 1 Caracteristica: Services

Requisito	**FPG**	**NFG**	**NSO**
The system shall allow users to create accounts and log in using secure authentication methods. Users should be able to assign different access levels (e.g., admin, guest) to various members of the household.	[ ]	[ ]	[ ]
The application shall enable users to remotely control connected devices from anywhere through the internet. Users should be able to turn devices on or off, adjust settings, or activate/deactivate automation rules.	[ ]	[ ]	[ ]
The application should use encryption to protect data transmitted between the smart devices and the cloud, ensuring the privacy and security of user data. It should also detect and alert users about suspicious activities (e.g., unauthorized access attempts).	[ ]	[ ]	[ ]
The system should support multiple user profiles with different permissions, allowing for personalized automation settings for different members of the household.	[ ]	[ ]	[ ]
The system shall allow users to back up their automation rules, device configurations, and user settings. It should also provide a way to restore this data in case of data loss or migration to a new device.	[ ]	[ ]	[ ]
The application should notify users about available firmware updates for connected devices and allow them to install updates through the interface. It should also support automatic updates for the application itself.	[ ]	[ ]	[ ]

Comentários sobre o Grupo 1: _____

Grupo 2 Caracteristica: Things

Requisito	**FPG**	**NFG**	**NSO**
The application should support integration with a wide range of smart home devices, including but not limited to smart lights, thermostats, security cameras, door locks, and appliances. It should provide an easy way for users to connect and configure new devices.	[ ]	[ ]	[ ]
The application should provide both mobile and web interfaces that are responsive and user-friendly, allowing users to control smart home devices from any device with internet access.	[ ]	[ ]	[ ]
The system shall provide geofencing features that allow automation to be triggered based on the location of the user's smartphone (e.g., turning on the lights when arriving home).	[ ]	[ ]	[ ]

Comentários sobre o Grupo 2: _____

Grupo 3 Caracteristica: Data

Requisito	**FPG**	**NFG**	**NSO**
The system shall provide real-time monitoring of all connected smart devices, allowing users to view the current status and settings (e.g., whether lights are on, thermostat temperature, door lock status).	[ ]	[ ]	[ ]
The system shall allow users to create custom automation rules that define how devices should behave based on specific conditions or events (e.g., turning on lights when motion is detected or lowering the thermostat at a set time). Users should be able to combine multiple conditions and actions in their automation scripts.	[ ]	[ ]	[ ]
The application should be able to connect with third-party services, such as weather forecasting services to adjust heating/cooling based on outdoor temperatures or geolocation services for setting home/away modes.	[ ]	[ ]	[ ]
The application should collect and store data from all connected devices for historical analysis. Users should be able to visualize this data through charts or reports to understand patterns in device usage.	[ ]	[ ]	[ ]

Comentários sobre o Grupo 3: _____

Grupo 4 Caracteristica: Interaction

Requisito	**FPG**	**NFG**	**NSO**
The system shall support integration with popular voice assistants like Amazon Alexa, Google Assistant, and Apple Siri, allowing users to control devices through voice commands.	[ ]	[ ]	[ ]

The system shall send notifications to users about important events, such as security alerts, changes in the status of devices, or failures in automation routines. Notifications should be customizable to allow users to choose the events they want to be notified about.	[ ]	[ ]	[ ]
The system shall provide features for monitoring energy usage of connected devices and suggest optimization strategies for reducing power consumption. Users should be able to view historical energy consumption data.	[ ]	[ ]	[ ]
The system should implement fail-safe mechanisms to handle situations where connectivity is lost, such as retaining the last known device state or automatically switching to a default state.	[ ]	[ ]	[ ]
The application should include a feedback mechanism for users to report bugs, suggest improvements, and access help resources. A support channel should be available for troubleshooting common issues.	[ ]	[ ]	[ ]
The system shall support various communication protocols (e.g., Wi-Fi, Zigbee, Z-Wave, Bluetooth) to ensure compatibility with a wide range of IoT devices.	[ ]	[ ]	[ ]

Comentários sobre o Grupo 4: _____

Grupo 5 Caracteristica: Action

Requisito	**FPG**	**NFG**	**NSO**
The application shall allow users to set schedules for connected devices (e.g., setting the lights to turn on at sunset or the thermostat to adjust temperature based on a daily routine).	[ ]	[ ]	[ ]

Comentários sobre o Grupo 5: _____

D.1 Formulário Likert

Declaração 1: Os resultados do método são úteis para o entendimento dos subsistemas da aplicação.

Explicação: Refere-se ao grau em que um participante acredita que usar o método produziria resultados desejáveis para o desenvolvimento do sistema.

[ ] Concordo Totalmente

[ ] Concordo

[ ] Não estou decidido

[ ] Discordo

[ ] Discordo Totalmente

Declaração 2: Usar os resultados do método (agrupamentos) poderia reduzir meu esforço em lidar com os requisitos de aplicações de IoT.

Explicação: Refere-se à percepção de quanto o resultado do método pode diminuir a quantidade de trabalho ou esforço necessário para atingir um objetivo.

[ ] Concordo Totalmente

[ ] Concordo

[ ] Não estou decidido

[ ] Discordo

[ ] Discordo Totalmente

Declaração 3: Eu poderia usar esse método se tivesse oportunidade.

Explicação: Refere-se à probabilidade de um participante usar o método no futuro para gerar agrupamentos de requisitos.

[ ] Concordo Totalmente

[ ] Concordo

[ ] Não estou decidido

[ ] Discordo

[ ] Discordo Totalmente