ReqMLSCity: Uma ferramenta de análise de requisitos utilizando aprendizado de máquina para aplicações de cidades inteligentes

Bruno Carvalho da Silva[1], **Rodrigo Nascimento**[1] **e Davi Viana**[1]

[1]Universidade Federal do Maranhão (UFMA)
São Luís – MA – Brasil

{bruno.carvalho1, rodrigo.nascimento}@discente.ufma.br,davi.viana@ufma.br

Abstract. *Ambiguity and incompleteness in the requirements elicitation phase and in the documents generated can potentially harm the rest of the development process. Hence, analyzing these documents is critical for proper software development progress. However, manual analysis of these documents is expensive and time-consuming, and prone to failure, particularly in complex systems. Automatic detection of such errors shows promise. In recent years, requirements engineering has faced the challenge of dealing with evolving contexts. Therefore, this paper aims to introduce the ReqMLSCity tool, which is an updated version of the ReqSCity tool. This tool assists in the requirements analysis process by automatically analyzing and detecting ambiguous language and misunderstandings that arise from faulty syntactic structures utilizing Natural Language Processing (NLP) techniques. The tool can evaluate if a requirement is relevant to smart cities and provide suggested improvements for meaning completion using the M3-Ontology domain ontology as its foundation. A pilot study was carried out to initially evaluate the proposal by analyzing 55 requirements and comparing the evaluation results of two software engineering experts.*

Resumo. *Erros como ambiguidade e incompletude na fase de eliciação de requisitos e nos documentos gerados podem prejudicar o restante do processo de desenvolvimento. Desta forma, uma análise desses documentos é essencial para o avanço adequado de desenvolvimento do software. A análise manual destes documentos de requisitos é de alto custo, tanto financeiro quanto de tempo, e tendenciosa a falhas, sobretudo quando o sistema possui uma grande complexidade. A detecção automática desses erros se apresenta como uma abordagem promissora. Adicionalmente, um problema recente no campo da engenharia de requisitos é sobre como tratar contextos recentes. O objetivo deste trabalho é apresentar a ferramenta ReqMLSCity, uma evolução da ferramenta ReqSCity, que auxilia o processo de análise de requisitos de maneira automática, focando na análise de ambiguidade e geração de incompreensão de sentido a partir de estruturas sintáticas defeituosas através do uso de PLN. A ferramenta é capaz também de avaliar se o requisito está no contexto de cidades inteligentes e fornece sugestões para melhorar a completude de sentido usando como base a ontologia de domínio M3-Ontology. A avaliação inicial da proposta foi feita realizando um estudo piloto com 55 requisitos e a comparação de análise é feita com base nos resultados da avaliação de dois especialistas em engenharia de software.*

1. Introdução

A Engenharia de Requisitos, conforme [Sommerville 2018], é o processo de descobrir, analisar, documentar e verificar as necessidades dos usuários de um sistema, bem como suas restrições. A engenharia de requisitos é uma das etapas mais importante no ciclo de vida de desenvolvimento de software. Consoante [Arya et al. 2012], após a compilação de diversos relatos de erros no desenvolvimento de software, relatam que 56% dos erros acontecem na etapa de requisitos. A propagação de um erro de requisito nas demais etapas do desenvolvimento, caso chegue a etapa de produção tal erro, o custo de reparo pode chegar a 10000 vezes do que seria caso fosse consertado na etapa de requisitos [Arya et al. 2012].

Em projetos de software, os requisitos ficam contidos em um documento chamado Especificação de Requisitos de Software (*Software Requirements Specification* ou SRS). A SRS geralmente é escrita em linguagem natural, o que facilita sua produção, contudo, agrega os problemas inerentes de tal linguagem, tais quais, ambiguidades, sinônimos, coloquialismos, linguagem específica do domínio, ironia e sarcasmo, por exemplo. Outros problemas inerentes do uso de linguagem natural são erros sintáticos que possam ser cometidos por quem produz o SRS ou requisitos com estruturas deletérias, no contexto de requisitos, como utilização de voz passiva.

Tendo em vista tais problemas a *Software Engineering Standards Committee of the IEEE Computer Society* lança a IEEE ISO 29148-2018 [855 2018] que orienta sobre boas práticas para o desenvolvimento dos documentos de SRS. Dentre as recomendações dadas em IEEE 29148-2018 são definidas nove características de um bom SRS que deve ser: *Necessary, Unambiguous, Complete, Appropriate, Feasible, Verifiable, Singular, Conforming, Correct*. A solução apresentada neste artigo visa contribuir para que seja garantida a característica de não ambiguidade (*unambiguous*).

A utilização de Processamento de linguagem natural (PLN) na área de engenharia de requisitos não é algo novo, contudo, vem em crescimento na última década [Sonbol et al. 2022]. A aplicação de PLN no contexto de Engenharia de Requisitos para aplicações de domínios específicos é um problema em aberto, pois cada domínio possui seu conjunto de jargões, regras de negócio e práticas [Dalpiaz et al. 2018]. Com base nessa problemática, a presente solução oferece suporte para classificação de requisitos de um mesmo domínio, de forma que os analistas de requisitos de tais aplicações possam obter facilmente um grupo de requisitos para análise. No caso do presente trabalho, o domínio considerado é de cidades inteligentes (do inglês, *smart cities*).

Soluções para cidades inteligentes visam atenuar problemáticas decorrentes da superlotação dos centros urbanos, por exemplo, crescimento descontrolado, poluição do ar e água, entre outros. Tais soluções necessitam ser corretamente identificadas e especificadas dada a sua criticidade e alto envolvimento com vidas humanas. Nesse sentido, urge uma solução para análise de requisitos de tal contexto. Ressalta-se, contudo, que não há um conjunto consolidado de práticas e técnicas de engenharia de requisitos para aplicações de contextos contemporâneos, como cidades inteligentes [da Silva et al. 2021].

Dado o exposto, o objetivo desta iniciação tecnológica é apoiar a análise de requisitos. Para isso, desenvolveu-se a ferramenta ReqMLSCity, desenvolvida como uma versão melhorada da ReqSCity, uma ferramenta que obteve um registro de programa de

computador no INPI nº BR512023000336-0 e um artigo aceito na Escola Regional de Computação Maranhão, Ceará e Piauí[1]. Ambas as ferramentas foram apoiadas por meio de editais do Programa Institucional de Bolsas de Iniciação em Desenvolvimento Tecnológico e Inovação (PIBITI) financiadas pelo CNPQ e FAPEMA. Além disso, este trabalho faz parte do INCT de internet do futuro para cidades inteligentes, projeto este que engloba nove instituições brasileiras, além de parceiros internacionais.

As melhoras apresentadas na evolução da solução versam sobre trazer consistência aos gatilhos, agregando técnicas de Aprendizado de Máquina no módulo de contextualização; uso de árvores sintáticas para detecção de ambiguidades, bem como acréscimo dos tipos *analitical* e *attachment*.

A ReqMLSCity, portanto, é ferramenta desenvolvida para analisar requisitos em três aspectos: ambiguidade, sintaticamente e contextualização em cidades inteligentes. Uma vez identificados os requisitos contextualizados, a ferramenta também fornece suporte a avaliação da especificação de sensores e atuadores, elementos centrais do contexto analisado. A linguagem natural que a ReqMLSCity analisa é a lingua inglesa, dada a sua ampla utilização na área da computação.

2. Ferramentas Relacionadas

A literatura apresenta um conjunto de ferramentas que visam apoiar a análise de requisitos. Contudo, ao comparar essas ferramentas com a ferramenta proposta neste trabalho, verificou-se que não há, até o presente momento, uma ferramenta que apoie a análise de requisitos para aplicações no contexto de cidades inteligentes de maneira específica. Desta forma, apresentam-se algumas soluções que serviram de inspiração para a ReqMLSCity.

Em [Gnesi et al. 2005] é apresentada a ferramenta QuARS, uma ferramenta para detecção de indicadores de qualidade. Os indicadores são divididos em três grupos: expressividade, consistência e completude. Para a definição do tipo de análise a ser feita, o usuário passa à ferramenta os termos correspondentes aos indicadores. Um outro tipo de operação feita pela QuARS é o agrupamento de requisitos com base em uma lista de termos. A ferramenta possui uma interface gráfica para desktop, com três telas principais: Dicionário; Entrada, onde é possível editar os requisitos; e Saída, onde o documento de requisitos é mostrado. O artigo traz um exemplo para o agrupamento de requisitos não funcionais relativos a segurança, contudo o artigo não apresenta um resultado claro sobre a eficiência desses agrupamentos.

[Huertas and Juárez-Ramírez 2012] apresentam a ferramenta NLARE (”Natural Language Automatic Requirements Evaluation”), que foca nos requisitos funcionais com o objetivo de detectar três problemas comuns em requisitos: Ambiguidade, Incompletude e Atomicidade. No módulo de incompletude esta ferramenta busca identificar o ator, quem executa a ação, a função, a ação em si, e o detalhe, complemento da ação. A NLARE utiliza Python como linguagem de implementação e utiliza a biblioteca NLTK para PLN. Como técnicas de PLN utilizadas tem-se: Tokenização, *POS tagger* e *Parsing*. Não se tem nesta ferramenta um aprofundamento na semântica dos requisitos, nem agrupamento dos mesmos. O artigo não apresenta nenhuma interface gráfica para a ferra-

[1]https://doi.org/10.5753/ercemapi.2022.226296

menta. Para avaliação a metodologia utilizada foi a de obter avaliações de desenvolvedores para um grupo de requisitos e foi posteriormente comparado o resultado da ferramenta com o resultado manual.

Por fim, em [Arya et al. 2012] é apresentada uma ferramenta que exclusivamente detecta ambiguidade. Os tipos de ambiguidades são as seguintes: Ambiguidade Léxica, que ocorre quando uma palavra possui múltiplos significados; Ambiguidade Sintática, quando a estrutura sintática de um requisito o leva a ter mais de um sentido; e Ambiguidade de Sintaxe, quando uma sentença não termina com ponto ou quando não se tem um sujeito. Um elemento essencial para esta ferramenta é o corpus de palavras ambíguas, várias palavras que por si só geram ambiguidade. A ferramenta possui uma interface para desktop demonstrada com três telas, uma para entrada, uma para o resultado do processamento e outra para o tipo de erro.

3. Arquitetura e Funcionalidades

Para o desenvolvimento da ReqMLSCity a linguagem utilizada foi o Python. Para as rotinas de PLN foi utilizada a biblioteca NLTK (*Natural Language Toolkit*) pois fornece vários corpus e recursos léxicos, como o Wordnet[2], e possui também ferramentas que dão suporte a tokenização, *Pos-tagger*,*chunking* e segmentação de sentenças. A biblioteca NLTK fornece também suporte a remoção de *stopwords* utilizada nos gatilhos. Para que a aplicação fosse web, o framework utilizado foi o Flask[3].

O módulo de contextualização dos requisitos foi construído a partir das bibliotecas Pandas[4], Numpy[5] e Sklearn[6]. Os requisitos antes de passados para os classificadores sofrem um processo de vetorização feito a partir da biblioteca Sklearn com a técnica *Bag-of-words* com *Term Frequency - Inverse Document Frequency*(TF-IDF). O TF-IDF combina dois indices, quanto o TF traz a importância do termo em relação as demais palavras do mesmo requisito, o IDF dá importância para palavras que são exclusivas de um determinado conjunto de requisitos. A utilização de TF-IDF visa, portanto, focar nas palavras que são importantes para os requisitos e ajuda a distinguir um requisito do restante.

Já para a análise dos requisitos, foi utilizada a ontologia M3-Ontology[7], que apresenta uma ontologia com termos heterogêneos de Internet das Coisas (do inglês, *Internet of Things*) e cidades inteligentes.

3.1. Arquitetura da ferramenta

A Figura 1 apresenta os principais componentes da ferramenta ReqMLSCity. No início do processo, os **requisitos** são recebidos em formato de documento de texto .txt ou .docx. Esses requisitos são divididos e colocados como elementos em uma lista, antes de serem encaminhados para a **Classe Texto**.

Na **Classe Texto**, os requisitos são tokenizados e etiquetados, de forma que se possa ter nas demais classes os requisitos com as suas devidas classes morfológicas. Além

[2]https://wordnet.princeton.edu/

[3]https://flask.palletsprojects.com/en/2.1.x/

[4]https://pandas.pydata.org/

[5]https://numpy.org/

[6]https://scikit-learn.org/

[7]https://databus.dbpedia.org/ontologies/sensormeasurement.appspot.com/m3

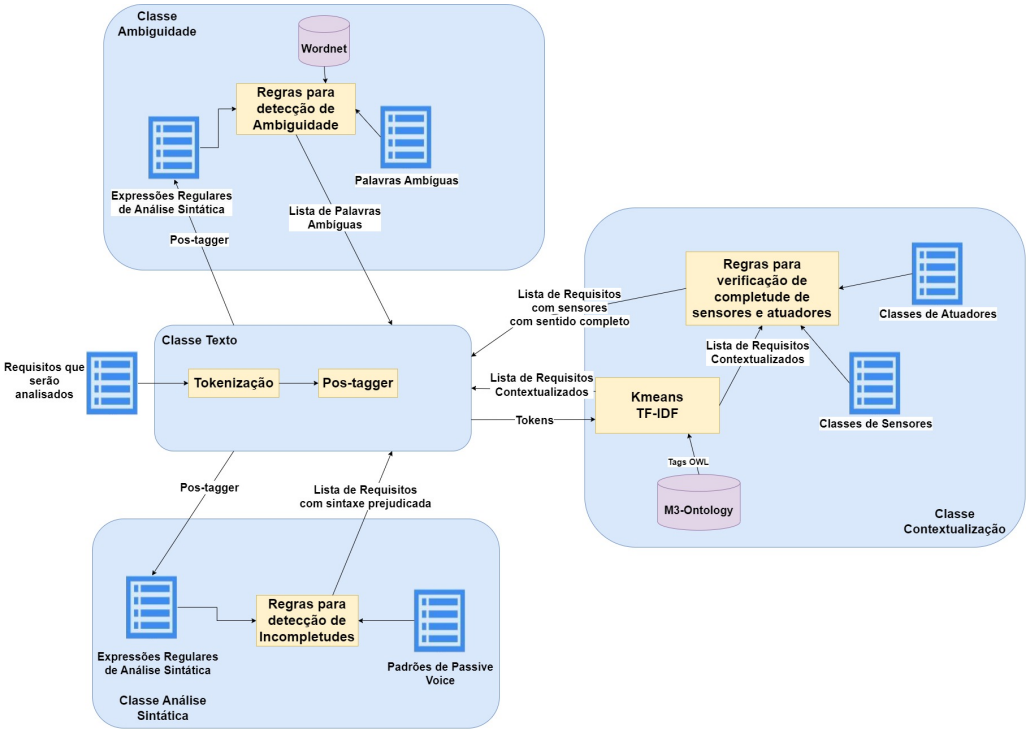

Figura 1. Arquitetura da ReqMLSCity. Fonte: Autores

do mais, essa classe é a responsável por obter o produto das demais classes juntando-os desta forma.

3.2. Classe Ambiguidade

Essa classe é dividida em duas partes: (1) trata as ambiguidades léxicas, aquelas que são ocasionadas a nível de palavras; e (2) as ambiguidades sintáticas que são provocadas por causa da estrutura dos requisitos.

Nas Ambiguidades léxicas, há os seguintes gatilhos (do inglês, *triggers*):

- `Palavras Ambíguas`: para o funcionamento deste gatilho tem-se o recebimento de uma lista de palavras ambíguas, palavras essas que podem levar os requisitos à incompreensão ou podem ser vagas, como *big*, *sometime*, *low* [Sandhu and Sikka 2015], entre outros;
- `Algoritmo Flexible Ambiguity`: O próximo gatilho tem como base [Bäumer and Geierhos 2018]. Uma vez recebidos os tokens e as palavras etiquetadas, o processo se inicia com a retirada de *stopwords*. Após essa atividade, sobram as palavras relevantes para o sentido do requisito e, então, se utiliza o Wordnet. Esse uso ocorre da seguinte maneira, é realizada uma consulta ao synset (conjunto de sinônimos cognitivos) de cada palavra relevante, caso a palavra tenha mais de três elementos em seu synset, ela é potencialmente ambígua, uma vez que ela possui uma alta quantidade de sentidos. Quando a quantidade de tokens potencialmente ambíguos exceder 75% em relação aos tokens totais de um requisito ele é considerado ambíguo.

Para as ambiguidades sintáticas, tem-se três gatilhos:

- `Coordination Ambiguity`: Uma *coordination ambiguity* ocorre quando um modificador, adjetivo por exemplo, se refere a uma ou mais *Noun Phrases* que são sucedidas por uma conjunção e portanto não fica claro a qual substantivo o modificador se refere.
- `Analitical Ambiguity`: Uma *analitical ambiguity* ocorre quando um adjetivo se refere a uma ou mais *Noun Phrases* e portanto não fica claro a qual substantivo o modificador se refere.
- `Attachment Ambiguity`: Uma *attachment ambiguity* ocorre quando um verbo é seguido por uma *Noun Phrase*, uma conjunção e outra *Noun Phrase*. A ambiguidade fica em não ser claro se a *Preposition Phrase* se liga ao verbo ou a primeira *Noun Phrase*.

3.3. Classe Análise Sintática

A classe **Análise Sintática** conta com quatro gatilhos que foram organizados de maneira lógica em etapas conforme a Figura 2. Utiliza majoritariamente da formação de *parsing tree* com base em expressões regulares. Para a montagem dos padrões de expressões regulares o livro de gramática inglesa de [Burton-Roberts 2013] foi consultado.

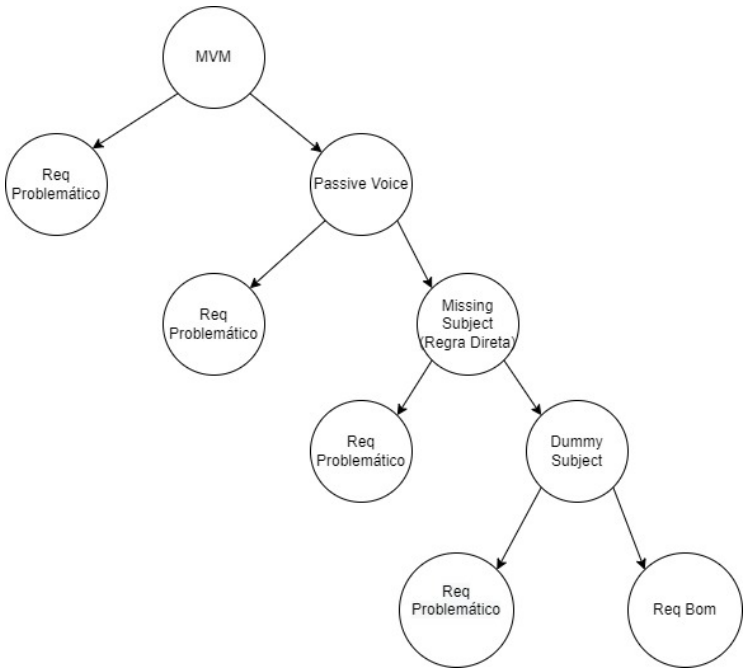

Figura 2. Sequência de processamento da Análise Sintática. Fonte: Autores

- `MVM-Missing Verb Mistake`: inicia o processamento da análise sintática. É a responsável por verificar a existência de verbo no requisito. Dado que os requisitos devem indicar algo que o sistema deva fazer ou ter, um requisito sem uma ação declarada torna-se vago.
- `Passive Voice`: requisitos em voz passiva podem implicar em um não entendimento sobre qual papel o sistema deve desempenhar e o que ele deve executar.
- `Missing Subject`: é responsável por verificar se o requisito possui um sujeito, algo que realiza a ação ou possui uma característica.

- `Dummy Subject`: foca em duas palavras serem utilizadas como sujeito da frase, no caso *"it"* e *"there"*. Ao utilizar essas palavras no sujeito, o sentido se torna vago, passando a ideia de que há algo para se fazer, mas sem saber quem será o responsável pela execução.

3.4. Classe Contextualização

É a classe responsável por avaliar quais requisitos de software estão no contexto de cidades inteligentes e, a posteriori, os que estão devidamente especificados. Tem-se dois gatilhos para essa classe:

- `Contextualiza`: é responsável por avaliar quais requisitos de fato estão no contexto de cidades inteligentes. Para isso, esse gatilho tem fluxo representado na Figura 3. Os requisitos chegam no gatilho etiquetados e passam por uma primeira redução de dimensionalidade com a remoção de *stopwords*, caracteres especiais e números. Posteriormente, os requisitos passam por um processo de lematização antes de serem vetorizados em um BOW com TF-IDF. Agora tem-se dois algoritmos de *clustering*. O primeiro utiliza o K-means inicializado com dois *clusters*, havendo dois grupos, para determinar qual grupo é o contextualizado é utilizada a M3-Ontology para formar um *score* que irá determinar isso. O segundo é um algoritmo que utiliza o vetor TF-IDF juntamente com a M3-Ontology para compor um *score* para cada requisito, os requisitos que tiverem seu *score* acima da média são os contextualizados. Por fim, os requisitos categorizados vão ser concatenados por meio de uma junção dos resultados, ou seja, basta que um algoritmo de *clustering* deduza que aquele requisito está contextualizado que ele será declarado como tal;

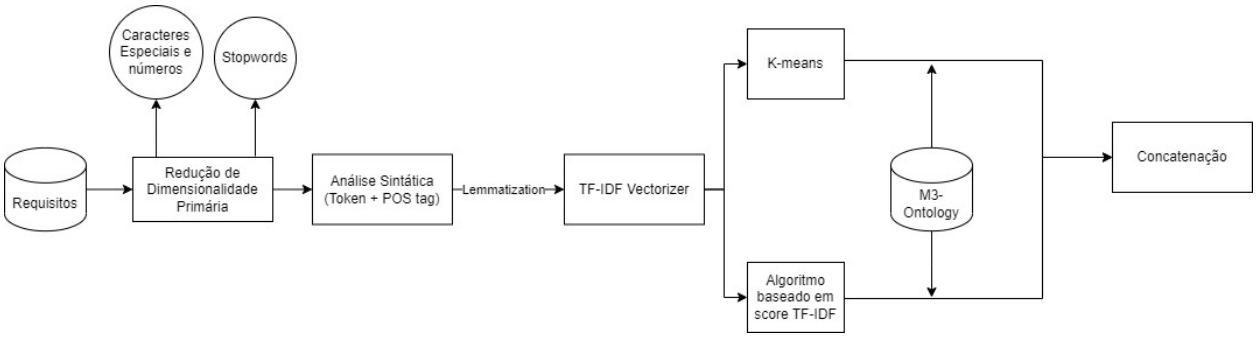

Figura 3. Fluxo dos requisitos para a contextualização. Fonte: Autores

- `Completude`: dado o contexto de cidades inteligentes, este trabalho também avalia se os requisitos contextualizados descrevem sensores e atuadores devidamente especificados. Por especificados entende-se que foi descrito o tipo de sensor/atuador e o que esse sensor/atuador faz. Para construir esse gatilho a M3-Ontology foi utilizada para criar dois arquivos contendo as classes de sensores e atuadores. O algoritmo irá avaliar somente os requisitos que estão contextualizados, logo, a saída do gatilho de contextualização é utilizada aqui.

4. Interface da ReqMLSCity e Estudo Piloto

A Ferramenta de Análise de Requisitos para aplicações de Cidades Inteligentes foi concebida de modo a ser uma ferramenta web de fácil uso. A ferramenta será disponibilizada

de forma gratuita. A seguir será apresentada sua interface e um estudo piloto para exemplificar o uso da ReqMLSCity e fazer uma avaliação inicial do desempenho da mesma.

4.1. Interface

A ReqMLSCity é composta basicamente por duas telas. A primeira é a tela inicial, mostrada na Figura 4, é a responsável pela funcionalidade de submissão (*upload*) do documento de requisitos, podendo ser nos formatos .txt ou .docx. Posteriormente, o usuário pode escolher qual tipo de análise será realizada dentre as opções: Análise Sintática (Incompletude), Ambiguidade, Contextualização ou Análise Completa. Ao clicar em enviar, o retorno do processamento para os três primeiros tipos será exibido na mesma página em um formato de tabela logo abaixo do campo de entrada. Já para Análise Completa, o processamento irá gerar um documento, contendo todas as análises e os requisitos processados no formato .docx que será baixado quando findado o processamento.

Figura 4. Tela inicial da ReqMLSCity. Fonte: Autores

A segunda tela, demonstrada na Figura 5, apresenta a parte da aplicação que serve como instruções de uso aos usuários. Aqui existem botões, que uma vez clicados, aparecerão diagramas com instruções a: entrada, ambiguidade, incompletude (análise sintática) e contextualização. A ferramenta está disponível em um repositório do GitHub [8].

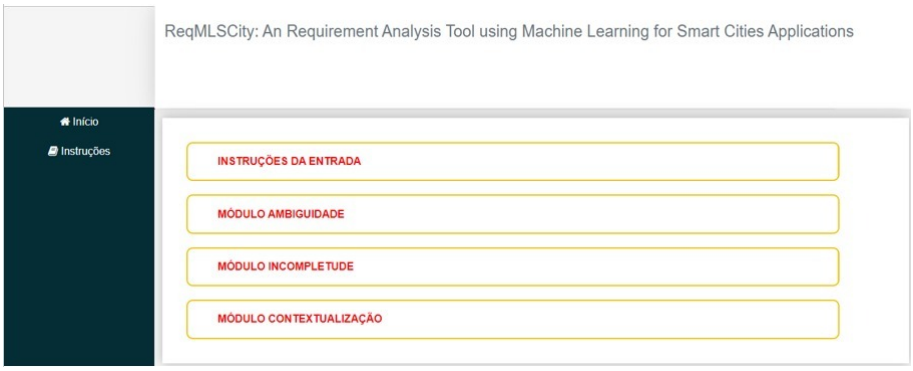

Figura 5. Tela instruções aos usuários. Fonte: Autores

4.2. Estudo Piloto

Para obter uma análise inicial da ferramenta proposta em relação ao seu propósito, bem como exemplificar o seu uso, foi realizado um estudo piloto. Neste estudo foram utilizados requisitos obtidos de projetos da disciplina de Tópicos Avançados em Cidades Inteligentes dos programas de pós-graduação em engenharia elétrica (PPGEE) e em ciência

[8] https://github.com/carvalhosilva42/ReqSCity2

da computação (PPGCC) da UFMA. Foram utilizados 55 requisitos de oito projetos da disciplina.

Os requisitos foram avaliados manualmente por dois especialistas da área de engenharia de software. As instruções passadas foram para classificá-los como ambíguos, incompletos ou no contexto de cidades inteligentes. O resultado desta avaliação está disponível no seguinte repositório [9] .

Em relação a ReqMLSCity, os requisitos foram submetidos para a ferramenta em formato .txt e foi escolhida a opção de análise completa e o resultado foi registrado. A Tabela 1 contém os campos relativos as avaliações de requisitos com estrutura sintática prejudicial, ambiguidade e o conjunto de requisitos que é do contexto de cidades inteligentes. Além disso, a tabela a apresenta uma comparação entre a ferramenta e os especialistas contendo quatro métricas utilizadas amplamente no contexto da classificação binária, tais quais: Acurácia, Precisão, *Recall* e F-1.

[redacted]	Especialista A			Especialista B		
[redacted]	Análise Sintática	Ambiguidade	Contexto Cidades Inteligentes	Análise Sintática	Ambiguidade	Contexto Cidades Inteligentes
Acurácia	55%	64%	69%	55%	62%	60%
Precisão	38%	18%	52%	46%	36%	52%
Recall	53%	67%	67%	52%	53%	53%
F1	44%	29%	59%	49%	43%	52%

Tabela 1. Resultados da avaliação do estudo piloto. Fonte: Autores

De acordo com tais resultados expostos na Tabela 1, percebe-se que o desempenho da ferramenta é satisfatório em algumas análises, chegando a 69% de acurácia para detecção de requisitos de contexto para os resultados do Especialista A. Contudo, alguns resultados encontram-se bem abaixo do esperado, como a precisão da ferramenta para detecção de ambiguidade na comparação para ambos especialistas.

Vale ressaltar que não foi gerada uma lista única com a avaliação dos dois especialistas, pois cada um deles possui experiências diferentes nas atividades de análise de requisitos e desenvolvimentos de aplicações para o contexto de IoT e Cidades Inteligentes. Os dados obtidos neste estudo piloto servem para analisar o seu funcionamento e serão fundamentais para a evolução da ferramenta em versões futuras.

5. Conclusão

Este artigo apresentou o desenvolvimento da iniciação tecnológica que resultou na a ReqMLSCity, uma ferramenta que visa automatizar o processo de análise de requisitos, por meio de técnicas de PLN. Essa análise considera três aspectos: ambiguidade, análise sintática e contextualização dos requisitos para Cidades Inteligentes. A ferramenta foi analisada em um estudo piloto e obteve alguns resultados promissores, além de resultados que permitem fazer ajustes nos gatilhos utilizados.

Como trabalhos futuros, pretende-se evoluir a ferramenta com base nos resultados já obtidos e executar novos estudos visando adequação para uso comercial. Além disso, pretende-se fornecer uma forma que os usuários possam atualizar os requisitos considerados ruins na própria ferramenta. Outro ponto considerado, seria contar ainda com uma

[9]https://zenodo.org/record/8338616

normalização interna, de modo a deixar o usuário com menos atribuições. Por fim, existe a necessidade de realizar novas avaliações, de modo a atestar sua acurácia e garantir resultados satisfatórios.

Agradecimentos

Agradecemos a Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) - Código de Financiamento 001 e PROCAD-Amazônia. Este trabalho também contou com o apoio do INCT de Internet do Futuro para Cidades Inteligentes (CNPq 465446/2014-0, CAPES 88887.136422/2017-00, FAPESP 14/50937-1 e 15/24485-9), CNPq (311608/2017-5, 420907/2016-5 e 312324/2015-4). O último autor agradece a FAPEMA (UNIVERSAL-00745/19 e Produtividade BEPP-01608/21). Por fim, o trabalho teve o apoio de bolsas de PIBITI (Iniciação Tecnológica) da FAPEMA, UFMA e CNPq.

Referências

(2018). Iso/iec/ieee international standard - systems and software engineering – life cycle processes – requirements engineering. *ISO/IEC/IEEE 29148:2018(E)*, pages 1–104.

Arya, N., Nigam, B., and Nigam, A. (2012). Tool for automatic discovery of ambiguity in requirements.

Burton-Roberts, N. (2013). *Analysing sentences: An introduction to english syntax, third edition*.

Bäumer, F. and Geierhos, M. (2018). Flexible ambiguity resolution and incompleteness detection in requirements descriptions via an indicator-based configuration of text analysis pipelines.

da Silva, D. V., de Souza, B. P., Gonçalves, T. G., and Travassos, G. H. (2021). A requirements engineering technology for the iot software systems. *Journal of Software Engineering Research and Development*.

Dalpiaz, F., Ferrari, A., Franch, X., and Palomares, C. (2018). Natural language processing for requirements engineering: The best is yet to come. *IEEE Software*, 35(5):115–119.

Gnesi, S., Lami, G., and Trentanni, G. (2005). An automatic tool for the analysis of natural language requirements. *Comput. Syst. Sci. Eng.*, 20.

Huertas, C. and Juárez-Ramírez, R. (2012). Nlare, a natural language processing tool for automatic requirements evaluation. In *Proceedings of the CUBE International Information Technology Conference*, CUBE '12, page 371–378, New York, NY, USA. Association for Computing Machinery.

Sandhu, G. and Sikka, S. (2015). State-of-art practices to detect inconsistencies and ambiguities from software requirements. In *International Conference on Computing, Communication Automation*, pages 812–817.

Sommerville, I. (2018). *Engenharia de Software*. Pearson Education.

Sonbol, R., Rebdawi, G., and Ghneim, N. (2022). The use of nlp-based text representation techniques to support requirement engineering tasks: A systematic mapping review. *IEEE Access*, 10:62811–62830.