Universidade Federal de Pernambuco
Centro de Informática

Raul Pereira Coelho

Detectando Falhas de Localização em Aplicativos Android: Uma Abordagem Automatizada

Recife

2025

Raul Pereira Coelho

Detectando Falhas de Localização em Aplicativos Android: Uma Abordagem Automatizada

Trabalho apresentado ao Programa de Graduação em Engenharia da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Graduado em Engenharia da Computação.

Área de Concentração: Engenharia de Software

Orientador (a): Breno Alexandro Ferreira de Miranda

Recife

2025

Ficha de identificação da obra elaborada pelo autor,
através do programa de geração automática do SIB/UFPE

Coelho, Raul Pereira.
Detectando falhas de localização em aplicativos android: uma abordagem automatizada / Raul Pereira Coelho. - Recife, 2025.
38 p. : il., tab.

Orientador(a): Breno Alexandro Ferreira de Miranda
Trabalho de Conclusão de Curso (Graduação) - Universidade Federal de Pernambuco, Centro de Informática, Engenharia da Computação - Bacharelado, 2025.
Inclui referências.

1. Localização. 2. Internacionalização. 3. L10n. 4. I18n. 5. Droidbot. 6. OCR. I. Miranda, Breno Alexandro Ferreira de. (Orientação). II. Título.

000 CDD (22.ed.)

RAUL PEREIRA COELHO

DETECTANDO FALHAS DE LOCALIZAÇÃO EM APLICATIVOS ANDROID: uma abordagem automatizada

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia da Computação da Universidade Federal de Pernambuco, como requisito parcial para obtenção do título de bacharel em Engenharia da Computação.

Aprovado em: 03/04/2025

BANCA EXAMINADORA

_____

Prof. Dr. Breno Alexandro Ferreira de Miranda (Orientador)

Universidade Federal de Pernambuco

_____

Profa. Dra. Paola Rodrigues de Godoy Accioly (Examinadora Interna)

Universidade Federal de Pernambuco

AGRADECIMENTOS

Agradeço à minha família e amigos pelo apoio, paciência e incentivo ao longo dessa jornada.

À minha namorada e futura esposa, Bárbara Alves: sem você, este trabalho não seria possível.

RESUMO

A internacionalização (I18N) e a localização (L10N) de software são processos essenciais para adaptar aplicativos a diferentes mercados e culturas. Com o aumento da diversidade linguística e cultural dos usuários, é fundamental garantir que as interfaces e mensagens exibidas nos aplicativos atendam aos padrões de cada idioma e região. Este trabalho propõe a integração de um programa de detecção de falhas de localização ao *DroidBot*, uma ferramenta amplamente utilizada para testes de aplicações *Android*, que navega automaticamente pelas telas do aplicativo. O objetivo principal é aproveitar as capacidades do DroidBot para coletar automaticamente capturas de tela e elementos da interface do usuário e, em seguida, aplicar uma análise especializada de identificação de problemas de l10n.

Palavras-chave: Localização. Internacionalização. L10n. I18n. Droidbot. OCR. Python. Imagem.

ABSTRACT

Internationalization (I18N) and localization (L10N) of software are essential processes for adapting applications to different markets and cultures. With the increasing linguistic and cultural diversity of users, it is crucial to ensure that the interfaces and messages displayed in applications meet the standards of each language and region. This work proposes the integration of a localization error detection program with *DroidBot*, a widely used tool for testing *Android* applications that automatically navigates through the app's screens. The main goal is to leverage DroidBot's capabilities to automatically capture screenshots and UI elements, and then apply specialized analysis to identify L10N issues.

Keywords: Localization. Internationalization. L10N. I18N. Droidbot. OCR. Python. Image.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

HTML	Hypertext Markup Language
I18N	Internacionalização
IA	Inteligência Artificial
JSON	*JavaScript Object Notation*
L10N	Localização
OCR	Reconhecimento Óptico de Caracteres
PIL	*Python Imaging Library*
XML	*Extensible Markup Language*

SUMÁRIO

1 INTRODUÇÃO

No atual mundo digital, alcançar mercados internacionais deixou de ser uma opção e se tornou uma necessidade para empresas de software que desejam crescer globalmente. Entretanto, expandir um produto para diferentes países vai além da simples tradução de textos. É essencial garantir que a aplicação respeite as nuances culturais, os requisitos legais e as preferências do público local. Para isso, dois processos desempenham um papel fundamental: a Internacionalização (I18N) e a Localização (L10N).

A I18N refere-se ao desenvolvimento de um software preparado para suportar múltiplos idiomas e culturas sem a necessidade de grandes modificações no código. Esse processo antecipa desafios linguísticos e culturais, tornando a adaptação para novos mercados mais simples.(GROSS, 2006) Já a localização é a personalização do software para um público específico, ajustando não apenas o idioma, mas também formatos de data, moeda, regulamentos e até elementos visuais que possam ter significados diferentes em cada cultura. (GROSS, 2006)

Ainda que haja uma crescente adoção dessas práticas, a localização ainda enfrenta desafios significativos. Erros como traduções ausentes, inconsistentes ou mal adaptadas podem comprometer a experiência do usuário e até afetar a aceitação do produto em um novo mercado. Para minimizar esses problemas, é essencial adotar ferramentas que auxiliem no processo de adaptação e garantam um alto nível de precisão nas traduções e ajustes culturais.(COUTO; MIRANDA, 2023)

Assim, este trabalho busca desenvolver uma aplicação voltada para identificar automaticamente falhas de localização em *software* multilíngue, com foco especial na detecção de problemas relacionados à ausência de tradução e também de problemas de elipse.

O restante deste trabalho está dividido da seguinte forma:

- **Conceitos Fundamentais:** em que são definidos termos essenciais.
- **Trabalhos Relacionados:** em que são mencionados trabalhos que auxiliaram este estudo.
- **Metodologia:** em que a metodologia do trabalho é dissecada.
- **Ferramenta:** em que a ferramenta é descrita.
- **Avaliação:** em que são apresentados resultados e avaliações.

- **Conclusão e Direções Futuras:** em que conclusões são tiradas e recomendações para trabalhos futuros são feitas.

2 CONCEITOS FUNDAMENTAIS

2.1 PROBLEMAS DE LOCALIZAÇÃO

A L10N envolve uma série de adaptações para garantir que um software seja adequado a diferentes locais, considerando elementos como formatação de datas e horários, unidades de medida, moedas, entre outros. No entanto, muitos aplicativos e sites ainda realizam esse processo de forma manual, o que aumenta a probabilidade de erros.(ZALIESKAITė, 2023)

Há diversos tipos de erros de localização que podem comprometer a experiência do usuário e alguns são mencionados a seguir.

2.1.1 Ausência de Tradução ou Tradução Incorreta

Acontece quando o texto não é traduzido para o idioma alvo, ou é traduzido, porém incorretamente.

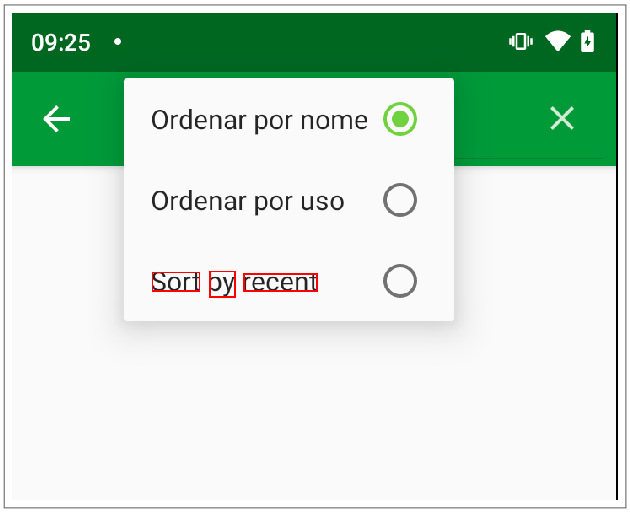

Figura 1 – Exemplo de ausência de tradução

2.1.2 Elipse

Acontece quando, após a tradução, o texto se torna muito longo para o espaço alocado, resultando em um corte com reticências("...").

Figura 2 – Exemplo de elipse

2.1.3 Sobreposição

Acontece quando o texto é sobreposto por elementos gráficos, interferindo na legibilidade.

Figura 3 – Exemplo de sobreposição

2.1.4 Truncamento

É bem similar à elipse, porém, neste caso, o texto é cortado repentinamente, sem reticências.

Figura 4 – Exemplo de truncamento

2.2 RECONHECIMENTO ÓPTICO DE CARACTERES

Reconhecimento Óptico de Caracteres (OCR)(EIKVIL, 1993) é uma tecnologia que transforma os caracteres presentes em uma imagem em texto. Ele permite a extração automática de caracteres impressos ou manuscritos, suporta múltiplos idiomas e é amplamente usado para digitalizar contratos, notas fiscais, entre outros. Atualmente, para atingir esse objetivo, técnicas de *Deep Learning* estão sendo bastante utilizadas.

3 TRABALHOS RELACIONADOS

Diversos estudos serviram de referência e inspiração para este trabalho. Um deles propõe uma técnica automatizada para detectar problemas de L10N em aplicações web (ALAMEER; MAHAJAN; HALFOND, 2016). Outro apresenta uma ferramenta voltada para auxiliar testadores, reduzindo o tempo necessário para testes manuais e ampliando a cobertura dos testes (FELIPE; MIRANDA, 2023).

Entre essas pesquisas, a mais relevante para este estudo é a de Lucena, que desenvolveu uma ferramenta automatizada para identificar falhas de tradução em interfaces de aplicativos móveis. Essa solução analisa capturas de tela por meio de OCR e as compara com arquivos XML contendo a hierarquia da interface (LUCENA, 2024). O presente trabalho se baseia fortemente nessa abordagem, buscando expandir as funcionalidades da ferramenta proposta.

4 METODOLOGIA

Este estudo adota uma abordagem empírica para avaliar a eficácia da ferramenta na detecção automática de falhas de L10N em aplicativos móveis. Essa abordagem permite testar a ferramenta em cenários reais, garantindo sua aplicabilidade prática e sua relevância para o uso em ambientes reais.

4.1 ESCOLHA DA ABORDAGEM

Optou-se por aprimorar a ferramenta(LUCENA, 2024) proposta por Lucena, pois ela já se mostrava uma solução robusta para a detecção de erros de tradução em interfaces gráficas. A ferramenta original utilizava reconhecimento óptico de caracteres (OCR) para identificar falhas a partir de capturas de tela, comparando-as com arquivos *Extensible Markup Language* (XML) que descrevem a hierarquia da interface.

Ao invés de desenvolver uma solução do zero, a escolha de partir dessa base permitiu concentrar esforços na melhoria da ferramenta e na ampliação de suas funcionalidades. Essas melhorias e novas funcionalidades serão detalhadas na seção 5.

4.2 PROCESSO DE TESTE

A validação da ferramenta será realizada por meio da análise de aplicativos reais, seguindo os passos abaixo:

1. **Seleção dos Aplicativos:** serão selecionados cinco aplicativos de diferentes categorias, para que haja maior diversificação.

2. **Execução da Ferramenta:** com o arquivo *apk* em mãos, a ferramenta já poderá ser executada, aproveitando as melhorias implementadas para a extração automatizada das imagens e da hierarquia da interface, sem a necessidade de intervenção manual (mais detalhes na próxima seção).

3. **Registro dos Resultados:** após a execução, serão registrados os seguintes dados: a quantidade de palavras não traduzidas ou traduzidas incorretamente, o número de erros de elipse, número total de telas analisadas e número de telas com erros. Como algumas

palavras são específicas de certos aplicativos, será fornercida uma lista de termos a serem ignorados durante a análise, como, por exemplo, a palavra *"Microsoft"* no aplicativo *Microsoft 365 Copilor*. Serão comparados os resultados com e sem essa lista de palavras a ignorar.

5 FERRAMENTA

A ferramenta que inspirou o presente trabalho(LUCENA, 2024) já proporcionava uma boa solução, mas a limitação estava na necessidade de extrair manualmente os arquivos XML e imagens, além da detecção apenas de um tipo de problema de L10N. Nesta seção, serão destacadas as melhorias implementadas, além das tecnologias utilizadas. Além disso, serão detalhados os parâmetros da nova ferramenta e seu processo de desenvolvimento.

5.1 TECNOLOGIAS UTILIZADAS

A linguagem *Python* foi escolhida para o desenvolvimento da ferramenta e as seguintes tecnologias foram utilizadas:

- **Pytesseract**(GOOGLE; CONTRIBUTORS, 2023)**:** biblioteca Python para OCR, utilizada para realizar o reconhecimento de texto nas imagens capturadas. É um *wrapper* de *Python* para o Tesseract (SMITH; GOOGLE, 2023). A acurácia do Tesseract para imagens nítidas varia de 80% a 95%, dependendo do tamanho da fonte, do fundo ser uniforme ou não, etc.
- **Python Imaging Library**(CLARK; CONTRIBUTORS, 2023)**:** usada para manipulação de imagens, como destaque de palavras mal traduzidas ou identificadas como falhas de L10N.
- **Argparse**(FOUNDATION, 2023)**:** biblioteca para criação de interfaces de linha de comando, permitindo o controle da execução da ferramenta de forma simples.
- **Droidbot**(LI et al., 2017)**:** ferramenta para automatizar a navegação em aplicativos Android. Ela foi utilizada para gerar o *dump* de arquivos *JavaScript Object Notation* (JSON) com a hierarquia da interface e capturar imagens PNG das telas dos aplicativos.

5.2 MELHORIAS IMPLEMENTADAS

A principal melhoria foi a automação do processo de extração de dados, utilizando o Droidbot para navegar automaticamente pelos aplicativos e realizar o dump de JSON e imagens PNG. A partir dessa captura automática, as imagens e os arquivos JSON passam por um

processo de filtragem, pois muitos desses dados são repetidos ou referem-se a *prints* do *launcher* do dispositivo, que não são relevantes para a análise.

Após a filtragem, a ferramenta inicia a análise dos arquivos JSON, buscando palavras que possam estar não traduzidas ou mal traduzidas. Em seguida, realiza-se a leitura OCR nas imagens, identificando os problemas encontrados nos JSONs.

Além disso, a ferramenta agora também detecta elipses (os três pontos "..."), que podem ser erros de tradução ou simplesmente elementos intencionais no *design* do aplicativo. Esse aspecto é crucial para uma análise mais precisa, já que nem todas as elipses representam falhas de L10N. Essa distinção será considerada durante a análise dos resultados.

Por fim, os erros identificados são destacados nas imagens utilizando a biblioteca *Python Imaging Library* (PIL), tornando a visualização e a correção dos problemas mais eficientes.

5.3 PARÂMETROS

A ferramenta possui os seguintes parâmetros:

- **apk-path:** especifica o caminho para o arquivo APK do aplicativo que será analisado.
- **exploration-time:** define o tempo, em segundos, que o *droidbot* deve passar navegando pelo aplicativo, gerando os arquivos JSON e as imagens PNG.
- **custom-ignore:** permite ao usuário fornecer uma lista personalizada de palavras que devem ser ignoradas durante a análise
- **output-directory:** especifica o diretório onde os resultados da análise serão armazenados.
- **lang:** especifica o idioma do aplicativo a ser analisado. Português Brasileiro (pt-br) e Espanhol (es) são suportados.

5.4 DESAFIOS DURANTE O DESENVOLVIMENTO

Durante o desenvolvimento da ferramenta, diversos desafios foram enfrentados e superados. Um dos primeiros obstáculos foi a escolha da abordagem para verificar se as palavras extraídas das imagens e arquivos JSON estavam presentes no idioma escolhido. Inicialmente, considerou-se o uso da biblioteca *PyMultiDictionary*, porém, verificou-se que ela não possuía

um vocabulário suficientemente amplo e apresentava alta latência, resultando em respostas lentas. Para contornar esse problema, optou-se por utilizar dicionários em português e espanhol de fontes confiáveis (IME/USP, 2024)(LERíN, 2024), o que proporcionou um desempenho significativamente melhor e uma cobertura mais abrangente dos idiomas.

Outro desafio surgiu com a navegação automatizada via *DroidBot*, que agora permite ao usuário explorar o aplicativo por tempo indeterminado. Essa maior liberdade gerou um problema de desempenho, pois a exploração prolongada pode resultar em um grande volume de arquivos JSON e imagens. No trabalho de Lucena, os resultados eram apresentados em relatórios Hypertext Markup Language (HTML), o que funcionava bem para análises mais limitadas. No entanto, com o aumento da quantidade de dados coletados, a criação de arquivos HTML se tornaria impraticável devido ao tamanho excessivo dos documentos.

Para resolver essa questão, optou-se por um formato mais simples e direto: em vez de gerar um relatório HTML extenso, a ferramenta agora organiza os resultados em um relatório em texto, listando as palavras detectadas, e inclui as imagens com as palavras destacadas. Tudo é armazenado na pasta output-directory definida pelo usuário, garantindo uma visualização mais prática e eficiente das informações coletadas.

6 AVALIAÇÃO

6.1 ESCOLHA DOS APLICATIVOS

Para a escolha dos aplicativos testados, adotou-se uma abordagem que prioriza diversidade. Parte dos aplicativos utilizados por Lucena foi mantida para permitir uma análise comparativa entre os resultados obtidos por sua ferramenta e pela proposta deste trabalho. No entanto, também foram incluídos novos aplicativos, com o objetivo de ampliar a pluralidade dos cenários testados. Essa adição busca avaliar o desempenho da ferramenta em diferentes contextos, considerando uma variedade maior de interfaces, domínios e abordagens de desenvolvimento, garantindo assim uma validação mais abrangente de sua eficácia na detecção de falhas de L10N.

Os aplicativos escolhidos foram:

- *Blood Pressure Monitor*: aplicativo para monitoramento da pressão arterial, permitindo o registro e análise de medições ao longo do tempo.
- *MoneyManagerEx*: ferramenta de gerenciamento financeiro de código aberto que auxilia no controle de despesas, receitas e planejamento orçamentário.
- *KitchenOwl*: aplicativo de código aberto para gerenciamento de listas de compras e planejamento de refeições, permitindo a organização de receitas e o controle de estoque de alimentos.
- *Kahoot!*: plataforma de aprendizado baseada em jogos interativos, amplamente utilizada em contextos educacionais para *quizzes* e testes.
- *Microsoft 365 Copilot*: assistente de produtividade baseado em IA, integrado ao ecossistema Microsoft 365 para otimizar tarefas e fluxos de trabalho.
- *VLC Media Player*: reprodutor de mídia de código aberto que suporta uma ampla variedade de formatos de áudio e vídeo, além de oferecer recursos como transmissão de rede e conversão de arquivos.
- *Sofascore*: aplicativo que fornece resultados esportivos ao vivo, estatísticas detalhadas e análises aprofundadas de diversas modalidades esportivas.

- *IMDb*: plataforma que oferece informações abrangentes sobre filmes, séries de TV e celebridades, incluindo sinopses, avaliações, trailers e detalhes de elenco.

  O aplicativos *Kahoot!*, *Microsoft 365 Copilot* ,*Sofascore*, *VLC Media Player* e *IMDb* estão disponíveis na *PlayStore*(LLC, 2024). Já o *MoneyManagerEx*, o *Blood Pressure Monitor* e o *KitchenOwl* estão dispníveis na *F-Droid*(LIMITED, 2024), que é uma loja de aplicativos alternativa para Android que distribui exclusivamente softwares de código aberto.

  A avaliação de Lucena incluiu os aplicativos *Kahoot!*, *Microsoft 365* (antiga versão do *Microsoft 365 Copilot*), *MoneyManagerEx* e *Blood Pressure Monitor*.

6.2 RESULTADOS

Abaixo estão os resultados obtidos para todos os aplicativos analisados. Antes de apresentá-los, algumas explicações são necessárias para a correta interpretação dos dados:

- Arquivo *ignore* trata-se de um arquivo contendo palavras específicas de um aplicativo que devem ser ignoradas durante a análise.
- O Filtro remove imagens idênticas e aquelas pertencentes ao *launcher* do sistema, garantindo que apenas elementos relevantes sejam analisados.
- Falsos positivos de elipse são casos em que a ferramenta detectou um possível erro de elipse, mas, após uma análise manual detalhada, constatou-se que se tratava de uma elipse intencional ou de uma elipse que ocorreria independentemente do idioma do aplicativo. Por esse motivo, tais casos não são considerados erros de L10N.
- Nas Tabelas de resultados, a coluna "Erros de elipse"contabiliza apenas os erros reais de elipse, excluindo os falsos positivos mencionados acima.

Todos os dados e arquivos *ignore* estão disponíveis no repositório público da ferramenta no *GitHub*(COELHO, 2025).

6.2.1 *MoneyManagerEx*

Conforme demonstrado na Tabela 1, o aplicativo apresenta alta incidência de erros de L10N, com destaque para:

Tabela 1 – Resultados da Análise para o *MoneyManagerEx*

Arquivo *ignore*	**Imagens antes do filtro**	**Imagens depois do filtro**	**Imagens com erro de L10N**	**Palavras com erro de tradução**	**Erros de elipse**	**Falsos positivos de erros de elipse**
Não	102	99	67	105	0	1
Sim	102	99	54	87	0	1

- Problemas de tradução: oitenta e sete ocorrências identificadas, mesmo com o uso de arquivo *ignore* para termos específicos.
- Elipses: apenas um falso positivo detectado (elipse intencional da interface).

Estes resultados sugerem que, embora o aplicativo implemente corretamente elementos gráficos (elipses), possui desafios significativos na qualidade das traduções. As Figuras 5 e 6 contém alguns erros de L10N do *MoneyManagerEx*.

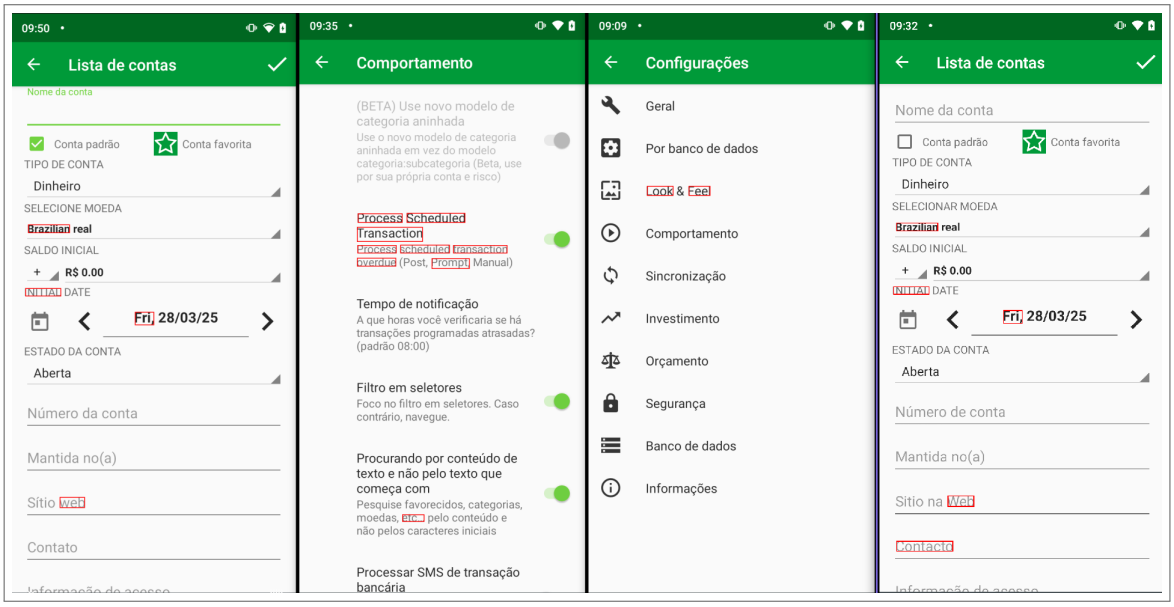

Figura 5 – Problemas de L10N do *MoneyManagerEx*

6.2.2 *Blood Pressure Monitor*

Conforme demonstrado na Tabela 2, o aplicativo apresenta baixa incidência de erros de L10N, com destaque para:

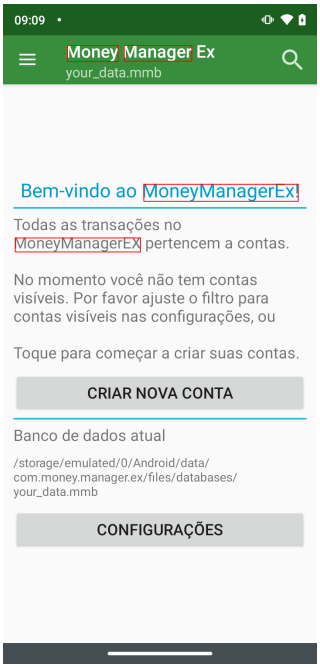

Figura 6 – Sem o arquivo *ignore*, palavras como o nome do aplicacativo são erroneamente detectadas como erros no *MoneyManagerEx*

Tabela 2 – Resultados da Análise para o *Blood Pressure Monitor*

Arquivo *ignore*	**Imagens antes do filtro**	**Imagens depois do filtro**	**Imagens com erro de L10N**	**Palavras com erro de tradução**	**Erros de elipse**	**Falsos positivos de erros de elipse**
Não	65	30	0	0	0	2
Sim	65	30	0	0	0	2

- Problemas de tradução: zero erro na interface principal (os 2 casos detectados ocorreram no seletor de arquivos do *Drive*, componente externo).
- Elipses: dois falsos positivos foram encontrados (também na interface do seletor de arquivos).
- Filtro: Trinta e cinco imagens desnecessárias foram removidas. Pode indicar um aplicativo simples (telas repetidas).

Os resultados mostram que o aplicativo possui excelente preparação para problemas de L10N.

6.2.3 *Microsoft 365 Copilot*

Tabela 3 – Resultados da Análise para o *Microsoft 365 Copilot*

Arquivo *ignore*	**Imagens antes do filtro**	**Imagens depois do filtro**	**Imagens com erro de L10N**	**Palavras com erro de tradução**	**Erros de elipse**	**Falsos positivos de erros de elipse**
Não	114	107	68	40	12	4
Sim	114	107	25	4	12	4

Da Tabela 3 destaca-se o seguinte:

- Problemas de tradução: Baixa incidência com arquivo *ignore*. Indica que muitas das palavras com erro são específicas do aplicativo (*Edge*, *Copilot*, etc).
- Elipses: Alta incidência de elipses.

Estes resultados sugerem que, embora o aplicativo possua boa tradução, pode melhorar bastante na implementação de elementos gráficos. As Figuras 7 e 8 contêm alguns erros de L10N do *Microsoft 365 Copilot*.

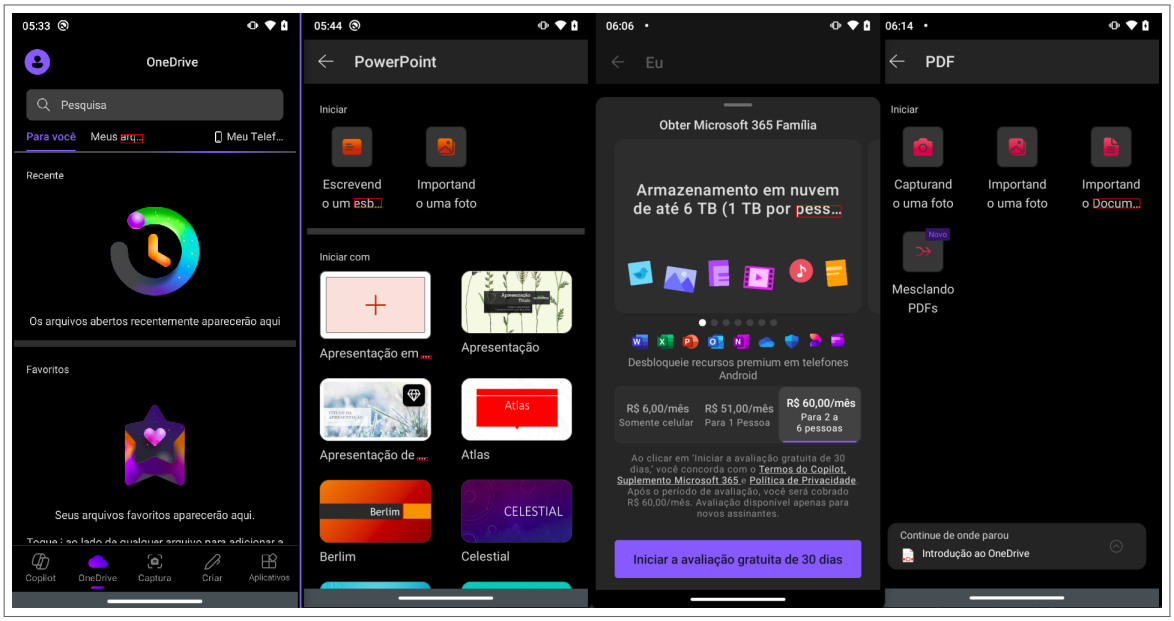

Figura 7 – Problemas de L10N do *Microsoft 365 Copilot*

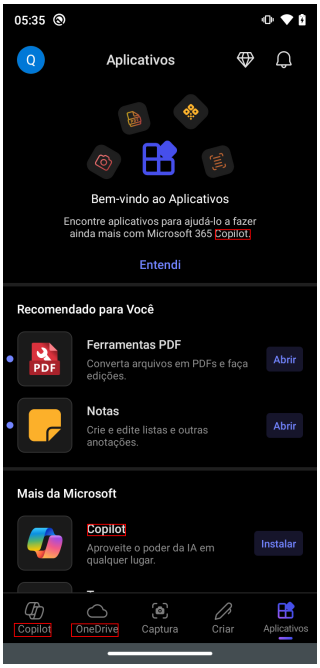

Figura 8 – Sem o arquivo *ignore*, palavras como *Coplitot* e *OneDrive* são erroneamente detectadas como erros no *Microsoft 365 Copilot*

Tabela 4 – Resultados da Análise para o *Kahoot!*

Arquivo *ignore*	**Imagens antes do filtro**	**Imagens depois do filtro**	**Imagens com erro de L10N**	**Palavras com erro de tradução**	**Erros de elipse**	**Falsos positivos de erros de elipse**
Não	95	92	60	70	2	6
Sim	95	92	19	40	2	6

6.2.4 *Kahoot!*

Da Tabela 4 destaca-se o seguinte:

- Problemas de tradução: aplicativo pode melhorar a tradução, já que há 40 incidências mesmo com arquivo *ignore*.
- Elipses: seis falsos positivos (problemas não apenas na versão traduzida, mas em todos idiomas).

Estes resultados sugerem que o aplicativo pode melhorar bastante a sua tradução. As Figura 9 e 10 contêm alguns erros de L10N do *Kahoot!*.

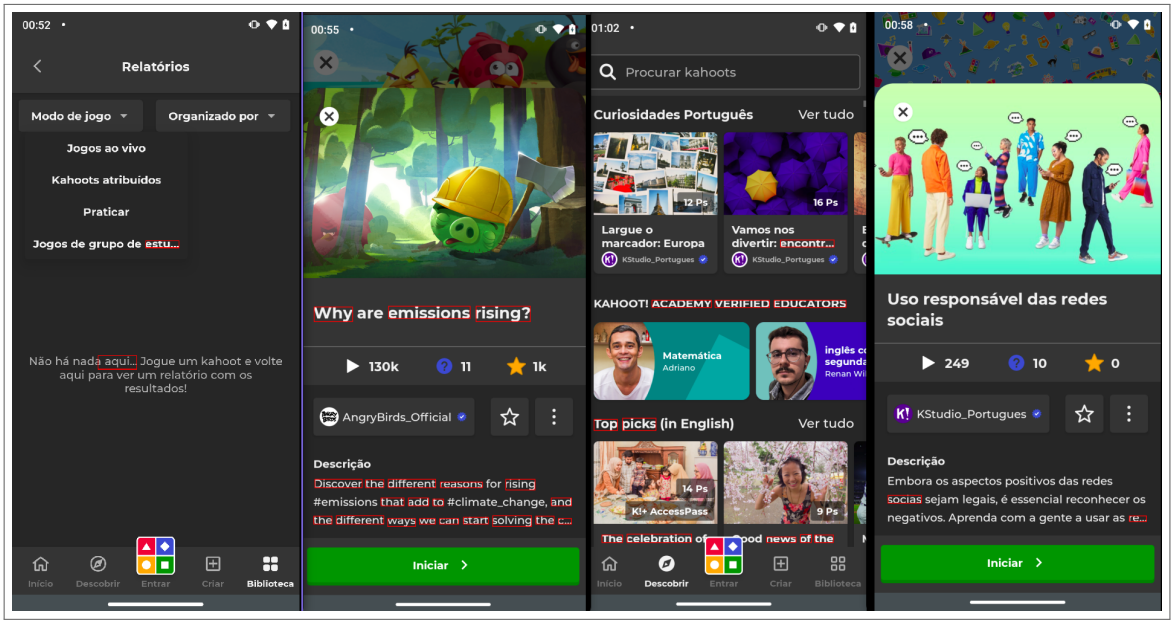

Figura 9 – Problemas de L10N do *Kahoot!*

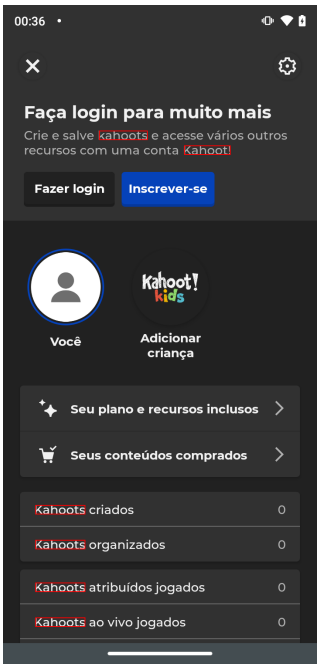

Figura 10 – Sem o arquivo *ignore*, o nome do aplicativo é erroneamente detectado como falha de L10N no *Kahoot!*

6.2.5 *Sofascore*

Da Tabela 5 destaca-se o seguinte:

- Problemas de tradução: a maioria dos erros detectados são nomes de times, competições, jogadores, etc. e somem com a introdução do arquivo *ignore*.

Tabela 5 – Resultados da Análise para o *Sofascore*

Arquivo *ignore*	**Imagens antes do filtro**	**Imagens depois do filtro**	**Imagens com erro de L10N**	**Palavras com erro de tradução**	**Erros de elipse**	**Falsos positivos de erros de elipse**
Não	141	138	108	183	4	0
Sim	141	138	11	3	4	0

- Elipses: alguns pequenos erros são encontrados e podem ser vistos na Figura 11.

Estes resultados indicam um aplicativo bem preparado para a erros de L10N, considerando o número expressivo de telas analisadas.

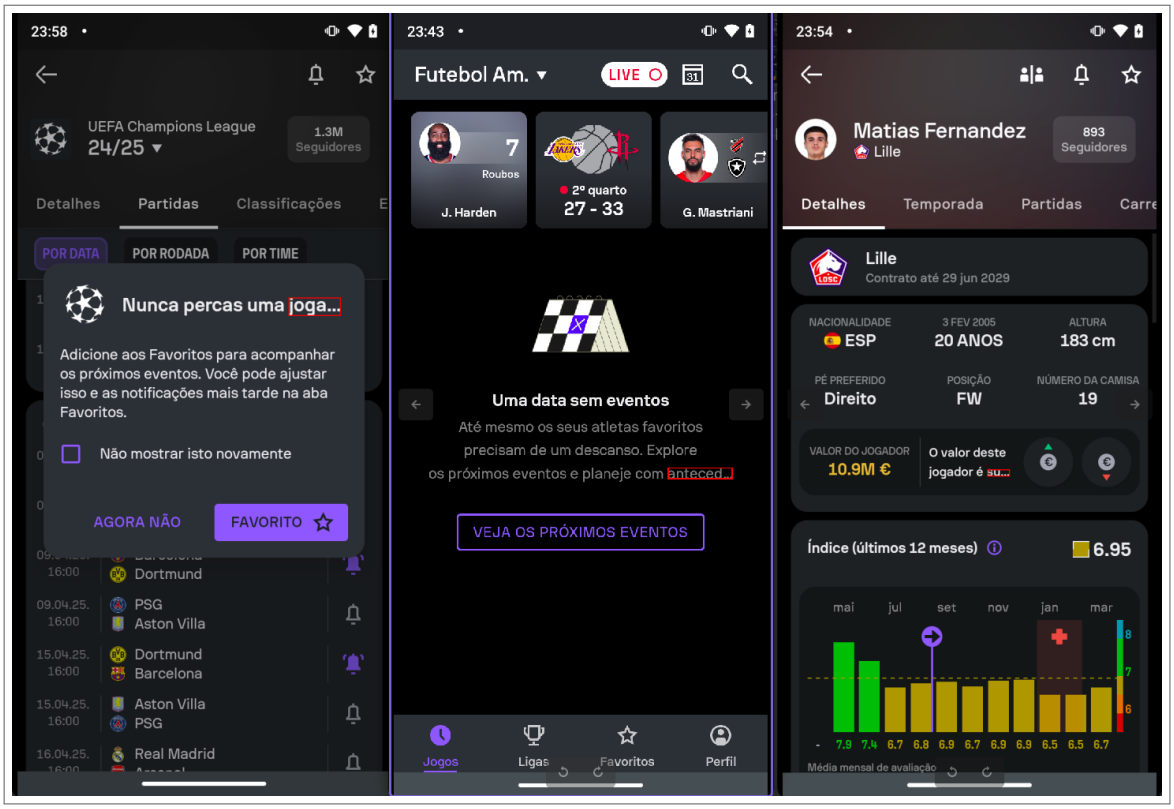

Figura 11 – Problemas de L10N do *Sofascore*

6.2.6 *IMDb*

Da Tabela 6 destaca-se o seguinte:

- Problemas de tradução: os títulos dos filmes e sua descrição muitas vezes não são traduzidos. Além disso, contém seções totalmente não traduzidas.

Figura 12 – Sem o arquivo *ignore*, nomes de times são detectados como erro de tradução no *Sofascore*

Tabela 6 – Resultados da Análise para o *IMDb*

Arquivo *ignore*	**Imagens antes do filtro**	**Imagens depois do filtro**	**Imagens com erro de L10N**	**Palavras com erro de tradução**	**Erros de elipse**	**Falsos positivos de erros de elipse**
Não	61	57	41	180	0	13
Sim	61	57	33	125	0	13

- Elipses: o aplicativo possui muitas elipses que não podem ser consideradas erros de L10N por existirem independentemente do idioma.

Estes resultados indicam um aplicativo muito pouco preparado para erros de L10N, com muitas falhas de tradução. As Figuras 13 e 14 demonstram alguns erros no *IMDb*.

6.2.7 *VLC*

Da Tabela 7 destaca-se o seguinte:

- Problemas de tradução: nenhum erro de tradução depois da introdução do arquivo *ignore*.
- Elipses: o aplicativo possui apenas um erro sério, que está presente em muitas imagens. A palavra *Playlist* foi traduzida para Listas de Reprodução e introduziu uma elipse na

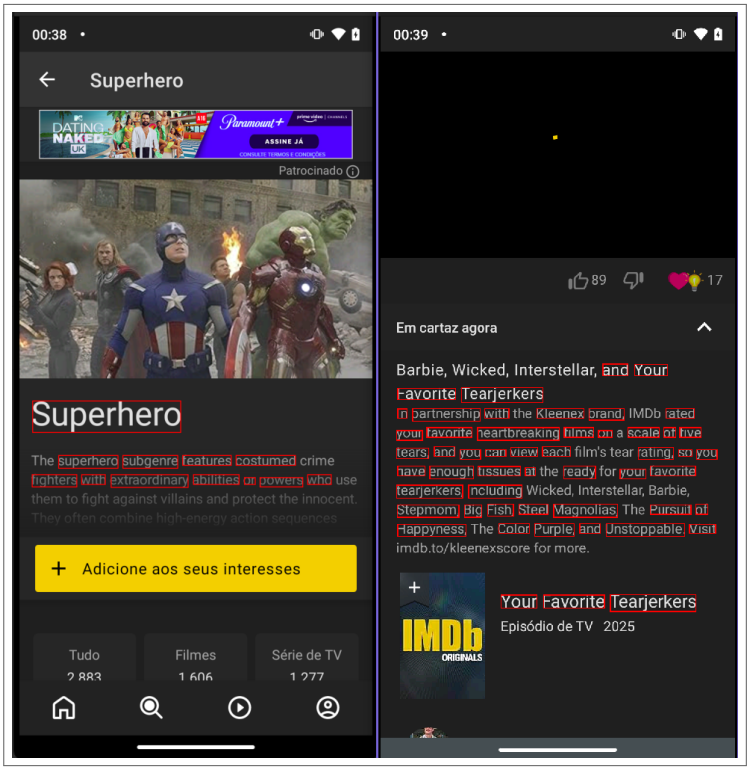

Figura 13 – Problemas de L10N do *IMDb*

Figura 14 – Sem o arquivo *ignore*, o nome do aplicativo é erroneamente detectado como falha no *IMDb*

barra de navegação, como pode ser visto na Figura 15.

Estes resultados indicam um aplicativo bem preparado para problemas de L10N.

Tabela 7 – Resultados da Análise para o *VLC*

Arquivo *ignore*	**Imagens antes do filtro**	**Imagens depois do filtro**	**Imagens com erro de L10N**	**Palavras com erro de tradução**	**Erros de elipse**	**Falsos positivos de erros de elipse**
Não	105	104	86	6	1	3
Sim	105	104	85	0	1	3

Figura 15 – Problema de elipse na barra de navegação do *VLC*

6.2.8 *KitchenOwl*

Tabela 8 – Resultados da Análise para o *KitchenOwl*

Arquivo *ignore*	**Imagens antes do filtro**	**Imagens depois do filtro**	**Imagens com erro de L10N**	**Palavras com erro de tradução**	**Erros de elipse**	**Falsos positivos de erros de elipse**
Não	25	20	5	3	0	0
Sim	25	20	0	0	0	0

Da Tabela 8 destaca-se o seguinte:

- Problemas de tradução: os únicos erros de tradução são provenientes de *inputs* do *DroidBot*. Com o arquivo *ignore*, não há erro. A Figura 16 mostra os erros detectados

antes sem o arquivo *ignore*.

- Elipses: não foram encontrados erros de elipse no *KitchenOwl*.

A interface do aplicativo é simples e o DroidBot teve problemas para explorar novas telas. Das poucas telas exploradas, todas tiveram tradução e interface coerente.

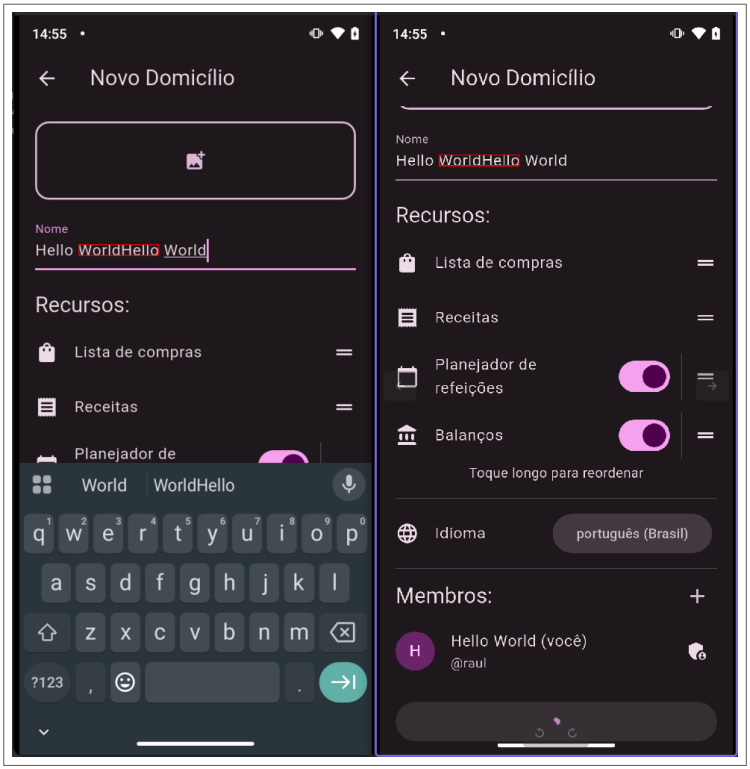

Figura 16 – Antes do arquivo *ignore* ser introduzido, input do *DroidBot* é identificado como erro

6.3 MELHORIAS

A integração da ferramenta com o *DroidBot* permite a realização de testes em um número significativamente maior de imagens, superando a abordagem utilizada no trabalho de Lucena. Além disso, o uso do *PyTesseract* demonstrou um alto desempenho, sendo capaz de processar mais de 100 imagens em menos de um minuto, o que torna a análise mais eficiente. Outra melhoria relevante é a inclusão da detecção de elipses como uma funcionalidade adicional, ampliando a capacidade da ferramenta na identificação de erros de localização.

6.4 LIMITAÇÕES

A ferramenta atualmente se limita à detecção de erros relacionados à ausência de tradução ou elipse. Uma solução mais abrangente também deveria identificar problemas como truncamento e sobreposição de texto. Além disso, a análise empírica realizada com oito aplicativos pode não ser suficiente para uma validação completa, sendo ideal a realização de testes adicionais para reforçar os resultados.

7 CONCLUSÃO E DIREÇÕES FUTURAS

A ferramenta desenvolvida demonstrou sucesso na detecção de erros de tradução e elipses, mesmo diante dos desafios inerentes ao processo de localização. As melhorias implementadas, como a integração com o *DroidBot* e o alto desempenho do *PyTesseract*, permitiram uma análise eficiente de um grande número de imagens, tornando a ferramenta mais robusta.

Há, contudo, espaço para aprimoramento. Um dos caminhos mais promissores para trabalhos futuros é a incorporação de inteligência artificial para aprimorar a detecção de erros de L10N. Um experimento realizado com o *ChatGPT* — cujo *prompt* está no *GitHub*(COELHO, 2025) da aplicação — demonstrou o potencial dessa abordagem. Nesse teste, foram fornecidas imagens contendo erros de L10N, e o modelo foi capaz de identificar corretamente todos os problemas apresentados, incluindo um truncamento. Esse resultado sugere que a integração de técnicas baseadas em *Inteligência Artificial (IA)* pode tornar a ferramenta ainda mais precisa e adaptável, ampliando sua capacidade de detecção e permitindo a identificação de uma gama mais extensa de falhas de localização. A Figura 17 mostra um trecho da conversa com o ChatGPT.

Dessa forma, este trabalho representa um avanço na detecção automatizada de erros de L10N, mas abre caminho para novas investigações que possam explorar abordagens mais sofisticadas.

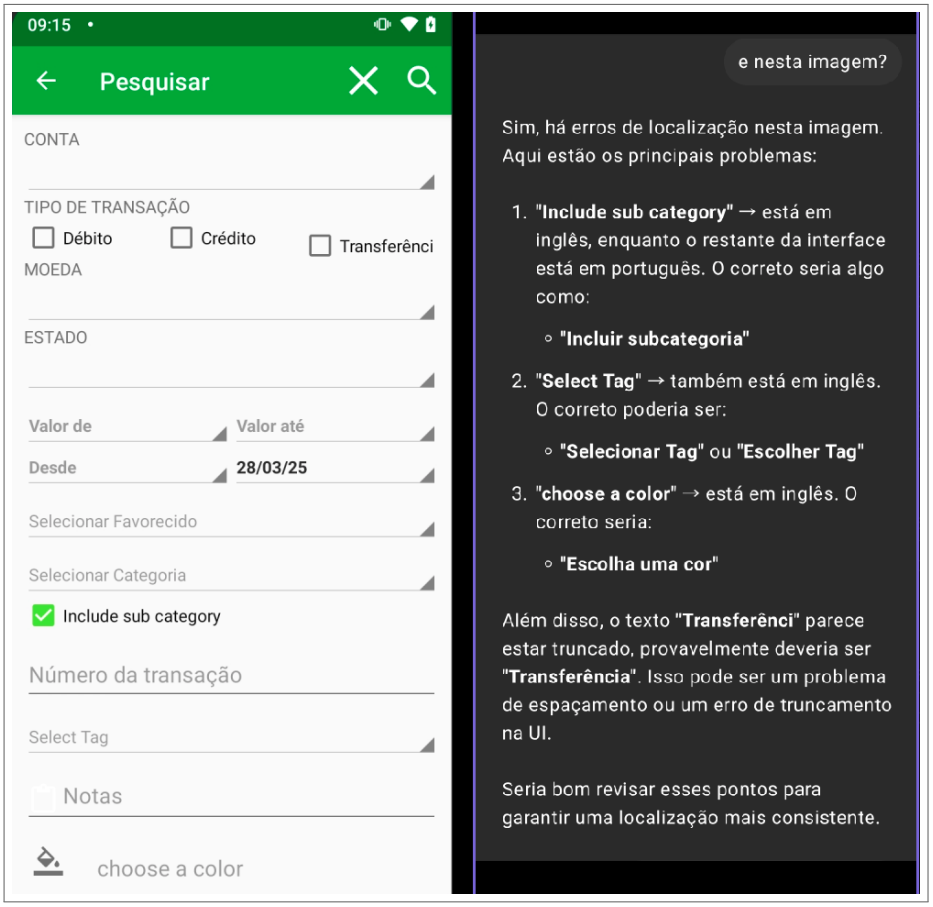

Figura 17 – Trecho da conversa com o ChatGPT mostra o potencial da IA na detecção de problemas de L10N

REFERÊNCIAS

ALAMEER, A.; MAHAJAN, S.; HALFOND, W. G. J. Detecting and localizing internationalization presentation failures in web applications. In: *2016 IEEE International Conference on Software Testing, Verification and Validation (ICST)*. [s.n.], 2016. p. 202–212. Disponível em: <https://ieeexplore.ieee.org/document/7515472>.

CLARK, A.; CONTRIBUTORS. *Pillow (PIL Fork) Documentation*. 2023. <https://pillow.readthedocs.io/>. Versão 10.0.0.

COELHO, R. *l10n-auto: Ferramenta para detecção de problemas de localização*. [S.l.]: GitHub, 2025. <https://github.com/raulpcoelho/l10n-auto-detection>.

COUTO, M.; MIRANDA, B. An industrial experience report on the challenges in training localization and internationalization testers. In: *Proceedings of the 8th Brazilian Symposium on Systematic and Automated Software Testing*. [S.l.: s.n.], 2023. p. 96–98.

EIKVIL, L. Optical character recognition. v. 26, 1993. Citado nas páginas 1, 4. Disponível em: <http://citeseer.ist.psu.edu/142042.html>.

FELIPE, L. P.; MIRANDA, B. Supporting localization testing through automated application navigation. In: *Anais Estendidos do XXII Simpósio Brasileiro de Qualidade de Software*. [S.l.]: SBC, 2023. p. 25–30.

FOUNDATION, P. S. *argparse — Parser for command-line options, arguments and subcommands*. 2023. <https://docs.python.org/3/library/argparse.html>. Python Standard Library.

GOOGLE; CONTRIBUTORS. *pytesseract: Python-tesseract*. 2023. <https://pypi.org/project/pytesseract/>. OCR tool wrapper.

GROSS, S. *Internationalization and localization of software*. Dissertação (Mestrado) — Eastern Michigan University, Department of Computer Science, Ypsilanti, Michigan, June 2006. Master of Science in Computer Science.

IME/USP, D. de Ciência da C. *Dicionário de palavras portuguesas*. 2024. Disponível em: <https://www.ime.usp.br/~pf/dicios/>.

LERíN, J. D. *Diccionario español (archivo de texto)*. 2024. <https://github.com/JorgeDuenasLerin/diccionario-espanol-txt/blob/master/0_palabras_todas.txt>.

LI, Y. et al. *Droidbot: A lightweight test input generator for Android*. 2017. <https://github.com/honeynet/droidbot>. Acesso em 27 fev. 2025.

LIMITED, F.-D. *F-Droid: Free and Open Source Android App Repository*. 2024. <https://f-droid.org/>. Versão 1.15, acesso em: 05 jan. 2025.

LLC, G. *Google Play Store*. 2024. <https://play.google.com/store>. Acesso em: 05 jan. 2025.

LUCENA, M. *Detecção Automática de Falhas de Localização em Aplicativos Android*. Trabalho de Graduação — Universidade Federal de Pernambuco, 2024.

SMITH, R.; GOOGLE. *Tesseract OCR*. 2023. <https://github.com/tesseract-ocr/tesseract>. Versão 5.3.0.

ZALIESKAITė, N. *Automating detection of software cultural elements inconsistencies with locale norms*. Dissertação (Master's thesis) — Vilnius University, Department of Mathematics and Informatics Faculty, 2023. Disponível em: <https://epublications.vu.lt/object/elaba:192827410>.