Universidade Federal de Juiz de Fora
Instituto de Ciências Exatas
Bacharelado em Sistemas de Informação

Tutor - PT: Sistema Web para Avaliação Semiautomática de Monografias em Língua Portuguesa

João Pedro Sequeto

JUIZ DE FORA
SETEMBRO, 2024

Tutor - PT: Sistema Web para Avaliação Semiautomática de Monografias em Língua Portuguesa

João Pedro Sequeto

Universidade Federal de Juiz de Fora

Instituto de Ciências Exatas

Departamento de Ciência da Computação

Bacharelado em Sistemas de Informação

Orientador: Fabrício Martins Mendonça

JUIZ DE FORA

SETEMBRO, 2024

Tutor - PT: Sistema Web para Avaliação Semiautomática de Monografias em Língua Portuguesa

João Pedro Sequeto

MONOGRAFIA SUBMETIDA AO CORPO DOCENTE DO INSTITUTO DE CIÊNCIAS EXATAS DA UNIVERSIDADE FEDERAL DE JUIZ DE FORA, COMO PARTE INTEGRANTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE BACHAREL EM SISTEMAS DE INFORMAÇÃO.

Aprovada por:

Fabrício Martins Mendonça
Doutor em Ciência da Informação UFMG

Alessandreia Marta de Oliveira Julio
Doutora em Computação UFF

Luciana Dias Campos Conceição
Doutora em Engenharia Elétrica UFJF

JUIZ DE FORA

23 DE SETEMBRO, 2024

Aos meus amigos e familiares.

Aos pais, pelo apoio e sustento.

Resumo

A busca por soluções tecnológicas na área da educação tem aumentado cada vez mais. O uso de ferramentas computacionais que auxiliem professores e alunos no processo de aprendizado e ensino se torna relevante para promover um ambiente educacional mais dinâmico e eficiente. O objetivo deste trabalho é desenvolver um *software* para avaliação semiautomática de textos em Língua Portuguesa, o qual foi denominado como Tutor-PT. O *software* desenvolvido foi integrado ao Google Docs, ferramenta *web* de edição de textos, para permitir a análise de textos redigidos em língua portuguesa, oferecendo *feedback* e sugestões de melhorias, assim como sugestões de pontuações a cada atributo avaliado. O *software* desenvolvido foi avaliado considerando algumas métricas de desempenho, acurácia e taxa de reconhecimento de erros sob uma base de textos de redação anotados disponíveis na *web*, analisando assim suas funcionalidades para identificação de erros ortográficos e gramaticais, sugestão de melhorias no estilo e clareza dos textos. Os resultados da análise foram modelados graficamente e mostram-se satisfatórios para os objetivos pretendidos nesta pesquisa.

Palavras-chave: Avaliação Automática de Redação, Correção Automática de Textos, Educação, *software*, Processamento de Linguagem Natural.

Abstract

The search for technologies in education are growing every time. The use of computational tools that support teachers and students in the learning and teaching process becomes relevant to promote a more dynamic and efficient educational environment. The goal of this work is develop a software to semi-automatic evaluation of texts in Portuguese, which named as Tutor-PT. The software developed was integrated to Google Docs, web tool for text editing, in a way to enable the analysis of texts written in Portuguese, providing feedback and suggestions for improvement, as well as offering scoring suggestions for each evaluated attribute. The software developed was evaluated considering some metrics like as performance, accuracy and error recognition rate on a text basis of essays annotated available in web, analysing their functions for identification of spelling and grammatical errors, suggesting of improvements in style and clarity of texts. The results of analysis were modelled graphically and its are satisfactory to the purposes of this research.

Keywords: Automated Essay Scoring, Automatic Text Correction, educational, software, Natural Language Processing.

Agradecimentos

A todos os meus parentes, pelo encorajamento e apoio.

Ao professor Fabrício pela orientação, amizade e principalmente, pela paciência, sem a qual este trabalho não se realizaria.

Aos professores do Departamento de Ciência da Computação pelos seus ensinamentos e aos funcionários do curso, que durante esses anos, contribuíram de algum modo para o nosso enriquecimento pessoal e profissional.

"Lembra que o sono é sagrado e alimenta de horizontes o tempo acordado de viver".

Beto Guedes (Amor de Índio)

Conteúdo

Lista de Figuras

Lista de Tabelas

Lista de Abreviações

DCC	Departamento de Ciência da Computução
UFJF	Universidade Federal de Juiz de Fora
AAR	Avaliação Automática de Redação
PLN	Processamento de Linguagem Natural
ML	*Machine Learning*
ENEM	Exame Nacional do Ensino Médio
LSTM	*Long Short-Term Memory*

1 Introdução

Nesse capítulo, é abordado o tema da pesquisa, a justificativa do trabalho, os objetivos e metodologia utilizada durante o desenvolvimento do TUTOR-PT, um *software* de avaliação semiautomática de monografias em língua portuguesa.

1.1 Apresentação do tema

Com o avanço da tecnologia e a demanda por formas modernas e inovadoras de ensino, ocorreu um aumento da busca por soluções tecnológicas que possam auxiliar na compreensão, instrução e avaliação da escrita (WILSON; ROSCOE, 2020). Uma área de aplicação que avançou nesse contexto foi a avaliação e correção automática de textos de redação.

A avaliação automática de redação é uma área de aplicação educacional com início a partir do projeto *Project Essay Grader*, utilizando técnicas de processamento de linguagem natural (PLN). (KE; NG, 2019). As aplicações de avaliação automática de redação (AAR) utilizam algoritmos de análise sintática e semântica em união com algoritmos estatísticos para obter uma análise do texto escrito (JUNIOR, 2021). Nos primeiros anos de estudo, essas ferramentas tinham como objetivo apenas a avaliação do texto em geral com uma única pontuação baseada em diversos fatores. Porém, ao longo do tempo, também ocorreram pesquisas considerando outras abordagens com maior foco em dimensões específicas do texto, como estrutura, ortografia e gramática (KE; NG, 2019). Considerando a Língua Portuguesa, não tivemos os mesmos avanços que já foram obtidos com relação à Língua Inglesa, que já possui um número expressivo de estudos e aplicações comerciais envolvendo o idioma (COSTA; OLIVEIRA; JÚNIOR, 2020). Ainda assim surgiram estudos abordando a produção de textos em Língua Portuguesa, com foco em diferentes níveis do sistema de ensino brasileiro, como, por exemplo, abordagens com textos nos moldes do Exame Nacional do Ensino Médio (ENEM) (JUNIOR, 2021). Nesse trabalho, é proposto um *software* de avaliação semiautomática de monografias em língua

portuguesa, visando avaliar se o uso de aplicações de AAR podem auxiliar no processo de escrita e avaliação de monografias.

1.2 Justificativa / Motivação

O processo de escrita da monografia é, de maneira geral, complexo, principalmente por parte de estudantes que ainda não estão familiarizados com a redação de textos científicos. A complexidade envolve as dificuldades dos estudantes na escrita dos textos e também de professores e tutores ao realizar a leitura e avaliação desses textos. Esse é um cenário comum em muitos cursos de graduação pelo Brasil. Alguns fatores podem contribuir no distanciamento dos indivíduos pela escrita durante vida escolar, tais como: a tradição oral do atual modelo de ensino ou a quantidade ineficiente de produção textual nos níveis de ensino fundamental e médio (BORGES et al., 2020). Além das dificuldades encontradas com o processo de escrita dos discentes, temos uma sobrecarga dos professores com o processo de avaliação dos textos produzidos, seja no ambiente de ensino superior ou médio, dado que os professores tendem a assumir muitas atividades, como, por exemplo, assumir diversas turmas em um único período (CRISTO, 2020).

Diante desse cenário e da possibilidade de desenvolvimento de ferramentas computacionais que possam auxiliar nesse processo, a construção de uma ferramenta de avaliação semiautomática direcionado à monografia pode auxiliar tanto alunos, quanto professores na redução de tempo e esforço empregados nessas tarefas.

1.3 Questões de pesquisa

O presente trabalho busca implementar e avaliar um software de avaliação semiautomática de textos em Língua Portuguesa integrado à plataforma Google Docs. A partir disso, surge a seguinte questão de pesquisa:

O uso de um software de avaliação semiautomática de textos em Língua Portuguesa, integrado a ambientes online, é uma solução viável para a identificação e correção de erros de escrita?

1.4 Objetivos

1.4.1 Objetivo geral

O **objetivo geral** desse trabalho é o desenvolvimento do sistema TUTOR-PT, uma extensão para Google Docs visando avaliação semiautomática de monografias, com uso de técnicas de processamento de linguagem natural, direcionado a professores, tutores e alunos de cursos de graduação e licenciatura. As funcionalidades do *software* proposto incluem extração de métricas textuais, identificação de erros ortográficos e gramaticais no desenvolvimento dos elementos textuais, pontuação desses erros, além da indicação de correções e melhorias no texto avaliado.

1.4.2 Objetivos específicos

Quanto aos **objetivos específicos** deste trabalho de pesquisa temos os seguintes:

- Desenvolvimento de *software* para avaliação semiautomática direcionado à monografia como uma extensão para o Google Docs.
- Definição de estratégias de avaliação para análise dos resultados do processamento de texto.
- Realização de testes do *software* proposto a partir de um conjunto de textos.
- Análise do tempo médio de processamento da aplicação em textos de redação.
- Análise da taxa de reconhecimento correto de erros da aplicação em textos de redação.
- Análise das categorias de erros mais identificadas pela aplicação.

1.5 Metodologia

A metodologia para realização do trabalho se inicia com a pesquisa pelas principais ferramentas e tecnologias para o desenvolvimento de sistemas *web*, técnicas de processamento

de linguagem natural, métricas de avaliação textual e, por fim, o desenvolvimento da ferramenta de avaliação semiautomática de monografias que auxilia professores e tutores na atividade de correção textual e pontuação.

De acordo com Pereira et al. (2018), é possível classificar o trabalho como uma **pesquisa aplicada**, porque envolve o desenvolvimento de um *software* (produto) a partir da pesquisa realizada, e também como uma **pesquisa bibliográfica**, já que inclui pesquisa à base de dados de artigos e outros textos científicos, como monografias e dissertações. Para cumprir com os objetivos pretendidos da pesquisa, foram elaboradas as seguintes atividades metodológicas:

- Revisar a literatura relacionada ao desenvolvimento de sistemas de avaliação automática.
- Buscar uma base de dados para aplicação e avaliação do *software*.
- Levantar requisitos para modelagem da ferramenta.
- Desenvolver um módulo de avaliação e correção semiautomática de textos.
- Desenvolver uma interface como extensão para o *software* de edição de textos Google Docs.
- Desenvolver uma integração entre a interface e módulos de avaliação e correção.
- Realizar testes com textos nos moldes do ENEM da base de redações da UOL.
- Analisar os resultados dos testes.
- Redigir o trabalho de conclusão de curso.

1.6 Organização do trabalho

O trabalho se organiza em 6 capítulos. No primeiro capítulo, é realizado uma introdução com a apresentação do tema, justificativa, objetivos e metodologia do trabalho. No segundo capítulo, é apresentado a fundamentação teórica, com alguns conceitos e técnicas utilizados no desenvolvimento do trabalho. No capítulo 3, são apresentados os trabalhos

relacionados com o tema do trabalho. No capítulo 4, é apresentado o desenvolvimento do *software* TUTOR-PT, assim como suas restrições. No capítulo 5, é apresentado uma discussão dos resultados obtidos com os testes na aplicação desenvolvida. No capítulo 6, apresentam-se as conclusões do trabalho e propostas de trabalhos futuros.

2 Fundamentação Teórica

Este capítulo pretende apresentar alguns conceitos, técnicas e pesquisas que definem ou discutem elementos usados na elaboração deste trabalho. Esses conceitos são importantes para o entendimento do contexto, motivação e problema que está se investigando, e também são fundamentais para a escolha das técnicas a serem utilizadas ao longo do desenvolvimento, além das métricas de avaliação utilizadas para a análise dos resultados em comparação com outras soluções.

2.1 Avaliação de textos

Para discorrer sobre a definição e conceitos de avaliação de textos, primeiro é preciso definir o que seria uma avaliação. De acordo com MARCURSCHI e SUASSUNA (2007), a avaliação pode ser definida como uma ação processual de construir um valor provisório para o ser focalizado, mediante categorias sociais, culturalmente marcadas e interativamente elaboradas. A ação avaliativa envolve concepções de mundo, conhecimento e emissão de valores por parte do indivíduo avaliador. Na decisão avaliativa, é estabelecido um procedimento comparativo entre o objeto alvo da avaliação e a expectativa estabelecida acerca do objeto e com esses procedimentos estabelecidos, ocorre a avaliação. É um processo com objetivo de ensino e passível de alterações ao decorrer das interações (MARCURSCHI; SUASSUNA, 2007).

Para o cenário de produção textual, a produção e avaliação de textos se estabelecem como atividades relacionadas, que ocorrem muitas vezes em ciclos, visando a melhoria do texto final (MARCURSCHI; SUASSUNA, 2007). Uma vez que a escrita possui dificuldades, o processo de avaliação também irá ser desconfortável para ambas as partes, avaliador e avaliado, e nem sempre o processo avaliativo abrange todos os aspectos necessários, muitas vezes focando apenas nos critérios gramaticais (CRISTO, 2020).

Com o avanço da tecnologia, surgiram algumas ferramentas para nos auxiliar diante dos desafios envolvendo o processo de escrita e avaliação de textos, sendo uma delas

a avaliação automática de redação. A avaliação automática de redação surge como subproduto das técnicas de Processamento de Linguagem Natural e Aprendizado de Máquina (MARINHO et al., 2022). A avaliação automática de redação é a aplicação de tecnologia para avaliar e pontuar automaticamente um texto escrito, sendo uma das principais aplicações educacionais de processamento de linguagem natural (KE; NG, 2019). A maioria dos sistemas de avaliação automática de redação são aplicações desenvolvidas com abordagem de aprendizado supervisionado, onde o objetivo é realizar tarefas de regressão e prever a pontuação de um texto, ou tarefas de classificação para categorizar um texto de acordo com um conjunto de classes (KE; NG, 2019).

2.2 Aprendizado de máquina

De acordo com Murphy (2012), aprendizado de máquina é definido como um conjunto de métodos que permitem a detecção automática de padrões em dados. Esses padrões podem ser usados para fazer previsões sobre dados futuros ou para tomar decisões em situações de incerteza. Com o crescimento da produção de dados, surge a necessidade por métodos de análise de dados capazes de lidar com esse cenário, e o aprendizado de máquina aparece como uma área de pesquisa para fornecer esses métodos (MURPHY, 2012).

Os algoritmos e modelos estudados no aprendizado de máquina são geralmente divididos em dois principais tipo: aprendizado supervisionado e não supervisionado. No aprendizado supervisionado, o objetivo é mapear um conjunto de dados para uma saída de acordo com dados que já foram anteriormente avaliados e rotulados anteriormente. Esse conjunto de dados rotulados que servirá de referência ao sistema é chamado conjunto de treino. Nesse campo de aprendizado supervisionado temos os algoritmos de classificação, regressão, entre outros. O segundo tipo, chamado aprendizado não supervisionado, tem como meta encontrar padrões em um conjunto de dados. Nesse grupo, somente é informado um conjunto de dados como entrada, e os modelos e algoritmos tentam extrair padrões que possam surgir a partir desse conjunto (MURPHY, 2012). Além desses dois principais subgrupos de algoritmos, atualmente também temos outras classificações de algoritmos e abordagens, como aprendizado por reforço. A escolha de quais modelos uti-

lizar depende do problema a ser resolvido, da quantidade de dados disponíveis e da forma como esses dados estão estruturados e rotulados (MAHESH, 2020).

Um ponto importante a destacar sobre aprendizado de máquina na análise textual diz respeito a desempenho. Para melhorar o desempenho de um modelo de aprendizado de máquina, alguns tratamentos devem ser aplicados nos dados de treinamento. Um desses tratamentos é a extração ou engenharia de características. A engenharia de características é um processo de transformação dos dados para gerar representações numéricas que podem ser melhor interpretadas pelo modelo de aprendizado de máquina (DONG; LIU, 2018). Geralmente esse processo é realizado pelo cientista de dados com maior domínio dos dados que estão sendo analisados, para gerar as características que sejam representações mais fáceis de aprender pelo modelo utilizado no treinamento (NARGESIAN et al., 2017).

2.3 Processamento de Linguagem Natural

Processamento de Linguagem Natural é um campo da computação e linguística que possui como objeto de enfoque a automatização da análise e representação da linguagem humana. Esses recursos para processamento da linguagem podem ser construídos como objetivo final ou como um meio para alcançar outro produto (CAMBRIA; WHITE, 2014). O desenvolvimento desses métodos surge de pesquisas relacionadas a diversas áreas, como, por exemplo, Linguística Computacional, Aprendizado de Máquina, Inteligência Artificial, Recuperação da Informação e Mineração de Dados (EISENSTEIN, 2018). O processamento no contexto da PLN, entende-se como trazer as máquinas certas capacidades próprias do ser humano em relação à linguagem, sendo elas compreender, gerar, extrair conhecimento e outros (CASELI; FREITAS; VIOLA, 2022).

De forma geral, um sistema de PLN opera em níveis e busca fazer algumas tarefas menores utilizadas na resolução de um problema maior. Essas análises básicas geralmente são: Análise Fonética, Análise Morfológica, Análise Sintática, Análise Semântica e Análise Pragmática (PINTO, 2015). As análises são realizadas para compreender aspectos específicos da língua, como os sons (fonética), a classificação das palavras de maneira isolada (morfológica), a relação entre as palavras em uma sentença (sintática), o significado das palavras (semântica) e a relação entre a linguagem e o contexto (pragmática)

(PINTO, 2015).

Uma das grandes dificuldades no desenvolvimento de aplicações de PLN é a utilização de dados que não estão em formato estruturado, o que leva a limitações na utilização de algoritmos de aprendizado de máquina, pois esses algoritmos normalmente necessitam que os dados estejam representados de uma maneira estruturada (SOARES; PRATI; MONARD, 2008). A transformação de textos não estruturados em dados estruturados requer uma etapa de preparação dos dados denominado pré-processamento do texto (SOARES; PRATI; MONARD, 2008). Essa etapa geralmente tem um alto custo computacional e requer um cuidadoso planejamento e execução para se obter bons resultados na utilização dos algoritmos (MARTINS; MONARD; MATSUBARA, 2003). Existem algumas técnicas utilizadas no pré-processamento, entre elas, tem-se (SOARES; PRATI; MONARD, 2008):

- Tokenização: quebra do texto em tokens.
- *Stemming*: redução dos termos ao seu radical.
- Taxonomias: Classificação de termos similares entre si.
- Remoção de *Stopwords*: remoção de termos comuns que não são significativos para o algoritmo.

Uma das formas de transformar textos não estruturados em dados estruturados é utilizando a abordagem *bag of words*. Nessa abordagem, a frequência dos termos são contadas e a partir dessa contagem é gerada uma tabela com informações relacionadas a frequência de cada palavra (SOARES; PRATI; MONARD, 2008). No uso de uma tabela do tipo *bag of words*, existem alguns detalhes a serem considerados. A utilização dessa estrutura considera geralmente a relação entre número de documentos e número de atributos que descrevem cada documento, o que pode levar a uma alta dimensionalidade, que seria um número alto de atributos para um número pequeno de documentos, o que leva cada documento a ter poucos atributos o descrevendo, gerando uma tabela muito esparsa que pode tornar o processamento ineficiente (SOARES; PRATI; MONARD, 2008). Outra técnica utilizada para representação de dados não estruturados é o *word embeddings*. *Word embeddings* é uma forma de mapear palavras para vetores de valores reais, permitindo uma

representação matemática das palavras (TURIAN; RATINOV; BENGIO, 2010). Sem a etapa de transformação dos dados não estruturados em uma representação viável de processamento pelo algoritmo, não é possível prosseguir e conseguir bons resultados com as técnicas de aprendizado de máquina e PLN, o que torna esta etapa fundamental ao fluxo de processamento de linguagem natural (MARTINS; MONARD; MATSUBARA, 2003).

2.4 Tecnologias

Essa seção apresenta as tecnologias utilizadas durante o desenvolvimento do *software*. São apresentadas as tecnologias utilizadas no desenvolvimento da interface, *React* e *Material UI*, assim como o *Google Apps Script*, tecnologia que permite integração com Google Docs. Também são apresentadas a biblioteca NLTK e o *LanguageTool*, ambas utilizadas no desenvolvimento do módulo de análise de textos da aplicação. Por fim, é apresentado o modelo C4, utilizado para descrever a arquitetura do *software*.

2.4.1 *React*

O *React* é uma biblioteca JavaScript usada para criar interfaces de usuário (UI). O *React* permite construir a interface a partir da composição de elementos menores, como botões, textos e imagens, permitindo a reutilização e organização desses componentes em diferentes níveis. Desde sites até aplicativos móveis, qualquer elemento visual pode ser fragmentado em componentes (Meta, Inc., 2024). A sintaxe utilizada para construção desses componentes é o JSX, que é uma extensão de sintaxe para JavaScript que permite escrever marcação semelhante a HTML em um arquivo JavaScript. Na Figura 2.1, é mostrado um exemplo de componente construído com a sintaxe JSX.

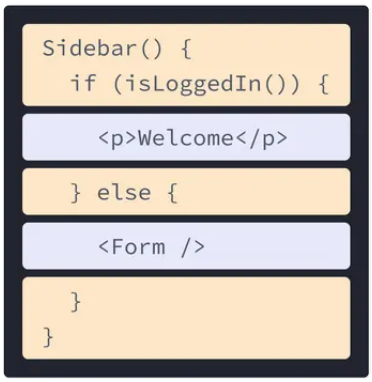

Figura 2.1: Exemplo de componente construído com JSX (Meta, Inc., 2024)

2.4.2 *Material UI*

O *Material UI* é uma biblioteca de design que segue os princípios do Material Design, desenvolvido pelo Google. Ela fornece um conjunto de componentes pré-projetados, estilos e ícones, permitindo um desenvolvimento mais rápido e padronizado de interfaces (MUI, 2024). A partir da utilização dessa biblioteca junto ao *React*, é possível utilizar os componentes visuais extendendo ou adicionando funcionalidades através da sintaxe JSX, explicada na Seção 2.4.1.

2.4.3 *Google Apps Script*

O *Google Apps Script* é uma plataforma JavaScript desenvolvida pelo Google que permite a integração e automação de tarefas nos produtos do Google, como Google Planilhas, Google Docs, Gmail, e outros serviços (Google, Inc., 2024). Utilizar o *Google Apps Script* traz algumas vantagens por ser uma plataforma com suporte da Google, como a segurança e o ambiente de desenvolvimento web que remove a necessidade de transferência de arquivos, backups, atualização do ambiente e outros aspectos do desenvolvimento de aplicações (FERREIRA, 2014).

2.4.4 NLTK

O NLTK é uma biblioteca Python utilizada para tarefas de processamento de linguagem natural. Ele oferece recursos para processamento de linguagem natural fáceis de utilizar,

com mais de 50 *corpora* e recursos lexicais, que podem incluir textos de jornais, livros e tweets, junto a um conjunto de recursos para as seguintes tarefas:

- Classificação: classificação pode ser definida como a tarefa de selecionar uma classe para uma determinada entrada (BIRD; KLEIN; LOPER, 2009).
- Tokenização: tokenização é uma tarefa de processamento de texto, no qual se divide o texto em palavras e pontuações. (BIRD; KLEIN; LOPER, 2009).
- *stemming*: *stemming* é um processo no qual as palavras são reduzidas a suas raízes, eliminando sufixos e prefixos, sendo um processo muito utilizado em processamento de linguagem natural para normalizar palavras e agrupar diferentes variantes de uma mesma palavra (BIRD; KLEIN; LOPER, 2009).
- marcação: marcação, também chamado POS-tagging ou simplesmente tagging, é processo de classificar palavras de acordo com suas partes do discurso e rotulá-las de maneira adequada. Os rótulos dados as palavras se referem a sua função gramatical, como verbo, substantivo, adjetivo, entre outros (BIRD; KLEIN; LOPER, 2009).
- *parsing*: *parsing* é o processo de analisar frases de entrada conforme as produções de uma gramática, construindo estruturas que estão conforme essa gramática. Esse processo permite que a gramática seja avaliada contra uma coleção de frases de teste, ajudando linguistas a identificar erros em suas análises gramaticais (BIRD; KLEIN; LOPER, 2009).

Essas tarefas são recorrentes no desenvolvimento de aplicações de Processamento de Linguagem Natural (PLN). Portanto, a disponibilidade de recursos que facilitem essas atividades proporciona maior agilidade e eficiência no processo de criação dessas aplicações.

2.4.5 *LanguageTool*

LanguageTool é uma ferramenta de código aberto dedicada à correção gramatical e à sugestão de avaliação. Tem suporte a várias línguas, entre elas, o português, e oferece correções e detalhamento para um conjunto de problemas gramaticais (Language-

Tool Community, 2024). O *LanguageTool* oferece uma API para integração com outras aplicações, que possui restrições para uso, que podem mudar de acordo com plano de contratação. A versão pública desse serviço possui uma restrição de 20 solicitações por minuto por IP e um limite de 20KB por requisição. Além disso, há várias bibliotecas que facilitam o uso desse serviço em diversas linguagens, como Python e Rust (LANGUAGE-TOOL, 2024).

2.4.6 Modelo C4

O modelo C4 é um conjunto de abstrações e diagramas hierárquicos que segue uma estrutura de camadas. Esse modelo pode ser usado para descrever a arquitetura de um *software* em diferentes níveis de abstração (BROWN, 2024). O modelo C4 propõe quatro níveis de abstração que são: contexto, *containers*, componentes e código. Cada nível de abstração apresenta uma perspectiva do *software*. O contexto foca em apresentar os atores e sistemas externos envolvidos no *software* desenvolvido, assim como suas interações. Os *containers* decompõem o *software* em componentes e descrevem suas funções, ocultando detalhes da estrutura interna desses componentes. No nível de componentes, a estrutura interna dos *containers*, descritos na camada anterior, são explicitados em maiores detalhes, descrevendo tecnologias, protocolos e responsabilidades de cada módulo e serviço interno. E na última camada, o detalhamento dos componentes chega ao nível de código, podendo ser utilizados diagramas UML (VÁZQUEZ-INGELMO; GARCÍA-HOLGADO; GARCÍA-PEÑALVO, 2020).

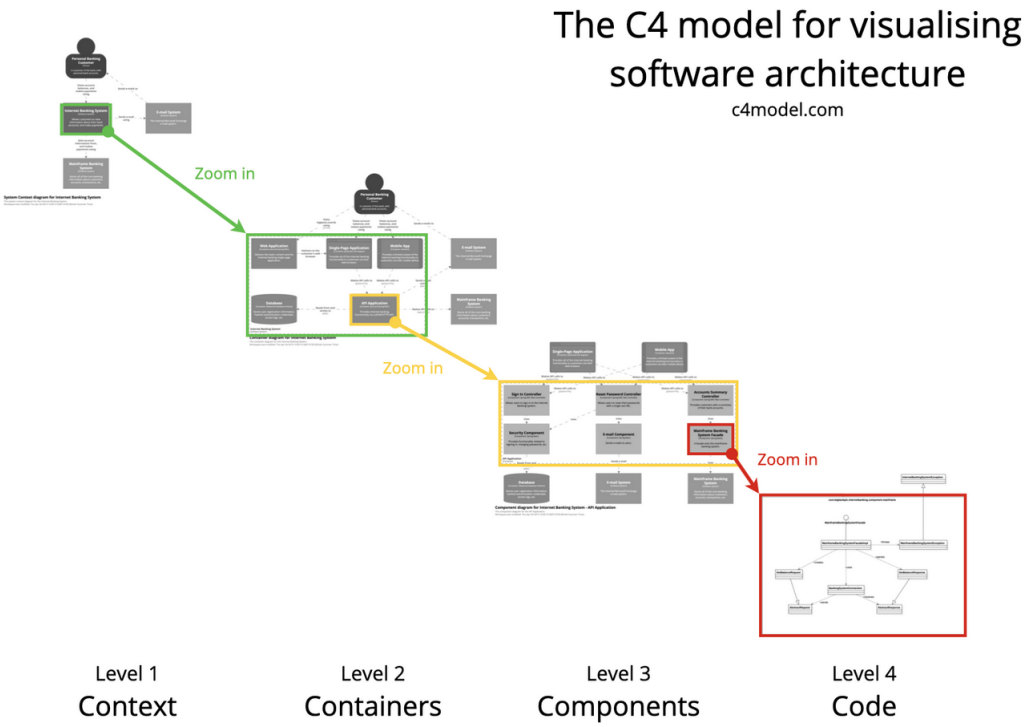

Figura 2.2: Modelo C4 para visualização de arquitetura de *software* (BROWN, 2024)

2.5 Considerações finais

No capítulo de Fundamentação Teórica foi apresentado alguns dos conceitos e pontos principais para compreensão da proposta deste trabalho.

Dentro dos pontos abordados foram apresentadas algumas definições do que é avaliação e correção e alguns dos desafios no cenário de produção textual e como o avanço nas áreas de aprendizado de máquina e processamento de linguagem natural permitiram o surgimento de ferramentas como a avaliação automática de redação que é um dos principais subprodutos dessas áreas com foco na educação. Também foi feita uma breve apresentação sobre aprendizado de máquina e processamento de linguagem Natural e alguns dos principais tópicos e abordagens nessas áreas. Por fim, foram apresentados definições das tecnologias utilizadas para o desenvolvimento de interfaces, aplicações de processamento de linguagem natural, integração com serviços do Google e descrição de arquitetura de *software*.

Em resumo, pode-se perceber que essas áreas se relacionam em alguns aspectos, e à medida que ocorram avanços, espera-se que cada vez mais contribuições e soluções

tecnológicas possam trazer melhorias no processo de produção de textos por parte alunos de diferentes áreas.

3 Trabalhos Relacionados

Neste capítulo são apresentados trabalhos relacionados a presente pesquisa. Da seção 3.1 até a seção 3.5 são apresentados 5 trabalhos, com seus objetivos, resultados obtidos, pontos positivos e negativos. Na seção 3.6 são apresentadas as considerações finais do capítulo, assim como uma tabela comparando os pontos relevantes de cada trabalho.

Para encontrar artigos com maior similaridade e relevância para a proposta do trabalho, foram elaboradas *strings* de busca utilizadas nas bases de dados Google Scholar e IEEE.

As *strings* utilizadas consistem na combinação de palavras-chave associadas ao trabalho proposto. Sendo assim, foram utilizados os seguintes termos nas buscas: *((natural language processing AND (automatic grading OR essay scoring)) AND (essay OR text))* em inglês e, em português: ((processamento de linguagem natural AND (avaliação automática de redação OR avaliação automática de texto)) AND (redação OR texto)) nas bases de dados Google Scholar e IEEE, ordenando por artigos mais atuais e se obteve 3000 resultados no scholar e 106 no IEEE.

Após a busca, foi realizado uma leitura inicial do resumo dos artigos selecionados para filtrar os trabalhos com maior similaridade a proposta deste trabalho.

Com base nesta busca, foram selecionados os seguintes trabalhos relacionados:

3.1 *Automated Essay Scoring: An approach based on ENEM competencies*

No artigo "*Automated Essay Scoring: An approach based on ENEM competencies*"(MARINHO et al., 2022), é proposto uma análise de algumas estratégias para avaliação automática de textos, seguindo os critérios estabelecidos para as cinco competências na matriz de referência da redação do Exame Nacional do Ensino Médio (ENEM).

As competências do ENEM são definidas da seguinte forma, de acordo com BRA-

SIL e INEP (2018):

- **Competência 1:** Demonstrar domínio da modalidade escrita formal da língua portuguesa.
- **Competência 2:** Compreender a proposta de redação e aplicar conceito das várias áreas de conhecimento para desenvolver o tema, dentro dos limites estruturais do texto dissertativo-argumentativo em prosa.
- **Competência 3:** Selecionar, relacionar, organizar e interpretar informações, fatos, opiniões e argumentos em defesa de um ponto de vista.
- **Competência 4:** Demonstrar conhecimento dos mecanismos linguísticos necessários para a construção da argumentação.
- **Competência 5:** Elaborar proposta de intervenção para o problema abordado, respeitando os direitos humanos.

Para a implementação das estratégias selecionadas foram utilizados principalmente técnicas de PLN e algoritmos de aprendizado de máquina supervisionado. Foram investigados as seguintes abordagens: Engenharia de *features*, *embeddings* e redes neurais recorrentes do tipo *Long Short-Term Memory*.

O estudo realiza a implementação de cinco modelos independentes aplicados a cada uma das competências da redação do ENEM. Esses modelos foram aplicados no corpus *Essay-Br*, que é um corpus formado a partir da extração automatizada de redações dos sites Universo Online (UOL) e Brasil Escola, e possui um total de 4570 redações com 86 temas produzidos entre 2015 e 2020 (MARINHO et al., 2022). Foram utilizados três métodos na proposta desses autores, baseados em engenharia de características, *embeddings* e redes neurais recorrentes.

Para a implementação do método baseado em engenharia de características, foram definidos conjuntos de características específicas para cada competência da matriz de referência do ENEM, a fim de buscar extrair aspectos léxicos, sintáticos e semânticos.

O método baseado em *embeddings* foi feito com uso de modelos Doc2Vec treinados sobre o corpus de redação. Foram treinados regressores independentes para cada

competência. Aplicou-se um pré-processamento no corpus de treinamento, após isso, foram treinados dois modelos de *embeddings*. Os modelos de *embeddings* resultantes do treinamento foram utilizados como entrada de treinamento para algoritmos tradicionais de regressão para predizer a nota de cada competência.

Para o método baseado em redes neurais recorrentes, foi formado uma estrutura de RNN com uma camada de *embeddings*, que recebe como entrada as redações já pré-processadas e gera uma matriz formada por vetores de *embeddings*.

Para avaliação dos modelos, foi utilizado o Kappa Quadrático Ponderado. Em virtude desse método considerar apenas valores discretos, as notas preditas pelos modelos foram discretizadas. Para a competência 1, o modelo de *features* obteve o melhor resultado, possivelmente pelo fato de conseguir capturar melhor os erros ortográficos e gramaticais através das features definidas. Para todas as outras competências, o modelo de redes obteve os melhores resultados.

3.2 Avaliação Automática baseada na Coleta de Atributos

Neste trabalho, é proposta uma avaliação automática de redação baseado na coleta de atributos (NETO et al., 2020). Foram considerados atributos de 4 dimensões, sendo esses: léxico, sintático, conteúdo e coerência.

O desenvolvimento seguiu uma arquitetura de 5 etapas, sendo essas: preparação do Corpus, pré-processamento, coleta de atributos, modelo de predição e avaliação. O corpus para aplicação e avaliação foi composto por 1000 redações de um concurso público da Universidade Federal do Oeste do Pará. Estas redações passaram por um processo de digitalização manual, sem nenhum tipo de correção e nem alterações no texto. Todas as redações foram previamente avaliadas por dois avaliadores humanos, recebendo uma pontuação entre 0 e 10. Caso duas pontuações divergissem por mais de um ponto, então um terceiro avaliador atribuia uma pontuação para resolver a discrepância.

O pré-processamento foi realizado com remoção de *stopwords*, remoção de sufixos

e tokenização, utilizando a biblioteca *Natural Language Toolkit* (NLTK)[1]. Na etapa de coleta de atributos, foram extraídos mais de 140 atributos. Para a tarefa de predição, utilizou-se um algoritmo de aprendizado de máquina supervisionado *Random Forest*, recebendo como entrada os atributos extraídos. Para medição da acurácia, foi utilizado o Kappa Quadrático.

Após experimentos no corpus de 1000 redações foi analisada a contribuição de cada um dos grupos de atributos na predição das notas. Foi identificado que os atributos da categoria 'conteúdo' obtiveram melhor acurácia e contribuíram mais na predição das notas. Também foi avaliado o impacto da combinação de atributos de diferentes categorias, em que os melhores resultados foram obtidos da combinação de atributos das categorias de conteúdo e coerência.

3.3 *Automatically Grading Brazilian Student Essays*

Fonseca et al. (2018) propôs neste estudo o desenvolvimento de um sistema de avaliação automática de textos para textos argumentativos nos moldes do ENEM de estudantes brasileiros. Foram investigadas duas abordagens: Redes Neurais Profundas e sistemas baseados em engenharia de características. Para avaliação do sistema, foi selecionado um dataset de 56644 textos retirados de uma plataforma online, onde os textos já haviam sido avaliados por professores humanos.

Para a abordagem com redes neurais profundas foi usada uma arquitetura de redes neurais com duas camadas. A primeira camada lê os vetores de palavras e gera vetores de sentenças, os quais são lidos pela segunda camada para gerar um único vetor de texto.

A abordagem baseada em engenharia de características foi desenvolvida usando um pré-processamento, tokenizando os textos e usando um *POS-tagger* para etiquetar os tokens de acordo com sua classificação morfológica. As características foram extraídas e categorizadas em métricas como contagem genérica, presença de expressões específicas, tokens e outros.

Para medição da acurácia foi utilizado o Kappa Quadrático Ponderado. Nesse

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

estudo, nas competências 1 a 4, o modelo de engenharia de características obteve os melhores resultados, enquanto o modelo de redes neurais obteve o melhor resultado para a competência 5. Nesse cenário, o modelo utilizando características pode ter se saído melhor pela escolha das características.

Outro ponto interessante é que os modelos performaram melhor para predição da nota agregada, em comparação a predição para cada competência separadamente.

3.4 Sistema de avaliação automática de redações do Enem

Júnior, Spalenza e Oliveira (2017) apresenta no artigo "Proposta de um sistema de avaliação automática de redações do Enem utilizando técnicas de aprendizagem de máquina e processamento de linguagem natural" a construção de um sistema de avaliação automática de redação.

Durante o desenvolvimento do trabalho, ocorreu primeiramente a etapa de pré-processamento do texto, removendo termos que não trazem significado ao texto, dessa forma diminuindo a complexidade da classificação. São removidos números, datas, símbolos e outros termos. A etapa seguinte foi a extração de características, onde cada redação passou a ser representada por um vetor com algumas características como quantidade de parágrafos, frases, palavras e erros. Os erros ortográficos são identificados utilizando o *Hunspell*[2], um verificador ortográfico, além de um algoritmo usando probabilidade *bayesiana* para analisar o contexto da palavra, para identificar palavras que estão incorretas em um contexto específico. Para identificação de erros gramaticais foi utilizado o CoGrOO, um *software* de correção gramatical (KINOSHITA; SALVADOR; MENEZES, 2006). O classificador selecionado foi o *Support Vector Machine*, devido sua boa capacidade de generalização, robustez e por possuir uma teoria bem definida (JÚNIOR; SPALENZA; OLIVEIRA, 2017).

As métricas definidas para avaliação do método de classificação foram *precision*, *recall* e erro médio absoluto. *Precision* mede a proporção de amostras classificadas como

[2]https://hunspell.github.io/

positivas que são realmente positivas. *Recall* mede a proporção de amostras positivas classificadas como positivas. Erro médio absoluto é a média das diferenças entre os valores reais e os preditos. O conjunto de dados utilizado possui 4547 redações do *site* UOL.

Considerando a apenas a competência 1 dos moldes do ENEM, o sistema conseguiu uma distância de 0,26 da nota do avaliador humano. Com esse resultado, e o resultado de 93% de *precision* e 52% de *recall*, o sistema indicou possibilidades de aplicação prática no dia-a-dia do avaliador humano. Porém, o sistema demonstra uma limitação que seria a aplicação em apenas um aspecto dos textos avaliados.

3.5 Identificação de Deficiências em Textos Educacionais

No artigo "Identificação de Deficiências em Textos Educacionais com a Aplicação de Processamento de Linguagem Natural e Aprendizado de Máquina"(PINHO et al., 2022), é realizado um processo de extração de dados numa base de redações, a fim de aplicar aprendizado de máquina e identificar deficiências nos textos com menor avaliação. A proposta do experimento seria utilizar algoritmos de aprendizado de máquina para mensurar a medida de similaridade entre as redações analisadas (PINHO et al., 2022).

Para o pré-processamento foi realizado a remoção de *stopwords*, *tokenização* do texto e *stemming* para normalização. A seguir foi empregado uma matriz TF IDF para medir o grau de importância de um termo em relação a um conjunto de documentos (PINHO et al., 2022) e com base nessas matrizes, foram gerados agrupamentos utilizando o algoritmo *Kmeans*. Após a clusterização, foram gerados mapas e gráficos para identificar padrões nos textos.

Para avaliação, foi utilizado uma base de redações com 695 textos já avaliados previamente. A aplicação das técnicas de PLN e aprendizado de máquina permitiram encontrar padrões de erros relacionados a ortografia e gramática, além de relações de similaridade entre redação e tema. Além disso, foi possível identificar uma relação entre a quantidade de sentenças e nota dada pelo avaliador, indicando assim uma possível métrica de avaliação. Foi demonstrado que as técnicas aplicadas durante o desenvolvimento podem

colaborar para minimizar o esforço de avaliadores, identificando de forma automatizada deficiências em textos dissertativos.

3.6 Considerações finais

Neste capítulo, foram descritos os trabalhos relacionados a esta pesquisa, com as técnicas utilizadas e resultados alcançados, visando identificar os métodos e processos mais comuns utilizados no desenvolvimento de sistemas de avaliação automática de textos. Podemos identificar bastante similaridade entre os 6 trabalhos apresentados, que em muitos momentos realizaram as mesmas etapas e utilizaram os mesmos métodos, como a preparação da base de dados, pré-processamento do texto e avaliação dos resultados de acordo uma métrica bem definida. É perceptível a importância da aplicação de tais etapas para o desenvolvimento de trabalhos no mesmo segmento.

Na Tabela 3.1, é possível comparar os pontos mais relevantes de semelhança e diferença dos trabalhos apresentados para este trabalho.

Tabela 3.1: Comparação entre os pontos dos trabalhos relacionados

	MARINHO et al., 2022	**NETO et al., 2020**	**FONSECA et al., 2018**	**Júnior; Spalenza; Oliveira, 2017**	**PINHO et al., 2022**
Objetivo	Avaliação automática das cinco competências do ENEM	Avaliação automática baseada em 4 dimensões de atributos	Avaliação automática de redações do ENEM	Avaliação automática de redação	Identificação de deficiências em textos
Técnicas e tecnologias	Features, embeddings, RNN, Doc2Vec	NLTK, Random Forest, remoção de stopwords, remoção de sufixos, tokenização	Redes Neurais Profundas e features, POS-tagger	Hunspell, CoGrOO, *Support Vector Machine*, *Apache OpenNLP*	Aprendizado de máquina, TF-IDF, K-means, tokenização, stemming, stopwords
Categorias avaliadas	Ortografia, gramática, semântica, coerência	Léxico, sintaxe, conteúdo e coerência	Avaliação de competências ENEM	Ortografia, gramática	Ortografia, gramática, similaridade de tema
Testes	Redações do ENEM	Redações de concurso público	Redações do ENEM	Redações do ENEM	Redações
Base de Dados	4570 redações do corpus Essay-Br	1000 redações de concurso público da UFOPA	56.644 redações avaliadas por professores humanos	4547 redações do site UOL	695 redações avaliadas
Métricas de Avaliação	Kappa Quadrático Ponderado	Kappa Quadrático	Kappa Quadrático Ponderado	Precisão, *recall*	Análise de gráficos

Os autores Pinho et al. (2022), Júnior, Spalenza e Oliveira (2017), Fonseca et al. (2018) e Marinho et al. (2022), trabalharam com as categorias ortográfia e gramática, que este trabalho também busca avaliar em seus objetivos. Além disso, os autores Marinho et al. (2022), Júnior, Spalenza e Oliveira (2017) e Fonseca et al. (2018) avaliam textos nos

moldes do exame nacional do ensino médio, utilizando como base de dados para testes o banco de redações do ENEM, que também é utilizado na avaliação desse trabalho.

Com relação ao objetivo, esse trabalho visa o desenvolvimento de um *software*, integrado ao Google Docs, para avaliação semiautomática de textos em português, especialmente monografias, abordando assim um tipo de texto diferente da maioria dos trabalhos relacionados, que tem seu foco principalmente em textos de estudantes do ensino médio. Apesar da diferença com relação ao tipo de texto abordado, a maioria do desenvolvimento deste trabalho utiliza algumas das etapas e métodos apresentados, visto que são parte fundamental do processo de análise do texto, independente do tipo de texto abordado.

4 Desenvolvimento

Neste capítulo, são apresentados detalhes sobre o processo de desenvolvimento do *software* como extensão para o conjunto de ferramentas do Google, abordando as tecnologias de desenvolvimento, arquitetura, estratégias de implementação utilizadas, além dos desafios enfrentados no desenvolvimento. O capítulo está dividido em seções que detalham a arquitetura do sistema, o módulo de análise textual, as interfaces desenvolvidas e as restrições da aplicação.

4.1 Arquitetura do Sistema

Para demonstrar e visualizar a arquitetura, processos e comunicações do *software*, foi utilizado o modelo C4 de documentação de arquiteturas. O fluxo mostrado na Figura 4.1 demonstra a arquitetura de *containers* do *software*, segundo o modelo C4 de documentação. Esse diagrama ilustra os elementos que compõem o *software* de maneira simplificada, indicando as tecnologias utilizadas e as formas de comunicação.

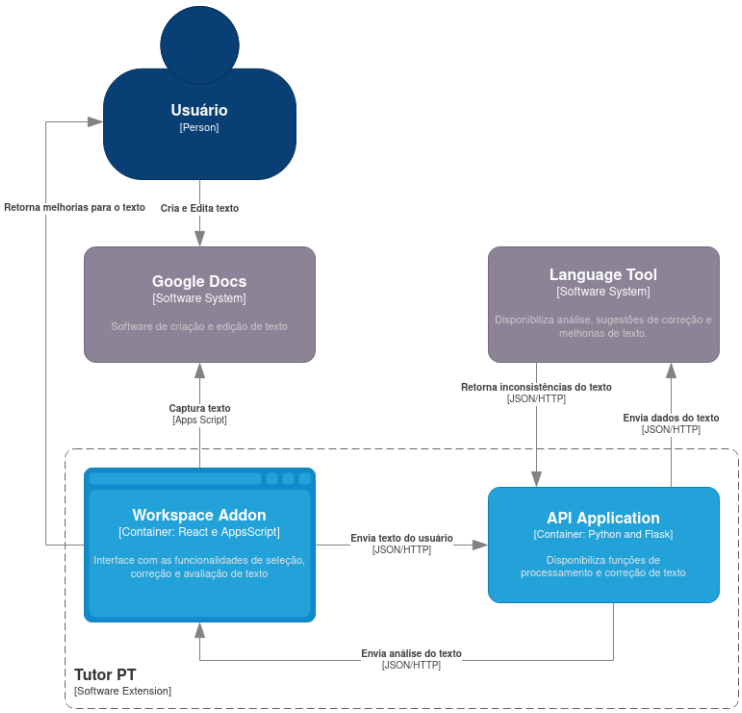

Figura 4.1: Diagrama de *containers*

A arquitetura da extensão foi projetada visando implementar uma solução para a correção e avaliação de textos no Google Docs. Nas seções seguintes deste capítulo, entraremos em detalhes sobre as tecnologias utilizadas na implementação e as restrições encontradas durante o desenvolvimento.

4.1.1 Análise textual

Como detalhado na Figura 4.2, o módulo desenvolvido é responsável por disponibilizar funções de análise e avaliação de texto. Essa parte do *software* foi desenvolvida em Python usando o *framework* FLASK, visando facilitar o uso de bibliotecas para processamento de linguagem natural. A biblioteca NLTK[3] é empregado nas tarefas de análise e processamento de texto. Após o processamento inicial do texto, o *software* utiliza o serviço do *LanguageTool*[4] para correção gramatical e ortográfica. Essa parte do *software* foi desenvolvida como uma API, de forma desacoplada do restante do *software*, sendo assim possível reutilizá-la em outras soluções.

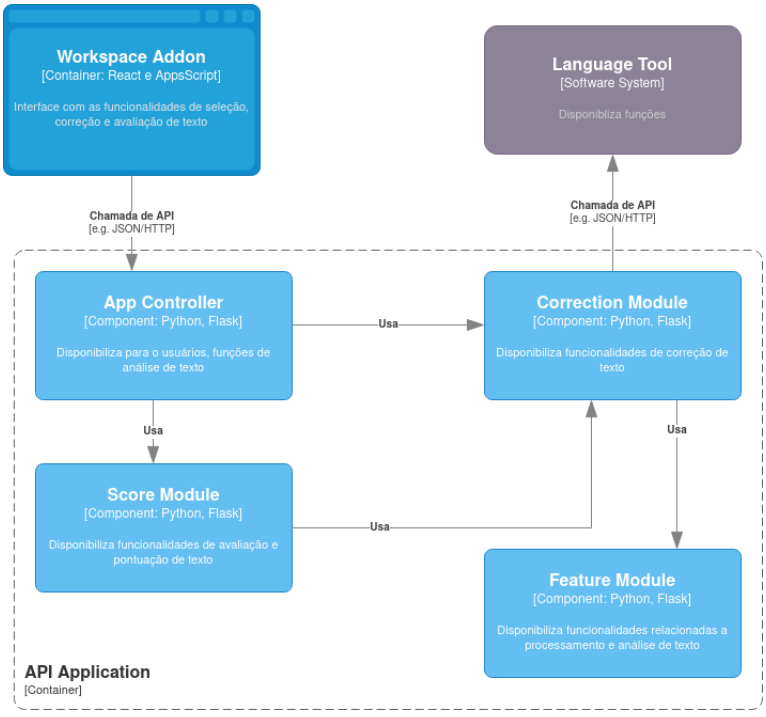

Figura 4.2: Diagrama de Componentes - API

O NLTK foi empregado para a análise e a segmentação do texto, permitindo a

[3]https://www.nltk.org/
[4]https://languagetool.org/pt-BR

manipulação para a extração de métricas textuais relevantes para a análise e avaliação do texto. Após a segmentação do texto em um conjunto de palavras e sentenças, esses segmentos são enviados ao *LanguageTool* que retorna os erros identificados nas sentenças, e sugestões de possíveis correções. Após obter os erros, a ferramenta realiza a contagem dos erros obtidos em cada uma das categorias, e decrementa do total de pontuação, o valor definido pelo usuário para cada erro. Após esse processamento, a pontuação e os erros, com as sugestões para cada erro, são retornados.

4.1.2 Interface

A interface foi desenvolvida para permitir que o usuário visualize as correções e aplique as sugestões junto a interface do Google Docs. Na Figura 4.3, é descrito como a interface foi desenvolvida, possuindo um módulo de interação com o Google Docs através do Google App Script, além de outros dois módulos que disponibilizam as telas de pontuação e sugestão.

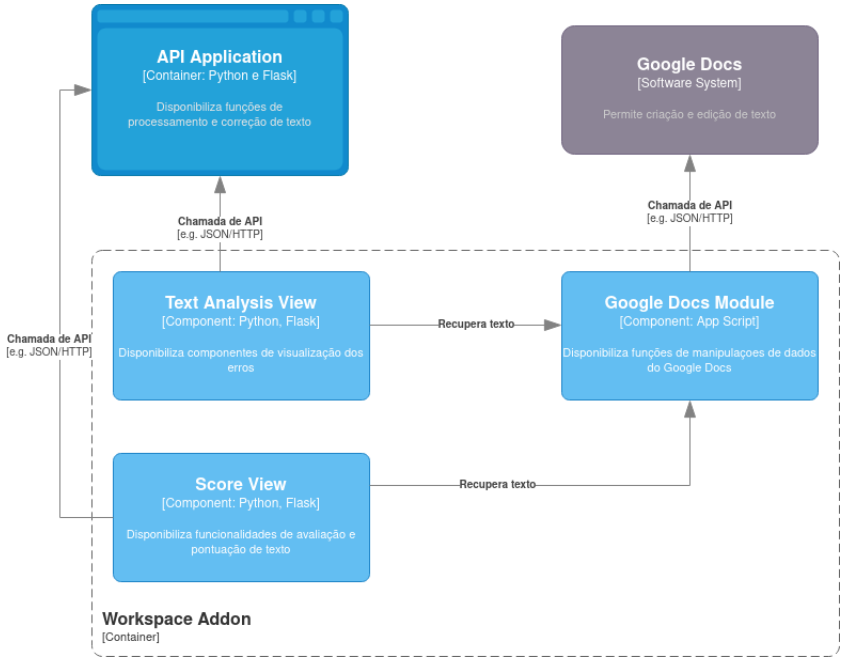

Figura 4.3: Diagrama de Componentes - Extensão

A interface inicial do *software* é apresentada na Figura 4.4. Todas as funcionalidades podem ser acessadas por meio do menu "Tutor-PT", localizado na aba superior do Google Docs. Na barra lateral, o usuário tem a opção de acionar o botão "Anali-

sar"durante a escrita do texto. Ao fazer isso, a extensão captura o conteúdo do documento e o envia para a API de análise descrita na Seção 4.1.1. Após o processamento, os erros identificados são exibidos nessa tela, organizados conforme as categorias explicadas na Seção 4.2.2. Quando o usuário seleciona um erro, ele é destacado em amarelo no texto, conforme mostrado na Figura 4.4, facilitando a localização exata do erro. Além disso, a pontuação atribuída a cada categoria é exibida, baseada na quantidade de erros detectados.

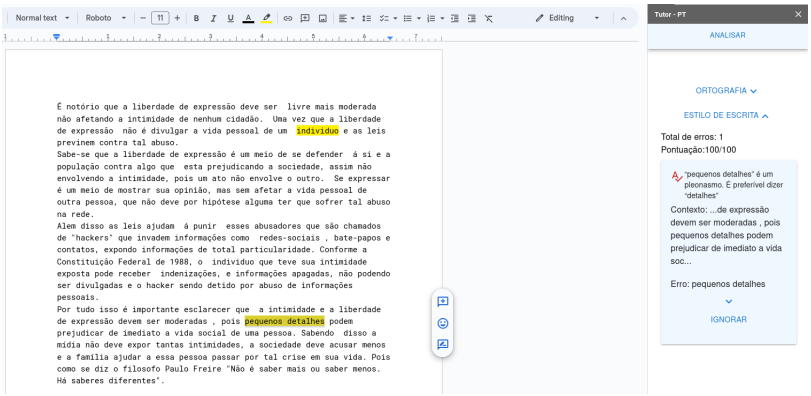

Figura 4.4: Tela de edição de documentos

A Figura 4.5 exibe a lista de sugestões geradas pelo *software*, contendo todas as opções de substituição para o erro identificado. Ao clicar em uma das alternativas, o usuário pode aplicar a sugestão escolhida, que automaticamente substituirá o erro no texto.

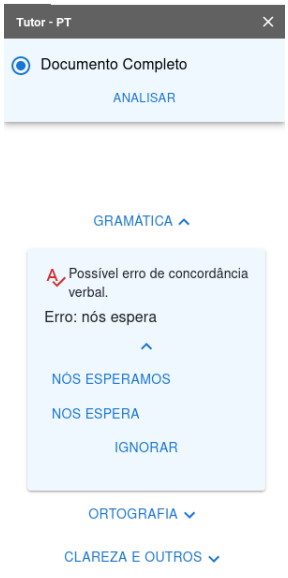

Figura 4.5: Tela de sugestões

Na tela de configuração de pontuação, ilustrada na Figura 4.6, o usuário pode ajustar tanto a pontuação total quanto a dedução de pontos para cada categoria, inserindo os valores desejados diretamente na tabela.

Figura 4.6: Tela de configuração de pontuação

Todas as telas foram desenvolvidas utilizando a biblioteca *React*[5], em conjunto com o *MaterialUI*[6]. A interface comunica-se com a API por meio de requisições HTTP, o que permite a renderização, no Google Docs, das inconsistências encontradas, das sugestões e da pontuação à medida que o usuário solicita uma nova análise do documento. Para garantir a padronização visual, foram empregados componentes da biblioteca *MaterialUI*, como cartões para exibir os erros identificados e listas para apresentar as sugestões de substituição.

4.1.3 *Google Apps Script*

Para permitir a integração da extensão com o Google Docs, foi utilizado o *Google Apps Script*. Através dessa plataforma, foi possível capturar o texto escrito pelo usuário, desta-

[5]https://react.dev/
[6]https://mui.com/

car no texto as palavras e sentenças com erros identificados e substituir, no texto, um erro pela opção selecionada pelo usuário. Todas essas interações foram realizadas utilizando o serviço direcionado a documentos [7] da plataforma *Google Apps Script*.

O serviço de documentos do *Google Apps Script* permite acessar e modificar arquivos do Google Docs, sendo necessário que o usuário conceda permissões específicas a aplicativos de terceiros por questões de segurança. Por isso, foi preciso configurar adequadamente o uso desse serviço, especificando quais recursos e permissões seriam solicitados pela aplicação, como a leitura e modificação dos documentos. Além disso, foi realizada a configuração da tela de autorização do usuário, que informa quais permissões a aplicação requer e permite ao usuário concedê-las de forma explícita. Dessa forma, no primeiro acesso à extensão, o usuário deve visualizar a tela de autorização e conceder as permissões solicitadas.

4.2 Restrições

Durante o desenvolvimento da extensão surgiram algumas restrições. Nesta seção, abordaremos os desafios encontrados durante o processo de desenvolvimento.

4.2.1 Integração da API do *LanguageTool*

Uma das restrições que surgiram foi o limite de uso da API do *LanguageTool*. Atualmente, a API do *LanguageTool* conta com alguns limites de requisições por token e por tempo que variam conforme o plano utilizado. Para este trabalho, utilizamos a versão gratuita da API que tem as seguintes limitações:

- Número Máximo de Requisições por Minuto: 20
- Número Máximo de Caracteres por Requisição: 20000
- Número Máximo de Caracteres por Minuto: 75000

Devido a essas restrições, foi necessário realizar um pré-processamento do texto enviado para análise do LanguageTool, dividindo o texto em segmentos de 20000 caracte-

[7] https://developers.google.com/apps-script/reference/document?hl=pt-br

res no máximo, de maneira que a análise se restringe a esses segmentos, podendo perder contextos que não estão dentro desse conjunto de texto. Além disso, pode ocorrer uma perda no desempenho da aplicação devido a esse pré-processamento que deve ocorrer antes do envio do texto para identificação dos erros.

4.2.2 Categorias de erros

Durante o processo de desenvolvimento do *software*, foi necessário realizar escolhas em relação à abordagem de identificação e correção dos erros. É importante ressaltar que nem todas as categorias de erros e sugestões gramaticais estão disponíveis para língua portuguesa ou foram incorporadas nas identificações do *software*. A decisão de focar em determinadas categorias específicas visa otimizar os resultados e sugestões do *software*, focando em aspectos mais recorrentes e impactantes na qualidade do texto. As categorias tratadas pela API do *LanguageTool* se baseiam em uma classificação descrita pela W3 para tecnologias de processamento de linguagem natural, apresentada na Tabela 4.1 (GROUP, 2024).

Não são todas as classificações da tabela utilizadas pela API ou para o processamento de texto em língua portuguesa. As categorias identificadas nos textos utilizados durante o desenvolvimento foram as seguintes:

- duplication
- inconsistency
- grammar
- locale-violation
- style
- misspelling
- whitespace
- uncategorized

Tabela 4.1: Tipos de erros e desvios de qualidade (GROUP, 2024)

Categoria	**Descrição**
terminology (terminologia)	Um termo incorreto ou de um domínio errado foi usado ou os termos são usados de forma inconsistente.
mistranslation (tradução incorreta)	O conteúdo traduzido não corresponde ao conteúdo original.
omission (omissão)	Texto necessário foi omitido da localização ou do original.
untranslated (não traduzido)	Conteúdo que deveria ser traduzido foi deixado sem tradução.
addition (adição)	O texto traduzido contém adições inadequadas.
duplication (duplicação)	Conteúdo foi duplicado de forma inadequada.
inconsistency (inconsistência)	O texto é inconsistente com ele mesmo ou foi traduzido de forma inconsistente.
grammar (gramática)	O texto contém um erro gramatical.
legal (jurídico)	O texto é legalmente problemático
register (registro linguístico)	O texto está escrito no registro linguístico incorreto ou utiliza gírias ou linguagem inadequada.
locale-specific-content (conteúdo específico do local)	A localização contém conteúdo que não se aplica ao local para o qual foi preparado.
locale-violation (violação de localidade)	O texto viola normas da localidade pretendida.
style (estilo)	O texto contém erros estilísticos.
characters (caracteres)	O texto contém caracteres corrompidos ou incorretos.
misspelling (erro ortográfico)	O texto contém um erro ortográfico.
typographical (erro tipográfico)	O texto contém erros tipográficos, como pontuação ou capitalização incorretas/omitidas.
formatting (formatação)	O texto está formatado incorretamente.
inconsistent-entities (entidades inconsistentes)	O texto original e o traduzido contêm entidades nomeadas diferentes.
numbers (números)	Os números são inconsistentes entre o original e a tradução.
markup (marcação)	Há um problema relacionado à marcação ou uma discrepância entre a marcação do original e da tradução.
pattern-problem (problema de padrão)	O texto não corresponde a um padrão permitido ou corresponde a um padrão que define conteúdo não permitido.
whitespace (espaçamento)	Há uma discrepância no espaçamento entre o original e a tradução.
internationalization (internacionalização)	Há um problema relacionado à internacionalização do conteúdo.
length (comprimento)	Há uma diferença significativa no comprimento entre o original e a tradução.
non-conformance (não conformidade)	O conteúdo apresenta baixa conformidade estatística com um corpus de referência.
uncategorized (não categorizado)	O problema não foi categorizado ou não pode ser categorizado.
other (outro)	Qualquer problema que não possa ser atribuído a nenhum dos valores listados acima.

Durante o desenvolvimento, foram realizados testes com textos de redações no modelo do ENEM, extraídos da base de dados do UOL (UOL, 2024), selecionados arbitrariamente, apenas para identificar quais categorias seriam identificadas com mais frequência. Como resultado, as categorias *duplication*, *grammar*, *misspelling*, *style*, *uncategorized* e *typographical* aparecerem pelo menos uma vez nos textos analisados e foram incluídos nas sugestões do *software*. As categorias *whitespace* e *locale-violation* também foram encontrados nos testes, porém, por apresentarem principalmente pontos que não interferem na integridade do texto, como, por exemplo, excesso de espaços e uso de palavras estrangeiras, essas categorias foram desconsideradas na inclusão das sugestões. A categoria *inconsistency* não foi indicada em nenhum texto, não sendo considerada pelo *software* para apresentar sugestões dessa categoria. Como citado anteriormente, não foi

possível identificar todas as categorias da Tabela 4.1 e também não foi possível encontrar na documentação, quais dessas categorias são utilizados para língua portuguesa. Sendo assim, foram selecionadas as categorias de duplicação, ortografia, gramática, estilo e não categorizados.

Duplicação

Erros de duplicação em texto se refere a repetição de palavras, frases ou segmentos, em uma composição no texto. Essa duplicação pode se manifestar de diversas formas, porém no *software* desenvolvido, consideramos apenas a repetição direta de uma palavra ou expressão.

Exemplo: O processo está está chegando ao fim.

Ortografia

Erros ortográficos são desvios e imprecisões na grafia de palavras do idioma estabelecido. Na língua portuguesa, esses desvios podem envolver o uso incorreto de letras, acentuações, hífen e outros elementos.

Exemplo: Ele quiz aprender a tocar violão.

Gramática

Erros gramaticais são desvios e imprecisões na estrutura gramatical do texto. Essas imprecisões podem abranger diversos aspectos, como concordância de verbos e nomes, uso de pronomes, regência, entre outros.

Exemplo: Pedro estudou, mais não passou no concurso.

Estilo

Na categoria 'estilo', a aplicação identifica e sugere recomendações relacionadas à expressão e organização do texto. Essas sugestões visam aprimorar alguns aspectos como coerência, variedade de vocábulo e formalidade. Como exemplo, na frase abaixo, o início da frase não é identificado como um erro, porém é considerada pelo sistema como uma expressão prolixa, podendo ser substituída por alguma alternativa.

Exemplo: Nos dias atuais, a fragilidade educacional tornou-se alvo de várias discussões, pelo fato de vários países, considerados subdesenvolvidos, apresentarem índices mínimos de produtividade e prosperidade econômica.

Clareza

Assim como na categoria de estilo, na categoria 'clareza', o *software* identifica e sugere recomendações, porém direcionadas a clareza do texto. Essas sugestões visam principalmente a organização e ambiguidade. Nessa classificação também estão os erros identificados na categoria *typographical*, que definem o uso adequado de recursos como pontuações.

Exemplo: Portanto observa-se que tal mudança seria vantajosa para o consumidor.

4.3 Considerações finais

Este capítulo teve como objetivo explicar o processo de desenvolvimento da extensão TUTOR-PT, assim como as tecnologias utilizadas, os desafios encontrados e as limitações do *software*, passando pelo desenvolvimento da interface da extensão, integração da interface com o Google Docs e desenvolvimento da API utilizada para pré-processamento e identificação dos erros e sugestões, junto com o *LanguageTool*.

O desenvolvimento da interface junto ao Google Docs é fundamental para viabilizar o uso prático da aplicação durante a escrita de um texto. Assim como a API para pré-processamento e comunicação com o *LanguageTool* é parte importante da aplicação para disponibilizar a identificação mais eficiente dos erros.

5 Discussão dos Resultados

Este capítulo pretende apresentar resultados obtidos durante este trabalho. A avaliação dos resultados será realizado por meio de uma análise do desempenho e da acurácia do *software* no reconhecimento de palavras e sentenças corretas e incorretas.

Para coletar os dados, foram empregados testes em um conjunto de textos avaliados previamente por profissionais humanos selecionados da base de dados de redações do UOL (MARINHO et al., 2022). Esses textos já possuem os seguintes atributos anotados que foram utilizados nos testes: pontuação final, erros e sugestões de correção. A base de textos possui o total de 2164 redações e, desse total, foi selecionado um conjunto de 100 textos para avaliação da aplicação. A redução do conjunto de textos foi realizada visando permitir a realização dos testes respeitando as restrições descritas da versão gratuita da API do *LanguageTool* para número máximo de requisições e textos analisados. Essa metodologia de avaliação foi utilizada visando identificar por meio de testes, o resultado que o *software* obteve no seu objetivo de identificar erros e desvios linguísticos. Não foi possível aplicar os mesmos testes a um conjunto de textos de monografia devido à falta de um conjunto de textos de monografia já avaliados e anotados por avaliadores humanos. Sendo assim, esse conjunto de testes ficou restrito a textos de redação.

Para as análises foram consideradas as seguintes métricas:

- **Desempenho**: tempo total decorrido, do envio do texto até o retorno do processamento (SANTANA et al., 1994). Neste trabalho, foi medido o tempo de processamento de cada texto e obtido o tempo médio de processamento da aplicação.

- **Acurácia**: proximidade entre os resultados obtidos com os valores de referência (MONICO et al., 2009). Neste trabalho, os valores de referência são os erros anotados pelos avaliadores humanos no conjunto de textos usados nos testes. A partir dessa comparação também obtemos:

  - Erros reconhecidos corretamente: relação de erros encontrados pela aplicação corretamente.

– Erros reconhecidos incorretamente: falsos positivos, erros indicados pela aplicação não existentes nos valores de referência

– Erros não reconhecidos: falsos negativos, erros que o sistema falhou em identificar presentes nos valores de referência.

5.1 Análise de desempenho

Para avaliação do desempenho, foi medido o tempo total de processamento da aplicação sobre todo conjunto de 100 textos selecionados.

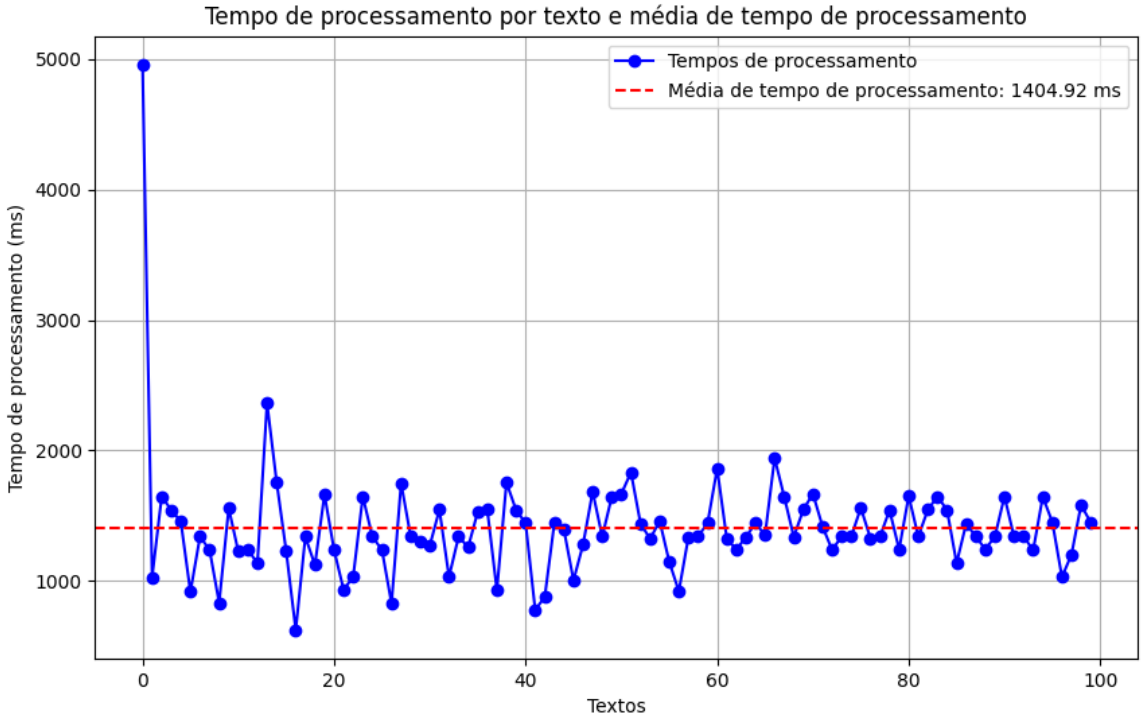

Figura 5.1: Tempo médio de processamento

O tempo médio de processamento do conjunto total de textos foi de 1404,92 milissegundos, demonstrando que o *software* é capaz de processar os textos em um tempo aceitável para utilização em aplicações *web* (NIELSEN, 1993). A quantidade média de palavras dos textos analisados foi de 282.37.

O tempo de processamento teve poucas variações, demonstrando não ter variações significativas de desempenho para textos com quantidades diferentes de erros.

5.2 Análise de acurácia

A capacidade do *software* de identificar corretamente erros no texto foi avaliada pela métrica de erros reconhecidos corretamente, que apresentou uma média de 2,32 erros por texto, representando 30,65% dos erros totais, indicando que, em média, aproximadamente um terço dos erros são detectados corretamente. A variação da taxa de reconhecimento para cada texto pode ser visualizado na Figura 5.2

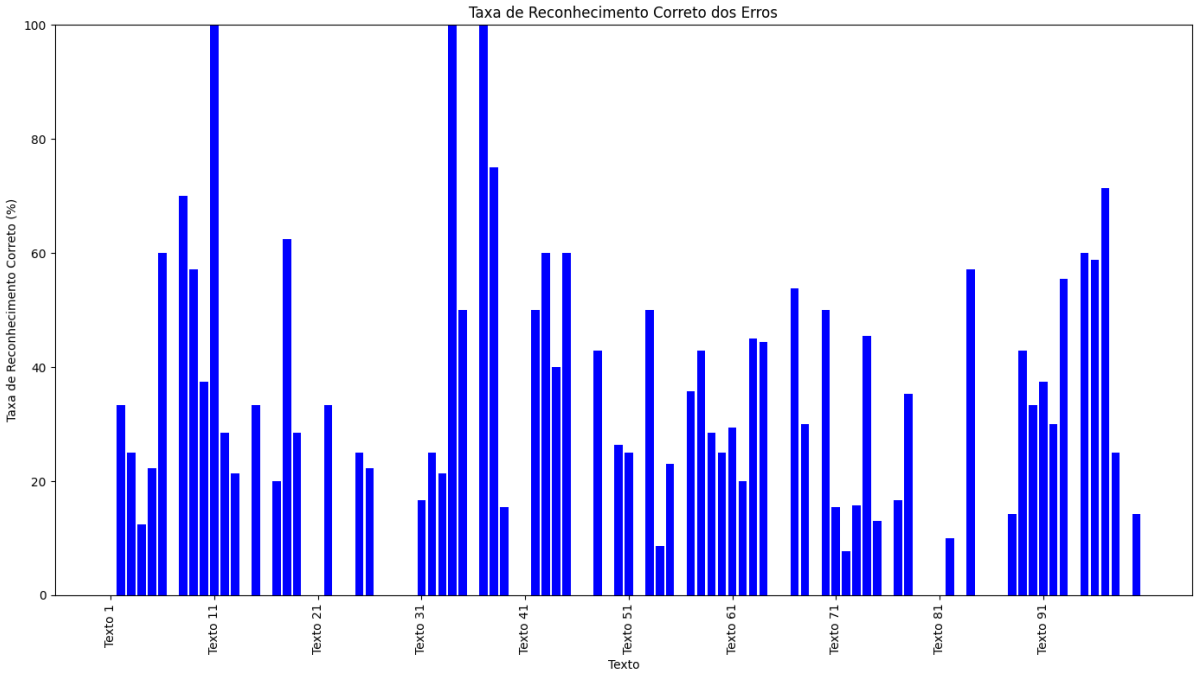

Figura 5.2: Taxa de reconhecimento por quantidade de erros

A aplicação demonstrou um melhor desempenho na identificação de erros ortográficos, tipográficos e gramaticais, como mostra a Figura 5.3. A identificação de erros de outras categorias pelos profissionais humanos nos textos utilizados para teste também pode ter impactado na variação da taxa de reconhecimento, visto a menor eficácia da plataforma para outros tipos de erros.

A quantidade de erros reconhecidos incorretamente teve uma média de 3,04, sugerindo que o *software* ainda apresenta limitações em distinguir corretamente alguns tipos de erros. A relação de erros reconhecidos correta e incorretamente em cada texto pode ser visualizado na Figura 5.4. Essa relação também é impactada por termos reconhecidos pelos profissionais humanos, mas que não estão incluídos no dicionário da aplicação, como os casos de nomes de marcas ou nomes próprios menos comuns.

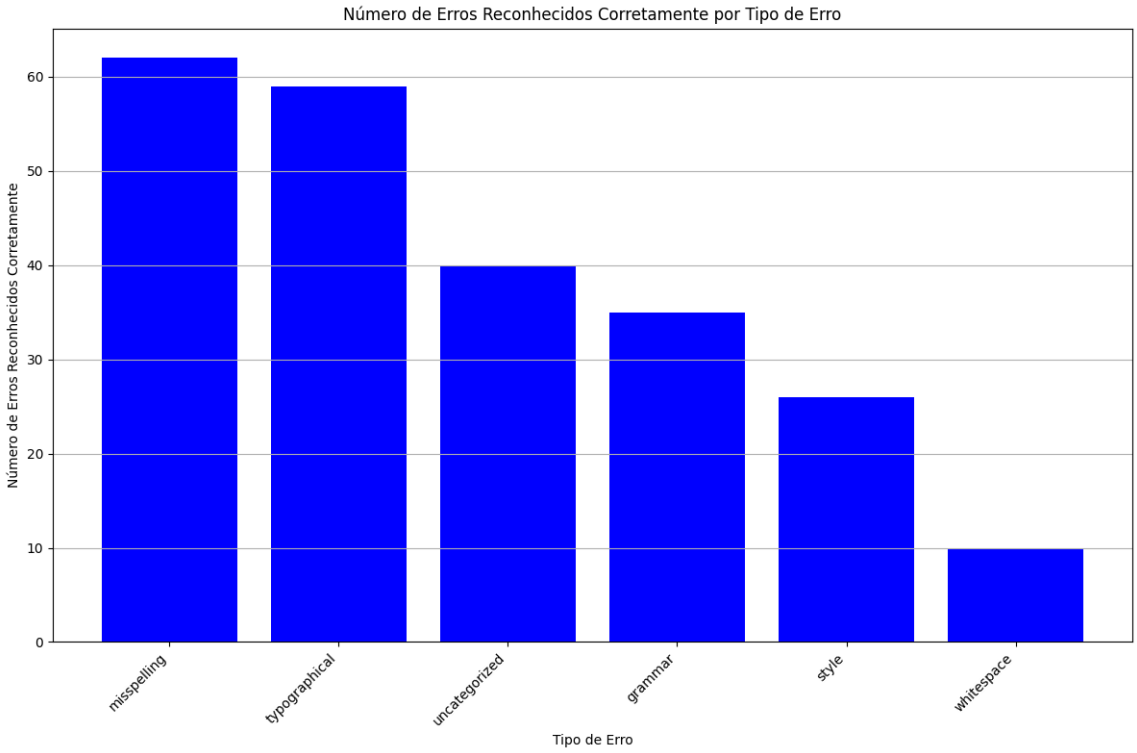

Figura 5.3: Relação dos tipos de erros reconhecidos corretamente

Para avaliar a acurácia da aplicação em relação a textos com diferentes quantidades de erros, foram analisados os 10 textos com o maior número de erros e os 10 com o menor. Nos textos com maior quantidade de erros, a média de erros reconhecidos corretamente subiu para 3,20, enquanto a quantidade de erros identificados incorretamente caiu para 2,00. O número de erros não reconhecidos aumentou para 6,10, mantendo a média de erros identificados por texto em 33,33%, a mesma porcentagem observada nos testes realizados com um conjunto de 100 textos. O tempo médio de processamento nesse cenário foi de 1167,94 milissegundos, indicando que não houve um aumento significativo no tempo de análise de textos com mais erros.

Por outro lado, nos textos com menor quantidade de erros, as médias de erros reconhecidos correta e incorretamente foram, respectivamente, de 0,60 e 2,10. O número de erros não reconhecidos foi reduzido para 2,60, o que levou a uma diminuição da média de erros reconhecidos por texto para 19,35%. Isso indica uma redução da acurácia em comparação com os testes realizados no conjunto total de textos. O tempo médio de processamento, nesse caso, foi de 1694,74 milissegundos, demonstrando que também não houve uma alteração significativa no tempo de análise para textos com uma média menor

Figura 5.4: Relação de erros reconhecidos correta e incorretamente

de erros.

5.3 Considerações Finais

Este capítulo apresentou uma análise do desempenho e da acurácia do *software* TUTOR-PT na identificação de erros em textos, utilizando uma amostra selecionada da base de dados de redações do UOL. A avaliação revelou alguns detalhes importantes sobre a eficácia do *software* e suas áreas de melhoria.

Em geral, o TUTOR-PT apresenta um tempo de processamento aceitável e uma taxa de reconhecimento de erros com maior acurácia para erros ortográficos e gramaticais, mas pouco eficiente no reconhecimento de erros em outras categorias. As variações na taxa de reconhecimento entre diferentes textos e a taxa de erros não reconhecidos apontam para áreas onde o *software* pode ser aprimorado.

6 Conclusões e Trabalhos Futuros

O presente trabalho apresentou o desenvolvimento de um *software* de avaliação semi-automática de textos em língua portuguesa, implementado como uma extensão da plataforma Google Docs. Inicialmente, foram discutidos alguns dos desafios enfrentados por alunos e professores no processo de escrita e avaliação de textos, destacando como ferramentas computacionais podem atuar como aliadas nessas tarefas. Também foram abordados conceitos fundamentais para o desenvolvimento de *softwares* de avaliação automática, além de trabalhos relacionados que apresentam propostas semelhantes a deste estudo.

O desenvolvimento da extensão baseou-se em uma arquitetura que integra ferramentas de frontend e backend para a análise textual. A interface foi construída com *React* e *Material UI*, integrando-se ao Google Docs por meio do *Google Apps Script*. Essa integração permitiu disponibilizar a extensão em uma plataforma de edição de texto amplamente utilizada, facilitando a adoção pelo público-alvo. O módulo de análise textual utilizou o *Natural Language Toolkit* para o pré-processamento do texto e a API externa do *LanguageTool* para a identificação de erros ortográficos e gramaticais. Entre os principais desafios do desenvolvimento, destaca-se o uso de uma API externa, o *LanguageTool*, para a análise dos textos, e a integração com o Google Docs. A disponibilização da extensão na loja oficial do Google foi outro obstáculo, já que a aplicação precisa atender a um conjunto de requisitos definidos pelo *Google Workspace Marketplace*.

Uma das abordagens sugeridas para a discussão de resultados é a realização de uma análise qualitativa da usabilidade e da experiência do usuário, por meio da disponibilização da extensão e da coleta de *feedback* através de formulários. A extensão desenvolvida durante este trabalho foi disponibilizada por um breve período, no entanto, devido às exigências da plataforma, não foi possível mantê-la acessível ao público, limitando-se ao uso local. Diante do baixo número de usuários que forneceram *feedback* durante esse período, optou-se por modificar a metodologia de avaliação, para realização de testes utilizando uma base de dados anotada por avaliadores humanos, especificamente um banco de

redações do ENEM. Assim, a análise dos resultados ficou restrita à análise do desempenho e da acurácia da aplicação. Durante a avaliação de desempenho, a extensão apresentou melhores resultados no reconhecimento de erros ortográficos e gramaticais, categorias que foram exploradas em trabalhos correlatos. A taxa média de reconhecimento de erros foi de aproximadamente 30%. Para textos com maior quantidade de erros, essa média se manteve em 33%, próxima à média obtida para todos os textos. Por outro lado, para textos com menor quantidade de erros, observou-se uma redução da acurácia, que caiu para 19%. O tempo médio de processamento da aplicação foi de 1.404,92 milissegundos, sem apresentar diferenças significativas entre textos com maior ou menor quantidade de erros.

Para trabalhos futuros, propõe-se a exploração de alternativas visando aprimorar a acurácia do reconhecimento de erros e expandir a análise para outras dimensões do texto. Também é possível realizar trabalhos visando a inclusão de validação da formatação do texto conforme as normas da ABNT e implementação de um módulo de detecção de plágio, o que pode beneficiar a conformidade e originalidade dos textos acadêmicos.

Outras possibilidades são a utilização de uma base de monografias previamente avaliadas por especialistas para verificar a eficácia do software nesse contexto e a implementação de novos algoritmos de detecção e correção de erros, visando aprimorar a eficiência da ferramenta para textos acadêmicos.

Bibliografia

BIRD, S.; KLEIN, E.; LOPER, E. *Natural language processing with Python: analyzing text with the natural language toolkit*. [S.l.]: "O'Reilly Media, Inc.", 2009.

BORGES, T. D. B.; MALINOSKI, S.; LIMA, V. M. do R.; SANTOS, A. M. dos. Escrita acadêmica e formação docente: contribuições possíveis. *Educação Por Escrito*, v. 11, n. 2, p. e31766–e31766, 2020.

BRASIL; INEP. *Redação no Enem 2018. Cartilha do participante*. [S.l.]: Diretoria de Avaliação da Educação Básica Brasília, 2018.

BROWN, S. *The C4 Model*. 2024. Disponível em: ⟨https://c4model.com/⟩.

CAMBRIA, E.; WHITE, B. Jumping nlp curves: A review of natural language processing research. *IEEE Computational intelligence magazine*, IEEE, v. 9, n. 2, p. 48–57, 2014.

CASELI, H.; FREITAS, C.; VIOLA, R. Processamento de linguagem natural. *Sociedade Brasileira de Computação*, 2022.

COSTA, L.; OLIVEIRA, E. H. T. de; JÚNIOR, A. C. Corretor automático de redações em língua portuguesa: um mapeamento sistemático de literatura. In: SBC. *Anais do XXXI Simpósio Brasileiro de Informática na Educação*. [S.l.], 2020. p. 1403–1412.

CRISTO, R. D. S. d. A prática docente na avaliação da produção textual: as marcas de correção. Universidade Estadual Paulista (Unesp), 2020.

DONG, G.; LIU, H. *Feature engineering for machine learning and data analytics*. [S.l.]: CRC press, 2018.

EISENSTEIN, J. Natural language processing. *Jacob Eisenstein*, 2018.

FERREIRA, J. *Google apps script: Web application development essentials*. [S.l.]: "O'Reilly Media, Inc.", 2014.

FONSECA, E.; MEDEIROS, I.; KAMIKAWACHI, D.; BOKAN, A. Automatically grading brazilian student essays. In: SPRINGER. *Computational Processing of the Portuguese Language: 13th International Conference, PROPOR 2018, Canela, Brazil, September 24–26, 2018, Proceedings 13*. [S.l.], 2018. p. 170–179.

Google, Inc. *Google Apps Script*. 2024. Acesso em: 17 de janeiro de 2024. Disponível em: ⟨https://developers.google.com/apps-script?hl=pt-br⟩.

GROUP, W. I. W. *Internationalization Tag Set (ITS) 2.0*. 2024. Disponível em: ⟨https://www.w3.org/International/multilingualweb/lt/drafts/its20/its20.html⟩.

JÚNIOR, C. R.; SPALENZA, M. A.; OLIVEIRA, E. de. Proposta de um sistema de avaliação automática de redações do enem utilizando técnicas de aprendizagem de máquina e processamento de linguagem natural. *Anais do Computer on the Beach*, p. 474–483, 2017.

JUNIOR, J. A. da S. Um avaliador automático de redaçoes. *Master's thesis, Universidade Federal do Espírito Santo*, 2021.

KE, Z.; NG, V. Automated essay scoring: A survey of the state of the art. In: *IJCAI*. [S.l.: s.n.], 2019. v. 19, p. 6300–6308.

KINOSHITA, J.; SALVADOR, L. do N.; MENEZES, C. E. D. de. Cogroo: a brazilian-portuguese grammar checker based on the cetenfolha corpus. In: CITESEER. *LREC*. [S.l.], 2006. p. 2190–2193.

LANGUAGETOOL. *Public HTTP Proofreading API*. 2024. ⟨https://dev.languagetool.org/public-http-api⟩. Acesso em: 01 out. 2024.

LanguageTool Community. *LanguageTool*. 2024. Acesso em: 17 de janeiro de 2024. Disponível em: ⟨https://languagetool.org/pt-BR/⟩.

MAHESH, B. Machine learning algorithms-a review. *International Journal of Science and Research (IJSR).[Internet]*, v. 9, p. 381–386, 2020.

MARCURSCHI, B.; SUASSUNA, L. Avaliação em língua portuguesa: contribuições para a prática pedagógica. *Belo Horizonte: Autêntica*, 2007.

MARINHO, J. C.; CORDEIRO, F.; ANCHIÊTA, R. T.; MOURA, R. S. Automated essay scoring: An approach based on enem competencies. In: SBC. *Anais do XIX Encontro Nacional de Inteligência Artificial e Computacional*. [S.l.], 2022. p. 49–60.

MARTINS, C. A.; MONARD, M. C.; MATSUBARA, E. T. Uma ferramenta computacional para auxiliar no pré-processamento de textos. In: *Anais do XXIII Congresso da Sociedade Brasileira de Computaçao-IV Encontro Nacional de Inteligência Artificial (ENIA), Campinas, SP*. [S.l.: s.n.], 2003. v. 6.

Meta, Inc. *React - A JavaScript library for building user interfaces*. 2024. Acesso em: 17 de janeiro de 2024. Disponível em: ⟨https://reactjs.org/⟩.

MONICO, J. F. G.; POZ, A. P. D.; GALO, M.; SANTOS, M. C. D.; OLIVEIRA, L. C. D. Acurácia e precisão: revendo os conceitos de forma acurada. *Boletim de Ciências Geodésicas*, Universidade Federal do Paraná, v. 15, n. 3, p. 469–483, 2009.

MUI. *Material-UI*. 2024. Acesso em: 17 de janeiro de 2024. Disponível em: ⟨https://mui.com/⟩.

MURPHY, K. P. *Machine learning: a probabilistic perspective*. [S.l.]: MIT press, 2012.

NARGESIAN, F.; SAMULOWITZ, H.; KHURANA, U.; KHALIL, E. B.; TURAGA, D. S. Learning feature engineering for classification. In: *Ijcai*. [S.l.: s.n.], 2017. v. 17, p. 2529–2535.

NETO, S. S. C.; FAVERO, E. L.; SANTOS, J. C. A. dos; FREITAS, S. N. de; JÚNIOR, M. A. N. Avaliação automática de redações na língua portuguesa baseada na coleta de atributos e aprendizagem de máquina. In: SBC. *Anais do XXXI Simpósio Brasileiro de Informática na Educação*. [S.l.], 2020. p. 1162–1171.

NIELSEN, J. Response times: The 3 important limits. *Nielsen Norman Group*, 1993. Accessed: 2024-09-11. Disponível em: ⟨https://www.nngroup.com/articles/response-times-3-important-limits/⟩.

PEREIRA, A. S.; SHITSUKA, D. M.; PARREIRA, F. J.; SHITSUKA, R. Metodologia da pesquisa científica. Brasil, 2018.

PINHO, C. M. d. A.; MOURA, A. F. d.; GASPAR, M. A.; NAPOLITANO, D. M. R. Identificação de deficiências em textos educacionais com a aplicação de processamento de linguagem natural e aprendizado de máquina. *ETD Educação Temática Digital*, UNICAMP, v. 24, n. 2, p. 350–372, 2022.

PINTO, S. C. S. *Processamento de linguagem natural e extração de conhecimento*. Tese (Doutorado), 2015.

SANTANA, R. H. C.; SANTANA, M. J.; ORLANDI, R. C. G. S.; SPOLON, R.; JUNIOR, N. C. Técnicas para avaliação de desempenho de sistemas computacionais. 1994.

SOARES, M.; PRATI, R. C.; MONARD, M. C. Pretext: a reestruturação da ferramenta de pré-processamento de textos. *Instituto de Ciências Matemáticas e de Computação. São Carlos: Universidade de São Paulo*, 2008.

TURIAN, J.; RATINOV, L.; BENGIO, Y. Word representations: a simple and general method for semi-supervised learning. In: *Proceedings of the 48th annual meeting of the association for computational linguistics*. [S.l.: s.n.], 2010. p. 384–394.

UOL. *Banco de Redações*. 2024. Acesso em: 14 set. 2024. Disponível em: ⟨https://educacao.uol.com.br/bancoderedacoes/⟩.

VÁZQUEZ-INGELMO, A.; GARCÍA-HOLGADO, A.; GARCÍA-PEÑALVO, F. J. C4 model in a software engineering subject to ease the comprehension of uml and the software. In: IEEE. *2020 IEEE Global Engineering Education Conference (EDUCON)*. [S.l.], 2020. p. 919–924.

WILSON, J.; ROSCOE, R. D. Automated writing evaluation and feedback: Multiple metrics of efficacy. *Journal of Educational Computing Research*, SAGE Publications Sage CA: Los Angeles, CA, v. 58, n. 1, p. 87–125, 2020.