UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO CENTRO DE CIÊNCIAS EXATAS, NATURAIS E DA SAÚDE

DEPARTAMENTO DE COMPUTAÇÃO
CIÊNCIA DA COMPUTAÇÃO

ALEXANDRE MELO MOULIN BREDA

APRIMORAMENTO NA FERRAMENTA GERADORA DE GRAMÁTICAS LOCAIS

ALEGRE - ES

SETEMBRO 2024

APRIMORAMENTO NA FERRAMENTA GERADORA DE GRAMÁTICAS LOCAIS

Trabalho de conclusão de curso apresentado ao Departamento de Computação do Centro de Ciências Exatas, Naturais e da Saúde da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação.

por

ALEXANDRE MELO MOULIN BREDA

Orientador
Jacson Rodrigues Correia da Silva

Coorientadora
Juliana Pinheiro Campos Pirovani

Universidade Federal do Espírito Santo

ALEGRE - ES
SETEMBRO 2024

ALEXANDRE MELO MOULIN BREDA

APRIMORAMENTO NA FERRAMENTA GERADORA DE GRAMÁTICAS LOCAIS

Trabalho de Conclusão de Curso apresentado ao Departamento de Computação do Centro de Ciências Exatas, Naturais e da Saúde da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação.

Aprovado em 01 de outubro de 2024.

COMISSÃO EXAMINADORA

Prof. Dr. Jacson Rodrigues Correia da Silva
Universidade Federal do Espírito Santo
Orientador

Prof.ª Dr.ª Juliana Pinheiro Campos Pirovani
Universidade Federal do Espírito Santo
Coorientadora

Prof. Dr. Rodrigo Freitas Silva
Universidade Federal do Espírito Santo

RESUMO

O Processamento de Linguagem Natural (PLN) é uma área da inteligência artificial que visa a interação entre computadores e humanos por meio da linguagem natural. Sua importância tem crescido exponencialmente com o destaque que ferramentas como o modelo GPT conquistaram. Outra ferramenta que auxilia no PLN como o Unitex tem a capacidade de pré-processar um texto e aplicar uma Gramática Local (GL) sobre ele, para realizar uma busca textual. O Gerador de GL é um facilitador para pessoas que desejam realizar uma busca textual, mas não têm o conhecimento necessário para criar as próprias GLs. Por isso, seu objetivo é gerar GLs a partir de exemplos fornecidos pelo usuário. Contudo, limitações no processo de generalização de regras foram encontradas. Por isso, este trabalho propôs um aprimoramento na análise em contexto dos exemplos e na correção de erros encontrados, para garantir a precisão das GLs geradas, aprofundando os estudos em PLN e contribuindo para o aumento da qualidade das buscas textuais realizadas nesta e em outras áreas do conhecimento.

Palavras-chave: Processamento de Linguagem Natural; Geração de Gramática Local; Inteligência Artificial; Unitex.

LISTA DE FIGURAS

Figura 1: Exemplo de uma gramática local..... 9
Figura 2: Exemplo de uma gramática local com código lexical..... 10
Figura 3: Fragmento do texto Senhora pré-processado..... 14
Figura 4: Lista de tokens gerada..... 15
Figura 5: Lista de palavras gerada..... 16
Figura 6: Alguns códigos lexicais (tags) do Unitex..... 16
Figura 7: Exemplo de grafo no Unitex..... 17
Figura 8: Exemplo de resultado da aplicação de um grafo em um texto.....17
Figura 9: Exemplo de arquivo .grf.....18
Figura 10: Tela Inicial da ferramenta Gerador de GL..... 20
Figura 11: Seleção de exemplos para gerar GL..... 21
Figura 12: GL gerada visualizada no Unitex.....21
Figura 13: Universal POS Tags..... 23
Figura 14: Grafo gerado com nós desalinhados.....26
Figura 15: Exemplos de selecionados para construir GL..... 27
Figura 16: GL gerada com exemplos da Figura 15..... 27
Figura 17: Imprecisão gerada por falta de verificação no contexto..... 27
Figura 18: Episódios classificados utilizando o Unitex..... 31
Figura 19: Respostas do serviço spaCy..... 31
Figura 20: Episódios depois da filtragem.....32
Figura 21: Fluxo de generalização de regras após a inclusão do serviço spaCy.....33
Figura 22: Exemplos quando o contexto importa..... 35
Figura 23: GL gerada sem levar o contexto em consideração..... 35
Figura 24: GL gerada levando o contexto em consideração..... 36
Figura 25: Tela de aplicação da GL no texto aberto..... 36
Figura 26: Exemplos de substantivos e números.....37
Figura 27: Erro - Selecionar caracteres não reconhecidos..... 38
Figura 28: Alerta - Selecionar caracteres não reconhecidos.....38
Figura 29: Erro - Gerar GL sem texto carregado.....38
Figura 30: Alerta - Gerar GL sem texto carregado..... 39
Figura 31: GL gerada com nós posicionados automaticamente..... 39

LISTA DE SIGLAS

PLN	Processamento de Linguagem Natural
LN	Linguagem Natural
IA	Inteligência Artificial
GLC	Gramáticas Livres de Contexto
ER	Expressões Regulares
GL	Gramática Local
JNI	Java Native Interface
MEV	Modelo de Espaço Vetorial
API	Application Programming Interface
REN	Reconhecimento de Entidade Nomeada
POS	Part Of Speech

SUMÁRIO

1. INTRODUÇÃO..... 8

1.1 Objetivos..... 11

1.1.1 Objetivo geral..... 11

1.1.2 Objetivos específicos..... 11

2. REVISÃO DE LITERATURA..... 12

2.1 Processamento de Linguagem Natural..... 12

2.2 Trabalhos relacionados..... 13

2.3 Unitex..... 13

2.4 Ferramenta Gerador de GL..... 19

2.5 spaCy..... 22

3. METODOLOGIA..... 25

3.1 Análise da Ferramenta..... 25

3.2 Integração com spaCy..... 28

3.3 Processo de generalização de regras..... 30

3.4 Aplicação de GL gerada..... 33

4. RESULTADOS..... 34

4.1 Aprimoramentos..... 34

4.2 Limitações..... 39

5. CONCLUSÃO..... 41

REFERÊNCIAS..... 42

1. INTRODUÇÃO

O Processamento de Linguagem Natural (PLN) estuda a compreensão e geração da Linguagem Natural (LN) pelos computadores. Um dos focos do PLN é fazer com que o computador consiga entender a LN, processar uma resposta e entregar essa resposta ao usuário em LN. Não é possível obrigar o usuário a aprender linguagem de máquina, mas é possível fazer o computador reconhecer LN (CHOPRA, PRASHAR, SAIN, 2013). LN é a linguagem usada no processo de comunicação entre pessoas e de acordo com Kumar (2013) “Processamento de Linguagem Natural é uma área da Ciência da Computação e Linguística preocupada com as interações entre computadores e linguagens humanas (naturais)”.

O estudo relacionado ao PLN tem origem nos anos de 1950, como uma subárea da Inteligência Artificial (IA) e linguística. Ambas áreas tinham um problema em comum, a busca por informações significativas em uma alta quantidade de textos. Foram inúmeras contribuições até os dias de hoje, mas é possível ressaltar as contribuições de: Chomsky em 1956 que investigou conceitos da estrutura linguística para criar gramáticas simples que conseguem representar todas as sentenças do Inglês (CHOMSKY, 1956); Backus Naur em 1963 e sua notação para representar Gramáticas Livres de Contexto (GLC); e Kleene que definiu Expressões Regulares (ER) que foram implementadas no comando grep do Unix (NADKARNI, OHNO-MACHADO, CHAPMAN, 2011).

O PLN se divide em: Processamento de Linguagem Natural que foca em analisar e produzir uma representação da linguagem; e Geração de Linguagem Natural que produz uma linguagem a partir de uma representação (LIDDY, 2001).

É possível ressaltar algumas tarefas pesquisadas na área do PLN como: Sumarização Automática focada em produzir um sumário do texto fornecido de forma coerente e consistente (RINO, PARDO, 2003); Análise de Discurso que busca identificar a estrutura de diferentes textos conectados, ou seja, a natureza das relações entre sentenças (CHOPRA, PRASHAR, SAIN, 2013) e Reconhecimento de Entidades Nomeadas (REN) que analisa um texto e determina se a entidade analisada é o nome de uma pessoa, um lugar, uma organização, uma data, entre outros (FONSECA *et al*, 2015).

Segundo (GROSS, 1999), “Gramáticas Locais (GL) são gramáticas de estado finito ou autômatos de estado finito que representam os conjuntos de regras de uma linguagem natural”. Por exemplo, um falante de língua portuguesa pode falar: “O dia está lindo”, e outro falar: “O dia realmente está lindo” que tem o mesmo sentido. É possível construir uma GL para representar ambos exemplos, como na Figura 1.

Figura 1: Exemplo de uma gramática local
Fonte: O autor

A GL presente na Figura 1 tem a seguinte estrutura: o primeiro nó (triângulo) representa o início do grafo; os nós em diante reconhecem uma palavra e o último nó (quadrado circunscrito) que demarca o fim do autômato.

Outras variações da sentença acima também poderiam ser representadas ao utilizar códigos lexicais e flexões adicionais com o auxílio de ferramentas específicas para o processamento de corpus textuais como a ferramenta Unitex/GramLab[1]. No exemplo da Figura 1, foi apresentada uma GL que é capaz de localizar somente duas frases. Com o auxílio de códigos lexicais e flexões, aquela GL pode ser modificada para identificar uma gama bem maior de frases. Assim, GLs são capazes de capturar conjuntos de expressões que possuem características em comum, sendo elas sintáticas ou semânticas. Na Figura 2, por exemplo, a utilização do código lexical <A>, usado para representar a classe gramatical dos adjetivos, no lugar do adjetivo “lindo” possibilita que a GL consiga identificar frases com adjetivos

[1] Ferramenta Unitex/GramLab: https://unitexgramlab.org/pt

diferentes, como: “O dia está maravilhoso”, “O dia realmente está horrível”, “O dia está calmo”, entre outros.

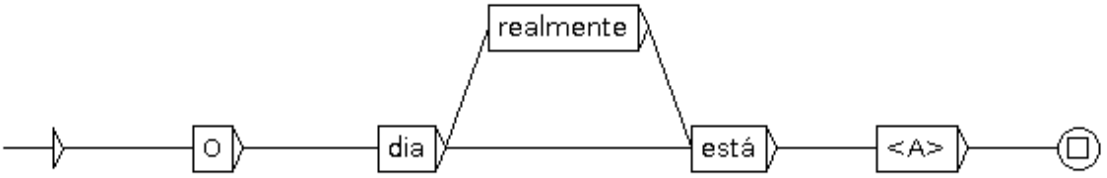

Figura 2: Exemplo de uma gramática local com código lexical
Fonte: O autor

Dessa forma, é possível para o PLN auxiliar na análise e compreensão de textos fornecidos pelo usuário da tecnologia, o qual pode ser uma obra de literatura, um grupo de frases e até mesmo toda uma conversa na LN.

ER e GL são formas para buscar padrões existentes em textos escritos em linguagem natural, mas muitas das vezes a construção dessas estruturas é feita de forma manual, o que pode acarretar em erros e no mau funcionamento do algoritmo, além de necessitar de um alto nível de conhecimento no assunto para construí-las. Mas, mesmo sendo ambas ferramentas para buscar padrões, elas são utilizadas em casos diferentes. ERs geralmente são utilizadas para localizar padrões presentes em cadeias de caracteres, ao passo que, quando há necessidade de localizar construções mais complexas, o uso de gramáticas locais é recomendado (PICOLI et al., 2015).

Para auxiliar nesse problema pode ser utilizado o Gerador de GL[2] (GANDES, 2020), uma ferramenta que possibilita um linguista ou qualquer pessoa interessada, sem conhecimento prévio de computação e de GL, obter GLs para extrair informações em textos.

Para que o Gerador de GL possibilite ao usuário extrair informações do texto, é necessário que esta ferramenta generalize os exemplos selecionados pelo usuário. Mas, a funcionalidade de generalização de regras não levava o contexto dos exemplos selecionados em consideração para efetuar a generalização. Além disso, o Gerador de GL só pode ser executado em sistemas operacionais Linux e

[2] Link do Gerador de GL: https://github.com/geradordegl

exemplos selecionados que contenham números, símbolos, siglas e abreviações causavam erros no funcionamento da ferramenta. O aumento na precisão das GL geradas pelo processo de generalização de regras e o tratamento dos erros descobertos durante os testes da ferramenta foram pontos de melhoria abordados. Portanto buscou-se nesse trabalho o aprimoramento da ferramenta Gerador de GL, trazendo bons resultados para o usuário final e utilizadores de gramáticas.

1.1 Objetivos

1.1.1 Objetivo geral

Este trabalho teve como objetivo aprimorar a ferramenta Gerador de GL, aprofundando o estudo relacionado a PLN e geração de GL.

1.1.2 Objetivos específicos

Como objetivos específicos é possível citar:

1. Analisar a ferramenta Gerador de GL em busca de pontos de melhoria;
2. Corrigir erros encontrados;
3. Melhorar o processo de generalização de regras e geração de GL;
4. Executar testes na nova versão da ferramenta.

2. REVISÃO DE LITERATURA

O PLN se baseia na capacidade do computador em entender LN, com isso, muitos trabalhos que aprofundaram o estudo de PLN com a criação de ferramentas que melhoram processos e tarefas importantes para a compreensão de LN e busca textual como o PyGER (FELTRIM e SOUZA, 2009), o trabalho de Choubey e Kharat (2009), o estudo de Zhong *et al*. (2018) e o ULMFiT de Howard e Ruder (2018). Dentre as ferramentas que utilizam conceitos de PLN, um conjunto muito utilizado de funções de PLN é o Unitex. Mas, este trabalho dará um maior destaque ao Gerador de GL e ao spaCy. Para facilitar o entendimento do leitor, este capítulo está dividido da seguinte forma, primeiramente o PLN, depois outros trabalhos que se utilizam de PLN, então o Unitex, por fim o Gerador de GL e o spaCy.

2.1 Processamento de Linguagem Natural

As ferramentas que auxiliam na análise textual, geralmente disponibilizam funções que permitem fazer o processamento do corpo textual desejado. Para isso, algumas técnicas de PLN essenciais são utilizadas.

Para realizar o pré-processamento de um corpo textual, é necessário normalizar o texto. Normalizar significa reescrever o texto em um formato que as ferramentas de análise textual entendam. O processo que consiste em separar as palavras de um texto para que elas possam ser futuramente classificadas se chama tokenização (JURAFSKY, MARTIN, 2024).

Após ter realizado a normalização é possível utilizar técnicas como *Part Of Speech* (POS) *tagging* e REN para gerar dados necessários para realizar a análise textual. No caso de *POS tagging*, todas as palavras são etiquetadas com uma classe gramatical como: substantivo, verbo, adjetivo, etc. Para o caso de REN, a técnica tenta localizar e classificar as palavras em nomes, datas, organizações, lugares, etc.

Algumas ferramentas se utilizam dessas técnicas para melhorar processos de análise textual, enquanto outras estudam formas de aprimorar algoritmos de análise textual que utilizam essas técnicas.

2.2 Trabalhos relacionados

Uma ferramenta que busca auxiliar a escrita de textos acadêmicos foi desenvolvida por Feltrim e Souza (2009). O PyGER - Python Gerador de Expressões Regulares se inspirou no AMADEUS e tem como papel auxiliar a ferramenta SciPo. O SciPo auxilia a escrita de textos acadêmicos na área da ciência da computação. A ideia foi utilizar o GERBoW e PyGER.

Em seu estudo, Zhong *et al*. (2018) comparam a efetividade que ERs geradas automaticamente têm ao aplicá-las em 3 bases de dados, sendo duas sintéticas (KB13 e NL-RX) e uma de dados do mundo real (RegexLib). Após analisar as três bases de dados, notou-se que as sentenças presentes em RegexLib são muito mais complexas do que as presentes nas bases de dados sintéticas. Os testes realizados permitem concluir que o aumento na complexidade das sentenças resultou na baixa efetividade encontrada ao aplicar as ERs geradas na base de dados do mundo real em comparação com as bases de dados sintéticas.

Um trabalho realizado por Choubey e Kharat (2009) explorou o impacto que o uso de um algoritmo genético tem no processo de indução de gramáticas livres de contexto. Ele nota que o processamento necessário para induzir a gramática pode ser otimizado pelo uso de algoritmos genéticos.

Howard e Ruder (2018) trabalharam o conceito de *transfer learning* no contexto de PLN. Então, apresentaram o modelo *Universal Language Model Fine-tuning* (ULMFiT), um modelo que visava a melhora da classificação textual. O modelo superou métodos tradicionais de classificação textual e se provou muito útil em cenários reais, por trabalhar bem com menos dados rotulados.

De forma parecida com os trabalhos citados, o Gerador de GL se utiliza de PLN para auxiliar na realização de uma tarefa. Similar ao PyGER, o Gerador de GL trabalha em conjunto com outra ferramenta, nesse caso, a ferramenta é o Unitex/GramLab.

2.3 Unitex

O Unitex pode ser definido como um conjunto de programas específicos para PLN e utiliza dicionários, gramáticas e tabelas do léxico-gramática. Atualmente, está na versão 3.3, postada no dia 09 de Setembro de 2021 (PAUMIER, 2021).

É um *software* de código aberto com suporte para várias línguas. Pode realizar um pré-processamento do texto analisado que consiste em: normalizar os separadores; dividir em sentenças; normalizar formas não ambíguas; executar a tokenização para gerar uma lista de ocorrência das palavras e símbolos encontrados; e aplicar um dicionário.

Para um exemplo prático, o texto será Senhora, de José de Alencar, que vem por padrão na instalação da ferramenta para o Português. Após abrir o Unitex, abrir o texto e realizar o pré-processamento, a ferramenta apresenta as três telas representadas pelas Figuras 3, 4 e 5.

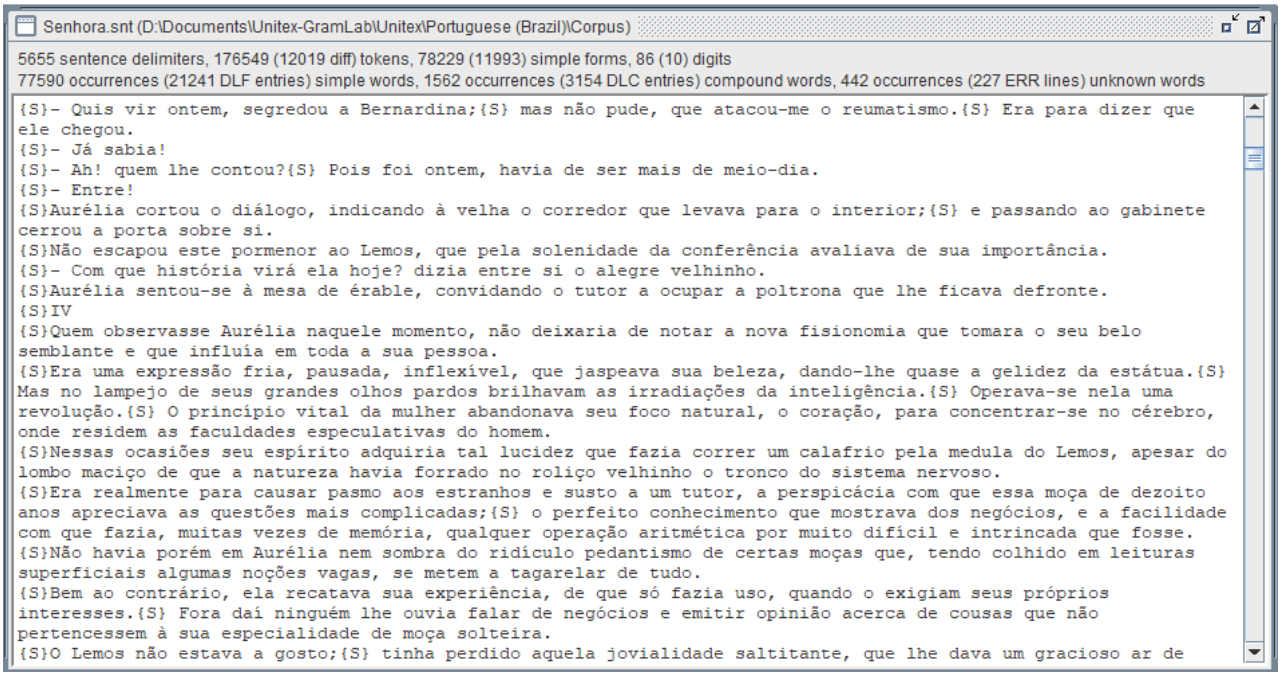

Figura 3: Fragmento do texto Senhora pré-processado

Fonte: O autor

Na Figura 3, é apresentado o fragmento do texto Senhora, após ser pré-processado. O Unitex demarca com um “{S}” cada sentença no texto e isso auxilia a aplicação de outras funções da ferramenta.

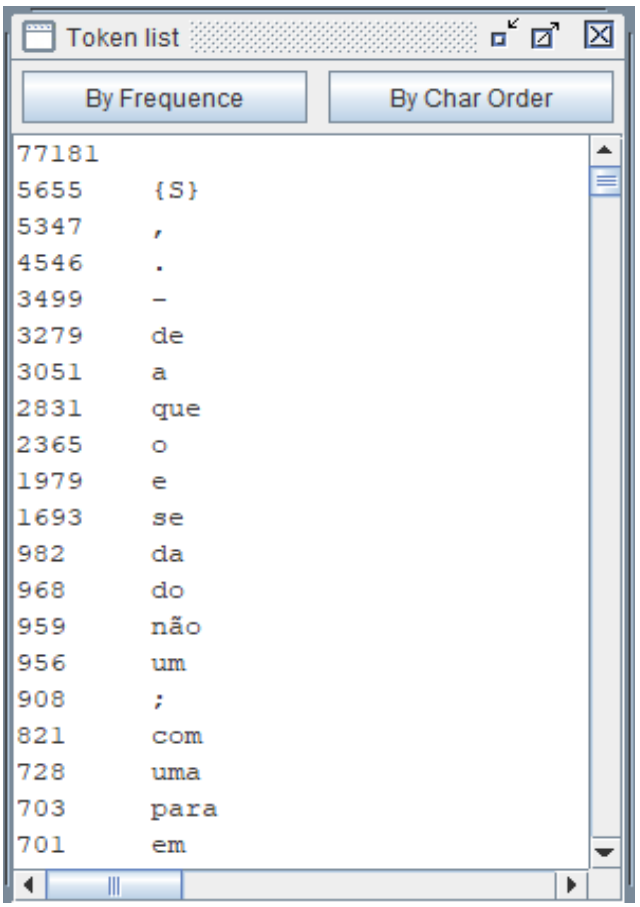

Figura 4: Lista de tokens gerada

Fonte: O autor

Na Figura 4 está uma lista de todos os *tokens* obtidos após o pré-processamento. A lista pode ser organizada por ordem alfabética e por ordem de frequência. Por padrão, a ordenação por frequência é usada.

Na Figura 5, à direita estão todas as palavras desconhecidas pelos dicionários do Unitex durante o pré-processamento, enquanto que à esquerda estão duas listas com palavras reconhecidas, uma com palavras simples e outra com palavras compostas. Uma curiosidade sobre o funcionamento do pré-processamento do Unitex é que, ao localizar uma palavra, ele apresenta todas as classificações gramaticais possíveis que aquela palavra pode assumir. Como no caso da palavra “a”, que pode ser uma preposição, um artigo, um pronome demonstrativo, entre outros. Em ambas as telas é possível ver palavras escritas em verde, essas palavras são os códigos lexicais que o Unitex reconhece. No manual de uso da ferramenta estão listados alguns códigos lexicais comuns em várias línguas diferentes, representados pela Figura 6.

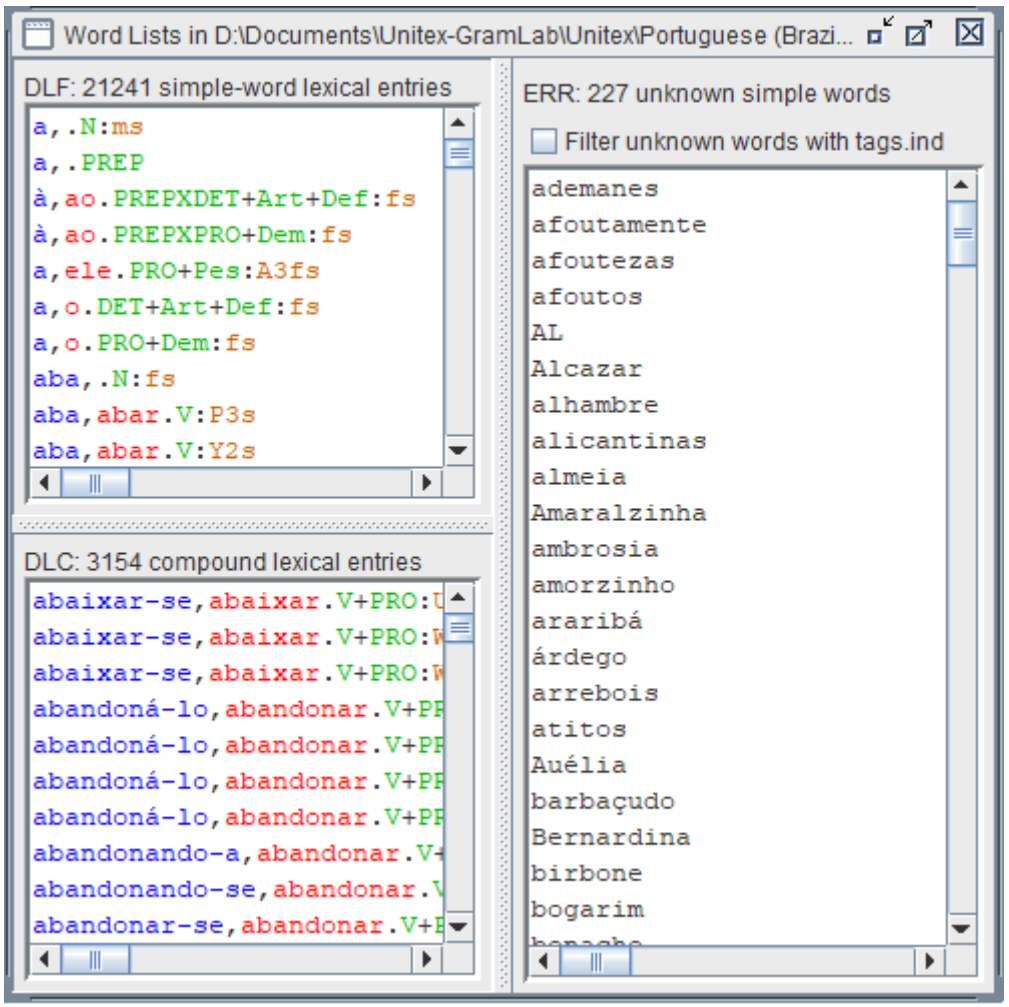

Figura 5: Lista de palavras gerada

Fonte: O autor

Code	Description	Examples
A	adjective	fabulous, broken-down
ADV	adverb	actually, years ago
CONJC	coordinating conjunction	but
CONJS	subordinating conjunction	because
DET	determiner	each
INTJ	interjection	eureka
N	noun	evidence, group theory
PREP	preposition	without
PRO	pronoun	you
V	verb	overeat, plug-and-play

Figura 6: Alguns códigos lexicais (*tags*) do Unitex

Fonte: Manual da ferramenta Unitex/GramLab

Após pré-processar um texto no Unitex é possível aplicar uma GL ao mesmo. As gramáticas geradas no Unitex são representadas por grafos e subgrafos, ou seja, um grafo inicia a execução de outro grafo construído previamente para reconhecer alguma estrutura e são formados por nós que se ligam por transições (linhas). Por exemplo, a Figura 7 mostra um grafo que localiza as ocorrências do verbo “levar” acompanhado do pronome “lhe” no tempo verbal do passado que é reconhecido pelo código lexical <levar.V:J>.

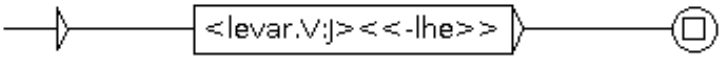

Figura 7: Exemplo de grafo no Unitex

Fonte: O autor

Ao aplicar o grafo da Figura 7 no texto pré-processado anteriormente é obtido o resultado presente na Figura 8. Clicar em uma das palavras marcadas leva o usuário a tela presente na Figura 3, mas com a palavra clicada marcada.

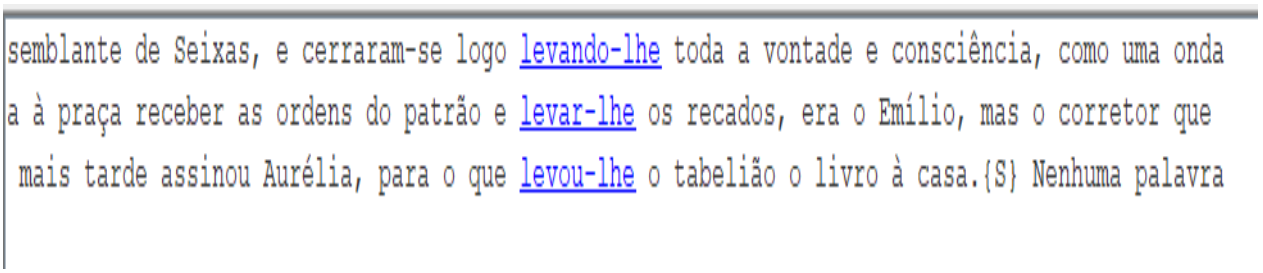

Figura 8: Exemplo de resultado da aplicação de um grafo em um texto

Fonte: O autor

```
1   #Unigraph
2   SIZE 1188 840
3   FONT Times New Roman:  10
4   OFONT Arial Unicode MS:B 12
5   BCOLOR 16777215
6   FCOLOR 0
7   ACOLOR 13487565
8   SCOLOR 16711680
9   CCOLOR 255
10  DBOXES y
11  DFRAME y
12  DDATE y
13  DFILE y
14  DDIR n
15  DRIG n
16  DRST n
17  FITS 100
18  PORIENT L
19  #
20  3
21  "<E>" 70 200 1 2
22  "" 353 200 0
23  "<levar.V:J><<-lhe>>" 151 200 1 1
```

Figura 9: Exemplo de arquivo .grf

Fonte: O autor

Os grafos do Unitex são salvos na extensão ".grf". Um arquivo .grf segue a estrutura mostrada pela Figura 9. O Unitex utiliza as informações desse arquivo para desenhar o grafo. A Figura 9 se refere ao grafo presente na Figura 7. O texto "#Unigraph" é somente um comentário. Começando na linha 2 até a linha 18 estão configurações referentes a apresentação e formatação do grafo. O número na linha 20 representa o número de nós presentes, incluindo os estados de início e fim do grafo. Essas linhas têm uma ordem, a linha 21 representa o nó de início do grafo, a linha seguinte é nó de fim do grafo e as próximas linhas são os demais nós do grafo. Os seguintes tópicos representam as informações que cada nó (representado em uma linha) deve conter, respectivamente:

- Palavra ou código lexical que deve ser reconhecido;
- Posição horizontal;
- Posição vertical;
- Número de transições que saem daquele nó;
- Nós alcançáveis a partir daquele nó.

Os nós recebem números de identificação de forma implícita, começando com 0 para o nó inicial, 1 para o nó final e 2 em diante para os demais nós do grafo. Por exemplo, na linha 23 o nó contém a expressão “<levar.V:J><<-lhe>>”, está localizado no ponto (151; 200) na tela de representação do Unitex, tem uma transição saindo dele e alcança o nó com o número 1 (nó final apresentado na linha 22 da Figura 9).

Um desenvolvedor que deseja utilizar as funções do Unitex em conjunto com uma linguagem de programação pode compilar a *Java Native Interface* (JNI). Esse recurso foi utilizado para implementar o Gerador de GL.

2.4 Ferramenta Gerador de GL

Tudo teve início no trabalho de Gandes (2020), o qual desenvolveu a ferramenta visando uma interface amigável e a função de gerar GL de forma semiautomática. Houve uma grande inspiração no algoritmo usado por Mannila, Toivonen, Verkamo (1997) devido a descoberta de eventos frequentes que acabavam gerando erros no cálculo de frequência da palavra. Representada pela Figura 10, a ferramenta possui uma tela inicial que permite fazer o carregamento de um texto, via opção Carregar Texto no menu Texto.

Figura 10: Tela Inicial da ferramenta Gerador de GL

Fonte: O autor

Antes de gerar a gramática, o usuário deve selecionar exemplos do que ele deseja que a GL gerada seja capaz de identificar, pensando em quais classes gramaticais ou palavras está buscando. Após selecionar os exemplos, é necessário clicar na opção Selecionar Texto Marcado Como Exemplo no menu Exemplos ou simplesmente pelo atalho Ctrl+S como consta na Figura 11. No caso da Figura 11, o usuário estava buscando localizar o artigo indefinido “um” seguido de um substantivo.

Com os exemplos selecionados, tudo que precisa ser feito é clicar na opção Gerar Gramática do menu Gramática ou usar o atalho Ctrl+G. A ferramenta gera um arquivo .grf que poderá ser salvo no diretório de escolha do usuário. A Figura 12 retrata a visualização obtida ao abrir a GL gerada anteriormente no aplicativo Unitex.

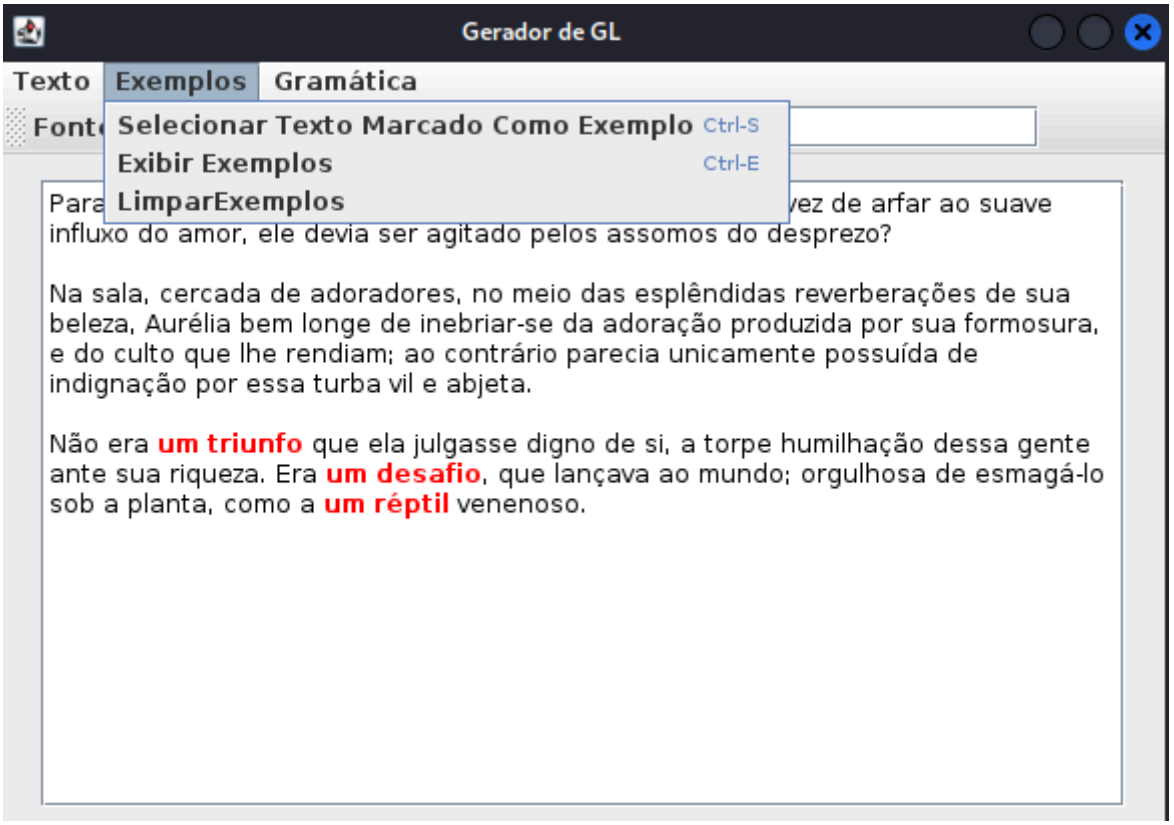

Figura 11: Seleção de exemplos para gerar GL

Fonte: O autor

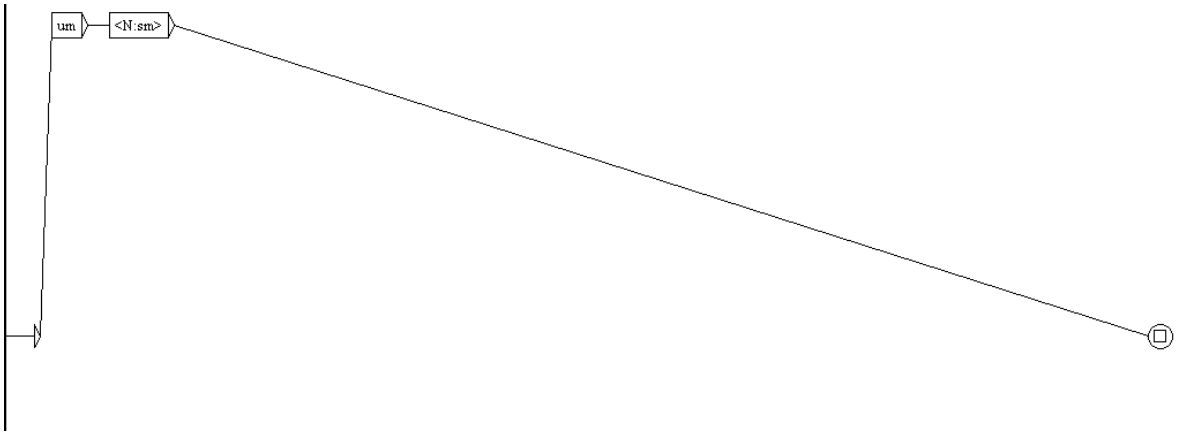

Figura 12: GL gerada visualizada no Unitex

Fonte: O autor

Ao final do trabalho de Gandes (2020), a ferramenta funcionava com algumas limitações como: os exemplos selecionados deveriam ser todos do mesmo tamanho; não havia o reconhecimento de flexões de palavras e existiam *bugs* com

exemplos já selecionados que eram removidos da seleção, mas mesmo assim continuavam sendo relevantes para o processo de geração da GL.

No trabalho de Minarrine (2022) houveram inúmeras melhorias, tanto em relação a erros na interface quanto a geração de GLs. A ferramenta manteve sua interface da mesma forma de antes, mas com erros corrigidos. O Gerador de GL passou a considerar flexões da palavra de exemplo e trabalha com exemplos de tamanho variado. Além disso, o problema relacionado à exclusão de exemplos foi resolvido.

Contudo, a ferramenta ainda apresenta alguns problemas, como a impossibilidade de utilização em outros sistemas operacionais, necessita de um melhor tratamento de erros e precisa de alterações no algoritmo de generalização de regras. O processo de generalização de regras não leva o contexto dos exemplos em consideração. Por isso, o auxílio de uma ferramenta que possibilite a análise em contexto era necessária.

2.5 spaCy

O spaCy é uma biblioteca de código aberto disponível na linguagem Python que fornece funções simplificadas de PLN e objetos que representam elementos linguísticos, as funções podem ser utilizadas para realizar os processos de tokenização, *POS tagging* e REN. Para utilizar essas estruturas é necessário importar um dos modelos pré-treinados que estão disponíveis para diversas línguas. Esses modelos estão disponíveis nos formatos “sm”, “md” e “lg”. O que muda de um tamanho para outro são as estruturas de vetores presentes internamente e o nível de treinamento que essas estruturas tiveram (VASILEV, 2020).

O processo de uso do spaCy pode ser explicado como um grupo de *pipelines* que primeiro realiza a tokenização e depois processa o texto fornecido utilizando o primeiro *pipeline*, gera um objeto contendo o texto classificado pelo *pipeline* e utiliza esse objeto no próximo *pipeline*, até que o processamento termine. Normalmente, um *pipeline* treinado inclui um etiquetador (*tagger*), um lematizador (*lemmatizer*), um analisador (*parser*) e um reconhecedor de entidades (*entity recognizer*) (EXPLOSION AI, 2024).

Os modelos pré-treinados do spacy são baseados em vetores de palavras, no caso dos modelos “sm”. Em contrapartida, os modelos “md” e “lg” utilizam

embeddings mais completos e densos que passaram por treinamentos mais detalhados. Normalmente, as *POS tags* que o spaCy atribui às palavras analisadas respeitam o *Universal POS Tags* que fazem parte do *framework Universal Dependencies*. A Figura 13 ilustra as POS tags que foram utilizadas nas análises e nas alterações do código do Gerador de GL.

ADJ: adjective
ADP: adposition
ADV: adverb
AUX: auxiliary
CCONJ: coordinating conjunction
DET: determiner
INTJ: interjection
NOUN: noun
NUM: numeral
PART: particle
PRON: pronoun
PROPN: proper noun
PUNCT: punctuation
SCONJ: subordinating conjunction
SYM: symbol
VERB: verb
X: other

Figura 13: *Universal POS Tags*
Fonte: Universal Dependencies

O conceito de embeddings se baseia em Modelos de Espaço Vetorial (MEV). Em alguns casos, é necessário treinar um modelo específico para o que se deseja trabalhar. Durante o treinamento as entradas fornecidas ao modelo são convertidas para vetores. Então, após treinado, o modelo consegue atribuir semelhanças entre entradas com base na distância de cada vetor no espaço. Ou seja, palavras com significados semelhantes ou utilizadas em contextos semelhantes possuem vetores semelhantes neste modelo, que estão próximos uns dos outros no espaço.

No contexto de PLN, o MEV é chamado de *espaço semântico,* a representação das entradas é chamada de representação distribuída e ao invés de falar somente *embeddings*, o termo *word embeddings* aparece com frequência. Então, as entradas de modelos de *word embeddings* podem ser palavras,

documentos, ou qualquer outro item com significado semântico que possa carregar similaridade com outros presentes no espaço semântico (PILEHVAR, CAMACHO-COLLADOS, 2021).

3. METODOLOGIA

Antes de realizar os testes na ferramenta Gerador de GL foi necessário estudar o manual da ferramenta Unitex e aprofundar os conhecimentos em PLN e GL. Os testes presentes nessa seção foram feitos utilizando o texto Senhora, de José de Alencar. Demais testes foram realizados para avaliar como a ferramenta se desempenhava quando o contexto deve ser levado em consideração e identificar erros a serem corrigidos.

O Gerador de GL foi desenvolvido na linguagem *Java* trabalhando em conjunto com a JNI do Unitex. Para realizar a análise em contexto dos exemplos informados foi utilizada a ferramenta spaCy e as demais alterações da ferramenta foram feitas utilizando bibliotecas da linguagem *Java*. Vale ressaltar que existem muitas outras bibliotecas que poderiam ser utilizadas para realizar a análise em contexto, como Apache OpenNLP[3], FreeLing[4] e NLTK[5]. Mas, o spaCy foi utilizado por possuir estruturas simples de usar, pela eficiência dos pipelines de processamento e por possuir modelos pré-treinados de fácil utilização.

3.1 Análise da Ferramenta

Foram realizados alguns testes na ferramenta Gerador de GL, tanto para entender seu funcionamento quanto para identificar erros de usabilidade e testar seu nível de generalização. Para execução dos testes foram usados a obra Senhora e alguns exemplos com palavras comumente usadas.

Utilizando sentenças simples como as da Figura 11, a ferramenta não tem problema em gerar uma GL aceitável. Mas, ao olhar para a Figura 12 é possível notar um problema de apresentação quando o arquivo .grf gerado é aberto no Unitex. Além de posicionar o nó final muito longe dos outros nós, os demais nós da gramática ficavam ilegíveis, o que prejudicava bastante a experiência do usuário, como representado pela Figura 14.

[3] Biblioteca Apache OpenNLP: https://opennlp.apache.org/
[4] Biblioteca FreeLing: https://nlp.lsi.upc.edu/freeling/node/1
[5] Biblioteca NLTK: https://www.nltk.org/

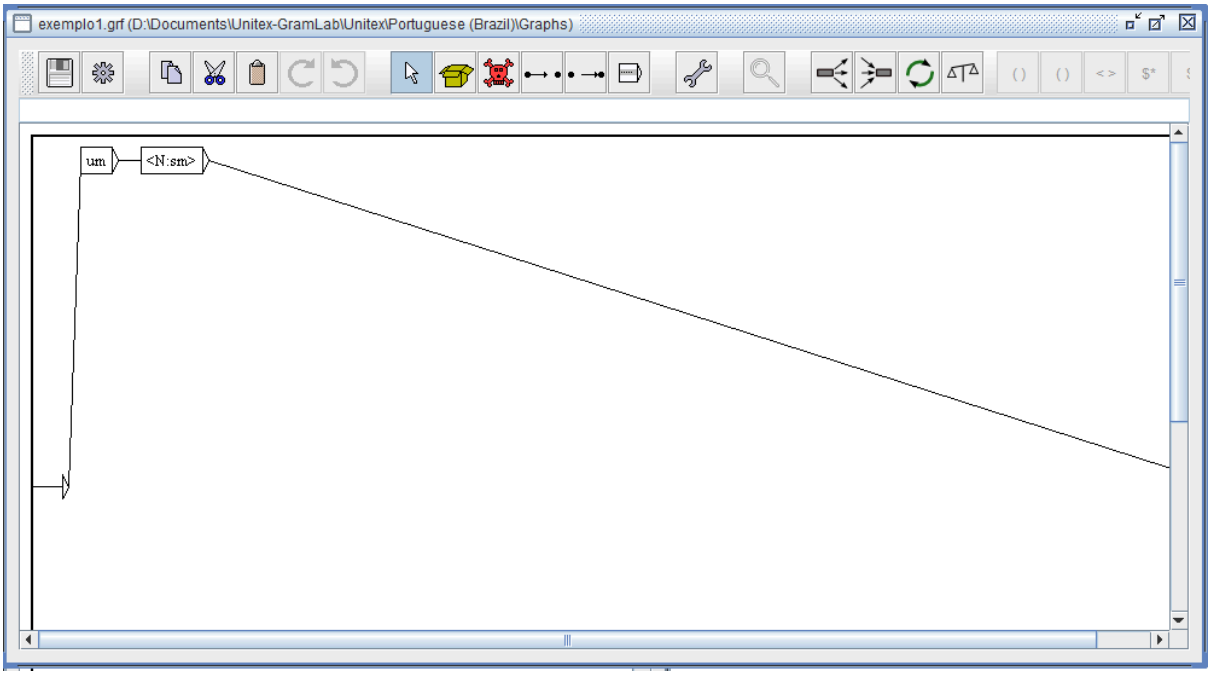

Figura 14: Grafo gerado com nós desalinhados

Fonte: O autor

Analisando outro caso, deseja-se construir uma GL a partir dos exemplos selecionados na Figura 15, no caso, o usuário deseja uma GL que localize frases iniciando com a palavra “o” seguida de um substantivo acompanhado de um conectivo ligando esse substantivo a um adjetivo ou a um verbo. Nos exemplos, as palavras “trabalho” e “jogo” são substantivos, devido ao contexto das frases. A GL foi gerada, salva e aberta no Unitex. Após organizar manualmente os nós, a Figura 16 mostra como a GL ficou. À primeira vista, a GL parece representar bem os exemplos fornecidos, mas observando os nós circulados na Figura 17, é possível notar a presença dos códigos lexicais N (substantivo) e V (verbo). O Gerador de GL não levou o contexto da frase em consideração e atribuiu para as palavras “trabalho” e “jogo” ambos códigos N e V, além disso, o algoritmo não entendeu que o usuário queria especificamente frases iniciadas com a palavra “o”, somente. Esses equívocos provocaram a imprecisão na geração da GL. Por isso, o reconhecimento de contexto dos exemplos é importante para o melhoramento do processo de geração de GL.

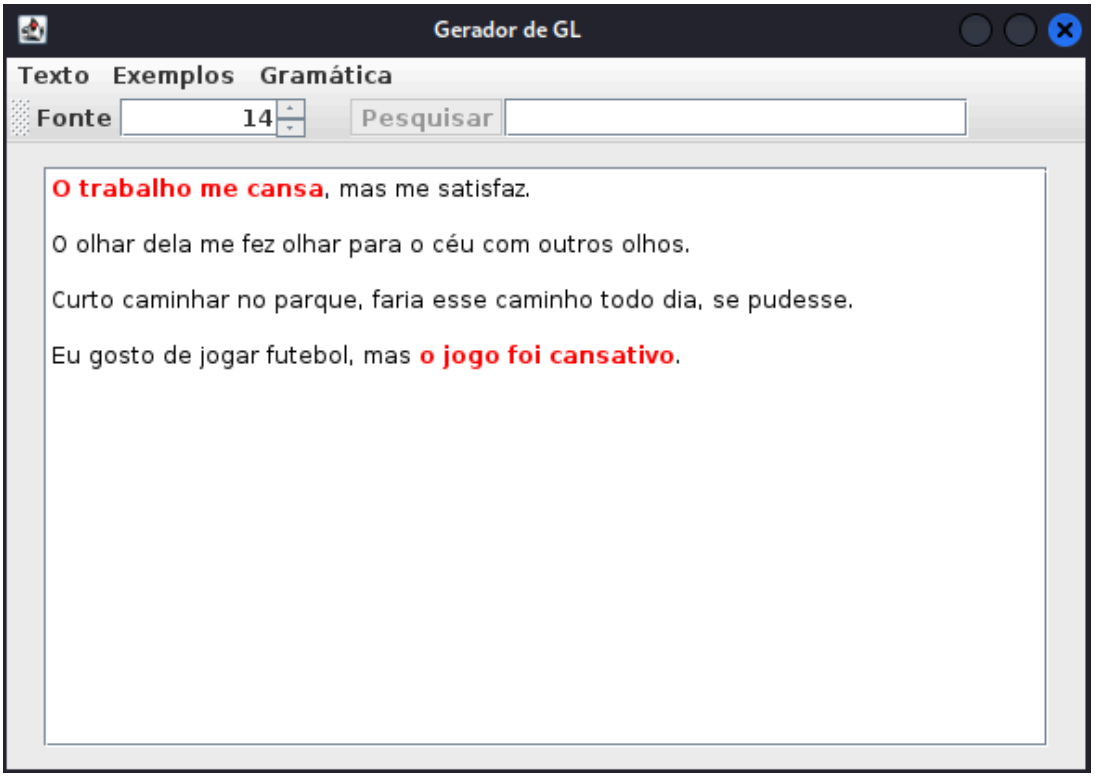

Figura 15: Exemplos de selecionados para construir GL

Fonte: O autor

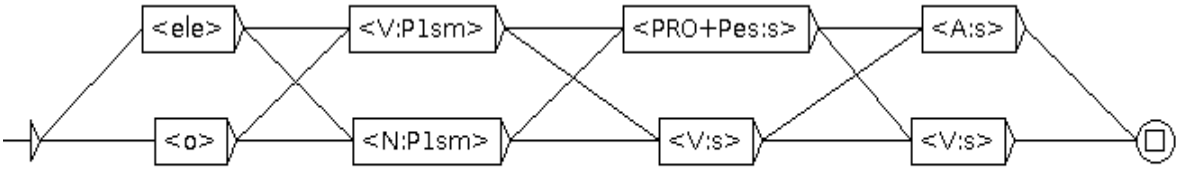

Figura 16: GL gerada com exemplos da Figura 15

Fonte: O autor

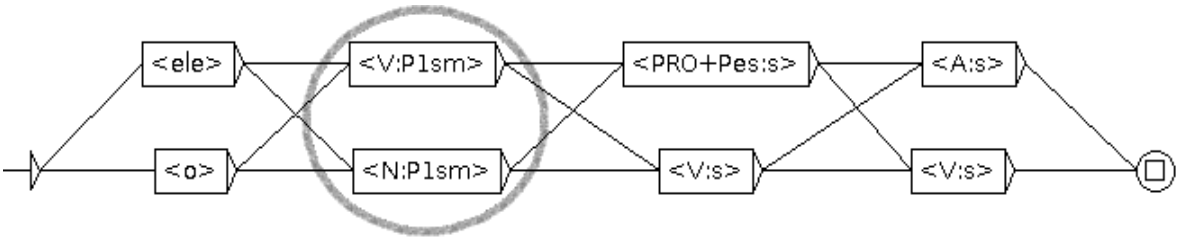

Figura 17: Imprecisão gerada por falta de verificação no contexto

Fonte: O autor

Alguns caracteres não reconhecidos pelo Unitex causavam um erro quando selecionados em um exemplo. O erro fazia a GL não ser gerada e uma mensagem de erro em branco aparecia na tela do Gerador de GL. Após a correção, quando um caractere não reconhecido é selecionado e o usuário decide gerar uma GL, um alerta que informa os caracteres não reconhecidos aparece para o usuário. Esse alerta impede o início do processo de geração da GL e notifica o usuário de caracteres que ele deve retirar de seus exemplos.

Os erros que foram levantados executando testes na ferramenta Gerador de GL foram corrigidos utilizando bibliotecas da própria linguagem *Java*, por exemplo, os alertas adicionados foram implementados utilizando a biblioteca *JOptionPane* do pacote *Javax.Swing*. Para aumentar a precisão do processo de generalização de regras, um serviço com a biblioteca spaCy do Python foi utilizado.

3.2 Integração com spaCy

Para realizar a melhoria que possibilita analisar o contexto dos exemplos informados pelo usuário, e assim, aumentar a precisão do processo de generalização de regras, foi escolhida a ferramenta *spaCy*. Diferente de outras ferramentas, o spaCy possui um funcionamento simplificado, possui modelos pré-treinados e tem uma documentação de fácil entendimento.

A utilização do spaCy é altamente simplificada. Antes de começar a usar as funções que o spaCy disponibiliza, o desenvolvedor deve realizar a instalação da biblioteca e precisa ter o modelo que deseja utilizar carregado no ambiente do Python. Após isso, é preciso importar a biblioteca, com o comando “import spacy”, e deve carregar o modelo que deseja utilizar em uma variável. O modelo escolhido foi o pt_core_news_sm e pode ser carregado na variável “nlp” com o comando “nlp = spacy.load("pt_core_news_sm")”. Para iniciar um pipeline sobre um texto desejado é necessário executar o comando “doc = nlp("um texto qualquer")” e ao final do processamento a variável “doc” irá receber um objeto contendo as *tags* que o modelo foi capaz de atribuir a cada parte do texto informado.

Uma das dificuldades enfrentadas para realizar o aprimoramento do processo de generalização de regras foi a diferença entre as *tags* utilizadas pelo spaCy e pelo Unitex. Analisando as *tags* encontradas nas Figuras 6 e 13 foi possível construir

uma relação entre ambos padrões de *tags*. A relação entre o padrão de *tags* nas duas ferramentas está representada pelo Quadro 1. Vale ressaltar que os códigos lexicais presentes no Quadro 1 que não estão presentes na Figura 6 foram confirmados utilizando o apêndice A do trabalho de Muniz (2004), que é a referência principal para os dicionários do Unitex para o Português Brasileiro.

Algumas *tags* reconhecidas pelo spaCy não são reconhecidas pelos dicionários em português disponíveis no Unitex. Logo, as *tags* SIGL e ABREV, que não constam na Figura 9, não têm relação direta com o spaCy, por isso, não constam na tabela de relação. Da mesma forma, as *tags* PART, PUNCT, SYM e X da Figura 13 também não constam no Quadro 1.

Unitex	**spaCy**	**Classe gramatical**
A	ADJ	Adjetivo
ADV	ADV	Advérbio
CONJC	CCONJ	Conjunção coordenada
CONJS	SCONJ	Conjunção subordinada
DET	DET	Determinante
INTJ	INTJ	Interjeição
N	NOUN	Substantivo
PREP	ADP	Preposição (Adposição)
PRO	PRON	Pronome
V	VERB, AUX	Verbo (Auxiliar)
N+Pr	PROPN	Nome próprio
Num	NUM	Numeral

Quadro 1: Relação entre *POS Tags* do Unitex e spaCy

Fonte: O autor

Após construída a relação entre os padrões de *tags*, foi necessário desenvolver um serviço que executasse uma solução implementada em Python com o spaCy. O serviço foi desenvolvido para receber um texto, iniciar um pipeline de análise sobre esse texto e retornar uma lista contendo as tags das palavras encontradas. Para garantir a precisão das respostas, foi escolhido o modelo

pt_core_news_lg. Com base em alguns testes, foi possível notar que a diferença no tempo de resposta entre esse modelo e o modelo mais rápido, a versão “sm”, era praticamente irrelevante, por isso, o uso desse modelo foi mantido.

Então, foi necessário analisar o funcionamento do processo de generalização de regras para adicionar as respostas do serviço spaCy no fluxo de análises, para conseguir a garantia que o contexto dos exemplos está sendo levado em consideração.

3.3 Processo de generalização de regras

O fluxo do processo de generalização de regras se inicia quando o usuário decide gerar uma GL utilizando exemplos marcados previamente. Após iniciar o processo são aplicadas ferramentas do Unitex para tokenizar os exemplos informados e classificar as palavras encontradas nos exemplos. Os exemplos são enviados para o serviço spaCy, um a um. A resposta retornada pelo serviço é então convertida do padrão de tags do spaCy para o padrão do Unitex. A estrutura utilizada para armazenar as informações contidas no Quadro 1 foi um *HashMap de String* para *String*, disponível em bibliotecas padrão da linguagem *Java*. Então, a seleção de *tags* do spaCy é guardada em um *ArrayList* de *ArrayList* de *String*. Utilizando o *ArrayList* contendo as *tags* corretas, o algoritmo passa por cada palavra de cada exemplo e compara com as tags que o Unitex atribuiu com a *tag* do *ArrayList*, se as *tags* forem iguais, a palavra do exemplo tem sua estrutura alterada para condizer somente com a *tag* encontrada. Caso nenhuma relação seja encontrada para aquela palavra, nada ocorre com a estrutura original das *tags* dela e o processo de generalização continua. Após ter finalizado o fluxo, a GL é gerada e salva no local desejado.

Por exemplo, o usuário selecionou os exemplos presentes na Figura 15, o Gerador de GL utiliza a JNI do Unitex para tokenizar e resgatar para cada palavra as seguintes características: lemas, códigos lexicais e flexões. O Gerador de GL considera internamente cada exemplo selecionado pelo usuário como um episódio e um evento cada palavra de um episódio e suas devidas classificações. Nesse caso, os episódios 1 e 2 foram classificados como na Figura 18. Nas Figuras 18 e 20 cada

caracter “|” significa um “ou” .

		Evento 1	Evento 2	Evento 3	Evento 4
Episódio 1	Literal	O	trabalho	me	cansa
	Lema	o | o | o | ele	trabalho | trabalhar	eu | eu | eu | eu | eu | eu | eu | eu	cansar | cansar
	Código lexical	PRO+Dem | N | DET+Art+Def | PRO+Pes	N | V	PRO+Pes | PRO+Pes | PRO+Pes | PRO+Pes | PRO+Pes | PRO+Pes | PRO+Pes | PRO+Pes	V | V
	Flexão	ms | ms | ms | A3ms	ms | P1s	O1ms | A1ms | D1ms | R1ms | O1fs | A1fs | D1fs | R1fs	P3s | Y2s
Episódio 2	Literal	o	jogo	foi	cansativo
	Lema	o | o | o | ele	jogo | jogar	ser | ir	cansativo
	Código lexical	PRO+Dem | N | DET+Art+Def | PRO+Pes	N | V	V | V	A
	Flexão	ms | ms | ms | A3ms	ms | P1s	J3s | J3s	ms

Figura 18: Episódios classificados utilizando o Unitex

Fonte: O autor

Os episódios, um de cada vez, são enviados para o serviço spaCy que executa o pipeline, classifica as palavras do exemplo e retorna os códigos lexicais para o Gerador de GL. A Figura 19 representa as classificações geradas pelo spaCy e seu código equivalente para o Unitex.

	Evento	Código spaCy	Código Unitex
Episódio 1	O	DET	DET
	trabalho	NOUN	N
	me	PRON	PRO
	cansa	VERB	V
Episódio 2	o	DET	DET
	jogo	NOUN	N
	foi	AUX	V
	cansativo	ADJ	A

Figura 19: Respostas do serviço spaCy

Fonte: O autor

Ao receber as classificações do serviço spaCy, o Gerador de GL utiliza a relação presente no Quadro 1 para construir um *ArrayList* para armazenar os

códigos lexicais do Unitex correspondentes aos códigos lexicais do spaCy. Então, o Gerador de GL utiliza esse *ArrayList* para filtrar as estruturas presentes na Figura 18. A filtragem primeiro checou, para cada evento, quais códigos lexicais contém o código lexical presente no *ArrayList*. Após isso, são salvos nos eventos somente aqueles casos que tiveram relação com o que estava no *ArrayList*. O resultado dessa filtragem está presente na Figura 20.

		Evento 1	Evento 2	Evento 3	Evento 4
Episódio 1	Literal	O	trabalho	me	cansa
	Lema	o	trabalho	eu | eu | eu | eu | eu | eu | eu | eu	cansar | cansar
	Código lexical	DET+Art+Def	N	PRO+Pes | PRO+Pes | PRO+Pes | PRO+Pes | PRO+Pes | PRO+Pes | PRO+Pes | PRO+Pes	V | V
	Flexão	ms	ms	O1ms | A1ms | D1ms | R1ms | O1fs | A1fs | D1fs | R1fs	P3s | Y2s
Episódio 2	Literal	o	jogo	foi	cansativo
	Lema	o	jogo	ser | ir	cansativo
	Código lexical	DET+Art+Def	N	V | V	A
	Flexão	ms	ms	J3s | J3s	ms

Figura 20: Episódios depois da filtragem

Fonte: O autor

Então, o Gerador de GL utiliza os eventos da Figura 20 para realizar o cálculo de frequências para cada evento (repetições no mesmo episódio são contadas somente uma vez). As características com maiores frequências foram inseridas na posição equivalente ao evento pertencente. Em casos de empate ambas as características são incluídas e se ambos episódios tiverem um evento igual, o literal do evento é utilizado. Logo, as posições dos nós da GL resultante foram:

- Posição 1: Todas as características são iguais, então é escolhido o literal “o”;
- Posição 2: Por ter obtido a maior frequência, o código lexical N foi escolhido;
- Posição 3: Como PRO+Pes e V tiveram frequência 1, ambos foram escolhidos;
- Posição 4: Ambos V e A tiveram frequência 1, por isso, ambos foram escolhidos.

A Figura 21 resume o processo descrito anteriormente, o processo tem início após o usuário selecionar os exemplos e clicar para gerar a gramática. Então, os exemplos são classificados pelo spaCy e pelo Unitex, as classificações resultantes são relacionadas utilizando o Quadro 1, então, a contagem de frequência de cada evento de cada exemplo é realizada, por fim, a GL é construída e entregue ao usuário no fim. A Figura 21 foi desenhada utilizando o software Whimsical[6].

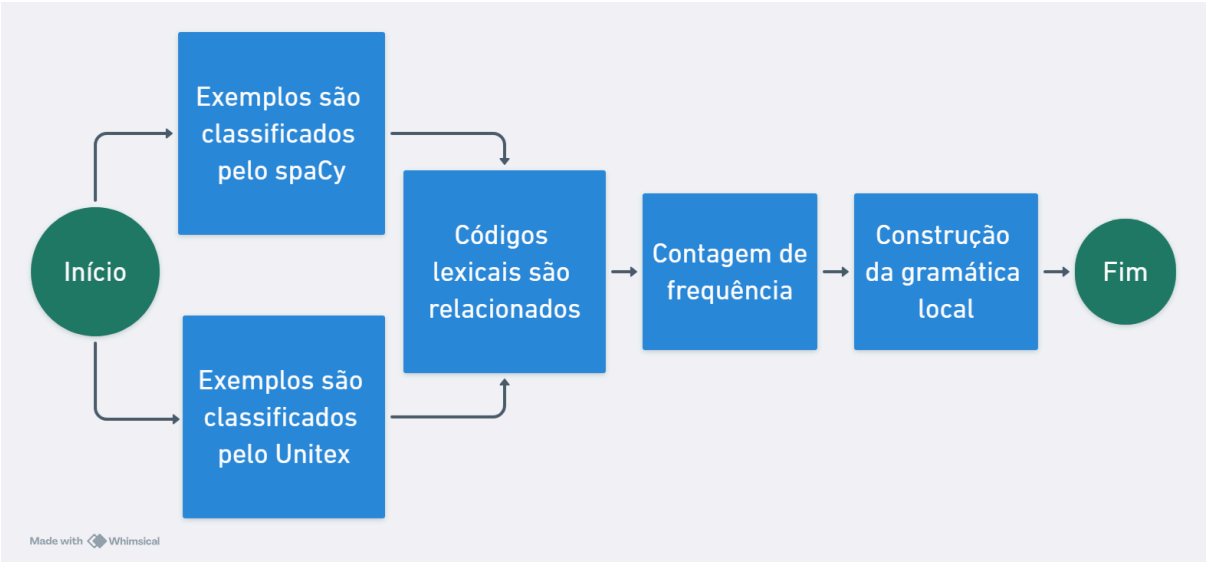

Figura 21: Fluxo de generalização de regras após a inclusão do serviço spaCy

Fonte: O autor

O serviço spaCy retorna somente códigos lexicais, por isso, o processo que escolhe qual flexão será a utilizada ao final da generalização não sofreu alterações. Mas, era necessário a utilização da ferramenta Unitex para aplicar a GL salva sobre o texto carregado no Gerador de GL.

3.4 Aplicação de GL gerada

Neste trabalho também foi implementado um código em *shell script* para automatizar o processo de aplicação de GL. Após salvar uma GL gerada o usuário tem a possibilidade de aplicar a GL gerada no corpo textual aberto no Gerador de GL, sem ter a necessidade de abrir o Unitex e realizar essa aplicação lá.

[6] Ferramenta Whimsical: https://whimsical.com/

4. RESULTADOS

Após o aprimoramento do Gerador de GL, telas de alerta foram adicionadas para impedir erros ocasionados por falhas no reconhecimento de alguma palavra ou símbolo selecionado e por tentativas de gerar uma GL em momento impróprio. Além disso, depois de implementadas as melhorias, foram feitas comparações nas GLs geradas pelo processo de generalização de regras para determinar se a precisão das GLs resultantes realmente havia aumentado por estarem levando o contexto em consideração.

4.1 Aprimoramentos

O aprimoramento no processo de generalização de regras levando o contexto das palavras em consideração aumentou a precisão das GLs geradas pelo Gerador de GL. Antes das alterações no processo, a utilização de exemplos de entrada como os presentes na Figura 22 gerava uma GL como na Figura 23. Observando a Figura 23, os dois nós que tem conexão com o nó de início representam a primeira palavra da frase. Analisando suas tags, N (substantivo) e V (verbo), é possível notar que o Gerador de GL sem reconhecimento de contexto, considera as palavras trabalho, caminho e jogo como substantivos ou verbos. Mas, observando o contexto dos exemplos, é possível notar que em nenhum caso elas estão conjugadas como verbos.

Depois da realização do aprimoramento, utilizando os mesmos exemplos, a GL gerada tem a estrutura presente na Figura 24. Agora, o Gerador de GL conseguiu identificar que as palavras trabalho, caminho e jogo são substantivos (N), devido ao contexto dos exemplos.

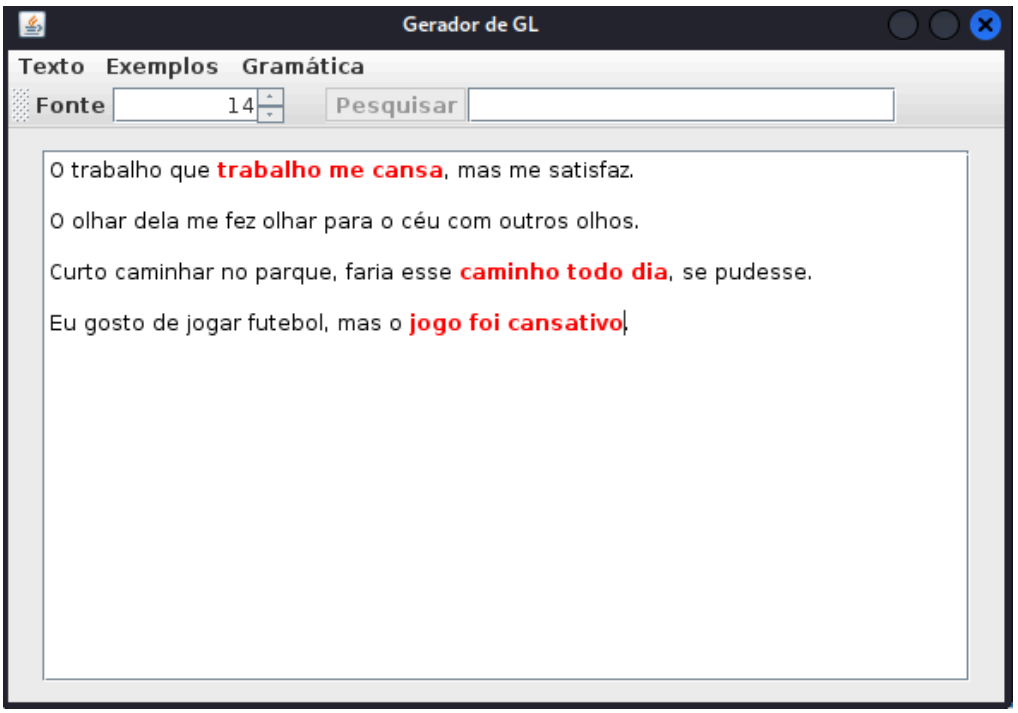

Figura 22: Exemplos quando o contexto importa

Fonte: O autor

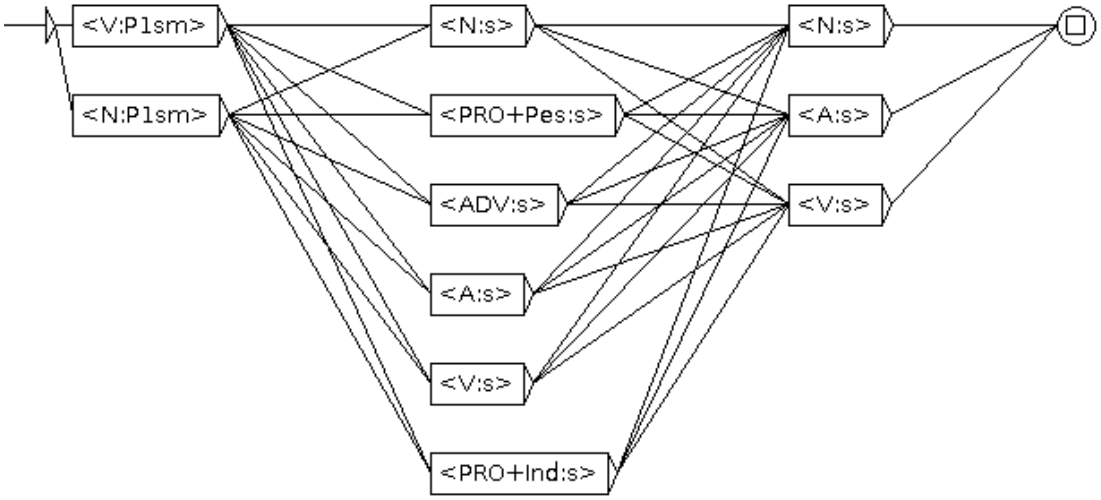

Figura 23: GL gerada sem levar o contexto em consideração

Fonte: O autor

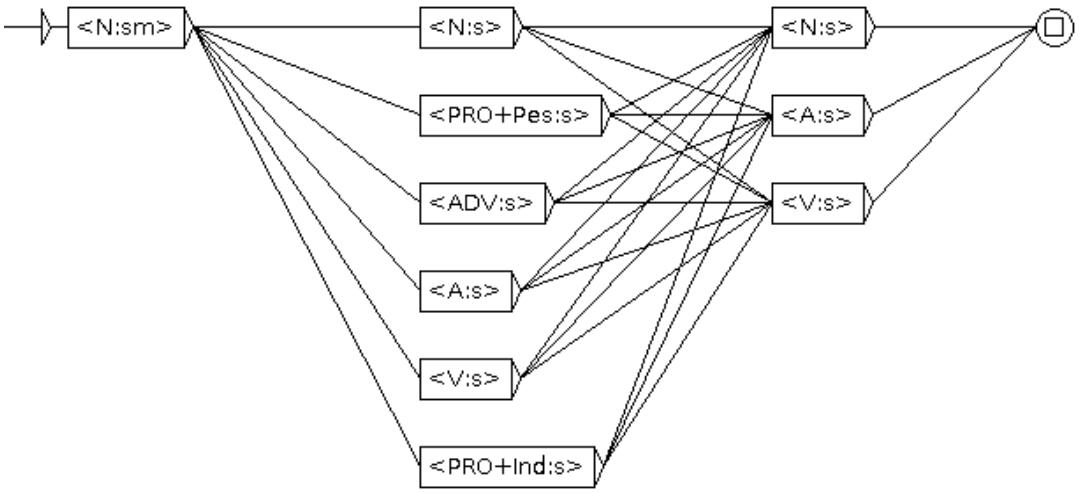

Figura 24: GL gerada levando o contexto em consideração

Fonte: O autor

O outro melhoramento implementado permite que o usuário aplique a GL gerada diretamente no texto aberto no Gerador de GL. Após salvar uma GL gerada, o usuário é apresentado à tela representada pela Figura 25. Ao selecionar a opção Sim, o procedimento para aplicação da GL é executado e uma pasta com o nome do arquivo ".grf" salvo é criada. Dentro dessa pasta existe um arquivo chamado Ocorrências.txt. Esse arquivo contém todas as ocorrências de palavras que a GL utilizada conseguiu localizar. Após realizadas as melhorias, foram realizados tratamentos nos erros encontrados.

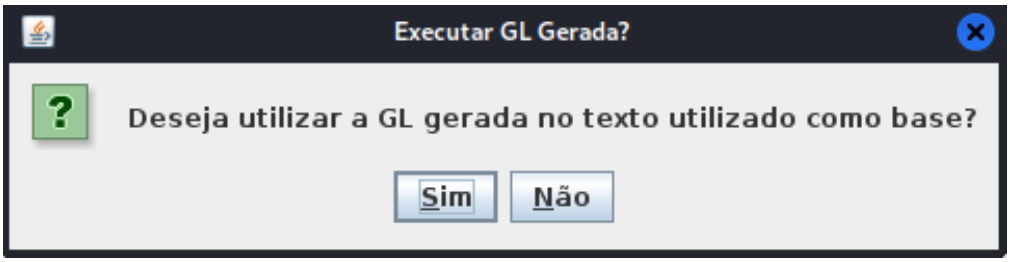

Figura 25: Tela de aplicação da GL no texto aberto

Fonte: O autor

A implementação dos tratamentos foi focada em adicionar telas de alerta para comunicar ao usuário que houve uma falha no procedimento que ele desejava

executar e qual falha foi essa, para que o próprio usuário possa corrigir o problema e executar o procedimento da maneira correta.

Na tentativa de selecionar exemplos que contenham caracteres que o Gerador de GL não reconheça, um erro acontece. A Figura 26 representa um caso onde o usuário tenta gerar uma GL fazendo a seleção de substantivos e números. Quando o usuário tenta gerar a GL, a tela presente na Figura 27 aparece e a GL não é gerada. Porém, após o tratamento do erro, a tela de alerta da Figura 28 aparece para o usuário, informando que os números 3, 2 e 7 não são reconhecidos pelo Gerador de GL. Na versão atual do Unitex os números não estão sendo reconhecidos, mas em versões anteriores os números são reconhecidos pelo código lexical <NB>. Os dicionários de números não estão sendo aplicados de forma correta nas funções da JNI do Unitex, o que ocasiona a falha no reconhecimento de números que não estão por extenso.

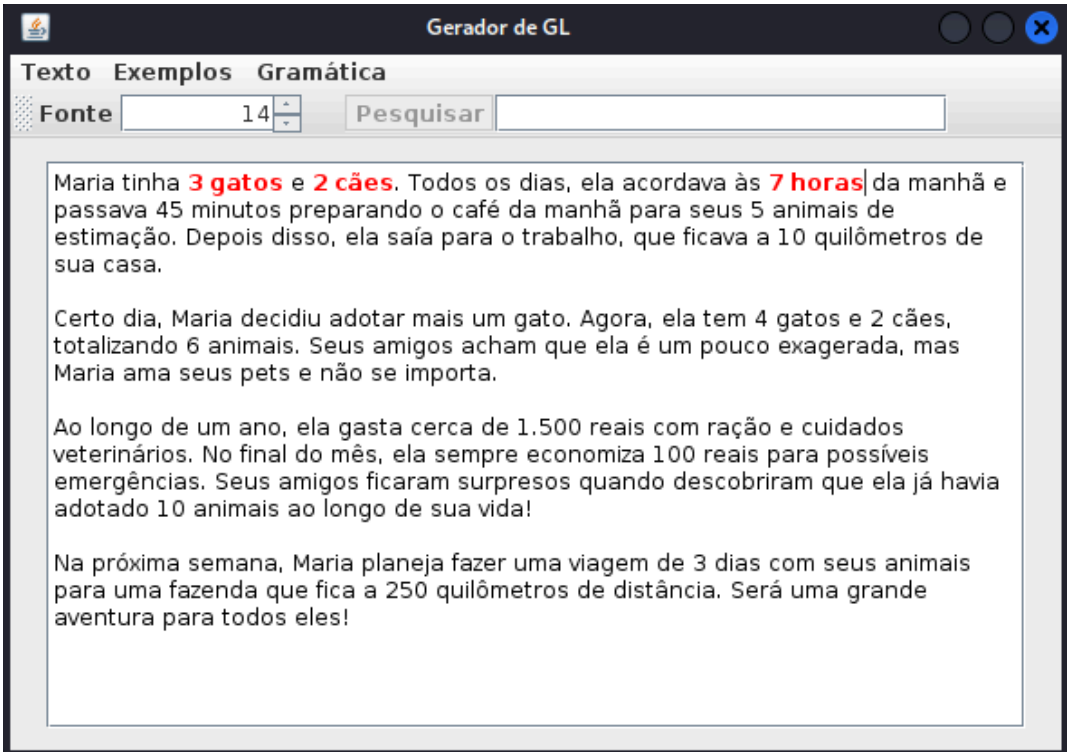

Figura 26: Exemplos de substantivos e números

Fonte: O autor

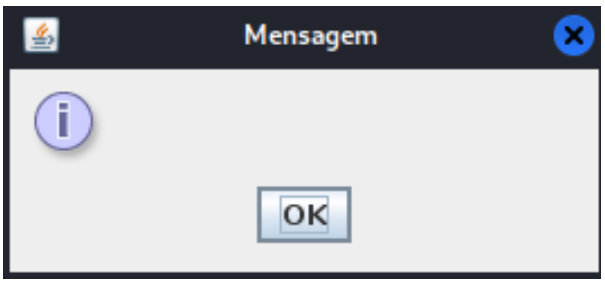

Figura 27: Erro - Selecionar caracteres não reconhecidos

Fonte: O autor

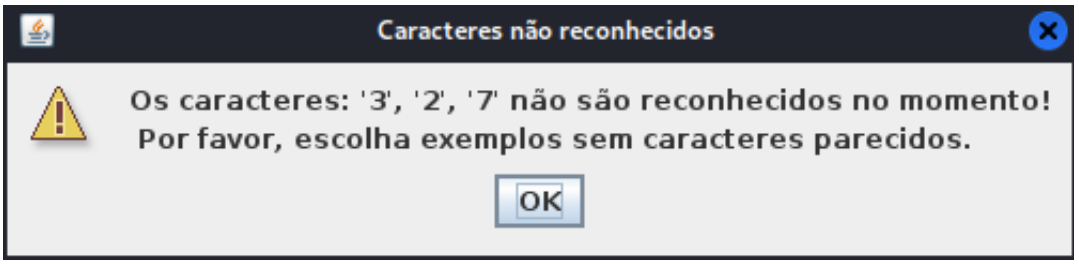

Figura 28: Alerta - Selecionar caracteres não reconhecidos

Fonte: O autor

Além disso, um erro que poderia incomodar o usuário acontecia quando ele tentava gerar uma GL sem ter um texto carregado na ferramenta. A Figura 29 ilustra a mensagem de erro que aparecia para o usuário. Contudo, com o tratamento do erro, a tela de alerta presente na Figura 30 é exibida.

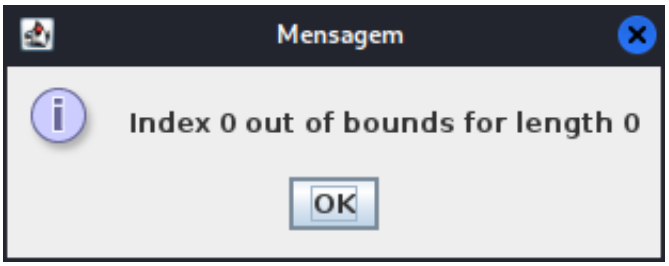

Figura 29: Erro - Gerar GL sem texto carregado

Fonte: O autor

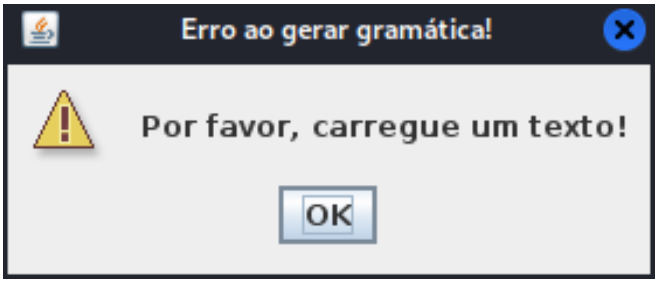

Figura 30: Alerta - Gerar GL sem texto carregado

Fonte: O autor

Um outro problema corrigido, foi um bem frequente que posicionava o nó de fim de uma GL fora da tela do usuário. Esse problema pode ser visto na Figura 14. Após a correção, o nó de fim se posiciona dentro da tela do usuário, levando em conta o número de outros nós presentes no grafo, como representado pela Figura 31.

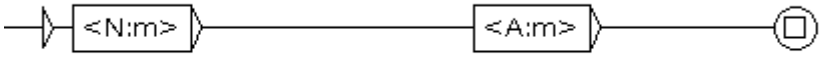

Figura 31: GL gerada com nós posicionados automaticamente

Fonte: O autor

Após as melhorias, correções e tratamentos de erros terem sido realizados, o Gerador de GL ainda apresenta limitações que podem ser resolvidas em trabalhos futuros.

4.2 Limitações

Depois de muitos testes e correções, algumas limitações que foram tratadas somente com telas de alerta persistem. Como a falha no reconhecimento de números, alguns nomes próprios, siglas e abreviações, porque, o processo de atribuição de características para algumas palavras não funciona. O processo não funciona porque o Unitex não reconhece as palavras desses tipos e isso acarreta em objetos vazios (null) na construção da estruturas de eventos, causando exceções no código.

Além disso, a ferramenta só pode ser utilizada em sistemas operacionais das distribuições Linux, por utilizar uma JNI para realizar a tokenização, o processo de

tagging e a geração de GL. Para que a ferramenta seja utilizada no Windows é necessário compilar uma JNI com extensão “.dll” e refatorar trechos do código do Gerador de GL.

5. CONCLUSÃO

Com o PLN, é possível desenvolver ferramentas que possibilitam facilitar as tarefas de busca e análise textual, como a utilização de GLs. As GLs são capazes de facilitar essas tarefas por terem a capacidade de identificar várias frases diferentes, mas gramaticalmente parecidas. Porém, para construir uma GL que realize análises e buscas textuais de qualidade é necessário conhecimento nas áreas de computação e linguística. Para possibilitar que pessoas sem muito conhecimento nessas áreas consigam suas próprias GLs, o Gerador de GL foi desenvolvido. Porém, ele apresentava imprecisão no processo de generalização de regras para geração de GL e erros quando o usuário selecionava palavras ou símbolos não reconhecidos pela ferramenta ou tentava gerar uma GL sem ter carregado um texto.

Este trabalho se propôs a aprimorar as limitações citadas anteriormente e foi capaz de possibilitar a análise em contexto dos exemplos selecionados pelo usuário, tratar os erros relacionados a palavras ou símbolos desconhecidos e tentativas de iniciar o processo de geração de GL em momento indevido, corrigir a forma como os nós da GL estavam localizados quando aberta na ferramenta Unitex e adicionar a possibilidade do usuário aplicar a GL gerada no texto que foi usado para extrair os exemplos, logo após de gerar a GL.

Como trabalhos futuros, possibilitar que o Gerador de GL execute em outros sistemas operacionais é uma necessidade. Aliado a isso, é importante buscar uma forma de aplicar mais dicionários ao Gerador de GL, para possibilitar a análise de números, que não sejam por extenso, siglas e abreviações. Para siglas e abreviações existe a necessidade de expandir os dicionários já existentes. Um outro caminho que pode ser interessante é criar ou encontrar um dicionário que realize a análise de códigos de programação, possibilitando a busca de códigos parecidos em documentos de texto.

REFERÊNCIAS

CHOMSKY, Noam. ***Three models for the description of language***. Massachusetts Institute of Technology. Cambridge, Massachusetts, p. 113-124, 1956.

CHOPRA, Abhimanyu. PRASHAR, Abhinav. SAIN, Chandresh. ***Natural Language Processing***. INTERNATIONAL JOURNAL OF TECHNOLOGY ENHANCEMENTS AND EMERGING ENGINEERING RESEARCH. IJTEEE, v. 1, ed. 4, p. 131-134, ISSN 2347-4289, 2013.

CHOUBEY, N. S.; KHARAT, M. U. ***Grammar induction and genetic algorithms-an overview*.** Pacific Journal of Science and Technology, v. 10, n. 2, p. 884–888, 2009.

EXPLOSION AI. ***spaCy Documentation***. Disponível em: <https://spacy.io>. Acesso em: 13 set. 2024.

FELTRIM, V. D.; SOUZA, V. **Pyger: Uma ferramenta geradora de expressões regulares a partir de um conjunto de expressões em linguagem natural.** In: Proceedings of the 1st Student Workshop on Information and Human Language Technology (TILic–STIL). [S.l.: s.n.], 2009. p. 1–5.

FONSECA, Evandro. *et al*. **Reconhecimento de Entidades Nomeadas para o Portugues Usando o OpenNLP.** Porto Alegre, Brasil. 2015.

GANDES, Thiago. **Geração De Gramáticas Locais A Partir De Exemplos.** Monografia (Graduação em Ciência da Computação) – Universidade Federal do Espírito Santo, Alegre, 2021.

GROSS, Maurice. ***A Bootstrap Method for Constructing Local Grammars*.** Proceedings of the Symposium on Contemporary Mathematics, University of Belgrad, p. 229-250, 1999.

HOWARD, Jeremy; RUDER, Sebastian. ***Fine-Tuned Language Models for Text Classification***. 2018. Disponível em: <https://arxiv.org/abs/1801.06146>. Acesso em: 13 set. 2024.

JURAFSKY, Daniel.; MARTIN, James H. ***Speech and Language Processing:*** *An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition with Language Models.* 3. ed. [S. l.]: Draft, 2023. Disponível em: <https://web.stanford.edu/~jurafsky/slp3/>. Acesso em: 12 set. 2024.

KUMAR, Ela. ***Natural Language Processing***. I.K International Publishing House. 2013.

LIDDY, E.D. ***Natural Language Processing***. Encyclopedia of Library and Information Science, ed. 2. Nova York. Marcel Decker, Inc. 2001.

MANNILA, H.; TOIVONEN, H.; VERKAMO, A. I. ***Discovery of frequent episodes in event sequences*.** Data mining and knowledge discovery, Springer, v. 1, n. 3, p. 259–289, 1997.

MINARRINE, Andrico. **Aprimoramentos Na Ferramenta De Construção Semi-Automática De Gramáticas Locais**. Monografia (Graduação em Ciência da Computação) – Universidade Federal do Espírito Santo, Alegre, 2023.

MUNIZ, Marcelo Caetano Martins. **A construção de recursos lingüisticos-computacionais para o português do Brasil**: o projeto Unitex-PB. 2004. Dissertação (Mestrado) – Universidade de São Paulo, São Carlos, 2004. Acesso em: 06 out. 2024.

NADKARNI, Prakash M. OHNO-MACHADO Lucila. CHAPMAN, Wendy W. ***Natural language processing:*** *an introduction*. Journal of the American Medical Informatics Association. 2011.

PAUMIER, Sébastien. ***Unitex 3.3:*** *User Manual*. Université Paris-Est Marne-la-Vallée, 2021. Disponível em: <https://unitexgramlab.org/releases/3.3/man/Unitex-GramLab-3.3-usermanual-en.pdf >. Acesso em 13 de Setembro de 2024.

PICOLI, Larissa; PIROVANI, Juliana Campos; OLIVEIRA, Elias de; LAPORTE, Eric. **Uso de uma Ferramenta de Processamento de Linguagem Natural como Auxílio à Coleta de Exemplos para o Estudo de Propriedades Sintático-Semânticas de Verbos**. *LinguaMÁTICA*, v. 7, n. 2, p. 35-44, 2015.

PILEHVAR, Mohammad Taher; CAMACHO-COLLADOS, José. ***Embeddings in Natural Language Processing:*** *Theory and Advances in Vector Representation of Meaning*. Morgan & Claypool Publishers, 2020.

RINO, Lucia. PARDO, Thiago. **A Sumarização Automática de Textos: Principais Características e Metodologias.** São Paulo, São Carlos, 2003.

VASILIEV, Yuli. **Natural Language Processing with Python and spaCy**. No Starch Press, 2020.

ZHONG, Z. *et al*. ***Generating regular expressions from natural language specifications:*** *Are we there yet?* In: Workshops at the Thirty-Second AAAI Conference on Artificial Intelligence. [S.l.: s.n.], 2018.