XIX Encontro Nacional de Tecnologia do Ambiente ‘
ENTAC 2022

Ambiente Construído: Resiliente e Sustentável
Canela, Brasil, 9 a 11 novembro de 2022

Processamento de Linguagem Natural (PLN) para automatização da checagem de conformidade: uma investigação do pré-processamento de um código regulatório urbanístico brasileiro

Natural language processing (NLP) for automated compliance checking: an investigation of the preprocessing of a Brazilian urban regulatory code

Paulo Victor Matos Leite de Ávila
Universidade Federal da Bahia | Salvador | Brasil | paulo.avila@ufba.com.br
Douglas Malheiro de Brito
Universidade Federal da Bahia | Salvador | Brasil | douglas_ssa@hotmail.com
Daniele Mota Santos
Universidade Federal da Bahia | Salvador | Brasil | danimosassa@gmail.com
Emerson de Andrade Marques Ferreira
Universidade Federal da Bahia | Salvador | Brasil | ferreira.eam@gmail.com

Resumo

A checagem manual de conformidade de projetos é uma tarefa custosa e sujeita a erros. Os parâmetros contidos nos códigos regulatórios podem ser automaticamente extraídos usando o Processamento de Linguagem Natural (PLN), tornando a checagem mais eficiente e segura. Este estudo investiga uma rotina utilizando técnicas de PLN para o pré-processamento – primeira etapa para extração de informação - de um código regulatório urbanístico brasileiro. Utilizou-se no experimento a linguagem de programação Python e a biblioteca Natural Language Tool Kit (NLTK). Obteve-se uma acurácia de 68% no desempenho do etiquetador, indicando a necessidade de aprimoramentos no pré-processamento para a língua portuguesa.

Palavras-chave: Processamento de Linguagem Natural. Automatização. Pré-Processamento. Inteligência Artificial. Código Urbanístico.

Como citar:
ÁVILA, P.V.M.L.; BRITO, D.M.; SANTOS, D.M.; FERREIRA, E.A.M. Processamento de Linguagem Natural (PLN) para automatização da checagem de conformidade: uma investigação do pré-processamento de um código regulatório urbanístico brasileiro. In: ENCONTRO NACIONAL DE TECNOLOGIA DO AMBIENTE CONSTRUÍDO, 19., 2022, Canela. **Anais**... Porto Alegre: ANTAC, 2022. p. 1-12.

Abstract

Manually checking for compliance is a resource-intensive and error-prone task. Information in regulatory codes can be extracted automatically using natural language processing (NLP) techniques, making compliance checking simpler and more reliable. This work investigates a script using NLP techniques for the pre-processing – first phase of information extraction - of a Brazilian regulatory code. For this, the Python programming language and the NLTK library were used. An accuracy of 68% was achieved the performance of the labeller, indicating the need for improvements in the pre-processing for the Portuguese language.

Keywords: Natural Language Processing. Automation. Pre-processing. Artificial intelligence. Urban code.

1. INTRODUÇÃO

A checagem de códigos é fundamental para empreendimentos mais seguros e eficientes, pois não conformidades podem ter efeitos catastróficos. Esses códigos impõem limites aceitáveis para garantir a segurança dos usuários [1]. Contudo, os processos de checagem atuais costumam ser manuais, demorados, caros e propensos a erros [2].

Os códigos geralmente têm um propósito natural de organizar, classificar, rotular e definir as regras, eventos e padrões a serem seguidos pelos empreendimentos para segurança, eficiência e economia. A informatização das regras e disposições relativas à checagem desses códigos tem interessado muitos pesquisadores e profissionais desde 1960 [3].

Desta forma, como a maioria dos códigos não foi desenvolvida em formato legível por computador, a conversão é complicada, não só do ponto de vista técnico, mas também do ponto de vista jurídico [4]. Esse processo de conversão, geralmente, depende de um desenvolvedor de software interpretando e traduzindo os regulamentos em uma linguagem de programação. A lógica das disposições do código escrito é formalmente interpretada e depois codificada em instruções de computador [5].

Com os códigos sendo modelados por programadores, mal-entendidos e dificuldades de entender o significado original podem causar erros. Outro desafio está na subjetividade, cujo significado nem sempre pode ser compreendido sem a necessidade do julgamento humano [6]. Devido à alta complexidade e terminologia específica do domínio, é difícil garantir a qualidade e consistência das traduções codificadas por humanos. Como os códigos regulatórios são alterados com frequência, é uma tarefa complexa manter uma versão digital atualizada, principalmente sem uma conexão direta com o texto original. Como resultado, a tradução manual desses códigos, cada um contendo centenas de regras, é onerosa e demorada [1].

Portanto, os sistemas de checagem automatizada exigem uma quantidade significativa de trabalho para os desenvolvedores traduzirem a linguagem humana para uma linguagem reconhecida por computador. Contudo, vale destacar os avanços que têm ocorrido na automatização da extração e transformação de regras diretamente do código [7]. Os métodos de Inteligência Artificial (IA) usam algoritmos de Processamento de Linguagem Natural (PLN), como análise de texto, monetização de conteúdo, classificação automática de conteúdo e mineração de texto. Esses métodos

visam permitir que os computadores obtenham automaticamente o significado do texto e gerem regras lógicas para fins de verificação [8].

Fuchs e Amor [1] definem o PLN como um campo da IA que visa processar computacionalmente e entender a linguagem humana. Nesse contexto, os recentes desenvolvimentos em PLN constituem uma solução promissora para automatizar a checagem, mas casos práticos devidamente testados ainda são pouco explorados na literatura.

Além disso, o pré-processamento é uma etapa fundamental do PLN, na qual o texto é preparado para a extração automatizada de informações. Nesta fase, o texto em linguagem natural é segmentado e representado em partes que podem ser mais facilmente interpretadas pelo computador [9].

Embora o PLN constitua uma área de estudo em crescimento, ainda há uma quantidade relativamente pequena de trabalhos que o investiga no contexto da língua portuguesa. Esse número é ainda mais reduzido quando se leva em conta apenas os trabalhos que utilizam o PLN com o objetivo de extrair e transformar a informação de códigos edilícios brasileiros. Com base no exposto, o objetivo deste estudo é realizar uma investigação preliminar de um conjunto de algoritmos utilizando PLN para o pré-processamento de onze cláusulas de um código regulatório urbanístico brasileiro da cidade de Salvador, visando automatizar a extração e transformação de informações.

2. PROCESSAMENTO DE LINGUAGEM NATURAL PARA AUTOMATIZAÇÃO DE CHECAGENS

Dentre as várias atividades compreendidas pelo PLN, há tarefas de suporte de menor nível, como a tokenização ou etiquetagem de frases, o *Part-of-Speech-tagger* (*POS-tagger*), que pode ser traduzido como marcação de partes da fala, e análise de dependência, além de tarefas de alto nível, como classificação de texto, extração de informações, resposta a perguntas e tradução automática [1]

O processamento de documentos refere-se à análise dos códigos para executar ações como desifenização, remoção de quebras de linha, notas de rodapé e divisão do documento em seções [1]. Contudo, antes de qualquer tarefa, um dos primeiros passos do PLN é o pré-processamento, que prepara o texto de entrada para modelos ou algoritmos possam ser utilizados [1].

Em relação aos trabalhos brasileiros que exploraram a temática, Nieves *et al.* [10] testaram o uso do PLN em etapas iniciais do pré-processamento dos códigos a serem transcritos de forma automática. Os referidos autores adotaram trechos da norma brasileira ABNT NBR 9077/2001, referente às saídas de emergência em edifícios, e aplicaram quatro técnicas: desifenização, tokenização, separação de frases e análise morfológica [10].

Nieves *et al.* [10] considerou os resultados da aplicação satisfatórios, porém, precisam ser cuidadosamente avaliados e certificados para que possam alimentar os processos subsequentes de PLN, como a extração de regras.

Barbosa *et al.* [11] investigou o uso do PLN em língua portuguesa com o uso da linguagem de programação Python e da biblioteca *Natural Language Tool Kit* (NLTK), plataforma específica para trabalhar com dados em linguagem natural. Os autores aplicaram técnicas como POS-tagging, tokenização e chunking por meio de experimentos e discutiram diferentes aplicações, como análise de sentimento e sumarização automática de textos [11].

Rodríguez e Bezerra [12] demonstram como o PLN pode ser usado para automatizar o reconhecimento de entidades nomeadas em documentos públicos. O NLTK foi usado para tokenizar documentos, reconhecer padrões no texto e criar modelos de classificação que possam identificar nomes próprios dentro de um conjunto de dados. Os referidos autores relataram uma acurácia de 92% na extração dos nomes de pessoas, resultado que foi considerado satisfatório para o objetivo proposto [12].

Automatizar a Extração de Informações (EI) é um passo importante para realizar a verificação automática da conformidade com diferentes códigos e normas [9]. O EI é um subcampo do PLN que busca extrair informações de fontes textuais compatíveis com modelos de informação pré-definidos. A EI pode ser baseada nas características sintáticas, semânticas, ou em ambas [13].

Uma técnica comum para agrupar e recuperar informações é avaliar a similaridade do texto com base em raízes de palavras ou representações vetoriais. A análise semântica e sintática de texto pode ser usada para determinar as características linguísticas de um texto. Recursos estruturais, como *POS-tagger*, eram comumente usados para extração de informações baseada em regras [1]. Além disso, existem poucos trabalhos em língua portuguesa que possuem uma avaliação quantitativa do desempenho da etiquetagem de pré-processamento utilizando *POS-tagger*.

A maioria das abordagens de PLN não conseguem lidar com toda a complexidade dos regulamentos, a exemplo de restrições, conjunções, exceções, listas e referências cruzadas. A escassez de dados para treinamento, a falta de conjuntos de dados abertos e o desacordo sobre uma representação adequada para os regulamentos de construção são obstáculos para uma melhoria mais rápida [1].

3. MÉTODO DE PESQUISA

As principais etapas desta pesquisa foram agrupadas em seis fases: revisão da literatura, seleção do código a ser analisado, planejamento do experimento, execução do fluxo de rotinas de pré-processamento, desenvolvimento da avaliação em *gold standard* e discussão dos resultados.

Na primeira etapa, foi realizada uma revisão da literatura sobre automatização da checagem de códigos, o PLN, extração de informações e *POS-tagging*, para identificar o problema de pesquisa e as existentes na literatura.

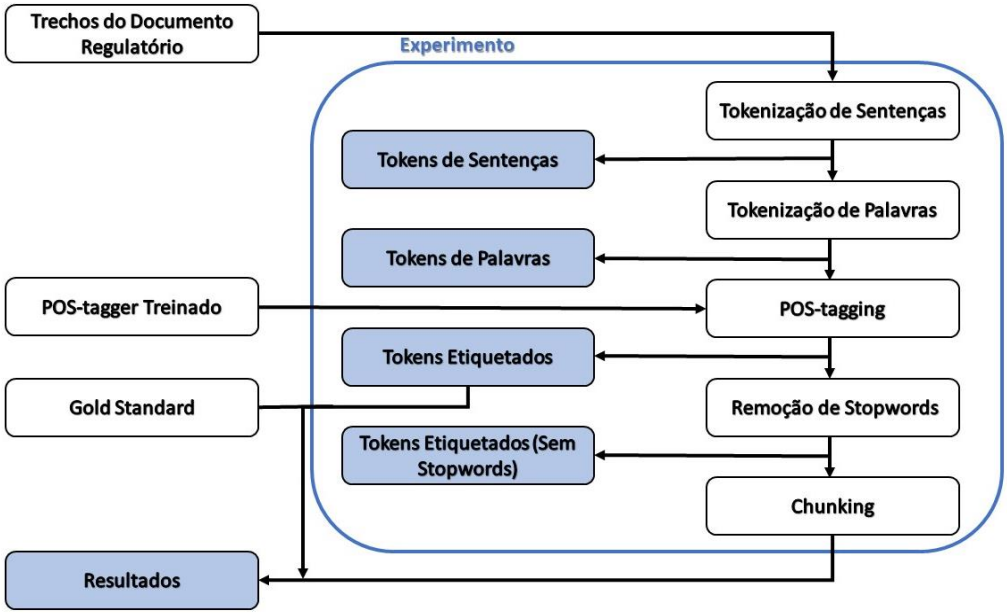
Em seguida, o código regulatório escolhido foi a Lei de Ordenamento do Uso e Ocupação do Solo (LOUOS), atualizada por meio da Lei 9.148/2016. Este documento regulamenta diferentes parâmetros da ocupação do solo na cidade de Salvador, uma cidade brasileira com aproximadamente três milhões de habitantes. Dentre as

motivações para sua escolha, estão o fato de ser um código brasileiro escrito em português - língua pouco explorada em outros estudos sobre o assunto, de não ter sido elaborado em formato legível por computador e de ser de uma cidade brasileira que está em processo de digitalização do licenciamento de obras para que as checagens sejam automatizadas.

Após a seleção do código, foi necessário escolher as suas cláusulas para a análise e planejamento do experimento. Para garantir a aplicabilidade e condições de contorno, os trechos selecionados deveriam ser concisos, não referenciar outras cláusulas ou anexos do código, e expressar uma relação clara de permissão ou restrição, o que reduziria ambiguidades e a necessidade de interpretação humana. Diante disso, onze cláusulas do regulamento foram selecionadas.

Em relação ao planejamento do experimento, a plataforma computacional escolhida foi o Python, distribuída gratuitamente e amplamente utilizada por pesquisadores e profissionais da área de computação, com ampla disponibilidade de bibliotecas de funções e métodos livres [9]. A versão do Python utilizada foi a 3.9, distribuída na plataforma Anaconda. O código foi construído no editor de código Spyder, também disponível na plataforma Anaconda. A biblioteca utilizada foi a NLTK, que possui funções específicas para o uso da PNL, inclusive no idioma português [14]. Utilizando recursos do NLTK, um tagger foi treinado para melhorar a precisão da etiquetagem. A sequência de execução do conjunto de algoritmos de pré-processamento utilizado é detalhada na Figura 1.

Figura 1: Sequência de execução dos algoritmos de pré-processamento



Fonte: Os autores.

No início, as cláusulas do código brasileiro foram extraídas manualmente do documento regulatório em formato de arquivo PDF e salvas no formato TXT, o que permitiu que elas fossem abertas como string pelo algoritmo Python.

A primeira etapa do pré-processamento foi a tokenização de sentenças, uma divisão do texto em sentenças para que a classificação pudesse ser feita com mais precisão. O passo seguinte consiste em segmentar as strings resultantes da tokenização de sentenças em unidades de texto. Essas unidades podem ser palavras, números, sinais de pontuação, símbolos, entre outros.

Posteriormente, foram marcados os tokens de palavras com um *POS-tagger*, que são as etiquetas morfossintáticas atribuídas às palavras para indicar sua função léxica [15]. As tags foram atribuídas depois que o texto foi tokenizado, e podem incluir tokens que não sejam palavras, como números e símbolos.

Para avaliar a acurácia do tagger, foi criado um gold standard, ao qual o *POS-tagger* foi comparado. O *gold standard* consiste no conjunto de trechos selecionados da LOUOS usados no experimento e suas respectivas tags, que foram posteriormente checadas manualmente por dois avaliadores. A abordagem para o desenvolvimento do *gold standard* foi baseada em [16], e está dividida em três etapas: etiquetagem dos tokens pelo *POS-tagger*; avaliação do resultado pelo primeiro avaliador, um discente de graduação em Letras na Universidade Federal do Rio de Janeiro (UFRJ), e por último, por nova avaliação por uma pós-graduada em letras da Universidade Federal da Bahia (UFBA). Durante as avaliações, as tags que foram consideradas pelos avaliadores como incorretamente atribuídas foram sinalizadas, e as tags corretas explicitadas.

Dentre as principais vantagens da abordagem proposta para o desenvolvimento do *POS-tagger*, destacam-se: maior rapidez no processo de construção do *gold standard*, já que os avaliadores possuem um ponto de partida na forma das etiquetas atribuídas pelo *POS-tagger*; uma redução na carga de trabalho dos avaliadores, diminuindo a possibilidade de fadiga e a chance de erros; possibilidade dos avaliadores perceberem padrões nos erros cometidos, à medida que o *gold standard* é construído, o que pode facilitar insights para a melhorar a acurácia do modelo testado.

O conjunto de etiquetas da língua portuguesa apresentado na Tabela 1 foi usado para a etiquetagem do corpus *MacMorpho* [17]. A etapa de remoção de *stopwords* consiste em remover palavras frequentes, mas que muitas vezes não agregam muito significado ao texto, simplificando a linguagem a ser processada e aumentando a precisão da análise. Uma lista de palavras consideradas de menor importância em português presentes no NLTK foi utilizada durante o experimento, como as preposições: “a”, “de” e “em”.

Posteriormente, a técnica de *chunking* foi utilizada para agrupar sequências de palavras que seguiam um determinado padrão em um mesmo conjunto que possuía uma relação semântica. Os padrões criados para a busca são chamados de *chunking grammar*, compostos por sequências de *tags POS* na ordem que se deseja encontrá-las no texto. Assim, os conjuntos de palavras retornadas são os trechos do texto que têm uma correspondência das suas tags com os padrões da gramática de agrupamento *chunking grammar*.

Por fim, na etapa de discussão dos resultados, o experimento foi analisado considerando os demais estudos relatados na literatura, identificando as semelhanças e particularidades ao se utilizar um código brasileiro em português.

Tabela 1: Tags *MacMorpho* adotadas

PARTE da fala - PORTUGUÊS	tag
Adjetivo	ADJ
Advérbio Conectivo Subordinativo	ADV-KS
Advérbio Relativo Subordinativo	ADV-KS-REL
Artigo (Def. Ou Indef.)	ART
Conjunção Coordenativa	KC
Conjunção Subordinativa	KS
Interjeição	IN
Substantivo	N
Substantivo Próprio	NPROP
Numeral	NUM
Particípio	PCP
Palavra Denotativa	PDEN
Preposição	PREP
Pronome Adjetivo	PROADJ
Pronome Conectivo Subordinativo	PRO-KS
Pronome Pessoal	PROPESS
Pronome Relativo Conectivo Subordinativo	PRO-KS-REL
Pronome substantivo	PROSUB
Verbo	V
Verbo Auxiliar	VAUX
Símbolo de Moeda Corrente	CUR
Contração e Ênclise	|+

Fonte: Os autores.

4. APLICAÇÃO DA ETAPA DE PRÉ-PROCESSAMENTO DO PLN NO CÓDIGO REGULATÓRIO BRASILEIRO

Utilizando a metodologia descrita, os artigos 8-I, 54-I, 55-I, 57, 61-II, 61-IV, 64-I, 75-II, 75-III, 84-I e 142-§3º do código da LOUOS do município brasileiro de Salvador foram selecionados para realizar o experimento de pré-processamento com PLN.

Antes da construção do algoritmo de pré-processamento, um tagger foi treinado usando o corpus etiquetado *Mac-Morpho* marcado da biblioteca NLTK [11] [14].

Para aumentar a precisão das tags, vários taggers são combinados utilizando o algoritmo backoff. O *tagger unigram* atribui a tag mais provável a um determinado token, sem considerar os tokens ao seu redor. Isso pode levar a erros na identificação de palavras como “uso”, que podem ser um verbo ou um substantivo, por exemplo. Para reduzir a chance de erros, *bigrams* e *trigrams* foram utilizados.

O contexto para um *tagger n-gram* é a palavra que está sendo analisada juntamente com os tokens n - 1 anteriores [11]. Assim, *bigrams* e *trigrams* podem analisar o contexto de cada palavra com base no treinamento com o corpus etiquetado, o que torna a classificação mais precisa, conforme mostrado na Figura 2.

Figura 2: Processo de treinamento do tagger

```
import nltk
tagged sents = nltk.corpus.mac_morpho.tagged_sents()
t0 = nltk.DefaultTagger('N')
t1 = nltk.UnigramTagger(tagged_sents, backoff=t0)
t2 = nltk.BigramTagger(tagged_sents, backoff=t1)
t3 = nltk.TrigramTagger(tagged_sents, backoff=t2)
```

Fonte: Os autores.

Depois que o tagger for treinado, ele será salvo para que possa ser usado sem que seja necessário treiná-lo novamente. Uma vez que as bibliotecas necessárias foram importadas, o passo seguinte para desenvolver o algoritmo foi abrir o tagger treinado, assim como o arquivo de texto com o trecho selecionado do código.

Em seguida, ocorreu a tokenização de sentenças. Como a cláusula exemplificada na Figura 3 consiste somente em uma sentença, um único token de sentença foi criado. Após a tokenização de sentenças, a lista resultante passa pela tokenização de palavras, conforme também pode ser visto na Figura 3.

Figura 3: Exemplo de tokens de palavras de uma cláusula pré-processada

```
wordTokens = nltk.word_tokenize(texto, language="portuguese")

>>>['II', '-', 'o', 'percentual', 'a', 'ser', 'reservado', 'para', 'as', 'áreas',
'institucionais', 'deverá ', 'atender', 'a', ',', 'não', 'mínimo', ',', '5', '%',
'(', 'cinco', 'por', 'cento', ')', 'da', 'área', 'total', 'fazer', 'terreno',
'.']
```

Fonte: Os autores.

Uma vez que os tokens foram criados, é possível classificar morfossintaticamente os tokens na lista resultante, de acordo com corpus *Mac-Morpho* adotado. O tagger analisa o token corrente e os tokens anteriores para atribuir uma etiqueta para cada palavra, de acordo com sua função morfossintática na sentença, como pode ser visto na Figura 4.

O próximo passo foi a remoção de *stopwords*. Nessa etapa, foram retiradas as seguintes palavras que não agregam significado relevante ao texto, com base em uma lista de *stopwords* em português: “o”, “a”, “para”, “as”, “no”, “da” e “do”, em sua maioria, preposições. A última etapa do experimento foi o *chunking*, no qual através da definição de uma gramática, os tokens classificados como substantivos foram indicados com a palavra “NOME”, conforme mostra a Figura 5.

Figura 4: Exemplo de tokens etiquetados de uma cláusula pré-processada

```
pos = tagger.tag(wordTokens)

>>>[('II', 'NPROP'), ('-', '-'), ('o', 'ART'), ('percentual', 'N'), ('a',
'PREP|+'), ('ser', 'VAUX'), ('reservado', 'PCP'), ('para', 'PREP'), ('as',
'ART'), ('áreas', 'N'), ('institucionais', 'ADJ'), ('deverá', 'VAUX'),
('atender', 'V'), ('a', 'ART'), (',', ','), ('no', 'KC'), ('mínimo', 'N'), (',',
','), ('5', 'NUM'), ('%', 'N'), ('(', '('), ('cinco', 'NUM'), ('por', 'PREP'),
('cento', 'N'), (')', ')'), ('da', 'NPROP'), ('área', 'N'), ('total', 'ADJ'),
('do', 'KS'), ('terreno', 'N'), ('.', '.')]
```

Fonte: Os autores.

Figura 5: Tokens de substantivos assinalados

```
gramatica = r"""NOME: {<N>+}"""
analiseGramatical = nltk.RegexpParser(gramatica)
chunk = analiseGramatical.parse(stopWords)

>>>(S
II/NPROP
  -/-
  (NOME percentual/N)
  ser/VAUX
  reservado/PCP
  (NOME áreas/N)
  institucionais/ADJ
  deverá/VAUX
  atender/V
  ,/,
  (NOME mínimo/N)
  ,/,
  5/NUM
  (NOME %/N)
  (/(
  cinco/NUM
  (NOME cento/N)
  )/)
  (NOME área/N)
  total/ADJ
  (NOME terreno/N)
  ./.)
```

Fonte: Os autores.

5. DISCUSSÃO DOS RESULTADOS E AVALIAÇÃO POR MEIO DO GOLDEN STANDARD

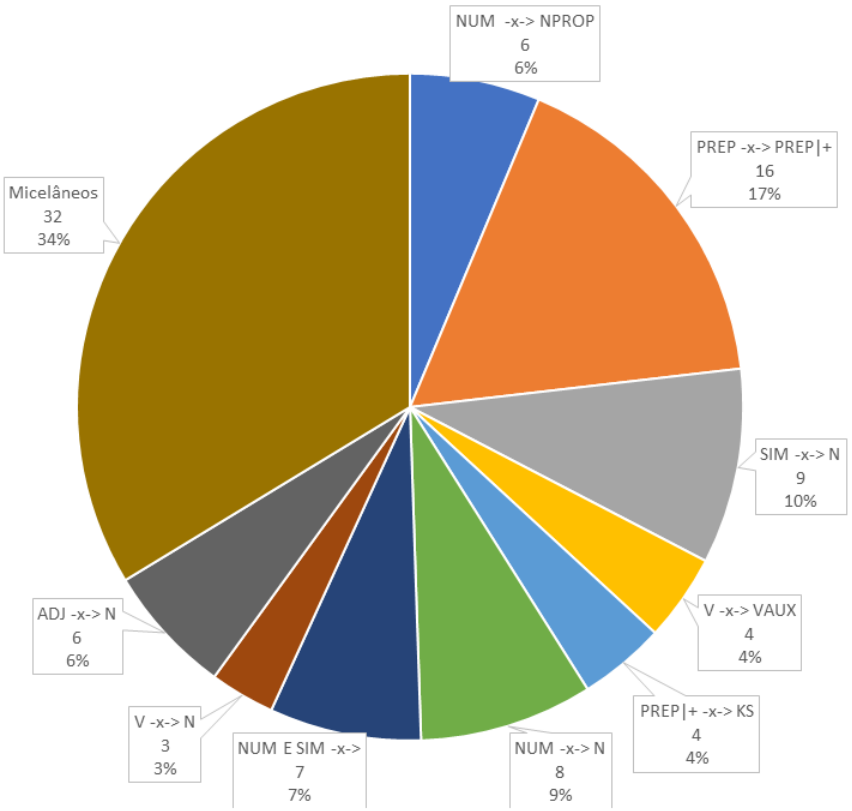
Os resultados obtidos mostram que existe um caminho relativamente claro para a implementação de algoritmos de pré-processamento, mas que ainda demanda refinamento. Em especial, os resultados indicaram a necessidade de uma maior atenção ao *POS-tagger*, já que tokens classificados erroneamente são uma causa comum de erros durante a EI baseada em regras [13] [15].

Um total de 300 tokens entre os 11 trechos selecionados foram etiquetados pelo *POS-tagger*. A comparação com o *gold standard* revelou que 205 tokens foram etiquetados corretamente, o que representa uma acurácia de, aproximadamente, 68%. Dentre as

95 classificações, o erro mais comum - 16 ocorrências - foi atribuição errônea da tag 'PREP|+' (preposição e contração) em tokens considerados como 'PREP' (preposição) Outros erros comuns incluem a classificação de numerais romanos como substantivos próprios, e a classificação de símbolos como substantivos, como ilustrado na Figura 6.

Como exemplo, no experimento realizado, percebeu-se que o tagger classificou o token "da" como um substantivo próprio. Embora esse token tenha sido removido na etapa de remoção de *stopwords*, por não representar uma palavra relevante, essa é uma falha a ser considerada, já que a palavra "da" é uma preposição combinada com um artigo ("de" + "a"), o que é intitulado de contração, muito comum na língua portuguesa. Essa mesma ocorrência foi percebida nos trabalhos de [10] e [12].

Figura 6: Distribuição dos erros do *POS-Tagger*



Fonte: Os autores.

A provável causa desses erros é a classificação atribuída ao token "da" no corpus de treinamento *Mac-Morpho*, no qual as classificações mais comuns da palavra são substantivo (N) e nome próprio (NPROP). É o caso também de numerais romanos, classificados como nomes próprios, e do símbolo "%", classificado como substantivo, pois não há uma etiqueta para símbolos.

Desta forma, buscou-se antecipar a etapa de remoção dos *stopword* para evitar que o token "da" fosse etiquetado, mas a retirada antes da etapa de *POS-tagging* não alterou significativamente o percentual geral de acurácia, pois introduziu novos erros. A tendência é que isso tenha acontecido devido à perda de partes do contexto original de alguns tokens. Sem *stopwords*, os *n-gram taggers* julgaram erroneamente a função

morfossintática do token na frase. Por esses motivos, foi considerado mais adequado seguir com a metodologia citada anteriormente.

A execução correta das etapas de pré-processamento necessárias é fundamental para obter sucesso ao realizar o PLN. Uma das estratégias para a EI envolve regras e features criadas com ferramentas como *POS-tags* [1]. Assim, é a partir das etapas de pré-processamento que informações sobre o texto analisado são usadas na EI, e, por essa razão, é fundamental que essas informações estejam corretas.

6. CONCLUSÃO

O PLN é um campo importante para viabilizar a extração de informação automática de documentos regulatórios, o que se inicia com o pré-processamento dos códigos. A partir desse contexto, esta pesquisa investigou um conjunto de algoritmos de PLN para o pré-processamento de cláusulas de um código regulatório urbanístico brasileiro, visando automatizar a extração e transformação de informação.

Os resultados obtidos resultaram em uma acurácia de, cerca de, 68% no desempenho na etiquetagem realizada pelos algoritmos, indicando que ainda é necessário aprimorar esses recursos disponíveis para a língua portuguesa, principalmente o *POS-tagger e* o corpus de treinamento *Mac-Morpho* da biblioteca NLTK utilizada.

A principal contribuição deste estudo é permitir uma maior compreensão das dificuldades e prováveis causas dos erros das etiquetas na fase de pré-processamento do PLN. Além disso, contribui também para uma maior representatividade de trabalhos que avaliam quantitativamente o desempenho da etiquetagem utilizando *POS-tagger* de códigos em língua portuguesa.

Futuramente, os autores planejam investigar soluções para os principais problemas encontrados, bem como selecionar trechos mais complexos do código regulatório. Além disso, é recomendável utilizar *chunking grammars* mais complexas para reconhecer padrões de apoio a EI, baseada em regras.

Dentre as sugestões para otimização das etapas do PLN, recomenda-se a investigação de rotinas de conversão automática de documentos normativos em HTML ou PDF para formatos como TXT ou XML, bem como investigar as vantagens e desvantagens desses formatos para o PLN. Espera-se que trabalhos futuros investiguem o desempenho de outras bibliotecas de PLN para Python em língua portuguesa, como a spaCy e NLPNET.

AGRADECIMENTOS

O presente trabalho foi realizado com apoio da Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq).

REFERÊNCIAS

[1] Fuchs, S., Amor, R. (2021). **Natural Language Processing for Building Code Interpretation: A Systematic Literature Review**.

[2] Beach, T. H, Hippolyte, J., Rezgui, Y. 2020. **Towards the adoption of automated regulatory compliance checking in the built environment**. Automation in Construction. 118, 103285.

[3] Nawari, N. 2012. **The Challenge of Computerizing Building Codes in a BIM Environment.** Comput. Civ. Eng. 1, 285–292.5

[4] Olsson, P., Axelsson, J., Hooper, M., Harrie, L. 2018. **Automation of building permission by integration of BIM and geospatial data**. International Journal of Geo-Information. 7 (8), 307.

[5] Nawari, N. O. (2019). **Generalized Adaptive Framework for Computerizing the Building Design Review Process**, Journal of architecture Engineering, 26(1), 04019026.

[6] Altıntas, Y. D., Ilal, M. 2021. **Loose coupling of GIS and BIM data models for automated compliance checking against zoning codes**, Automation in Construction, 128, p. 103743.

[7] Kim, I., Choi, J., Teo, E.A.L., Sun, H. 2020. **Development of KBIM e-submission prototypical system for the openBIM-based building permit framework.** Journal of Civil Engineering and Management. 26 (8), 744-756.

[8] Shahi, K., McCabe, B.Y., Shahi, A.. **Framework for Automated Model-Based e-Permitting System for Municipal Jurisdictions**, Journal of Management in Engineering, 35 (6), 04019025. 2019

[9] Salama, D. M.; El-Gohary, N.M. **Semantic Text Classification for Supporting Automated Compliance Checking in Construction**, Journal of Computing in Civil Engineering, 30(1), 04014106. 2014

[10] Nieves, T.; Mendonça, E. A. de.; Ferreira, S. L.. **Processamento de Linguagem Natural na indústria AEC: uma abordagem para tradução de regulamentos edilícios brasileiros para o domínio BIM**. In: SIMPÓSIO BRASILEIRO DE TECNOLOGIA DA INFORMAÇÃO E COMUNICAÇÃO NA CONSTRUÇÃO, 3., 2021, Uberlândia. Anais [...]. Porto Alegre: ANTAC, 2021. p. 1-14. Disponível em: https://eventos.antac.org.br/index.php/sbtic/article/view/613, accessed em: 03 ago. 2021.

[11] Barbosa, J.; Vieira, J.; Santos, R.; Junior, G.; Muniz, M.; Moura, R. Introdução ao Processamento de Linguagem Natural usando Python. In: III Escola Regional de Informática do Piauí. **Livro Anais - Artigos e Minicursos**. 2017. P. 336-360.

[12] Rodriguez, M. Bezerra, B. (2020**). Processamento de Linguagem Natural para Reconhecimento de Entidades Nomeadas em Textos Jurídicos de Atos Administrativos (Portarias)**. Revista de Engenharia e Pesquisa Aplicada. Special Edition, p. 67-77.

[13] Zhang, J.; El-Gohary, N. M. **Extending Building Information Models Semi automatically Using Semantic Natural Language Processing Techniques**, Journal of Computing in Civil Engineering, 30(5), C4016004.

[14] NLTK Project. (n.d.). **Natural Language Toolkit — NLTK 3.6.2 documentation**. [online] Available at: https://www.nltk.org/index.html, accessed March 2022.

[15] Zhang, J.; El-Gohary, N. M. **Semantic NLP-Based Information Extraction from Construction Regulatory Documents for Automated Compliance Checking**, Journal of Computing in Civil Engineering, 30(2), 04015014.

[16] Xue, X.; Zhang, J. Evaluation of Seven Part-of-Speech Taggers in Tagging Building Codes: Identifying the Best Performing Taggers and Common Soucers of Erros. In. Construction Research Congress 2020. 2020. Tempe. **Construction Research Congress 2020: Computer Applications**. p.498-507

[17] Aluísio, S. M.; Pelizzoni, J. M.; Marchi, A. R.; Oliveira, L. H.; Manenti, R. E.; Marquivafável, V. (2003). **An Account of the Challenge of Tagging a Reference Corpus for Brazilian Portuguese**, pages 110–117. Springer Berlin Heidelberg, Berlin, Heidelberg.