

Processamento de Linguagem Natural para Reconhecimento de Entidades Nomeadas em Textos Jurídicos de Atos Administrativos (Portarias)

Natural Language Processing for Named Entities Recognition in Legal Texts of Administrative Acts (Ordinances)

Marcia Marina Miranda da Silva Rodríguez
Byron Leite Dantas Bezerra

¹ Escola Politécnica de Pernambuco, Universidade de Pernambuco, Recife, Brasil,

² Pós-graduação em Ciência dos Dados e Analytics, Escola Politécnica de Pernambuco, Recife, Brasil.

E-mail do autor principal: Marcia Rodríguez marciam.rodriguez@gmail.com

Resumo

É fato que a necessidade da integração de mecanismos e plataformas capazes de desempenhar atividades semelhantes às realizadas pelo humano se faz cada vez mais necessário na sociedade atual. O Processamento de Linguagem Natural (PLN) é uma ferramenta que facilita essa interação, pois consiste em uma subárea da inteligência artificial e refere-se, de forma ampla, ao estudo e desenvolvimento de sistemas computacionais que podem interpretar a fala e o texto conforme naturalmente os seres humanos falam e o digitam, além de desenvolver melhor a forma de interpretação da linguagem humana em diferentes dispositivos. Dito isto, será demonstrado nesta pesquisa como o Processamento de Linguagem Natural pode ajudar a automatizar o reconhecimento de Entidades Nomeadas (Agentes Públicos) em uma base de Portarias. Para isto será utilizado o *NLTK (Natural Language Toolkit)*, uma plataforma usada para construir programas *Python* que trabalham com dados de linguagem humana para aplicação em PLN. O *NLTK* será útil para separar as sentenças em um parágrafo, separar as palavras dentro de cada sentença, reconhecer padrões no texto e criar modelos de classificação que permitam, por exemplo, identificar nomes próprios dentro de conjunto de dados.

Palavras-Chave: Linguagem; Texto; Processamento; Python; NLTK; PLN.

Abstract

It is a fact that the need for the integration of mechanisms and platforms capable of carrying out activities similar to those carried out by the human being becomes more and more necessary in today's society. The Natural Language Processing (NLP) is a tool that facilitates this interaction, since it consists of a subarea of artificial intelligence and refers, in a wide way, to the study and development of computational systems that can interpret speech and text as naturally human beings speak and write it, as well as developing better the way human language is interpreted in different devices. That said, it will be demonstrated in this research how Natural Language Processing can help automate the recognition of Named Entities (Public Agents) on a Ordinances. This will use NLTK (Natural Language Toolkit), a platform used to build Python programs that work with human language data for PLN application. NLTK will be useful for separating sentences into a paragraph, separating words within each sentence, recognizing patterns in the text, and creating classification templates that allow, for example, identifying proper names within a data set.

Key-words: Language; Text; Processing; Python; NLTK; NLP;

1 Introdução

Um dos maiores problemas relacionados ao Processamento de Linguagem Natural (PNL) é a questão da dimensionalidade do vocabulário. A comunicação humana é frustrantemente vaga às vezes, pois a maioria das pessoas acabam utilizando coloquialismos, abreviações e muitas vezes não há preocupação em corrigir erros ortográficos, principalmente no universo da internet e dos dados digitais, que se expande não só para incluir um número cada vez maior de pessoas *on-line*, mas também para incluir todas as informações esperadas da *IoT* (Internet das Coisas).

O Processamento de Linguagem Natural (PLN) surgiu para resolver problemas relacionados à recuperação da informação, ao observar que os documentos e as expressões de busca são apenas objetos linguísticos. Assim, o PLN visa promover um nível mais alto de compreensão da linguagem natural através do uso de recursos computacionais, com o emprego de técnicas para o rápido processamento de texto [1]. Trata-se de um mecanismo criado não somente para extrair as informações de textos, mas também para facilitar a entrada de dados nos sistemas e a estruturação desses dados [2].

De acordo com Aranha [3], o PLN é o campo da Ciência da Computação e da Linguística que abrange um conjunto de métodos formais para analisar textos e gerar frases em um idioma humano através do uso de programas computacionais. Através de modelos você pode ensinar seu sistema a identificar regras e padrões estabelecidos pela linguagem na qual o texto está escrito, como por exemplo, os nomes das pessoas que geralmente seguem fórmulas generalizadas de duas ou três palavras de substantivos e substantivos próprios.

Nesta pesquisa, será possível identificar a importância das análises morfológicas (por exemplo, remoção de *stopwords* e *stemming*), sintáticas (por exemplo, *POS tagger* e *Parser*) e semânticas (por exemplo, utilização de dicionários semânticos para contextualizar sentenças), e conforme observado em Irfan [4], a etapa de pré-processamento tem papel importante para a tarefa de mineração de textos.

Por este motivo, antes de executar os algoritmos, os sistemas precisam pré-processar o texto e extrair suas características estruturais. Segundo Manning [5], existem diversas técnicas para normalizar os dados, e essas técnicas variam de acordo com a necessidade da aplicação. Seguem algumas:

- **Normalização:** A normalização abrange *tokenização*, transformação de letras maiúsculas para minúsculas, remoção de caracteres especiais, remoção de *tags*, etc. A identificação de *tokens* é uma importante etapa do pré-processamento para extrair unidades mínimas de textos. Cada unidade é chamada de *token* e, normalmente,

corresponde a uma palavra do texto, podendo estar relacionada também a símbolos e caracteres de pontuação, com " ", ".", "!" [5].

- **Remoção de Numerais:** Por não agregarem informação relevante à semântica do texto.
- **Remoção de *Stopwords*:** Remoção de palavras que ocorrem frequentemente, mas que na maioria das vezes não são informações muito relevantes para a construção do modelo. Uma lista de *stopword* é constituída por palavras que adicionam pouco valor à análise. Normalmente, correspondem aos artigos, preposições, pontuação, conjunções e pronomes de uma língua [6].
- ***Stemming*:** Após a retirada das *stopwords*, pode-se realizar a técnica de *stemming* para reduzir cada palavra do léxico. A raiz de uma palavra é encontrada, na maioria das vezes, eliminando os prefixos e sufixos que indicam variação na forma da palavra, como plural e tempos verbais. Segundo Brito [7], o processo de *stemming* traz benefícios no pré-processamento, sendo possível reduzir drasticamente o tamanho do léxico e também o esforço computacional, aumentando assim a precisão dos resultados.

Dito isto, nesta pesquisa apresentaremos algumas estruturas de dados básicos de *Python* para a manipulação de textos. Nos algoritmos será utilizada a biblioteca *NLTK* (*Natural Language Toolkit*), que foi desenvolvida em *Python* e apresenta grande volume de recursos, como: classificação, *tokenização*, *stemming*, *tagging*, *parsing* e raciocínio semântico. Todas essas funções são utilizadas para análise dos textos. Esta biblioteca foi escolhida pela sua curva de aprendizagem, sua sintaxe transparente e pela facilidade de manipular as funções.

2 Trabalhos Relacionados

Esta seção abordará alguns estudos recentes relacionados ao tema estudado.

2.1 Classificação de Documentos com Processamento de Linguagem Natural

Neste projeto Santos [8] analisa e compara técnicas de classificação com PLN, através da linguagem *Python* e com apoio da biblioteca *NLTK*.

Um dos principais objetivos foi aplicar os classificadores implementados na classificação de literatura na área das proteínas, e para isto foram implementados algoritmos para classificação de documentos, com técnicas de *Text Mining* e PLN, utilizando classificadores como Redes Neurais, SVM's e Redes Bayesianas.

O trabalho realizado teve como objetivo criar uma plataforma web para classificação de documentos baseado em algoritmos de aprendizagem supervisionada utilizando ferramentas de NLP e fazendo vários estudos de forma a avaliar os resultados dos algoritmos implementados.

Os classificadores foram implementados utilizando a biblioteca *NLTK* e *Scikit-Learn*. Foram usados os seguintes classificadores referentes ao *NLTK*: *Decision Tree*, *Maximum Entropy* (algoritmo iis e algoritmo gis). Os classificadores utilizados provenientes do *Scikit-Learn* foram estes: *Logistic Regression*, *NB Bernoulli*, *NB Multinomial*, *Linear SVC*, *Nu SVC*, e *SVC*.

Os classificadores com melhores resultados nos diferentes casos de estudo de Santos foram o *Linear SVC* (100% acurácia), o *Logistic Regression* (100% acurácia), e *Multinomial NB* (98,8% acurácia). Ambos com a transformação do texto para letras minúsculas e com remoção de *non letters* (pontuações e caracteres especiais) e *stopwords*.

Santos conclui ainda que a velocidade da execução de classificação com o *NLTK* e *Scikit-Learn* é de uma grande velocidade e eficiência, no qual foi possível treinar os modelos e os documentos em poucos segundos.

2.2 Criando um Corpus sobre Desastres Climáticos com Apoio da Ferramenta NLTK

No trabalho publicado por Molina e Steinberger-Elias [9] foram demonstradas etapas de composição e descrição de um corpus de textos sobre o terremoto do Haiti com apoio do pacote *NLTK* e do PLN, com objetivo de identificar as entidades assistenciais atuantes nestes eventos e na prestação de socorro às vítimas.

O corpus adotado nesta pesquisa constitui-se de textos extraídos da Folha de São Paulo no período de 12/01/2010 a 12/02/2011 utilizando como busca o termo "Haiti". Foram levantados 842 textos noticiando sobre o terremoto ocorrido no início de 2010, sendo o corpo de texto salvo em formato texto e os demais dados (Identidade do Evento (ID), Identidade Numérica (Nº), Data, Título da Matéria, Subtítulo, *Link*, Instituição, Autoria, Seção, Local, Figura e Legenda) em planilhas eletrônicas.

A tabela da figura 1 apresenta os filtros aplicados, o número total de ocorrências (em *types* e *tokens*), densidade lexical (*tokens/types*), porcentagem de ocorrências acumuladas das

100 primeiras expressões com relação ao todo e a quantidade de palavras acrescida à lista de 100 mais ocorrentes com relação ao filtro anterior.

Filtro	Nº de tokens	Nº de types	Densidade lexical	Representação do recorte sobre o total	Acréscimo de novas palavras
Nenhum	429135	34788	12,34	48,53%	-
Eliminando stopwords (Filtro 1)	307559	34643	8,88	34,52%	39
(Filtro 1) + tomando somente alfabéticos (Filtro 2)	234355	33456	7,00	21,24%	21
Maiúscula sem estar após "." (Filtro 3)	46712	8425	5,54	36,25%	-
(Filtro 3) + sem stopwords (Filtro 4)	42394	8328	5,09	34,32%	16

Figura 1: Dados de aplicações de filtros sobre 100 primeiras ocorrências em frequência.

Fonte: Molina e Steinberger (2011)

A aplicação do Filtro 2 permitiu a visualização de um perfil de itens lexicais que possuem alto conteúdo semântico (como substantivos e verbos, por exemplo). Levando em conta que dentro dos resultados deste filtro é que se encontraram as informações buscadas, os resultados apontaram que em 21,24% se concentra nas 100 primeiras ocorrências.

O uso do pacote *NLTK* foi considerado por Molina e Steinberger-Elias como bem-sucedido, com retornos em frequência de itens lexicais, *collocations*, aplicação de filtros para gerar listas com propriedades pré-determinadas léxico-gramaticais, manipulação do corpus para obter dados que permitissem validar estatisticamente retornos trazidos pelo pacote e análise de itens e combinações de itens em categorias [9].

2.3 Análise de Sentimentos do Twitter com Naïve Bayes e NLTK

O artigo de Weiand [10] propõe um algoritmo de análise de sentimentos dos *tweets* do *Twitter*, utilizando o modelo probabilístico de *Naïve Bayes*, buscando apresentar um algoritmo de análise de sentimentos de *tweets* coletados com auxílio da biblioteca desenvolvida por Sanders [11], e o algoritmo de *Naïve Bayes*, implementando este sistema sob a linguagem de programação *Python*, juntamente com a utilização da biblioteca *NLTK* e *ScikitLearn*.

Foi utilizado o *Naïve Bayes* por ser um modelo probabilístico simples com base na regra de Bayes com seleção de recursos independentes, no qual foi implementado no sistema e foi considerado como tendo bons resultados na categorização de texto. Ele possibilita a não restrição do número de classes ou atributos, assim como é um dos algoritmos de aprendizado mais rápido para a fase de treinamento de um sistema deste tipo [10].

Já a utilização do *NLTK* teve como principal objetivo o auxílio ao desenvolvimento e implementação do algoritmo *Naïve Bayes*, mas,

DOI: 10.xxxx/s11468-014-9759-3

também foram utilizados outros recursos disponíveis por ela, como *stemming* e treino de palavras. E o *Scikit-Learn* auxiliou no algoritmo de validação cruzada.

Para o desenvolvimento da pesquisa de Weiland foi necessário construir um corpus com as classificações dos *tweets* em positivos e negativos, para que desta forma fossem realizadas as inferências necessárias ao algoritmo de treinamento.

Segue na figura 2 as etapas executadas para obtenção do corpus de forma íntegra.



Fonte: Autoria Própria, 2017.

Figura 2: Fluxo de análise.

Fonte: Weiland (2017)

Após estas etapas, foram utilizadas as bibliotecas *NLTK* para a aplicação do algoritmo de *Naïve Bayes*, a redução dos radicais das palavras, o treino e também para a medição de acurácia das aplicações. A *Scikit-Learn* foi utilizada para a aplicação da validação cruzada após a etapa de treino.

Com a utilização da biblioteca de Sanders [11], foi possível Weiland trabalhar em um corpus de 5.513 *tweets* pré-processados, porém, necessitando a recuperação dos mesmos através da *RestAPI* do *Twitter*. Porém, na extração dos dados da *RestAPI* só foram possíveis localizar 494 *tweets*, correspondendo a 8,9% do *dataset* original.

O *dataset* foi separado de modo aleatório em dois grupos, um com 80% dos dados, os quais foram utilizados para o treino do algoritmo, 10% dos dados para a validação cruzada e os 10% restantes para o teste. Para a aplicação destas etapas foram utilizadas as bibliotecas *NLTK* e *Scikit-Learn*, sendo que os resultados obtidos trouxeram um número de acerto considerado por Weiland relativamente alto 0,91 nos conjuntos mencionados.

Após o treino, foram aplicados os testes para constituição da validação cruzada, resultando nos dados descritos na figura 3.

	Pré-Classificação	Classificação
Positivo	19	15
Negativo	30	34
Validação Cruzada	49	49

Fonte: Autoria Própria, 2017.

Figura 3: Validação cruzada.

Fonte: Weiland (2017)

3 Sistema Proposto

3.1 Etapas do PLN

Ao trabalhar com PLN, uma prática comum é dividir os problemas em tarefas menores: segmentação de sentenças, *tokenização*, *part-of-speech (POS) tagging*, *chunking*, *parsing*, entre outras [12], conforme necessidade identificada.

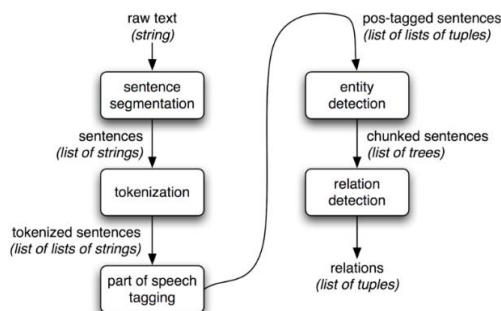


Figura 4: Exemplos de etapas do PLN.

Fonte: <http://www.nltk.org/book/ch07.html>

A figura 4 demonstra as etapas de processamento de um documento usando vários dos procedimentos citados nesta pesquisa. Primeiro, o texto bruto do documento é dividido em sentenças usando um segmentador de sentenças e cada sentença é subdividida em palavras usando um *tokenizador*. Em seguida, cada sentença é marcada com *tags* (*part of speech*), o que será muito útil na etapa seguinte, denominada *pos-tagged*. Nesta etapa, são procuradas menções de entidades interessantes em cada sentença. Finalmente, é realizada a etapa de *chunking* para procurar relações prováveis entre diferentes entidades no texto.

3.2 Metodologia

A metodologia aplicada nesta pesquisa será a sugerida por Capobianco [13], representada na Figura 5.

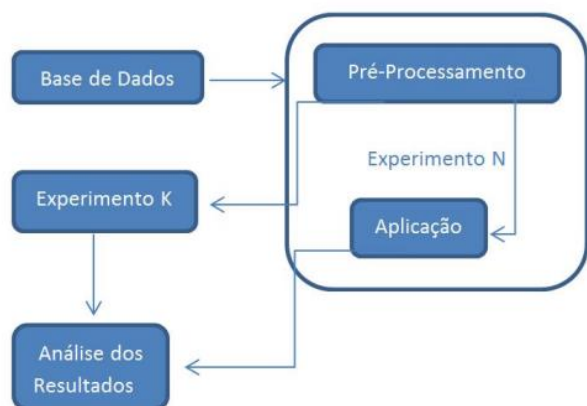


Figura 5: Metodologia aplicada. Etapas seguidas para realização do experimento e obtenção dos resultados.
Fonte: Capobianco (2016).

3.3 Base de Dados

As informações utilizadas na construção da base de dados desta pesquisa foram extraídas do Diário Oficial da União (DOU), em Seções nas quais Portarias de nomeação, exoneração, aposentadoria e outros atos relativos a servidores da administração são divulgadas. Exemplo:

"Poder Judiciário - TRIBUNAL REGIONAL DO TRABALHO DA 15ª REGIÃO DESEMBARGADOR PRESIDENTE DO TRIBUNAL REGIONAL DO TRABALHO DA 15ª REGIÃO, no uso de suas atribuições legais e regimentais, resolve: Nº 91 - Exonerar, **JOSE PAULO DELCI**, Técnico Judiciário, área Administrativa, do Quadro Permanente da Secretaria deste Tribunal, do cargo em comissão de Diretor de Serviço de Distribuição dos Feitos, CJ-02, do mesmo Quadro, em virtude de transformação do cargo em comissão prevista no art. 1º da Resolução Administrativa nº 29/2017 deste Tribunal, conforme Anexo I daquele normativo."

O objetivo será isolar os trechos que representam os nomes de pessoas através de classificadores somados a técnicas de processamento de linguagem natural. A abordagem mais simples para a identificação de nomes próprios é observar as letras maiúsculas, e por se tratar de uma Seção onde são divulgados dados formais, este tipo de estrutura será uma grande aliada para a análise semântica dos classificadores. Porém, em textos jurídicos diversas Entidades são citadas, e o desafio será separá-las dos nomes de pessoas, obtendo assim extrações com maiores precisões.

3.4 Pré-Processamento

Esta etapa é executada imediatamente após a coleta dos dados e tem como objetivo prover alguma formatação da massa textual. Para Gonçalves [14], o principal objetivo de pré-processar um texto consiste

na filtragem e limpeza dos dados, eliminando redundâncias e informações desnecessárias para o conhecimento que se deseja extrair.

Para esta base, e com grande importância para identificação de nomes próprios e de pessoas, não foram alteradas, no geral, as estruturas de letras maiúsculas e minúsculas, uma vez que durante os testes de processamento textual se mostrou essencial na classificação das *tags*. Porém, um dicionário foi criado para este projeto com intuito de alterar algumas palavras específicas para a forma minúscula, pois eram frequentemente classificadas de forma equivocada pelo algoritmo como um nome próprio.

Nesta base também foram identificadas algumas características que necessitaram de tratamentos específicos para que as extrações fossem mais precisas, pois o intuito é identificar apenas os nomes de pessoas mencionados nas Portarias (como nomeações, exonerações, aposentaria e atos). Em grande volume destes textos há a citação de pessoas que "assinaram/autorizaram" as Portarias, e estas não deverão ser extraídas. Para isto, foram criados algoritmos para retirada prévia dos nomes citados no final do texto. Exemplo:

"MINISTERIO DA DEFESA - DECRETO DE 12 DE JUNHO DE 2018 O PRESIDENTE DA REPÚBLICA, no uso da atribuição que lhe confere o art. 84, caput, incisos I e XXV, da Constituição, resolve NOMEAR **JOAQUIM SILVA E LUNA**, para exercer o cargo de Ministro de Estado da Defesa, ficando exonerado do que atualmente ocupa. Brasília, 12 de junho de 2018; 197ª da Independência e 130ª da República. **MICHEL TEMER Torquato Jardim**"

Como complemento da etapa de pré-processamento, foram utilizados algoritmos de remoção de acentos, caracteres especiais e quebras de linhas.

3.5 Tokenização

Um dos principais passos de uma operação de normalização é a *tokenização* e sua execução tem como finalidade desmembrar um documento textual em unidades mínimas, mas que representem a mesma semântica original do texto. É utilizada durante o processamento de linguagem natural para segmentação de palavras, realizando a quebra da sequência de caracteres em um texto localizando o limite de cada palavra, ou seja, separar palavras ou sentenças em unidades.

A *tokenização* lexical marca cada palavra como um *token* no texto, identificando-a mesmo se tiver encostada em alguma pontuação.

Exemplo:

['NOMEAR','JOAQUIM','SILVA', 'E', 'LUNA', ',', 'para', 'exercer', 'o', 'cargo', '.']

A *tokenização* sentencial identifica e marca sentenças.

Exemplo:

['NOMEAR JOAQUIM SILVA E LUNA,' , 'para exercer o cargo.']

Por fim, o principal objetivo de criar *tokens* é a tradução de um texto em dimensões possíveis de se avaliar, analisar, para obtenção de um conjunto de dados estruturados [15].

3.6 POS-tagging

Em sequência é executado o processo de etiquetagem. A marcação *part-of-speech* é a tarefa mais complexa na extração de entidades. A ideia é combinar os *tokens* com as *tags* correspondentes (substantivos, verbos, adjetivos, advérbios etc.). O processo de classificar e rotular as palavras é conhecido como *POS-tagging* ou simplesmente *tagging*.

A etiquetagem (*POS tagging*), conforme Andrade [16], tem como objetivo atribuir uma categoria morfosintática a cada *token* da sentença processada. O processo de etiquetagem possui como entrada a sentença *tokenizada*, obtendo como saída uma lista de marcações. A identificação precisa dos elementos morfosintáticos de uma sentença é de grande importância, pois ao classificar uma única palavra de forma incorreta, pode-se gerar erros de processamento em fases posteriores. Exemplo:

[('Nomear', 'JJ'), ('JOAQUIM', 'NNP'), ('SILVA', 'NNP'), ('E', 'NNP'), ('LUNA', 'NNP'), (',', ','), ('para', 'NN'), ('exercer', 'NN'), ('o', 'JJ'), ('cargo', 'NN'), ('de', 'FW'), ('Ministro', 'NNP'), ('de', 'FW'), ('Estado', 'NNP'), ('da', 'NN'), ('Defesa', 'NNP'), (',', ','), ('ficando', 'NN'), ('exonerado', 'NN'), ('do', 'VBP'), ('que', 'RB'), ('atualmente', 'VB'), ('ocupa', 'NN'), (',', ','), ('Brasília', 'NNP'), (',', ','), ('12', 'CD'), ('de', 'FW'), ('junho', 'FW'), ('de', 'IN'), ('2018', 'CD'), (',', ','), ('1970', 'CD'), ('da', 'NN'), ('Independência', 'NNP'), ('e', 'VBZ'), ('1300', 'CD'), ('da', 'NN'), ('República', 'NN')]

Um exemplo em árvore:

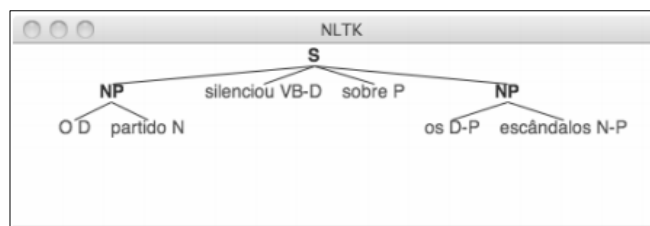


Figura 6: Exemplo de *POS-tagging* gerada pelo NLTK.

Fonte: Alencar (2011).

3.7 Chunking

A técnica básica utilizada para detectar as entidades é chamada de *chunking*, a qual segmenta sequências de *multi-tokens*.

Segundo Sang [17], *chunking* é conhecida como uma tarefa de pré-processamento de um *parser*, que tem por objetivo a divisão de um texto em frases, de forma que as palavras relacionadas sintaticamente façam parte de um mesmo conjunto. Sua utilização ocorre de modo sequencial, analisando os elementos através de suas etiquetas e assim agrupando-os.

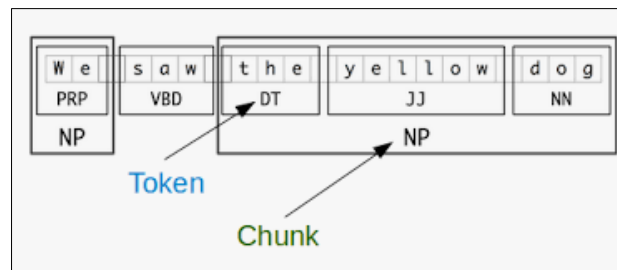


Figura 7: Segmentação e Etiquetagem nos níveis *token* e *chunk*.

Fonte: <http://www.nltk.org/book/ch07.html>

Definimos uma gramática *chunk* (*chunk grammar*) consistindo em padrões que indicam como o trecho deve ser extraído.

```
grammar = r"""
NP: {<DT|JJ|NN.*>+}      # Chunk sequences of DT, JJ, NN
PP: {<IN><NP>}             # Chunk prepositions followed by NP
VP: {<VB.*><NP|PP|CLAUSE>+$} # Chunk verbs and their arguments
CLAUSE: {<NP><VP>}         # Chunk NP, VP
"""

cp = nltk.RegexpParser(grammar)
sentence = [ ("Mary", "NN"), ("saw", "VBD"), ("the", "DT"), ("cat", "NN"),
             ("sit", "VB"), ("on", "IN"), ("the", "DT"), ("mat", "NN")]

>>> print(cp.parse(sentence))
(S
 (NP Mary/NN)
 saw/VBD
 (CLAUSE
  (NP the/DT cat/NN)
  (VP sit/VB (PP on/IN (NP the/DT mat/NN)))))
```

Figura 8: Exemplo de um *chunk grammar* baseado em expressão regular simples.

Fonte: <https://emotionrobots.com/nltk-part-2/>

Como o intuito desta pesquisa é a extração de entidades nomeadas (nomes de pessoas), a sequência a ser observada será o conjunto de *tags* "NNP", que indica nomes próprios.

gramatica = r"""NOME: {(<NNP>+)}"""

```
(NOME JOAQUIM/NNP SILVA/NNP E/NNP LUNA/NNP)
(NOME Ministro/NNP)
(NOME Estado/NNP)
(NOME Defesa/NNP)
(NOME Brasília/NNP)
```

4 Estudo de Caso

Nos algoritmos descritos nesta pesquisa serão aplicadas as técnicas apresentadas nos tópicos anteriores, de pré-processamento, *tokenização*, *pos-tagging*, *chunking* e apresentação dos trechos tratados e extraídos, para reconhecimento das Entidades Nomeadas (nomes de pessoas).

4.1 Descrição da Base de Dados

A base extraída e inserida na planilha Base10000.xlsx possui 10.000 registros, onde na coluna1 da planilha consta o texto da Portaria e na coluna2 consta o Nome já existente na base e extraído de forma manual. Cada bloco de texto possui em média 102 palavras e cada nome extraído possui em média 3,6 palavras. Há blocos em que poderão ser citados mais de um nome.

O texto da Portaria será utilizado nos algoritmos para extração dos nomes de pessoas e o Nome contido na coluna2 será utilizado para comparativos e análise do resultado (acurácia).

Texto	Nome
Ministerio da Educacao - INSTITUTO FEDER	ABDON SANTOS NOGUEIRA
Presidencia da Republica - CASA CIVIL O M	ABEL FERREIRA LEITE NETO
Presidencia da Republica - SECRETARIA DE	ABEL FERREIRA LEITE NETO
Ministerio do Planejamento Desenvolvi	ABELARDO DE JESUS FILHO
Ministerio do Planejamento Desenvolvi	ABELARDO DE JESUS FILHO
Ministerio da Educacao - CAMPUS CODÓ DI	ABIAS RODRIGUES DA CRUZ
Entidades de Fiscalizacao do Exercicio das	ABNER ALCANTARA SAMHA SANTOS
Ministerio da Educacao - INSTITUTO FEDER	ABNER NUNES EMERICH DE PAULA
Ministerio da Educacao - UNIVERSIDADE FE	ABRAHAM LINCOLN RABELO DE SOUSA
Ministerio da Ciencia Tecnologia Inovaoe	ADACYR ALVES FERRAZ
Poder Judiciario - TRIBUNAL REGIONAL DO	ADAIL BENEDITO DA SILVA
Ministerio da Educacao - INSTITUTO FEDER	ADAILDE DO CARMO SANTOS
Ministerio da Educacao - INSTITUTO FEDER	ADAILDE DO CARMO SANTOS

Figura 9: Base de textos das Portarias e seus respectivos nomes extraídos

Fonte: Autoria própria (2018)

4.2 Etapas do Experimento

Nesta seção serão demonstradas as bases utilizadas, o pré-processamento realizado e o resultado obtido na extração dos Nomes de Pessoas contidos na base através da aplicação de algumas etapas do PLN/NTLK.

- 1) Carregamento das bases necessárias para o algoritmo

Nome	Tamanho	Tipo
Base10000	2.030 KB	Planilha do Microsoft Excel
ListaNomes	55 KB	Documento de Texto
PalavrasMinusculas	2 KB	Documento de Texto

Figura 10: Bases utilizadas no experimento

Fonte: Autoria própria (2018)

As bases poderão ser encontradas no link: <https://github.com/marciamrodriguez/projeto>

Base10000.xlsx: Planilha contendo 10.000 registros de textos de Portarias e Nomes de Pessoas para o treinamento.

PalavrasMinusculas.txt: Arquivo texto contendo lista de palavras (205 linhas) que deverão ser transformadas para minúsculas, pois frequentemente estavam sendo confundidas como nomes próprios. A lista de palavras foi criada durante o experimento para melhoria da assertividade na extração dos nomes.

ListaNomes.txt: Arquivo texto contendo lista de nomes de pessoas (6.551 linhas). Utilizado no algoritmo para melhorar a precisão da extração dos nomes. Os nomes foram extraídos dos links abaixo, sendo a lista complementada também por novos nomes encontrados durante o experimento.

<http://nomesportugueses.blogspot.com.br/p/nomes-brasileiros-de-z.html>

http://nomesportugueses.blogspot.com.br/p/nomes-masculinos-z_28.html

- 2) Percorrer a lista de textos da base Base10000.xls, limpando caracteres (como traços, aspas, dois pontos), removendo quebras de linha e acentuações.
- 3) Serão retirados os nomes de pessoas citadas no final do texto da Portaria, após o último "ponto final".

Algoritmo:

```
match_pattern =
re.search(r'\.(([A-Z])|([a-z])|(\s))+$',
texto)
if str(match_pattern) != "None":
    texto
    texto.replace(match_pattern.group(0), "")
```

Resultado:

"Nomear JOAQUIM SILVA E LUNA, para exercer o cargo de Ministro de Estado da Defesa, ficando exonerado do que atualmente ocupa. Brasília, 12 de junho de 2018; 197º da Independência e 130º da República.
MICHEL TEMER Torquato Jardim"

- 4) Converter as palavras contidas no arquivo PalavrasMinusculas.txt em formato minúsculo, sendo utilizado nos passos seguintes somente o novo texto criado.

4.1 Esta lista foi criada durante o treinamento do experimento, após identificação de "erros" repetitivos de palavras intituladas como "Nome Próprio".

4.2 Após conversão destas palavras para formato minúsculo os "erros" foram minimizados e a porcentagem de acerto foi elevada.

Original:

"NOMEAR GABRIEL FARIA OLIVEIRA para exercer o cargo de Defensor Público Geral Federal"

Novo texto:

"nomear GABRIEL FARIA OLIVEIRA
para exercer o cargo de Defensor Público
geral federal"

- 5) Aplicar o algoritmo de *tokenização* (utilizando como parâmetro a linguagem "portuguese")

Algoritmo:

```
tokens=nlk.word_tokenize(novotexto,  
language='portuguese')
```

Resultado:

```
['nomear', 'JOAQUIM', 'SILVA', 'E', 'LUNA', ',', 'para', 'exercer',  
'o', 'cargo', 'de', 'Ministro', 'de', 'estado', 'da', 'defesa', ',',  
'ficando', 'exonerado', 'do', 'que', 'atualmente',  
'ocupa', ',', 'Brasília', ',', '12', 'de', 'junho', 'de', '2018',  
, ',', '1970', 'da', 'Independência', 'e', '130º', 'da', 'república']
```

- 6) Aplicar o algoritmo de *POS-tagging*

Algoritmo:

```
classes = nlk.pos_tag(tokens)
```

Resultado:

```
[('nomear', 'JJ'), ('JOAQUIM', 'NNP'), ('SILVA', 'NNP'),  
( 'E', 'NNP'), ('LUNA', 'NNP'), (',', ','), ('para', 'NN'),  
( 'exercer', 'NN'), ('o', 'JJ'), ('cargo', 'NN'), ('de', 'FW'),  
( 'Ministro', 'NNP'), ('de', 'FW'), ('estado', 'FW'),  
( 'da', 'NN'), ('defesa', 'NN'), (',', ','), ('ficando', 'NN'),  
( 'exonerado', 'NN'), ('do', 'VBP'), ('que', 'RB'),  
( 'atualmente', 'VB'), ('ocupa', 'NN'), (',', ','), ('Brasília', 'NNP'),  
( ',', ','), ('12', 'CD'), ('de', 'FW'), ('junho', 'FW'), ('de', 'IN'),  
( '2018', 'CD'), (',', ','), ('1970', 'CD'), ('da', 'NN'),  
( 'Independência', 'NNP'), ('e', 'VBZ'), ('130º', 'CD'),  
( 'da', 'NN'), ('república', 'NN')]
```

- 7) Aplicar o algoritmo de *chunking*

Algoritmo:

```
entidades = nlk.chunk.ne_chunk(classes)
```

Resultado:

```
(S nomear/JJ  
ORGANIZATION JOAQUIM/NNP  
SILVA/NNP E/NNP LUNA/NNP  
para/NN exercer/NN o/JJ cargo/NN  
de/FW Ministro/NNP de/FW  
estado/FW da/NN defesa/NN  
ficando/NN exonerado/NN  
do/VBP que/RB atualmente/VB ocupa/NN  
(PERSON Brasília/NNP)  
12/CD de/FW junho/FW de/IN  
2018/CD ;/: 1970/CD da/NN  
(GPE Independência/NNP)  
e/VBZ 130º/CD da/NN  
república/NN)
```

- 8) Aplicar o algoritmo de gramática *chunk*

Algoritmo:

```
gramatica = r"""  
FALSO: {(<NN><NN><NNP><NNP><NN><NN>)}  
FALSO: {(<NN><NN>)}  
FALSO: {(<IN><FW><NNP>)}  
NOME: {(<NNP>>{3,})(<NN>|<FW>|<IN>|<VBP>|<VBD>|<VBZ>)  
(<NNP>+)(<NN>|<FW>|<IN>|<VBP>|<VBD>|<VBZ>)(<NNP>+)}  
NOME: {(<NNP>>{2,})(<NN>|<FW>|<IN>|<VBP>|<VBD>|<VBZ>)  
(<NNP>+)(<NN>|<FW>|<IN>|<VBP>|<VBD>|<VBZ>)(<NNP>+)}  
NOME: {(<NNP>>{1,})(<NN>|<FW>|<IN>|<VBP>|<VBD>|<VBZ>)  
(<NNP>+)(<NN>|<FW>|<IN>|<VBP>|<VBD>|<VBZ>)(<NNP>+)}  
NOME: {(<NNP>>{3,})(<NN>|<FW>|<IN>|<VBP>|<VBD>|<VBZ>)(<NNP>+)}  
NOME: {(<NNP>>{2,})(<NN>|<FW>|<IN>|<VBP>|<VBD>|<VBZ>)(<NNP>+)}  
NOME: {(<NNP>>{1,})(<NN>|<FW>|<IN>|<VBP>|<VBD>|<VBZ>)(<NNP>+)}  
NOME: {(<NNP>>{3,})(<NN>|<FW>|<IN>|<VBP>|<VBD>|<VBZ>)(<NNP>+)}  
NOME: {(<NNP>>{2,})(<NN>|<FW>|<IN>|<VBP>|<VBD>|<VBZ>)(<NNP>+)}  
NOME: {(<NNP>>{1,})(<NN>|<FW>|<IN>|<VBP>|<VBD>|<VBZ>)(<NNP>+)}  
NOME: {(<NNP>>{3,})(<NN>|<FW>|<IN>|<VBP>|<VBD>|<VBZ>)(<NNP>+)}  
NOME: {(<NNP>>{2,})(<NN>|<FW>|<IN>|<VBP>|<VBD>|<VBZ>)(<NNP>+)}  
NOME: {(<NNP>>{1,})(<NN>|<FW>|<IN>|<VBP>|<VBD>|<VBZ>)(<NNP>+)}  
NOME: {(<NNP>>{4,})}"""
```

```
analiseGramatical = nlk.RegexpParser(gramatica)  
resultado = analiseGramatical.parse(classes)
```

Resultado:

```
(S nomear/JJ  
(NOME JOAQUIM/NNP SILVA/NNP E/NNP LUNA/NNP)  
(FALSO para/NN exercer/NN)  
o/JJ cargo/NN de/FW Ministro/NNP de/FW  
estado/FW  
(FALSO da/NN defesa/NN)  
(FALSO ficando/NN exonerado/NN)  
do/VBP que/RB atualmente/VB ocupa/NN  
Brasília/NNP 12/CD de/FW junho/FW de/IN  
2018/CD ;/: 1970/CD da/NN Independência/NNP  
e/VBZ 130º/CD  
(FALSO da/NN república/NN))
```

- 9) Comparação dos Nomes Extraídos do algoritmo com a Lista de Nomes (ListaNomes.txt)

A Lista de Nomes tem com intuito realizar uma última verificação de que há evidências em que o trecho extraído seja um Nome de Pessoa.

Para todos os trechos considerados pelo algoritmo da etapa de *chunking* como "NOME", a primeira palavra será extraída e verificada a existência na Lista de Nomes. Caso exista, todo o trecho será considerado como um Nome Próprio. Caso não, o trecho será desconsiderado da extração.

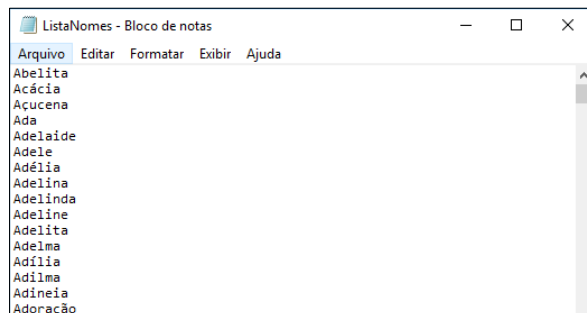
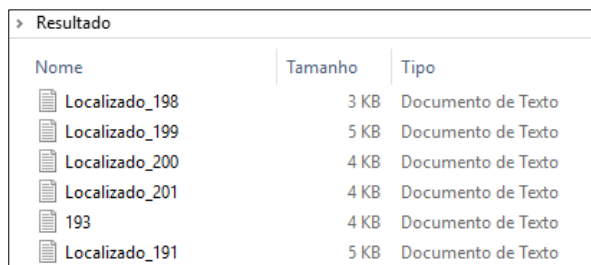


Figura 11: Lista de Nomes
Fonte: Autoria própria (2018)

- 10) Os resultados são armazenados em arquivos textos individuais.



Nome	Tamanho	Tipo
Localizado_198	3 KB	Documento de Texto
Localizado_199	5 KB	Documento de Texto
Localizado_200	4 KB	Documento de Texto
Localizado_201	4 KB	Documento de Texto
193	4 KB	Documento de Texto
Localizado_191	5 KB	Documento de Texto

Figura 9: Resultado obtido no experimento. Lista dos arquivos textos.

Fonte: Autoria própria (2018)

Para cada processamento de um texto, o resultado é armazenado em um arquivo .txt, com identificação da numeração da linha na planilha e com nomenclatura de "Localizado" ou não.

Os casos contendo a informação de "Localizado" significam que o Nome Extraído pelo algoritmo foi igual ao Nome previamente cadastrado na base (coluna 2 na planilha). Ou seja, a extração ocorreu conforme esperado.

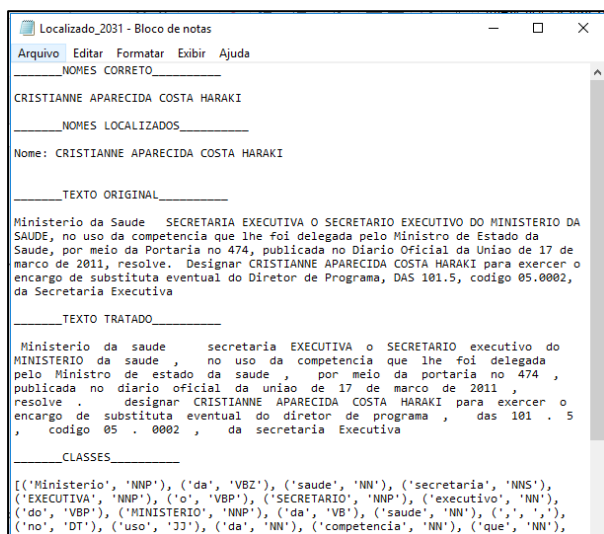


Figura 12: Resultado armazenado no arquivo texto para análise

Fonte: Autoria própria (2018)

4.3 Resultado e Análise do Experimento

Após a conclusão do processamento dos 10.000 registros, foi identificada uma assertividade de 92,9% dos nomes, quando comparado o Nome Extraído ao Nome contido na base (coluna 2 na planilha) por igualdade.

Os casos considerados como não "Não Localizados" podem ser analisados através dos arquivos textos armazenados durante o processamento.

Quando analisados, foram encontradas situações como:

- Nomes extraídos corretamente pela biblioteca, porém não existentes na ListaNomes.txt, sendo posteriormente desconsiderados da extração devido a regra de validação dos nomes. Encontrados em cerca de 1,5% dos casos "não localizados".
- Palavras classificadas equivocadamente como "Nome Próprio" (tag NNP) e posteriormente descartadas pela ListaNomes.txt. Este tipo de erro pode ser definido como um Falso-Positivo, que ocorre quando um resultado de algum teste ou equação dá positivo devido as suas características, mas na verdade o resultado é negativo. Encontrados em cerca de 4,7% dos casos "não localizados".
- Nomes ou Sobrenomes não identificados pelo algoritmo do NLTK como "Nome Próprio" (tag NNP), erro este que pode ser definido como um Falso-Negativo, quando um resultado dá negativo devido as suas características, mas na verdade o resultado é positivo. Encontrados em cerca de 1,8% dos casos "não localizados".

Em algumas situações, após análise do resultado, foram possíveis a identificação de medidas corretivas para as classificações, com intuito de aumentar a porcentagem de assertividade das extrações.

Exemplos:

- Cadastrar novos nomes na ListaNomes.txt.
- Cadastrar novas palavras na lista de PalavrasMinusculas.txt, para que seja minimizada a classificação equivocada de "Nomes Próprios".

5 Conclusão

Neste trabalho foi abordado o tema de Processamento de Linguagem Natural, apresentando o conceito, etapas, utilidades e vantagens do PLN em um conteúdo textual. Foi adotada e explorada a plataforma NLTK com Python para manipular sentenças a partir dos recursos de PLN, oferecendo um conjunto vasto de bibliotecas de processamento de textos para classificação, tokenização, stemming, marcação e análise semântica.

O experimento desenvolvido baseia-se em receber textos referentes a Portarias extraídas do Diário Oficial da União, no qual nomeiam e exoneram pessoas. Através das técnicas aplicadas de tokenização, pos-tagger e chunk foi demonstrada a análise, classificação e marcação semântica para cada sentença e palavra, numa dada linguagem, atribuindo classificações como substantivo, verbo,

entre outros, possibilitando a extração dos trechos contendo Nomes Próprios, objetivo principal deste projeto.

O método de *tokenização* foi utilizado para separar um conjunto de dados em pedaços menores significativos. Para este experimento, o conjunto é qualquer sentença informada, e os *tokens* são as palavras separadas da sentença. Estes *tokens* foram utilizados para serem analisados e classificados.

A qualidade da classificação, neste caso, depende quase completamente do processo de treinamento, pois se a mesma não for realizada com qualidade, o processamento irá produzir *outputs* indesejados para seu propósito. Com isto, foram realizadas etapas de pré-processamento, como remoção de caracteres e acentuações, substituição de palavras maiúsculas para minúsculas, escolha dos trechos a serem extraídos (através de expressões regulares), com intuito de obter maior assertividade na identificação dos Nomes de Pessoas a qual estavam sendo nomeados ou exonerados em suas respectivas Portarias.

Como conclusão deste experimento, foi possível observar uma acurácia de 92,9% na extração dos Nomes de Pessoas, percentual considerado bastante aceitável para o objetivo proposto.

Para os trabalhos futuros, sugere-se explorar também o *spaCy*, biblioteca Python também para PLN, já com um modelo pronto em português. O desenvolvimento da *spaCy* foi feito especificamente para uso em produção e ajudar a criar aplicações que consigam processar e “entender” um grande volume de texto. Ela pode ser usada para extrair informações ou entendimento de linguagem natural ou pré-processar texto para uso com *deep learning*. Em 2015, uma pesquisa da *Emory University* e *Yahoo! Labs* mostrou que o *spaCy* oferecia o *parser* sintático mais rápido do mundo e que sua acurácia estava entre as melhores disponíveis [18].

Referências

- [1] MACHADO, A. P.; FERREIRA, R.; BITTENCOURT, I. I.; ELIAS, E.; BRITO, P.; COSTA, E. Mineração de Texto em Redes sociais virtuais Aplicada à Educação a Distância. Revista Digital da CVA - Ricesu, ISSN 1519- 8529, v. 6, n. 23, Julho de 2010.
- [2] BULEGON, H.; MORO, C. M. C. Mineração de texto e o processamento de linguagem natural em sumários de alta hospitalar. Journal of Health Informatics, 2010.
- [3] ARANHA, C. N. Uma Abordagem de Pré-Processamento Automático para Mineração de Textos em Português: Sob o Enfoque da Inteligência Computacional. Tese (Doutorado em Engenharia Elétrica) Pontifica Universidade Católica do Rio de Janeiro – PUC - Rio, Rio de Janeiro, 2007.
- [4] IRFAN, R., King, C. K., Grages, D., Ewen, S., Khan, S. U., Madani, S. A., Kolodziej, J., Wang, L., Chen, D., Rayes, A., et al. (2015). A survey on text mining in social networks. The Knowledge Engineering Review, 30(02):157–170.
- [5] MANNING, C. D. et al. Introduction to information retrieval. [S.l.]: Cambridge university press Cambridge, 2008.
- [6] INDURKHYA, N.; DAMERAU, F. J. Handbook of natural language processing. [S.l.]: CRC Press, 2010.
- [7] BRITO, E. M. N. Mineração de Textos: detecção automática de sentimentos em comentários nas mídias sociais. Universidade FUMEC, Belo Horizonte-MG, 2017.
- [8] SANTOS, C. M. Classificação de Documentos com Processamento de Linguagem Natural. Instituto Superior de Engenharia de Coimbra, Coimbra, 2015. Disponível em: <http://files.isec.pt/DOCUMENTOS/SERVICOS/BIBLIO/Teses/Tese_Mest_Cedric-Michael-Santos.pdf>. Acesso em: 19 de novembro de 2018.
- [9] MOLINA, R. A., STEINBERGER-ELIAS M.B. Criando um corpus sobre desastres climáticos com apoio da ferramenta NLTK. Centro de Engenharia, Universidade Federal do ABC (UFABC), Santo André, SP, 2011.
- [10] WEIAND, A. Análise de sentimentos do Twitter com Naïve Bayes e NLTK. PUC, Rio Grande do Sul, 2016.
- [11] SANDERS, Niek J. Twitter Sentiment Corpus. Sanders Analytics. Sanders Analytics LLC., 2011.
- [12] JURAFSKY, D. and Martin, J. H. (2000). Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition. Prentice Hall PTR, 1st edition.
- [13] CAPOBIANCO, K. R. Avaliação da etapa de pré-processamento na Mineração de Texto em Redes Sociais Digitais. Universidade Estadual de Londrina, Londrina-PR, 2016.
- [14] GONÇALVES, T. et al. Analysing part-of-speech for portuguese text classification. In: Computational Linguistics and Intelligent Text Processing. [S.l.]: Springer, 2006. p. 551–562.

[15] JACKSON, P.; MOULINIER, I. Natural language processing for online applications: Text retrieval, extraction and categorization. [S.l.]: John Benjamins Publishing, 2007.

[16] ANDRADE, A. M. Descoberta de Relacionamentos Semânticos Não Taxonômicos entre Termos Ontológicos. Universidade Federal de São Carlos, São Carlos-SP, 2017.

[17] SANG, E. F. T. K.; BUCHHOLZ, S. Introduction to the conll-2000 shared task: Chunking. In: ASSOCIATION FOR COMPUTATIONAL LINGUISTICS. Proceedings of the 2nd workshop on Learning language in logic and the 4th conference on Computational natural language learning-Volume 7. [S.l.], 2000.

[18] CHOI, J. D. It Depends: Dependency Parser Comparison Using A Web-based Evaluation Tool. Emory University, Emory University, Atlanta, USA, 2015.