UNIVERSIDADE FEDERAL DO CEARÁ

CAMPUS DE QUIXADÁ

CURSO DE GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

LETÍCIA SARAIVA CHAVES

UTILIZANDO UM MODELO *TRANSFORMER* NO PROCESSO DE IDENTIFICAÇÃO DE ENTIDADES NOMEADAS EM TEXTOS CRIMINAIS

QUIXADÁ

2021

LETÍCIA SARAIVA CHAVES

UTILIZANDO UM MODELO *TRANSFORMER* NO PROCESSO DE IDENTIFICAÇÃO DE ENTIDADES NOMEADAS EM TEXTOS CRIMINAIS

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Ciência da Computação do Campus de Quixadá da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Ciência da Computação.

Orientador: Prof. Dr. Regis Pires Magalhães

QUIXADÁ

2021

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca Universitária
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

C439u Chaves, Letícia Saraiva.
Utilizando um modelo Transformer no processo de identificação de entidades nomeadas em textos criminais / Letícia Saraiva Chaves. – 2021.
61 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Campus de Quixadá, Curso de Ciência da Computação, Quixadá, 2021.
Orientação: Prof. Dr. Regis Pires Magalhães.

1. Processamento de linguagem natural. 2. Reconhecimento de entidade nomeada. 3. Redes neurais (Computação). 4. Aprendizagem profunda. I. Título.

CDD 004

LETÍCIA SARAIVA CHAVES

UTILIZANDO UM MODELO *TRANSFORMER* NO PROCESSO DE IDENTIFICAÇÃO DE ENTIDADES NOMEADAS EM TEXTOS CRIMINAIS

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Ciência da Computação do Campus de Quixadá da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Ciência da Computação.

Aprovada em: __/__/____.

BANCA EXAMINADORA

Prof. Dr. Regis Pires Magalhães (Orientador)
Universidade Federal do Ceará (UFC)

Profª. Drª. Maria Viviane de Menezes
Universidade Federal do Ceará (UFC)

Profª. Drª. Ticiana Linhares Coelho da Silva
Universidade Federal do Ceará (UFC)

Profª. Ma. Lívia Almada Cruz
Universidade Federal do Ceará (UFC)

À toda minha família, em especial minha mãe, minha tia Fabiana e meus avós, Maria José e Gerardo Araújo.

AGRADECIMENTOS

Agradeço a minha mãe, Genair, e ao meu pai, Neywellington, por todo incentivo, apoio, amor, confiança e esforço. À minha irmã, Larissa, por também sempre estar comigo e ao meu irmão, Lorenzo Miguel, por ser o motivo do meu sorriso mesmo em meio ao caos. Amo vocês incondicionalmente. Essa conquista é nossa.

Agradeço a pessoa mais fascinante e incrível que já conheci, meu namorado, Johnny Marcos, por todo amor, parceria, companheirismo, apoio, incentivo, carinho e compreensão durante todos esses anos. Obrigada por estar comigo em todos os momentos. Amo você.

À minha tia Fabiana, que é como uma mãe para mim, e meus avós, Maria José e Gerardo Araújo, obrigada pelo amor e apoio de sempre e também por todos os esforços para que esse sonho se tornasse real. Essa conquista também é de vocês. Amo vocês imensamente.

Aos amigos que conquistei durante esses anos e fizeram essa jornada mais leve, em especial ao Marcelo Martins, Camila Diógenes, Gabriel Uchôa, Robertty Freitas, Carlos Alberto, Claro Henrique, Alessandro Sousa, Matheus Xavier, Lázaro Henrique, Ricardo Lopes, Stherfany Alves, Rayanne Queiroz, Gabriel Andrade e Gustavo Ivens. Desejo todo sucesso para vocês.

Aos meus queridos amigos Rhavana Façanha, Lísia Cavalcante e Átila Rodrigues que sempre torceram por mim. Obrigada pela amizade, conversas e apoio durante todos esses anos. Amo vocês e sou muito grata por tê-los.

Agradeço ao professor Regis Pires, pela orientação e conhecimentos compartilhados. Às professoras Ticiana Linhares, Maria Viviane e Lívia Almada pela disponibilidade em participar da banca e pelas contribuições significantes para este trabalho.

Agradeço a todos que compõem o corpo docente e servidores da UFC Quixadá, em especial a professora Carla Ilane, obrigada pelo apoio, ensinamentos e oportunidades.

Agradeço aos professores Paulo Rego e Gilvan Maia, pelos ensinamentos, confiança, oportunidades e conversas.

Agradeço também a Bárbara Neves, pela disponibilidade, apoio e por todo o auxílio prestado à mim durante a produção deste trabalho. Bárbara, você é uma pessoal excepcional. Meu muito obrigada.

Por fim, agradeço a todos que contribuíram de alguma forma na minha formação profissional e pessoal.

“O sucesso nasce do querer, da determinação e persistência em se chegar a um objetivo. Mesmo não atingindo o alvo, quem busca e vence obstáculos, no mínimo fará coisas admiráveis.”

(José de Alencar)

RESUMO

A identificação de informações importantes facilita a análise de documentos, como registros criminais. A Secretaria de Segurança Pública e Defesa Social do Ceará divulga, mensalmente, suas estatísticas separadas por indicadores criminais, para o acompanhamento do progresso da criminalidade e da violência. Um dos principais indicadores é o de Crimes Violentos Letais e Intencionais (CVLI), que têm relatórios diários divulgados com informações importantes. A extração de informações relevantes utiliza uma técnica de Processamento de Linguagem Natural chamada Reconhecimento de Entidades Nomeadas (NER do inglês *Named Entity Recognition*), que consiste em localizar e classificar entidades nomeadas em textos. Os modelos NER normalmente são baseados em redes neurais convolucionais ou recorrentes. Porém, os *Transformers* estão se tornando um novo paradigma da NER. O modelo *Transformer* facilita a paralelização durante o treinamento, com isso, permite o treinamento em conjuntos de dados maiores do que era possível antes de ser introduzido. A partir disso, foram desenvolvidos sistemas pré-treinados, como BERT (*Bidirectional Encoder Representations from Transformers*), treinado com um enorme conjunto de dados de linguagem geral, e pode ser ajustado para tarefas específicas de linguagem, como NER. Este trabalho propõe a produção de um modelo NER utilizando dados estratificados, capaz de reconhecer entidades nomeadas em textos CVLI, com o modelo *Transformer* pré-treinado em português. O modelo obteve bons resultados, a métrica *F1-score* obteve o valor de 71,44% e a Acurácia Balanceada de 66,73%, superando os *baselines*.

Palavras-chave: Processamento de linguagem natural. Reconhecimento de entidade nomeada. Redes neurais. Aprendizagem profunda.

ABSTRACT

Identifying essential information makes it easier to analyze documents such as criminal records. The Secretariat of Public Security and Social Defense of Ceará publishes monthly its statistics separated by criminal indicators to monitor the progress of crime and violence. One of the main indicators is the Lethal and Intentional Violent Crimes (CVLI in the Portuguese acronym), which have daily reports released with important information. The extraction of relevant information uses a Natural Language Processing technique called Named Entity Recognition (NER), which involves locating and classifying named entities in texts. NER models are usually based on convolutional or recurrent neural networks. However, Transformers are becoming a new paradigm of NER. The Transformer model facilitates parallelization during training, enabling training on larger data sets than was possible before it was introduced. From this, pre-trained systems were developed, such as BERT (Bidirectional Encoder Representations from Transformers), trained with a massive set of general language data, and can be tuned for language-specific tasks, such as NER. This work proposes the production of a NER model using stratified data, capable of recognizing named entities in CVLI texts, with the Transformer model pre-trained in Portuguese. The model obtained good results. The metric F1-score obtained the value of 71.44% and the Balanced Accuracy of 66.73%, surpassing the baselines.

Keywords: Natural language processing. Entity named recognition. Neural network. Deep learning.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

CNN	*Convolutional Neural Networks*
RNN	*Recurrent Neural Network*
LSTM	*Long Short-Term Memory*
NER	*Named Entity Recognition*
BiLSTM	*Bidirectional Long Short-Term Memory*
BERT	*Bidirectional Encoder Representations from Transformers*
CRF	*Conditional Random Field*
EI	Extração da Informação
SSPDS/CE	Secretaria da Segurança Pública e Defesa Social do Ceará
CVLI	Crimes Violentos Letais e Intencionais
CVP	Crimes Violentos Contra o Patrimônio
PLN	Processamento de Linguagem Natural
MLP	*Multilayer Perceptron*
HNERD	*Human Named Entity Recognition with Deep Learning*
AB	Acurácia Balanceada
GPU	*Graphics Processing Unit*

SUMÁRIO

1 INTRODUÇÃO

A utilização contínua da tecnologia no dia a dia das pessoas permite que uma grande quantidade de documentos seja gerada, tornando difícil acompanhar a quantidade massiva de informações disponibilizadas (AIRES *et al.*, 2017). É possível extrair dados importantes desses documentos utilizando a Extração da Informação (EI), que é o processo de obtenção de informação relevante (dados estruturados) com base em fontes que não são interpretadas de forma direta por máquina, como textos (MAYNARD *et al.*, 2016).

A EI objetiva identificar informações significativas de um documento, ou de uma coleção de documentos, com textos em linguagem natural. A identificação destas informações importantes torna a manipulação e análise dos documentos, de forma automática, mais fácil, através de um grupo de padrões e regras de extração (COWIE; LEHNERT, 1996).

Através da EI, a análise de documentos, como por exemplo, registros criminais é facilitada. Com isso, são apresentadas mensalmente as estatísticas da Secretaria da Segurança Pública e Defesa Social do Ceará (SSPDS/CE) no site oficial[1], separadas por indicadores criminais, com o propósito de acompanhar o progresso da criminalidade e da violência. Os dados utilizados para a construção das estatísticas são oriundos da combinação de diferentes fontes (CEARÁ, 2020).

Dentre os principais indicadores criminais destas estatísticas, divulgados oficialmente pela SSPDS/CE, está o Crimes Violentos Letais e Intencionais (CVLI). O indicador CVLI reúne as categorias de homicídios dolosos/feminicídios, lesões corporais e latrocínios. Outros exemplos de indicadores criminais publicados pela SSPDS/CE são: Crimes Violentos Contra o Patrimônio (CVP), Apreensão de entorpecentes ou armas de fogo, Furto e Crimes Sexuais (CEARÁ, 2020).

São divulgados relatórios diários das estatísticas de CVLI com informes cruciais, como natureza do fato e tipo da arma utilizado pelo(s) agressor(es), além do nome da vítima. Através da técnica de Reconhecimento de Entidades Nomeadas (NER do inglês *Named Entity Recognition*) é possível extrair informações importantes desses documentos. O NER é uma abordagem de Processamento de Linguagem Natural (PLN), que consistem em identificar entidades nomeadas em um texto e classificá-las de acordo com rótulos pré-definidos, por exemplo, localização e nomes próprios (LI *et al.*, 2020).

O NER é uma tarefa primordial na área de EI e tem sido amplamente utilizado através de métodos de *Deep Learning* (aprendizagem profunda), como aprendizagem supervisionada

[1] https://www.sspds.ce.gov.br/estatisticas-2-3/

para classificar entidades como pessoa, lugar, organização, ou ainda, doenças e genes, em textos em geral e resumos das áreas médicas e biológicas (CHINCHOR *et al.*, 1993; SHEN *et al.*, 2017).

Existem inúmeras abordagens com domínios diferentes onde o NER pode ser utilizado, embora haja dificuldade de adaptar um modelo NER para diversos idiomas e para domínios que possuem mais categorias do que o conjunto padrão de rótulos (MA; XIA, 2014). O trabalho de Fukuda *et al.* (1998) propõe um método de extração de nomes de genes e de proteínas em artigos publicados. Já o trabalho de Ganti *et al.* (2008) enfoca o problema de categorizar entidades extraídas em coleções de documentos não estruturados, analisando não apenas o contexto do documento local no qual as entidades ocorrem, mas toda a coleção de textos. E o trabalho de Liu *et al.* (2019) busca reconhecer e extrair corretamente entidades nomeadas, como nomes de doenças, medidas médicas e terapias, a partir de dados de diagnóstico médico online.

Deste modo, este trabalho visa a extração de informações importantes de textos de boletins de ocorrência de CVLI. Diferente dos trabalhos citados anteriormente, este trabalho propõe a criação de um modelo NER de *Deep Learning* capaz de reconhecer entidades nomeadas em textos de boletins de ocorrência de CVLI utilizando dados estratificados e arquitetura *Transformer*.

1.1 Objetivos

Nesta seção, serão apresentados o objetivo geral e os objetivos específicos deste trabalho.

1.1.1 Objetivo geral

O objetivo geral deste trabalho é produzir um modelo NER de *Deep Learning* utilizando dados estratificados aptos a reconhecerem entidades nomeadas em textos CVLI conforme rótulos preestabelecidos.

1.1.2 Objetivos específicos

- Classificar os textos CVLI de acordo com as classes definidas no domínio criminal.
- Disponibilizar um modelo NER de *Deep Learning* na língua portuguesa adaptado para o contexto de boletins de ocorrência utilizando BERT.

- Identificar as melhores métricas para avaliar o desempenho do modelo.
- Comparar o desempenho entre diferentes modelos do estado da arte usados para tarefas NER.

1.2 Organização

O restante deste trabalho está estruturado da seguinte maneira: no Capítulo 2 são apresentados conceitos importantes para o entendimento deste trabalho. Os trabalhos relacionados com este trabalho são apresentados no Capítulo 3. No Capítulo 4 é abordada a metodologia utilizada neste trabalho. O Capítulo 5 apresenta os resultados e, por fim, o Capítulo 6 apresenta as conclusões e trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo, serão apresentados os principais conceitos associados a este trabalho.

2.1 Processamento de Linguagem Natural

A linguagem natural é uma das ferramentas mais complexas usadas pelos seres humanos por várias razões, por exemplo, para se comunicar com outras pessoas, para expressar pensamentos, sentimentos, para fazer perguntas ou para dar instruções. Portanto, é crucial que os computadores possuam a capacidade de usar a mesma ferramenta para interagir efetivamente com os seres humanos.

Esta é uma área muito ativa no âmbito da pesquisa e do desenvolvimento, logo, não há uma única definição estabelecida que satisfaça a todos, mas existem alguns aspectos em comum entre as definições. Liddy (2001) define Processamento de Linguagem Natural como sendo uma variedade de técnicas computacionais para analisar e representar textos que ocorrem naturalmente, com o objetivo de obter processamento de linguagem semelhante ao humano para uma variedade de tarefas ou aplicativos.

Geralmente, o processamento de linguagem natural utiliza algoritmos baseados em aprendizado de máquina, que conseguem aprender e compreender a linguagem humana a partir de dados de treinamento passados para o modelo. Existem várias tarefas de PLN como, por exemplo, tokenização, análise de dependência, remoção de *stopwords* e reconhecimento de entidades nomeadas (JACKSON; MOULINIER, 2007). O presente trabalho utilizará o reconhecimento de entidades nomeadas para extração de entidades de textos criminais.

2.1.1 Reconhecimento de Entidades Nomeadas

Reconhecimento de Entidades Nomeadas (NER, sigla do termo em inglês *Named Entity Recognition*) é a tarefa de EI para identificar e classificar menções de locais, quantidades, valores monetários, organizações, pessoas e outras entidades nomeadas em um texto (NADEAU; SEKINE, 2007).

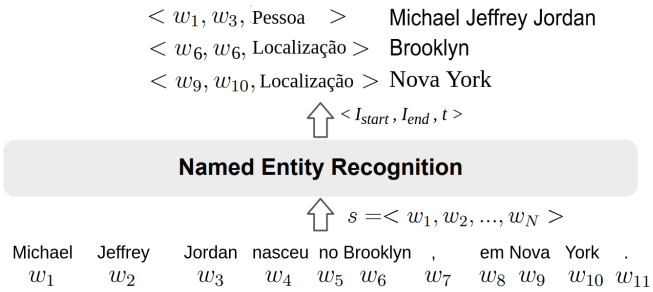
O principal propósito do NER é extrair as informações cruciais de todas as entidades mencionadas no documento. E baseia-se em duas tarefas essenciais: primeiro a identificação dos *tokens* em um texto e depois a classificação para alocar cada *token* em uma categoria específica (SPECK; NGOMO, 2014). É possível adaptar um modelo NER a um contexto específico

adicionando um conjunto de entidades nomeadas definidas, porém obter um ótimo desempenho com esse modelo ainda é uma tarefa desafiadora (MA; XIA, 2014).

Exemplos de entidades nomeadas são nomes de organizações, pessoas e locais no domínio geral; nomes de genes, proteínas, drogas e doenças no domínio biomédico. NER é o processo de localização e classificação de entidades nomeadas no texto em categorias de entidades predefinidas (LI *et al.*, 2020).

Formalmente, dada uma sequência de *tokens* $s = (w_1, w_2, ..., w_N)$, um modelo NER produz uma lista de tuplas (I_{start}, I_{end}, t), onde cada tupla é uma entidade nomeada mencionada em s. Aqui, $I_{start} \in [1, N]$ e $I_{end} \in [1, N]$ são os índices inicial e final de uma menção de entidade nomeada e t é o tipo de entidade de um conjunto de categorias predefinido (LI *et al.*, 2020). A Figura 1 mostra um exemplo em que um sistema NER reconhece três entidades nomeadas de uma determinada frase.

Figura 1 – Exemplo do reconhecimento de entidades nomeadas em uma frase



Fonte: Adaptado de Li *et al.* (2020)

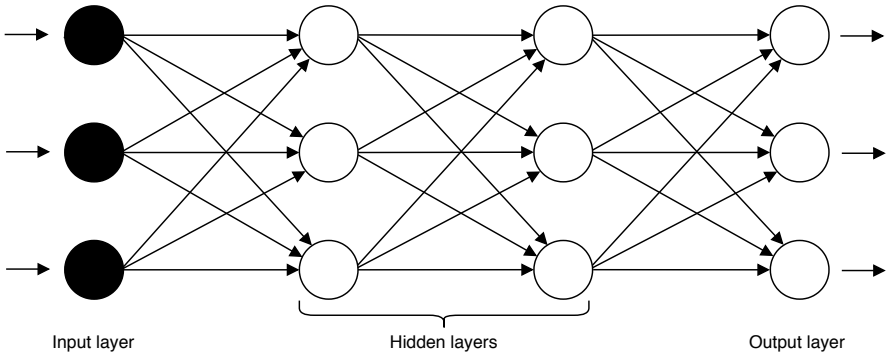
NER não atua apenas como uma ferramenta para extração de informações, mas desempenha um papel essencial em uma variedade de tarefas de PLN, como compreensão de texto, recuperação de informação, resumo automático de texto, tradução automática. Com isso, o presente trabalho faz o uso de uma ferramenta para tarefas de classificação NER, visando criar um modelo NER para reconhecimento de entidades em textos CVLI.

2.2 *Deep Learning*

Deep Learning, também conhecido como Aprendizagem Profunda em português, é uma subárea do Aprendizado de Máquina e diz respeito a aprendizagem a partir de dados utilizando camadas sucessivas de representações de dados. O *Deep Learning* emprega algoritmos inspirados na estrutura e função do cérebro, denominados redes neurais artificiais (CHOLLET,).

Para processar informações, por exemplo, o *Deep Learning* utiliza camadas de neurônios matemáticos em uma rede que possui uma camada de entrada (primeira camada), uma camada de saída (última camada) e camadas ocultas (todas as camadas entre as camadas de entrada e saída). Cada camada, normalmente, é um algoritmo simples e uniforme contendo um tipo de função de ativação. A informação é passada através de cada camada, com a saída da camada anterior fornecendo entrada para a próxima camada, como mostrado na Figura 2 (TALON, 2020).

Figura 2 – Arquitetura de uma rede de *Deep Learning*



Fonte: Adaptado de Jain *et al.* (1996)

Nos últimos anos, surgiram várias arquiteturas e modelos com diferentes combinações de técnicas, como, por exemplo, árvores de decisão, regressão logística e *clustering*, e estes facilitaram a criação de arquiteturas de *Deep Learning* aplicadas à tarefas de reconhecimento de entidades nomeadas que vêm avançando constantemente no estado da arte (SHEN *et al.*, 2017).

Dessa forma, o presente trabalho utiliza os modelos de *Deep Learning* de Redes Neurais Convolucionais e BiLSTM-CRF, ambos são utilizados como *baselines*. E, por fim, o modelo BERT, que é proposto neste trabalho.

2.3 Representação textual

As entradas passadas para os modelos de *Deep Learning* são vetores (matrizes de números), pois os modelos são incapazes de processar *strings* em sua forma bruta. Então, ao trabalhar com texto, a primeira tarefa é converter esse texto em números para alimentar o modelo, esse processo também é chamado de vetorizar o texto.

Existem várias estratégias para realizar a vetorização dos textos, algumas delas são: codificações *one-hot*, onde o texto é representado por uma matriz de valores binários. Cada palavra é associada a um índice em um vetor com tamanho N, onde N é o tamanho do vocabulário

(palavras sem repetições existentes no texto). O vetor é preenchido com 0, em todas as posições, com exceção de um único 1 no índice da palavra em questão. Por exemplo, a Figura 3 mostra a codificação *one-hot* do texto ”O gato deitou no tapete”:

Figura 3 – Exemplo de codificação *one-hot*

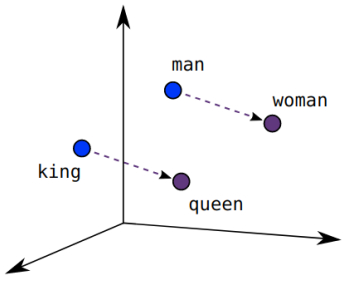
O	=	1	0	0	0	0
gato	=	0	1	0	0	0
deitou	=	0	0	1	0	0
no	=	0	0	0	1	0
tapete	=	0	0	0	0	1

Fonte: Elaborado pela autora.

Outra estratégia de vetorização é a de *word embeddings*, que são representações de palavras de forma vetorial, e basicamente, são técnicas de identificação de semelhanças entre palavras de um vocabulário, mapeadas em um espaço vetorial (CHOLLET,). Isso é feito associando um vetor numérico a cada palavra em um dicionário, de forma que a distância entre quaisquer dois vetores captura parte da relação semântica entre as duas palavras associadas.

Os *word embeddings* fornecem uma maneira de usar uma representação eficiente e densa, onde palavras semelhantes possuem uma codificação semelhante, enquanto os vetores obtidos com a codificação *one-hot* são esparços. A Figura 4 mostra uma aplicação real de *word embeddings* em que os vetores representam a relação semântica entre gêneros.

Figura 4 – Representação de *word embeddings*



Fonte: (DEVELOPERS, 2020)

2.4 Redes Neurais Convolucionais

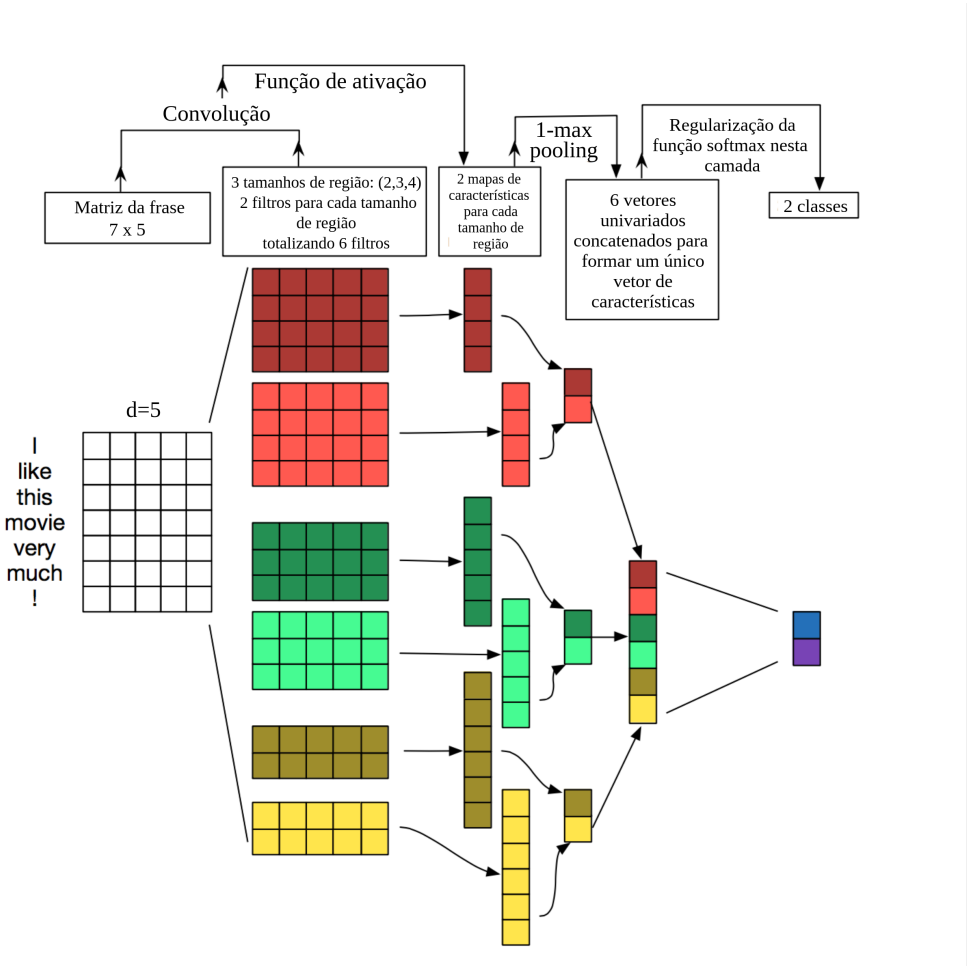
Uma Rede Neural Convolucional (do Inglês, *Convolutional Neural Network* (CNN)) é um algoritmo de *Deep Learning* utilizado na área de visão computacional bastante eficaz no reconhecimento e classificação de imagens, entretanto, vem sendo aplicada em problemas de

PLN, como análise e classificação de textos (TALON, 2020).

A CNN foi desenvolvida originalmente para analisar imagens e consiste basicamente em aplicar filtros em uma imagem original, dada como entrada, para extrair propriedades visuais que podem ser utilizadas para classificar essa imagem. Esses filtros são matrizes numéricas utilizadas para evidenciar as características visuais de uma pequena área de uma imagem.

No entanto, Lopez e Kalita (2017) afirmam que as CNNs também têm se mostrado eficazes para tarefas de classificação de frases e para capturar semelhanças em modelos de vinculação de entidades. Em vez de *pixels* da imagem, a entrada para uma CNN, para tarefas de PLN, é composta por frases ou documentos representados como uma matriz, onde cada linha dessa matriz representa uma palavra de forma vetorial. A dimensionalidade da matriz vai depender do tamanho da frase e da dimensionalidade dos vetores das palavras. A Figura 5 mostra uma arquitetura de uma CNN que contém como exemplo de entrada a sentença "*I like this movie very much!*", tendo, a matriz de entrada, dimensionalidade 7×5.

Figura 5 – Arquitetura de uma CNN para classificação de sentenças



Fonte: Adaptado de Zhang e Wallace (2015)

Na Figura 5, a rede recebe a matriz de entrada, em seguida, a segunda camada, que contém 6 filtros, realiza uma convolução nesta matriz onde cada filtro gera mapas de características com comprimento variável. A próxima camada é executada em cada mapa gerado na camada anterior, que é a camada *1-max pooling* e gera um vetor de características para a penúltima camada, que é a camada *softmax*. Essa camada recebe o vetor de características como entrada e utiliza-o para classificar a frase, que neste caso, é uma classificação binária, então tem dois estados de saída possíveis (ZHANG; WALLACE, 2015).

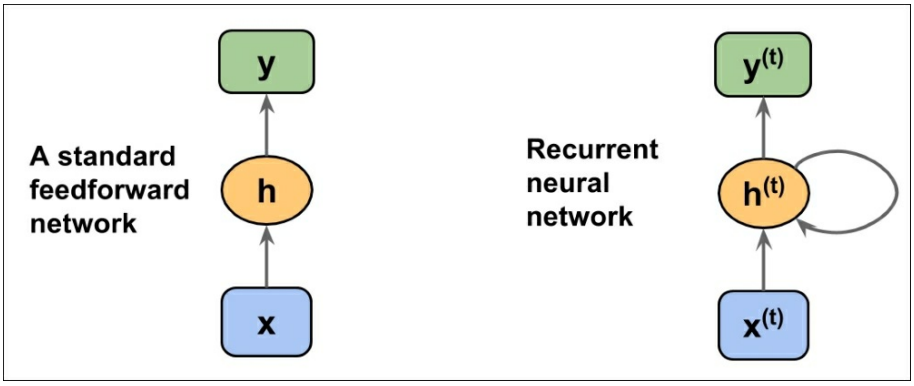
Dentre as principais vantagens de utilizar CNN com textos está a versatilidade das arquiteturas das redes neurais, além de ser uma rede que ajuda a lidar com as palavras fora do vocabulário de uma tarefa NER, portanto, este trabalho faz uso de uma CNN com o objetivo de classificar entidades em textos CVLI, como *baseline*.

2.5 Redes *Long Short-Term Memory* (LSTM)

Redes Neurais Recorrentes (do Inglês, *Recurrent Neural Networks* (RNN)) são essenciais para tarefas de NLP. LSTM é uma variação de RNN e é capaz de classificar, processar e prever séries temporais com intervalos de tempo indeterminado (TALON, 2020).

As redes neurais tradicionais não têm capacidade de guardar informações, e isso dificulta suas aplicações na resolução de diversos problemas. As RNNs são redes com ciclos que possibilitam a persistência das informações. A Figura 6 mostra uma rede neural *feedforward* padrão, que é o tipo mais comum de rede neural em aplicações práticas, e uma RNN, lado a lado para comparação. Ambas as redes têm apenas uma camada oculta. A camada de entrada (x), a camada oculta (h) e a camada de saída (y) são vetores que contêm muitas unidades.

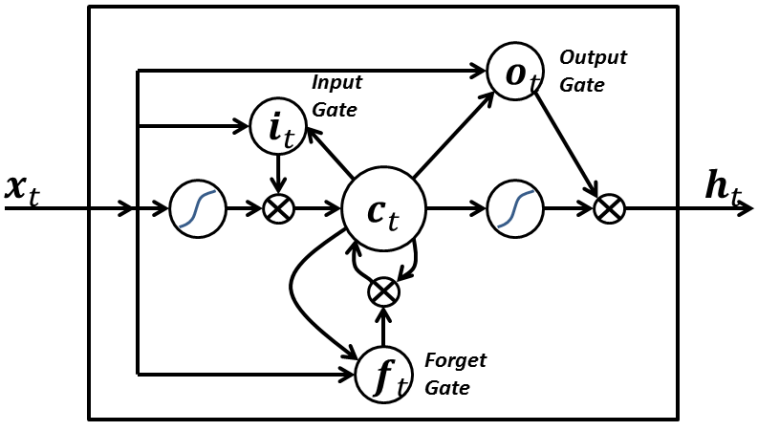
Figura 6 – Comparação entre uma rede neural *feedforward* padrão e uma RNN



Fonte: Raschka e Mirjalili (2017)

Uma RNN não garante que todo o estado da sequencia seja levado a próxima camada. Portanto, os modelos RNN não são adeptos da modelagem de conexões de longo prazo do passado. Uma rede LSTM, que é uma modificação da RNN, tem um estado que atua como sua memória e internamente, esse estado decide o que manter, ou não, na memória. Esses tipos de unidades são muito eficientes na captura de dependências de longo prazo (HUANG *et al.*, 2015). A Figura 7 mostra a célula de uma rede LSTM.

Figura 7 – Célula de uma rede LSTM

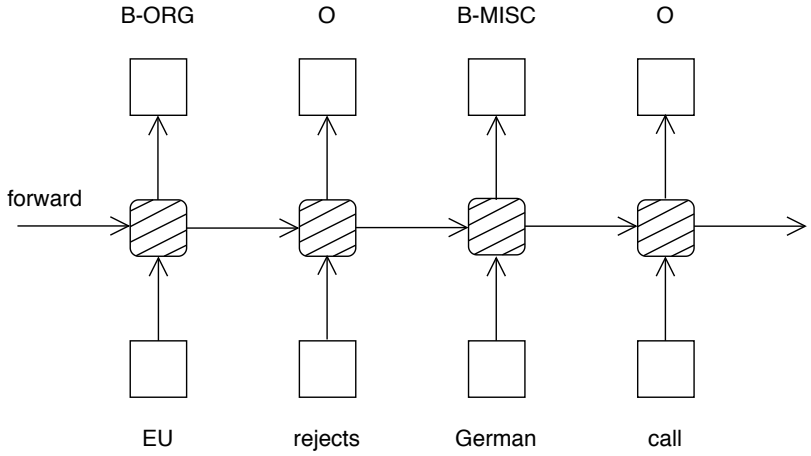


Fonte: Adaptado de Greff *et al.* (2016)

Uma rede LSTM pode remover ou adicionar informações ao estado da célula através de portões. Os portões são divididos em três: de entrada, que serve para atualizar o estado da célula, de saída, que decide de forma condicional o que imprimir na memória, e de esquecimento, que decide quais informações devem ser descartadas ou mantidas, representados respectivamente por i, o e f na Figura 7. Cada um desses portões calculam uma ativação de uma soma ponderada usando uma função de ativação, por exemplo a *sigmoid*.

A Figura 8 mostra um modelo de uma rede LSTM aplicada ao problema de NER dada como entrada a frase ”*EU rejects German call*”. Na Figura, as caixas na parte inferior formam a camada de entrada, que são representadas por palavras de uma frase em um problema de NER, as caixas tracejadas com cantos arredondados representam células LSTM, que formam a camada oculta da rede, e a camada de saída são as caixas na parte superior da figura, que classifica cada palavra dada como entrada com Outros (O), ou um dos quatro tipos de entidade: Pessoa (PER), Local (LOC), Organização (ORG) e Diversos (MISC).

Figura 8 – Rede LSTM aplicada ao problema de NER



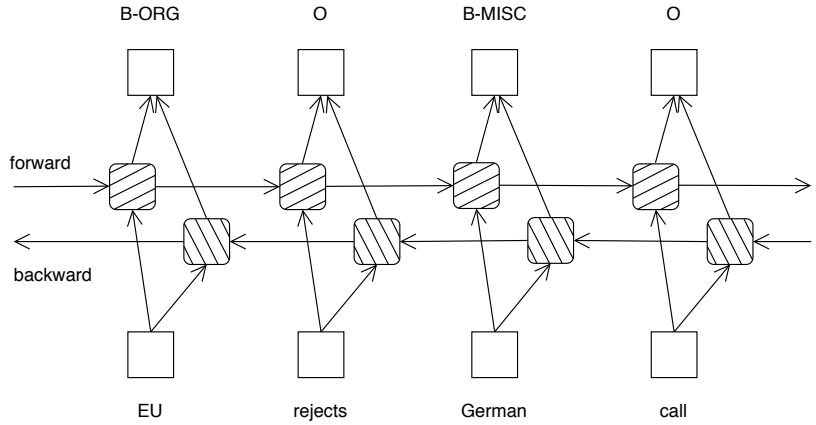
Fonte: Huang *et al.* (2015)

2.5.1 Redes Bidirectional Long Short-Term Memory (BiLSTM)

Nas redes neurais LSTM (unidirecionais), as informações fluem apenas de trás para frente como mostrado na Figura 8, já nas redes BiLSTM a informação flui de trás para frente, como também de frente para trás. Esta bidirecionalidade dos dados faz com que este tipo de rede possa compreender melhor o contexto. Com a rede BiLSTM é possível fazer uso eficiente de recursos passados (por meio de estados avançados) e recursos futuros (por meio de estados anteriores) por um período de tempo específico (SHARFUDDIN *et al.*, 2018).

Este tipo de rede utiliza duas camadas de LSTM, uma camada é responsável pelos estados passados e a outra é responsável pelos estados futuros, como ilustrado na Figura 9, com o mesmo exemplo de NER mostrado na Figura 8. Desta forma, o presente trabalho fará o uso de uma rede BiLSTM para tarefas de classificação NER em textos CVLI, também como *baseline*.

Figura 9 – Rede BiLSTM aplicada ao problema de NER



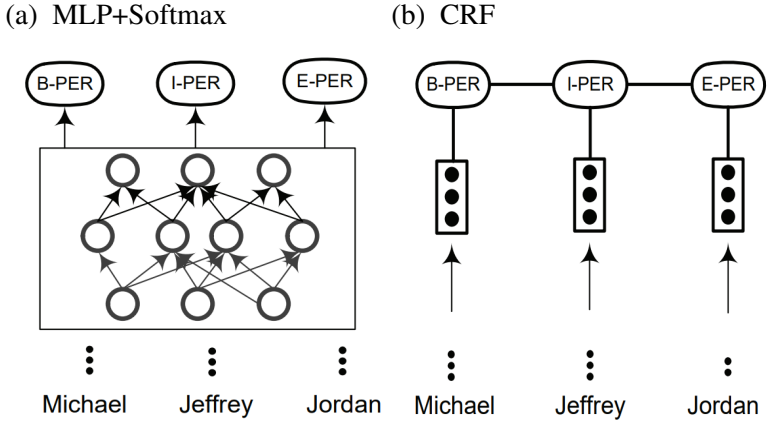
Fonte: Huang *et al.* (2015)

2.6 Arquiteturas de decodificadores de rótulos

O decodificador de rótulos é o estágio final em um modelo NER. Ele pega representações dependentes do contexto como entrada e produz uma sequência de rótulos correspondentes à sequência de entrada.

A Figura 10 mostra duas arquiteturas de decodificadores de rótulos: *Multilayer Perceptron* (MLP) + camada Softmax e *Conditional Random Field* (CRF). Muitos modelos NER usam uma camada CRF como o decodificador de rótulo, é a escolha mais comum (LI *et al.*, 2020).

Figura 10 – Decodificadores de rótulos



Fonte: Li *et al.* (2020)

2.6.1 MLP+Softmax

Um *Perceptron* é um classificador linear, um algoritmo simples destinado a realizar a classificação binária. Já um *Multilayer Perceptron* (MLP) é uma rede neural artificial composta por mais de um *Perceptron*. E é composto por uma camada de entrada, uma camada de saída, e entre elas, um número arbitrário de camadas ocultas que são o verdadeiro mecanismo computacional do *MLP* (TALON, 2020).

O NER é, em geral, formulado como um problema de rotulagem de sequência. Com um MLP e uma camada *Softmax* como a camada decodificadora de rótulo, a tarefa de rotulagem de sequência é lançada como um problema de classificação multi-classe. O rótulo para cada palavra é predito independentemente com base nas representações dependentes do contexto (LI *et al.*, 2020).

2.6.2 Conditional Random Field (CRF)

Conditional Random Field (CRF) é um método de modelagem estatística constantemente aplicado em reconhecimento de padrões e aprendizado de máquina. Muitos modelos de NER baseados em *Deep Learning* usam uma camada CRF como o decodificador de rótulo, por exemplo, no topo de uma camada BiLSTM (ZHENG *et al.*, 2015).

O CRF tem sido muito utilizado em tarefas de NER pelo fato de ser capaz de receber sequências de elementos e definir uma classe para cada elemento. Para tarefas de classificação de sequência é importante considerar as correlações entre as classes nas vizinhanças e decodificar em conjunto a melhor sequência de classes para uma determinada sentença de entrada (LAFFERTY *et al.*, 2001).

Dessa forma, o uso de CRF para tarefas NER é importante, pois ele maximiza a probabilidade de acontecer uma classificação que faça sentido, já que leva em consideração as sequências de classes fornecidas pelo conjunto de treino.

2.7 *Transformer*

O *Transformer* é uma rede neural com arquitetura *encoder-decoder* baseada em um mecanismo de atenção que aprende as relações contextuais entre palavras em um texto. A rede recebe uma sequência de palavras como entrada, codifica-as em representações nas camadas de atenção e as decodifica em palavras novamente (VASWANI *et al.*, 2017).

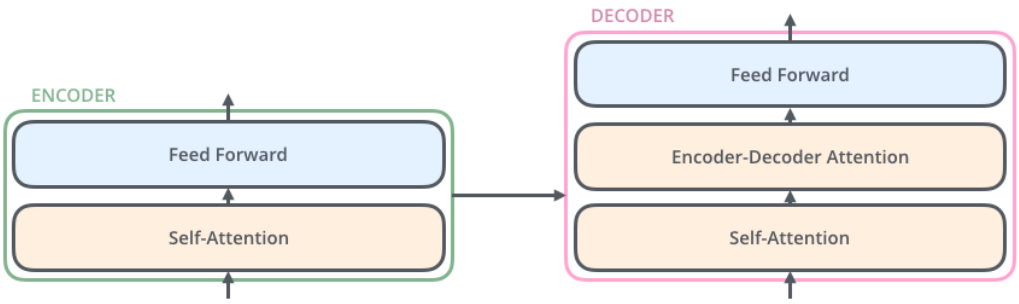
O mecanismo de atenção calcula a pontuação de uma palavra em relação às demais em uma sequência, assim, ele determina o quanto o modelo deve focar em cada posição da sequência ao codificar uma certa palavra. E a autoatenção, é um mecanismo de atenção que relaciona diferentes posições de uma única sequência para computar uma representação da sequência, a ideia é de codificar um *token* como a soma ponderada de seu contexto. (VASWANI *et al.*, 2017).

A Figura 11 mostra a arquitetura dos componentes de uma rede *Transformer*. A rede *Transformer* é dividida em dois componentes: *encoder* e *decoder*. O componente *encoder* é composto por uma pilha de codificadores e cada um é dividido em duas subcamadas, a primeira é um mecanismo de autoatenção, uma camada que ajuda o *encoder* a olhar para outras palavras na frase de entrada conforme codifica uma palavra específica, e a segunda é uma rede neural *feedforward* (VASWANI *et al.*, 2017). A autoatenção permite encontrar correlações entre

diferentes palavras de entrada, indicando a estrutura sintática e contextual da frase.

Já o componente *decoder* possui um pilha de decodificadores, que também são divididos em camadas. Ele possui as duas camadas do *encoder*, mas entre elas tem uma camada de atenção que faz o processamento dos dados recebidos da camada *encoder* (VASWANI *et al.*, 2017).

Figura 11 – Representação da arquitetura dos componentes *encoder* e *decoder*



Fonte: Alammar (2018)

Alguns modelos *Transformers* foram treinados como modelos de linguagem e em grandes quantidades de texto bruto de forma auto-supervisionada. Esses tipos de modelos desenvolvem uma compreensão estatística da linguagem em que foram treinados. Então, o modelo geral pré-treinado passa por um processo chamado aprendizagem por transferência. Durante esse processo, o modelo é ajustado de forma supervisionada em uma determinada tarefa. Esse trabalho utiliza o modelo de linguagem BERT, explicado na Seção 2.8.

2.8 *Bidirectional Encoder Representations from Transformers* (BERT)

Bidirectional Encoder Representations from Transformers (BERT) é um modelo de representação de linguagem projetado para pré-treinar representações bidirecionais profundas de texto não rotulado, condicionando conjuntamente no contexto esquerdo e direito em todas as camadas (DEVLIN *et al.*, 2019).

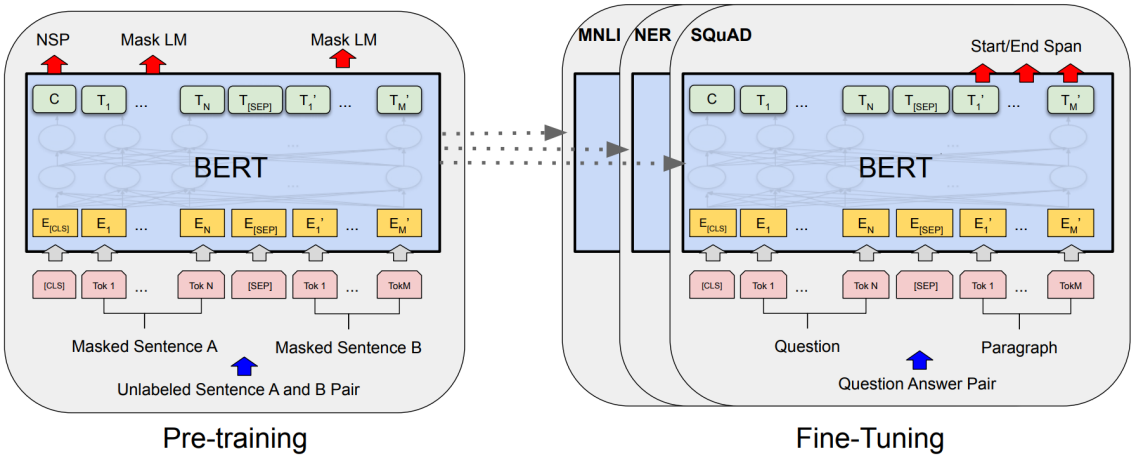
O BERT é baseado na arquitetura do modelo *Transformer*. Em sua forma original, o *Transformer* inclui os dois mecanismos separados, um codificador e um decodificador. Como o objetivo do BERT é gerar um modelo de linguagem, apenas o mecanismo do codificador é usado. Portanto, o BERT tem apenas codificadores *Transformers* empilhados uns sobre os outros.

Existem duas arquiteturas do modelo BERT que são diferenciadas pelo número de camadas *Transformer*, número de núcleos de atenção e tamanho da camada oculta. O (1)

$\text{BERT}_{\text{BASE}}$ contém 12 camadas *Transformer*, 12 núcleos de atenção e tamanho 768 na camada oculta, já o (2) $\text{BERT}_{\text{LARGE}}$ contém 24 camadas *Transformer*, 16 núcleos e tamanho 1024 na camada oculta (DEVLIN *et al.*, 2019).

O treinamento de um modelo BERT envolve duas etapas: o pré-treinamento e o *fine-tuning*, a Figura 12 mostra uma representação de cada uma das etapas, à esquerda, a etapa de pré-treinamento (*pre-training*) e à direita, o *fine-tuning*. A primeira etapa é feita em duas tarefas: *Masked Language Model* (*Mask* LM) e *Next Sentence Prediction* (NSP).

Figura 12 – Representação da etapa de treinamento de um modelo BERT



Fonte: Devlin *et al.* (2019)

A tarefa *Mask* LM objetiva ”mascarar” e prever a palavra mascarada levando em conta o contexto das outras palavras não ocultadas. O BERT mascara cerca de 15% das palavras de uma sentença aleatoriamente. Para mascarar os *tokens*, é utilizado o símbolo [MASC] no lugar do próprio *token*, então o modelo tenta predizer o *token* original observando o contexto. Porém, nem sempre as palavras ”mascaradas” são substituídas pelo *token* [MASC], o que acontece 80% das vezes, outros 10% das vezes a palavra é substituída por outra palavra aleatória e outros 10% a palavra é mantida (DEVLIN *et al.*, 2019).

Já a tarefa NSP recebe como entrada pares de sentenças (A, B), onde o modelo deve identificar se uma determinada sentença B pode ser considerada como subsequente à sentença A ou não. Isso é motivado pelo fato de que, para ter um bom desempenho em algumas tarefas, o modelo precisa codificar relações entre sentenças ou recorrer a informações que estão além do limite da frase (DEVLIN *et al.*, 2019; PILEHVAR; CAMACHO-COLLADOS, 2020). Ao escolher os pares de sentença para cada exemplo, 50% das vezes as sentenças são subsequentes e 50% das vezes B é uma sentença aleatória do *corpus* (DEVLIN *et al.*, 2019).

O BERT deu origem a vários modelos subsequentes, muitos dos quais são na verdade variações do BERT original em termos do objetivo de treino ou do número de parâmetros (PILEHVAR; CAMACHO-COLLADOS, 2020). O BERT pode ser ajustado e usado para diferentes tarefas, como análise de sentimentos, sistema de perguntas e respostas, reconhecimento de entidades nomeadas, classificação de frases e outros, com apenas uma camada de saída adicional.

O *fine-tuning* é o treinamento feito depois que um modelo foi pré-treinado. Para realizar o *fine-tuning*, primeiro é adquirido um modelo de linguagem pré-treinado e, em seguida, é realizado um treinamento adicional com um conjunto de dados específico para a tarefa desejada. O processo de *fine-tuning* é bem mais barato computacionalmente do que o pré-treinamento, necessitando apenas de ajustes para se adaptar a tarefa. Usando BERT, um modelo NER pode ser treinado alimentando o vetor de saída de cada *token* em uma camada de classificação que prevê o rótulo NER. Com isso, o presente trabalho visa utilizar o modelo BERT pré-treinado na língua portuguesa ajustado para tarefas de NER no contexto de boletins de ocorrência.

2.9 Estratificação dos dados

A aprendizagem supervisionada requer dados de treinamento rotulados e, em problemas de classificação, para cada amostra dos dados é atribuída uma classe. Na classificação binária, que normalmente é usada para discutir o desequilíbrio de classes, existem amostras de dados de dois grupos e esse desequilíbrio ocorre quando uma classe contém signitivamente menos amostras do que a outra classe (JOHNSON; KHOSHGOFTAAR, 2019). Pode ser muito difícil para um modelo aprender com esses conjuntos de dados desequilibrados.

Segundo Li *et al.* (2019) muitas tarefas de PLN enfrentam o grave problema de desequilíbrio de dados. Esse desequilíbrio resulta em discrepância da etapa de teste e de treinamento. Sem equilibrar os rótulos, o processo de aprendizagem tende a convergir para um ponto que inclina fortemente para a classe com o rótulo da maioria.

A estratificação de dados divide um conjunto de dados de modo que a proporção de exemplos de cada classe em cada subconjunto seja, aproximadamente, igual a do conjunto de dados completo. Em tarefas de classificação de um único rótulo para cada dado, os grupos são diferenciados com base no valor do atributo alvo. Porém, em tarefas de aprendizagem multi-rótulo para cada amostra, onde existem múltiplas variáveis-alvo, o processo de estratificação é difícil de ser realizado (SECHIDIS *et al.*, 2011).

O *scikit-multilearn* é uma biblioteca *Python* para realizar a classificação de vários rótulos e também permite a estratificação multi-rótulo, fornecendo uma distribuição bem equilibrada dos rótulos (SZYMANSKI; KAJDANOWICZ, 2017). A biblioteca será utilizada no presente trabalho para auxiliar no processo de estratificação dos dados rotulados.

2.10 *Human in the Loop*

Yu *et al.* (2015) afirmam que embora tenha havido um progresso notável no desempenho dos algoritmos de *Deep Learning*, avanços estão faltando na construção do conjunto de dados. Os modelos mais atuais tendem a ser gananciosos por dados, porém, grandes conjuntos de dados de treinamento rotulados, caros e tediosos de produzir, são necessários para otimizar milhões de parâmetros nesses modelos. Os conjuntos de dados disponíveis estão rapidamente se tornando desatualizados.

Para melhorar o treinamento dos algoritmos foi desenvolvida a técnica *Human in the Loop* que melhora o aprendizado de modelos de *Deep Learning* ampliando o esforço humano através de um esquema de rotulagem parcialmente automatizado, aproveitando o aprendizado profundo com humanos no ciclo (YU *et al.*, 2015).

O *Human in the Loop* descreve o processo de quando um sistema é incapaz de resolver um problema e precisa de intervenção humana para fornecer melhores resultados, seja em etapas de treinamento ou testes de desenvolvimento de algoritmos. Tradicionalmente, os humanos rotulam dados para treinamento de um modelo, esses dados são considerados de alta qualidade, e assim, o algoritmo aprende a tomar decisões a partir desses dados.

O *Human in the Loop* faz parte da ferramenta *Human Named Entity Recognition with Deep Learning* (HNERD), na qual modelos de *Deep Learning* aprendem com o auxílio de ações humanas nos dados, e, portanto, será utilizado neste trabalho.

2.11 Métricas de avaliação

Os sistemas NER são geralmente avaliados comparando suas saídas com anotações humanas. As métricas de avaliação informam o grau de acerto nas tarefas de NER. Este trabalho utilizará quatro métricas de avaliação empírica que são amplamente utilizadas para a avaliação da performance de sistemas PLN são elas: *Precision*, *Recall*, *F1-score* e e Acurácia Balanceada. Essas métricas são formuladas através das seguintes variáveis (SAMMUT; WEBB, 2011):

- TP (*True Positive*): Indica que o termo é corretamente identificado e recebe a classificação correta.
- FP (*False Positive*): Quando um termo que não pertence a determinado rótulo, é classificado como tal.
- FN (*False Negative*): Indica que o termo de um determinado rótulo não é classificado como tal.

2.11.1 Precision

Precision refere-se à porcentagem dos resultados do sistema que são reconhecidos corretamente. É a razão entre os termos identificados corretamente (*True Positive*) e a soma dos termos *True Positive* e *False Positive*, definida pela Equação 2.1.

$$Precision = \frac{TP}{TP + FP} \tag{2.1}$$

2.11.2 Recall

Recall refere-se à porcentagem do total de entidades corretamente reconhecidas pelo sistema. Indica com que frequência o classificador está encontrando exemplos de uma classe. É a razão entre os termos *True Positive* e a soma dos termos *True Positive* e *False Negative*, definida pela Equação 2.2.

$$Recall = \frac{TP}{TP + FN} \tag{2.2}$$

2.11.3 F1-Score

Média harmônica das métricas *Precision* e *Recall*, definida pela Equação 2.3. Essa métrica combina o *Precision* e o *Recall* de modo a trazer um número único que indique a qualidade geral do modelo e trabalha bem até com conjuntos de dados que possuem classes desproporcionais.

$$F1\text{-}Score = 2 \times \frac{precision \times recall}{precision + recall} \tag{2.3}$$

2.11.4 Acurácia Balanceada (AB)

A Acurácia Balanceada (AB) é uma métrica que não é influenciada pelo desbalanceamento das classes, visto que ela é calculada de forma a ponderar os valores de uma métrica para cada rótulo utilizando como peso a quantidade de exemplos do rótulo (MOSLEY, 2013).

A AB é definida como a média do *Recall* obtido em cada rótulo na Equação 2.4, onde a quantidade total de rótulos do conjunto R é representado por $|R|$ e o valor de *Recall* obtido do rótulo r é indicado por $Recall(r)$.

$$AB = \frac{1}{|R|} \sum_{r=1}^{|R|} Recall(r) \tag{2.4}$$

3 TRABALHOS RELACIONADOS

Esta seção apresenta os trabalhos relacionados com o presente projeto e sua relação com o trabalho proposto. O trabalho presente na Seção 3.1, apresenta uma arquitetura de rede neural de *deep learning* capaz de resolver o problema de Desambiguação de Rótulos. Já o trabalho presente na Seção 3.2 utiliza o modelo BERT para treinamento com textos da Língua Portuguesa.

A Seção 3.3 apresenta um *framework* que aproveita o poder dos modelos de linguagem pré-treinados (por exemplo, BERT) para melhorar o desempenho de previsão de modelos NER. Já o trabalho presente na Seção 3.4, se trata de um *framework* que permite o usuário realizar algumas tarefas de classificação.

O trabalho da Seção 3.5 propõe um *framework* para facilitar a extração de entidades nomeadas em boletins de ocorrências. A Seção 3.6 apresenta o trabalho que propõe um modelo NER apto a reconhecer entidades nomeadas em textos de roubos. Por fim, na Seção 3.7 é feita uma comparação entre os trabalhos relacionados e este trabalho conforme quatro atributos.

3.1 *Novel Approach for Label Disambiguation via Deep Learning*

Silva *et al.* (2019a) propõe uma arquitetura de rede neural livre de bases de conhecimento, recursos específicos de idioma (por exemplo, dicionários geográficos, *tags* de parte da fala) ou recursos artesanais (por exemplo, padrões de ortografia de palavras e de capitalização) capaz de resolver um problema chamado de Desambiguação de Rótulos.

O modelo precisa lidar com a desambiguação de rótulo ou de classe em relação ao contexto da sentença, pois um modelo pode classificar duas entidades diferentes em uma mesma classe se ambas possuírem ortografias semelhantes.

O trabalho de Silva *et al.* (2019a) utiliza uma rede neural chamada *Char*-BLSTM-CRF, que é composta por uma rede neural BiLSTM e CRF e assemelha-se com o presente trabalho pelo fato de ambos gerarem um modelo NER de *Deep Learning* para textos criminais.

3.2 *Portuguese Named Entity Recognition using BERT-CRF*

No trabalho de Souza *et al.* (2019) foram treinados modelos *Portuguese* BERT e empregado uma arquitetura BERT-CRF para a tarefa NER na língua portuguesa, combinando as capacidades de transferência do BERT com as previsões estruturadas de CRF. Foram avaliadas

diversas arquiteturas neurais utilizando modelos BERT para a tarefa NER em português e comparações de estratégias de treinamento baseadas em recursos e em *fine-tuning*.

Este trabalho assemelha-se com o de Souza *et al.* (2019) pelo motivo de ambos utilizarem o BERT com textos em português, porém, os conjuntos de dados e domínio são diferentes.

3.3 *BOND: BERT-Assisted Open-Domain Named Entity Recognition with Distant Supervision*

O trabalho de Liang *et al.* (2020) propõe uma estrutural computacional chamada BOND, que uma abreviatura de BERT-*Assisted Open-Domain Named Entity Recognition with Distant Supervision*, que aprende identificadores de entidade nomeadas precisos de supervisão distante, que é usada para criar dados rotulados de uma forma (semi-) automática, e aproveita o poder dos modelos de linguagem pré-treinados (por exemplo, BERT e RoBERTa) para melhorar o desempenho de previsão dos modelos NER.

A supervisão à distância é um esquema de aprendizagem em que um classificador é aprendido a partir de um conjunto onde os dados de treinamento são rotulados automaticamente com base em heurísticas ou regras. A supervisão à distância tornou-se o método padrão para extração de relações (QIN *et al.*, 2018).

Liang *et al.* (2020) propoem um algoritmo em dois estágios: (I) o modelo de linguagem pré-treinado é adaptado às tarefas NER usando rótulos distantes, o que pode melhorar as métricas de *recall* e *precision*; (II) os rótulos distantes são eliminados e é proposta uma abordagem de autotreinamento para melhorar ainda mais o desempenho do modelo.

O presente trabalho se assemelha com o de Liang *et al.* (2020) pelo fato de ambos utilizarem o modelo BERT para reconhecimento de entidades nomeadas, porém utilizando abordagens diferentes, este trabalho utiliza uma abordagem tradicional para NER, enquanto o trabalho de Liang *et al.* (2020) utiliza uma abordagem de supervisão à distância para gerar rótulos automaticamente.

3.4 *Improving Named Entity Recognition using Deep Learning with Human in the Loop*

O trabalho de Silva *et al.* (2019b) propõe o HNERD (abreviação de *Human Named Entity Recognition with Deep Learning*) que é uma estrutura interativa para auxiliar o usuário nas

tarefas de classificação NER, desde a criação de um grande conjunto de dados até a construção / manutenção de um modelo NER de *Deep Learning*.

A ferramenta possui uma arquitetura que considera dois tipos de usuários: o revisor, que atribui rótulos aos trechos dos textos que serão usados pelo modelo. E o segundo tipo é o cientista de dados, que executa ações sobre os modelos de *Deep Learning*, como por exemplo realizar o treinamento. A ferramenta incorpora modelos CNN de *Deep Learning* baseados no *framework spaCy*.

O trabalho de Silva *et al.* (2019b) assemelha-se com o presente trabalho devido à utilização de uma ferramenta que auxilia na classificação NER. Além deste trabalho utilizar o modelo CNN do *framework spaCy* como *baseline*.

3.5 Reconhecimento de Entidades Nomeadas em Textos de Boletins de Ocorrências

O trabalho de Araújo (2019) é uma extensão do trabalho de Silva *et al.* (2019b) e propõe o *framework* HNERD com o foco de tornar eficiente o processo de extração de entidades nomeadas em boletins de ocorrências e também dois modelos NER, um utilizando o modelo CNN proposto no *framework spaCy* e um modelo BiLSTM.

Foram construídos modelos para classes ambíguas e não ambíguas. Para classes não ambíguas, o modelo CNN construído utilizando o *framework spaCy* teve um melhor desempenho que o modelo BiLSTM, construído com a biblioteca *Keras*.

Já para classes ambíguas, como é preciso obter informações do contexto no qual a classe está, a arquitetura do modelo CNN não consegue extrair essas informações, fazendo com que o modelo tivesse um desempenho ruim para o problema.

O trabalho de Araújo (2019) se assemelha com este trabalho pela geração de um modelo CNN utilizando o *framework spaCy*. Além de o presente trabalho fazer o uso da ferramenta HNERD, para realizar as classificações dos textos manualmente.

3.6 Aprendizado Profundo para Reconhecimento de Entidades Nomeadas em Narrativas de Roubos

O trabalho de Oliveira (2020) propõe um modelo NER capaz de reconhecer entidades nomeadas em textos de roubos, utilizando técnicas para o desbalanceamento dos dados e *word embeddings* que representam as relações sintáticas e semânticas das narrativas de roubos. O

modelo proposto no trabalho é um modelo que tem como base a arquitetura de rede neural BiLSTM-CRF.

Foram treinados 53 modelos BiLSTM-CRF, variando a arquitetura dos modelos, com adição das *word embeddings* pré-treinadas, ajustadas ou não durante o treino, e utilizando diferentes funções de *loss*, além de tratamento de palavras fora do vocabulário e técnicas para o desbalanceamento de nível arquitetural e dos dados.

O trabalho de Oliveira (2020) assemelha-se com o presente trabalho pela utilização de um modelo BiLSTM-CRF, que será utilizado como *baseline* neste trabalho, e também pela utilização de dados distribuídos de forma estratificada.

3.7 Análise comparativa

O Quadro 1 apresenta um comparativo entre os trabalhos relacionados com este trabalho, destacando as principais semelhanças e diferenças entre eles de acordo com quatro atributos: domínio do NER, conjunto de dados utilizado, rede neural utilizada, e se propõe algum *framework*.

Dos sete trabalhos relacionados listados, quatro são de domínio criminal, e o restante tem domínio variado que consistem em textos de uma ampla variedade de fontes, como conversação de transmissão, transmissão de notícias, notícias, revista, conversa por telefone e texto da web.

O conjunto de dados utilizado nos trabalhos de Silva *et al.* (2019a), Silva *et al.* (2019b) e Araújo (2019) são documentos policiais com relatos de homicídios e no trabalho de Oliveira (2020) são documentos policiais com relatos de roubos, enquanto os trabalhos de *OntoNotes 5.0* e Liang *et al.* (2020) utilizam os dados de CoNLL-2003, *Twitter*, *OntoNotes 5.0* *Wikigold* e *Webpage* e o trabalho de Souza *et al.* (2019) utiliza o *First* HAREM e MiniHAREM.

O próximo atributo indica as redes neurais utilizadas em cada trabalho. Silva *et al.* (2019a) e Oliveira (2020) utilizam rede uma BiLSTM e CRF. Já o trabalho de Araújo (2019) utiliza as mesmas redes em modelos separados. E Souza *et al.* (2019) utiliza uma rede BERT-CRF, enquanto Liang *et al.* (2020) utiliza apenas uma rede BERT. E por fim, o trabalho de Silva *et al.* (2019b) utiliza uma rede CNN.

O último atributo é relacionado a proposta de um *framework*. Apenas os trabalhos de Liang *et al.* (2020), Silva *et al.* (2019b) e Araújo (2019) propõem *frameworks*. O BOND é proposto por Liang *et al.* (2020) e o HNERD é proposto pelos outros dois trabalhos citados

anteriormente.

O símbolo denotado por — no Quadro 1 significa que o trabalho não possui o atributo referente.

Quadro 1 – Comparação entre os trabalhos relacionados e o trabalho proposto.

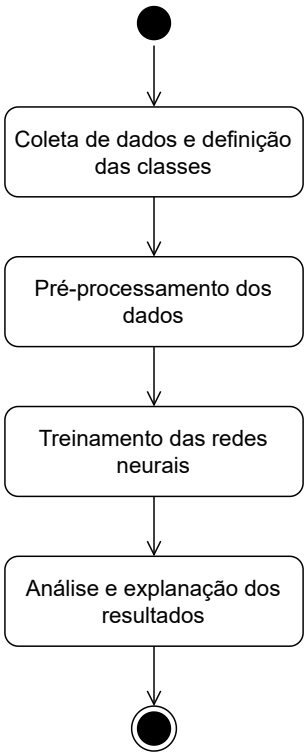
	Domínio do NER	**Conjunto de dados utilizado**	**Rede neural**	***Framework* proposto**
Silva *et al.* (2019a)	Criminal	Documentos policiais sobre homicídios	BiLSTM-CRF	—
Souza *et al.* (2019)	Variado	*First* HAREM e MiniHAREM	BERT-CRF	—
Liang *et al.* (2020)	Variado	CoNLL-2003, *Wikigold*, *OntoNotes 5.0*, *Twitter* e *Webpage*	BERT	BOND
Silva *et al.* (2019b)	Criminal	Documentos policiais sobre homicídios	CNN	HNERD
Araújo (2019)	Criminal	Documentos policiais sobre homicídios	BiLSTM e CNN	HNERD
Oliveira (2020)	Criminal	Documentos policiais sobre roubos	BiLSTM-CRF	—
Trabalho proposto	Criminal	Documentos policiais sobre CVLI	BERT	—

Fonte: Elaborado pela autora.
Nota: O símbolo — denota que o trabalho não apresenta o referido atributo.

4 PROCEDIMENTOS METODOLÓGICOS

Esta seção apresenta os passos necessários para a execução deste trabalho e consiste nas seguintes etapas: coleta de dados e definição das classes, pré-processamento dos dados, em seguida, treinamento das redes neurais, e por fim, análise e explanação dos resultados. A Figura 13 mostra as etapas da metodologia e todas elas são descritas a seguir.

Figura 13 – Etapas do desenvolvimento



Fonte: Elaborado pela autora

4.1 Coleta de dados e definição das classes

A primeira etapa consiste em extrair os textos CVLI da base de dados da SSPDS/CE. Os textos são coletados de Órgãos Governamentais do Estado do Ceará e são dados que possuem confidencialidade.

O próximo passo da primeira etapa é a definição das classes que são utilizadas na Seção 4.2, etapa de pré-processamento. As classes são usadas para identificar as entidades nomeadas nos textos coletados.

4.2 Pré-processamento dos dados

Após a coleta dos dados e com as classes definidas para a classificação dos textos, com o auxílio da ferramenta HNERD, o conjunto de treino é criado a partir dos textos CVLI e contém textos anotados no qual cada palavra ou sentença dos textos possui uma classificação com uma das classes definidas.

Após a classificação, os dados são divididos em conjuntos de treino, validação e teste, de maneira estratificada utilizando a biblioteca *scikit-multilearn*, fazendo com que o conjunto de dados seja dividido de forma que a proporção de cada classe nos conjuntos de treino, validação e teste sejam aproximadamente iguais a de todo o conjunto de dados (SZYMANSKI; KAJDANOWICZ, 2017).

4.3 Treinamento das Redes Neurais

Os *baselines* utilizados para este trabalho foram treinados com os dados coletados utilizando uma rede CNN, proposta pelo *framework spaCy*, e outro modelo utilizando a rede BiLSTM-CRF, que é proposto em Oliveira (2020), com alguns ajustes. E por fim foi treinado também o modelo BERT proposto pelo presente trabalho.

4.3.1 Modelo CNN

O *spaCy* é uma biblioteca de código aberto para PLN, para treinamento da rede convolucional, e oferece modelos de redes neurais para os idiomas inglês, alemão, espanhol, português, francês, italiano, holandês, além de suporte para uma NER multilíngue (SPACY.IO, 2020).

Os modelos do *spaCy* são estatísticos e cada decisão tomada por eles é uma previsão, que é baseada nos exemplos passados para o modelo durante o treinamento. Quando um novo dado, sem classificação, é passado para o modelo, este fará uma previsão. Os exemplos de treinamento são processados em várias iterações e a cada iteração os dados são embaralhados, garantindo que o modelo não faça generalizações baseado na ordem dos dados.

4.3.2 *Modelo BiLSTM-CRF*

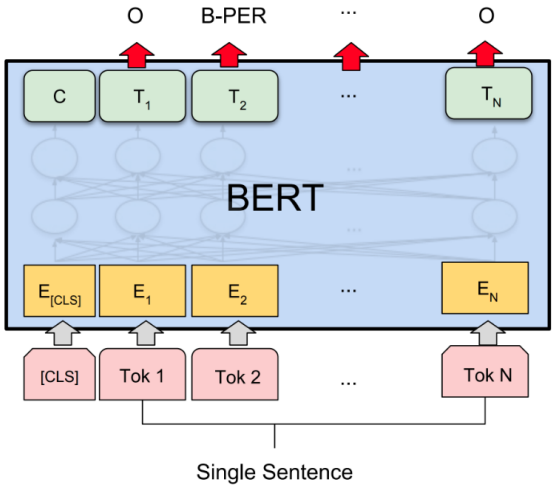
O modelo BiLSTM-CRF tem uma camada de entrada, em seguida uma camada de *embedding*, que fornece uma representação densa de palavras e seus significados relativos, depois uma camada LSTM bidirecional e em seguida uma camada *TimeDistributed*, que é definida como um *wrapper* que permite aplicar uma camada a cada fatia temporal de uma entrada. Por fim, uma camada CRF, como camada de saída, para prever as classes.

4.3.3 *Modelo BERT*

Para o modelo BERT é possível utilizar pesos pré-treinados disponíveis e usá-los para ajustar o modelo para uma tarefa em um conjunto de dados específico. Neste trabalho, é utilizado o modelo pré-treinado em português (BERT *base portuguese cased*[1], que também é conhecido como BERTimbau) para o processo de *fine-tuning* que utiliza a arquitetura $\text{BERT}_{\text{BASE}}$.

A arquitetura do modelo é composta de um modelo pré-treinado BERT seguido de uma camada linear com ativação *Softmax*. A classificação final atribuída para cada *token* é dada pelo rótulo que o modelo atribui maior probabilidade. A Figura 14 mostra o modelo BERT para a tarefa NER.

Figura 14 – Representação do BERT para NER



Fonte: Devlin *et al.* (2019)

[1] Disponível para *download* em https://github.com/neuralmind-ai/portuguese-bert

4.4 Análise e explanação dos resultados

Após o treinamento das redes, cada modelo aprenderá sobre as classes presentes nos textos do conjunto de treino e deverá prever as classes de cada entidade de um novo texto CVLI. Para validar os modelos, um conjunto de textos (conjunto de teste) é passado como entrada para cada modelo. Esses textos são utilizados para que os modelos possam identificar e classificar as entidades nomeadas presentes e retorná-las, juntamente com os valores das métricas calculadas.

Os resultados serão coletados e comparados através das métricas *Precision*, *Recall* e *F1-score*, definidas na Seção 2.11, para cada classe e para uma avaliação de forma geral dos modelos é utilizada a AB e a média ponderada dos resultados das métricas para cada classe. Essas métricas foram escolhidas afim de avaliar a precisão do modelo NER, dado que a avaliação baseada nesses indicadores representa um papel central na estimativa de desempenho de sistemas PLN (GOUTTE; GAUSSIER, 2005).

5 EXPERIMENTOS E RESULTADOS

Este capítulo apresenta os resultados obtidos na execução dos experimentos. Inicialmente, são descritos os processos de obtenção e classificação, estratificação e construção dos modelos. Por fim, a vetorização dos dados, treino dos modelos e os resultados obtidos.

As ferramentas utilizadas neste trabalho foram: a linguagem de programação *Python*, o *Google Colaboratory*, para execução dos experimentos, uma *Graphics Processing Unit* (GPU), para treinar os modelos. Algumas bibliotecas também foram utilizadas, as principais são: *scikit-multilearning*, para estratificação, *spaCy*, *Keras* e *Transformers* para treino dos *baselines* e modelos e um *framework* em *Python* chamado *seqeval*[1] para avaliação dos modelos.

5.1 Obtenção e classificação dos dados

Inicialmente, foram obtidos um total de 1.094 textos CVLI da base de dados da SSPDS/CE. Esses dados foram classificados manualmente utilizando um total de 29 classes. Após realizar uma exploração nos dados do *dataset* foram identificados 80 textos duplicados, que foram excluídos, restando um total de 1.014 textos, do conjunto de dados original, que compõem o conjunto de dados utilizado neste trabalho.

Além das 29 classes que já estavam sendo utilizadas nas classificações dos textos, foram estabelecidas mais 2 classes pelos especialistas da SSPDS/CE, então foi definido um total de 31 classes que foram utilizadas para classificar os textos do *dataset*.

Foi realizada uma nova classificação dos dados, tomando como base as rotulações que já existiam nos textos, e utilizando as 31 classes definidas. Para isso, foi criado um modelo no *framework* HNERD, com o conjunto de dados do *dataset* que possuía apenas os textos, juntamente com as classes estabelecidas para realizar o processo de anotação das classes manualmente para cada texto, utilizando a ferramenta.

5.2 Estratificação dos dados

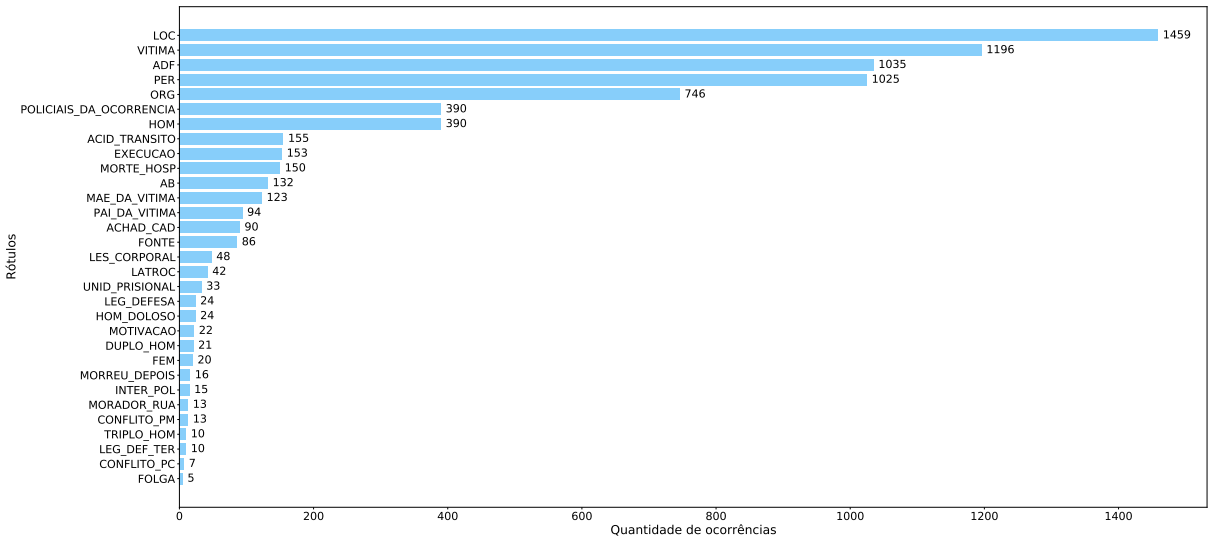
Após a classificação dos textos, foi possível notar que a disposição das classes era bastante desbalanceada, como mostra a Figura 15. O rótulo LOC é o que tem mais ocorrências (1.459), enquanto o rótulo FOLGA tem apenas 5 ocorrências no *dataset*. Esse desbalanceamento pode fazer com que o modelo não obtenha bons resultados, pois, apesar de os exemplos das clas-

[1] Disponível em https://github.com/chakki-works/seqeval

ses de maior proporção serem classificados corretamente, com grande frequência, normalmente os exemplos das classes de menor proporção não são classificados corretamente.

Foi realizada uma divisão estratificada no *dataset*, que foi separado em conjuntos de treino, validação e teste. A estratificação foi realizada utilizando a biblioteca *scikit-multilearn* para garantir a mesma proporção de classes nos conjuntos.

Figura 15 – Quantidade de ocorrências de cada classe



Fonte: Elaborado pela autora.

A biblioteca *scikit-multilearn* fornece uma implementação de estratificação iterativa com o objetivo de proporcionar uma distribuição bem equilibrada de evidências de relações dos rótulos (SZYMANSKI; KAJDANOWICZ, 2017). A divisão começa com a verificação das classes que possuem menos ocorrências e vai seguindo a cada iteração.

O incentivo do algoritmo de estratificação iterativa é: se rótulos que possuem menos ocorrências não forem examinados em prioridade, eles podem ser distribuídos de maneira indesejada, e isso não pode ser consertado em seguida. Por outro lado, com rótulos frequentes, tem-se a chance posteriormente de modificar a distribuição atual para o desejado, devido à disponibilidade de mais exemplos (SECHIDIS *et al.*, 2011).

Para a entrada da biblioteca *scikit-multilearn*, foi feita uma representação vetorial das classes gerando um vetor de 31 posições para cada texto, onde cada posição do vetor representa uma classe com a quantidade de ocorrências existentes na frase. E também foi feita uma representação vetorial dos textos, obtendo as palavras mais relevantes, com a tokenização, remoção de caracteres especiais e *stopwords*. Por fim, foi feito o mapeamento vetorial das palavras dos textos, através do vocabulário que foi construído com o conjunto das palavras

sem repetição do *dataset*, onde as palavras foram representadas pelo índice correspondente no vocabulário.

A divisão ficou da seguinte maneira: conjunto de treino com 70% dos dados (702 textos), conjunto de validação com 10% dos dados (108 textos) e o conjunto de teste com 20% dos dados (204 textos). A Tabela 1 mostra a quantidade de ocorrências e a proporção da divisão (indicada pelo número com o símbolo %) de cada classe nos conjuntos após a divisão estratificada do *dataset*. Todos os conjuntos (treino, validação e teste) possuem exemplos dos 31 rótulos.

Tabela 1 – Quantidade de ocorrências e proporção das classes na distribuição dos dados estratificados

Classes	**Conjuntos de dados**						
	Dataset	**Treino (%)**		**Validação (%)**		**Teste (%)**	
ARMA_BRANCA	132	92	(69,70%)	14	(10,60%)	26	(19,70%)
ACHAD_CAD	90	64	(71,11%)	9	(10,00%)	17	(18,89%)
ACID_TRANSITO	155	114	(73,55%)	14	(9,03%)	27	(17,42%)
ARMA_DE_FOGO	1035	728	(70,34%)	109	(10,53%)	198	(19,13%)
CONFLITO_PC	7	5	(71,44%)	1	(14,28%)	1	(14,28%)
CONFLITO_PM	13	10	(76,92%)	1	(7,70%)	2	(15,38%)
DUPLO_HOM	21	15	(71,44%)	3	(14,28%)	3	(14,28%)
EXECUCAO	153	110	(71,90%)	13	(8,50%)	30	(19,60%)
FEM	20	16	(80,00%)	1	(5,00%)	3	(15,00%)
FOLGA	5	1	(20,00%)	2	(40,00%)	2	(40,00%)
FONTE	86	63	(73,25%)	8	(9,30%)	15	(17,45%)
HOM_DOLOSO	24	17	(70,83%)	3	(12,50%)	4	(16,67%)
HOM	390	278	(71,28%)	37	(9,49%)	75	(19,23%)
INTER_POL	15	12	(80,00%)	1	(6,67%)	2	(13,33%)
LATROC	42	30	(71,44%)	4	(9,52%)	8	(19,04%)
LEG_DEF_TER	10	7	(70,00%)	1	(10,00%)	2	(20,00%)
LEG_DEFESA	24	18	(75,00%)	1	(4,17%)	5	(20,83%)
LES_CORPORAL	48	34	(70,83%)	6	(12,5%)	8	(16,67%)
LOC	1459	1069	(73,27%)	116	(7,95%)	274	(18,78%)
MAE_DA_VITIMA	123	87	(70,73%)	8	(6,50%)	28	(22,77%)
MORADOR_RUA	13	8	(61,54%)	1	(7,70%)	4	(30,76%)
MORREU_DEPOIS	16	10	(62,50%)	2	(12,50%)	4	(25,00%)
MORTE_HOSP	150	103	(68,67%)	17	(11,33%)	30	(20,00%)
MOTIVACAO	22	15	(68,18%)	2	(9,09%)	5	(22,73%)
ORG	746	520	(69,70%)	72	(9,65%)	154	(20,65%)
PAI_DA_VITIMA	94	67	(71,28%)	7	(7,45%)	20	(21,27%)
PER	1025	727	(70,93%)	74	(7,22%)	224	(21,85%)
POLICIAIS_DA_OCORRENCIA	390	279	(71,54%)	36	(9,23%)	75	(19,23%)
TRIPLO_HOM	10	7	(70,00%)	1	(10,00%)	2	(20,00%)
UNID_PRISIONAL	33	22	(66,67%)	3	(9,09%)	8	(24,24%)
VITIMA	1196	847	(70,82%)	103	(8,61%)	246	(20,57%)

Fonte: Elaborado pela autora.

5.3 Modelos implementados

Neste trabalho, foram treinados e avaliados três modelos para NER: um modelo CNN, um modelo BiLSTM-CRF, ambos considerados *baseline* para este trabalho, e um modelo BERT.

5.3.1 Modelo CNN

O *spaCy* apresenta um sistema de reconhecimento de entidade estatística extremamente rápido, que atribui rótulos a sequências de *tokens*. É possível adicionar novas classes e atualizar o modelo com novos exemplos. Este trabalho atualiza o *Wiki*NER, que é o modelo pré-treinado em Português, do *spaCy*, na versão v2.2.5, com os exemplos do conjunto de treino, para utilizar como *baseline*.

Para a maioria das tarefas, o *spaCy* usa uma rede neural baseada na CNN com alguns ajustes. Especificamente para NER é usada uma estrutura de quatro etapas: vetorização, codificação, atenção e predição. Primeiro as palavras são alteradas para representações vetoriais, e, dada uma sequência de vetores dessas palavras, a etapa de codificação calcula uma matriz de frase, levando em consideração o contexto, utilizando uma CNN para codificação. A camada de atenção da CNN reduz a representação da matriz produzida pela etapa de codificação para um único vetor, que é passado para uma rede MLP+*Softmax* para predição.

5.3.2 Modelo BiLSTM-CRF

Uma rede BiLSTM consiste em duas redes LSTM, a primeira leva a entrada na direção para frente e a segunda leva a entrada para trás. Combinar as saídas das duas redes produz um contexto que fornece informações sobre as amostras que cercam cada *token* individual. A saída do BiLSTM é então alimentada para um CRF de cadeia linear, que pode gerar previsões usando este contexto aprimorado.

Este trabalho utiliza a arquitetura da rede BiLSTM-CRF proposta em Oliveira (2020), porém, sem a camada de função de *loss*. A rede é treinada e avaliada com os textos de boletins de ocorrências CVLI coletados e classificados, para que os resultados sejam utilizados como *baselines* para comparação com os resultados do modelo BERT proposto neste trabalho.

5.3.3 Modelo BERT

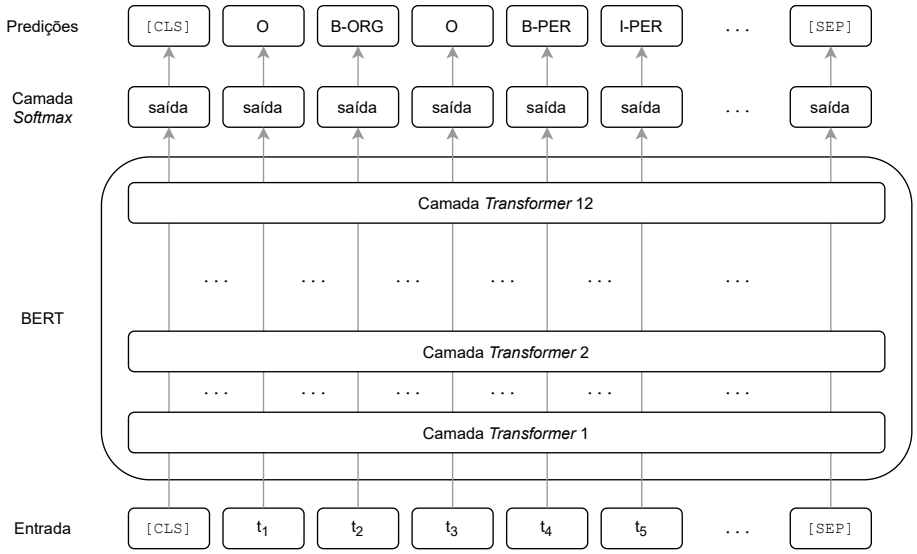
O modelo BERT pré-treinado pode ser ajustado com apenas uma camada de saída adicional para criar modelos de última geração para uma ampla gama de tarefas, incluindo NER. Quando o BERT é ajustado em uma tarefa, o *Transformer* pré-treinado funciona como um codificador, e, um classificador inicializado aleatoriamente é adicionado no topo do modelo. No caso do NER, o classificador é simplesmente uma projeção do tamanho do estado oculto

do *token* para o tamanho do conjunto de rótulos, com uma operação *Softmax* subsequente para transformar as pontuações em probabilidades. O classificador de *token* é compartilhado em todas as posições. Os estados ocultos finais (a saída do *Transformer*) de cada *token* de entrada são alimentados para a camada de classificação para obter uma previsão para cada *token*.

Este trabalho utiliza o modelo BERT *base portuguese cased*[2], que é um modelo BERT pré-treinado para a língua portuguesa, usando a arquitetura *base* (BERT-PT$_{\text{BASE}}$). O BERT para a língua portuguesa foi pré-treinado em um *corpus* de português brasileiro, brWaC (FILHO *et al.*, 2018) (*Brasilian Web as Corpus*), que contém 2.68 bilhões de *tokens* e conta com um vocabulário de 29.794 *tokens*.

A Figura 16 mostra a arquitetura geral do modelo BERT-PT$_{\text{BASE}}$ para a tarefa de NER proposto neste trabalho. O modelo recebe como entrada uma sequência de *tokens* e retorna uma sequência de rótulos para cada *token*.

Figura 16 – Arquitetura geral do modelo BERT utilizado neste trabalho



Fonte: Elaborado pela autora.

Como o tokenizador *WordPiece* divide algumas palavras em subpalavras, a previsão de apenas o primeiro *token* de uma palavra é considerada. Se um *token* original for dividido em várias *WordPieces*, todas as *WordPieces* serão marcadas pelo classificador, mas apenas as previsões principais serão incluídas no cálculo da *loss* e na saída em tempo de execução.

O modelo BERT-PT$_{\text{BASE}}$ foi criado e treinado para reconhecer entidades nomeadas em textos CVLI, lidando com dados distribuídos de forma estratificada. O modelo foi criado utilizando o *spaCy* v3.1, que interopera com a biblioteca *Transformers*[3] da *Hugging Face*. O

[2] Disponível em https://github.com/neuralmind-ai/portuguese-bert
[3] Disponível em https://huggingface.co/transformers/

treinamento nessa nova versão do *spaCy* é totalmente configurável e extensível, e é possível definir modelos personalizados usando *PyTorch*, *TensorFlow* e outras bibliotecas (SPACY.IO, 2021). O modelo BERT-PT$_{\text{BASE}}$, assim como outras configurações e hiperparâmetros para treinar um *pipeline*, podem ser obtidos através do site do *spaCy*[4].

5.4 Vetorização dos dados

As redes neurais possuem entradas de dados distintas. Após a estratificação dos dados, os conjuntos foram vetorizados de acordo com cada entrada dos modelos.

O modelo CNN do *framework spaCy* recebe como entrada uma lista contendo tuplas que possuem o texto e outra lista de tuplas com as informações sobre as classificações deste texto. Essas classificações indicam as posições inicial e final da parte classificada no texto, bem como qual a classe foi atribuída. O tipo de entrada de dados no modelo pode ser visualizado da seguinte forma: *array*(texto, {'entities': [(posição inicial, posição final, nome da entidade)]}).

Para os dados de entrada do modelo BiLSTM foi realizado um processo de tokenização, remoção de caracteres especiais e *stopwords*, antes da representação vetorial e numéricas dos textos, o mesmo processo feito para a entrada da biblioteca *scikit-multilearn*, explicado na Seção 5.2. O esquema de rotulagem utilizado foi o formato IOBES (também chamado de BILOU), onde cada *token* é previsto com uma *tag* indicada por B- (início), I- (dentro), E- (fim), S- (único) de uma entidade nomeada com seu tipo, ou O-, indicando que o *token* está fora de entidades nomeadas Por fim, foi aplicada a codificação *one-hot* nos rótulos.

E, por fim, o modelo BERT recebe como entrada uma sentença e utiliza *tokens* especiais para entender a entrada de forma correta: o *token* especial, `[CLS]`, é um *token* de classificação que é inserido no início da sequência, e o *token* `[SEP]` é inserido sempre no final de uma sequência. E o *token* `[PAD]` é utilizado para preencher posições. Nesse caso, o esquema de rotulagem utilizado foi o formato BIO. O prefixo "B" indica o início de uma entidade, o prefixo "I" indica que o *token* está dentro de uma entidade, e a *tag* "O" indica que o *token* não pertence a nenhuma entidade.

O BERT utiliza a entrada de *embeddings* do *WordPiece* para *tokens*. Cada *token* da entrada é mapeado para um *embedding* correspondente no vocabulário. As palavras ausentes no vocabulário são quebradas em subpalavras e caracteres menores e é adicionado o símbolo ## nas partes não iniciais de cada subpalavra, por exemplo, *brin* e *##car*, como mostrado na Figura 17,

[4] https://spacy.io/usage/training

de forma que essas subpalavras e caracteres menores estejam presentes no vocabulário.

Figura 17 – Representação dos *tokens* de um modelo BERT

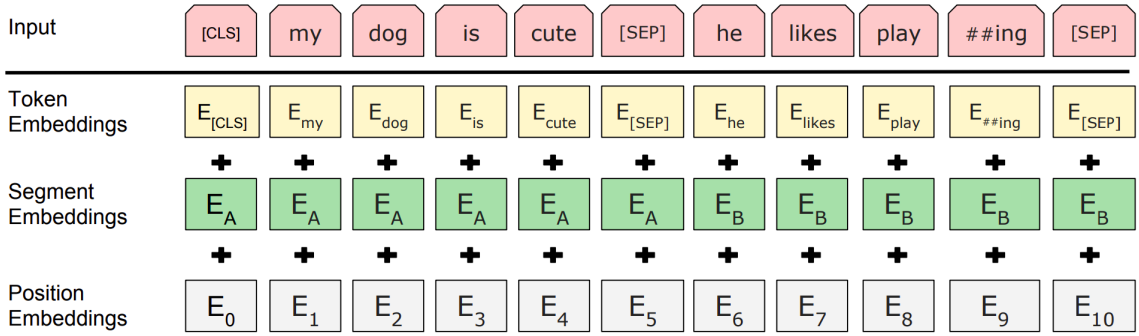
```
['[CLS]', 'Meu', 'cachorro', 'gosta', 'de', 'brin', '##car', '[SEP]']
```

Fonte: Elaborado pela autora.

O BERT espera que cada frase comece com um *token* [CLS] e termine com um *token* [SEP]. Esses *tokens* especiais não são particularmente relevantes para a tarefa NER, considerando que a classificação é feita por meio de *tokens* e os *tokens* especiais não têm rótulo associado. No entanto, eles devem ser incluídos para que a entrada de *fine-tuning* não seja muito diferente da entrada de pré-treinamento.

Juntamente com os *embeddings* de *token*, são utilizados *embeddings* posicionais, que contêm informações sobre a posição dos *tokens* na sequência, e *embeddings* de segmento, que são úteis quando a entrada do modelo tem pares de frases, para cada *token*. Os *embeddings* finais utilizados pela arquitetura do modelo BERT são a soma dos *embeddings* de tokens, *embeddings* de posição, bem como *embeddings* de segmento. A Figura 18 demonstra como é a representação de entrada do BERT.

Figura 18 – Representação da entrada de um modelo BERT



Fonte: Devlin *et al.* (2019)

5.5 Avaliação comparativa de NER com e sem estratificação dos dados

Um possível cenário de experimentação é a execução de um modelo com dados distribuídos de forma não estratificada. Com isso, também foi realizada uma divisão dos dados rotulados de maneira não estratificada, com o objetivo de avaliar dois modelos, o primeiro que utiliza textos separados de forma não estratificada, ou seja, as classes estavam desproporcionais

nos conjuntos de treino, validação e teste. E o segundo modelo que utiliza textos divididos de maneira estratificada, garantindo a mesma proporção de classes nos conjuntos.

A divisão dos dados não estratificados (Tabela 2) ficou da seguinte maneira: conjunto de treino com 70% dos dados (711 textos), conjunto de validação com 10% dos dados (101 textos) e o conjunto de teste com 20% dos dados (202 textos).

Tabela 2 – Quantidade de ocorrências e proporção das classes na distribuição dos dados não estratificados

Classes	**Conjuntos de dados**						
	Dataset	**Treino (%)**		**Validação (%)**		**Teste (%)**	
ARMA_BRANCA	132	80	(60,60%)	15	(11,36%)	37	(28,04%)
ACHAD_CAD	90	48	(53,33%)	9	(10,00%)	33	(36,67%)
ACID_TRANSITO	155	125	(80,64%)	23	(14,84%)	7	(4,52%)
ARMA_DE_FOGO	1035	764	(73,82%)	94	(9,08%)	177	(17,10%)
CONFLITO_PC	7	7	(100,00%)	0	(0,00%)	0	(0,00%)
CONFLITO_PM	13	12	(92,30%)	0	(0,00%)	1	(7,70%)
DUPLO_HOM	21	15	(71,43%)	1	(4,76%)	5	(23,81%)
EXECUCAO	153	121	(79,08%)	12	(7,84%)	20	(13,08%)
FEM	20	14	(70,00%)	0	(0,00%)	6	(30,00%)
FOLGA	5	1	(20,00%)	0	(0,00%)	4	(80,00%)
FONTE	86	53	(61,63%)	0	(0,00%)	33	(38,37%)
HOM_DOLOSO	24	18	(75,00%)	4	(16,67%)	2	(8,33%)
HOM	390	281	(72,05%)	46	(11,80%)	63	(16,15%)
INTER_POL	15	12	(80,00%)	0	(0,00%)	3	(20,00%)
LATROC	42	26	(61,90%)	3	(7,14%)	13	(30,96%)
LEG_DEF_TER	10	6	(60,00%)	0	(0,00%)	4	(40,00%)
LEG_DEFESA	24	13	(54,17%)	1	(4,16%)	10	(41,67%)
LES_CORPORAL	48	30	(62,50%)	2	(4,17%)	16	(33,33%)
LOC	1459	1072	(73,47%)	75	(5,14%)	312	(21,39%)
MAE_DA_VITIMA	123	86	(69,92%)	1	(0,81%)	36	(29,27%)
MORADOR_RUA	13	6	(46,15%)	2	(15,38%)	5	(38,47%)
MORREU_DEPOIS	16	15	(93,75%)	0	(0,00%)	1	(6,25%)
MORTE_HOSP	150	113	(75,33%)	9	(6,00%)	28	(18,67%)
MOTIVACAO	22	12	(54,55%)	0	(0,00%)	10	(45,45%)
ORG	746	556	(74,53%)	76	(10,19%)	114	(15,28%)
PAI_DA_VITIMA	94	64	(68,08%)	1	(1,07%)	29	(30,85%)
PER	1025	766	(74,74%)	64	(6,24%)	195	(19,02%)
POLICIAIS_DA_OCORRENCIA	390	259	(66,41%)	8	(2,05%)	123	(31,54%)
TRIPLO_HOM	10	8	(80,00%)	1	(10,00%)	1	(10,00%)
UNID_PRISIONAL	33	19	(57,57%)	3	(9,10%)	11	(33,33%)
VITIMA	1196	937	(78,34%)	103	(8,61%)	156	(13,05%)

Fonte: Elaborado pela autora.

A Tabela 2 mostra a quantidade de ocorrências e também a proporção em relação a divisão (indicada pelo número com o símbolo %) de cada classe nos conjuntos dos dados não estratificados. É possível perceber que alguns rótulos não estão presentes no conjunto de validação ou teste, evidenciando a desproporcionalidade das classes nos conjuntos quando comparado com a distribuição da Tabela 1.

Os dados foram vetorizados e os modelos foram criados e treinados utilizando uma CNN do *framework spaCy* com 100 épocas e as 31 classes. A Tabela 3 mostra a avaliação geral dos modelos, onde é apresentada a média ponderada das métricas *Precision*, *Recall* e *F1-score*,

além dos valores da Acurácia Balanceada de cada modelo.

Tabela 3 – Comparação geral dos modelos

Dados dos modelos	**Métricas**			
	Precision	***Recall***	***F1-score***	**AB**
Dados estratificados	**64,76**	**62,03**	**63,20**	**34,41**
Dados não estratificados	58,34	55,21	56,11	28,52

Fonte: Elaborado pela autora.

É possível perceber na Tabela 3 que o modelo treinado com os dados estratificados obteve melhores resultados em todas as métricas, com isso, este trabalho faz o uso dos dados estratificados para o treinamento dos modelos seguintes. A Tabela 4 mostra os resultados das métricas para cada classe dos dois modelos.

Tabela 4 – Resultados das métricas para cada classe dos modelos

Classes	**Total de ocorrências**	**Modelos CNN**					
		Dados estratificados			**Dados não estratificados**		
		Precision	***Recall***	***F1-score***	***Precision***	***Recall***	***F1-score***
ARMA_BRANCA	132	94,12	76,19	84,21	86,96	62,50	72,73
ACHAD_CAD	90	36,36	23,53	28,57	53,85	21,21	30,43
ACID_TRANSITO	155	50,00	40,74	44,90	100,00	42,86	60,00
ARMA_DE_FOGO	1035	85,25	85,71	85,48	76,73	76,73	76,73
CONFLITO_PC	7	**0,00**	**0,00**	**0,00**	**0,00**	**0,00**	**0,00**
CONFLITO_PM	13	**0,00**	**0,00**	**0,00**	**0,00**	**0,00**	**0,00**
DUPLO_HOM	21	50,00	33,33	40,00	50,00	25,00	33,33
EXECUCAO	153	**0,00**	**0,00**	**0,00**	08,33	05,56	06,67
FEM	20	**0,00**	**0,00**	**0,00**	**0,00**	**0,00**	**0,00**
FOLGA	5	**0,00**	**0,00**	**0,00**	**0,00**	**0,00**	**0,00**
FONTE	86	93,33	93,33	93,33	96,88	93,94	95,38
HOM	24	85,51	84,29	84,89	83,33	75,00	78,95
HOM_DOLOSO	390	75,00	100,00	85,71	100,00	50,00	66,67
INTER_POL	15	**0,00**	**0,00**	**0,00**	**0,00**	**0,00**	**0,00**
LATROC	42	60,00	37,50	46,15	**0,00**	**0,00**	**0,00**
LEG_DEFESA	24	50,00	20,00	28,57	**0,00**	**0,00**	**0,00**
LEG_DEF_TER	10	**0,00**	**0,00**	**0,00**	**0,00**	**0,00**	**0,00**
LES_CORPORAL	48	**0,00**	**0,00**	**0,00**	**0,00**	**0,00**	**0,00**
LOC	1459	73,66	67,90	70,66	62,91	62,45	62,68
MAE_DA_VITIMA	123	62,50	55,56	58,82	69,23	75,00	72,00
MORADOR_RUA	13	50,00	33,33	40,00	**0,00**	**0,00**	**0,00**
MORREU_DEPOIS	16	**0,00**	**0,00**	**0,00**	**0,00**	**0,00**	**0,00**
MORTE_HOSP	150	**0,00**	**0,00**	**0,00**	18,18	07,41	10,53
MOTIVACAO	22	**0,00**	**0,00**	**0,00**	**0,00**	**0,00**	**0,00**
ORG	746	69,48	75,89	72,54	37,98	47,12	42,06
PAI_DA_VITIMA	94	66,67	60,00	63,16	60,00	72,41	65,62
PER	1025	45,18	42,13	43,60	44,00	52,38	47,83
POLICIAIS_DA_OCORRENCIA	390	90,16	73,33	80,88	84,26	73,98	78,79
TRIPLO_HOM	10	**0,00**	**0,00**	**0,00**	**0,00**	**0,00**	**0,00**
UNID_PRISIONAL	33	**0,00**	**0,00**	**0,00**	**0,00**	**0,00**	**0,00**
VITIMA	1196	61,96	63,92	62,93	58,42	40,69	47,97

Fonte: Elaborado pela autora.

Por ter poucos exemplos de alguns rótulos, houveram muitas classes que as métricas tiveram valor zero em ambos os modelos. No modelo com os dados estratificados, 12 classes ficaram com valor zero, e no modelo com os dados não estratificados, 14 classes tiveram valor zero.

Já que os modelos não foram efetivos nos resultados, então foi escolhido retirar os

rótulos que possuem menos de 50 ocorrências do *dataset*. Um total de 16 rótulos foram excluídos: CONFLITO_PC, CONFLITO_PM, DUPLO_HOM, FEM, FOLGA, HOM_DOLOSO, INTER_POL, LATROC, LEG_DEF_TER, LEG_DEFESA, LES_CORPORAL, MORADOR_RUA, MORREU_DEPOIS, MOTIVACAO, TRIPLO_HOM e UNID_PRISIONAL. E as anotações dessas classes foram removidas do conjunto de dados.

Embora mais da metade das classes tivessem sido retiradas, todas elas estão presentes em apenas 22,98% dos textos, um total de 233 textos do total. Então, ao final do processo de retirada dos rótulos com menos de 50 ocorrências, restaram 15 rótulos utilizados nas classificações do *dataset*.

5.5.1 Lidando com as classes majoritárias

Após a remoção das classificações, foi feita uma nova estratificação no *dataset*, já que os dados continuam desbalanceados. A nova estratificação manteve a mesma proporção dos dados para os subconjuntos, 70% para treino (701 textos), 10% para validação (103 textos) e 20% para teste (210 textos). E a Tabela 5 mostra a quantidade de ocorrências e a proporção da divisão (indicada pelo número com símbolo %) de cada uma das 15 classe nos subconjuntos.

Tabela 5 – Quantidade de ocorrências e proporção das 15 classes na distribuição dos dados estratificados

Classes	**Conjuntos de dados**						
	Dataset	**Treino (%)**		**Validação (%)**		**Teste (%)**	
ARMA_BRANCA	132	89	(67,42%)	18	(13,64%)	25	(18,94%)
ACHAD_CAD	90	62	(68,89%)	11	(12,22%)	17	(18,89%)
ACID_TRANSITO	155	108	(69,68%)	14	(9,03%)	33	(21,29%)
ARMA_DE_FOGO	1035	725	(70,04%)	101	(9,76%)	209	(20,20%)
EXECUCAO	153	104	(67,97%)	15	(9,80%)	34	(22,23%)
FONTE	86	62	(72,10%)	8	(9,30%)	16	(18,60%)
HOM	390	270	(69,23%)	42	(10,77%)	78	(20,00%)
LOC	1459	1056	(72,38%)	135	(9,25%)	268	(18,37%)
MAE_DA_VITIMA	123	81	(65,85%)	12	(9,75%)	30	(24,40%)
MORTE_HOSP	150	101	(67,33%)	14	(9,33%)	35	(23,34%)
ORG	746	498	(66,76%)	74	(9,92%)	174	(23,32%)
PAI_DA_VITIMA	94	67	(71,27%)	9	(9,58%)	18	(19,15%)
PER	1025	699	(68,20%)	83	(8,10%)	243	(23,70%)
POLICIAIS_DA_OCORRENCIA	390	276	(70,77%)	34	(8,72%)	80	(20,51%)
VITIMA	1196	854	(71,40%)	129	(10,78%)	213	(17,82%)

Fonte: Elaborado pela autora.

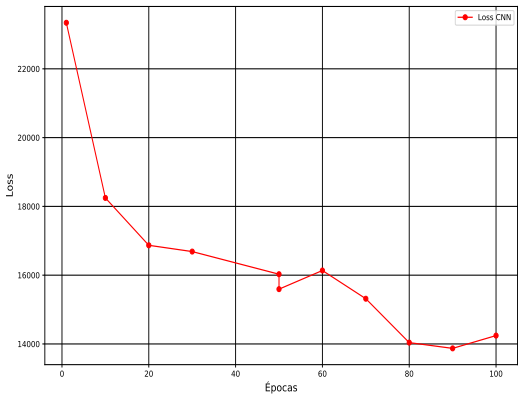
A partir desses dados, com a remoção das classe minoritárias, a nova estratificação e vetorização, os modelos foram treinados e avaliados utilizando as métricas *Precision*, *Recall*, *F1-score* e Acurácia Balanceada (AB).

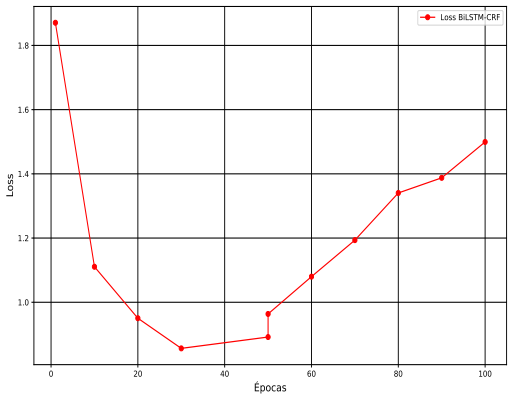
5.6 Resultados

A plataforma *Google Colaboratory*, também chamado de *Colab*, foi utilizada como ambiente de desenvolvimento e implementação das redes neurais deste trabalho. O *Colab* é uma plataforma voltada para a criação e execução de código em *Python*, diretamente do navegador. Foi utilizado um ambiente de 35 GB de memória RAM para a execução do modelo BERT-PT$_{\text{BASE}}$.

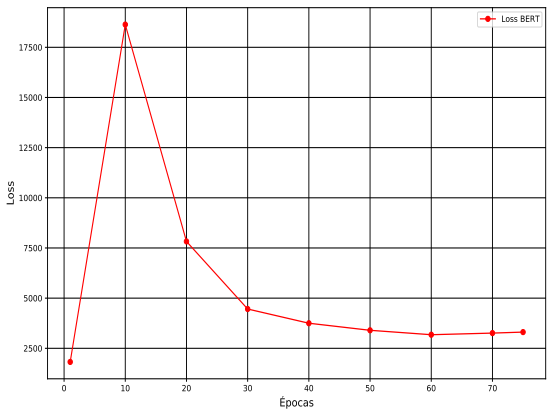
Os modelos do *baseline* foram treinados por 100 épocas cada. Já o modelo BERT-PT$_{\text{BASE}}$ foi treinado por 75 épocas com *batch* tamanho 128. Os valores de épocas de treinamento dos modelos foram escolhidos analisando os valores de *loss* de cada modelo para os dados de validação. A Figura 19 mostra os valores das perdas a cada 10 épocas de treino de cada modelo.

Figura 19 – Valores das *losses* obtidos nos treinamentos dos modelos
(a) CNN (b) BiLSTM-CRF





(c) BERT-PT$_{\text{BASE}}$



Fonte: Elaborado pela autora.

Todos os modelos foram treinados e avaliados com o mesmo conjunto de dados estratificado e as 15 classes. Os resultados obtidos a partir da avaliação do modelo BERT-PT$_{\text{BASE}}$ foram comparados com o modelo CNN e o modelo BiLSTM-CRF, escolhidos como *baselines*, principalmente por serem modelos do estado da arte usados para tarefas NER.

A Tabela 6 exibe os resultados dos modelos BERT-PT$_{\text{BASE}}$, CNN e BiLSTM-CRF especificados pelas classes, utilizando as métricas *Precision*, identificado por **P** na tabela, *Recall*, identificado por **R** e o *F1-score*, identificado por **F1**.

Os maiores valores para a métrica *F1-score* de acordo com cada classe foi marcado em negrito e os valores em 0,00 indicam que o modelo não conseguiu reconhecer a entidade nomeada. O modelo BERT-PT$_{\text{BASE}}$ obteve os melhores resultados da métrica *F1-score* para a maioria das classes, com diferenças significativas quando comparado com os resultados dos *baselines*.

Tabela 6 – Resultados do modelo BERT-PT$_{\text{BASE}}$ com os *baselines* para as 15 classes.

Classes	***Baselines***						**Modelo BERT-PT$_{\text{BASE}}$**		
	CNN			**BiLSTM-CRF**					
	P	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**
ARMA_BRANCA	70,37	76,00	73,07	70,00	60,87	65,12	90,00	78,26	**83,72**
ACHAD_CAD	15,00	17,65	16,22	0,00	0,00	0,00	37,50	18,75	**25,00**
ACID_TRANSITO	40,00	30,30	34,48	12,50	30,30	17,70	51,72	45,45	**48,39**
ARMA_DE_FOGO	75,11	79,42	**77,20**	65,81	82,35	73,16	65,55	83,42	73,41
EXECUCAO	7,14	5,88	6,45	09,84	17,65	12,63	36,36	35,29	**35,82**
FONTE	87,50	87,50	87,50	53,33	53,33	53,33	92,86	86,67	**89,66**
HOM	78,82	85,89	82,20	62,77	80,82	70,66	87,50	86,30	**86,90**
LOC	57,03	58,95	57,98	28,21	46,81	35,20	69,50	76,60	**72,87**
MAE_DA_VITIMA	64,52	66,67	65,57	41,67	53,57	46,88	75,86	78,57	**77,19**
MORTE_HOSP	4,34	2,85	3,44	04,96	17,65	07,74	25,64	29,41	**27,40**
ORG	62,63	65,51	64,04	45,40	49,07	47,16	70,59	74,53	**72,51**
PAI_DA_VITIMA	61,11	61,11	61,11	18,92	41,18	25,93	78,95	88,24	**83,33**
PER	56,89	40,74	47,48	21,72	30,64	25,42	70,99	66,47	**68,66**
POLICIAIS_DA_OCORRENCIA	83,11	80,00	**81,52**	39,80	51,32	44,83	81,25	68,42	74,29
VITIMA	63,10	61,03	62,05	28,19	29,12	28,65	75,86	84,62	**80,00**

Fonte: Elaborado pela autora.
Nota: As letras **P**, **R** e **F1** referem-se às métricas de avaliação *Precision*, *Recall* e *F1-score*, respectivamente.

O modelo BERT-PT$_{\text{BASE}}$ superou os *baselines*, reconhecendo boa parte das classes de forma mais eficaz que o modelo CNN, e todas as classes com mais eficiência que o modelo BiLSTM-CRF. E os resultados gerais mostrados na Tabela 7 comprovam que o modelo BERT-PT$_{\text{BASE}}$ teve melhor desempenho que os *baselines*.

A Tabela 7 mostra a avaliação geral dos três modelos, onde é apresentada a média ponderada das métricas *Precision*, *Recall* e *F1-score*, além dos valores da Acurácia Balanceada de cada modelo. Os melhores valores obtidos para cada métrica estão destacados em negrito.

O modelo BERT-PT$_{\text{BASE}}$ obteve um aumento significativo em relação aos *baselines*. Na métrica *F1-score*, teve um resultado de 6,98% a mais que o modelo CNN e de 31,84% a mais que o modelo BiLSTM-CRF. Na métrica AB, teve um resultado de 12,09% a mais que o modelo CNN e de 23,75% a mais que o modelo BiLSTM-CRF.

Tabela 7 – Comparação geral do modelo BERT-PT$_{\text{BASE}}$ com os *baselines*

Modelos	**Métricas**			
	Precision	***Recall***	***F1-score***	**AB**
CNN	65,62	63,62	64,46	54,64
BiLSTM-CRF	33,89	47,63	39,60	42,98
BERT-PT$_{\text{BASE}}$	**69,89**	**73,66**	**71,44**	**66,73**

Fonte: Elaborado pela autora.

Considerando os resultados obtidos por classe e os resultados gerais, o modelo BERT-PT$_{\text{BASE}}$ proposto neste trabalho obteve os melhores resultados, superando os valores das métricas para quase todas as classes e no geral, em relação aos *baselines*.

6 CONCLUSÕES E TRABALHOS FUTUROS

O uso de *Deep Learning* vem sendo empregado em sistemas NER, gerando um desempenho de última geração. Neste trabalho foi proposto um modelo NER para textos de boletins de ocorrência de CVLI, divididos de forma estratificada, e utilizando o BERT-PT$_{\text{BASE}}$, um modelo *Transformer* pré-treinado na língua portuguesa. O modelo proposto conseguiu reconhecer as 15 entidades nomeadas do domínio, de forma que o processo de extração de informações importantes de textos CVLI se torne uma tarefa mais fácil, utilizando uma arquitetura *Transformer*.

O BERT é uma boa base para obter resultados de ponta sobre NER, e o *fine-tuning* para essa tarefa é relativamente simples de implementar. Este trabalho contribui com a utilização de um modelo de representação de linguagem para a tarefa NER no domínio criminal.

Nos experimentos, foram feitas etapas de pré-processamento dos dados para a criação dos conjuntos utilizados para treinar e avaliar o modelo. Além da divisão dos dados de forma estratificadas entre esses conjuntos e também as etapas de treinamento e avaliação dos modelos implementados.

Na etapa de avaliação, o modelo BERT-PT$_{\text{BASE}}$ proposto foi comparado com os modelos CNN e BiLSTM-CRF, considerados *baselines* para este trabalho, e superou os valores das métricas *Precision*, *Recall* e *F1-score* para quase todas as classes. Na avaliação geral, o modelo proposto teve melhorias significativas para as métricas acima, com 69,89% de *Precision*, 73,66% de *Recall* e 71,44% *F1-score* e também para a Acurácia Balanceada atingindo o valor de 66,73%. Esta última métrica teve ganhos entre 12,09% e 23,75%, em relação aos *baselines*.

Como trabalhos futuros pretende-se classificar mais textos utilizando todos os rótulos e fazer o processo de *fine-tuning* para NER do modelo BERT pré-treinado em português com os 31 rótulos iniciais. Também é possível utilizar o BERT como *word embeddings* em outros modelos, extraindo as representações de palavras geradas na etapa de pré-treinamento. Essa abordagem é chamada *feature-based*, uma técnica de *Transfer Learning*. Devlin *et al.* (2019) mostram os resultados do BERT nessa abordagem para NER, em uma rede BiLSTM, e as representações geradas pelo BERT são acuradas mesmo sem o processo de *fine-tuning*.

REFERÊNCIAS

AIRES, A. C.; ESCOVAR, J. V.; CARDOSO, M. L. **Em um mundo conectado, dados armazenados tornam-se protagonistas**. 2017. Disponível em: https://paineira.usp.br/aun/index.php/2017/08/21/em-um-mundo-conectado-dados-armazenados-tornam-se-protagonistas/. Acesso em: 11 ago. 2020.

ALAMMAR, J. **The Illustrated Transformer**. 2018. Disponível em: https://jalammar.github.io/illustrated-transformer/. Acesso em: 02 ago. 2021.

ARAÚJO, N. d. S. **Reconhecimento de entidades nomeadas em textos de boletins de ocorrências**. 2019. 41 f. Trabalho de Conclusão de Curso (Graduação em Ciência da Computação) - Universidade Federal do Ceará, Campus de Quixadá, Quixadá, 2019.

CEARÁ. Secretaria de Segurança Pública - SSP. **Crimes violentos letais e intencionais – CVLI**. Fortaleza, CE: [*S. n.*], 2020. Disponível em: https://www.sspds.ce.gov.br/wp-content/uploads/sites/24/2020/04/01-CVLI-Estat%C3%ADsticas-Mensais.pdf. Acesso em: 30 abr. 2020.

CHINCHOR, N.; HIRSCHMAN, L.; LEWIS, D. D. Evaluating message understanding systems: An analysis of the third message understanding conference (MUC-3). **Comput. Linguistics**, v. 19, n. 3, p. 409–449, 1993.

CHOLLET, F. **Deep Learning with Python**. [*S. n.*]. ISBN 9781617294433. Disponível em: https://books.google.com.br/books?id=Yo3CAQAACAAJ. Acesso em: 12 mai. 2020.

COWIE, J.; LEHNERT, W. Information extraction. **Communications of the ACM**, ACM New York, NY, USA, v. 39, n. 1, p. 80–91, 1996.

DEVELOPERS, G. **Embeddings**: translating to a lower-dimensional space. 2020. Machine Learning Crash Course with TensorFlow APIs. Disponível em: https://developers.google.com/machine-learning/crash-course/embeddings/translating-to-a-lower-dimensional-space. Acesso em: 02 jul. 2021.

DEVLIN, J.; CHANG, M.; LEE, K.; TOUTANOVA, K. BERT: pre-training of deep bidirectional transformers for language understanding. In: **Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT**. Minneapolis, MN, USA: Association for Computational Linguistics, 2019. p. 4171–4186. Disponível em: https://doi.org/10.18653/v1/n19-1423. Acesso em: 3 jul. 2020.

FILHO, J. A. W.; WILKENS, R.; IDIART, M.; VILLAVICENCIO, A. The brWaC corpus: A new open resource for Brazilian Portuguese. In: INTERNATIONAL CONFERENCE ON LANGUAGE RESOURCES AND EVALUATION, 11., Miyazaki, Japan. **Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)**. Miyazaki, Japan, 2018.

FUKUDA, K.-i.; TSUNODA, T.; TAMURA, A.; TAKAGI, T. *et al.* Toward information extraction: identifying protein names from biological papers. In: CITESEER . **Pac symp biocomput**. [*S. l.*], 1998. v. 707, n. 18, p. 707–718.

GANTI, V.; KÖNIG, A. C.; VERNICA, R. Entity categorization over large document collections. In: INTERNATIONAL CONFERENCE ON KNOWLEDGE DISCOVERY AND DATA MINING, 14., Las Vegas, Nevada, USA. **Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining**. New York, NY, United States, 2008. p. 274–282.

GOUTTE, C.; GAUSSIER, E. A probabilistic interpretation of precision, recall and f-score, with implication for evaluation. In: SPRINGER. **Advances in Information Retrieval**. Berlin, Heidelberg, 2005. p. 345–359.

GREFF, K.; SRIVASTAVA, R. K.; KOUTNÍK, J.; STEUNEBRINK, B. R.; SCHMIDHUBER, J. Lstm: A search space odyssey. **IEEE transactions on neural networks and learning systems**, IEEE, v. 28, n. 10, p. 2222–2232, 2016.

HUANG, Z.; XU, W.; YU, K. Bidirectional LSTM-CRF models for sequence tagging. **arXiv preprint arXiv:1508.01991**, abs/1508.01991, 2015. Disponível em: http://arxiv.org/abs/1508.01991. Acesso em: 6 jun. 2020.

JACKSON, P.; MOULINIER, I. **Natural language processing for online applications**: text retrieval, extraction and categorization. [*S. l.*]: John Benjamins Publishing, 2007. v. 5.

JAIN, A. K.; MAO, J.; MOHIUDDIN, K. M. Artificial neural networks: A tutorial. **Computer**, IEEE, v. 29, n. 3, p. 31–44, 1996.

JOHNSON, J. M.; KHOSHGOFTAAR, T. M. Survey on deep learning with class imbalance. **Journal of Big Data**, Springer, v. 6, n. 1, p. 27, 2019.

LAFFERTY, J. D.; MCCALLUM, A.; PEREIRA, F. C. N. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In: INTERNATIONAL CONFERENCE ON MACHINE LEARNING, 18., San Francisco. **Proceedings of the Eighteenth International Conference on Machine Learning**. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2001. (ICML '01), p. 282–289. ISBN 1558607781.

LI, J.; SUN, A.; HAN, J.; LI, C. A survey on deep learning for named entity recognition. **IEEE Transactions on Knowledge and Data Engineering**, IEEE, 2020.

LI, X.; SUN, X.; MENG, Y.; LIANG, J.; WU, F.; LI, J. Dice loss for data-imbalanced nlp tasks. **arXiv preprint arXiv:1911.02855**, Cornell University, Ithaca, Nova Iorque, EUA, 2019.

LIANG, C.; YU, Y.; JIANG, H.; ER, S.; WANG, R.; ZHAO, T.; ZHANG, C. BOND: bert-assisted open-domain named entity recognition with distant supervision. **CoRR**, abs/2006.15509, 2020. Disponível em: https://arxiv.org/abs/2006.15509. Acesso em: 22 jul. 2020.

LIDDY, E. D. Natural language processing. **Encyclopedia of Library and Information Science**, Marcel Decker, Inc., New York, 2001.

LIU, X.; ZHOU, Y.; WANG, Z. Recognition and extraction of named entities in online medical diagnosis data based on a deep neural network. **Journal of Visual Communication and Image Representation**, Elsevier, v. 60, p. 1–15, 2019.

LOPEZ, M. M.; KALITA, J. Deep learning applied to NLP. **CoRR**, abs/1703.03091, 2017. Disponível em: http://arxiv.org/abs/1703.03091. Acesso em: 9 ago. 2020.

MA, X.; XIA, F. Unsupervised dependency parsing with transferring distribution via parallel guidance and entropy regularization. In: MEETING OF THE ASSOCIATION FOR COMPUTATIONAL LINGUISTICS, 52., Baltimore, Maryland. **Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)**. Baltimore, Maryland, 2014. p. 1337–1348.

MAYNARD, D.; BONTCHEVA, K.; AUGENSTEIN, I. Natural language processing for the semantic web. **Synthesis Lectures on the Semantic Web: Theory and Technology**, Morgan & Claypool Publishers, v. 6, n. 2, p. 1–194, 2016.

MOSLEY, L. A balanced approach to the multi-class imbalance problem. **Iowa State University Capstones, Theses and Dissertations**, Ames, Iowa, EUA, 2013.

NADEAU, D.; SEKINE, S. A survey of named entity recognition and classification. **Lingvisticae Investigationes**, John Benjamins, v. 30, n. 1, p. 3–26, 2007.

OLIVEIRA, B. S. N. **Aprendizado profundo para reconhecimento de entidades nomeadas em narrativas de roubos**. 2020. 99 f. Trabalho de Conclusão de Curso (Graduação em Ciência da Computação) - Universidade Federal do Ceará, Campus de Quixadá, Quixadá, 2020.

PILEHVAR, M. T.; CAMACHO-COLLADOS, J. **Embeddings in Natural Language Processing**. [*S. l.*]: Morgan & Claypool Publishers, 2020.

QIN, P.; XU, W.; WANG, W. Y. Robust distant supervision relation extraction via deep reinforcement learning. **CoRR**, abs/1805.09927, 2018. Disponível em: http://arxiv.org/abs/1805.09927. Acesso em: 27 ago. 2021.

RASCHKA, S.; MIRJALILI, V. **Python machine learning**. [*S. l.*]: Packt Publishing Ltd, 2017.

SAMMUT, C.; WEBB, G. I. **Encyclopedia of machine learning**. [*S. l.*]: Springer Science & Business Media, 2011.

SECHIDIS, K.; TSOUMAKAS, G.; VLAHAVAS, I. On the stratification of multi-label data. In: SPRINGER. **Joint European Conference on Machine Learning and Knowledge Discovery in Databases**. Berlin, Heidelberg: Springer-Verlag, 2011. (ECML PKDD'11), p. 145–158. ISBN 9783642238079.

SHARFUDDIN, A. A.; TIHAMI, M. N.; ISLAM, M. S. A deep recurrent neural network with bilstm model for sentiment classification. In: INTERNATIONAL CONFERENCE ON BANGLA SPEECH AND LANGUAGE PROCESSING, Sylhet, Bangladesh. **2018 International Conference on Bangla Speech and Language Processing (ICBSLP)**. Sylhet, Bangladesh, 2018. p. 1–4.

SHEN, Y.; YUN, H.; LIPTON, Z. C.; KRONROD, Y.; ANANDKUMAR, A. Deep active learning for named entity recognition. **CoRR**, abs/1707.05928, 2017. Disponível em: http://arxiv.org/abs/1707.05928. Acesso em: 16 nov. 2020.

SILVA, T. L. C. da; ARAÚJO, N. da S.; MACÊDO, J. A. F. de; ARAÚJO, D.; SOARES, F. M.; REGO, P. A. L.; NETO, A. V. L. Novel approach for label disambiguation via deep learning. In: INTERNATIONAL CONFERENCE ON MACHINE LEARNING AND DATA MINING IN PATTERN RECOGNITION (MLDM), 15., 2019, New York. **Machine Learning and Data Mining in Pattern Recognition**. New York, NY, USA: ibai publishing, 2019.

SILVA, T. L. C. da; MAGALHÃES, R. P.; MACÊDO, J. A. de; ARAÚJO, D.; ARAÚJO, N.; MELO, V. de; OLÍMPIO, P.; REGO, P. A.; NETO, A. V. L. Improving named entity recognition using deep learning with human in the loop. In: INTERNATIONAL CONFERENCE ON EXTENDING DATABASE TECHNOLOGY, 22., Lisboa, Portugal. **EDBT/ICDT 2019 Joint Conference**. Lisboa, Portugal, 2019. p. 594–597.

SOUZA, F.; NOGUEIRA, R. F.; LOTUFO, R. de A. Portuguese named entity recognition using BERT-CRF. **CoRR**, abs/1909.10649, 2019. Disponível em: http://arxiv.org/abs/1909.10649. Acesso em: 6 jul. 2020.

SPACY.IO. **Models Languages | spaCy Usage Documentation**. 2020. Disponível em: https://spacy.io/usage/models#language. Acesso em: 20 ago. 2020.

SPACY.IO. **Training Pipelines & Models | spaCy Usage Documentation**. 2021. Disponível em: https://spacy.io/usage/training. Acesso em: 04 ago. 2021.

SPECK, R.; NGOMO, A.-C. N. Ensemble learning for named entity recognition. In: INTERNATIONAL SEMANTIC WEB CONFERENCE, 13., Berlim. **International Semantic Web - ISWC 2014**. Berlin, Heidelberg: Springer-Verlag, 2014. p. 519–534.

SZYMANSKI, P.; KAJDANOWICZ, T. A scikit-based python environment for performing multi-label classification. **CoRR**, abs/1702.01460, 2017. Disponível em: http://arxiv.org/abs/1702.01460. Acesso em: 13 jul. 2020.

TALON. **Deep Learning Book**. 2020. Disponível em: http://www.deeplearningbook.com.br/. Acesso em: 2 ago. 2020.

VASWANI, A.; SHAZEER, N.; PARMAR, N.; USZKOREIT, J.; JONES, L.; GOMEZ, A. N.; KAISER, L.; POLOSUKHIN, I. Attention is all you need. **CoRR**, abs/1706.03762, 2017. Disponível em: http://arxiv.org/abs/1706.03762. Acesso em: 27 jun. 2021.

YU, F.; ZHANG, Y.; SONG, S.; SEFF, A.; XIAO, J. LSUN: construction of a large-scale image dataset using deep learning with humans in the loop. **CoRR**, abs/1506.03365, 2015. Disponível em: http://arxiv.org/abs/1506.03365. Acesso em: 10 jun. 2020.

ZHANG, Y.; WALLACE, B. C. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. **CoRR**, abs/1510.03820, 2015. Disponível em: http://arxiv.org/abs/1510.03820. Acesso em: 11 mai. 2021.

ZHENG, S.; JAYASUMANA, S.; ROMERA-PAREDES, B.; VINEET, V.; SU, Z.; DU, D.; HUANG, C.; TORR, P. H. S. Conditional random fields as recurrent neural networks. **CoRR**, abs/1502.03240, 2015. Disponível em: http://arxiv.org/abs/1502.03240. Acesso em: 20 mai. 2020.