



UNIVERSIDADE FEDERAL DE CATALÃO – UFCAT
UNIDADE ACADÊMICA INSTITUTO DE MATEMÁTICA E TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM MODELAGEM E OTIMIZAÇÃO



Carlos Silvio Gomes Junior

**AVALIAÇÃO DA INFLUÊNCIA DO USO DE EMBEDDINGS NO
RECONHECIMENTO DE ENTIDADES NOMEADAS PARA O
PORTUGUÊS**

DISSERTAÇÃO DE MESTRADO

CATALÃO – GO, 2023



UNIVERSIDADE FEDERAL DE GOIÁS
UNIDADE ACADÊMICA ESPECIAL DE MATEMÁTICA E TECNOLOGIA

TERMO DE CIÊNCIA E DE AUTORIZAÇÃO (TECA) PARA DISPONIBILIZAR VERSÕES ELETRÔNICAS DE TESES E DISSERTAÇÕES NA BIBLIOTECA DIGITAL DA UFG

Na qualidade de titular dos direitos de autor, autorizo a Universidade Federal de Goiás (UFG) a disponibilizar, gratuitamente, por meio da Biblioteca Digital de Teses e Dissertações (BDTD/UFG), regulamentada pela Resolução CEPEC nº 832/2007, sem ressarcimento dos direitos autorais, de acordo com a [Lei 9.610/98](#), o documento conforme permissões assinaladas abaixo, para fins de leitura, impressão e/ou download, a título de divulgação da produção científica brasileira, a partir desta data.

O conteúdo das Teses e Dissertações disponibilizado na BDTD/UFG é de responsabilidade exclusiva do autor. Ao encaminhar o produto final, o autor(a) e o(a) orientador(a) firmam o compromisso de que o trabalho não contém nenhuma violação de quaisquer direitos autorais ou outro direito de terceiros.

1. Identificação do material bibliográfico

☒ Dissertação ☐ Tese ☐ Outro*: _____

*No caso de mestrado/doutorado profissional, indique o formato do Trabalho de Conclusão de Curso, permitido no documento de área, correspondente ao programa de pós-graduação, orientado pela legislação vigente da CAPES.

Exemplos: Estudo de caso ou Revisão sistemática ou outros formatos.

2. Nome completo do autor

Carlos Silvio Gomes Junior

3. Título do trabalho

AVALIAÇÃO DA INFLUÊNCIA DO USO DE EMBEDDINGS NO RECONHECIMENTO DE ENTIDADES NOMEADAS PARA O PORTUGUÊS

4. Informações de acesso ao documento (este campo deve ser preenchido pelo orientador)

Concorda com a liberação total do documento ☒ SIM ☐ NÃO¹

[1] Neste caso o documento será embargado por até um ano a partir da data de defesa. Após esse período, a possível disponibilização ocorrerá apenas mediante:

- a) consulta ao(a) autor(a) e ao(a) orientador(a);
 - b) novo Termo de Ciência e de Autorização (TECA) assinado e inserido no arquivo da tese ou dissertação.
- O documento não será disponibilizado durante o período de embargo.

Casos de embargo:

- Solicitação de registro de patente;
- Submissão de artigo em revista científica;
- Publicação como capítulo de livro;
- Publicação da dissertação/tese em livro.

Obs. Este termo deverá ser assinado no SEI pelo orientador e pelo autor.



Documento assinado eletronicamente por **Nadia Felix Felipe Da Silva, Orientadora**, em 16/10/2023, às 14:39, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do [Decreto nº 10.543, de 13 de novembro de 2020](#).



Documento assinado eletronicamente por **CARLOS SILVIO GOMES JUNIOR, Discente**, em 16/10/2023, às 15:04, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do [Decreto nº 10.543, de 13 de novembro de 2020](#).



A autenticidade deste documento pode ser conferida no site https://sei.ufg.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **4123760** e o código CRC **BA118C23**.

CARLOS SILVIO GOMES JUNIOR

AVALIAÇÃO DA INFLUÊNCIA DO USO DE EMBEDDINGS NO
RECONHECIMENTO DE ENTIDADES NOMEADAS PARA O
PORTUGUÊS

Dissertação apresentada ao Programa de Pós-Graduação em Modelagem e Otimização, da Unidade Instituto de Matemática e Tecnologia, da Universidade Federal de Catalão – UFGat, como requisito para obtenção do título de Mestre em Modelagem e Otimização. Área de concentração: Modelagem e Otimização. Linha de pesquisa: Modelagem e Otimização.

Orientadora:
Nádia Félix Felipe da Silva

CATALÃO – GO

2023

Ficha de identificação da obra elaborada pelo autor, através do
Programa de Geração Automática do Sistema de Bibliotecas da UFCAT.

Gomes Junior, Carlos Silvio

Avaliação da influência do uso de Embeddings no Reconhecimento
de Entidades Nomeadas para o Português / Carlos Silvio Gomes
Junior. - 2023.
53, f.

Orientadora: Profa. Dra. Nadia Felix Felipe da Silva.

Dissertação (Mestrado) - Universidade Federal de Catalão, Instituto
de Matemática e Tecnologia, Catalão, Programa de Pós-Graduação
em Modelagem e Otimização, Catalão, 2023.

Inclui lista de figuras, lista de tabelas.

1. REN. 2. Aprendizado Profundo. 3. BiLSTM-CRF. 4. Vetores de
Palavras. I. Silva, Nadia Felix Felipe da, orient. II. Título.

CDU 5



UNIVERSIDADE FEDERAL DE GOIÁS

UNIDADE ACADÊMICA ESPECIAL DE MATEMÁTICA E TECNOLOGIA

ATA DE DEFESA DE DISSERTAÇÃO

Ata nº 21 da sessão de Defesa de Dissertação de **Carlos Silvio Gomes Junior**, que confere o título de **Mestre(a) em Modelagem e Otimização**, na área de concentração em **Modelagem e Otimização**.

Aos **trinta e um dias de agosto de 2023**, a partir das **09h**, via Web Conferência (Google Meet - https://teams.microsoft.com/l/meetup-join/19%3akVr_44lbtXFSIEFMGHmRUXqqwgtZ), reuniram-se os componentes da banca examinadora, professores **Dra. Nádia Félix Felipe da Silva (PPGMO / INF /UF)**, (**orientadora**), **Dr. José dos Reis Vieira de Moura Júnior (PPGMO / IMTec / UFCAT)**, membro titular interno e **Dr. Douglas Cordeiro (FIC-UFG)**, membro titular externo, para, em sessão pública, procederem a avaliação da Dissertação intitulada “**AVALIAÇÃO DA INFLUÊNCIA DO USO DE EMBEDDINGS NO RECONHECIMENTO DE ENTIDADES NOMEADAS PARA O PORTUGUÊS**”, de autoria de **Carlos Silvio Gomes Junior**, discente do Programa de Pós-graduação em Modelagem e Otimização – PPGMO, da Universidade Federal de Catalão - UFCAT. A sessão foi aberta pelo presidente, que fez a apresentação formal dos membros da banca. Em seguida, a palavra foi concedida ao discente que, em 35 min procedeu a apresentação. Terminada a apresentação, cada membro da banca arguiu o(a) examinando(a). Terminada a fase de arguição, procedeu-se a avaliação da Dissertação, que foi considerado(a): (**X**) **Aprovado(a)** ou () **Reprovado(a)**. Cumpridas as formalidades de pauta, a presidência da mesa encerrou a sessão e, para constar, lavrou-se a presente ata que, depois de lida e aprovada, segue assinada pelos membros da banca examinadora.

TÍTULO SUGERIDO PELA BANCA



Documento assinado eletronicamente por **Nadia Felix Felipe Da Silva, Orientadora**, em 11/10/2023, às 11:23, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do [Decreto nº 10.543, de 13 de novembro de 2020](#).



Documento assinado eletronicamente por **Douglas Farias Cordeiro, Professor do Magistério Superior**, em 11/10/2023, às 11:44, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do [Decreto nº 10.543, de 13 de novembro de 2020](#).



Documento assinado eletronicamente por **Jose Dos Reis Vieira De Moura Junior, Professor do Magistério Superior**, em 11/10/2023, às 18:04, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do [Decreto nº 10.543, de 13 de novembro de 2020](#).



A autenticidade deste documento pode ser conferida no site https://sei.ufg.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **4115597** e o código CRC **E54612EC**.

Referência: Processo nº 23070.043717/2023-12

SEI nº 4115597

Agradecimentos

O presente trabalho está sendo realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES). Gostaria de agradecer à CAPES pelo apoio financeiro concedido para a realização desta pesquisa. O suporte da CAPES foi essencial para o sucesso deste estudo e para o meu desenvolvimento acadêmico. Sou grato pela oportunidade de aprimorar meus estudos e conhecimentos, e por contribuir para a produção de novos conhecimentos na área. Obrigado, CAPES. Também expresso meus agradecimentos ao grupo de pesquisa Ulysses, cuja colaboração se estabeleceu em parceria com a Câmara dos Deputados, bem como as instituições ICMC e UFG, pelo apoio inestimável e pela colaboração conjunta.

RESUMO

O Reconhecimento de Entidades Nomeadas (REN) é uma tarefa crucial no contexto atual, uma vez que há uma crescente demanda por ferramentas automatizadas capazes de extrair informações de textos. Embora seja uma área amplamente estudada, a REN ainda é complexa e exige um alto custo de tempo e recursos computacionais, especialmente em línguas com poucos recursos, como o português, que carece de dados anotados e modelos treinados com alto desempenho. Com o objetivo de contribuir para o desenvolvimento dessa tarefa na língua portuguesa, este trabalho aplicou a arquitetura de Aprendizado Profundo (AP), *BiLSTM-CRF*, a partir do *Framework FlairNLP*, testando diferentes tipos de Vetores de Palavras, Vetores Estáticos e Vetores de Contexto, bem como suas concatenações, para analisar a influência de cada um nos resultados obtidos. Os modelos treinados mostraram um ganho expressivo no uso de Vetores de Contexto e concatenação de vetores, atingindo um novo estado da arte com uma média F1 de 98,88% para o domínio geral no Corpus multilíngue *MultiWikiNER*, no domínio legal 89,31% para o Corpus LeNER-Br e 86,86% para o Pl-Corpus, que faz parte do UlyssesNER-Br

Palavras-chaves: REN, Aprendizado Profundo, *BiLSTM-CRF*, Vetores de Palavras.

ABSTRACT

Named Entity Recognition (NER) is a crucial task in today's context, as there is an increasing demand for automated tools capable of extracting information from texts. Although NER is a widely studied area, it remains complex and requires a high cost in terms of time and computational resources, especially in languages with few resources, such as Portuguese, which lacks annotated data and high-performance trained models. In order to contribute to the development of this task in Portuguese language, this work applied the Deep Learning (DL) architecture, BiLSTM-CRF, using the FlairNLP Framework, testing different types of Word Embeddings, Static Embeddings, Contextualize Embeddings, and their concatenations, to analyze their influence on the results obtained. The trained models showed a significant gain in the use of Contextualize Embeddings and Embeddings concatenation, achieving a new state of the art with an F1 score of 98.88% for the general domain in the MultiWikiNER multilingual corpus, in the legal domain 89.31% for the LeNER-Br Corpus and 86.86% for the PI-Corpus, which is part of the UlyssesNER-Br.

Keywords: NER, Deep Learning, BiLSTM-CRF, Word Embeddings.

LISTA DE FIGURAS

Figura 4.1 – <i>Pipeline</i>	29
Figura 4.2 – Planos <i>Google Colaboratory</i>	31
Figura 4.3 – <i>BiLSTM-CRF</i>	33
Figura 4.4 – Entrada da rede com os diferentes Vetores de Palavras.	34
Figura 4.5 – Log de treinamento de um modelo com Vetores Estáticos e de Contexto concatenados.	35
Figura 4.6 – Log de treinamento, identificação dos parâmetros da rede.	36

LISTA DE TABELAS

Tabela 3.1 – Apresentação dos resultados apresentados nos trabalhos relacionados. . .	26
Tabela 4.1 – Arquiteturas de rede	28
Tabela 4.2 – Experimentos realizados	30
Tabela 4.3 – Número de Sentenças no Treinamento, Teste e Validação no Corpus <i>MultiWikiNER</i>	38
Tabela 4.4 – Número de <i>tokens</i> por entidade nos experimentos realizados	38
Tabela 4.5 – Número de Sentenças no Treinamento, Teste e Validação no Corpus LeNER-Br	39
Tabela 4.6 – Número de palavras em entidade por conjunto no Corpus LeNER-Br . . .	40
Tabela 4.7 – Número de Sentenças no Treinamento, Teste e Validação no Corpus PL-Corpus	41
Tabela 5.1 – Resultados com o Corpus <i>MultiWikiNER</i>	44
Tabela 5.2 – Detalhamento do melhor resultado com o Corpus <i>MultiWikiNER</i>	45
Tabela 5.3 – Resultados com o Corpus LeNER-Br.	45
Tabela 5.4 – Detalhamento do melhor resultado obtido com o Corpus LeNER-Br, em comparação com o trabalho que propôs o Corpus	46
Tabela 5.5 – Resultados obtidos com o PL-Corpus.	46
Tabela 5.6 – Detalhamento do melhor resultado obtido com o PL-Corpus, em comparação com o trabalho que propôs o Corpus.	46

SUMÁRIO

1	INTRODUÇÃO	9
1.1	Processamento de Linguagem Natural (PLN) e Reconhecimento De Entidades Nomeadas (REN)	9
1.2	Justificativa	11
1.3	Objetivos	12
1.3.1	Objetivo Geral:	12
1.3.2	Objetivos Específicos:	12
1.4	Hipótese de Pesquisa	12
1.5	Contribuições	12
1.6	Organização da Dissertação	13
2	FUNDAMENTAÇÃO TEÓRICA	14
2.1	Representação Textual	14
2.2	Aprendizado Profundo (AP)	15
2.2.1	Redes Neurais Artificiais (RNA)	15
2.3	Corpus	16
2.4	Vetores de Palavras e Modelos de Linguagem	17
2.4.1	Concatenação de Vetores de Palavras	19
3	TRABALHOS RELACIONADOS	21
3.1	Vetores de Palavras para REN em Português	23
3.2	Discussão Sobre os Trabalhos Relacionados	26
4	METODOLOGIA E MODELAGEM DOS EXPERIMENTOS	28
4.1	Metodologia	28
4.2	Modelagem dos Experimentos	29
4.3	Arquitetura de Rede	32
4.4	Corpus	37
4.4.1	Corpus MutiWikiNER	37
4.4.2	Corpus LeNER-Br	39
4.4.3	Corpus Ulysses_br	40
4.5	Vetores de Palavras	41
4.5.1	Pt-Wiki-Fasttext	41

4.5.2	<i>FlairEmbeddings</i>	42
4.5.3	<i>BERTimbau</i>	42
5	RESULTADOS	44
6	CONCLUSÕES E TRABALHOS FUTUROS	48
6.1	Trabalhos futuros	49
	REFERÊNCIAS	50

Capítulo 1

INTRODUÇÃO

Somos capazes de identificar entidades, contexto, sentimentos e etc, em textos, imagens e sons com certa facilidade, no entanto, para os computadores essa é uma tarefa complicada e a forma mais comum de lidar com esse problema é estruturar os dados de forma que o computador consiga interpretá-los, como é feito em bancos de dados estruturados. As técnicas mais comuns se dão através da intervenção humana na estruturação dos dados, para que esses possam então, ser interpretados pelo computador.

Em bancos de dados estruturados são feitas manualmente relações entre os dados, com as quais o computador consegue trabalhar. Um exemplo é uma simples planilha de *Excel* para registro de clientes, onde cada dado é relacionado a uma coluna/linha que define o que aquele dado representa e suas relações. Dessa forma, é possível buscar informações ou gerar relatórios como idade dos clientes, sexo, renda, localização, entre diversas outras relações que podem ser feitas a partir da estrutura de dados disponível.

1.1 Processamento de Linguagem Natural (PLN) e Reconhecimento De Entidades Nomeadas (REN)

O problema de estruturar os dados manualmente se dá principalmente devido ao grande volume de dados digitais que são gerados de forma não estruturada, como em redes sociais, artigos, revistas, documentos jurídicos entre outros. O Processamento de Linguagem Natural (PLN) é um processo automatizado que trabalha com textos não estruturados e consegue extrair informações desses dados e estruturá-los, (MAYNARD; BONTCHEVA; AUGENSTEIN, 2016) ou ainda, segundo Eisenstein (2019) o PLN é um conjunto de métodos que faz a linguagem humana acessível a computadores.

O objetivo principal do PLN é permitir que os computadores possam entender, interpretar e gerar texto em linguagem natural, como um ser humano. Isso significa que o PLN busca desenvolver algoritmos e técnicas que possam analisar textos, discursos, conversas

e outros tipos de linguagem humana, de forma a poder utilizá-los em diversas aplicações práticas como: assistentes pessoais, *chatbots*, motores de busca, atendentes virtuais, entre outras.

De acordo com [Kang et al. \(2020\)](#) entre os principais objetivos do PLN estão:

- Compreender o significado e contexto das palavras em um texto ou conversa, de forma a poder interpretá-las corretamente e responder de forma adequada.
- Analisar sentimentos em textos, para identificar emoções e opiniões expressas por um usuário.
- Extrair informações relevantes de um texto, como nomes, data, locais e outras informações importantes.
- Classificar textos em diferentes categorias, como positivo, negativo ou neutro, por exemplo.
- Gerar texto automaticamente, como em respostas automáticas de *e-mail* ou na criação de textos para *marketing*.

O PLN é uma área de pesquisa interdisciplinar que envolve conhecimentos em linguística, ciência da computação, inteligência artificial, psicologia e outras áreas relacionadas. As técnicas de PLN têm aplicações em diversas áreas, como atendimento ao cliente, análise de dados, tradução automática, educação, medicina, entre outras. O objetivo final do PLN é criar sistemas computacionais capazes de interagir com seres humanos de forma natural, facilitando a comunicação e aumentando a produtividade e a eficiência em diversas áreas.

O Reconhecimento de Entidades Nomeadas (REN) é uma tarefa computacional que consiste em extrair informações relevantes de um texto, como nomes, data, locais e outras informações importantes, as denominadas Entidades Nomeadas (EN) predefinidas ([GHADDAR; LANGLAIS, 2018](#)). As EN, como visto anteriormente, são nomes próprios que se referem a entidades do mundo real, o REN detecta os limites das ENs, o que possibilita identificá-las no texto e identificar a qual tipo de EN ela pertence ([MAYNARD; BONTCHEVA; AUGENSTEIN, 2016](#)), como é possível ver no exemplo detalhado abaixo:

"Barack Obama foi o presidente dos Estados Unidos eleito pelo Partido Democrata nas eleições de 2008"

"<Tipo=Pessoa **Barack Obama**> foi o <Tipo=pessoa **presidente**> dos <Tipo=Local **Estados Unidos**> eleito pelo <Tipo=Organização **Partido Democrata**> nas <Tipo=Evento **eleições**> de <Tipo=Número **2008**>."

Neste exemplo os tipos de EN predefinidas utilizadas foram: Pessoa, Local, Organização, Evento e Número. O REN é capaz de identificar nessa frase palavras que correspondem a esses tipos de entidades. Além disso é capaz de identificar uma continuação de uma entidade que esteja segmentada no texto, como é o caso da entidade do tipo Pessoa, que aparece primeiro em <Tipo=Pessoa **Barack Obama**> e depois em <Tipo=pessoa **presidente**>. Perceba o uso da primeira letra em minúsculo representando uma sequência em relação ao tipo de entidade.

1.2 Justificativa

Técnicas de PLN são essenciais no contexto atual da internet. São mais de cinco bilhões de usuários, mais de um bilhão e novecentos milhões de sites, mais de seis bilhões de buscas feitas diariamente no *Google*¹, mais de seiscentos milhões de *Tweets*² enviados por dia e mais de duzentos bilhões de e-mails enviados diariamente ([Internet Live Stats, 2021](#)).

Dentro do grande volume de dados gerados diariamente, estão textos em linguagem natural, ou seja, dados não estruturados como: artigos científicos, jornais, revistas, livros, decisões judiciais, relatórios diversos, páginas, postagens e comentários em redes sociais. Conseguir estruturar esses dados de forma manual se tornou totalmente inviável, embora essa estruturação seja essencial para tarefas corriqueiras do dia a dia, como uma busca no *Google*, que consegue sugerir bons resultados e completa sua busca com dados das buscas que estão sendo feitas, ou uma poderosa ferramenta de *marketing* que trabalha justamente, extraíndo informações das nossas buscas, textos, entre muitos dados que geramos ao navegar na internet.

Outro exemplo de uso da tarefa REN foi o explorado por [Junior e Claro \(2011\)](#) que a utilizou no domínio legal em análises forenses, na identificação de entidades em perícia de dados apreendidos de operações da polícia federal, demonstrando que com o uso de REN é possível se obter resultados similares aos obtidos a partir da análise manual dos dados feita pelos agentes, no entanto de forma consideravelmente mais rápida e com menor custo de trabalho humano. Já [Silva et al. \(2021\)](#) utilizou a tarefa REN para conseguir identificar entidades em anúncios em parceria com as Lojas Americanas com o intuito de melhorar a classificação dos anúncios de seu *marketplace*.

O grande volume de dados rotulados e não rotulados, e o alto número de pesquisas, faz com que a tarefa REN esteja bem desenvolvida para a língua inglesa, cenário diferente da língua portuguesa, que carece de dados anotados, pesquisas e melhorias nos modelos atuais para conseguir um bom desempenho nessa e em outras tarefas de PLN. Em domínio geral, enquanto estão disponíveis modelos com *F-score* de 94.6% na língua inglesa ([WANG et al.](#),

¹ <www.google.com>

² <<https://twitter.com>>

2021), na língua portuguesa existem modelos com *F-scores* de 83.22% (CASTRO *et al.*, 2019).

Sendo assim, este trabalho treina modelos que contribuíram tanto para compreender a influência do uso dos Vetores de Palavras, quanto para disponibilizar modelos de REN para o português, com resultados competitivos treinados nos Corpora *MultiWikiNER* para domínio geral, *LeNER-Br* e *PL-Corpus* para o domínio legal.

1.3 Objetivos

1.3.1 Objetivo Geral:

Verificar como o Aprendizado Profundo (AP) pode ser aplicado em conjunto com diferentes Vetores de Palavras e suas concatenações e atingir um novo estado da arte na tarefa REN para o português em diferentes corpora.

1.3.2 Objetivos Específicos:

- Realizar estudo acerca de arquiteturas de REN, baseadas em AP;
- Avaliar a partir dos estudos de arquiteturas realizados, modelos de REN para a língua portuguesa treinados em corpora de referência do idioma;
- Avaliar o uso de Vetores Estáticos, Vetores de Contexto e a concatenação desses, nos Corpora disponíveis de forma aberta para experimentos: *MultiWikiner* (NOTHMAN *et al.*, 2013), *LeNER-Br* (ARAUJO *et al.*, 2018) e *PL-Corpus* que é parte do projeto *UlyssesNER-Br* (ALBUQUERQUE *et al.*, 2022).

1.4 Hipótese de Pesquisa

A partir da análise dos melhores modelos de REN disponíveis para o português, acreditamos que o uso de diferentes Vetores de Palavras, em especial Vetores de Contexto e a concatenação de diferentes vetores, pode trazer bons resultados nos modelos de REN para o português.

1.5 Contribuições

Trazemos um grande número de testes explorando diferentes Vetores de Palavras e modelos que resultaram em um novo estado da arte, com resultados de média F1 98,88% para o domínio geral no Corpus *MultiWikiner* e no domínio legal, 89,31% para o Corpus *LeNER-Br* e 86,86% para o *PL-Corpus*. Todos os códigos para replicação e o melhor modelo de cada corpus treinado para a língua portuguesa a partir da realização deste trabalho, estão

disponíveis em repositório *Github*³. Bem como o melhor modelo treinado de cada Corpus está disponível em repositório *Huggingface*⁴.

Também foi feita junto a outros pesquisadores a publicação do *survey* (OLIVEIRA, 2023), intitulado "*Named Entity Recognition: a Survey for the Portuguese Language*" na revista "*Sociedad Española para el Procesamiento del Lenguaje Natural*", sobre a tarefa REN para o português.

1.6 Organização da Dissertação

O trabalho está organizado em seis capítulos, no Capítulo 2 iniciamos revisando os conceitos relacionados ao PLN sendo eles: Representação Textual, Aprendizado Profundo (AP), Redes Neurais Artificiais (RNA), Corpus, Vetores de Palavras e Modelos de Linguagem. No Capítulo 3 são apresentados e analisados os trabalhos relacionados, onde identificamos as lacunas e os caminhos mais promissores, em busca de preencher as lacunas e colaborar com o desenvolvimento da tarefa REN para o português. No Capítulo 4 é apresentada a metodologia e a modelagem dos experimentos realizados, no Capítulo 5 são apresentados os resultados dos experimentos e, por fim, no Capítulo 6 são feitas as considerações finais e apontados os trabalhos futuros.

³ <<https://github.com/Darkm4ge/Avaliacao-da-Influencia-do-uso-de-Embeddings-no-Reconhecimento-de-Entidades-Nomeadas-para-o-Portugue>>

⁴ <<https://huggingface.co/CarlosGomes>>

Capítulo 2

FUNDAMENTAÇÃO TEÓRICA

2.1 Representação Textual

A representação das palavras em textos, com o uso de caracteres como: letras, números e outros símbolos como pontuações, é bem estabelecida com suas línguas, regras gramaticais e semânticas, mas isso para a compreensão humana. Para representar textos de forma compreensível para um computador a forma mais comum é a representação binária, onde cada caractere é codificado em cadeias de bits. Um exemplo é a tabela *UNICODE*, que faz uma representação completa dos caracteres utilizados na escrita em cadeias binárias de 0 e 1 ([BERTAGLIA, 2017](#)).

Ainda segundo [Bertaglia \(2017\)](#), as cadeias binárias como o *UNICODE* são pouco úteis para o PLN, pois não guardam relação com a linguagem humana, nem a correlação entre caracteres, muito menos entre as palavras e o contexto, algo importante para compreensão de texto, que é um dado sequencial, de forma que em uma frase como: "Fui ao Banco sacar dinheiro", as palavras que vem antes da palavra Banco e depois dela influenciam no seu sentido.

Uma das formas de representação textual possível, é a utilização de vetores numéricos conhecidos como *Word Embeddings* ou Vetores de Palavras para representar as palavras. Uma das técnicas pioneiras foi o *one-hot* que cria um identificador binário único para cada palavra, que veio a ser aprimorado com a técnica *bag-of-words* que passa a levar em consideração a quantidade de vezes que a palavra aparece no texto. O uso de Vetores de Palavras trouxe consideráveis vantagens no PLN, principalmente em conjunto com arquiteturas de AP baseadas em redes neurais artificiais, como destaca [Turian, Ratinov e Bengio \(2010\)](#).

Outros autores também sugerem a normalização das entradas como é possível ver em [Bertaglia \(2017\)](#), e o uso de recursos léxicos e de capitalização como apresentado em [Ghadar e Langlais \(2018\)](#), além da representação por caracteres como no *embedding FastText* ([BOJANOWSKI et al., 2017](#)).

2.2 Aprendizado Profundo (AP)

Em [Deng e Yu \(2014\)](#) *Aprendizado Profundo* (AP) é definido como, "*Uma classe de técnicas de aprendizado de máquina que explora muitas camadas de processamento de informações não lineares para extração e transformação de recursos*". Esse aprendizado pode ser supervisionado, semi-supervisionado ou não supervisionado e também possibilita análise e classificação de padrões. Sendo assim, para melhor compreender o AP, primeiro precisamos falar um pouco de aprendizado de máquina.

O princípio por trás do aprendizado de máquina é que um software seja capaz de aprender com as experiências e se aprimorar, como por exemplo um software médico que consiga identificar qual tratamento tem mais resultado para uma doença, baseado nos dados de tratamentos anteriores, ou ainda um algoritmo de indicação de filmes e séries em um serviço de *stream*, que pode melhorar suas indicações baseado nos filmes e séries que o usuário assistiu anteriormente, como visto em [Mitchell \(1997\)](#), que faz a seguinte definição: "Diz-se que um programa de computador aprende com a experiência E no que diz respeito a alguma classe de tarefas T e medida de desempenho P , se seu desempenho nas tarefas em T , conforme medido por P , melhora com a experiência E ."

Também é possível dizer que o AP foi uma evolução natural do Aprendizado de Máquina para lidar com problemas complexos aplicando múltiplas camadas de processamento. Nesse trabalho usaremos Inteligência Artificial (IA), mais especificamente AP, para a tarefa de Reconhecimento de Entidades Nomeadas (REN), um processo muito similar ao de entidade e relacionamento, aplicado em bancos de dados estruturados, no entanto de forma automatizada.

Foi possível observar através da revisão bibliográfica que atualmente o uso de AP para a tarefa REN é ideal, como veremos ao analisar o estado da arte nessa tarefa, tanto no inglês, língua altamente explorada, quanto no idioma português. Visando contribuir com o baixo número de pesquisas na língua portuguesa e aprimorar o desempenho dos modelos atuais, trabalharemos com a arquitetura *BiLSTM* (*Bidirectional long short term memory* ou Memória longa a curto prazo bidirecional), e realizaremos experimentos com diferentes Corpora e Vetores de Palavras.

2.2.1 Redes Neurais Artificiais (RNA)

Como o nome sugere, as RNAs são redes baseadas no funcionamento dos neurônios biológicos que são capazes de aprender com a experiência, e sua história como é possível ver em [Strachnyi \(2019\)](#) inicia em 1943 quando McCulloch, um neurofisiologista, e Walter Pitt um matemático escreveram um artigo descrevendo e modelando o funcionamento de um neurônio biológico. Em 1949, Donald Hebb em seu livro *The Organization of Behavior* aprimorou o conceito e acrescentou que as ligações feitas são fortalecidas com a recorrência.

Já na década de 1950 Nathaniel Rochester do laboratório de pesquisas da *IBM*, fez o primeiro esforço de simulação computacional de uma rede neural.

Em 1958, Frank Rosenblatt um neurobiologista de Cornell, iniciou seu trabalho no *Perceptron*, um *Hardware* que simula um neurônio com pesos de ativação, que permite o neurônio artificial lidar com problemas de classificação em duas classes, o que resultou em uma rede de camada única. No decorrer dos anos muito se discutiu sobre esse modelo e muitas limitações foram encontradas e solucionadas.

Dando um salto nas pesquisas para 1997, uma rede recorrente de múltiplas camadas foi proposta por Schmidhuber e Hochreiter, a rede *LSTM*. Essa rede tem capacidade de trabalhar com problemas de classificação e regressão, como é a tarefa de REN que estamos trabalhando, e tem características de memória, que é uma importante característica para trabalhar com dados sequenciais como textos.

Atualmente ao pesquisar sobre a tarefa REN é possível ver que arquiteturas de RNAs como a *LSTM* são fortes opções para essa tarefa. Como exemplo, em Ghaddar e Langlais (2018) é aplicada uma rede do tipo *BiLSTM* que aprimora ainda mais a capacidade de lidar com dados sequenciais, rodando duas redes *LSTM* em direções opostas, dando a habilidade de considerar em um texto por exemplo, não só as palavras que vieram antes, mas também as que virão na sequência da frase reforçando a capacidade de extrair contexto da mesma.

2.3 Corpus

Um Corpus é um conjunto de documentos textuais que é usado para treinar modelos de linguagem e sistemas de processamento de linguagem natural (PLN). No contexto do reconhecimento de entidades nomeadas, um corpus é uma coleção de textos que foram anotados com informações sobre as entidades nomeadas que aparecem neles.

Essas anotações podem incluir o tipo de entidade nomeada (por exemplo, pessoa, localização ou organização), bem como a localização exata da entidade dentro do texto. Os corpus geralmente são criados por especialistas em linguística ou por anotadores humanos que identificam manualmente as entidades nomeadas nos textos, como é o caso dos corpora de domínio legal LeNER-Br (ARAÚJO *et al.*, 2018) e Pl-Corpus que é parte do projeto UlyssesNER-Br (ALBUQUERQUE *et al.*, 2022). Outro método de anotação que vem sendo explorado é a criação automática de anotações utilizando modelos de linguagem ou utilizando estruturas de links como as da Wikipédia, como é o caso do corpus de domínio geral *MultiWikiNER* (NOTHMAN *et al.*, 2013).

Os modelos de reconhecimento de entidades nomeadas são treinados usando esses corpus anotados para aprender a identificar e classificar entidades nomeadas em textos novos não vistos anteriormente. Quanto maior e mais diverso o corpus, melhor o modelo pode

ser treinado para reconhecer entidades nomeadas em diferentes contextos e tipos de texto.

2.4 Vetores de Palavras e Modelos de Linguagem

Os Vetores de Palavras, também conhecidos como *Word Embeddings*, são uma técnica popular e eficaz usada em modelos de linguagem para representar palavras em um espaço vetorial contínuo. Essa técnica transforma palavras em vetores numéricos que podem ser manipulados pelos algoritmos de aprendizado de máquina. Os modelos de linguagem que utilizam vetores de palavras são capazes de entender melhor a semântica das palavras, permitindo a captura de relações semânticas e sintáticas entre elas. Isso ocorre porque as palavras são representadas em um espaço vetorial que permite a realização de operações matemáticas, como soma, subtração e multiplicação, que podem refletir as relações semânticas entre elas.

Por exemplo, se forem considerados os vetores correspondentes às palavras "rei" e "rainha", é possível subtrair o vetor de "rei" do vetor de "rainha" para obter um vetor que representa a relação de gênero entre as palavras. Essas operações mostram como os vetores de palavras podem ser usados para capturar relações semânticas entre as palavras.

Existem várias técnicas para criar vetores de palavras, incluindo os populares métodos *Word2Vec* (MIKOLOV; YIH; ZWEIG, 2013) e *FastText* (BOJANOWSKI *et al.*, 2017), que utilizam redes neurais para aprender representações vetoriais de palavras. Esses algoritmos são capazes de aprender vetores de palavras com base em grandes conjuntos de dados textuais, onde as palavras são representadas em um espaço vetorial que leva em consideração a sua distribuição em um contexto.

Embora os vetores de palavras tenham provado ser uma técnica poderosa para modelos de linguagem, eles não são isentos de limitações. Por exemplo, eles não conseguem lidar com palavras que têm múltiplos significados, como "banco", que pode se referir tanto a um banco financeiro quanto a uma estrutura de madeira para sentar. Além disso, a representação vetorial de uma palavra pode variar dependendo do contexto em que ela aparece, o que pode levar a imprecisões.

Por fim, os vetores de palavras são uma técnica popular e eficaz usada em modelos de linguagem para representar palavras em um espaço vetorial contínuo. Essa técnica é capaz de capturar relações semânticas e sintáticas entre as palavras, permitindo que os modelos de linguagem entendam melhor o significado das palavras. No entanto, existem limitações que precisam ser consideradas ao utilizar essa técnica.

Os Vetores de Contexto, também conhecidos como *Contextualized Embeddings* como *GloVe* (PENNINGTON; SOCHER; MANNING, 2014) e *BERT* são técnicas mais avançadas de representação de palavras em modelos de linguagem, que permite superar algumas das li-

mitações dos vetores de palavras tradicionais. Ao contrário dos vetores de palavras tradicionais, que representam cada palavra como um vetor fixo, os vetores de contexto levam em consideração o contexto em que a palavra aparece para gerar sua representação vetorial. Isso significa que a representação vetorial de uma palavra pode variar dependendo do contexto em que ela aparece, permitindo capturar melhor a sua semântica.

O *BERT* (*Bidirectional Encoder Representations from Transformers*) por exemplo é um modelo pré-treinado que utiliza uma rede neural bidirecional chamada *transformers* para aprender representações vetoriais de palavras com base em grandes conjuntos de dados textuais. Durante o treinamento, o modelo é exposto a pares de sentenças e aprende a prever a próxima palavra em uma dessas sentenças com base no contexto global em que ela aparece. Esse processo permite ao modelo aprender uma representação vetorial de cada palavra que leva em consideração o contexto em que ela aparece.

Além disso, tanto o *BERT* quanto outros Vetores de Contexto são capazes de lidar com palavras polissêmicas, ou seja, palavras que têm múltiplos significados. Isso ocorre porque o modelo considera todas as instâncias de uma palavra em um determinado contexto e aprende a gerar uma representação vetorial que é sensível ao sentido específico da palavra nesse contexto. Os Vetores de Contexto têm sido usados com sucesso em tarefas de PLN, como REN, resposta a perguntas e geração de texto. Essa técnica tem sido especialmente útil para melhorar o desempenho em tarefas que exigem uma compreensão mais profunda da semântica das palavras e do contexto em que elas aparecem.

O *FastText* é um modelo de Vetor de Palavras proposto pelo Facebook AI Research em [Bojanowski et al. \(2017\)](#). Ele é capaz de lidar com palavras raras e desconhecidas, aprendendo a representação de subpalavras (n-gramas de caracteres) em vez de representar cada palavra como um todo. Essa abordagem permite que o modelo aprenda a relação entre palavras com base em suas subpalavras compartilhadas. Os Vetores Estáticos *FastText* foram pré-treinados em dados da Wikipédia para capturar diferentes nuances.

Já o *Global Vectors for Word Representation (GloVe)*, desenvolvido por [Pennington, Socher e Manning \(2014\)](#) é outro modelo de Vetores de Palavras que utiliza estatísticas de coocorrência para aprender representações de palavras. O *GloVe* usa uma matriz de coocorrência para representar a relação entre as palavras no vocabulário. Em seguida, usa uma técnica de decomposição de valores singulares para aprender as representações de palavras, o que permite que o modelo capture a relação semântica entre as palavras.

Uma das principais diferenças entre os vetores *FastText* e *GloVe* é a forma como eles representam as palavras. Enquanto o *FastText* representa palavras como uma combinação de sub palavras, o *GloVe* representa palavras como vetores de números reais aprendidos a partir de uma matriz de coocorrência.

Além disso, é importante destacar a diferença entre Vetores Estáticos e de Contexto.

Os Vetores Estáticos são treinados em grandes quantidades de texto e, portanto, são independentes do contexto em que aparecem. Em outras palavras, a representação de uma palavra é a mesma, independentemente do contexto em que é usada. Por outro lado, os Vetores de Contexto levam em consideração o contexto em que uma palavra aparece e, portanto, geram uma representação diferente para a mesma palavra em diferentes contextos.

Outra técnica utilizada para desenvolver representações vetoriais de palavras é o uso de modelos de linguagem como o *BERT* (*Bidirectional Encoder Representations from Transformers*), desenvolvido e apresentado por [Devlin et al. \(2018\)](#), *BERTimbau*, proposto em [Souza, Nogueira e Lotufo \(2020\)](#) e o *ELMO* (*Embeddings from Language Models*) desenvolvido e apresentado por [Peng, Yan e Lu \(2019\)](#). Esses são três exemplos de modelos de linguagem populares que usam técnicas de AP para aprenderem a representação de palavras em um texto. O *BERT* foi treinado em uma grande quantidade de dados não rotulados, enquanto o *ELMO* usa uma abordagem de pré-treinamento em que o modelo é treinado em tarefas auxiliares, como prever a próxima palavra em uma frase.

Ambos os modelos, *BERT* e *ELMO* alcançaram resultados impressionantes em várias tarefas de processamento de linguagem natural e são amplamente usados em aplicações como *chatbots*, assistentes virtuais, análise de sentimentos, entre outras. A principal vantagem desses modelos é que eles podem ser usados para entender melhor o significado das palavras em um texto, levando em consideração o contexto em que elas aparecem.

2.4.1 Concatenação de Vetores de Palavras

Diversos modelos atuais conforme vimos nos trabalhos relacionados, utilizam a concatenação de vetores de palavras. A concatenação é uma técnica usada no PLN para combinar múltiplas representações de palavras ou frases em um único vetor de alta dimensão. O objetivo é capturar informações complementares e ricas que podem ser úteis para tarefas de PLN, como classificação de texto, reconhecimento de entidades nomeadas ou análise de sentimentos.

Uma das características notáveis do *framework Flair* ([AKBIK et al., 2019](#)) é a abordagem de concatenação de vetores denominados *Stacked Embeddings*, que se baseia na concatenação de diferentes tipos de vetores para formar representações mais abrangentes das palavras ou frases.

A ideia por trás da concatenação de vetores é utilizar vetores de contextuais, como os gerados por modelos como *BERT*, *ELMo* ou *GPT*, juntamente com vetores não-contextuais pré-treinados, como *Word2Vec* ou *Fasttext*. Essa combinação permite que o modelo capture tanto informações específicas de contexto quanto informações semânticas gerais.

O processo de concatenação de vetores é geralmente realizado em várias camadas, onde cada camada é responsável por concatenar diferentes tipos de vetor. Por exemplo, na

primeira camada, pode-se concatenar um vetor não-contextual com um vetor contextual. Em camadas subsequentes, pode-se adicionar mais vetores contextuais ou não-contextuais, ampliando a capacidade do modelo de capturar diferentes nuances das palavras e frases.

A abordagem de concatenação de vetores tem algumas vantagens significativas. Ao combinar várias representações, o modelo pode melhorar a captura de informações complexas em diferentes níveis, o que pode ser benéfico para tarefas de PLN. Além disso, essa técnica permite que os vetores sejam treinados conjuntamente durante o processo de aprendizado, o que pode resultar em melhorias de desempenho.

No entanto, é importante notar que a concatenação de vetores também pode aumentar a dimensionalidade do espaço de representação, o que pode levar a um aumento no custo computacional e exigir mais dados para um treinamento efetivo. Portanto, a seleção cuidadosa dos tipos de vetores a serem combinados e a quantidade de camadas a serem utilizadas são decisões importantes no uso da técnica. Dessa forma, a concatenação de vetores, é uma técnica valiosa no processamento de linguagem natural que pode melhorar o desempenho do modelo, capturando informações contextuais e semânticas importantes para várias tarefas de PLN.

Capítulo 3

TRABALHOS RELACIONADOS

Neste capítulo, são apresentados estudos que obtiveram resultados competitivos na tarefa REN em diferentes corpora, utilizando arquiteturas de AP, vetores estáticos de palavras, vetores de contexto e a concatenação de diferentes vetores de palavras. Essa apresentação demonstra que o uso de vetores de palavras e suas combinações representa um avanço significativo na tarefa de REN, especialmente para a língua portuguesa.

Quanto à arquitetura para a tarefa de REN, existiram diversos estudos, como o de [Amaral \(2013\)](#), que demonstrava o uso do classificador *CRF* - acrônimo para *Conditional Random Fields*, como uma arquitetura viável naquele momento. No entanto, com o avanço das redes neurais, aprendizado de máquina e, finalmente, aprendizado profundo, estudos como o de [Akbik, Blythe e Vollgraf \(2018\)](#) mostraram que o uso do *CRF* como camada de saída em arquiteturas de RNA como: Redes Neurais Convolucionais (CNN - acrônimo para *Convolutional Neural Networks*) e Redes Neurais *LSTM* é uma combinação consolidada e continua gerando bons resultados até hoje.

Segundo [Akbik, Blythe e Vollgraf \(2018\)](#), a arquitetura *LSTM* em sua variação bidirecional com uma camada *CRF* de saída, chamada *BiLSTM-CRF*, representa o estado da arte para tarefas de rotulação sequencial como REN na língua inglesa e alemã, conclusão que é compartilhada também com os trabalhos que discutiremos abaixo para o português. Em virtude disso, essa é a arquitetura utilizada em grande parte dos modelos nos trabalhos analisados e consequentemente nos testes realizados neste trabalho.

A arquitetura *BiLSTM-CRF* é um tipo de modelo de processamento de linguagem natural que combina uma rede *BiLSTM* com um decodificador *CRF*. A *BiLSTM* é uma arquitetura de rede neural recorrente que utiliza duas camadas *LSTM* executadas em direções opostas, uma para a frente e outra para trás, para capturar informações contextuais tanto anteriores quanto posteriores de uma sequência de entrada. Isso permite que o modelo tenha uma compreensão mais abrangente da sequência.

O *CRF*, por sua vez, é um modelo probabilístico que modela a dependência entre ró-

tulos em uma sequência. O decodificador *CRF* é usado para escolher a sequência de rótulos mais provável para uma dada sequência de entrada, considerando as dependências entre rótulos e as probabilidades de transição entre eles. Isso ajuda a garantir que o modelo produza sequências de rótulos coerentes e suaves. Portanto nessa arquitetura, a *BiLSTM* é usada para capturar informações contextuais e o *CRF* é usado para escolher a sequência de rótulos mais provável para a sequência de entrada. Juntas, essas técnicas podem levar a um desempenho aprimorado em tarefas de marcação de sequência.

Um componente importante nessa abordagem são os vetores de palavras. O estudo de [Akbik, Blythe e Vollgraf \(2018\)](#) propõe os *FlairEmbeddings*, um tipo de Vetor de Contexto ao nível de caractere, capaz de combinar as características consideradas importantes, como pré-treinamento em grandes volumes de dados não rotulados. Captura do significado da palavra no contexto e, portanto a produção de diferentes vetores de palavras para palavras polissêmicas, dependendo de seu uso. Por fim modelagem de palavras e contexto fundamentalmente como sequências de caracteres, tanto para lidar melhor com palavras raras e com erros ortográficos quanto para modelar estruturas de sub palavras como prefixos e terminações.

Segundos os autores a versão em língua inglesa do *Flair Embeddings* foi treinada no Corpus de 1 bilhão de palavras e o modelo em alemão, em um corpus de meio bilhão de palavras agregadas de vários corpora de código aberto do projeto OPUSproject2. Os *Flair Embeddings* pré-treinados foram disponibilizados pelos autores, incluindo uma versão em português, utilizada nos testes deste trabalho. Portanto, serão abordados os *Flair Embeddings* em mais detalhes na descrição dos experimentos aqui realizados.

A referida técnica cria a partir de uma arquitetura de modelo de linguagem ou *Language Model (LM)*, *BiLSTM-LM* ao nível de caractere, dois vetores de palavras, que são então concatenados resultando em um Vetor de Contexto, que os autores chamam de *Flair Embeddings*. A capacidade de concatenar vetores de palavras não se restringe aos vetores de contexto, e também possibilita concatenar outros vetores de palavras estáticos e ou de contexto. Um exemplo é o vetor estático *Glove* e o vetor de contexto *Flair Embeddings*, combinação que atingiu um novo estado da arte para REN no Inglês, atingindo F1 de 93,07% e 88,20% para o Alemão nos resultados de [Akbik, Blythe e Vollgraf \(2018\)](#), no CoNLL03.

Já no trabalho de [Akbik et al. \(2019\)](#), o *Framework FlairNLP* é desenvolvido e apresentado, prometendo treinar modelos ao nível do estado da arte para tarefas de rotulagem sequencial, entre elas REN, com uso da arquitetura *BiLSTM-CRF*. O *FlairNLP* possibilita como diferencial, uma forma simples de concatenar diferentes Corpora e Vetores de Palavras, como os que o *framework* considera como clássicos ou estáticos: *Glove* e *FastText*, e muitos outros Vetores de Contexto, como: *Flair Embeddings*, *ELMO* e o *BERT*.

A seguir é apresentado, como trabalhos de REN em português, utilizaram arquiteturas de AP e obtiveram resultados a nível de estado da arte em seus respectivos domínios,

observando em especial o uso de diferentes vetores de palavras.

3.1 Vetores de Palavras para REN em Português

No trabalho de Santos *et al.* (2019), foi utilizada a arquitetura *BiLSTM-CRF* para medir o impacto do uso de Vetores de Contexto e outras combinações de vetores, para a tarefa REN no português, atingindo um novo estado da arte no *Benchmark* do corpus HAREM. Foram comparados 16 cenários diferentes, sendo eles divididos em dois grupos treinados em diferentes corpora: o primeiro grupo treinado na combinação dos Corpora brWaC, *BlogSet-BR* e *ptwiki-20190301*, e o segundo utilizando os Corpora NILC's.

O Vetor de Contexto, chamado pelos autores de *FlairBBP*, foi treinado com a combinação dos corpora brWaC, *BlogSet-BR* e *ptwiki-20190301*, seguido da concatenação dos vetores estáticos *Word2Vec Skip-gram*. O *FlairBBP* obteve vantagem sobre o então estado da arte na tarefa REN de 4,31% e 5,99% no HAREM completo e no seletivo, respectivamente, obtendo F1 de 82,26% no cenário seletivo.

Os autores comparam seus resultados a dois outros modelos no *Benchmark* do HAREM, um utilizando uma arquitetura CharWNN e outro utilizando a arquitetura *BiLSTM-CRF*. Nessa comparação percebemos o destaque da arquitetura *BiLSTM-CRF* que já abria vantagem de aproximadamente 5% em relação a CharWNN em ambos os cenários do HAREM. O uso do conjunto de vetores de palavras, também influenciou positivamente no modelo, ampliando a vantagem da arquitetura *BiLSTM-CRF* em outros aproximados 5% em ambos cenários. Os autores também citaram como vantagem do uso de Vetores de Contexto, uma maior estabilidade do modelo apresentando menor variação em diferentes treinos.

Ainda no domínio geral, o trabalho de Castro *et al.* (2019) apresenta um novo estado da arte no Corpus HAREM com o script de avaliação CoNLL, atingindo F1 de 83,22% no cenário seletivo, o autor utilizou para isso a arquitetura de AP ELMO com ajuste fino com o corpus brWaC, junto a uma rede CNN a nível de caractere, com o Vetor Estático *Wang2Vec Skip-Gram (ELMo-af-brWaC+CNN+Wang2Vec-SG)*.

Em Consoli *et al.* (2020) a arquitetura *BiLSTM-CRF* foi utilizada para a tarefa REN em português no domínio da Geologia, no Corpus Geocorpus-2, único corpus em português para o domínio da Geologia. Os autores também testaram diferentes combinações de vetores de palavras, como vetores estáticos e vetores de contexto. Foram feitos 8 testes, os vetores estáticos testados foram *GeoWE* com F1 de 53,71% e *Word2Vec Skip-Gram* com F1 de 71,33%. Dos vetores de contexto, foram testados o *FlairBBP* treinado com a combinação dos corpora brWaC, *BlogSet-BR* e *ptwiki-20190301*, com F1 de 83,10%. Uma variação denominada pelos autores de *FlairBBPGeoFT*, que consistiu em um *finetunning* do *FlairBBP* utilizando o Geocorpus-2, obteve uma pequena melhora com F1 de 84,20%. Por fim, no conjunto de testes com vetores concatenados, foram testadas as combinações

GeoWE+FlairBBP com F1 78,84% e *W2V-SKPG+FlairBBP* com F1 84,04% e uma variação de cada um dos dois anteriores substituindo *FlairBBP* por *FlairBBPGeoFT*, obtendo F1 de 83,74% no *GeoWE+FlairBBPGeoFT* e o melhor resultado entre os testes com F1 de 84,63% para o *W2V-SKPG+FlairBBPGeoFT*.

No trabalho de [Batista et al. \(2021\)](#), a arquitetura *BiLSTM-CRF* foi utilizada também com o intuito de testar a influência de diferentes combinações de Vetores de Palavras. Foram treinados e comparados 7 modelos com corpus de domínio jurídico, produzido pelos autores do trabalho a partir de petições iniciais extraídas de processos provenientes de tribunais brasileiros. O intuito era utilizar a tarefa REN para identificar entidades em petições iniciais, portanto, não houve comparação com outros trabalhos, mas em relação às diferentes combinações de vetores de palavras. De acordo com os autores, a combinação dos Vetores de Palavras: *Glove* + Vetores de Caracteres nas palavras não classificadas pelo *Glove* + Vetores de Contexto (*Flair Embeddings*), foi a configuração que atingiu melhor resultado com F1 de 86,74%, um resultado bastante satisfatório segundo os autores.

O trabalho de [Albuquerque et al. \(2022\)](#) apresenta o Corpus UlyssesNER-Br, um corpus do domínio legal em português, segmentado em duas partes o PL-Corpus e o ST-Corpus. Os corpora foram produzidos a partir de duas fontes de documentos legais da Câmara dos Deputados, os Projetos de Lei, disponíveis no site da Câmara, dando origem ao PL-Corpus. As solicitações de trabalho, documentos sigilosos que contém pedidos de Deputados sobre a situação de determinado tema nas tramitações da Câmara, esses por sua vez dando origem ao ST-Corpus. O ST-Corpus, em virtude dos dados utilizados no treinamento, não está disponível de forma livre.

Além de desenvolver e apresentar o Corpus UlyssesNER-Br, os autores executaram o treinamento de três modelos para a tarefa NER. Foram dois modelos clássicos, o *HMM* e o *CRF*, e um modelo de AP com a arquitetura *BiLSTM-CRF* e o Vetor Estático *Glove*. As médias F1 para o PL-Corpus foram de 44,56% para o modelo *HMM*, 76,30% para o *CRF* e 76,89% para a *BiLSTM-CRF*. No ST-Corpus, as médias F1 foram respectivamente 18,36%, 61,32% e 59,67%. O PL-Corpus, também faz parte do conjunto de testes executados neste trabalho.

Outro trabalho que desenvolve, apresenta e treina modelos para NER no domínio legal em português é o [Araujo et al. \(2018\)](#), no qual os autores produziram um corpus chamado LeNER-Br, composto por documentos de cortes Brasileiras como STF e STJ. O melhor modelo treinado pelos autores no corpus foi uma arquitetura *BiLSTM-CRF* com o Vetor Estático *Glove*, atingindo uma média F1 de 92,53%. O Corpus LeNER-Br também fez parte do conjunto de experimentos, portanto teremos mais detalhes sobre ele mais a frente.

Ainda no domínio jurídico, [Mota et al. \(2021\)](#) desenvolveu um Corpus próprio para o português a partir de petições iniciais. Os autores compararam as configurações padrão de três sistemas, uma rede CNN com conexões residuais e usando *Bloom Embeddings* na ferramenta *Spacy*, e duas redes *BiLSTM-CRF*, uma com o *Framework TensorFlow* utilizando como

Vetor Estático o *FastText* a nível de caractere, e outra com o *Framework FlairNLP* utilizando os *Flair Embeddings*.

Segundo os autores, os resultados apresentados reforçam mais uma vez a percepção da superioridade da arquitetura *BiLSTM-CRF*, com as duas redes atingindo resultados bem superiores a rede CNN. A rede CNN obteve F1 de 38%, a rede *BiLSTM-CRF* com o *Framework TensorFlow* e Vetor Estático o *FastText*, obteve F1 de 72% e a com *Framework FlairNLP* utilizando os *Flair Embeddings* atingiu F1 de 76%.

Por fim, o trabalho de Santos *et al.* (2021) que também utilizou a arquitetura *BiLSTM-CRF*, no domínio da saúde para o português, mais especificamente utilizou-se a tarefa REN apenas com a entidade pessoa para identificar nomes de pacientes em documentos médicos e remover os mesmos, afim de remover essas identificações dos dados para poder utiliza-los preservando a identidade dos pacientes.

Neste caso, o *Framework FlairNLP* foi utilizado para testar diferentes conjuntos de Vetores de Palavras, e também para testar a concatenação dos Corpora para o treinamento do modelo. No que tange os corpora, os experimentos conduzidos consistiram em fatiar os corpora em partes e ir adicionando essas partes a cada treinamento, com o objetivo de identificar qual a porção ideal dos corpora e da concatenação deles. Para isso foram utilizados dois corpora, o *ERH-Names* corpus de domínio médico, produzido pelos autores e, o HAREM.

Em relação aos testes com os corpora, o melhor resultado foi utilizando o corpus *ERH-Names* completo, demonstrando a importância de um corpus robusto e nesse caso descartando a concatenação com o corpus HAREM, o que gradativamente reduziu a eficiência do modelo. Em relação aos Vetores de Palavras, foram feitos 7 experimentos com três variações de vetores de palavras e quatro combinações. São dois *W2V-SKPG*, um treinado com o corpora NILC (*WE-NILC*) e o outro com o corpus *EHR-Names* (*WE-EHR-Notes*), e o anteriormente citado no trabalho de (CONSOLI *et al.*, 2020) *FlairBBP*, além de uma variação, obtida através de *finetuning* com o corpus desenvolvido pelos autores, que foi nomeada de *FlairBBPFnTg*.

Os resultados do trabalho de Santos *et al.* (2021), demonstram que a concatenação de Vetores de Palavras e os Vetores de Contexto trazem bons resultados, sendo o melhor resultado encontrado no modelo que concatenou os vetores de palavras *FlairBBP+WE-NILC* atingindo F1 de 94,13%, seguido do *FlairBBPFnTg+WE-EHR-Notes* com 93,56%, *FlairBBPFnTg+WE-NILC* 93,37%, *FlairBBP+WE-EHR-Notes* com 93,19% e apenas o *FlairBBP* com F1 de 92,93%, enquanto os testes apenas com Vetores Estáticos *WE-NILC* e *WE-EHR-Notes*, atingiram respectivamente F1 de 88,36% e 80,59%.

3.2 Discussão Sobre os Trabalhos Relacionados

Para discutir os trabalhos apresentados, a Tabela 3.1 simplifica a visualização dos trabalhos, fornecendo uma visão geral dos modelos testados e seus resultados, destacando as arquiteturas utilizadas, os Corpora, Vetores de Palavras e as médias F1 atingidas.

Tabela 3.1 – Apresentação dos resultados apresentados nos trabalhos relacionados.

Trabalho	Arquitetura	Corpus	Vetor de Palavras	Média F1
(CASTRO <i>et al.</i> , 2019)	<i>BiLSTM-CRF</i>	HAREM I	<i>ELMo-af-brWaC+Glove</i>	82,89%
(CASTRO <i>et al.</i> , 2019)	<i>BiLSTM-CRF</i>	HAREM I	<i>ELMo-af-brWaC+CNN-SG Wang2Vec</i>	83,22%
(AKBIK; BLYTHE; VOLLGRAF, 2018)	<i>BiLSTM-CRF</i>	<i>1-billion word Corpus</i> (Inglês)	<i>Glove+FlairEmbeddings</i>	93,07%
(AKBIK; BLYTHE; VOLLGRAF, 2018)	<i>BiLSTM-CRF</i>	OPUS project2 (Alemão)	<i>Glove+FlairEmbeddings</i>	88,20%
(SANTOS <i>et al.</i> , 2019)	<i>BiLSTM-CRF</i>	NILC (HAREM SELETIVO)	FlairBBP+W2V-CBOW	81,27%
(SANTOS <i>et al.</i> , 2019)	<i>BiLSTM-CRF</i>	NILC (HAREM SELETIVO)	FlairBBP+FT-SKPG	80,91%
(SANTOS <i>et al.</i> , 2019)	<i>BiLSTM-CRF</i>	NILC (HAREM SELETIVO)	FlairBBP+FT-CBOW	76,76%
(SANTOS <i>et al.</i> , 2019)	<i>BiLSTM-CRF</i>	NILC (HAREM SELETIVO)	FlairEL+W2V-SKPG	80,80%
(SANTOS <i>et al.</i> , 2019)	<i>BiLSTM-CRF</i>	NILC (HAREM SELETIVO)	FlairEL+W2V-CBOW	81,48%
(SANTOS <i>et al.</i> , 2019)	<i>BiLSTM-CRF</i>	NILC (HAREM SELETIVO)	FlairEL+FT-SKPG	81,88%
(SANTOS <i>et al.</i> , 2019)	<i>BiLSTM-CRF</i>	NILC (HAREM SELETIVO)	FlairEL+FT-CBOW	75,89%
(SANTOS <i>et al.</i> , 2019)	<i>BiLSTM-CRF</i>	NILC (HAREM SELETIVO)	FlairBBP+W2V-SKPG	82,26%
(ARAUJO <i>et al.</i> , 2018)	<i>BiLSTM-CRF</i>	LeNER-Br	<i>Glove</i>	92,53%
Proposto	<i>BiLSTM-CRF</i>	LeNER-Br	<i>Pt-Wiki-Fasttext+ FlairEmbeddings-Pt</i>	89,02%
(ALBUQUERQUE <i>et al.</i> , 2022)	HMM	PL-Corpus	-	40,74%
(ALBUQUERQUE <i>et al.</i> , 2022)	CRF	PL-Corpus	-	76,28%
(ALBUQUERQUE <i>et al.</i> , 2022)	<i>BiLSTM-CRF</i>	PL-Corpus	<i>Glove</i>	76,89%
Proposto	<i>BiLSTM-CRF</i>	PL-Corpus	<i>Pt-Wiki-Fasttext+ FlairEmbeddings-Pt</i>	86,89%
(CONSOLI <i>et al.</i> , 2020)	<i>BiLSTM-CRF</i>	GeoCorpus-2	<i>W2V-SKPG+FlairBBPGeoFT</i>	84,63%
(BATISTA <i>et al.</i> , 2021)	<i>BiLSTM-CRF</i>	Legal (Próprio)	<i>CHAR+WORD+POOL</i>	86,74%
(MOTA <i>et al.</i> , 2021)	CNN	Legal (Próprio)	-	38%
(MOTA <i>et al.</i> , 2021)	<i>BiLSTM-CRF</i>	Legal (Próprio)	<i>FastText</i>	71%
(MOTA <i>et al.</i> , 2021)	<i>BiLSTM-CRF</i>	Legal (Próprio)	<i>FlairEmbeddings</i>	76%
(SANTOS <i>et al.</i> , 2021)	<i>BiLSTM-CRF</i>	ERH-Names (Médico/Próprio)	<i>FlairBBP+WE-NILC</i>	94,13%

Fonte: o autor

A Tabela 3.1 reafirma a prevalência da arquitetura *BiLSTM-CRF* em conjunto com Vetores de Palavras. Outro ponto que é possível observar é que a estratégia seguida por diversos trabalhos apresentados foi a de testar diferentes Vetores de Palavras e suas concatenações para melhorar os resultados de seus modelos. É possível perceber que o uso de Vetores de

Contexto, quase sempre representa melhoras, assim como o uso desses Vetores concatenados, portanto é justificável e promissor executar essa implementação em outros Corpus, tanto quanto para diferentes domínios.

Outra técnica utilizada no trabalho de [Castro et al. \(2019\)](#) e no trabalho de [Consoli et al. \(2020\)](#) foi o *finetuning*. No primeiro o modelo de linguagem *ELMo-af-brWaC+CNN-SG* é um Vetor de Contexto ELMo, pré-treinado a nível de caractere no corpus brWaC, com *finetuning* no corpus da tarefa, no caso o HAREM. No segundo caso é utilizado o Vetor de Contexto *FlairBBP* com *finetuning* no corpus da tarefa, no caso o GeoCorpus-2, em ambos os casos, o processo apresentou melhoria nos resultados segundo os autores.

Como pode-se observar neste capítulo, em especial na Tabela 3.1, para o domínio geral no português não se encontra modelos treinados em diferentes Corpus, se restringindo ao HAREM, de forma que é trago, portanto, o corpus *MultiWikiNER*, para ampliar o leque de opções de modelos de domínio geral para o português. Outro ponto observado é que a concatenação de vetores foi utilizada em diversos corpus desenvolvidos pelo próprios autores, mas que ainda não havia sido testado nos corpus de domínio legal LeNER-BR e PL-Corpus, outra lacuna preenchida com este trabalho.

Em face das constatações obtidas a partir da análise dos trabalhos, foi possível definir experimentos para este trabalho que considerem os avanços alcançados, conforme veremos no próximo capítulo, o que garantiu bons resultados nos testes, que apresentaram comportamento muito parecido aos analisados nos demais trabalhos, demonstrando a confiabilidade e robustez da arquitetura *BiLSTM-CRF*, e o ganho de desempenho no uso dos diferentes tipos de Vetores de Palavras.

Capítulo 4

METODOLOGIA E MODELAGEM DOS EXPERIMENTOS

Neste capítulo será descrito como foi feita a pesquisa bibliográfica e como foram montados os 10 experimentos que foram executados neste trabalho.

4.1 Metodologia

O levantamento bibliográfico se ancorou no trabalho (OLIVEIRA, 2023), este *survey* mapeou técnicas, métodos e recursos para a tarefa REN na língua portuguesa. O *survey* aplicou buscas manuais e automatizadas, recuperando 447 estudos primários, dos quais 45 foram incluídos na revisão. Destes, 21 estudos focaram na análise comparativa entre técnicas e ferramentas. Outros 24 apresentaram corpora REN novos ou atualizados, em diversos domínios. As técnicas de pré-processamento de texto mais utilizadas foram *tokenização*, vetores de palavras e *PoS Tagging*, enquanto os métodos/algoritmos mais utilizados foram baseados nos modelos *BiLSTM*, *CRF* e *BERT*.

Com base nas informações encontradas no recorte adaptado de Oliveira (2023), na Tabela 4.1 é possível dizer que a arquitetura de rede *BiLSTM-CRF* estava presente em 10 trabalhos, sendo a mais utilizada e tendo a melhor faixa de desempenho. Também foi possível identificar que as técnicas de *tokenização* e vetores de palavras estão presentes em diversos trabalhos e trazem melhorias de desempenho aos modelos.

Tabela 4.1 – Arquiteturas de rede

Arquitetura	Total	Média F1
<i>BiLSTM+CRF</i>	10	67% ~ 97%
<i>CRF</i>	7	48% ~ 97%
<i>CRF+LG</i>	6	53% ~ 76%
<i>BiLSTM</i>	3	53% ~ 93%

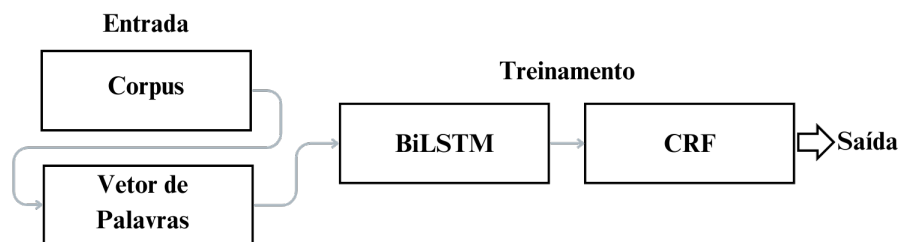
Fonte: Recorte e adaptação de (OLIVEIRA, 2023)

No capítulo dos trabalhos relacionados foram apresentados e discutidos diversos trabalhos que implementaram a arquitetura *BiLSTM-CRF* e os Vetores de Palavras, analisando os melhores resultados e a forma como os autores chegaram aos seus resultados. Foi definido então o *framework* para implementação, corpora e vetores de palavras a serem testados, a seguir veremos em detalhes a modelagem destes experimentos.

4.2 Modelagem dos Experimentos

A Figura 4.1, apresenta o *pipeline* de treinamento dos modelos, segmentando em entrada, onde são feitas as alterações nestes experimentos, treinamento na *BiLSTM* e a classificação no *CRF* que emite a saída.

Figura 4.1 – Pipeline.



Fonte: o autor

Foram executados 10 experimentos, todos utilizaram a mesma arquitetura *BiLSTM-CRF* a partir do *framework FlairNLP*, o que difere entre os experimentos como é possível ver na Tabela 4.2, são as entradas da rede, mais especificamente os Corpora: *MultiWikiNER*, *LeNER-Br* e *PL-Corpus* e o pré-processamento dos mesmos a partir dos Vetores de Palavras: *Pt-Wiki-Fasttext*, *BERTimbau-base* e *FlairEmbeddings-Pt*.

Procurou-se utilizar nos testes, Vetores de Palavras que não haviam sido testados em outros trabalhos nos respectivos Corpora, os experimentos avaliam a influência do uso desses diferentes vetores e suas concatenações, em comparação entre os vetores experimentados e também os resultados apresentados em outros trabalhos com os mesmos corpora e outros vetores.

Tabela 4.2 – Experimentos realizados

Corpus	Vetor de Palavras	Ciclos	Tempo	RAM / VRAM	Tamanho
MultiWikiNER	Vetor Estático <i>Pt-Wiki-Fasttext</i>	100~100	15h	6,5GB / 2GB	700MB
MultiWikiNER	Vetor de Contexto <i>FlairEmbeddings-Pt</i>	65~65	40h	7,5GB / 3GB	250MB
MultiWikiNER	Vetor de Contexto <i>BERTimbau-base</i>	30~30	7h	7GB / 15,5GB	430MB
MultiWikiNER	Vetores Concatenados <i>Pt-Wiki-Fasttext+</i> <i>FlairEmbeddings-Pt</i>	50~50	20h	8,5GB / 3,5GB	940MB
LeNER	Vetor Estático <i>Pt-Wiki-Fasttext</i>	47~100	1h	7GB / 2,5GB	700MB
LeNER	Vetor de Contexto <i>FlairEmbeddings-Pt</i>	60~100	1h 30m	10GB / 7,5GB	250MB
LeNER	Vetores Concatenados <i>Pt-Wiki-Fasttext+</i> <i>FlairEmbeddings-Pt</i>	50~100	1h 30m	12GB / 8,5GB	940MB
PL-Corpus	Vetor Estático <i>Pt-Wiki-Fasttext</i>	93~100	1h	6GB / 2GB	700MB
PL-Corpus	Vetor de Contexto <i>FlairEmbeddings-Pt</i>	64~100	1h	7GB / 4,5GB	250MB
PL-Corpus	Vetores Concatenados <i>Pt-Wiki-Fasttext+</i> <i>FlairEmbeddings-Pt</i>	64~100	1h	8,5GB / 4,5GB	940MB

Fonte: o autor

Também na Tabela 4.2 é possível ver a quantidade de ciclos de treinamento, executados até que o modelo parasse de ganhar desempenho por 3 ciclos consecutivos ou atingisse a limite máximo definido. O tempo aproximado de execução de treinamento para cada modelo, o consumo aproximado de memória e o tamanho aproximado do modelo após treinado. Informações que podem ajudar a ponderar sobre a viabilidade de cada modelo a depender do hardware e do tempo que se pode despendar no treinamento.

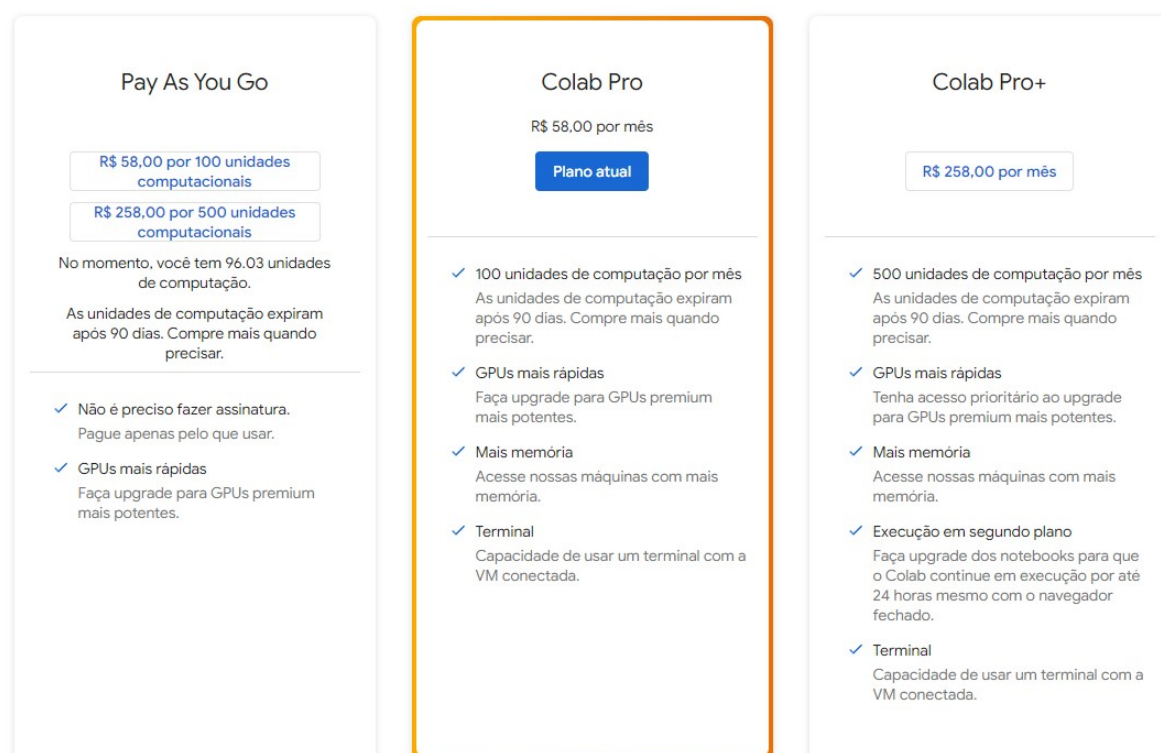
Os experimentos foram executados na nuvem através do serviço "Google Colaboratory" Google (2023a), que possibilita desenvolver código em Python em "notebooks" virtuais através do navegador, acessando o site e logando com uma conta Google. As configurações de *hardware* dos *notebooks* variam de acordo com a disponibilidade e com o tipo de plano utilizado como é possível ver na Figura 4.2, estando disponíveis a versão gratuita e três planos pagos: Sob demanda, Colab Pro e Colab Pro+.

A versão gratuita conta com acesso a processamento em CPU¹ e acesso limitado a processamento em TPUs² e GPUs³ de alto desempenho. Outro limitador da versão gratuita é a necessidade de manter o navegador aberto e interatividade, sendo a seção encerrada em

¹ Central Processing Unit (CPU), é o processador do computador, um processador é capaz de processar modelos, no entanto perde em desempenho para TPUs e GPUs

² Tensor Processing Unit (TPU), são hardwares que fazer o trabalho de processamento como os CPUs e as GPUs, sendo no entanto especializados em arquiteturas que utilizem *Tensor*, nativos em *frameworks* como TensorFlow e PyThorch

³ Graphic Processor Unit (GPU), são basicamente placas de vídeo, esses hardwares podem ter grandes volumes de memória VRAM e tecnologias como CUDA da Nvidia, características que também beneficiam o desempenho no treinamento de modelos em especial de AP

Figura 4.2 – Planos *Google Colaboratory*.

Fonte: Página de planos Google Colaboratory ([Google, 2023b](#))

caso de inatividade por um período que varia de acordo com a disponibilidade de hardware. Em relação a memória RAM o padrão são 12GB, e uma segunda opção de 25GB que pode ficar disponível para a assinatura gratuita a depender da disponibilidade no momento. Nos testes realizados na versão gratuita alguns modelos chegavam a precisar de mais de 24 horas de treinamento exigindo o uso de técnicas de pausa e continuação no treinamento.

O Colab PRO, que custa no momento dos experimentos R\$58,00 mensais, foi a opção utilizada nos experimentos realizados neste trabalho. O diferencial em relação ao gratuito é a preferência no uso de TPUs e GPUs de alto desempenho tornando os experimentos consideravelmente mais rápidos. Atualmente são duas opções de hardware com GPUs de alto desempenho, a V100 com 25GB de RAM e 16GB de VRAM e o sistema com a placa A100 com 84GB de RAM e 40GB de VRAM, suportando modelos com maior número de parâmetros de treinamento.

Essa opção Colab PRO disponibiliza 100 unidades computacionais por mês, essas unidades são consumidas durante o uso dos sistemas com GPUs, no caso dos experimentos realizados neste trabalho, elas foram consumidas a uma média de 5 por hora de uso com a V100 e 13 por hora com a A100. Os tempos de treinamento da Tabela 4.2, são relativos ao uso da A100 que já atendia às necessidade dos experimentos e estava sempre disponível, na V100 o tempo gasto é em média a metade.

Também é possível assinar por R\$258,00 por mês o Colab PRO+, que possibilita ainda mais prioridade nos recursos de hardware e seções que podem continuar por 24 horas em execução, mesmo que o navegador seja fechando. Essa opção disponibiliza 500 unidades computacionais por mês, embora o custo por unidade de processamento seja menor, o valor em reais é consideravelmente mais elevado e não foi necessário no caso dos experimentos realizados neste trabalho. A opção de pagar conforme usa, te da a opção de comprar unidades de computação sem a necessidade de fazer uma assinatura mensal com as mesmas características do Colab Pro e Colab Pro+.

Todos os códigos utilizados nos experimentos estão disponíveis em repositório *Github*⁴, em formato de notebook, pronto para serem analisados e replicados. Assim como o melhor modelo treinado de cada Corpus está disponível em repositório *Huggingface*⁵.

4.3 Arquitetura de Rede

Foram conduzidos experimentos empregando o *Framework Flair*, proposto por [Ak-bik et al. \(2019\)](#). A arquitetura de rede empregada por este *Framework* é a *BiLSTM-CRF*. A entrada da rede é formada pelo Corpus em conjunto com os Vetores de Palavras. A entrada é processada por uma camada de rede avançando e uma retrocedendo. Os resultados dessas duas direções são concatenados, formando um resultado que incorpora informações de ambas. Este resultado é então utilizado como entrada para a próxima camada da rede, que repete o procedimento da camada anterior, em um processo sucessivo através das camadas ocultas. A última camada da rede, que é uma camada *CRF*, realiza a classificação e gera a saída final do sistema. Vale ressaltar que a utilização dessas várias camadas ocultas constitui o aspecto distintivo do aprendizado profundo em comparação com o aprendizado de máquina convencional, conforme ilustrado na Figura 4.3.

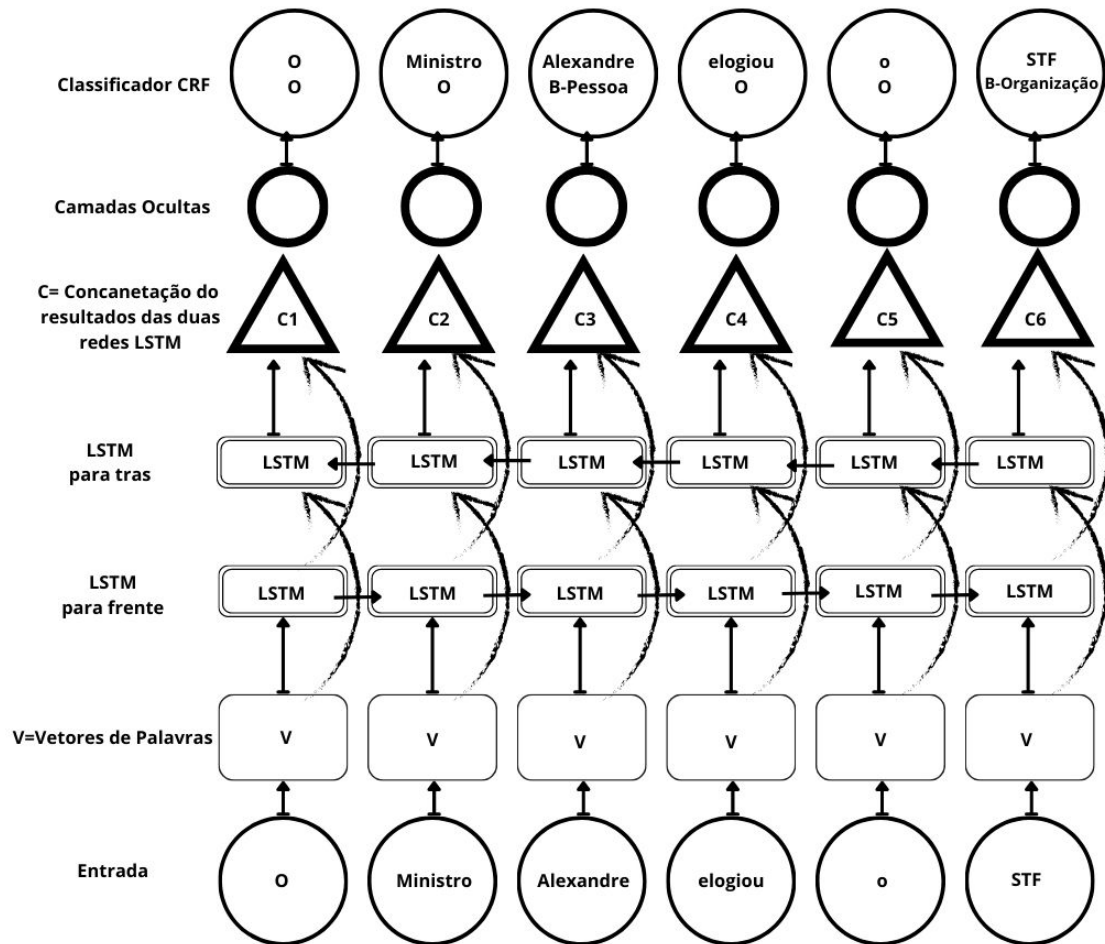
No exemplo ilustrado na Figura 4.3, observamos o uso de duas entidades: Pessoa e Organização, e a notação **BIO**, onde **B** classifica uma entidade de palavra única ou a primeira palavra de uma entidade composta, **I** classifica uma palavra intermediária ou final de uma entidade composta e **O** classifica as não entidades. Dessa forma, o modelo identificou na frase: "O Ministro Alexandre elogiou o STF", a palavra Alexandre como primeira ou única em uma entidade do tipo Pessoa e a palavra STF como primeira ou única em uma entidade do tipo Organização.

A diferença entre os experimentos se deu na entrada, alterando o Vetor de Palavras utilizado e concatenando os vetores. Na Figura 4.4 é possível ver em mais detalhes como é o pré-processamento da entrada com o uso de vetores estáticos, contextuais e a concatenação deles. Quando utilizando Vetores Estáticos, tanto os vetores pré-treinados e o corpus

⁴ <<https://github.com/Darkm4ge/Avaliacao-da-Influencia-do-uso-de-Embeddings-no-Reconhecimento-de-Entidades-Nomeadas-para-o-Portugue>>

⁵ <<https://huggingface.co/CarlosGomes>>

Figura 4.3 – BiLSTM-CRF.

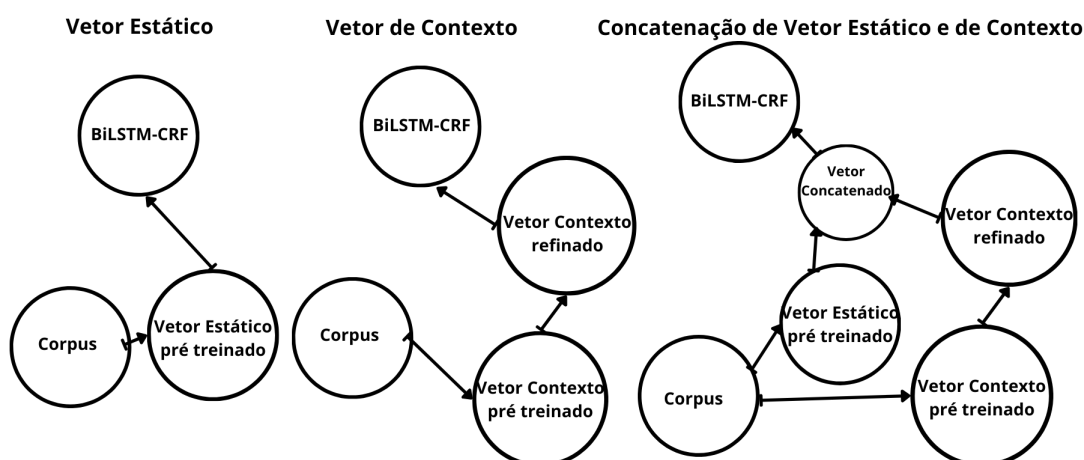


Fonte: o autor

são passados como entrada da rede. No caso dos Vetores de Contexto, os vetores pré treinados, passam por um processo de refinamento com o corpus da tarefa, resultando em um vetor de contexto afinado para a tarefa com o corpus de treinamento. Este vetor resultante juntamente ao corpus então é fornecido como entrada da rede. Por último no exemplo da concatenação de vetores de contexto e estático, o vetor de contexto afinado é concatenado com o vetor estático pré treinado e juntamente com o corpus são utilizados como a entrada da rede.

Na Figura 4.5, é possível ver o Log de treinamento de um modelo. Dentro do retângulo 1 marcado em vermelho, o trecho "*StackedEmbeddings*" indica que existem vetores concatenados e logo em seguida é possível ver o Vetor Estático "*(list_embedding_0): WordEmbed-*

Figura 4.4 – Entrada da rede com os diferentes Vetores de Palavras.



Fonte: o autor

dings('pt') e os dois Vetores de Contexto sendo para frente e para trás, *"(list_embedding_1): FlairEmbeddings-Pt"* que são então concatenados em um resultante. No retângulo 2 marcado em vermelho, encontra-se a rede bidirecional resultante, em ambos os retângulos marcados em vermelho as informações sobre as dimensões dos Vetores de Palavras e da rede que é apresentada em mais detalhes na abordagem dos diferentes Vetores, pois esses interferem diretamente nesses dados.

As outras características que estão fora dos retângulos são comuns a todos os modelos dos experimentos realizados. O termo *"WordDropout"* é uma camada que remove aleatoriamente palavras da entrada com uma probabilidade $p=0.05$. Para Akbik *et al.* (2019) essa técnica de regularização ajuda a evitar *overfitting* (sobreajuste no corpus de treinamento), e melhora a generalização da rede. Já *"LockedDropout"* é uma camada que aleatoriamente “desliga” (zera) certos neurônios da rede durante o treinamento. Isso ajuda a evitar o *overfitting* e aumenta a capacidade da rede de lidar com diferentes entradas, melhorando a sua robustez.

Por fim ainda na Figura 4.5, abaixo do retângulo 2, o termo *"ViterbiLoss"* que é uma função de perda que utiliza o algoritmo de *"Viterbi"* para calcular a melhor sequência de rótulos para uma sequência de palavras. E por fim, a camada de saída *CRF*: é uma camada que utiliza o algoritmo de decodificação *CRF* para realizar a marcação de sequências. Essa camada considera as probabilidades das transições entre os rótulos adjacentes e leva em conta o contexto global das sequências para gerar as previsões de marcação mais precisas.

Na Figura 4.6, é possível ver novamente o Log de Treinamento. O trecho apresenta os

Figura 4.5 – Log de treinamento de um modelo com Vetores Estáticos e de Contexto concatenados.

```

2022-09-22 02:59:26,806 Model: "SequenceTagger(
  (embeddings): StackedEmbeddings(
    (list_embedding_0): WordEmbeddings(
      'pt'
      (embedding): Embedding(592108, 300)
    )
    (list_embedding_1): FlairEmbeddings(
      (lm): LanguageModel(
        (drop): Dropout(p=0.5, inplace=False)
        (encoder): Embedding(275, 100)
        (rnn): LSTM(100, 2048)
        (decoder): Linear(in_features=2048, out_features=275, bias=True)
      )
    )
    (list_embedding_2): FlairEmbeddings(
      (lm): LanguageModel(
        (drop): Dropout(p=0.5, inplace=False)
        (encoder): Embedding(275, 100)
        (rnn): LSTM(100, 2048)
        (decoder): Linear(in_features=2048, out_features=275, bias=True)
      )
    )
  )
  (word_dropout): WordDropout(p=0.05)
  (locked_dropout): LockedDropout(p=0.5)
  (embedding2nn): Linear(in_features=4396, out_features=4396, bias=True)
  (rnn): LSTM(4396, 256, batch_first=True, bidirectional=True)
  (linear): Linear(in features=512, out features=31, bias=True)
  (loss_function): ViterbiLoss()
  (crf): CRF()
)"

```

Fonte: o autor

parâmetros utilizados no treinamento dos modelos dos experimentos. Esses são os parâmetros padrão do *Framework Flair*, sugeridos por [Akbik et al. \(2019\)](#). Além disso, vale destacar que o intuito do trabalho é analisar a influência dos diferentes Vetores de Palavras e concatenação dos mesmos nos diferentes Corpora. Portanto é importante que os testes com os diferentes vetores em cada corpus utilizem os mesmos parâmetros, com exceção do número de ciclos necessários para o treinamento que tiveram variação.

Na Figura 4.6 é possível ver os parâmetros utilizados nos experimentos e abaixo é possível ver uma lista que descreve estes parâmetros de treinamento sugeridos por ([AKBIK et al., 2019](#)) e definidos como padrão no *Framework Flair*:

- *learning_rate*: é a taxa de aprendizado da rede. Define o tamanho do passo que o algoritmo de otimização dá em cada iteração para atualizar os pesos da rede. Um valor

Figura 4.6 – Log de treinamento, identificação dos parâmetros da rede.

```
Parameters:
- learning_rate: "0.100000"
- mini_batch_size: "32"
- patience: "3"
- anneal_factor: "0.5"
- max_epochs: "70"
- shuffle: "True"
- train_with_dev: "False"
- batch_growth_annealing: "False"
```

Fonte: o autor

maior pode levar a uma convergência mais rápida, mas também pode fazer com que a rede pule sobre mínimos/máximos locais e não alcance o mínimo/máximo global. Um valor menor pode fazer com que a rede demore mais para convergir, mas pode aumentar a precisão do modelo.

- *mini_batch_size*: é o tamanho do *mini-batch* usado durante o treinamento. O *mini-batch* é um subconjunto dos dados de treinamento que é usado para atualizar os pesos da rede em cada iteração. Um tamanho de *mini-batch* maior pode levar a uma convergência mais rápida, mas também pode exigir mais memória e pode ser menos eficiente em termos de tempo de computação.
- *patience*: é o número de épocas que a rede espera antes de parar o treinamento se não houver melhoria na função de perda. Esse parâmetro é usado em conjunto com o *anneal_factor* para ajustar o *learning_rate* e tentar escapar de mínimos locais.
- *anneal_factor*: é o fator pelo qual o *learning_rate* é reduzido quando o *patience* é atingido. Um valor de 0,5 significa que o *learning_rate* é reduzido pela metade. O *patience* com o *anneal_factor* podem parar o treinamento antes do *max_epochs*, quando reduzem demais o *learning_rate*.
- *max_epochs*: é o número máximo de épocas (ciclos ou passagens completas pelo conjunto de treinamento), permitido antes que o treinamento pare.
- *shuffle*: é um parâmetro booleano que indica se os dados de treinamento são embaralhados antes de cada época. Embaralhar os dados pode ajudar a evitar que a rede memorize a ordem dos dados. "True"liga o embaralhamento e "False"desliga.
- *train_with_dev*: é um parâmetro booleano que indica se os dados de validação (conjunto de validação) devem ser usados durante o treinamento. "True"permite o uso e "False"não permite.
- *batch_growth_annealing*: é um parâmetro booleano que indica se o tamanho do *mini-batch* deve ser aumentado à medida que o treinamento avança. Esse recurso pode

ajudar a encontrar melhores mínimos locais e melhorar a eficiência do treinamento. "True"liga o recurso e "False"desliga.

Em relação a avaliação, o *framework* utiliza na classificação de múltiplas classes, os valores de medida F1 das diferentes classes neste caso tipos de entidades que passam por uma macro-média, resultando em uma métrica que combina igualmente as contribuições de todas as entidades, independentemente de suas proporções individuais. Conforme a Equação 4.1, onde pk e rk são precisão e recuperação para classes, e n é o número total de classes. Conceitualmente, a medida F1 nada mais é do que a média harmônica de precisão e recuperação entre as classes (BERGER; GUDA, 2020).

$$macro - F = \frac{1}{n} \sum_{k=1}^n \frac{2pkrk}{pk + rk} \quad (4.1)$$

4.4 Corpus

Para os experimentos desenvolvidos nesse trabalho foram utilizados para treinamento dos modelos o Corpus de domínio geral MutiWikiNER e os Corpora de domínio legal LeNER-Br e PL-Corpus, os corpora foram utilizados em formato .txt sem nenhuma alteração. A seguir são apresentadas as principais características desses corpora como: tipo de anotação, tipos de textos anotados, entidades definidas, volume de dados anotados, entre outras características.

4.4.1 Corpus MutiWikiNER

O Corpus *MultiWikiNER* desenvolvido por Nothman *et al.* (2013), é um corpus multilíngue de domínio geral, livre e gratuito, que conta com nove línguas diferentes sendo elas: Inglês, Alemão, Francês, Polonês, Italiano, Espanhol, Holandês, Português e Russo. Os autores apresentam uma abordagem para a criação automática de anotações para treinamento de modelos de REN, usando a estrutura e conteúdo da Wikipédia. As entidades nomeadas existentes no Corpus *MultiWikiNER* são: Pessoas (PER), Localização (LOC), Outros (MISC) e Organização (ORG).

A criação de anotações para REN é um processo caro e demorado que requer a anotação manual de grandes quantidades de dados. O trabalho descrito pelos autores, propõe uma abordagem mais eficiente, que usa a estrutura e o conteúdo da Wikipédia para criar anotações de treinamento em larga escala, multilíngue e de qualidade aceitável, a partir de uma abordagem inicial que classifica automaticamente cada artigo da Wikipédia em tipos de entidades nomeadas.

A técnica consiste em executar a classificação automática de um grande volume de dados, sendo o maior volume para o inglês chegando a 3,4 milhões de artigos, e pelo menos

100 milhões de sentenças por língua, utilizando a estrutura de hiperlinks da Wikipédia. Em seguida essa anotação foi avaliada utilizando um conjunto de 7.200 artigos da Wikipédia anotados manualmente.

Para a língua portuguesa foram utilizados os dados que estavam disponíveis na Wikipédia em português, na data de 04/08/2010, contendo 598.446 artigos e 120.137.554 "*tokens*", classificados automaticamente e então avaliados em um conjunto de dados anotado de forma manual. A abordagem proposta alcança uma Precisão de até 95% nos nove idiomas diferentes, o que é uma excelente conquista, considerando a dificuldade de se criar anotações precisas e abrangentes em múltiplos idiomas. Embora as anotações geradas automaticamente não pudessem competir diretamente com as anotações manuais em termos de qualidade, com o uso de heurísticas foi possível alinhá-las mais precisamente com as anotações padrão ouro.

Os autores apresentam uma abordagem inovadora e promissora para o treinamento de modelos de REN que pode ajudar a superar a escassez de dados de treinamento anotados manualmente e melhorar a eficiência do processo de desenvolvimento de sistemas de processamento de linguagem natural, em especial em casos com o da língua portuguesa que passa por essa escassez. No experimento que executamos com o corpus *MultiWikiNER*, ele continha 142.153 sentenças, divididas em 3 conjuntos, treinamento, teste e validação, conforme é possível ver na Tabela 4.3

Tabela 4.3 – Número de Sentenças no Treinamento, Teste e Validação no Corpus *MultiWikiNER*

Conjunto	Sentenças
Treinamento	115.144
Teste	14.215
Validação	12.794

Fonte: o autor

Já na Tabela 4.4, é possível ver o número de *tokens* anotados no corpus utilizado nos experimentos, somando 247.246 *tokens*, divididos por tipo de entidade entre as entidades: PER, LOC, MISC e ORG. É possível notar que a diferença em número de *tokens* por entidade é relativamente baixa como veremos ao analisar os demais corpora, isso possibilita ao modelo treinado com esse corpus, ter desempenho similar na classificação das diferentes entidades, o que foi confirmado nos experimentos.

Tabela 4.4 – Número de *tokens* por entidade nos experimentos realizados

Entidades	Tokens
PER	77.978
LOC	69.479
MISC	59.575
ORG	40.214

Fonte: o autor

4.4.2 Corpus LeNER-Br

No trabalho de [Araujo et al. \(2018\)](#) é apresentado o Corpus LeNER-Br, anotado de forma manual para a tarefa REN no domínio jurídico Brasileiro, com intuito de extrair informações de documentos legais para auxiliar na tomada de decisão. O corpus é composto totalmente de documentos legais e trabalha com as entidades: Pessoa, Data, Local, Organização, Legislação e Jurisprudência, sendo as últimas duas específicas do domínio legal, e portanto o diferencial do corpus em relação aos de domínio geral como HAREM. Os próprios autores treinaram modelos REN com o corpus e obtiveram resultados F1 de 97.04% e 88.82% para Legislação e Jurisprudência, respectivamente, conforme já foi discutido em mais detalhes nos trabalhos relacionados.

Segundo os autores o Corpus LeNER-Br foi o primeiro conjunto de dados de referência no domínio legal em português para a tarefa REN. O processo de anotação foi manual o que garante melhores resultados na anotação. Foram usados 66 documentos legais de diferentes Cortes pelo Brasil e quatro legislações como a Lei Maria da Penha, compondo um total de 70 documentos.

Inicialmente os documentos foram divididos em sentenças e *tokenizados* com o *Natural Language Toolkit* (NLTK) ([LOPER; BIRD, 2002](#)), ou Kit de Ferramentas de Linguagem Natural, a partir de sua biblioteca no Python. Depois de *tokenizados* cada documento foi manualmente anotado com auxílio da ferramenta WebAnno utilizando o esquema de anotação *In Out Begin* (IOB), onde *begin* ou início classifica a primeira palavra de uma entidade composta, e *in* ou dentro classifica as demais palavras de uma entidade composta, por fim *out* ou fora, classifica as palavras que não são entidades.

Para os experimentos o corpus é dividido em três conjuntos, sendo eles: Treinamento, Teste e Validação. Na Tabela 4.5 apresenta-se a quantidade de documentos, sentenças e *tokens* para cada conjunto.

Tabela 4.5 – Número de Sentenças no Treinamento, Teste e Validação no Corpus LeNER-Br

Conjunto	Documentos	Sentenças	Tokens
Treinamento	50	7.827	229.277
Teste	10	1.389	47.630
Validação	10	1.176	41.166

Fonte: o autor

Já na Tabela 4.6, é possível ver o número de palavras anotadas em entidades em cada conjunto do corpus utilizado nos experimentos, somando 44.513 palavras no total, sendo 32.049 palavras no conjunto de Treinamento, 5.823 no conjunto de Teste e finalmente 6.641 palavras no conjunto de Validação. Na tabela é possível ver esse número de palavras divididas por tipo de entidade entre as entidades: PESSOA, LOCAL, ORGANIZACAO, TEMPO, LEGISLACAO e JURISPRUDENCIA. É possível notar que a diferença em número de palavras por entidade é considerável, isso implica em uma diferença no desempenho dessas enti-

dades, mostrando uma relação entre o aumento no número de palavras por entidade e o aumento no desempenho, confirmados nos resultados apresentados pelos autores e nos experimentos deste trabalho.

Tabela 4.6 – Número de palavras em entidade por conjunto no Corpus LeNER-Br

Entidades	Treinamento	Teste	Validação
PESSOA	4.612	735	894
LOCAL	1.417	132	244
ORGANIZACAO	6.671	1.367	1.608
TEMPO	2.343	260	543
LEGISLACAO	13.039	2.669	2.609
JURISPRUDENCIA	3.967	660	743

Fonte: o autor

4.4.3 Corpus Ulysses_br

O projeto Ulysses visa aumentar a transparência e apoiar a atividade legislativa com modelos de linguagem capazes de analisar documentos legais e extrair informações relevantes. O artigo de [Albuquerque et al. \(2022\)](#), descreve o processo por detrás da anotação do conjunto de dados e suas características, seguido de uma avaliação do UlyssesNER-Br. Foram criados dois corpora a partir de dois tipos de documentos legislativos, os projetos de lei (PL) que são documentos públicos disponíveis no site da Câmara, resultando no PL-Corpus, e as solicitações de trabalho, documentos sigilosos que contém pedidos de Deputados sobre a situação de determinado tema nas tramitações da Câmara, esses por sua vez dando origem ao ST-Corpus, que em virtude dos dados utilizados no treinamento, não está disponível de forma livre.

Outra característica dos corpora do projeto Ulysses está relacionada às entidades. A priori os corpora utilizam sete entidades sendo elas: FUNDAMENTO, ORGANIZACAO, PRODUTOLEI, LOCAL, PESSOA, DATA e EVENTO. No entanto também é possível utilizar os corpora com uma subdivisão das entidades, que resultou em dezoito entidades sendo elas: FUNDlei, ORGgovernamental, PRODUTOoutros, LOCALvirtual, FUNDapelido, PESSOAcargo, LOCALconcreto, PESSOAindividual, FUNDprojetoilei, DATA, ORGnaogovernamental, FUNDsolicitacaotrabalho, PESSOAgrupocargo, PRODUTOprograma, PRODUTOsisistema, EVENTO, ORGpartido, PESSOAgrupointd. Segundo a avaliação dos autores não houve diferença de desempenho entre as duas possibilidades de divisão de entidades.

Nos experimentos foi utilizado o PL-Corpus em sua configuração de sete entidades. Foram utilizados na anotação desse corpus, a ferramenta INCEption e 150 projetos de lei. Por fim o corpus é constituído de 9.526 sentenças com 138.741 *tokens*, dos quais 11.833 são únicos. Os autores também apresentaram a frequência de cada entidade no corpus sendo a entidade mais comum FUNDAMENTO, com quase 7.000 aparições, as demais com aproximadamente 2.000, e EVENTO com aproximadamente 200, mostrando um desbalançamento

em número de aparições e consequentemente no desempenho, em especial na entidade EVENTO com número muito baixo de aparições e baixo desempenho. Na Tabela 4.7, é possível ver os números de sentenças por conjunto e constatar que são muito similares aos do corpus LeNER-Br, outra característica em que esses dois corpora se assemelham.

Tabela 4.7 – Número de Sentenças no Treinamento, Teste e Validação no Corpus PL-Corpus

Conjunto	Sentenças
Treinamento	6.667
Teste	1.430
Validação	1.429

Fonte: o autor

Os experimentos realizados neste trabalho, com esse corpus, foram comparados entre si e com os resultados apresentados pelos desenvolvedores do corpus, e trazem ganho de desempenho em relação aos resultados originais apresentados. Esses ganhos serão detalhados nos resultados dos experimentos.

4.5 Vetores de Palavras

Nesta seção são apresentados os Vetores de Palavras utilizados nos experimentos, sendo eles: *Pt-Wiki-FastText*, *FlairEmbeddings-Pt* e *BERTimbau*. O *Framework* oferece fácil implementação dos vetores de palavras, que são carregados direto de seus repositórios no *Hungginface*.

Procurou-se utilizar nos testes, Vetores de Palavras que não haviam sido testados em outros trabalhos nos respectivos Corpora, os experimentos avaliam a influência do uso desses diferentes vetores e suas concatenações, em comparação entre os vetores experimentados e também os resultados apresentados em outros trabalhos com os mesmos corpora e outros vetores.

4.5.1 Pt-Wiki-Fasttext

O *Pt-Wiki-Fasttext* é um tipo de vetor de palavras estático, que é usado para a representação de palavras em modelos de processamento de linguagem natural. Esse vetor foi pré-treinado usando o método Skip-gram, descrito em [Bojanowski et al. \(2017\)](#), e utiliza os dados e estrutura de links disponíveis na Wikipédia em português. Ele possui 300 dimensões e 1 milhão de palavras. O *Pt-wiki-fasttext* é apenas uma das várias opções de vetores de palavras pré-treinados disponíveis no site do *FastText*. Existem vetores pré-treinados para 294 idiomas diferentes, todos baseados nos dados da Wikipédia.

Nos experimentos realizados com este vetor, quando utilizado sozinho, tem-se desempenho inferior aos modelos treinados com os vetores de contexto, no entanto em conjunto com vetores de contexto foi percebida uma pequena melhora, como veremos na seção

de resultados. Outro ponto que vale ressaltar em relação aos experimentos com este vetor, é que o treinamento é consideravelmente mais rápido que utilizando os demais vetores testados, em virtude de seu tamanho reduzido em comparação aos demais, de forma que os modelos treinados utilizando este vetor, foram treinados em menos da metade do tempo dos demais experimentos que executamos neste trabalho.

4.5.2 *FlairEmbeddings*

Ao contrário dos vetores de palavras estáticos, que representam cada palavra com um vetor fixo, os vetores de contexto como o *FlairEmbeddings* (AKBIK; BLYTHE; VOLLGRAF, 2018) levam em consideração o contexto em que cada palavra aparece. Isso significa que cada palavra é representada por um vetor único que depende das palavras ao redor da frase.

Para treinar os Vetores de Contexto *FlairEmbeddings*, a estrutura *Flair* (AKBIK *et al.*, 2019) usa uma rede neural do tipo *BiLSTM-LM*. A Rede processa o corpus a nível de caractere, de forma a permitir levar em consideração a ordem das palavras e o contexto em que cada palavra aparece. A saída da rede é uma sequência de vetores, um para cada letra na sentença. Esses vetores são então combinados para criar uma única representação vetorial de toda a sentença. Os vetores são pré-treinados utilizando um grande volume de dados não rotulados, no momento do treinamento do modelo para a tarefa esperada, no caso dos experimentos REN realizados neste trabalho, os vetores pré-treinados passam por um processo de *finetuning*, ou ajuste fino no próprio corpus rotulado de treinamento do modelo.

Vetores de contexto como o *FlairEmbeddings* ou *BERTimbau*, são muito eficazes em uma ampla gama de tarefas de PLN, incluindo classificação de texto, REN e análise de sentimento. Como eles levam em consideração o contexto em que cada palavra aparece, eles são particularmente úteis para estas tarefas em que o contexto é importante. Nos experimentos realizados na tarefa REN utilizando esse vetor de contexto, foi observada melhora expressiva no desempenho em relação aos modelos treinados com o vetor estático, podendo ainda ser combinado ao vetor estático o que pode acrescentar uma pequena melhora.

4.5.3 *BERTimbau*

O *BERTimbau* desenvolvido por Souza, Nogueira e Lotufo (2020) é um modelo *BERT* pré-treinado que foi ajustado para o português brasileiro. Esse modelo alcança um desempenho de ponta em tarefas de recuperação de informações, análise de sentimentos e reconhecimento de entidades nomeadas. O *BERTimbau* é uma visão geral de alto nível da arquitetura do *BERT* e descreve os procedimentos para pré-treinar modelos *BERT* para o português brasileiro. Os autores apresentam várias abordagens para ajustar o *BERT* para a classificação de texto, incluindo o uso de transferência de aprendizado e a adição de características específicas de domínio. Eles também discutem a importância do pré-processamento

dos dados e a necessidade de avaliar cuidadosamente o desempenho de qualquer modelo utilizado em um contexto específico.

Grandes modelos de linguagem pré-treinados (LMs) são inestimáveis, particularmente para idiomas com escassos recursos anotados, mas abundantes dados não rotulados, como o português. Considerando esse cenário os autores treinaram modelos *BERT* especificamente para o Português Brasileiro, denominados "*BERTimbau*", utilizando dados do extenso e diversificado corpus de páginas da web o *brWaC*. A avaliação contemplou três tarefas de PNL: similaridade textual de sentença, reconhecimento de implicação textual e REN. Também, foi comparado o desempenho dos *tokenizadores BERTimbau* e *mBERT* nestas tarefas, explorando suas implicações. Os resultados mostram a superioridade do *BERTimbau* sobre modelos multilíngues e métodos monolíngues anteriores, reforçando ainda mais a eficácia de grandes LMs pré-treinados para o português.

Além disso, a abordagem descrita no *BERTimbau* para pré-treinar modelos *BERT* para o português brasileiro pode ser aplicada em outras áreas e domínios, permitindo o desenvolvimento de sistemas de PLN mais precisos e eficientes. O *BERTimbau* assim como o *BERT* e outros modelos de linguagem pode ser utilizados como vetor de contexto. O *BERTimbau* está disponível no *github* em duas versões, a versão base com 12 camadas 110 milhões de parâmetros, utilizada neste trabalho. E a versão grande com 24 camadas e 335 milhões de parâmetros que não pode ser utilizada pois excedia o limite do memória RAM do *notebook* no *Google Colaboratory*.

Capítulo 5

RESULTADOS

Neste capítulo são apresentados e discutidos os resultados obtidos a partir dos experimentos executados. Foram treinados dez modelos para REN, sendo quatro modelos treinados e avaliados com o Corpus multilíngue incluindo português *MultiWikiNER* de domínio geral, três com o LeNER-Br e três com o PL-Corpus, ambos Corpora de domínio legal para o português. Foram executados três testes em cada modelo, na Tabela 5.1 na coluna Média F1, a média dos três resultados e o desvio padrão.

A Tabela 5.1, apresenta os resultados obtidos nos 4 modelos treinados no Corpus *MultiWikiNER* de domínio geral, o modelo que utiliza o Vetor Estático *Pt-Wiki-Fasttext* obteve o pior resultado com média F1 de 79,325%, seguido do Vetor de Contexto *FlairEmbeddings-Pt* com 86,82% demonstrando um ganho de performance no uso do *FlairEmbeddings-Pt*, a concatenação dos dois Vetores de Palavras, também apresentou uma pequena melhora de 2%, chegando aos 88,83%.

Tabela 5.1 – Resultados com o Corpus *MultiWikiNER*.

Vetor de Palavras	Média F1 (macro)	Melhor Resultado	Resultado Médio	Resultado Mínimo
Vetor Estático <i>Pt-Wiki-Fasttext</i>	79,325% $\pm$ 0,205%	F1 (micro) 81,24%	F1 (micro) 81,19%	F1 (micro) 80,88%
		F1 (macro) 79,53%	F1 (macro) 79,47%	F1 (macro) 79,12%
		Precisão 71,63%	Precisão 71,4%	Precisão 71,09%
Vetor de Contexto <i>FlairEmbeddings-Pt</i>	86,82% $\pm$ 0,86%	F1 (micro) 88,13%	F1 (micro) 88,09%	F1 (micro) 87,4%
		F1 (macro) 87,25%	F1 (macro) 87,25%	F1 (macro) 86,39%
		Precisão 84,07%	Precisão 84,13%	Precisão 83,36%
Vetor de Contexto BERTimbau-base	96,945% $\pm$ 1,935%	F1 (micro) 98,96%	F1 (micro) 96,55%	F1 (micro) 95,4%
		F1 (macro) 98,88%	F1 (macro) 96,3%	F1 (macro) 95,01%
		Precisão 98,5%	Precisão 95,05%	Precisão 93,41%
Vetores Concatenados <i>Pt-Wiki-Fasttext+</i> <i>FlairEmbeddings-Pt</i>	88,83% $\pm$ 0,34%	F1 (micro) 89,82%	F1 (micro) 89,65%	F1 (micro) 89,45%
		F1 (macro) 89,17%	F1 (macro) 88,81%	F1 (macro) 88,49%
		Precisão 85,72%	Precisão 85,39%	Precisão 85,38%

Fonte: o autor

O melhor resultado atingido no Corpus *MultiWikiNER* foi com o Vetor de Contexto *BERTimbau-base*, atingindo média F1 de 96,945% apresentando uma melhora considerável em relação a concatenação de vetores. Na Tabela 5.2, são apresentados os resultados desse modelo em detalhes por tipo de entidade. O Corpus *MultiWikiNER*, como é possível ob-

servar na coluna *Apoio*, apresenta uma distribuição no número de entidades mais uniforme em comparação aos demais corpora, o que justifica um aprendizado também mais uniforme entre as entidades.

Tabela 5.2 – Detalhamento do melhor resultado com o Corpus *MultiWikiNER*.

Arquitetura e Vetor de Palavras	Entidade	Precisão	Recuperação	F1	Apoio
<i>BiLSTM-CRF</i>	PER	99,66%	99,43%	99,55%	9722
	LOC	98,79%	98,90%	98,84%	8646
<i>BERTimbau-base</i>	MISC	98,70%	98,64%	98,67%	7444
	ORG	98,45%	98,43%	98,44%	4983

Fonte: o autor

Os resultados dos três modelos treinados com o Corpus de domínio legal LeNER-Br, estão na Tabela 5.3, como é possível perceber, eles seguem a mesma tendência dos resultados do Corpus anterior, com uma melhora considerável ao utilizar os Vetores de Contexto e uma pequena melhora no modelo resultante da concatenação dos vetores, que atingiu o melhor resultado com uma média F1 de 89,02%, no entanto, inferior aos 92,53% atingidos no trabalho dos criadores do Corpus, (ARAÚJO *et al.*, 2018), com a mesma arquitetura *BiLSTM-CRF*, mas com o Vetor Estático *Glove*.

Tabela 5.3 – Resultados com o Corpus LeNER-Br.

Vetor de Palavras	Média F1 (macro)	Melhor Resultado	Resultado Médio	Resultado Mínimo
Vetor Estático <i>Pt-Wiki-Fasttext</i>	82,88% $\pm$ 0,695%	F1 (micro) 85,21%	F1 (micro) 84,25%	F1 (micro) 83,83%
		F1 (macro) 83,58%	F1 (macro) 82,81%	F1 (macro) 82,19%
		Precisão 74,94%	Precisão 74,17%	Precisão 72,8%
Vetor de Contexto <i>FlairEmbeddings-Pt</i>	88,64% $\pm$ 0,625%	F1 (micro) 90,4%	F1 (micro) 89,87%	F1 (micro) 89,48%
		F1 (macro) 89,27%	F1 (macro) 88,42%	F1 (macro) 88,02%
		Precisão 83,32%	Precisão 82,29%	Precisão 81,87%
Vetores Concatenados		F1 (micro) 90,39%	F1 (micro) 90,79%	F1 (micro) 90,55%
<i>Pt-Wiki-Fasttext+</i>	89,02% $\pm$ 0,29%	F1 (macro) 89,31%	F1 (macro) 89,19%	F1 (macro) 88,73%
<i>FlairEmbeddings-Pt</i>		Precisão 83,16%	Precisão 83,88%	Precisão 83,41%

Fonte: o autor

Na Tabela 5.4, o melhor modelo com o corpus LeNER-Br é apresentado em detalhes por tipo de entidade, ao lado dos resultados por tipo de entidade apresentados por Araujo *et al.* (2018), aqui é possível observar que comparando os resultados, os modelos tem desempenho muito semelhante. Outro ponto que chama atenção é que a discrepância na quantidade de entidades anotadas influencia na uniformidade do desempenho entre as entidades, e o baixo número de exemplos anotados como no caso da entidade "LOCAL" prejudica o aprendizado.

Por fim, encontram-se na Tabela 5.5, os resultados dos outros três modelos de domínio legal, treinados a partir do PL-Corpus, assim como observado nos demais testes, apresentam melhora nos resultados, quando comparamos o Vetor Estático *Pt-Wiki-Fasttext* ao Vetor de Contexto *FlairEmbeddings-Pt* e um resultado ainda melhor quando concatenamos os dois vetores.

Tabela 5.4 – Detalhamento do melhor resultado obtido com o Corpus LeNER-Br, em comparação com o trabalho que propôs o Corpus

Arquitetura e Vetor de Palavras	Entidade	Precisão	Recuperação	F1	Apoio
<i>BiLSTM-CRF</i> <i>Pt-Wiki-Fasttext+</i> <i>FlairEmbeddings-Pt</i>	PESSOA	90,98%	95,28%	93,08%	233
	LOCAL	73,08%	80,85%	76,77%	47
	ORGANIZACAO	83,46%	85,63%	84,85%	501
	TEMPO	95,60%	90,62%	93,05%	192
	LEGISLACAO	96,28%	95,77%	96,02%	378
<i>BiLSTM-CRF</i> <i>Glove</i> (<i>ARAUJO et al., 2018</i>)	JURISPRUDENCIA	89,39%	95,68%	92,43%	185
	PESSOA	94,44%	92,52%	93,47%	233
	LOCAL	61,24%	59,85%	60,54%	47
	ORGANIZACAO	91,27%	85,66%	88,38%	501
	TEMPO	91,15%	91,15%	91,15%	192
	LEGISLACAO	97,08%	97,00%	97,04%	378
	JURISPRUDENCIA	87,39%	90,30%	88,82%	185

Fonte: o autor

Tabela 5.5 – Resultados obtidos com o PL-Corpus.

Embeddings	Média F1 (macro)	Melhor Resultado	Resultado Médio	Resultado Mínimo
Vetor Estático <i>Pt-Wiki-Fasttext</i>	81,45% ± 1,19%	F1 (micro) 85,64%	F1 (micro) 85,69%	F1 (micro) 85,09%
		F1 (macro) 82,64%	F1 (macro) 81,71%	F1 (macro) 80,26%
		Precisão 75,45%	Precisão 75,41%	Precisão 74,37%
Vetor de Contexto <i>FlairEmbeddings-Pt</i>	84,635% ± 0,865%	F1 (micro) 89,46%	F1 (micro) 88,01%	F1 (micro) 89,42%
		F1 (macro) 85,6%	F1 (macro) 83,98%	F1 (macro) 83,87%
		Precisão 81,93%	Precisão 80,0%	Precisão 82,22%
Vetores Concatenados <i>Pt-Wiki-Fasttext+</i> <i>FlairEmbeddings-Pt</i>	86,01% ± 0,85%	F1 (micro) 89,67%	F1 (micro) 88,89%	F1 (micro) 89,33%
		F1 (macro) 86,86%	F1 (macro) 86,79%	F1 (macro) 85,51%
		Precisão 82,01%	Precisão 80,59%	Precisão 81,46%

Fonte: o autor

Na Tabela 5.6, vê-se o detalhamento por tipo de entidade do melhor resultado obtido com o PL-Corpus utilizando a concatenação do *Pt-Wiki-Fasttext* com o *FlairEmbeddings-Pt*, vê-se também abaixo na tabela, o detalhamento dos resultados apresentados por [Albuquerque et al. \(2022\)](#), criadores do corpus.

Tabela 5.6 – Detalhamento do melhor resultado obtido com o PL-Corpus, em comparação com o trabalho que propôs o Corpus.

Arquitetura e Vetor de Palavras	Entidade	Precisão	Recuperação	F1	Apoio
<i>BiLSTM-CRF</i> <i>Pt-Wiki-Fasttext+</i> <i>FlairEmbeddings-Pt</i>	FUNDAMENTO	94,21%	91,94%	93,06%	124
	PESSOA	94,92%	94,12%	94,51%	119
	LOCAL	81,13%	85,15%	83,09%	101
	DATA	96,00%	97,96%	96,97%	98
	ORGANIZACAO	83,67%	87,23%	85,42%	94
	PRODUTOLEI	82,35%	77,78%	80,00%	54
	EVENTO	85,71%	66,67%	75,00%	9
<i>CRF</i> (<i>ALBUQUERQUE et al., 2022</i>)	FUNDAMENTO	67,61%	75,39%	71,29%	124
	PESSOA	82,61%	51,12%	63,16%	119
	LOCAL	72,50%	71,08%	71,78%	101
	DATA	88,61%	68,29%	77,13%	98
	ORGANIZACAO	66,25%	51,21%	57,77%	94
	PRODUTOLEI	74,14%	31,85%	44,56%	54
	EVENTO	0,00%	0,00%	0,00%	9

Fonte: o autor

Os resultados demonstram que a arquitetura *BiLSTM-CRF* no geral apresenta bom desempenho, que a uniformidade no quantitativo de entidades anotadas entre as diferentes entidades auxilia na qualidade do treinamento e um baixo número de dados anotados atrapalha no desempenho. Os resultados também demonstram a influência do uso dos vetores de palavras e as diferenças de desempenho entre eles.

Capítulo 6

CONCLUSÕES E TRABALHOS FUTUROS

O principal objetivo deste estudo foi investigar a aplicação sinérgica do Aprendizado Profundo (AP) em conjunto com distintos Vetores de Palavras, bem como suas combinações, visando alcançar um novo patamar de excelência na tarefa de Reconhecimento de Entidades Nomeadas (REN) para o idioma português, considerando diferentes conjuntos de dados.

Como uma valiosa contribuição adicional, além da compilação de uma série de experimentos, códigos e os melhores modelos, todos disponibilizados de forma aberta para a comunidade, também foram propostas contribuições acadêmicas por meio de publicações no âmbito acadêmico.

A análise da literatura relevante e a avaliação de trabalhos correlatos permitiram constatar que a arquitetura AP *BiLSTM-CRF* emergiu como a escolha recorrente em muitas pesquisas, sendo empregada em diversos modelos que representam o estado da arte em diversas tarefas de Processamento de Linguagem Natural (PLN), especialmente no contexto da tarefa REN.

Nos experimentos executados neste trabalho, foi possível constatar a robustez da arquitetura *BiLSTM-CRF*, evidenciada pelo seu desempenho notável em conjuntos de dados de maior envergadura com distribuição mais uniforme de entidades, como é o caso do *MultiWikiNER*. Além disso, a arquitetura demonstrou eficácia mesmo diante de situações desafiadoras, como conjuntos de dados mais restritos em termos de volume e com distribuição irregular de entidades, exemplificadas pelos conjuntos LeNER e Pl-Corpus.

No que tange à utilização de Vetores de Palavras para aprimorar o treinamento de modelos de REN, a revisão da literatura existente e a análise dos experimentos realizados sustentam a afirmação de que a incorporação desses vetores traz ganhos significativos no desempenho dos modelos. Alguns estudos até mesmo exploraram a diversidade de Vetores de Palavras em seus modelos, contudo, ainda subsistem lacunas significativas. Diversos

conjuntos de dados carecem de experimentos que avaliem a aplicabilidade dos diferentes vetores disponíveis. Neste contexto, os experimentos realizados neste trabalho enriquecem a pesquisa e o desenvolvimento de modelos mais avançados para o idioma português.

Ainda em relação aos Vetores de Palavras, foi possível perceber que vetores estáticos ajudam no desempenho do modelo e são menos complexos, de forma que o treinamento é consideravelmente mais rápido e com menor custo computacional se comparado ao uso de vetores de contexto. No entanto o salto de desempenho do vetor de contexto em relação ao estático é considerável valendo a pena seu uso, o mesmo não pode ser dito em relação a concatenação dos vetores que embora possa apresentar uma pequena melhora, tem um tempo de treinamento e custo computacional muito elevados.

Outra contribuição relevante apresentada por este trabalho foi o quadro especificando quais os requisitos de hardware e o tempo de treinamento de cada modelo, informações muito importantes que não são destacada em muitos dos trabalhos analisado, o que dificulta o planejamento e a escolha de *frameworks*, corpus e vetores e palavras. Por fim vale destacar o impacto direto deste trabalho no crescimento pessoal e profissional como professor do pesquisador e o impacto indireto na formação de seus alunos.

6.1 Trabalhos futuros

Como trabalhos futuros espera-se:

- Experimentar outros Vetores de Palavras como *BERTimbau* (SOUZA; NOGUEIRA; LOUTUFO, 2020), *Glove* (PENNINGTON; SOCHER; MANNING, 2014) e outras combinações, além de outros Corpora em diferentes domínios;
- Está sendo desenvolvido com objetivo de ser publicado, um mapeamento sistemático sobre a tarefa REN no domínio legal;
- Desenvolver uma aplicação com o melhor modelo.

REFERÊNCIAS

- AKBIK, A. *et al.* FLAIR: An easy-to-use framework for state-of-the-art NLP. In: *NAACL 2019, 2019 Annual Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*. [S.l.: s.n.], 2019. p. 54–59. Citado 6 vezes nas páginas 19, 22, 32, 34, 35 e 42.
- AKBIK, A.; BLYTHE, D.; VOLLGRAFF, R. Contextual string embeddings for sequence labeling. In: *COLING 2018, 27th International Conference on Computational Linguistics*. [S.l.: s.n.], 2018. p. 1638–1649. Citado 4 vezes nas páginas 21, 22, 26 e 42.
- ALBUQUERQUE, H. O. *et al.* Ulyssesner-br: A corpus of brazilian legislative documents for named entity recognition. In: *Computational Processing of the Portuguese Language*. [S.l.]: Springer International Publishing, 2022. ISBN 978-3-030-98305-5. Citado 6 vezes nas páginas 12, 16, 24, 26, 40 e 46.
- AMARAL, D. O. F. d. *O reconhecimento de entidades nomeadas por meio de conditional random fields para a língua portuguesa*. Dissertação (Mestrado) — Pontifícia Universidade Católica do Rio Grande do Sul, 2013. Citado na página 21.
- ARAUJO, P. H. L. de *et al.* Lener-br: A dataset for named entity recognition in brazilian legal text. In: *PROPOR*. [S.l.: s.n.], 2018. Citado 7 vezes nas páginas 12, 16, 24, 26, 39, 45 e 46.
- BATISTA, H. H. *et al.* A comparative analysis of text embedding approach to extract named entities in portuguese legal documents. In: SBC. *Anais do XVIII Encontro Nacional de Inteligência Artificial e Computacional*. [S.l.], 2021. p. 221–232. Citado 2 vezes nas páginas 24 e 26.
- BERGER, A.; GUDA, S. Threshold optimization for f measure of macro-averaged precision and recall. *Pattern Recognition*, v. 102, p. 107250, 2020. ISSN 0031-3203. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S003132032030056X>>. Citado na página 37.
- BERTAGLIA, T. F. C. *Normalização textual de conteúdo gerado por usuário*. Tese (Doutorado) — Universidade de São Paulo, 2017. Citado na página 14.
- BOJANOWSKI, P. *et al.* Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, v. 5, p. 135–146, 2017. ISSN 2307-387X. Citado 4 vezes nas páginas 14, 17, 18 e 41.

CASTRO, P. V. Q. d. *et al.* Aprendizagem profunda para reconhecimento de entidades nomeadas em domínio jurídico. Universidade Federal de Goiás, 2019. Disponível em: <<http://repositorio.bc.ufg.br/tede/handle/tede/10276>>. Citado 4 vezes nas páginas 12, 23, 26 e 27.

CONSOLI, B. *et al.* Embeddings for named entity recognition in geoscience portuguese literature. In: LREC. [S.l.], 2020. Citado 4 vezes nas páginas 23, 25, 26 e 27.

DENG, L.; YU, D. *Deep Learning: Methods and Applications*. [S.l.], 2014. Disponível em: <<https://www.microsoft.com/en-us/research/publication/deep-learning-methods-and-applications/>>. Citado na página 15.

DEVLIN, J. *et al.* BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv, 2018. Disponível em: <<https://arxiv.org/abs/1810.04805>>. Citado na página 19.

EISENSTEIN, J. *Introduction to Natural Language Processing*. MIT Press, 2019. (Adaptive Computation and Machine Learning series). ISBN 9780262042840. Disponível em: <<https://books.google.com.br/books?id=72yuDwAAQBAJ>>. Citado na página 9.

GHADDAR, A.; LANGLAIS, P. Robust lexical features for improved neural network named-entity recognition. In: *Proceedings of the 27th International Conference on Computational Linguistics*. Santa Fe, New Mexico, USA: Association for Computational Linguistics, 2018. p. 1896–1907. Disponível em: <<https://aclanthology.org/C18-1161>>. Citado 3 vezes nas páginas 10, 14 e 16.

Google. *Google Colab*. 2023. Disponível em: <<https://colab.research.google.com/>>. Acesso em: 12 de agosto 2023. Citado na página 30.

_____. *Google Colab Signup*. 2023. Disponível em: <<https://colab.research.google.com/signup>>. Acesso em: 12 de agosto 2023. Citado na página 31.

Internet Live Stats. *Overview*. 2021. Access date: 29 oct. 2021. Disponível em: <<https://www.internetlivestats.com/>>. Citado na página 11.

JUNIOR, O. D.; CLARO, D. B. Uma análise do reconhecimento textual de nomes de pessoas e organizações na computação forense. In: *Proceeding of the 6th International Conference on Forensic Computer Science*. [S.l.: s.n.], 2011. p. 7–15. Citado na página 11.

KANG, Y. *et al.* Natural language processing (nlp) in management research: A literature review. *Journal of Management Analytics*, Taylor Francis, v. 7, n. 2, p. 139–172, 2020. Disponível em: <<https://doi.org/10.1080/23270012.2020.1756939>>. Citado na página 10.

LOPER, E.; BIRD, S. NLTK: the natural language toolkit. *CoRR*, cs.CL/0205028, 2002. Disponível em: <<https://arxiv.org/abs/cs/0205028>>. Citado na página 39.

MAYNARD, D.; BONTCHEVA, K.; AUGENSTEIN, I. *Natural Language Processing for the Semantic Web*. Morgan & Claypool Publishers, 2016. (Synthesis Lectures on the Semantic Web: Theory and Technology). ISBN 9781627056328. Disponível em: <<https://books.google.com.br/books?id=5fm3DQAAQBAJ>>. Citado 2 vezes nas páginas 9 e 10.

MIKOLOV, T.; YIH, W.-t.; ZWEIG, G. Linguistic regularities in continuous space word representations. In: *Proceedings of the 2013 conference of the north american chapter of the association for computational linguistics: Human language technologies*. [S.l.: s.n.], 2013. p. 746–751. Citado na página 17.

MITCHELL, T. *Machine Learning*. McGraw-Hill Education, 1997. (McGraw-Hill international editions - computer science series). ISBN 9780070428072. Disponível em: <<https://books.google.com.br/books?id=xOGAngEACAAJ>>. Citado na página 15.

MOTA, C. C. *et al.* Reconhecimento de entidades nomeadas em documentos jurídicos em português utilizando redes neurais. In: SBC. *Anais do XVIII Encontro Nacional de Inteligência Artificial e Computacional*. [S.l.], 2021. p. 130–140. Citado 2 vezes nas páginas 24 e 26.

NOTHMAN, J. *et al.* Learning multilingual named entity recognition from wikipedia. *Artificial Intelligence*, Elsevier, v. 194, p. 151–175, 2013. Citado 3 vezes nas páginas 12, 16 e 37.

OLIVEIRA, H. O. A. y Ellen Souza y Carlos Gomes y Matheus Henrique de C. Pinto y Ricardo P. S. Filho y Rosimeire Costa y Vinícius Teixeira de M. Lopes y Nádia F. F. da Silva y André C. P. L. F. de Carvalho y A. L. I. Named entity recognition: a survey for the portuguese language. *Procesamiento del Lenguaje Natural*, v. 70, n. 0, p. 171–185, 2023. ISSN 1989-7553. Disponível em: <<http://journal.sepln.org/sepln/ojs/ojs/index.php/pln/article/view/6488>>. Citado 2 vezes nas páginas 13 e 28.

PENG, Y.; YAN, S.; LU, Z. Transfer learning in biomedical natural language processing: An evaluation of bert and elmo on ten benchmarking datasets. In: *BioNLP@ACL*. [S.l.: s.n.], 2019. Citado na página 19.

PENNINGTON, J.; SOCHER, R.; MANNING, C. D. Glove: Global vectors for word representation. In: *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. [S.l.: s.n.], 2014. p. 1532–1543. Citado 3 vezes nas páginas 17, 18 e 49.

SANTOS, J. *et al.* Assessing the impact of contextual embeddings for portuguese named entity recognition. In: IEEE. *2019 8th Brazilian Conference on Intelligent Systems (BRACIS)*. [S.l.], 2019. p. 437–442. Citado 2 vezes nas páginas 23 e 26.

_____. De-identification of clinical notes using contextualized language models and a token classifier. In: SPRINGER. *Brazilian Conference on Intelligent Systems*. [S.l.], 2021. p. 33–41. Citado 2 vezes nas páginas 25 e 26.

SILVA, D. F. *et al.* Named entity recognition for brazilian portuguese product titles. In: SPRINGER. *Intelligent Systems: 10th Brazilian Conference, BRACIS 2021, Virtual Event, November 29–December 3, 2021, Proceedings, Part II 10*. [S.l.], 2021. p. 526–541. Citado na página 11.

SOUZA, F.; NOGUEIRA, R.; LOTUFO, R. BERTimbau: pretrained BERT models for Brazilian Portuguese. In: *9th Brazilian Conference on Intelligent Systems, BRACIS, Rio Grande do Sul, Brazil, October 20-23 (to appear)*. [S.l.: s.n.], 2020. Citado 3 vezes nas páginas 19, 42 e 49.

STRACHNYI, K. *Brief History of Neural Networks*. 2019. Disponível em: <<https://medium.com/analytics-vidhya/brief-history-of-neural-networks-44c2bf72eec>>. Acesso em: 15 novembro 2021. Citado na página 15.

TURIAN, J.; RATINOV, L.; BENGIO, Y. Word representations: a simple and general method for semi-supervised learning. In: *Proceedings of the 48th annual meeting of the association for computational linguistics*. [S.l.: s.n.], 2010. p. 384–394. Citado na página 14.

WANG, X. *et al.* Automated Concatenation of Embeddings for Structured Prediction. In: *the Joint Conference of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (ACL-IJCNLP 2021)*. [S.l.]: Association for Computational Linguistics, 2021. Citado na página [12](#).