UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
DEPARTAMENTO DE AUTOMAÇÃO E SISTEMAS
CURSO DE GRADUAÇÃO EM ENGENHARIA DE CONTROLE E AUTOMAÇÃO

Amabyle Rabeche Heck

Processamento de Linguagem Natural Aplicado a Reconhecimento de Entidades Nomeadas em Textos Legais em Português Brasileiro

Florianópolis
2022

Amabyle Rabeche Heck

Processamento de Linguagem Natural Aplicado a Reconhecimento de Entidades Nomeadas em Textos Legais em Português Brasileiro

Relatório final da disciplina DAS5511 (Projeto de Fim de Curso) como Trabalho de Conclusão do Curso de Graduação em Engenharia de Controle e Automação da Universidade Federal de Santa Catarina em Florianópolis.
Orientador: Prof. Eric Aislan Antonelo, Dr.
Supervisor: Sergio Schmiegelow, Eng.

Florianópolis
2022

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Heck, Amabyle Rabeche
Processamento de Linguagem Natural Aplicado a Reconhecimento de Entidades Nomeadas em Textos Legais em Português Brasileiro / Amabyle Rabeche Heck ; orientador, Eric Aislan Antonelo, 2022.
71 p.

Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Santa Catarina, Centro Tecnológico, Graduação em Engenharia de Controle e Automação, Florianópolis, 2022.

Inclui referências.

1. Engenharia de Controle e Automação. 2. Processamento de Linguagem Natural. 3. Reconhecimento de Entidades Nomeadas. 4. Redes neurais. I. Antonelo, Eric Aislan. II. Universidade Federal de Santa Catarina. Graduação em Engenharia de Controle e Automação. III. Título.

Amabyle Rabeche Heck

Processamento de Linguagem Natural Aplicado a Reconhecimento de Entidades Nomeadas em Textos Legais em Português Brasileiro

Esta monografia foi julgada no contexto da disciplina DAS5511 (Projeto de Fim de Curso) e aprovada em sua forma final pelo Curso de Graduação em Engenharia de Controle e Automação

Florianópolis, 21 de março de 2022.

Prof. Hector Bessa Silveira, Dr.
Coordenador do Curso

Banca Examinadora:

Prof. Eric Aislan Antonelo, Dr.
Orientador
UFSC/CTC/DAS

Sérgio Schmiegelow, Eng.
Supervisor
Empresa Dígitro Tecnologia

Prof. Jomi Fred Hubner, Dr.
Avaliador
Instituição UFSC

Prof. Eduardo Camponogara, Dr.
Presidente da Banca
UFSC/CTC/DAS

AGRADECIMENTOS

Agradeço à minha família pelo apoio e incentivo durante todo o curso e, em especial, no período de desenvolvimento deste trabalho.

Ao meu supervisor, Sérgio, e ao meu orientador, professor Eric, obrigada por me ensinarem e guiarem durante este projeto.

Por último, agradeço aos colegas de curso e a todos os professores, principalmente do corpo docente do Departamento de Automação e Sistemas, que foram essenciais para minha formação acadêmica.

RESUMO

O Reconhecimento de Entidades Nomeadas (REN) (ou *Named Entity Recognition* (NER)) é uma técnica de Processamento de Linguagem Natural (PLN) que consiste em encontrar e identificar nomes de entidades pré-definidas (como pessoas, organizações, locais, etc.) em dados textuais. Essa tarefa é uma forma de extração de informação de documentos, o que facilita a análise e a manipulação deles. Este projeto tem como objetivo obter um modelo de Inteligência Artificial (IA) que realize a tarefa de NER especificamente em textos legais escritos no idioma Português Brasileiro, reconhecendo as entidades Pessoa, Tempo, Local, Organização, Legislação e Jurisprudência. As arquiteturas de redes neurais utilizadas foram *Bidirectional Long Short-Term Memory* (BiLSTM) e *Bidirectional Encoder Representation from Transformers* (BERT) (pré-treinada em língua portuguesa brasileira), com e sem camadas *Conditional Random Field* (CRF). O *dataset* de textos jurídicos usado para realizar os treinamentos foi o *LeNER-Br*. As métricas para avaliar os desempenhos foram precisão, *recall* e *F1-score*, e o modelo treinado com melhor performance, uma rede BERT-CRF, obteve valores de métricas de, respectivamente, 90,16%, 91,86% e 91,00%, superando o modelo *baseline*.

Palavras-chave: Processamento de Linguagem Natural. Reconhecimento de Entidades Nomeadas. Redes neurais.

ABSTRACT

Named Entity Recognition is a Natural Language Processing technique that consists in finding and identifying names of pre-defined entities (such as people, organizations, locations, etc.) in textual data. This task is a type of information extraction from documents that facilitate their analysis and manipulation. This project's objective is to obtain a Artificial Intelligence model that perform the NER task specifically in legal texts written in Brazilian Portuguese, recognizing the entities Person, Time, Location, Organization, Legislation (Laws) and Legal Cases. The neural networks architectures chosen were Bidirectional Long Short-Term Memory (BiLSTM) and Bidirectional Encoder Representation from Transformers (BERT) (pre-trained in Brazilian Portuguese), both with and without Conditional Random Fields layers. The legal texts dataset used for training was *LeNER-Br*. Metrics used to evaluate the performances were precision, recall and F1-score, and the trained model with the best results (a BERT-CRF neural network) achieved the following values of metrics, respectively, 90,16%, 91,86% and 91,00%, surpassing the baseline model.

Keywords: Natural Language Processing. Named Entity Recognition. Neural networks.

LISTA DE FIGURAS

Figura 1 – Tópicos do PLN. 20
Figura 2 – Reconhecimento de entidades em frase. 22
Figura 3 – Esquemas de *tagging* - exemplo 1. 23
Figura 4 – Esquemas de *tagging* - exemplo 2. 23
Figura 5 – Aprendizado supervisionado. 24
Figura 6 – Diagrama: IA, *Machine Learning* (ML) e *deep learning*. 25
Figura 7 – Representação de um neurônio artificial. 27
Figura 8 – Rede neural simples vs rede neural profunda. 27
Figura 9 – Esquema básico de *Recurrent Neural Networks* (RNN). 29
Figura 10 – Arquitetura de uma RNN bidirecional. 29
Figura 11 – Esquema de uma célula *Long Short-Term Memory* (LSTM). 31
Figura 12 – Treinamento de modelo *CBoW*. 33
Figura 13 – Treinamento de modelo *Skip-Gram*. 33
Figura 14 – Arquitetura *encoder-decoder*. 34
Figura 15 – Treinamento de uma rede com RNNs de *encoder* e *decoder*. 35
Figura 16 – Exemplo de uso de atenção. 36
Figura 17 – Esquema de atenção *multi-head*. 37
Figura 18 – Arquitetura *transformer*. 39
Figura 19 – Arquitetura *transformer* - blocos de *self-attention*. 40
Figura 20 – Representação da entrada de uma rede BERT. 41
Figura 21 – Pré-treinamento e ajuste fino de uma rede BERT. 42
Figura 22 – Rede LSTM sem e com camada CRF. 43
Figura 23 – Proporção de publicações que usam *TensorFlow* e *PyTorch* ao passar dos anos. 51
Figura 24 – Trecho de texto em formato JSON. 52
Figura 25 – Trecho de texto em formato *Conference of Natural Language Learning* (CoNLL). 53
Figura 26 – Exemplo de entidade corretamente (*exact match*) e não corretamente (*partial match*) classificada 56
Figura 27 – Comparação entre inferências do modelo *baseline*, de Luz de Araujo *et al.* (2018), e BERT-CRF obtido neste trabalho. 61

LISTA DE QUADROS

Quadro 1 – Especificações do *hardware* usado. 56
Quadro 2 – Fontes de *corpora* coletados para formar o grande *corpus* de treinamento. 70

LISTA DE TABELAS

Tabela 1 – Parâmetros de treinamento de rede LSTM-CRF de Luz de Araujo *et al.* (2018). 46
Tabela 2 – Sentenças, *tokens* e contagem de documentos para cada conjunto do *dataset LeNER-Br*. 49
Tabela 3 – Contagem de palavras de cada classe de entidade nomeada nos conjuntos do *dataset LeNER-Br*. 50
Tabela 4 – Parâmetros base para treinamento de redes LSTM e LSTM-CRF neste trabalho. 54
Tabela 5 – Parâmetros base para treinamento de redes BERT e BERT-CRF neste trabalho. 55
Tabela 6 – Comparação entre BiLSTM e BiLSTM-CRF. 57
Tabela 7 – Comparação entre *embeddings GloVe* e *Wang2Vec*. 57
Tabela 8 – Comparação entre modelos BERT e BERT-CRF *base* e *large*. . . . 58
Tabela 9 – Comparação de resultados dos modelos NER. 59
Tabela 10 – Métricas do modelo BiLSTM-CRF de Luz de Araujo *et al.* (2018). . . 59
Tabela 11 – Métricas do melhor modelo BERT-CRF obtido neste trabalho. . . . 59

LISTA DE ABREVIATURAS E SIGLAS

BERT	*Bidirectional Encoder Representation from Transformers*
BiLSTM	*Bidirectional Long Short-Term Memory*
CoNLL	*Conference of Natural Language Learning*
CRF	*Conditional Random Field*
EI	Extração de Informação
IA	Inteligência Artificial
LSTM	*Long Short-Term Memory*
ML	*Machine Learning*
MLM	*Masked-Language Modeling*
NER	*Named Entity Recognition*
NILC	Núcleo Interinstitucional de Linguística Computacional
NSP	*Next Sentence Prediction*
PLN	Processamento de Linguagem Natural
REN	Reconhecimento de Entidades Nomeadas
RNN	*Recurrent Neural Networks*

LISTA DE SÍMBOLOS

x_k	Entradas binárias do neurônio matemático
Θ	Limiar de ativação do neurônio matemático
i	Entrada inibitória do neurônio matemático
$\sigma(x)$	Saída binária do neurônio matemático
x_t	Palavra que ocorre no *step time* t
h_{t-1}	*Hidden state* no *step time* t-1
C_t	*Cell state* no *step time* t
I_t	*Input gate* no *step time* t
F_t	*Forget gate* no *step time* t
O_t	*Output gate* no *step time* t
$e_{t,i}$	*Alignment scores* no *step time* t da saída com relação ao *step time* i da entrada
c_t	*Context vector* no *step time* t
TP	*True Positive*
FP	*False Positive*
FN	*False Negative*
β	Saída binária do neurônio matemático

SUMÁRIO

1 INTRODUÇÃO 14
1.1 OBJETIVOS 15
1.2 METODOLOGIA 15
1.3 ESTRUTURA DO RELATÓRIO 16
2 EMPRESA 17
3 FUNDAMENTAÇÃO TEÓRICA 19
3.1 PROCESSAMENTO DE LINGUAGEM NATURAL 19
3.1.1 *Named Entity Recognition* (NER) 20
3.1.2 *Tagging* 22
3.2 APRENDIZADO DE MÁQUINA 22
3.3 REDES NEURAIS 26
3.3.1 *Recurrent Neural Networks* (RNN) 27
3.3.1.1 *Long Short-Term Memory* (LSTM) 29
3.4 TOKENIZAÇÃO 30
3.5 EMBEDDINGS 31
3.6 MECANISMO DE ATENÇÃO 32
3.6.1 Arquitetura *encoder-decoder* 33
3.6.2 Atenção de Bahdanau 34
3.6.3 Atenção *Multi-Head* 37
3.7 *TRANSFORMERS* 38
3.7.1 *Bidirectional Encoder Representation from Transformers* (BERT) 39
3.8 *CONDITIONAL RANDOM FIELD* (CRF) 41
3.9 MÉTRICAS DE AVALIAÇÃO 42
3.9.1 Precisão 43
3.9.2 *Recall* 44
3.9.3 *F1-score* 44
3.9.4 Métricas *micro-average* 44
4 TRABALHOS RELACIONADOS 46
4.1 *LENER-BR: A DATASET FOR NAMED ENTITY RECOGNITION IN BRAZILIAN LEGAL TEXT* 46
4.2 *PORTUGUESE NAMED ENTITY RECOGNITION USING BERT-CRF* 46
4.3 *BERTBR: A PRETRAINED LANGUAGE MODEL FOR LAW TEXTS* . 47
4.4 RECONHECIMENTO DE ENTIDADES NOMEADAS EM TEXTOS DE BOLETINS DE OCORRÊNCIAS 47
4.5 RECONHECIMENTO DE ENTIDADES NOMEADAS PARA O PORTUGUÊS USANDO REDES NEURAIS 47
5 MATERIAIS E MÉTODOS 49

5.1 *DATASET* 49
5.2 BIBLIOTECAS 50
5.3 FORMATOS DE ARQUIVOS 51
5.4 MODELOS PRÉ-TREINADOS 53
5.5 PARÂMETROS E *EMBEDDINGS* 54
5.6 AVALIAÇÃO DE RESULTADOS 55
5.7 *HARDWARE* 56
6 RESULTADOS 57
6.1 *BIDIRECTIONAL LONG SHORT-TERM MEMORY* (BILSTM) 57
6.2 *BIDIRECTIONAL ENCODER REPRESENTATION FROM TRANSFORMERS* (BERT) 58
6.3 COMPARAÇÃO DE RESULTADOS 58
7 CONCLUSÃO 62
REFERÊNCIAS 63
ANEXO A – *CORPORA* UTILIZADOS NOS TREINAMENTOS DE *EMBEDDINGS* DO Núcleo Interinstitucional de Linguística Computacional (NILC) 70

1 INTRODUÇÃO

Atualmente há uma enorme quantidade de informação disponível em formato de texto, a ponto de não ser mais possível para os humanos a lerem por inteiro. Por isso, torna-se cada vez mais necessária a utilização da tecnologia para sumarizar documentos, indicando pontos significativos. O tipo de tecnologia empregado para isto é chamado de Extração de Informação (EI), que tem como objetivo localizar, estruturar e armazenar informação relevante de textos (CASTRO, 2013). Através de técnicas de EI, facilitam-se a análise, a organização e a manipulação de documentos.

A tarefa de Reconhecimento de Entidades Nomeadas (REN), mais conhecida pelo nome em inglês *Named Entity Recognition* (NER), é fundamental para extração de informação. Ela é uma técnica de Processamento de Linguagem Natural (PLN) que consiste em identificar nomes de entidades específicas em dados textuais de acordo com rótulos pré-definidos, como nomes de pessoas, de locais ou de organizações. (FONSECA, E. B. *et al.*, 2015).

Modelos de Inteligência Artificial (IA) e *deep learning* (aprendizado profundo) que buscam cumprir a tarefa de NER usam muito frequentemente aprendizagem supervisionada, que necessita de conjuntos de dados anotados. Além disso, não é fácil adequar modelos treinados em um idioma para outro. Mesmo modelos multi-idiomas pré-treinados, criados para serem tão adaptáveis quanto o possível através de aprendizado por transferência, têm performances melhores para idiomas bem representados nos dados de pré-treino ou linguisticamente semelhantes a estes . O desempenho se degrada quanto menos representada no pré-treino é a linguagem em questão, a ponto de, em casos mais extremos, a performance ser efetivamente aleatória (EBRAHIMI; KANN, 2021). Por isso, no geral, modelos treinados especificamente para um idioma têm melhores desempenhos.

Boa parte dos modelos e *datasets* hoje disponíveis são em língua inglesa. Há a necessidade de pesquisa em outros idiomas, como a língua portuguesa, escopo em que este trabalho se encontra. Além disso, diferente de projetos de cunho exclusivamente acadêmico, este foi realizado tendo em vista uma demanda apresentada na empresa em que o trabalho se desenvolveu.

As utilidades da EI, e notadamente de REN, são diversas. Por exemplo, podem ser voltadas para textos criminais (CHAVES, 2021) ou médicos/clínicos (DE SOUZA *et al.*, 2019). No caso deste projeto, com intento de aplicação principalmente em soluções de inteligência investigativa e segurança pública, o foco será em textos legais/jurídicos, como petições, mandatos e decisões judiciais em geral. O resultado desejado deve ser um ou mais modelos que ajudem a identificar facilmente elementos-chave em textos legais em língua portuguesa, permitindo extrair de uma grande quantidade de documentos diversos apenas textos/trechos de interesse.

Para isso, os dois tipos básicos de modelos de *deep learning* escolhidos para se treinar, testar e comparar foram *Bidirectional Encoder Representation from Transformers* (BERT) e *Long Short-Term Memory* (LSTM). Junto com cada um deles, foram também usados *Conditional Random Fields* (CRFs) na busca de melhorar os desempenhos. As métricas escolhidas para avaliar os desempenhos foram: precisão, *recall* e *F1-score*.

1.1 OBJETIVOS

O objetivo geral deste projeto é produzir um modelo treinado de *deep learning* baseado em IA (especificamente, em técnicas de PLN) que encontre a citação de entidades pré-definidas em textos legais em língua portuguesa (isto é, que realize a tarefa de NER).

Os objetivos específicos são:

- Pesquisar e utilizar diferentes tipos de modelos de *deep learning*;
- Disponibilizar um modelo que seja adaptado para o contexto da empresa em que o projeto foi realizado, isto é, para utilização em inteligência investigativa e segurança pública;
- Identificar dentre as métricas existentes e aplicáveis para avaliar o desempenho de modelos NER quais são as mais relevantes;
- Comparar desempenhos de diferentes modelos, em especial entre modelos de redes neurais (*Recurrent Neural Networks* (RNN) e arquitetura *transformer*);
- Comparar desempenho dos modelos obtidos com os de trabalhos relacionados.

1.2 METODOLOGIA

O método de trabalho utilizado neste projeto foi composto pelos passos citados a seguir, realizados respectivamente:

- Revisão bibliográfica sobre os conceitos teóricos necessários;
- Pesquisa sobre ferramentas, modelos, *datasets*, bibliotecas e pacotes a serem utilizados na implementação e treinamento de modelos;
- Treinamento de modelos de rede neural para realizar a tarefa NER;
- Avaliação dos resultados com base nas métricas escolhidas.

1.3 ESTRUTURA DO RELATÓRIO

A estruturação deste documento se dá da seguinte maneira:

No Capítulo 2 é feita uma sucinta descrição da empresa onde o projeto foi realizado, apontando sua área de atuação e em que contexto nela se encaixa o trabalho desenvolvido.

No Capítulo 3 há a fundamentação teórica, com os aspectos conceituais necessários para compreender os procedimentos realizados neste projeto.

No Capítulo 4 são apresentados alguns trabalhos acadêmicos relacionados a este.

No Capítulo 5 estão expostos as ferramentas e os procedimentos adotados.

No Capítulo 6 estão os resultados obtidos.

Por fim, o Capítulo 7 engloba a conclusão e sugestões de futuros trabalhos voltados para este tema.

2 EMPRESA

Este projeto foi realizado na empresa Dígitro.

Ela surgiu em 1977 como uma start-up, pioneira no cenário de tecnologia em Florianópolis. Hoje, tem mais de 300 colaboradores e clientes em todo o Brasil e na América Latina.

A Dígitro cria soluções em duas principais áreas: comunicação corporativa e inteligência investigativa. Nesta segunda, ela se especializa em segurança e defesa, e é reconhecida como Empresa Estratégica de Defesa pelo Ministério Brasileiro de Defesa (ABIMDE, 2021).

A inovação é um dos pilares da Dígitro, que entende a transformação, a exploração de diferentes conceitos e o desenvolvimento de novas soluções como necessários para atender as demandas do mercado. Para pesquisar e implementar novidades é que existe a equipe de Arquitetura e Pesquisa na empresa, onde foi realizado o projeto, voltado para a área de inteligência investigativa.

A Dígitro desenvolve módulos/soluções com base na sua plataforma de inteligência *IntelleTotum*, que oferece recursos com o objetivo de auxiliar no direcionamento das ações estratégicas, táticas e operacionais das instituições (DÍGITRO, 2017). Para isso, a plataforma facilita a obtenção de dados, a análise de informações e a disseminação de conhecimento.

O *IntelleTotum* realiza coleta inteligente de informações e conteúdos dispersos em uma grande base de dados, que engloba tanto materiais de fontes abertas (como notícias e mídias sociais) quanto documentos internos, conforme desejo do cliente, tudo isso em sintonia com a legislação nacional e alinhado aos padrões do Sistema Brasileiro de Inteligência (SISBIN - Lei Nº 9.883, de 7 de dezembro de 1999).

Os módulos da Solução de Inteligência Investigativa e Estratégica englobam, conforme Dígitro (2017):

- Coleta de dados em fontes estruturadas e não estruturadas
- Sistemas confiáveis e seguros de gravação
- Cruzamento de informações
- Monitoramentos preventivos
- Diferentes perspectivas para análise de dados
- Indicadores para tomadas de decisão
- Disparo de ações integradas e críticas

Para clientes envolvidos na segurança pública, existe também o módulo de análise de interceptações, o Guardião, que provém ferramentas de apoio às tarefas

relacionadas à recepção, ao tratamento e à análise de informações obtidas através da quebra dos sigilos telefônico, telemático, de dados e financeiros.

Existe uma demanda interna para incluir nos produtos de inteligência, para clientes públicos e privados, uma ferramenta que ajude os usuários a identificar *tags* ou entidades-chaves em textos em língua portuguesa sem ter que lê-los para isso. Um passo essencial para esse desenvolvimento é a criação de um modelo de IA que realize a tarefa de NER. É nesse contexto em que este trabalho está inserido, e é a necessidade interna de modelos REN para serem usados no novo módulo citado que busca atender.

3 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo, são apresentados os principais aspectos conceituais usados como base para a realização deste projeto.

3.1 PROCESSAMENTO DE LINGUAGEM NATURAL

A linguagem natural é como os humanos se comunicam e se expressam de forma falada ou escrita. Ela é altamente ambígua e variável, está sempre mudando e evoluindo. No entanto, ao mesmo tempo em que os humanos têm uma grande capacidade de produzir e compreender linguagens, a ponto de interpretar até nuances sutis e elaboradas, é muito difícil para nós entender formalmente e descrever regras que as regem (GOLDBERG, 2017). Sendo assim, usar computadores para assimilar e produzir linguagem é bastante desafiador, porém cada vez mais necessário, com grandes quantidades de informação em formato de linguagem natural sendo geradas a cada minuto e com a busca de maior interação entre seres humanos e máquinas.

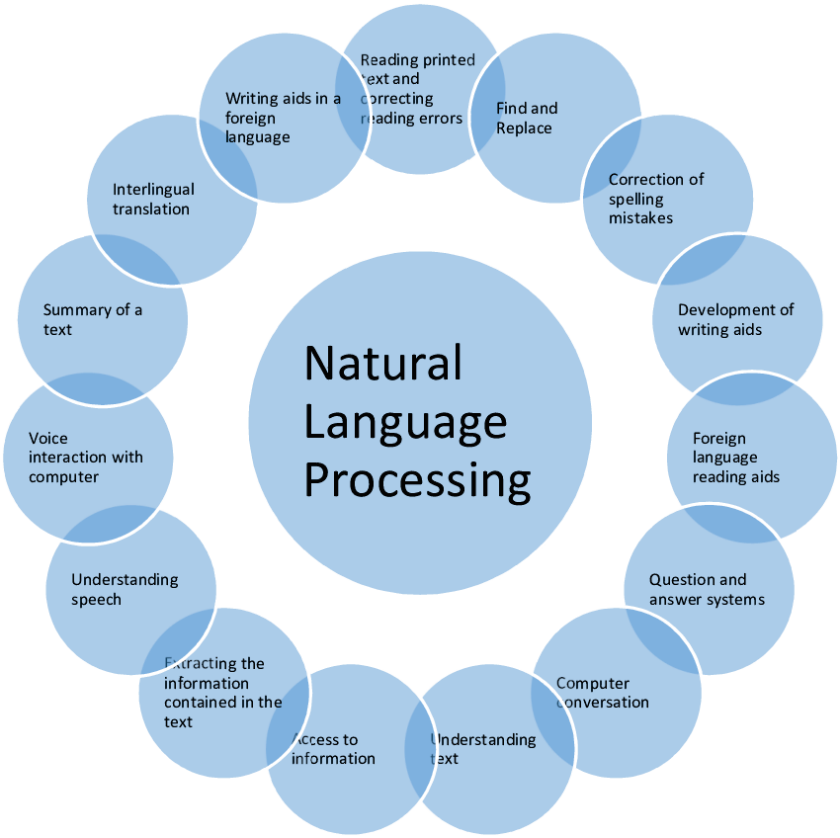
O Processamento de Linguagem Natural é definido como a área de pesquisa e aplicação que explora como os computadores podem ser usados para entender e manipular linguagem natural em formato de fala ou texto para alguma utilidade (CHOWDHURY, 2003). Este é um campo de pesquisa amplo e com muita aplicabilidade, tanto em tecnologias atuais quanto nas que ainda estão sendo desenvolvidas para o futuro. A Figura 1 mostra alguns dos tópicos incluídos neste âmbito, como por exemplo: traduções entre idiomas, sumarização de textos, análise de sentimentos e sistemas de pergunta e resposta. O PLN também é amplo por englobar diversas outras áreas, como matemática, ciência da computação e linguística.

O início das pesquisas relacionadas ao PLN datam dos anos 40 e 50, e a primeira aplicação de interesse era a tradução automática. Nas décadas iniciais, o desenvolvimento se dava principalmente em abordagens simbólicas, na forma de aplicação de um conjunto de regras aos dados de interesse (KUMAR, 2011). A partir dos anos 80 e 90, com a introdução e popularização de modelos de *machine learning*, abordagens estatísticas surgiram. Isso, somado a fatores como o aumento da disponibilidade de textos em formato eletrônico, popularização de computadores com velocidades de processamento e memórias superiores e o advento da Internet, viabilizou o crescimento deste campo (LIDDY, 2001). Mais recentemente, nas duas últimas décadas, com a difusão de modelos de redes neurais e *deep learning*, as pesquisas em PLN têm alcançado resultados cada vez melhores com abordagens conexionistas e híbridas (que misturam técnicas).

Ainda que todo este campo seja chamado de PLN, há na verdade dois focos distintos: o **processamento** e a **geração** de linguagem. Enquanto o primeiro se refere à análise de linguagem com o propósito de produzir alguma representação dela, o

segundo tem o intuito contrário, produzir linguagem a partir de uma representação (LIDDY, 2001). A tarefa de processamento de linguagem natural é comparável à de um leitor ou ouvinte humano, e a de geração à de um escritor ou orador. O foco deste trabalho é na análise de texto. Portanto, apesar de muito da teoria e da tecnologia ser compartilhado entre as duas divisões, os fundamentos apresentados aqui são especialmente aplicados ao processamento de linguagem natural.

Figura 1 – Tópicos do PLN.



Fonte: Sutcu e Aytekin (2019)

3.1.1 *Named Entity Recognition* (NER)

O Reconhecimento de Entidades Nomeadas (REN, ou em inglês NER) é uma técnica de PLN que realiza a varredura de textos, encontra neles algumas entidades fundamentais e as classifica entre categorias predefinidas.

Alguns exemplos de entidades muitas vezes usadas são (MARSHALL, 2019):

- *Pessoa* (e. g. Alexandre Dumas, Elvis Presley);
- *Organização* (e. g. Google, Universidade Federal de Santa Catarina);
- *Tempo* (e. g. 02/02/2022, 13:07);

- *Local* (e. g. Florianópolis, Praça XV de Novembro);
- *Valor* (e. g. quatro, 20);
- *Obra* (e. g. Hamlet, A Noite Estrelada);
- *Coisa* (e. g. guarda-chuvas, caderno);
- *Abstração* (e. g. desespero, felicidade);
- *Quantidade de dinheiro* (e. g. £600, R$ 9,90).

A escolha de entidades de interesse depende diretamente do tipo de texto em que o modelo NER será utilizado, diferindo de aplicação para aplicação, podendo incluir, por exemplo, entidades relacionadas a medicações, doenças e outros termos biomédicos (NADEAU; SEKINE, 2007).

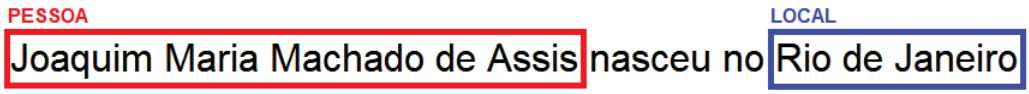
NER, no geral, é utilizado em situações em que é útil se ter uma visão global e de alto nível de uma grande quantidade de textos. Um modelo de REN facilita as tarefas de entender assuntos tratados em documentos e agrupar conjuntos de textos se baseando em relevância ou similaridade. Alguns possíveis usos desta técnica são (MARSHALL, 2019):

- *Recursos humanos*: agilizar o processo de contratação por ajudar a selecionar currículos de interesse; melhorar fluxo de trabalho interno por categorizar perguntas ou reclamações de funcionários;
- *Suporte ao cliente*: diminuir o tempo de resposta por categorizar solicitações, questionamentos e reclamações e realizar filtragem de acordo com prioridades;
- *Mecanismos de busca e recomendação*: melhorar a velocidade e relevância de resultados de buscas e recomendações por sumarizar textos descritivos e avaliações (Booking.com é um exemplo de empresa que utiliza esta estratégia (MAVRIDIS, 2016));
- *Classificação de conteúdo*: navegar mais facilmente por conteúdo e obter *insights* do que é popular por identificar assuntos e temas em *posts* e artigos;
- *Assistência médica*: melhorar padrão de atendimento aos clientes e reduzir cargas de trabalhos por extrair informação essencial de relatórios médicos (a empresa Roche usa isso em relatórios de patologia e radiologia (PANDIT; ELLAFI; SHARMA, 2019));
- *Academia*: permitir que estudantes e pesquisadores encontrem material relevante mais rápido por sumarizar documentos e destacar termos-chave, tópicos e temas (a plataforma digital Europeana usa NER para tornar jornais históricos digitalizados pesquisáveis (NEUDECKER, 2014)).

3.1.2 *Tagging*

Em geral, um modelo de NER recebe como entrada textos ou trechos de textos no formato de uma sequência de *tokens* (x_1, x_2, ..., x_n) - representando palavras e sinais de pontuação - e sua saída é uma sequência de *tags* (y_1, y_2, ..., y_n) de tipos pré-determinados. Um exemplo básico para fins explicativos desta ideia pode ser apresentado usando a frase na Figura 2. Se a entrada do modelo for (‘‘Joaquim’’, ‘‘Maria’’, ‘‘Machado’’, ‘‘de’’, ‘‘Assis’’, ‘‘nasceu’’, ‘‘no’’, ‘‘Rio’’, ‘‘de’’, ‘‘Janeiro’’), a saída desejada seria algo como (PESSOA, PESSOA, PESSOA, PESSOA, PESSOA, -, -, LOCAL, LOCAL, LOCAL), com as posições das *tags* ordenadas, considerando o vetor de entrada.

Figura 2 – Reconhecimento de entidades em frase.



Fonte: Arquivo pessoal.

As *tags* serão definidas por dois fatores: os tipos de entidade de interesse e o esquema de *tagging* usado. Há dois esquemas principais que serão apresentados a seguir (SOUZA; NOGUEIRA; LOTUFO, 2019).

O esquema IOB2 usa as *tags* {B-, I-, O} combinadas com os nomes de entidades. A *tag* B- (do inglês *beginning*) indica o início de uma entidade, a *tag* I- (de *inside*) marca os *tokens* que pertencem a uma mesma entidade e a O (de *outside*) é usada para *tokens* que não pertencem a nenhuma das entidades.

O esquema IOBES (ou BILOU) usa as mesmas *tags* do IOB2 e mais duas: a *tag* E-/L- (*ending*/*last*) é usada para marcar o último *token* de uma entidade e a *tag* S-/U- (*single*/*unit*) indica uma entidade formada por um único *token*.

Dois exemplos de *tagging* usando ambos os esquemas estão na Figura 3 e Figura 4. As entidades consideradas são PESSOA (PES) e LOCAL (LOC).

3.2 APRENDIZADO DE MÁQUINA

O processo de aprendizado é definido por dois fatores: a aquisição de conhecimento e a capacidade de usá-lo (WITTEN; FRANK; HALL, 2011). Relacionado a este conceito, o Aprendizado de Máquina, ou *Machine Learning* (ML), é uma área da IA cujo objetivo é desenvolver técnicas computacionais que permitam que sistemas sejam capazes de aprender, identificar padrões e tomar decisões com o mínimo de intervenção humana. Isso ocorre tipicamente através de observação de dados e interações com o ambiente.

Figura 3 – Esquemas de *tagging* - exemplo 1.

Joaquim	Maria	Machado	de	Assis	nasceu	no	Rio	de	Janeiro
IOB2 tagging									
B-PES	I-PES	I-PES	I-PES	I-PES	O	O	B-LOC	I-LOC	I-LOC
IOBES tagging									
B-PES	I-PES	I-PES	I-PES	E-PES	O	O	B-LOC	I-LOC	E-LOC

Fonte: Arquivo pessoal.

Figura 4 – Esquemas de *tagging* - exemplo 2.

Joaquim	nasceu	no	Rio	de	Janeiro
IOB2 tagging					
B-PES	O	O	B-LOC	I-LOC	I-LOC
IOBES tagging					
S-PES	O	O	B-LOC	I-LOC	E-LOC

Fonte: Arquivo pessoal.

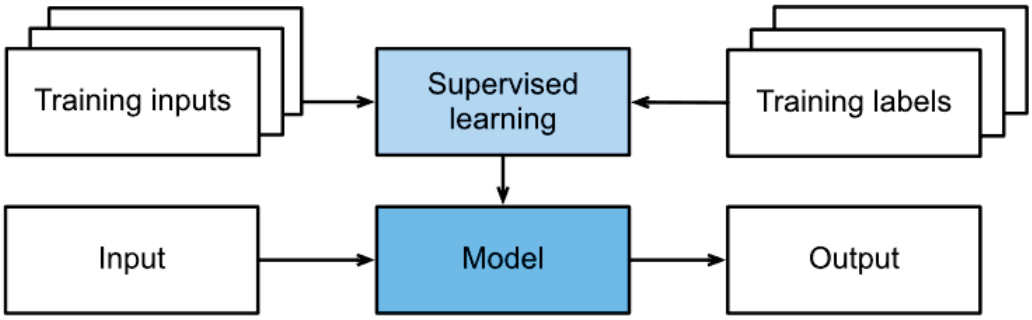
Os métodos de aprendizado mais populares são (SAS-INSIGHTS, 2021):

- *Aprendizado supervisionado*: algoritmos são treinados usando exemplos rotulados, ou seja, com entradas para quais as saídas são conhecidas. Ao observar estes dados de treinamento, o algoritmo encontra padrões para criar um modelo que preveja as saídas corretas para dados de entrada não-rotulados, conforme a Figura 5. Este tipo de aprendizado é comumente usado em aplicações nas quais dados históricos preveem prováveis eventos futuros. Exemplos: regressão logística, *support-vector machine*;
- *Aprendizado não-supervisionado*: o treinamento é realizado sem dados rotulados. Isso significa que o algoritmo deve explorar os dados e encontrar alguma estrutura ou padrão. Exemplos: agrupamento *k-means*, mapas auto-organizáveis.

- *Aprendizado semi-supervisionado*: é utilizado para as mesmas aplicações do aprendizado supervisionado, mas este método manipula dados rotulados e não-rotulados no treinamento (no geral uma quantidade grande de dados sem rótulos e uma menor quantidade de rotulados). É bastante útil quando o custo associado à rotulação é alto demais para possibilitar um processo de treinamento totalmente rotulado.

- *Aprendizado por reforço*: o algoritmo descobre através de testes do tipo "tentativa e erro" quais ações rendem as maiores recompensas. Este tipo de aprendizado possui três componentes principais: o **agente** (o aprendiz ou tomador de decisão), o **ambiente** (tudo com que o agente interage) e as **ações** (o que o agente pode fazer). O objetivo é que o agente escolha ações que maximizem a recompensa total em um período de tempo determinado. A política de recompensas é definida pelo programador. O agente atingirá o objetivo da melhor maneira possível e mais rapidamente se seguir um bom caminho, tomando boas decisões; o foco do aprendizado por reforço é descobrir o melhor caminho até o objetivo. Este método é normalmente usado em jogos, robótica e navegação.

Figura 5 – Aprendizado supervisionado.



Fonte: Zhang *et al.* (2021)

Neste projeto, o método usado foi aprendizado supervisionado.

No geral, os dados rotulados usados para a criação de um modelo deste tipo de aprendizado são separados em três conjuntos: **treinamento**, **validação** e **teste**.

O conjunto de treinamento é formado pela maior parte dos dados disponíveis e é o que será exposto para o algoritmo enquanto este cria e modifica o modelo, ao longo de várias iterações.

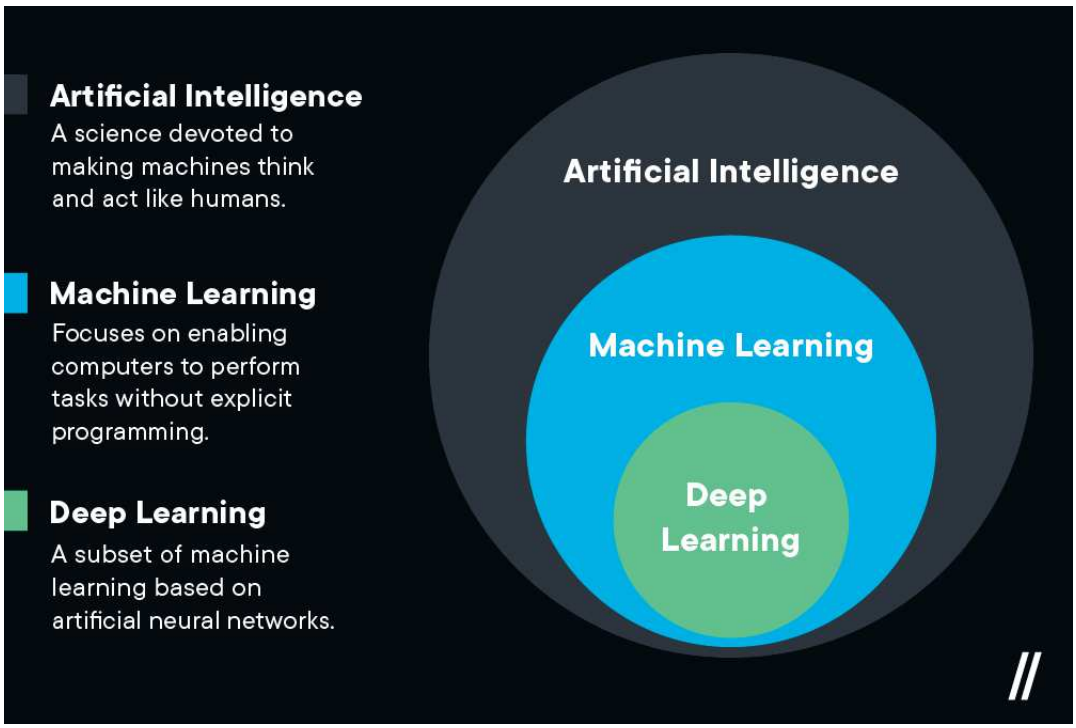
Alguns parâmetros matemáticos do modelo são variáveis, porém normalmente não são atualizados ao longo das rodadas de treino. Para escolher os melhores valores para eles, usam-se avaliações do modelo no conjunto de validação. Essas avaliações

também podem ser utilizadas para verificar se as predições do modelo estão melhorando ou piorando, sabendo assim qual o melhor momento para parar o treinamento.

Por último, o conjunto de testes é usado para analisar o desempenho do modelo obtido através do treinamento. Como os dados de teste também são rotulados, assim como os dois outros conjuntos citados, é possível calcular as porcentagens de acertos que o modelo tem, comparando suas saídas de inferência para cada entrada com as já conhecidas.

Neste trabalho, o foco é num ramo do aprendizado de máquina chamado *deep learning* (aprendizado profundo), que é baseado no uso de redes neurais artificiais com três ou mais camadas que realizam a extração de *features* (características) dos dados de entrada automaticamente. Modelos de *deep learning* apresentam uma abordagem sofisticada para tratar problemas complexos que os algoritmos mais simples de *machine learning* têm dificuldade em resolver, como tarefas que envolvem análise de imagens, vídeos, áudios ou textos. A Figura 6 resume o relacionamento entre IA, ML e *deep learning*, bem como suas definições.

Figura 6 – Diagrama: IA, ML e *deep learning*.



Fonte: Middleton (2021)

3.3 REDES NEURAIS

As redes neurais artificiais são inspiradas no cérebro animal. Basicamente, elas são formadas por conjuntos de unidades chamadas neurônios, cujos parâmetros matemáticos são ajustados durante o treinamento, e podem aprender e modelar relações entre dados de entrada e saída altamente complexas e não-lineares. Hoje, os resultados obtidos de modelos de redes neurais são o estado-da-arte em diversas áreas ligadas à IA, como tradução automática, reconhecimento de fala, processamento de textos, visão computacional e outras. Além de serem a base do *deep learning*, as redes neurais como um todo formam o ramo da IA de nome Inteligência Artificial Conexionista.

O nascimento das redes neurais se deu com a primeira publicação de um modelo matemático de um neurônio biológico por McCulloch e Pitts (1943), que pode ser expresso pela Equação (1) com entradas e saídas binárias, onde x_k são as entradas, Θ é o limiar que deve ser ultrapassado para que a saída seja 1 e i é uma entrada inibitória, que quando ativada não permite que o neurônio dispare uma saída.

$$\sigma(x) = \begin{cases} 1 & \text{se } \sum_{k=1}^{n} x_k > \Theta \text{ e } i = 0 \\ 0 & \text{caso contrário} \end{cases} \tag{1}$$

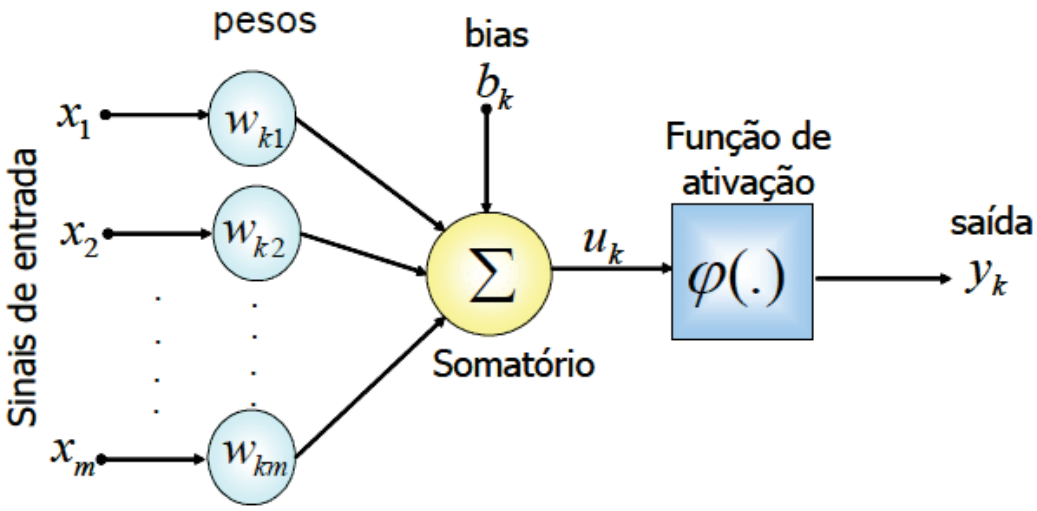
Mais tarde, o conceito de ***perceptron*** foi criado por Rosenblatt (1958), introduzindo pesos (números reais que determinam a importância de cada entrada) ao neurônio matemático.

O modelo atualmente aceito como o padrão de neurônio é o apresentado na Figura 7. Além das entradas (incluindo o *bias*, uma entrada fixa com peso variável), dos pesos e do somatório, já presentes em modelos anteriores, neste há uma chamada **função de ativação**, que determina qual será a saída do neurônio com base na soma das entradas.

Uma rede neural é formada por um conjunto de neurônios artificiais conectados entre si. De forma geral, uma rede pode ser treinada através do ajuste de seus pesos para que, para determinados tipos de dados de entrada, obtenham-se saídas desejadas. Faz-se isso com o propósito de que essa rede neural que “aprendeu” com os dados de treinamento possa ser utilizada para prever saídas para novos dados.

A primeira publicação com uma rede neural multicamadas foi feita por Fukushima (1975). Hoje, vários modelos de redes possuem diversas camadas. Na Figura 8 vemos uma comparação entre redes neurais simples e redes neurais profundas, com muitas camadas intermediárias (ou ocultas/*hidden*).

Figura 7 – Representação de um neurônio artificial.

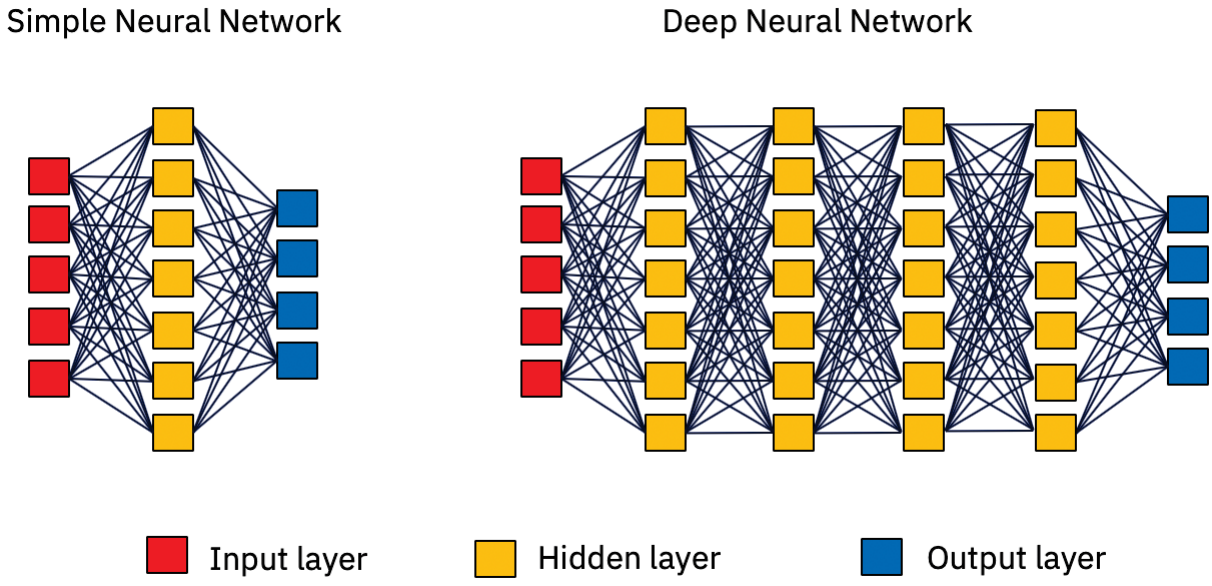


Fonte: Soares e Silva (2011)

3.3.1 *Recurrent Neural Networks* (RNN)

Para lidar com diferentes tipos de dados, existem modelos diversos de redes neurais, com adaptações que se encaixam em cada aplicação. Para imagens, por exemplo, é comum o uso das Redes Neurais Convolucionais, que possuem camadas

Figura 8 – Rede neural simples vs rede neural profunda.



Fonte: Ceron (2020)

de filtros e processam informação espacial de forma eficiente. Para dados de entrada textuais, é importante ter maneiras de preservar sequências. Com esse objetivo, foram criadas as Redes Neurais Recorrentes, mais conhecidas pelo nome em inglês, *Recurrent Neural Networks* (RNNs).

RNNs introduziram o conceito de memória com variáveis de estado para guardar informação passada que, junto com entradas atuais, determinam as saídas (ZHANG *et al.*, 2021). Diferente das redes *feed-forward* (cujo fluxo de dados se dá em uma única direção, como os exemplos já mostrados neste capítulo), elas podem ser bidirecionais, processando dados em ambas as direções.

Um conceito importante para entender o funcionamento de RNNs é o de *hidden state* ou *hidden variable*. Considera-se que um texto é uma sequência temporal e que a probabilidade de uma determinada palavra ocorrer na posição x_t (no *step time* t) depende de palavras anteriores (o que é chamado de modelo *n-gram*). Incluir o efeito de palavras cada vez mais antigas aumenta exponencialmente o número de parâmetros em um modelo (ZHANG *et al.*, 2021). Então, prefere-se o uso da alternativa na Equação (2), onde h_{t-1} é o *hidden state* que guarda a sequência de informações até o *step time* t-1.

$$P(\, x_t \mid x_{t-1}, ..., x_1 \,) \approx P(\, x_t \mid h_{t-1} \,) \tag{2}$$

Em geral, o *hidden state* em qualquer *step time* t pode ser computado baseado na entrada atual (x_t) e no *hidden state* anterior, h_{t-1}, conforme Equação (3). A função f é escolhida de forma que o *hidden state* seja uma boa aproximação de tudo que a rede "viu" até o momento, sem que os cálculos e o modelo final sejam muito custosos em questão de processamento ou memória.

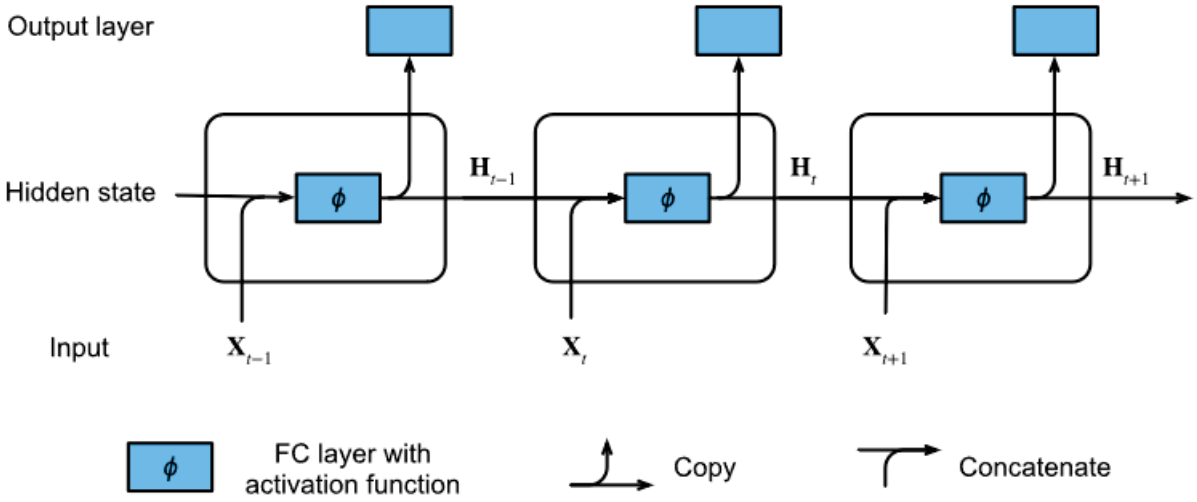
$$h_t = f(\, x_t,\ h_{t-1} \,) \tag{3}$$

O esquema básico de uma RNN é ilustrado na Figura 9, com três *time steps* adjacentes. Para qualquer passo t, o cálculo do *hidden state* H_t se dá em dois passos: (1) concatenar entrada x_t com o *hidden state* H_{t-1}, (2) utilizar essa junção como entrada da próxima camada. Na imagem, os ϕ são as funções de ativação. Assim, o *hidden state* H_t do passo atual será usado para computar o do próximo passo t+1 e também na camada de saída (no caso do exemplo na figura).

Já no caso de uma RNN bidirecional, introduzida pela primeira vez por Schuster e Paliwal (1997), há a adição de informação das palavras futuras do texto. Um esquema simplificado está na Figura 10. Neste caso, o *hidden state* H_t (total) passa a ser a junção dos *hidden states* em ambas as direções ($\overrightarrow{H_t}$ e $\overleftarrow{H_t}$).

Na prática, camadas bidirecionais são usadas com menos frequência. Dependendo da aplicação, utilizá-las erroneamente pode acarretar em bons resultados durante o treinamento, quando há acesso aos textos completos (palavras do passado e

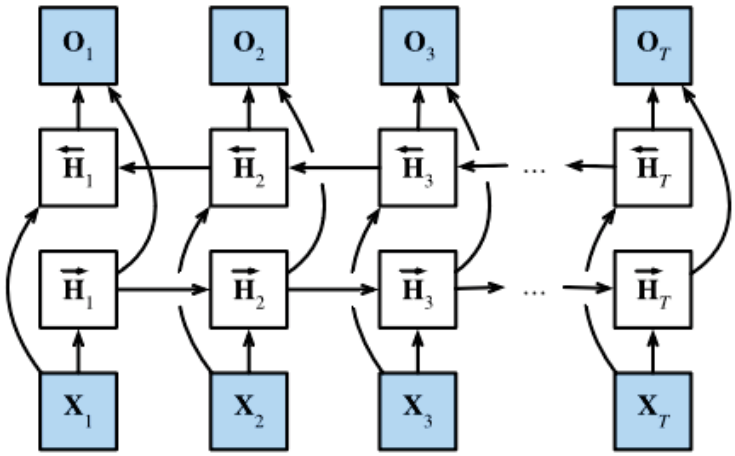
Figura 9 – Esquema básico de RNN.



Fonte: Zhang *et al.* (2021)

do futuro), e ruins no uso real. Também podem ser modelos lentos. Contudo, uma das aplicações em que este tipo de RNN obtém bons resultados é na tarefa de NER.

Figura 10 – Arquitetura de uma RNN bidirecional.



Fonte: Zhang *et al.* (2021)

3.3.1.1 *Long Short-Term Memory* (LSTM)

Para lidar com o desafio de preservar informação de um passado mais distante, o modelo de rede chamado LSTM foi criado por Hochreiter e Schmidhuber (1997).

É mais difícil para uma RNN comum, nos moldes vistos até aqui, conectar às entradas atuais um contexto que está a algumas frases de distância (IBM-CLOUD-EDUCATION, 2020). Na tentativa de solucionar isso, LSTMs têm as células de memória, ou *memory cells*, nas camadas ocultas com três *gates* principais: *input*, *output* e *forget*. Esses *gates*, cujo comportamento é aprendido durante o treinamento, controlam o fluxo de informação necessária para prever a saída da rede. A memória de longo prazo (*long-term memory*) é usualmente chamada neste modelo de *cell state* e será simbolizada aqui por C_t.

O *input gate* determina que informação entra no *cell state*, o *forget gate* informa ao *cell state* que informação esquecer e o *output gate* é necessário para determinar que informação do *cell state* irá para o próximo *hidden state*.

Um esquema de uma célula de LSTM é apresentado na Figura 11. As funções que regem as variáveis estão na Equação (4), onde a nomenclatura $W_{x\_}$ representa as matrizes com os pesos matemáticos, $b\_$ são os *bias*, σ é a função sigmoide ($\sigma(x) = \frac{1}{1+e^{-x}}$), I_t é o *input gate*, F_t é o *forget gate*, O_t é o *output gate*, $\tilde{C}_t$ uma variável que chamaremos aqui de *candidate memory* e H_t é o *hidden state*. O operador $\odot$ é o produto de Hadamard, ou produto *element-wise*.

$$
\begin{aligned}
I_t &= \sigma(\, X_t W_{xi} + H_{t-1} W_{hi} + b_i \,), \\
F_t &= \sigma(\, X_t W_{xf} + H_{t-1} W_{hf} + b_f \,), \\
O_t &= \sigma(\, X_t W_{xo} + H_{t-1} W_{ho} + b_o \,), \\
\tilde{C}_t &= \tanh(\, X_t W_{xc} + H_{t-1} W_{hc} + b_c \,), \\
C_t &= F_t \odot C_{t-1} + I_t \odot \tilde{C}_t, \\
H_t &= O_t \odot \tanh(\, C_t \,)
\end{aligned}
\tag{4}
$$

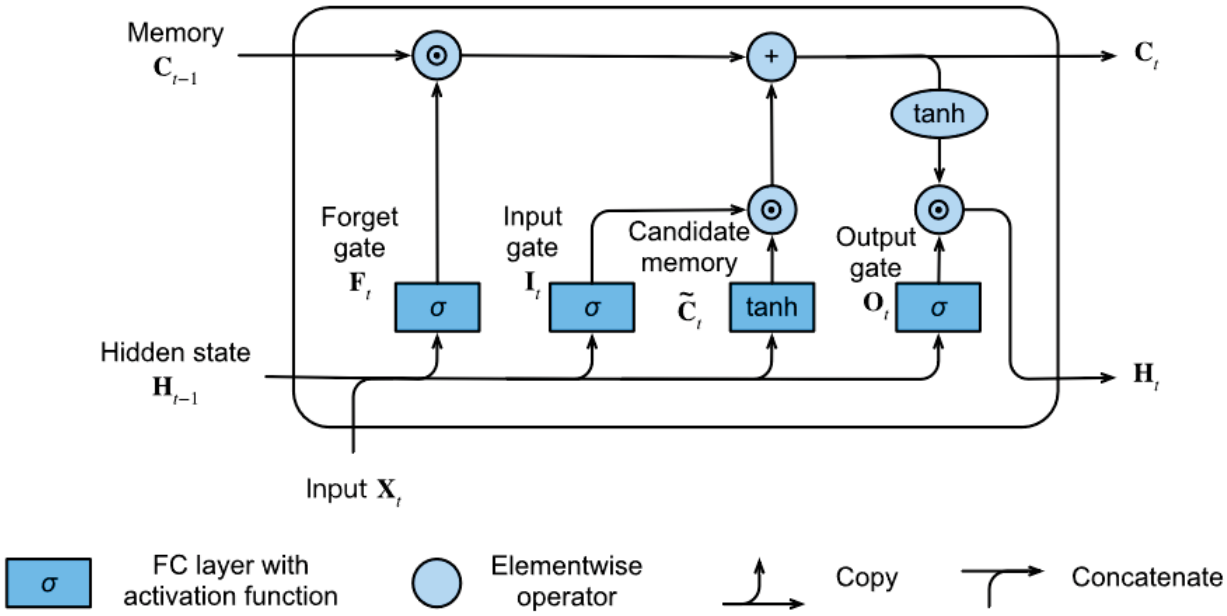
3.4 TOKENIZAÇÃO

A tokenização, ou segmentação de palavras, quebra a sequência de caracteres de um texto localizando os limites das palavras nele (BARBOSA *et al.*, 2017). As palavras identificadas (assim como outros caracteres, como vírgulas, pontos e outros, quando for pertinente) são chamadas de *tokens*.

A tokenização é bem estabelecida e entendida para linguagens artificiais, como linguagens de programação (PALMER, 2010), que muitas vezes são definidas para eliminar ambiguidades léxicas e estruturais. Mas nem sempre isso acontece com linguagens naturais, em que os mesmos caracteres podem servir para diferentes fins e a sintaxe não é estritamente definida.

Há uma diferença entre abordagens de tokenização para as linguagens delimitadas por espaços e para as não segmentadas. Em linguagens delimitadas por espaço,

Figura 11 – Esquema de uma célula LSTM.



Fonte: Zhang *et al.* (2021)

com é o caso da língua portuguesa e a maioria das europeias, grande parte dos *tokens* são separados entre si por espaços em branco e a maioria das ambiguidades estão no uso de sinais de pontuação, que podem ter diferentes funções em uma mesma sentença. Em linguagens não segmentadas, como o chinês e o tailandês, as palavras são escritas em sucessão, sem indicação de limites entre elas, e sua tokenização requer informação léxica e morfológica adicional (PALMER, 2010).

Ao usar tokenização com modelos *machine learning*, transformam-se textos e sentenças em *tokens* conhecidos, que formam o chamado **vocabulário**.

Além da tokenização tradicional, palavra por palavra, há outros tipos, como a tokenização *subword-level*. Ela tem um vocabulário formado por unidades que são "pedaços" de palavras, também chamados *subtokens*, com sequências de caracteres, e palavras inteiras.

3.5 EMBEDDINGS

Para usar um texto ou frase como entrada de um modelo matemático de *deep learning* é preciso transformá-lo em uma informação numérica, um vetor. Conhecidos como *word vectors*, esses vetores são representações de palavras.

O método mais simples de vetorizar uma palavra é transformá-la em um vetor *one-hot*. Suponha que todo o vocabulário que será considerado pelo modelo tem N palavras diferentes, cada uma com um índice variando de 0 a $N-1$. Assim, podemos

representar qualquer palavra de índice *i* como um vetor de tamanho *N* com o valor 1 na posição *i* e 0 nas demais.

Essa técnica é simples, porém tem limitações. Uma delas é que os *word vectors* terão o tamanho do vocabulário, que pode chegar a dezenas de milhares de palavras, tornando-os grandes. Outra desvantagem é que vetores *one-hot* não conseguem expressar a similaridade entre diferentes palavras (e.g. por similaridade de cossenos, uma métrica bastante usada), já que extraem pouca informação sintática e semântica de textos (FONSECA, C., 2021).

A solução é utilizar outro método: *word embedding*. Essa técnica transforma palavras em vetores reais de tamanho fixo, no geral entre 100 e 1000, e cada dimensão deles guarda informação semântica da palavra que representam, como gênero ou conjugação verbal. Deseja-se que palavras semelhantes tenham vetores próximos no espaço semântico criado.

O usuário não configura manualmente quais informações semânticas ele deseja, um algoritmo de *machine learning* (normalmente uma rede neural) aprende isso a partir do contexto das sentenças usadas em seu treinamento, partindo do princípio que o significado de uma palavra está relacionado às que comumente aparecem junto dela. Esse método contorna as duas limitações de vetores *one-hot* citadas, já que o número de dimensões do *embedding* é de escolha do programador.

Um dos algoritmos mais famosos de treinamento de *embeddings* é *Word2Vec*. Nele busca-se criar representações semânticas significativas através de probabilidades condicionais, que podem ser vistas como inferência de palavras com base nas que a cercam em textos e frases (ZHANG *et al.*, 2021). Há dois tipos de treinamento para *embeddings Word2Vec*: ***CBoW***, em que treina-se considerando as probabilidades condicionais de gerar uma palavra central numa frase baseada no contexto que a cerca (demais palavras) (Figura 12), e ***SkipGram***, em que o processo de treinamento se dá da maneira oposta, considerando as probabilidades condicionais de gerar contexto (demais palavras de uma frase) em torno de uma palavra central (Figura 13).

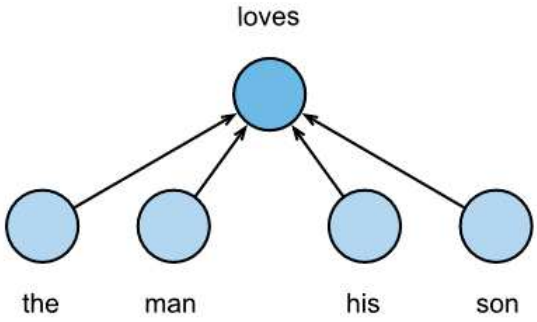
No geral, se usam camadas de *embeddings* com pesos pré-treinados no início de uma rede. *Word embeddings* pré-treinados em vários idiomas e algoritmos estão disponíveis em repositórios na internet, como o do Núcleo Interinstitucional de Linguística Computacional (NILC, 2017).

3.6 MECANISMO DE ATENÇÃO

Na psicologia, o termo atenção é utilizado para se referir à seletividade do processamento de informações (NETO, 2002). Para redes neurais, que em alguns aspectos se esforçam em emular o funcionamento do cérebro humano, o mecanismo de atenção é uma tentativa de implementar essa definição em modelos de *deep learning* com o objetivo de se concentrar mais em informação relevante e ignorar o resto.

Figura 12 – Treinamento de modelo *CBoW*.

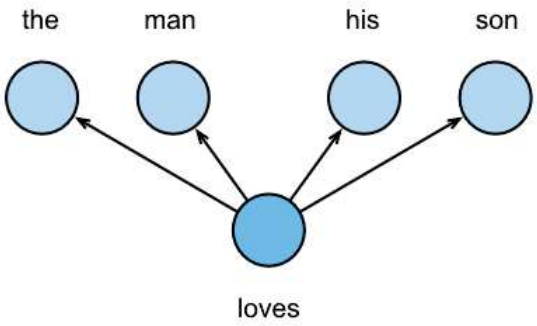
$$P(\text{"loves"} \mid \text{"the"}, \text{"man"}, \text{"his"}, \text{"son"}).$$



Fonte: Zhang *et al.* (2021)

Figura 13 – Treinamento de modelo *Skip-Gram*.

$$P(\text{"the"} \mid \text{"loves"}) \cdot P(\text{"man"} \mid \text{"loves"}) \cdot P(\text{"his"} \mid \text{"loves"}) \cdot P(\text{"son"} \mid \text{"loves"})$$



Fonte: Zhang *et al.* (2021)

O mecanismo de atenção surgiu como uma melhora no sistema de tradução automática baseado na arquitetura *encoder-decoder* para PLN. Depois, ele e suas variantes começaram a ser usados em outras aplicações, incluindo visão computacional e processamento de texto.

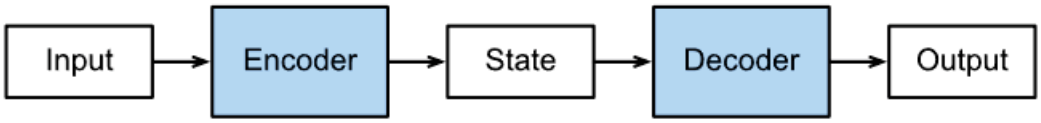
A seguir, veremos um resumo do que é a arquitetura *encoder-decoder* e algumas formas simples de mecanismo de atenção.

3.6.1 Arquitetura *encoder-decoder*

A arquitetura *encoder-decoder* (Figura 14) foi criada para aplicações que possuem entrada e saída de tamanhos variáveis, em especial tradução de textos (CHO

et al., 2014b). Esse tipo de modelo é formado por dois componentes principais (cada um deles sendo, por exemplo, uma RNN ou LSTM, já vistas em seções anteriores): o ***encoder***, que transforma uma sequência de tamanho variável (por exemplo, uma frase) num estado de tamanho fixo, geralmente chamado de *context variable/vector*, que, espera-se, será uma boa síntese dos dados de entrada; e o ***decoder***, que mapeia o estado de tamanho fixo numa sequência de tamanho variável.

Figura 14 – Arquitetura *encoder-decoder*.



Fonte: Zhang *et al.* (2021)

Tomaremos como exemplo um treinamento de rede para tradução de inglês para francês. Tendo como entrada a sequência (‘‘`They`’’, ‘‘`are`’’, ‘‘`watching`’’, ‘‘`.`’’), que em português seria (‘‘`Eles`’’, ‘‘`estão`’’, ‘‘`assistindo`’’, ‘‘`.`’’), a arquitetura *encoder-decoder* primeiro codifica esta sequência num estado de tamanho fixo, o *context vector*, e depois decodifica este estado numa sequência de saída traduzida (‘‘`Ils`’’, ‘‘`regardent`’’, ‘‘`.`’’), como mostrado na Figura 15.

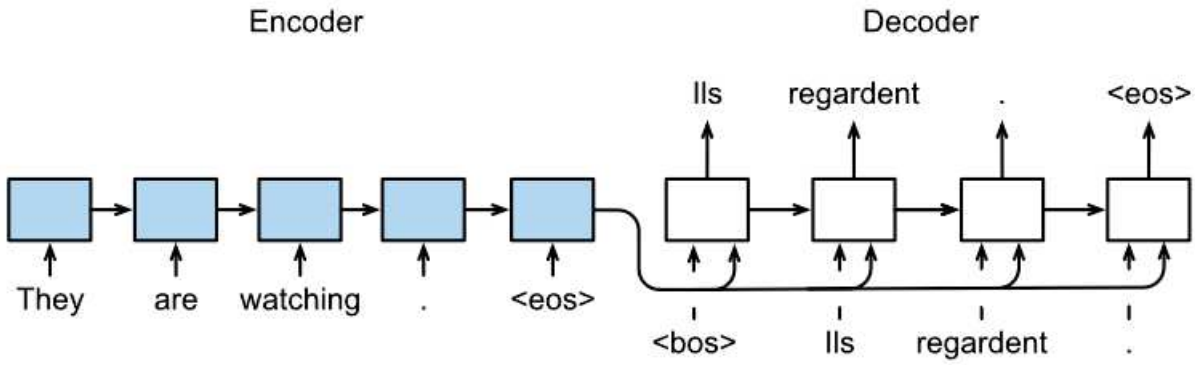
No exemplo mostrado, que usa uma RNN como *encoder* e outra como *decoder*, o *token* especial ‘‘`<eos>`’’ marca o fim de uma sequência e o ‘‘`<bos>`’’ indica o início de uma sequência. O *context vector* é o último *hidden state* da RNN *encoder* e é utilizado para iniciar o *hidden state* da RNN *decoder*. Em alguns casos, o *hidden state* do *encoder* pode também contar como parte das entradas do *decoder* em cada *step time* (CHO *et al.*, 2014b) - que na imagem são (‘‘`<bos>`’’, ‘‘`Ils`’’, ‘‘`regardent`’’, ‘‘`.`’’).

3.6.2 Atenção de Bahdanau

O problema de se usar a codificação de uma sequência num vetor de tamanho fixo é que o *decoder* tem um acesso limitado à informação de entrada. Isso é bastante problemático para sequências de entrada muito longas (especialmente se forem mais longas do que as sequências usadas no treinamento), que terão representações do mesmo tamanho que sequências curtas e mais simples. A performance de uma rede *encoder-decoder* básica se deteriora quanto maior o comprimento da entrada (CHO *et al.*, 2014a).

O mecanismo de atenção foi introduzido por Bahdanau, Cho e Bengio (2016) como uma tentativa de contornar essa questão. É uma maneira de procurar na sequência de entrada as concentrações de informação mais relevante.

Figura 15 – Treinamento de uma rede com RNNs de *encoder* e *decoder*.



Fonte: Zhang *et al.* (2021)

O diagrama de um exemplo de uso de atenção, presente no artigo original, está na Figura 16. Os x_n representam as entradas, os $\overrightarrow{h_n}$ e $\overleftarrow{h_n}$ são os *hidden states* (bidirecionais) da parte *encoder* e o *decoder* é formado pelos y_{t-1} e y_t, na imagem, com seus *hidden states* sendo s_{t-1} e s_t.

Obter a atenção neste caso se dá em três etapas. Primeiro há o cálculo dos escores de alinhamento (*alignment scores*, $e_{t,i}$) , que indicam o quão bem os elementos da sequência de entrada se alinham com os da saída. Podem ser representados pela expressão na Equação (5), onde a função $a()$ é chamada de modelo de alinhamento e foi implementada usando uma rede neural no artigo original.

$$e_{t,i} = a(\, s_{t-1}, h_i \,) \tag{5}$$

Depois, há o cálculo dos pesos $\alpha_{t,i}$, aplicando a função *softmax* nos escores de alinhamento, conforme Equação (6).

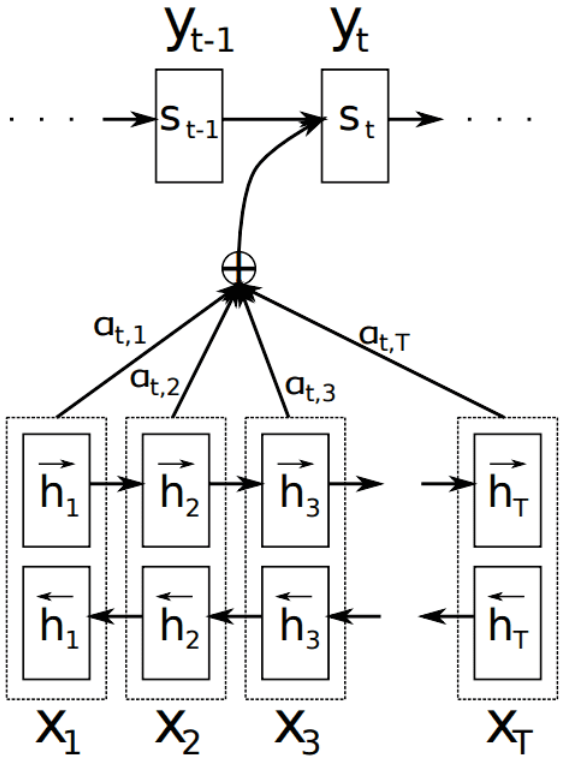
$$\alpha_{t,i} = softmax(\, e_{t,i} \,) \tag{6}$$

Por último, o cálculo do *context vector* c_t na Equação (7), que é a soma ponderada dos *hidden states* do *encoder* (*hidden states* totais, junção dos *hidden states* em ambas as direções, $\overrightarrow{h_T}$ e $\overleftarrow{h_T}$).

$$c_t = \sum_{i=1}^{T} \alpha_{t,i} \cdot h_i \tag{7}$$

Desta forma, em vez de o *context vector* ser apenas o último *hidden state* do *encoder*, como foi no exemplo na Figura 15, ele passa a ser formado por todos os *hidden states*.

Figura 16 – Exemplo de uso de atenção.



Fonte: Bahdanau, Cho e Bengio (2016)

Vimos a ideia básica por trás do mecanismo de atenção, mas ele pode ser generalizado para casos diferentes que não usam RNNs como *encoder* e *decoder*. Para isso, cabe aqui uma breve explicação do que são os vetores *query* Q, *key* K e *value* V. Eles são abstrações criadas para trabalhar com mecanismos de atenção. Cada palavra numa sequência de entrada numa rede tem seus próprios vetores *query*, *key* e *value*, que são gerados multiplicando a representação do *encoder* para a palavra em questão por três diferentes matrizes de pesos, ajustadas durante o treinamento do modelo.

Em essência, o mecanismo de atenção generalizado recebe uma sequência de palavras, usa o vetor *query* atribuído a cada palavra para calcular um *score* entre ele e os vetores *key* de todos os dados (conforme Equação (8)), capturando como cada palavra se relaciona com as demais na sequência. Então, como mostrado na Equação (10), calcula-se uma soma ponderada dos vetores *value* de acordo com os pesos da atenção (calculados através dos *scores* entre *queries* e *keys* na Equação (9)) para manter o foco em palavras mais relevantes. Assim, obtêm-se a saída de um bloco de atenção para cada palavra.

$$e_{q,k_i} = q \cdot k_i \tag{8}$$

$$\alpha_{q,k_i} = softmax(\ e_{q,k_i}\ ) \tag{9}$$

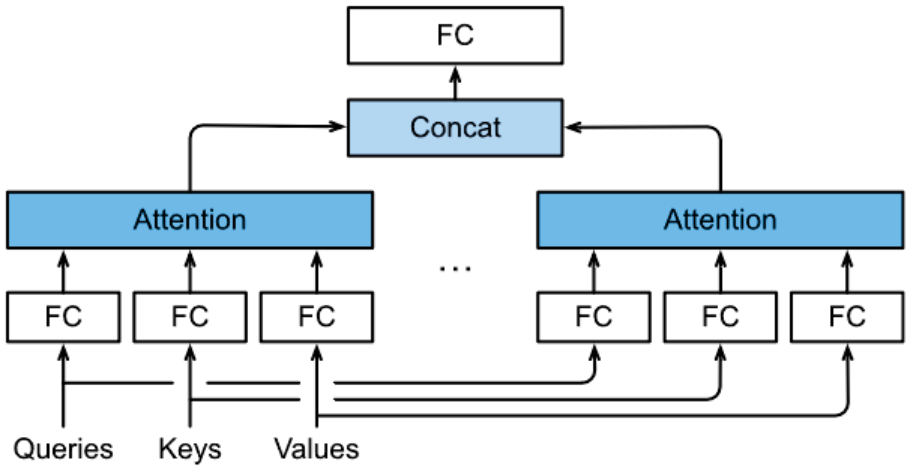
$$attention(q, K, V) = \sum_{i=1}^{T} \alpha_{q,k_i} \cdot v_{k_i} \tag{10}$$

3.6.3 Atenção *Multi-Head*

Na prática, gostaríamos que um modelo combinasse diferentes comportamentos relacionados a atenção, usando, por exemplo, dependências de palavras a longas e curtas distâncias umas das outras. Para fazer isso, usam-se subespaços de representação dos vetores *queries*, *keys* e *values* através de *n* projeções lineares independentes, que são então combinadas para produzir um *score* final de atenção. Esse esquema é chamado de atenção *multi-head*, onde cada projeção linear é chamada de *head* (VASWANI *et al.*, 2017).

Em resumo, atenção *multi-head* é formada por várias funções de atenção em paralelo que são combinadas para gerar a saída definitiva do bloco. Na Figura 17 vemos um esquema de atenção *multi-head*, com camadas *fully-connected* (FC) realizando as transformações lineares, aprendidas durante o treinamento.

Figura 17 – Esquema de atenção *multi-head*.



Fonte: Zhang *et al.* (2021)

Matematicamente, a atenção *multi-head* pode ser definida conforme a Equação (11), com as atenções individuas concatenadas e multiplicadas por W, uma matriz de parâmetros aprendidos.

$$multiHead(Q, K, V) = [head_1, head_2, ..., head_n]W \tag{11}$$

3.7 *TRANSFORMERS*

As redes neurais recorrentes têm algumas desvantagens. Elas têm dificuldade em lidar com sequências ou textos longos, muitas vezes "esquecendo" o início deles quando chegam no final. Elas também são difíceis de treinar, porque são suscetíveis ao desaparecimento/explosão de gradiente (um problema que ocorre durante o ajuste de pesos no treinamento) e processam palavras sequencialmente, sendo difícil paralelizar o treino.

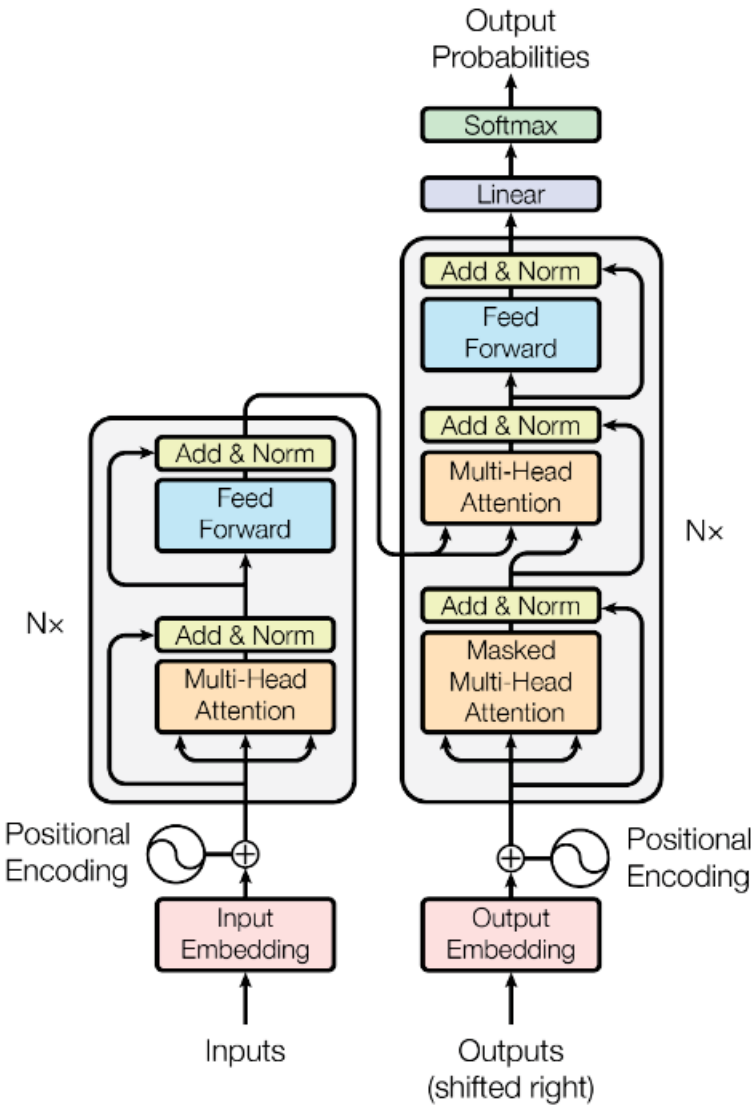
A arquitetura *transformer*, um tipo de rede *encoder-decoder*, surgiu como uma alternativa às RNNs e foi inicialmente pensada para a tarefa de tradução (VASWANI *et al.*, 2017). O principal objetivo desse tipo de rede era permitir uma paralelização maior por depender inteiramente de mecanismos de atenção, em vez de utilizar modelos recorrentes que são intrinsecamente sequenciais (SOUZA; NOGUEIRA; LOTUFO, 2019). O esquema da arquitetura apresentado no artigo original está na Figura 18.

Há duas grandes inovações na arquitetura *transformer*: ***positional encoding*** e ***self-attention*** (auto-atenção).

Positional encodings são a maneira pela qual as redes *transformers* escapam do problema de ter que aceitar dados sequencialmente, como as RNNs. Na linguagem natural, a ordem em que estão as palavras faz toda a diferença, por isso esta informação tem que estar disponível para a rede. A ideia do *positional encoding* é representar a posição absoluta dos *tokens* numa sequência. A maneira mais simples de fazer isso seria numerá-los em ordem com índices inteiros (1, 2, 3, ...). No artigo original de Vaswani *et al.* (2017) usam-se funções sinusoidais para criar *positional encodings* vetoriais, contudo o princípio é o mesmo. A rede aprende a interpretar os *positional encodings* ao longo do treinamento.

A segunda inovação citada foi a *self-attention*. Numa explicação resumida, é usar a atenção *multi-head* já vista na Seção 3.6.3 (que inclusive foi proposta no mesmo artigo da arquitetura *transformer*) com os *queries*, *keys* e *values* referentes a *uma mesma sequência de tokens*. Ou seja, a auto-atenção é o mecanismo de atenção aplicado nas sequências de entrada individualmente; é usada para relacionar cada palavra de uma sentença com as demais, comparando-as na tentativa de "compreender o significado" de cada uma com base em seu contexto. Esse mecanismo, destacado na Figura 19, permite que durante o treinamento o modelo analise as estruturas das sequências de entrada uma a uma e "entenda" padrões, a ponto de construir uma representação interna do idioma em que foi treinado, automaticamente aprendendo regras de gramática ou quais palavras são sinônimos, por exemplo (MARKOWITZ, 2021). Com essa modelagem de um idioma, é possível usar a rede treinada ou partes

Figura 18 – Arquitetura *transformer*.



Fonte: Vaswani *et al.* (2017)

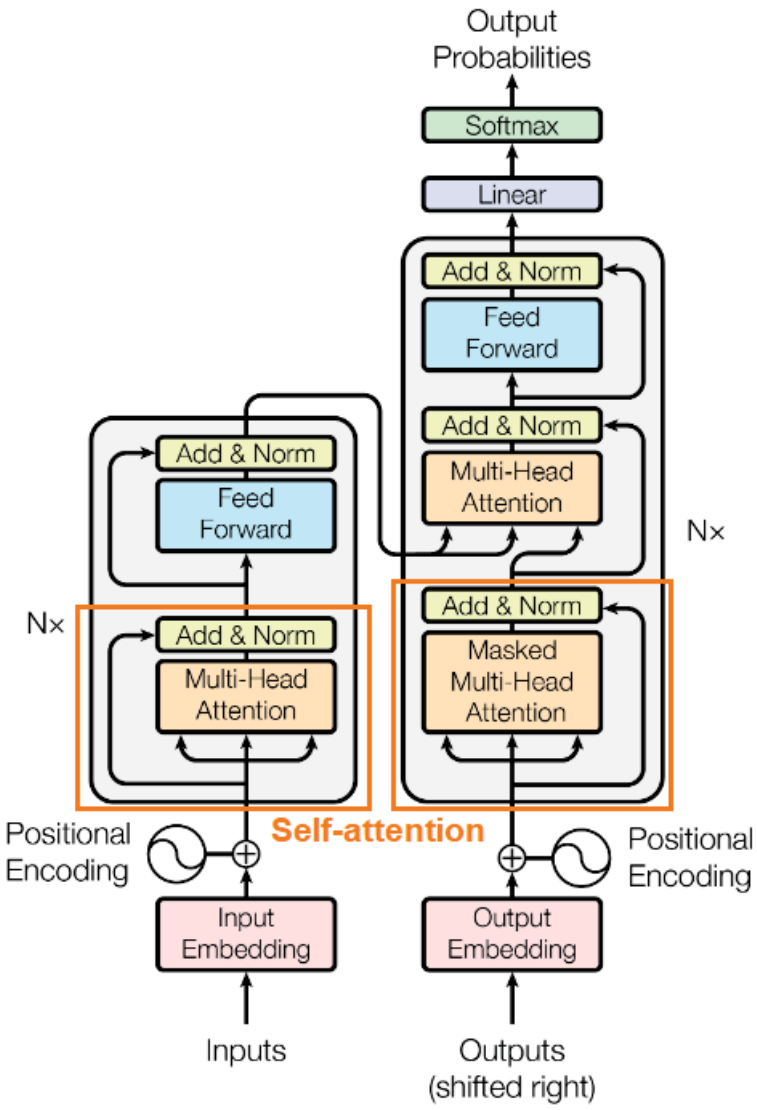
dela em várias tarefas diferentes. Quanto melhor a representação interna da linguagem aprendida pela rede, melhor ela se sairá nas tarefas de PLN.

3.7.1 ***Bidirectional Encoder Representation from Transformers*** **(BERT)**

O BERT é um tipo de *transformer* que usa apenas a parte do *encoder* da arquitetura para gerar um modelo da linguagem. Esse modelo é aprendido durante o pré-treinamento com grande quantidade de textos no idioma desejado.

A entrada desta rede é formada por três tipos de *embeddings*: os *token embeddings* (com a sequência de entrada dividida em *tokens*), os *segment embeddings* (que indicam a que frase cada *token* pertence) e os *position embeddings* (resultados do

Figura 19 – Arquitetura *transformer* - blocos de *self-attention*.



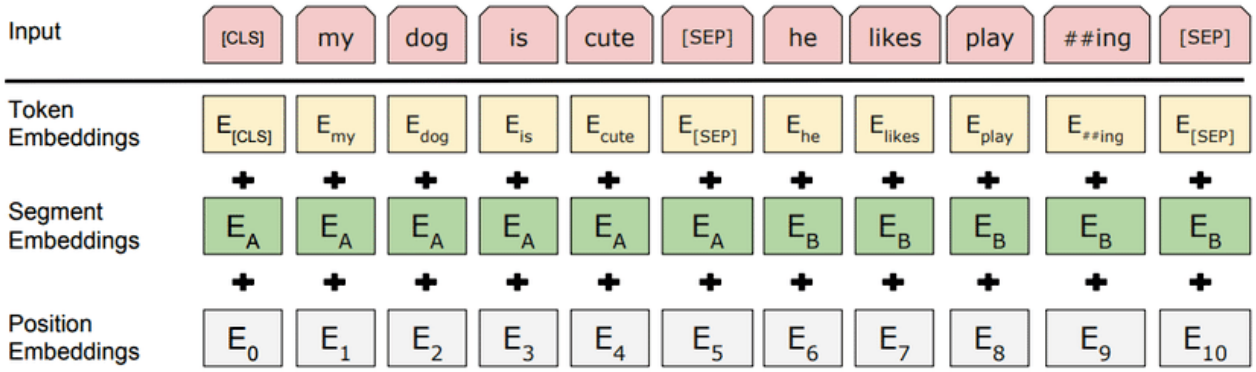
Adaptada de: Vaswani *et al.* (2017)

positional encoding, indicam a ordem dos *tokens* no texto), conforme Figura 20.

Uma rede BERT é pré-treinada em duas tarefas, *Masked-Language Modeling* (MLM) e *Next Sentence Prediction* (NSP). No MLM, 15% dos *tokens* são escolhidos aleatoriamente antes das sequências de texto serem usadas como entrada da rede no pré-treino; destes, 80% são substituídos pelo *token* `[MASK]`, 10% são substituídos por *tokens* aleatórios e 10% não são alterados. (DEVLIN *et al.*, 2018). O modelo então tenta prever os *tokens* originais com base no contexto provindo dos demais. Assim, ao longo de várias iterações de treino o modelo lentamente aprende como funciona o contexto no idioma utilizado.

Na tarefa de NSP, o modelo recebe pares de sentenças como entrada e aprende a inferir se a segunda sentença é subsequente à primeira no texto original. No pré-

Figura 20 – Representação da entrada de uma rede BERT.



Fonte: Devlin *et al.* (2018)

treinamento, 50% dos pares de entrada são de frases subsequentes uma a outra num texto e os outros 50% têm sua segunda frase escolhida aleatoriamente no *corpus* usado (assume-se que ela será desconectada da primeira).

Depois do pré-treino, o ajuste fino é realizado levando em conta a tarefa final para que o modelo será utilizado (Figura 21), como NER no caso deste trabalho. Comparado ao pré-treinamento, que é bastante custoso e pode levar mais de uma semana, o ajuste fino é pouco dispendioso e pode ser realizado em poucas horas numa TPU ou GPU. Para tarefas que não podem ser facilmente representadas por uma arquitetura *encoder transformer*, pode-se usar uma abordagem *feature based*, adicionando modelos específicos ao BERT e mantendo seus parâmetros pré-treinados (DEVLIN *et al.*, 2018).

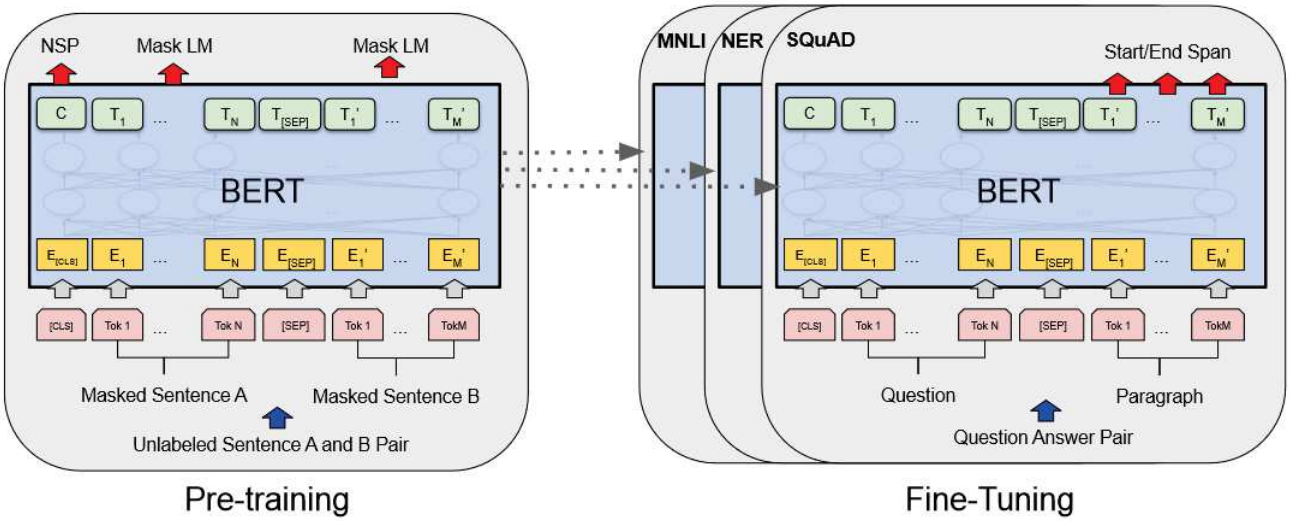
Dois modelos de BERT podem ser usados como base: **BERT *base***, que possui 12 camadas/*encoders*, 12 *heads* de atenção e 110 milhões de parâmetros, e **BERT *large***, com 24 camadas/*encoders*, 16 *heads* de atenção e 340 milhões de parâmetros (DEVLIN *et al.*, 2018).

3.8 *CONDITIONAL RANDOM FIELD* (CRF)

CRF, ou Campo Aleatório Condicional, é uma classe de modelos estatísticos geralmente aplicados em reconhecimento de padrões e ML, em especial tarefas de classificação de dados sequenciais, como NER. Foram introduzidos por Lafferty, McCallum e Pereira (2001), mas são utilizados até hoje em modelos estado-da-arte.

Usando como exemplo a tarefa de NER, enquanto uma camada de classificação comum faz a predição de *tags* para cada *token* independentemente, uma camada de CRF leva em conta o contexto, considerando os *tokens* vizinhos (STIEGLER, 2021). Para fazer isso, o modelo usa um grafo para expressar as dependências condicionais

Figura 21 – Pré-treinamento e ajuste fino de uma rede BERT.



Fonte: Devlin *et al.* (2018)

entre variáveis (o que é chamado de modelo probabilístico gráfico ou estruturado). O tipo de grafo utilizado depende da aplicação; para PLN é habitual o uso de CRF de cadeia linear (*linear chain*), onde as predições dependem apenas dos vizinhos imediatos.

Na Figura 22 há esquemas mostrando uma rede LSTM bidirecional sem e com CRF na saída. Vê-se a diferença já citada: em um caso as previsões de entidades são feitas de forma individual, no outro usam-se probabilidades envolvendo as palavras vizinhas para inferir qual a sequência de *tags* de saída. Apesar do exemplo mostrado ser com uma rede LSTM, camadas de CRF podem ser usadas com outros modelos, inclusive BERT.

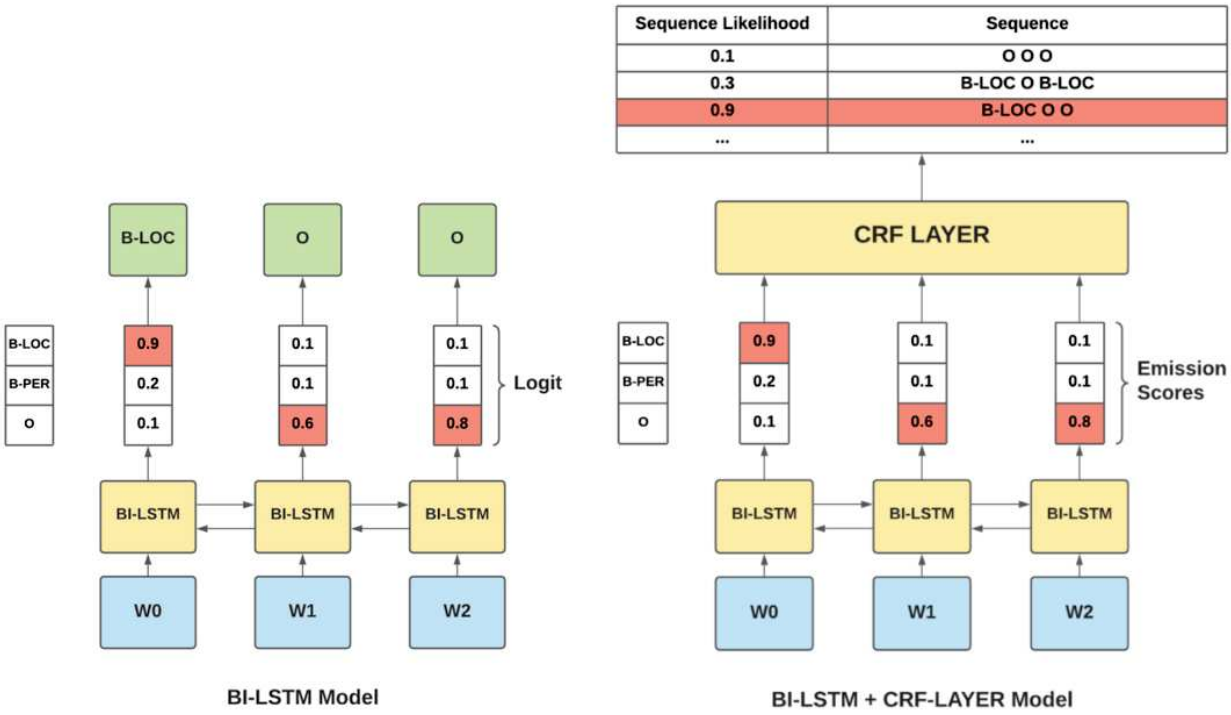
O principal parâmetro de uma camada CRF é a chamada **matriz de probabilidades de transições**. Ela é formada pelos *scores* de transição, que são as probabilidades condicionais de possíveis próximos estados (no caso de NER, as *tags*) dado o estado atual (LAFFERTY; MCCALLUM; PEREIRA, 2001). Essa matriz é o que será ajustado durante o treinamento da rede.

3.9 MÉTRICAS DE AVALIAÇÃO

Para avaliar a qualidade de um modelo NER, é comum comparar a saída inferida por ele com entidades de referência que provêm de exemplos anotados por humanos. Diferentes tipos de métricas são usados para quantificar esse desempenho.

Neste projeto, foram usadas três diferentes métricas: precisão, *recall* e *F1-score*. As expressões matemáticas usadas para calcular cada uma delas dependem de três

Figura 22 – Rede LSTM sem e com camada CRF.



Adaptada de: Stiegler (2021)

variáveis. No contexto de REN, são definidas como:

- ***True Positive* (*TP*):** Uma entidade que é corretamente localizada e identificada.
- ***False Positive* (*FP*):** Um termo que não é de determinada entidade, porém é rotulado como tal.
- ***False Negative* (*FN*):** Um termo que é de determinada entidade, porém não é rotulado como tal.

A seguir, as três métricas são definidas.

3.9.1 Precisão

Precisão é a porcentagem de entidades nomeadas encontradas pelo algoritmo que estão corretas. É definida como a razão entre entidades corretamente localizadas e identificadas e todas as entidades (de uma mesma classe) inferidas pelo modelo, conforme Equação (12).

$$\text{Precisão} = \frac{TP}{TP + FP} \tag{12}$$

3.9.2 *Recall*

Recall é a porcentagem de entidades nomeadas encontradas pelo algoritmo. É definida como a razão entre entidades corretamente localizadas e identificadas pelo modelo e todas as entidades (novamente, de uma mesma classe) de referência (tanto as que o algoritmo classificou corretamente quanto as que não). A expressão matemática desta métrica é a Equação (13).

$$\textit{Recall} = \frac{TP}{TP + FN} \tag{13}$$

3.9.3 *F1-score*

Idealmente, deseja-se que um modelo NER tenha valores altos tanto de precisão quanto de *recall*, isto é, que o algoritmo identifique o máximo das entidades de referência e apenas elas. No entanto, às vezes o aumento de uma destas métricas é feito às custas da diminuição da outra.

Na busca de um bom equilíbrio entre as duas métricas, usa-se o valor de *F1-score*, que combina ambas e que funciona bem com classes desbalanceadas (diferentes números de entidades de cada tipo) (KORSTANJE, 2021).

F1-score é uma média harmônica entre precisão e *recall*. É definida pela Equação (14).

$$\textit{F1-score} = \frac{2 \cdot \text{Precisão} \cdot \textit{Recall}}{\text{Precisão} + \textit{Recall}} \tag{14}$$

F1-score é um caso específico de *F-score*, ou F_β *-score* (Equação (15)), que usa um fator real positivo β, escolhido de forma que o *recall* é considerado β vezes mais importante que a precisão.

$$F_\beta\textit{-score} = \frac{(\beta^2 + 1) \cdot \text{Precisão} \cdot \textit{Recall}}{\beta^2 \cdot \text{Precisão} + \textit{Recall}} \tag{15}$$

3.9.4 Métricas *micro-average*

Para calcular as métricas médias de diversas classes de entidade, usou-se a abordagem *micro-average*.

Isso significa que a precisão geral de um modelo será a soma dos *TP*s de todas as *n* classes dividida pela soma de todos os *TP*s e *FP*s, conforme Equação (16).

$$\textit{Micro-average}\ \text{precisão} = \frac{TP_1 + TP_2 + ... + TP_n}{TP_1 + TP_2 + ... + TP_n + FP_1 + FP_2 + ... + FP_n} \tag{16}$$

Já o *recall* geral, seguindo os mesmos moldes, será calculado como mostrado na Equação (17).

$$\textit{Micro-average recall} = \frac{TP_1 + TP_2 + ... + TP_n}{TP_1 + TP_2 + ... + TP_n + FN_1 + FN_2 + ... + FN_n} \quad (17)$$

O cálculo do *F1-score* geral de um modelo mantém a mesma equação já apresentada, mas usando a média *micro-average* de precisão e *recall*.

4 TRABALHOS RELACIONADOS

Neste capítulo, serão brevemente apresentados outros trabalhos com temas relacionados, alguns dos quais foram usados como referências diretas para este. Os resultados numéricos específicos que forem pertinentes para comparação com os deste trabalho serão apresentados no Capítulo 6.

4.1 *LENER-BR: A DATASET FOR NAMED ENTITY RECOGNITION IN BRAZILIAN LEGAL TEXT*

Este trabalho, *LeNER-Br: a Dataset for Named Entity Recognition in Brazilian Legal Text* (LUZ DE ARAUJO *et al.*, 2018), foi o que introduziu o *dataset* de textos legais em português brasileiro *LeNER-Br*, formado por 70 textos diferentes divididos entre treino, validação e teste. Nele também é treinado um modelo de rede LSTM-CRF bidirecional, comumente chamado BiLSTM (BiLSTM-CRF), que serve como *baseline* .

As entidades especificamente jurídicas criadas para este *dataset* foram LEGISLAÇÃO e JURISPRUDÊNCIA.

Os parâmetros de rede usados no trabalho estão na Tabela 1.

Tabela 1 – Parâmetros de treinamento de rede LSTM-CRF de Luz de Araujo *et al.* (2018).

Parâmetros	**Valores**
Dimensão do *word embedding*	300
Dimensão do *character embedding*	50
Número de épocas	55
Dropout rate	0,5
Tamanho do *batch*	10
Função de otimização	SGD
Taxa de aprendizado	0,015
Decaimento da taxa de aprendizado	0,95
Número de *hidden units* na primeira camada da rede	25
Número de *hidden units* na segunda camada da rede	100

Fonte: Luz de Araujo *et al.* (2018).

4.2 *PORTUGUESE NAMED ENTITY RECOGNITION USING BERT-CRF*

O trabalho de Souza, Nogueira e Lotufo (2019) entregou um modelo BERT pré-treinado em português brasileiro, nomeado *BERTimbau*.

Este modelo foi pré-treinado com o maior *corpus* aberto em português: *brWaC* (WAGNER FILHO *et al.*, 2018), formado por textos variados obtidos da internet, contendo bilhões de *tokens* e milhões de textos.

O modelo *BERTimbau*, pré-treinado no *brWaC* e treinado para ajuste fino no *dataset HAREM* (proposto pela Linguateca (2013) e usado como padrão para avaliar os modelos NER em português brasileiro em trabalhos acadêmicos), obteve resultados que superaram os modelos estado-da-arte no idioma.

4.3 *BERTBR: A PRETRAINED LANGUAGE MODEL FOR LAW TEXTS*

Na monografia, Ciurlino (2021) usa o modelo BERT-CRF pré-treinado por Souza, Nogueira e Lotufo (2019) e realiza nele mais pré-treinamento, desta vez com textos jurídicos do *Iudicium Textum Dataset* proposto por Willian Sousa e Fabro (2019), que contém 50.928 documentos dos anos de 2018 a 2020. A performance final dele foi obtida treinando-o e testando-o com o *dataset LeNER*.

O melhor modelo pré-treinado por Ciurlino (2021) no domínio específico obteve 95,25% de *F1-score*, mas os cálculos de métricas consideram acertos parciais, diferente deste trabalho (conforme será visto na Seção 5.6), por isso os valores de métricas não podem ser comparados diretamente com os obtidos aqui. Além disso, esse modelo, *BertBR*, não foi encontrado em nenhum repositório, por isso seu uso em experimentos neste projeto não foi possível.

4.4 RECONHECIMENTO DE ENTIDADES NOMEADAS EM TEXTOS DE BOLETINS DE OCORRÊNCIAS

Outro trabalho relacionado é o de Araújo (2019) que é voltado para análise de boletins de ocorrência. Além de classes padrão (como ORGANIZAÇÃO e LOCAL), esse modelo busca entidades relacionadas ao tipo de texto alvo, como VÍTIMA, ASSASSINO e TIPO DE ARMA.

O tipo de modelo escolhido para fazer a tarefa de NER neste trabalho foi BiLSTM-CRF. Ele também propõe uma estrutura interativa chamada *Human Named Entity Recognition with Deep Learning* (HNERD) com o foco de tornar eficiente a extração de entidades nomeadas em boletins de ocorrências. Este *framework* foi inclusive usado em outros trabalhos, como os de Oliveira (2020) e Chaves (2021), que trataram de contextos de narrativas de roubos e textos criminais.

4.5 RECONHECIMENTO DE ENTIDADES NOMEADAS PARA O PORTUGUÊS USANDO REDES NEURAIS

dos Santos Neto (2019) fez um trabalho em que se utilizam modelos de redes neurais em diversos domínios específicos de textos-alvo, fazendo uso de dados clínicos, policias e geológicos (como o *dataset* da conferência de avaliação *Iberian*

Languages Evaluation Forum (IberLEF 2019) e o GeoCorpus), assim como do *HAREM*.

Além disso, ele usou variados modelos, com diferentes *embeddings* e conjuntos deles, em combinação com a arquitetura BiLSTM-CRF.

5 MATERIAIS E MÉTODOS

Neste capítulo, são citadas as principais ferramentas utilizadas na implementação do projeto, além de explicar, de maneira geral, os processos realizados para a obtenção dos modelos.

5.1 *DATASET*

O *dataset* de textos legais em português brasileiro *LeNER-Br* foi utilizado para o treinamento de modelos *deep learning* neste trabalho. Ele está disponível no repositório *git* de Luz de Araujo (2020).

O *LeNER-Br*, apresentado por Luz de Araujo *et al.* (2018), é formado por 70 textos jurídicos manualmente anotados, sendo 66 documentos legais de diversas cortes brasileiras e 4 documentos de legislações. As *tags*/classes usadas nele são:

- **ORGANIZAÇÃO** para nomes de organizações;
- **PESSOA** para nomes próprios de pessoas;
- **TEMPO** para entidades de tempo;
- **LOCAL** para localizações;
- **LEGISLAÇÃO** para leis;
- **JURISPRUDÊNCIA** para decisões legais.

O esquema de *tagging* usado no *dataset* é IOB2 (apresentado na Seção 3.1.2).

Os textos são aleatoriamente divididos entre os três conjuntos, treino, validação e teste, conforme a Tabela 2. A contagem de palavras pertencentes a cada classe de entidade é mostrada na Tabela 3.

Tabela 2 – Sentenças, *tokens* e contagem de documentos para cada conjunto do *dataset LeNER-Br*.

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

Fonte: Luz de Araujo *et al.* (2018).

Tabela 3 – Contagem de palavras de cada classe de entidade nomeada nos conjuntos do *dataset LeNER-Br*.

Classe	Conjunto de treino	Conjunto de validação	Conjunto de teste
PESSOA	4.612	894	735
JURISPRUDÊNCIA	3.967	743	660
TEMPO	2.343	543	260
LOCAL	1.417	244	132
LEGISLAÇÃO	13.039	2.609	2.669
ORGANIZAÇÃO	6.671	1.608	1.367

Fonte: Luz de Araujo *et al.* (2018).

5.2 BIBLIOTECAS

Python foi eleita a linguagem de programação a ser usada no projeto. Ela é a mais popular para uso em IA e ML (LUASHCHUK, 2019), e uma das principais razões para isso é a vasta gama de bibliotecas (módulos com funções em código pré-escrito que permitem que o usuário obtenha determinadas funcionalidades) para ela disponíveis. Como em IA e ML é necessário acessar e processar dados, criar e treinar algoritmos e fazer inferências, existem ferramentas prontas para auxiliar o programador nestas tarefas. Em destaque, duas bibliotecas são citadas aqui como essenciais para este trabalho: *NumPy* e *PyTorch*.

NumPy é uma biblioteca numérica e *open-source* em Python. É uma das mais poderosas e otimizadas para cálculos que envolvem *arrays* multidimensionais (MALIK, 2019). Ela disponibiliza diversas operações matemáticas, incluindo funções trigonométricas, algébricas, estatísticas e transformações.

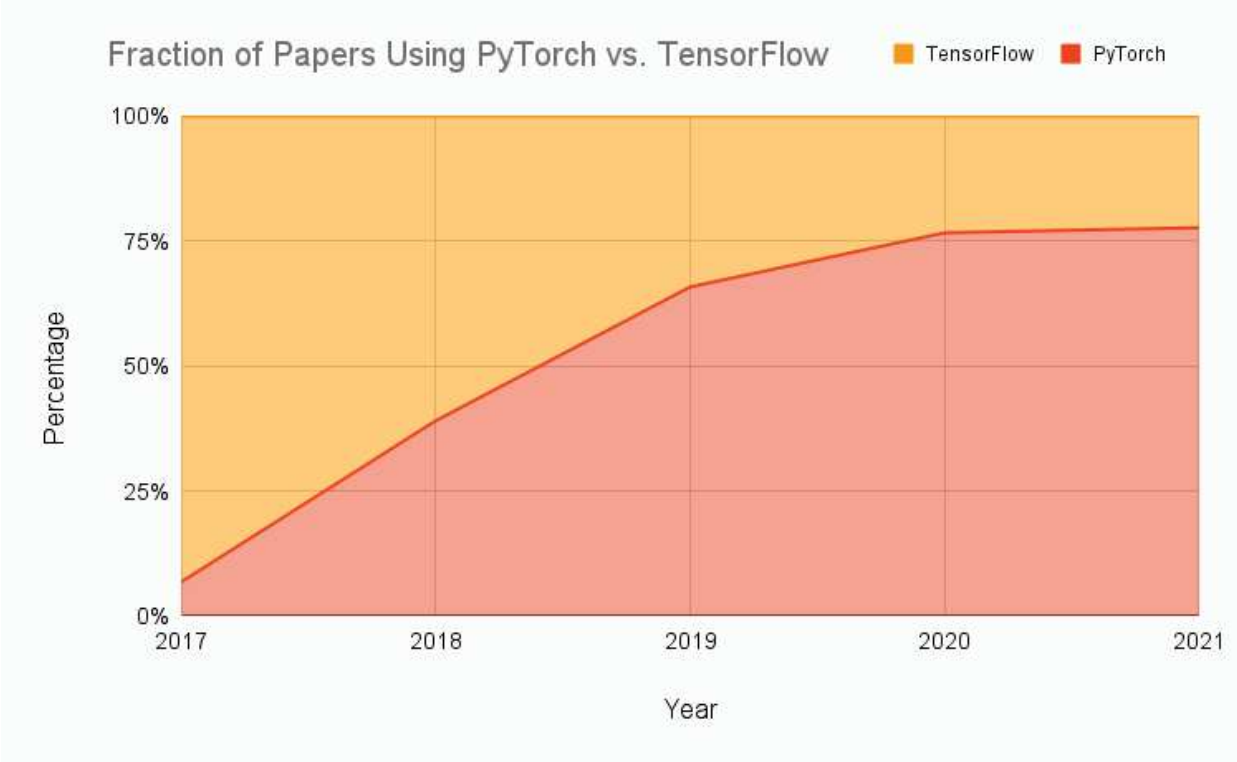
Como *framework* de *deep learning*, entre os dois principais, *TensorFlow* e *PyTorch*, escolheu-se usar o segundo.

PyTorch é baseado em *Torch*, um *framework* implementado em C e dedicado a computação científica. Segundo Johns (2020) e O'Connor (2021), há algumas vantagens em utilizá-lo em detrimento de outros, como:

- Melhor uso de memória e otimização;
- Melhor compatibilidade com *NumPy*;
- Mais controle sobre as estruturas de modelos;
- Estilo orientado a objeto, o que torna a implementação e alteração de modelos mais ágeis e o código mais modular;
- Todos os modelos estado-da-arte atualmente têm versões em *PyTorch*;

- A maioria das publicações de pesquisa é feita com este *framework*, como pode ser visto na Figura 23, conforme levantamento de dados realizado por O'Connor (2021).

Figura 23 – Proporção de publicações que usam *TensorFlow* e *PyTorch* ao passar dos anos.



Fonte: O'Connor (2021).

5.3 FORMATOS DE ARQUIVOS

Os algoritmos para treinamento das redes neurais usados neste trabalho são baseados nos apresentados por Souza, Nogueira e Lotufo (2019) (disponível no repositório de NeuralMind (2019)) e Wang (2020) para BERT e LSTM, respectivamente. Como formato de texto de entrada para treino de cada um desses tipos de rede, foram usados JSON e CoNLL.

O formato JSON, utilizado no algoritmo BERT de NeuralMind (2019), é padronizado para então passar por etapas de pré-processamento, onde cada texto é divido em sentenças, *tokenizado* e convertido em *embeddings*, conforme já visto no Capítulo 3. Esse formato traz todas as informações necessárias sobre cada texto e suas entidades. Um trecho de exemplo dele pode ser visto na Figura 24. Ele é formado por três campos principais: o ID do documento, a *string* do texto completo e um conjunto de entidades.

Este último campo contém outros cinco: o ID da entidade (um índice/número inteiro), a classe (ou tipo) da entidade, o caractere no texto em que se inicia a entidade, o caractere no texto em que termina a entidade e o recorte que indica as palavras no texto que formam a entidade completa.

Figura 24 – Trecho de texto em formato JSON.


```
doc_id : INSTRUCAOON06043378120186000000
doc_text :     TRIBUNAL SUPERIOR
          ELEITORAL\nRESOLUÇÃO Nº 23.561\nINSTRUÇÃO
          Nº 0604337-81.2018.6.00.0000 - CLASSE 19
          - BRASÍLIA -\nDISTRITO FEDERAL\nRelator:
          Ministro Luiz Fux ...
▼ entities [31]
   ▼ 0 {5}
          entity_id : 1787
          text : TRIBUNAL SUPERIOR ELEITORAL
          label : ORGANIZACAO
          start_offset : 4
          end_offset : 31
   ▼ 1 {5}
          entity_id : 1788
          text : INSTRUÇÃO Nº 0604337-81.2018.6.00.0000
          label : LEGISLACAO
          start_offset : 52
          end_offset : 90
   ▼ 2 {5}
          entity_id : 1789
          text : BRASÍLIA -\nDISTRITO FEDERAL
          label : LOCAL
          start_offset : 105
          end_offset : 132
   ▼ 3 {5}
          entity_id : 1790
          text : Luiz Fux
          label : PESSOA
          start_offset : 151
          end_offset : 159
   ...
```


Fonte: Arquivo pessoal.

Já o formato CoNLL para NER, que é usado na conferência anual de mesmo

nome e no algoritmo de Wang (2020), consiste basicamente de um arquivo de texto com duas colunas. Na primeira estão as palavras e pontuações que formam o texto e na segunda estão as *tags*, ou classes, associadas a cada uma delas. Um exemplo é mostrado na Figura 25.

Figura 25 – Trecho de texto em formato CoNLL.

```
TRIBUNAL B-ORGANIZACAO
SUPERIOR I-ORGANIZACAO
ELEITORAL I-ORGANIZACAO
RESOLUÇÃO O
N° O
23.561 O
INSTRUÇÃO B-LEGISLACAO
N° I-LEGISLACAO
0604337-81.2018.6.00.0000 I-LEGISLACAO
- O
CLASSE O
19 O
- O
BRASÍLIA B-LOCAL
- I-LOCAL
DISTRITO I-LOCAL
FEDERAL I-LOCAL
Relator O
: O
Ministro O
Luiz B-PESSOA
Fux I-PESSOA
...
```

Fonte: Arquivo pessoal.

Outro detalhe de pré-processamento do *dataset* disponibilizado por Luz de Araujo (2020) é que há uma diferença de representação das aspas nos arquivos de *raw text* e nos do formato CoNLL [1]. Nos primeiros, as aspas duplas eram representadas pelo sinal `"` , presente nos teclados padrão ABNT e ABNT2; nos outros, as aspas usadas eram `“ ”`. Como escolha de projeto, levando em conta que o sinal `"` é usado tanto como aspas direitas quanto esquerdas e portanto não as diferencia, optou-se por usar apenas os sinais `“ ”` nos textos em ambos os formatos, JSON e CoNLL.

5.4 MODELOS PRÉ-TREINADOS

Os modelos BERT pré-treinados (*base* e *large*, vistos na Seção 3.7.1) resultados do artigo de Souza, Nogueira e Lotufo (2019) foram usados neste projeto (disponível

[1] Os arquivos em formato JSON não existem no diretório do *dataset* de Luz de Araujo (2020). Eles foram criados para este projeto especificamente.

no repositório NeuralMind (2019)).

O pré-treino deste tipo de arquitetura se beneficia de grandes quantidades de dados não rotulados, de preferência separados em documentos e não apenas em sentenças, porque assim o modelo de linguagem aprende a identificar dependências de longo alcance. Como já citado, o *dataset* utilizado para isso por Souza, Nogueira e Lotufo (2019) foi o *Brazilian Web as Corpus* (*brWaC*), em português brasileiro, que contém 2,68 bilhões de *tokens* de 3,53 milhões de documentos obtidos na internet (WAGNER FILHO *et al.*, 2018). Além do tamanho, o *brWaC* contém documentos completos e sua metodologia garante grande diversidade e qualidade de conteúdo. O pré-treinamento foi realizado usando apenas o corpo dos textos, ignorando títulos, e houve um passo de pré-processamento nos dados, a remoção de *mojibakes* (corrupções de textos que ocorrem quando *strings* são decodificadas usando a codificação de caracteres incorreta) e de *tags* HTML remanescentes.

Como *hardware* para esta etapa, Souza, Nogueira e Lotufo (2019) usaram o processador TPUv3-8 durante um milhão de passos. Os tempos de pré-treinamento, conforme relatado por Souza, Nogueira e Lotufo (2019), foram 4 dias para BERT *base* e 7 dias para BERT *large*.

Para os modelos LSTM, não houve pré-treinamento.

5.5 PARÂMETROS E *EMBEDDINGS*

Os parâmetros para LSTM utilizados neste trabalho foram escolhidos com base nos de Luz de Araujo *et al.* (2018). Eles são apresentados na Tabela 4. Para as redes BERT, os parâmetros são baseados nos de Souza, Nogueira e Lotufo (2019) e são mostrados na Tabela 5.

Tabela 4 – Parâmetros base para treinamento de redes LSTM e LSTM-CRF neste trabalho.

Parâmetros	**Valores**
Dimensão do *word embedding*	100 e 300
Dimensão do *character embedding*	30
Número de épocas	100
Paciência	15
Dropout rate	0,5
Tamanho do *batch*	10
Função de otimização	SGD
Taxa de aprendizado	0,015
Decaimento da taxa de aprendizado	0,95
Número de *hidden units* na primeira camada da rede	50
Número de *hidden units* na segunda camada da rede	100

Fonte: Arquivo pessoal.

Os *embeddings* selecionados para serem usados no treinamento das redes LSTM foram os disponibilizados pelo NILC no seu repositório (NILC, 2017). Este núcleo surgiu em 1993 e hoje reúne cientistas da computação, linguistas e outros pesquisadores de PLN de várias universidades brasileiras. O repositório em questão traz *embeddings* gerados a partir de um grande *corpus* de português brasileiro e europeu formado por 17 *corpora* diferentes, de fontes e gêneros variados, com o número total de *tokens* de 1.395.926.282. Mais detalhes sobre os *corpora* estão no quadro do Apêndice A.

Tabela 5 – Parâmetros base para treinamento de redes BERT e BERT-CRF neste trabalho.

Parâmetros	**Valores**
Sequência de entrada máxima (*tokens*) [1]	512
Stride [2]	128
Tamanho do *batch* por dispositivo (GPU/CPU)	16
Gradient accumulation steps	1
Taxa de aprendizado (BERT)	5e-5
Taxa de aprendizado (*layers* de classificação e CRF)	1e-3
Função de otimização	AdamW
β [3]	(0,9, 0,999)
ϵ [3]	1e-8
weight decay [3]	0,01
Número de épocas	15 (com CRF), 50 (sem CRF)

[1] Sequências de entrada mais longas que este valor serão separadas em múltiplos fragmentos. Sequências menores terão o espaço excedente preenchido com *tokens* "em branco".
[2] Quando um texto é separado em fragmentos, este valor é o "passo" adotado entre a obtenção de cada fragmento.
[3] Parâmetro da função de otimização.

Fonte: Arquivo pessoal.

5.6 AVALIAÇÃO DE RESULTADOS

Para realizar a avaliação dos resultados, escolheu-se usar o método da tarefa compartilhada CoNLL-2003 (TJONG KIM SANG; DE MEULDER, 2003). As métricas utilizadas no *script* são as mesmas apresentadas na Seção 3.9.

Um detalhe importante da avaliação é que uma entidade nomeada só é considerada corretamente classificada pelo modelo quando ele reconhece a combinação exata de *tokens* que a formam (i.e. as métricas são baseadas no chamado *exact match* ou *exact span* de entidades). Isso significa que reconhecimentos parciais não contam como acertos. Isso é mostrado nos exemplos na Figura 26.

Figura 26 – Exemplo de entidade corretamente (*exact match*) e não corretamente (*partial match*) classificada .

Fonte: Arquivo pessoal.

5.7 *HARDWARE*

Para treinamento e teste das redes neurais, uma máquina com as especificações apresentadas no Quadro 1 foi usada. A GPU desse computador foi especialmente utilizada para o treinamento dos modelos. Os testes foram, em sua maioria, realizados com uso de CPU, mas este detalhe não afeta os resultados, levando em conta as métricas escolhidas.

Quadro 1 – Especificações do *hardware* usado.

GPU	GeForce RTX 2080
CPU	AMD Ryzen Threadripper 2920X 12-Core
Sistema Operacional	Ubuntu 18.04
Memória RAM	64 GiB

Fonte: Arquivo pessoal.

6 RESULTADOS

6.1 *BIDIRECTIONAL LONG SHORT-TERM MEMORY* (BILSTM)

O primeiro experimento realizado foi treinar redes BiLSTM com e sem a camada CRF. Para isso, escolheu-se um mesmo tipo de *embedding* para ambos os cenários: *Wang2Vec* (criado como uma variação do já citado *Word2Vec*) de dimensão 100 e algoritmo de treinamento *SkipGram*. A comparação de métricas obtidas está na Tabela 6[2]. Como o esperado, o uso da uma camada CRF influencia nos modelos, resultando numa melhora significativa nas inferências.

Tabela 6 – Comparação entre BiLSTM e BiLSTM-CRF.

Modelo	**Precisão**	***Recall***	***F1-score***
BiLSTM	75,90%	82,23%	78,94%
BiLSTM-CRF	87,82%	86,85%	87,33%

Fonte: Arquivo pessoal.

O próximo experimento foi variar os tipos de *embeddings* usados, bem como as dimensões deles. Com a rede de base contendo a camada CRF na saída, dois tipos de *embeddings* foram escolhidos com base em resultados obtidos por Hartmann *et al.* (2017): *Wang2Vec* (com dois tipos de algoritmo de treinamento, *SkipGram* e *CBoW*) e *GloVe*. As duas dimensões escolhidas foram 100 e 300. Os valores obtidos nas métricas de avaliação destes modelos estão na Tabela 7. Os melhores resultados alcançados através das variações de BiLSTM-CRF treinadas estão em negrito. Como a métrica *F1-score* é uma combinação das outras duas (precisão e *recall*), seu valor é usado para determinar o melhor modelo, que neste caso foi *Wang2Vec - CBoW - dimensão 100*.

Tabela 7 – Comparação entre *embeddings GloVe* e *Wang2Vec*.

Modelo	**Precisão**	***Recall***	***F1-score***
GloVe - dimensão 100	87,78%	86,98%	87,38%
GloVe - dimensão 300	87,82%	86,85%	87,33%
Wang2Vec - SkipGram - dimensão 100	87,11%	**88,02%**	87,56%
Wang2Vec - SkipGram - dimensão 300	88,06%	86,39%	87,22%
Wang2Vec - CBoW - dimensão 100	**88,71%**	87,50%	**88,10%**
Wang2Vec - CBoW - dimensão 300	87,11%	85,81%	86,45%

Fonte: Arquivo pessoal.

[2] Todas as métricas gerais dos modelos nesta seção foram calculadas com *micro-average*, conforme explicado na Seção 3.9.4

6.2 *BIDIRECTIONAL ENCODER REPRESENTATION FROM TRANSFORMERS* (BERT)

As variações de redes BERT experimentadas foram *large* e *base*, com e sem a camada CRF. Os resultados são apresentados na Tabela 8. Conforme antecipado pelos resultados de outros trabalhos, a rede BERT *large* tem melhor desempenho. Redes com CRF também obtiveram valores de métricas mais altos do que as sem esta camada, ainda que a diferença entre ambas as arquiteturas seja significativamente menor do que a encontrada entre BiLSTM e BiLSTM-CRF (cerca de 7 a 8 vezes menor).

Tabela 8 – Comparação entre modelos BERT e BERT-CRF *base* e *large*.

Modelo	**Precisão**	***Recall***	***F1-score***
BERT *base*	87,72%	90,69%	89,18%
BERT-CRF *base*	89,70%	91,28%	90,48%
BERT *large*	89,07%	91,21%	90,13%
BERT-CRF *large*	**90,16%**	**91,86%**	**91,00%**

Fonte: Arquivo pessoal.

6.3 COMPARAÇÃO DE RESULTADOS

A comparação entre os melhores resultados obtidos neste projeto e os apresentados no modelo *baseline*, do trabalho de Luz de Araujo *et al.* (2018), está na Tabela 9. Vê-se que, dentre os modelos, a rede BERT-CRF, em destaque, apresentou os melhores resultados, superando o desempenho do *baseline* (em aproximadamente 4,4 pontos percentuais, se tratando do *F1-score*) e do BiLSTM-CRF (em 2,9 pontos percentuais). Como atualmente os modelos estado-da-arte de PLN, no geral, são redes BERT, o resultado dessa comparação está de acordo com o estudado e esperado.

Além de modelos BERT terem superado o *baseline*, o melhor modelo BiLSTM também fez isso em quase 1,5 ponto percentual. Isso pode ter se dado pelo uso de diferentes *embeddings* entre este trabalho e o de Luz de Araujo *et al.* (2018) e por outras alterações na rede neural, como o tamanho de cada camada.

As Tabela 10 e Tabela 11 mostram com mais detalhes as performances do modelo *baseline* e do melhor modelo obtido neste trabalho. Constata-se que o desempenho em todas as classes de entidade do *baseline* é superado pelo modelo BERT-CRF por, em média, cerca de 3 pontos percentuais na precisão, 9,5 pontos percentuais no *recall* e 6,2 pontos percentuais no *F1-score*. A melhora mais significativa foi na classe PESSOA, com 13,62 pontos percentuais de diferença no *F1-score*, seguida da classe LOCAL, com uma melhora de 12,18 pontos.

Tabela 9 – Comparação de resultados dos modelos NER.

Modelo	Precisão	*Recall*	*F1-score*
Melhor BiLSTM-CRF	88,71%	87,50%	88,10%
Melhor BERT-CRF	**90,16%**	**91,86%**	**91,00%**
Luz de Araujo *et al.* (2018)	87,98%	85,29%	86,61%

Fonte: Arquivo pessoal.

A classe com pior desempenho em ambos os modelos foi LOCAL. Isso se dá porque, segundo Luz de Araujo *et al.* (2018), entidades dessa classe são as mais raras no *dataset*, representando 0,61% e 0,28% das palavras dos conjuntos de treinamento e teste, respectivamente. Isso dificulta o modelo treinado de reconhecer essa entidade, facilmente classificando-a como PESSOA ou ORGANIZAÇÃO (por exemplo, não identificando o trecho “avenida José Faria da Rocha” como uma localização e sim como uma pessoa, “José Faria da Rocha”).

Tabela 10 – Métricas do modelo BiLSTM-CRF de Luz de Araujo *et al.* (2018).

Classe	Precisão	*Recall*	*F1-score*
LEGISLACAO	93,93%	94,18%	94,06%
PESSOA	85,58%	78,97%	82,14%
ORGANIZACAO	88,30%	82,83%	85,48%
JURISPRUDENCIA	79,29%	84,86%	81,98%
TEMPO	91,30%	87,50%	89,36%
LOCAL	69,77%	63,83%	66.67%
Total	87,98%	85,29%	86,61%

Fonte: Luz de Araujo *et al.* (2018).

Tabela 11 – Métricas do melhor modelo BERT-CRF obtido neste trabalho.

Classe	Precisão	*Recall*	*F1-score*
LEGISLACAO	94,49%	95,24%	94,86%
PESSOA	94,56%	97,00%	95,76%
ORGANIZACAO	88,38%	88,02%	88,20%
JURISPRUDENCIA	82,91%	89,19%	85,94%
TEMPO	93,68%	92,71%	93,19%
LOCAL	71,93%	87,23%	78,85%
Total	90,16%	91,86%	91,00%

Fonte: Arquivo pessoal.

Na Figura 27, há comparações entre as inferências em três trechos de textos legais feitas por ambos os modelos, o *baseline*, de Luz de Araujo *et al.* (2018), e o BERT-CRF obtido neste trabalho. As entidades encontradas por cada um deles estão destacadas em negrito e sublinhadas, com as diferentes cores representando classes distintas, conforme indicado na legenda da própria figura.

No primeiro trecho da Figura 27, vemos que ambos os modelos encontraram todas as entidades corretamente, sendo elas, na ordem:

- PESSOA: “Cezar Peluso”
- JURISPRUDENCIA: “MI 822/DF”
- LEGISLAÇÃO: “§ 6º do art. 57 da Lei nº 8.213”
- TEMPO: “24 de julho de 1991”
- LEGISLAÇÃO: “inciso II do art . 22 da Lei nº 8.212”
- TEMPO: “24 de julho de 1991”

No segundo trecho da Figura 27, o modelo *baseline* deixou de encontrar parte de um nome (“Direito”) que deveria ser inferido como PESSOA e que o modelo BERT-CRF classificou corretamente. As entidades corretas na ordem são:

- JURISPRUDENCIA: “Rcl 6873/SP”
- PESSOA: “Menezes Direito”
- TEMPO: “5/11/2008”

Por último, no terceiro trecho comparado na Figura 27, o modelo *baseline* cometeu dois erros: deixar de classificar “Agravo de Instrumento em” como parte de uma entidade JURISPRUDÊNCIA (erro este que também foi cometido pelo modelo BERT-CRF) e não inferir “EMGEPRON” como ORGANIZAÇÃO (enquanto o modelo BERT-CRF encontrou esta entidade). A lista de entidades nesse trecho é:

- JURISPRUDENCIA: “Agravo de Instrumento em Recurso de Revista n.º TST-AIRR-702/2002-070-01-40.3”
- ORGANIZAÇÃO: “EMPRESA GERENCIAL DE PROJETOS NAVAIS”
- ORGANIZAÇÃO: “EMGEPRON”
- PESSOA: “FERNANDO CHAVES”
- ORGANIZAÇÃO: “SOLUTION FIBER DO BRASIL LTDA”

Figura 27 – Comparação entre inferências do modelo *baseline*, de Luz de Araujo *et al.* (2018), e BERT-CRF obtido neste trabalho.

Modelo *baseline*	Modelo BERT-CRF
Nesse sentido , rememoro manifestação do Ministro Cezar Peluso , ao julgamento do MI 822/DF : " Dispõe o § 6º do art. 57 da Lei nº 8.213, de 24 de julho de 1991, que a aposentadoria especial será custeada pela contribuição prevista no inciso II do art . 22 da Lei nº 8.212, de 24 de julho de 1991, que, por sua vez, estabelece uma contribuição social devida pela empresa na qual trabalhadores são expostos a riscos ambientais.	Nesse sentido , rememoro manifestação do Ministro Cezar Peluso , ao julgamento do MI 822/DF : " Dispõe o § 6º do art. 57 da Lei nº 8.213, de 24 de julho de 1991, que a aposentadoria especial será custeada pela contribuição prevista no inciso II do art . 22 da Lei nº 8.212, de 24 de julho de 1991, que, por sua vez, estabelece uma contribuição social devida pela empresa na qual trabalhadores são expostos a riscos ambientais.
(Rcl 6873/SP, rel. Min. Menezes Direito, decisão monocrática publicada no DJe divulgado em 5/11/2008, págs. 111/112)	(Rcl 6873/SP, rel. Min. Menezes Direito, decisão monocrática publicada no DJe divulgado em 5/11/2008, págs. 111/112)
Vistos, relatados e discutidos estes autos de Agravo de Instrumento em Recurso de Revista n.º TST-AIRR-702/2002-070-01-40.3, em que é Agravante EMPRESA GERENCIAL DE PROJETOS NAVAIS - EMGEPRON e Agravados FERNANDO CHAVES e SOLUTION FIBER DO BRASIL LTDA.	Vistos, relatados e discutidos estes autos de Agravo de Instrumento em Recurso de Revista n.º TST-AIRR-702/2002-070-01-40.3, em que é Agravante EMPRESA GERENCIAL DE PROJETOS NAVAIS - EMGEPRON e Agravados FERNANDO CHAVES e SOLUTION FIBER DO BRASIL LTDA.

Entidades

- PESSOA
- TEMPO
- ORGANIZAÇÃO
- LOCAL
- LEGISLAÇÃO
- JURISPRUDÊNCIA

Fonte: Arquivo pessoal.

7 CONCLUSÃO

Neste trabalho foram apresentados o desenvolvimento e os resultados do projeto realizado, em que utilizaram-se técnicas de IA e PLN para realizar a tarefa de REN em textos jurídicos escritos na língua portuguesa brasileira.

Os modelos de redes neurais usados foram LSTM e BERT, este último pré-treinado no trabalho de Souza, Nogueira e Lotufo (2019). O *dataset* escolhido para o treinamento/ajuste fino foi o *LeNER-Br*, introduzido no artigo de Luz de Araujo *et al.* (2018) (cujo modelo apresentado foi usado como *baseline* de comparação para este projeto). As métricas e seus cálculos foram os mesmos da tarefa compartilhada CoNLL-2003. O melhor modelo obtido neste trabalho, uma rede BERT-CRF *large*, alcançou os seguintes valores nas métricas de avaliação no conjunto de testes: precisão de 90,16%, *recall* de 91,86% e *F1-score* de 91,00%, superando o modelo *baseline*.

O objetivo geral, de produzir um modelo treinado que encontre a citação de entidades pré-definidas em textos legais, foi atingido. Cumpriram-se também os objetivos específicos, com a pesquisa e a utilização de diferentes modelos de *deep learning* (RNN e arquitetura *transformer*), a identificação das métricas mais relevantes dentre as disponíveis para avaliar o desempenho na tarefa NER e a comparação entre os modelos obtidos neste trabalho e em outros.

Quanto à utilidade deste projeto para a empresa em que ele foi realizado, no momento a equipe local de desenvolvimento trabalha num módulo para REN que será adicionado às soluções de inteligência investigativa, onde planeja-se utilizar um ou mais modelos obtidos neste trabalho.

Para trabalhos futuros, recomenda-se expandir o *dataset* usado para treinamento com mais documentos jurídicos e de legislações, aumentando também a proporção de palavras pertencentes a classe LOCAL, a que tem menor representação no conjunto de dados. Além disso, outra possível forma de melhorar o desempenho do modelo BERT-CRF encontrado é por treinar *word embeddings* especificamente com textos do domínio legal. Uma terceira possibilidade seria utilizar um modelo pré-treinado com textos legais, como o do trabalho de Ciurlino (2021), que, juntamente com a aplicação das demais sugestões, tem potencial para alcançar um desempenho superior.

REFERÊNCIAS

ABIMDE. **Empresas Estratégicas de Defesa**. [*S.l.*: *s.n.*], 2021. Disponível em: https://abimde.org.br/pt-br/associado/empresas-estrategica-de-defesa/. Acesso em: 20 jan. 2022.

ARAÚJO, Natanael da Silva. **Reconhecimento de Entidades Nomeadas em Textos de Boletins de Ocorrências**. [*S.l.*: *s.n.*], 2019. Disponível em: http://www.repositorio.ufc.br/handle/riufc/49711. Acesso em: 20 dez. 2021.

BAHDANAU, Dzmitry; CHO, Kyunghyun; BENGIO, Yoshua. **Neural Machine Translation by Jointly Learning to Align and Translate**. [*S.l.*: *s.n.*], 2016. arXiv: 1409.0473 [cs.CL].

BARBOSA, Jardeson Leandro Nascimento; VIEIRA, João Paulo Albuquerque; DE SALES SANTOS, Roney Lira; MAGALHÃES JR., Gilvan Veras; MUNIZ, Mariana dos Santos; MOURA, Raimundo Santos. Introdução ao Processamento de Linguagem Natural usando Python. *In*: p. 336–360.

CASTRO, Robinson Santos. **Extração de Informação**: conceitos, plataformas e sistemas. 2013. Universidade Federal do Maranhão. Disponível em: https://rosario.ufma.br/jspui/bitstream/123456789/3272/1/ROBINSON-CASTRO.pdf. Acesso em: 30 nov. 2021.

CERON, Rodrigo. **A Inteligência Artificial hoje: dados, treinamento e inferência**. [*S.l.*: *s.n.*], 2020. Disponível em: https://www.ibm.com/blogs/systems/br-pt/2020/01/a-inteligencia-artificial-hoje-dados-treinamento-e-inferencia/. Acesso em: 15 jan. 2022.

CHAVES, Letícia Saraiva. **Utilizando um modelo transformer no processo de identificação de entidades nomeadas em textos criminais**. 2021. Universidade Federal do Ceará. Disponível em: http://www.repositorio.ufc.br/handle/riufc/61224. Acesso em: 4 jan. 2022.

CHO, Kyunghyun; VAN MERRIËNBOER, Bart; BAHDANAU, Dzmitry; BENGIO, Yoshua. On the properties of neural machine translation: Encoder-decoder approaches. **arXiv preprint arXiv:1409.1259**, 2014a.

CHO, Kyunghyun; VAN MERRIËNBOER, Bart; GULCEHRE, Caglar; BAHDANAU, Dzmitry; BOUGARES, Fethi; SCHWENK, Holger; BENGIO, Yoshua. Learning phrase representations using RNN encoder-decoder for statistical machine translation. **arXiv preprint arXiv:1406.1078**, 2014b.

CHOWDHURY, Gobinda G. Natural language processing. **Annual review of information science and technology**, Wiley Online Library, v. 37, n. 1, p. 51–89, 2003.

CIURLINO, Victor Hugo. **BertBR: A Pretrained Language Model for Law Texts**. [*S.l.*: *s.n.*], 2021. Disponível em: `https://bdm.unb.br/handle/10483/27824`. Acesso em: 15 jan. 2022.

DE SOUZA, João Vitor Andrioli; GUMIEL, Yohan Bonescki; SILVA E OLIVEIRA, Lucas Emanuel; MORO, Claudia Maria Cabral. Named Entity Recognition for Clinical Portuguese Corpus with Conditional Random Fields and Semantic Groups, 2019. Disponível em: `https://sol.sbc.org.br/index.php/sbcas/article/download/6269/6167`. Acesso em: 30 nov. 2021.

DEVLIN, Jacob; CHANG, Ming-Wei; LEE, Kenton; TOUTANOVA, Kristina. Bert: Pre-training of deep bidirectional transformers for language understanding. **arXiv preprint arXiv:1810.04805**, 2018.

DÍGITRO. **Inteligência Investigativa e Estratégica**. [*S.l.*]: Portal Dígitro, 2017. Disponível em: `http://portaldigitro.com.br/pt/index.php/inteligencia-digitro/inteligencia`. Acesso em: 11 mar. 2022.

DOS SANTOS NETO, Joaquim Francisco. **Reconhecimento de Entidades Nomeadas para o Português Usando Redes Neurais**. [*S.l.*: *s.n.*], 2019. Disponível em: `https://repositorio.pucrs.br/dspace/bitstream/10923/16456/1/000497219-Texto%5C%2Bcompleto-0.pdf`. Acesso em: 20 dez. 2021.

EBRAHIMI, Abteen; KANN, Katharina. **How to Adapt Your Pretrained Multilingual Model to 1600 Languages**. [*S.l.*: *s.n.*], 2021. Disponível em: `https://hdl.handle.net/10923/14040`. Acesso em: 4 jan. 2022.

FONSECA, Camilla. **Word Embedding: fazendo o computador entender o significado das palavras**. [*S.l.*]: Medium, 2021. Disponível em:

https://medium.com/turing-talks/word-embedding-fazendo-o-computador-entender-o-significado-das-palavras-92fe22745057. Acesso em: 20 jan. 2022.

FONSECA, Evandro Brasil; CHIELE, Gabriel; VANIM, Aline; VIEIRA, Renata. Reconhecimento de Entidades Nomeadas para o Português Usando o OpenNLP, 2015. Disponível em: https://hdl.handle.net/10923/14040. Acesso em: 30 nov. 2021.

FUKUSHIMA, Kunihiko. Cognitron: A self-organizing multilayered neural network. **Biological cybernetics**, Springer, v. 20, n. 3, p. 121–136, 1975.

GOLDBERG, Yoav. Neural network methods for natural language processing. **Synthesis lectures on human language technologies**, Morgan & Claypool Publishers, v. 10, n. 1, p. 1–309, 2017.

HARTMANN, Nathan; FONSECA, Erick R.; SHULBY, Christopher; TREVISO, Marcos Vinıcius; RODRIGUES, Jéssica S.; ALUISIO, Sandra M. Portuguese Word Embeddings: Evaluating on Word Analogies and Natural Language Tasks. **CoRR**, abs/1708.06025, 2017. arXiv: 1708.06025. Disponível em: http://arxiv.org/abs/1708.06025.

HOCHREITER, Sepp; SCHMIDHUBER, Jürgen. Long short-term memory. **Neural computation**, MIT Press, v. 9, n. 8, p. 1735–1780, 1997.

IBM-CLOUD-EDUCATION. **Recurrent Neural Networks**. [*S.l.*: *s.n.*], 2020. Disponível em: https://www.ibm.com/cloud/learn/recurrent-neural-networks. Acesso em: 18 jan. 2022.

JOHNS, Ray. **PyTorch vs TensorFlow for Your Python Deep Learning Project**. [*S.l.*]: Real Python, 2020. Disponível em: https://realpython.com/pytorch-vs-tensorflow/#who-uses-pytorch. Acesso em: 12 jan. 2022.

KORSTANJE, Joos. **The F1 score**. [*S.l.*]: Towards Data Science, 2021. Disponível em: https://towardsdatascience.com/the-f1-score-bec2bbc38aa6. Acesso em: 7 jan. 2022.

KUMAR, Ela. **Natural language processing**. [*S.l.*]: IK International Pvt Ltd, 2011.

LAFFERTY, John; MCCALLUM, Andrew; PEREIRA, Fernando CN. Conditional random fields: Probabilistic models for segmenting and labeling sequence data, 2001.

LIDDY, Elizabeth D. Natural language processing, 2001. Disponível em: https://surface.syr.edu/cgi/viewcontent.cgi?referer=https://scholar.google.com.br/&httpsredir=1&article=1019&context=cnlp. Acesso em: 23 dez. 2021.

LINGUATECA. **HAREM: Reconhecimento de entidades mencionadas em português**. [*S.l.*: *s.n.*], 2013. Disponível em: https://www.linguateca.pt/HAREM/. Acesso em: 15 jan. 2022.

LUASHCHUK, Andrew. **Why I Think Python is Perfect for Machine Learning and Artificial Intelligence**. [*S.l.*]: Towards Data Science, 2019. Disponível em: https://towardsdatascience.com/8-reasons-why-python-is-good-for-artificial-intelligence-and-machine-learning-4a23f6bed2e6. Acesso em: 20 dez. 2021.

LUZ DE ARAUJO, Pedro Henrique. **lener-br**. [*S.l.*]: GitHub, 2020. Disponível em: https://github.com/peluz/lener-br. Acesso em: 15 dez. 2021.

LUZ DE ARAUJO, Pedro Henrique; DE CAMPOS, Teófilo E; DE OLIVEIRA, Renato RR; STAUFFER, Matheus; COUTO, Samuel; BERMEJO, Paulo. LeNER-Br: A Dataset for Named Entity Recognition in Brazilian Legal Text. *In*: SPRINGER. INTERNATIONAL Conference on Computational Processing of the Portuguese Language. [*S.l.*: *s.n.*], 2018. P. 313–323.

MALIK, Farhad. **Why Should We Use NumPy?** [*S.l.*]: Medium, 2019. Disponível em: https://medium.com/fintechexplained/why-should-we-use-numpy-c14a4fb03ee9. Acesso em: 10 jan. 2022.

MARKOWITZ, Dale. **Transformers, Explained: Understand the Model Behind GPT-3, BERT, and T5**. [*S.l.*]: Google Cloud Tech, 2021. Disponível em: https://www.youtube.com/watch?v=SZorAJ4I-sA. Acesso em: 13 jan. 2022.

MARSHALL, Christopher. **What is named entity recognition (NER) and how can I use it?** [*S.l.*]: Medium - super.AI, 2019. Disponível em: https://medium.com/mysuperai/what-is-named-entity-recognition-ner-and-how-can-i-use-it-2b68cf6f545d. Acesso em: 7 jan. 2022.

MAVRIDIS, Themis. **Named Entity Classification**. [*S.l.*]: Medium - Booking.datascience, 2016. Disponível em: https://booking.ai/named-entity-classification-d14d857cb0d5. Acesso em: 7 jan. 2022.

MCCULLOCH, Warren S; PITTS, Walter. A logical calculus of the ideas immanent in nervous activity. **The bulletin of mathematical biophysics**, Springer, v. 5, n. 4, p. 115–133, 1943.

MIDDLETON, Michael. **Deep Learning vs. Machine Learning — What's the Difference?** [*S.l.*]: Flatiron School Blog, 2021. Disponível em: https://flatironschool.com/blog/deep-learning-vs-machine-learning. Acesso em: 6 jan. 2022.

NADEAU, David; SEKINE, Satoshi. A Survey of Named Entity Recognition and Classification. **Lingvisticae Investigationes**, v. 30, ago. 2007. DOI: 10.1075/li.30.1.03nad.

NETO, Octavio David. Psicóloga estuda os mecanismos da atenção. **Boletim AUN**, Agência Universitária de Notícias - USP, n. 15, 2002.

NEUDECKER, Clemens. **Named Entity Recognition for digitised newspapers**. [*S.l.*]: Europeana Newspapers, 2014. Disponível em: http://www.europeana-newspapers.eu/named-entity-recognition-for-digitised-newspapers/. Acesso em: 6 jan. 2022.

NEURALMIND. **Portuguese-BERT**. [*S.l.*]: GitHub, 2019. Disponível em: https://github.com/neuralmind-ai/portuguese-bert. Acesso em: 5 dez. 2021.

NILC. **Repositório de Word Embeddings do NILC**. [*S.l.*: *s.n.*], 2017. Disponível em: http://www.nilc.icmc.usp.br/embeddings. Acesso em: 21 dez. 2021.

O'CONNOR, Ryan. **PyTorch vs TensorFlow in 2022**. [*S.l.*]: AssemblyAI, 2021. Disponível em: https://www.assemblyai.com/blog/pytorch-vs-tensorflow-in-2022/. Acesso em: 21 dez. 2021.

OLIVEIRA, Bárbara Stéphanie Neves. **Aprendizado Profundo para Reconhecimento de Entidades Nomeadas em Narrativas de Roubos**. 2020.

Universidade Federal do Ceará. Disponível em: http://www.repositorio.ufc.br/handle/riufc/55717. Acesso em: 4 jan. 2022.

PALMER, David D. Text preprocessing. *In*: HANDBOOK of natural language processing. 2. ed. [*S.l.*]: CRC Press, 2010. cap. 2, p. 9–30.

[*S.l.*]. **Spark NLP: How Roche automates knowledge extraction from pathology and radiology reports**. [*S.l.*: *s.n.*], 2019. Disponível em: https://conferences.oreilly.com/strata/strata-ca-2019/public/schedule/detail/72568.html. Acesso em: 6 jan. 2022.

ROSENBLATT, Frank. The perceptron: a probabilistic model for information storage and organization in the brain. **Psychological review**, American Psychological Association, v. 65, n. 6, p. 386, 1958.

SAS-INSIGHTS. **Machine Learning - O que é e qual sua importância?** [*S.l.*]: Statistical Analysis System, 2021. Disponível em: https://www.sas.com/pt_br/insights/analytics/machine-learning.html. Acesso em: 7 jan. 2022.

SCHUSTER, Mike; PALIWAL, Kuldip K. Bidirectional recurrent neural networks. **IEEE transactions on Signal Processing**, Ieee, v. 45, n. 11, p. 2673–2681, 1997.

SOARES, Pablo; SILVA, José da. Aplicação de Redes Neurais Artificiais em Conjunto com o Método Vetorial da Propagação de Feixes na Análise de um Acoplador Direcional Baseado em Fibra Ótica. **Revista Brasileira de Computação Aplicada**, v. 3, dez. 2011. DOI: 10.5335/rbca.2013.1803.

SOUZA, Fábio; NOGUEIRA, Rodrigo; LOTUFO, Roberto. Portuguese Named Entity Recognition using BERT-CRF. **arXiv preprint arXiv:1909.10649**, 2019. Disponível em: http://arxiv.org/abs/1909.10649.

STIEGLER, Arnaud. **Exploring Conditional Random Fields for NLP Applications**. [*S.l.*]: Hyperscience, 2021. Disponível em: https://hyperscience.com/tech-blog/exploring-crfs-for-nlp-applications/. Acesso em: 28 jan. 2022.

SUTCU, Cem Sefa; AYTEKIN, Cigdem. AN EXAMPLE OF PRAGMATIC ANALYSIS IN NATURAL LANGUAGE PROCESSING: SENTIMENTAL ANALYSIS OF MOVIE REVIEWS. *In*: p. 61–74. DOI: 10.7456/ctc_2019_05.

TJONG KIM SANG, Erik F.; DE MEULDER, Fien. Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition. *In*: PROCEEDINGS of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003. [*S.l.*: *s.n.*], 2003. P. 142–147. Disponível em: https://aclanthology.org/W03-0419. Acesso em: 7 jan. 2022.

VASWANI, Ashish; SHAZEER, Noam; PARMAR, Niki; USZKOREIT, Jakob; JONES, Llion; GOMEZ, Aidan N.; KAISER, Lukasz; POLOSUKHIN, Illia. **Attention Is All You Need**. [*S.l.*: *s.n.*], 2017. arXiv: 1706.03762 [cs.CL].

WAGNER FILHO, Jorge A.; WILKENS, Rodrigo; IDIART, Marco; VILLAVICENCIO, Aline. The brWaC Corpus: A New Open Resource for Brazilian Portuguese. *In*: PROCEEDINGS of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018). Miyazaki, Japan: European Language Resources Association (ELRA), mai. 2018. Disponível em: https://aclanthology.org/L18-1686.

WANG, Jiawei. **pytorch-NER**. [*S.l.*]: GitHub, 2020. Disponível em: https://github.com/cswangjiawei/pytorch-NER. Acesso em: 15 dez. 2021.

WILLIAN SOUSA, Antonio; FABRO, Marcos. Iudicium Textum Dataset Uma Base de Textos Jurídicos para NLP. *In*.

WITTEN, Ian H.; FRANK, Eibe; HALL, Mark A. **Data Mining: Practical Machine Learning Tools and Techniques**. 3. ed. [*S.l.*]: Elsevier, 2011.

ZHANG, Aston; LIPTON, Zachary C; LI, Mu; SMOLA, Alexander J. Dive into deep learning. **arXiv preprint arXiv:2106.11342**, 2021.

ANEXO A – *CORPORA* UTILIZADOS NOS TREINAMENTOS DE *EMBEDDINGS* DO NILC

O Quadro 2 mostra os 17 diferentes *corpora* usados nos treinamentos de *embeddings* realizados pelo NILC, junto com seu gênero e uma breve descrição. Ele é traduzido e adaptado do NILC (2017).

Quadro 2 – Fontes de *corpora* coletados para formar o grande *corpus* de treinamento.

Corpus	**Gênero**	**Descrição**
LX-Corpus	Vários gêneros	Grande coleção de textos de 19 fontes. A maioria é escrita em Português Europeu.
Wikipedia	Enciclopédia	*Dump* da Wikipedia de 20/10/2016.
GoogleNews	Informativo	Notícias extraídas do serviço GoogleNews.
SubIMDB-PT	Linguagem falada	Legendas extraídas do website IMDb.
G1	Informativo	Notícias extraídas do portal de notícias G1 entre 2014 e 2015.
PLN-Br	Informativo	Grande *corpus* do projeto PLN-Br com textos amostrados de 1994 a 2005.
Trabalhos literários em domínio público	Prosa	Coleção de 138.268 trabalhos literários do website do Domínio Público.
Lacio-web	Vários gêneros	Textos de vários gêneros, e.g. literário e subdivisões (prosa, poesia e dramaturgia), informativo, científico, legal, didático técnico, etc.
E-books em português	Prosa	Coleção de livros clássicos de ficção escritos em Português Brasileiro extraídos do website Literatura Brasileira.
Mundo Estranho	Informativo	Textos extraídos da revista Mundo Estranho.
CHC	Informativo	Textos extraídos do website Ciência Hoje das Crianças (CHC).
FAPESP	Comunicação científica	Textos brasileiros de divulgação científica da revista Pesquisa FAPESP.
Livros-texto	Didático	Textos para crianças do 3º ao 7º ano do ensino fundamental.
Folhinha	Informativo	Notícias escritas para crianças, extraídas em 2015 da Folhinha do jornal Folha de São Paulo.
Subcorpus NILC	Informativo	Textos escritos para crianças entre o 3º e o 4º ano do ensino fundamental.
Para seus filhos ler	Informativo	Notícias escritas para crianças, do jornal Zero Hora.
SARESP	Didático	Questões escritas de matemática, ciências humanas, ciências naturais e escrita de dissertações/redações para avaliar estudantes.

Fonte: NILC (2017).